UNIVERSIDAD NACIONAL DE LA PLATA
FACULTAD DE INFORMÁTICA

Desarrollo basado en conocimiento siguiendo prácticas ágiles

Trabajo de tesis
presentado para obtener el grado de
Magíster en Ingeniería de Software

Julio de 2015

Autora: Esp. Loraine Gimson
Director: Mg. Gustavo Daniel Gil
Co-Director: Dr. Gustavo Rossi

A mi madre, Graciela

AGRADECIMIENTOS

Este trabajo sólo fue posible gracias al aporte, directo o indirecto de muchas personas. A "todos" ellos quiero agradecer, especialmente:

- A Dios por permitirme culminar esta etapa.
- A mi Director por su tiempo dedicado, constante guía y aliento para seguir adelante.
- A mi co-Director por apoyar este trabajo y creer que era posible.

INDICE

INDICE DE FIGURAS

INDICE DE TABLAS

1. INTRODUCCIÓN

El presente trabajo tiene como objetivo poder elaborar una propuesta de prácticas ágiles para el desarrollo basado en Conocimiento. No se plantea realizar trabajo experimental, sino un análisis de las metodologías o procesos de desarrollo utilizadas en empresas salteñas (públicas y privadas) que trabajan con desarrollo basado en Conocimiento para posteriormente elaborar una propuesta de prácticas ágiles para desarrollo de software basado en conocimiento.

1.1 MOTIVACIÓN

En este trabajo se combinan dos conceptos novedosos, fundamentalmente para Salta Capital, respecto a la forma de encarar un desarrollo de software, tanto en el ámbito público como privado: las metodologías ágiles y el desarrollo basado en conocimiento (o también llamadas bases de datos del conocimiento). Si bien en la actualidad es más frecuente escuchar hablar de metodologías ágiles, no es común encontrar en la ciudad de Salta una empresa pública o privada que aplique concretamente alguna de ellas. En esta ciudad recién se está comenzando a tratar de incorporar algunas de las prácticas que estas metodologías proponen, y capacitar al personal en estas metodologías (mayormente en SCRUM). Además existen varias empresas públicas y privadas que están trabajando con bases de datos del conocimiento sin una metodología de desarrollo bien definida, tratando de definir un proceso de desarrollo poco burocrático que podría verse enriquecido si se incorporar un marco de trabajo como el que proponen las metodologías ágiles.

Por todo lo antes expuesto, se considera importante poder investigar diferentes propuestas que plantean autores de metodologías ágiles y profundizar en el conocimiento de las bases de datos del conocimiento. También es importante poder analizar el uso de bases del conocimiento en la ciudad de Salta, por parte de empresas locales salteñas públicas y privadas, para poder armar un mapa de la situación actual de cómo se está trabajando y ver si es posible sugerir una serie de prácticas ágiles que acompañen este tipo de desarrollos. La virtud más importante de lo que se pudiera obtener es la posibilidad de combinar dos metodologías novedosas y que a simple vista no son combinables.

1.2 OBJETIVO GENERAL

Este trabajo busca poder elaborar una propuesta de prácticas ágiles para el desarrollo basado en Conocimiento, que pueda enriquecer la forma de trabajar con esta última metodología. Para ello se plantea en primer lugar, realizar una investigación bibliográfica tendiente a exponer los fundamentos de diferentes metodologías ágiles propuestas para el desarrollo de sistemas y también realizar paralelamente una recopilación bibliográfica sobre el desarrollo basado en conocimiento. Posteriormente, realizar un relevamiento en Salta Capital tanto en empresas públicas como privadas en sus procesos de desarrollo trabajando con desarrollo basado en conocimiento y realizar un análisis de campo de la información recolectada para elaborar una propuesta de prácticas ágiles para estas organizaciones.

1.1.1 Metodologías ágiles

Hace casi dos décadas que se comenzó a buscar una alternativa a las metodologías formales o tradicionales que estaban sobrecargadas de técnicas y herramientas y que se consideraban excesivamente “pesadas” y rígidas por su carácter normativo y fuerte dependencia de planificaciones detalladas previas al desarrollo

Las metodologías ágiles conllevan una filosofía de desarrollo de software liviana, debido a que hace uso de modelos ágiles. Se considera que un modelo es ágil o liviano cuando se emplea para su construcción una herramienta o técnica sencilla, que apunta a desarrollar un modelo aceptablemente bueno y suficiente en lugar de un modelo perfecto y complejo.

Existen actualmente una serie de metodologías que responden a las características de las metodologías ágiles y cada vez están teniendo más adeptos. Aunque los creadores e impulsores de las metodologías ágiles más populares han suscrito el manifiesto ágil y coinciden

con sus postulados y principios, cada metodología ágil tiene características propias y hace hincapié en algunos aspectos más específicos.

1.1.2 Bases de Datos del Conocimiento

Actualmente se pretende poder desarrollar software en el menor tiempo posible y con el menor costo. Para tratar de reducir el tiempo de programación, la solución no está relacionada tanto en mejorar más todavía los lenguajes de programación sino en la programación en sí. En los desarrollos de sistemas tradicionales se desarrolla y se realiza el mantenimiento con programación manual. Si se "describe" en vez de "programar", se pueden maximizar las descripciones declarativas y minimizar las especificaciones procedurales, haciendo desarrollo basado en conocimiento y no en programación. Esta pretensión constituye un cambio esencial de paradigma e implica un choque cultural.

Los sistemas basados en conocimiento (SBC) tratan de mantener una gran cantidad de conocimiento y aportan mecanismos para manejarlo. La representación es declarativa: se separa el conocimiento del dominio de los mecanismos de deducción. Esto permite reutilizar tanto la Base del Conocimiento como los mecanismos de razonamiento [Alonso 2004]. En este trabajo de tesis, el paradigma de desarrollo basado en conocimiento al que se hace referencia, consta de dos partes principales: la Base de conocimiento y el motor de inferencia o proceso de razonamiento, donde el objetivo es obtener por inferencia la base de datos y los programas para manejar los objetos descritos por el usuario. Pero, no busca emular capacidades de un dominio experto. Este paradigma está orientado más a los sistemas de gestión. Permiten partir de la descripción de las visiones de todos los usuarios del sistema a desarrollar y genera la base de datos y los programas necesarios para satisfacer dichas visiones.

La Base del Conocimiento inicialmente tiene asociado un conjunto de mecanismos de inferencia y contiene reglas generales que son independientes de cualquier aplicación particular. Al describir la realidad del usuario objeto, se almacenan las descripciones en el Modelo Externo. El sistema, automáticamente, captura todo el conocimiento contenido en el Modelo Externo y lo sistematiza, agregándolo también a la Base del conocimiento. Adicionalmente, sobre el conocimiento anterior, el sistema infiere lógicamente un conjunto de resultados que ayudan a mejorar la eficiencia de las inferencias posteriores. En este tipo de desarrollo el foco está en ocuparse únicamente del Modelo Externo (el “qué”) y abstenerse de tratar la Base del Conocimiento, que lo contiene y lo mantiene, (y que forma parte del “cómo”).

1.3 PUBLICACIONES Y TRABAJOS RELACIONADOS

Los siguientes artículos publicados y trabajos de investigación son algunos de los resultados obtenidos respecto al tema de esta tesis:

> “METODOLOGÍAS ÁGILES Y DESARROLLO BASADO EN CONOCIMIENTO” – XIII Workshops de Investigadores en Ciencias de la Computación (WICC 2011). RedUNCI Universidad Nacional de Rosario (UNR) - Rosario – Santa Fe - 5 y 6 de mayo de 2011, Rosario**.** Gil, G., Gimson, L., Ramírez, J., Aballay, P., Ortega, V., Torres, M.
>
> “METODOLOGÍAS ÁGILES Y DESARROLLO BASADO EN EL CONOCIMIENTO, EVALUACIÓN CUANTITATIVA DE F/OSS PARA LA REUTILIZACIÓN, NORMAS ISO Y SU APLICACIÓN EN CENTROS EDUCATIVOS” - XIV Workshop de Investigadores en Ciencias de la Computación (WICC 2012) Universidad Nacional de Misiones (UNM) - Posadas – Misiones – 26 y 27 de abril de 2012, Posadas. Gil, G., Arias Figueroa, D., Gimson,L., Ramirez, J., Silvera,J.
>
> Director en el Trabajo de Investigación “METODOLOGÍAS ÁGILES Y DESARROLLO BASADO EN CONOCIMIENTO”. Centro de Investigación y Desarrollo de Informática Aplicada de la UNSa. Período: 01/01/2010 al 31/12/2012.
>
> “DESARROLLO BASADO EN CONOCIMIENTO SIGUIENDO PRÁCTICAS ÁGILES” – XVI Workshops de Investigadores en Ciencias de la Computación (WICC 2014). RedUNCI Universidad Nacional de Tierra del Fuego (UNTDF) – Ushuaia – Tierra del Fuego – 7 y 8 de mayo de 2014, Ushuaia**.** Gil, G., Gimson, L. Silvera, J.

Director en el Trabajo de Investigación “DESARROLLO BASADO EN CONOCIMIENTO SIGUIENDO PRÁCTICAS ÁGILES”. Centro de Investigación y Desarrollo de Informática Aplicada de la UNSa. Período: 01/01/2013 a 31/12/2014.

“DESARROLLO BASADO EN CONOCIMIENTO SIGUIENDO PRÁCTICAS ÁGILES” – XVII Workshops de Investigadores en Ciencias de la Computación (WICC 2015). RedUNCI Universidad Nacional de Salta (UNSa) – Salta Capital – Salta – 16 y 17 de abril de 2015, Salta**.** Gimson, L., Gil, G., Arias Figueroa, D.

Actualmente también se está dirigiendo dos Tesinas de grado para la Licenciatura en Análisis de Sistemas en la Facultad de Ciencias Exactas de la UNSa sobre desarrollo de aplicaciones específicas aplicando Desarrollo basado en Conocimiento siguiendo prácticas ágiles.

1.4 CONTENIDOS

El trabajo de tesis se divide en cinco partes principales: Introducción, Metodologías Ágiles, Desarrollo Basado en Conocimiento, Propuesta de Prácticas Ágiles y Conclusiones

A continuación se describen los cuatro capítulos restantes:

Metodologías Ágiles presenta un marco teórico de las Metodologías ágiles en general, su razón de ser, el manifiesto ágil para, posteriormente, adentrarse en diferentes metodologías investigadas donde se exponen sus fundamentos.

Desarrollo Basado en Conocimiento contiene un marco teórico de esta metodología y una descripción de la herramienta actual que soporta este tipo de desarrollo y una futura herramienta que está siendo desarrollada.

Propuesta de Prácticas Ágiles presenta en primer lugar un estudio de campo según datos recopilados de encuestas y entrevistas realizadas a organizaciones del sector público como privado que trabajan con Desarrollo Basado en Conocimiento dentro del ámbito de Salta Capital y se realiza un análisis de campo de la situación. Luego se proponen ciertas prácticas ágiles para el desarrollo basado en conocimiento, teniendo en cuenta el estudio de campo obtenido.

Conclusiones expone las reflexiones y conclusiones elaboradas a partir de lo aprendido e investigado y se plantean líneas futuras de investigación.

2 METODOLOGÍAS ÁGILES

2.1 GENERALIDADES SOBRE LAS METODOLOGÍAS ÁGILES

"Desde hace unos pocos años ha habido un interés creciente en las metodologías ágiles (léase "livianas"). Caracterizadas alternativamente como antídoto a la burocracia, han suscitado interés en el panorama del software." Martin Fowler [Fowler 2003a]

2.1.1 Inicios de las Metodologías ágiles

Tal como lo menciona Martin Fowler en su artículo *La nueva metodología*, podría decirse que el desarrollo de software es una actividad caótica, frecuentemente caracterizada por la frase "codifica y corrige". El software se escribe con un mínimo plan subyacente, y el diseño del sistema se arma con muchas decisiones a corto plazo. Esto realmente funciona muy bien si el sistema es pequeño, pero conforme el sistema crece llega a ser cada vez más difícil agregar nuevos aspectos al mismo. Además los errores llegan a ser cada vez más frecuentes y más difíciles de corregir.

Se ha sobrevivido con este estilo de desarrollo por mucho tiempo, pero también surgió una alternativa desde hace muchos años: emplear una Metodología de Desarrollo Ingenieril. Las metodologías imponen un proceso disciplinado sobre el desarrollo de software con el fin de hacerlo más predecible y eficiente. Lo hacen, desarrollando un proceso detallado con un fuerte énfasis en planificar inspirado por otras disciplinas de la ingeniería. Dichos procesos definen artefactos de desarrollo, documentación a producir, herramientas y notaciones a ser utilizadas, y actividades a realizar y su orden de ejecución, entre otras definiciones. Como resultado, los procesos generan gran cantidad de documentación con el objetivo de facilitar compartir el conocimiento entre los integrantes del equipo de trabajo. Si bien existen varios procesos de desarrollo – Proceso Unificado [Jacobson 1999], Proceso V [IABG web], etc., la mayoría de estos procesos se derivan del Modelo de Cascada propuesto por Boehm [Bohem 1976]. Las metodologías ingenieriles, denominadas *tradicionales*, han estado presentes durante mucho tiempo y han demostrado ser efectivas, particularmente en lo que respecta a la administración de recursos a utilizar y a la planificación de los tiempos de desarrollo, en proyectos de gran tamaño con requerimientos estables. La crítica más frecuente a estas metodologías es que son burocráticas. Fowler afirma que hay tanto que hacer para seguir la metodología que el ritmo entero del desarrollo se retarda.

Sin embargo, debido a entornos comerciales más competitivos, conducidos por la rapidez para producir y entregar servicios [Ridderstrale 2000], el enfoque propuesto por estas metodologías tradicionales o burocráticas no resulta el más adecuado. Usualmente, estos nuevos entornos se caracterizan por el desarrollo de proyectos donde los requerimientos del sistema son desconocidos, inestables o muy cambiantes, los tiempos de desarrollo se deben reducir drásticamente, y al mismo tiempo, se espera la producción de un producto de alta calidad, que a su vez garantice mínimo riesgo ante la necesidad de introducir cambios a los requerimientos.

Como una reacción a estas metodologías tradicionales, "pesadas", complejas, muy estructuradas y estrictas y sobrecargadas de técnicas y herramientas, ha surgido un nuevo grupo de metodologías desde los años 90. Durante algún tiempo se conocían como las metodologías ligeras, pero el término aceptado y definido por la Agile Alliance es *metodologías ágiles*. Estas metodologías están especialmente indicadas para productos cuya definición detallada es difícil de obtener desde el comienzo, o que si se definiera, tendría menor valor que si el producto se construyera con una retro-alimentación continua durante el proceso de desarrollo. Manteniendo prácticas esenciales de las metodologías tradicionales, las metodologías ágiles se centran en otras dimensiones del proyecto, como por ejemplo: intentan trabajar con un mínimo de documentación necesaria, reemplazándola por la comunicación directa y cara a cara entre todos los integrantes del equipo; la colaboración activa de los usuarios durante todas las etapas del proceso de desarrollo; el desarrollo incremental del software con iteraciones muy cortas y que entregan una solución a medida; y la reducción drástica de los tiempos de desarrollo pero a su vez manteniendo una alta calidad del producto.

Buscan un justo medio entre ningún proceso y demasiado proceso, proporcionando simplemente suficiente proceso para que, como dice Fowler, el esfuerzo valga la pena.

Por todo lo mencionado puede verse fácilmente que este nuevo concepto dentro de la ingeniería de software, denominado modelado ágil de sistemas o metodologías ágiles, se trata de una filosofía de desarrollo liviana, debido a que hace uso de modelos ágiles. Logró adeptos, inició una corriente filosófica dentro del concierto de metodologías para el desarrollo de sistemas de software, conformó un corpus documental de sus principios y de sus prácticas y obtuvo éxitos resonantes en proyectos de envergadura. [Giro 2002] [Ferrer 2003] [Canós 2003]

En el año 2001, miembros prominentes de la comunidad se reunieron en Snowbird, Utah, y adoptaron el nombre de "metodologías ágiles", definiendo además el manifiesto ágil. Poco después, algunas de estas personas formaron la Agile Alliance, una organización sin fines de lucro que promueve el desarrollo ágil de aplicaciones. Muchos métodos ágiles fueron creados antes del 2000. Entre los más notables se encuentran: Scrum (1986), Crystal Clear (cristal transparente), Programación Extrema o XP (1996), Desarrollo de Software Adaptativo, Dynamic Systems Development Method o DSDM (Método de desarrollo de sistemas dinámicos) (1995).

2.1.2 El manifiesto ágil

El Manifiesto Ágil es el documento que define los principios rectores de las metodologías ágiles de desarrollo de software. Este importante documento fue creado por un grupo de académicos y expertos de la industria del software reunido en Snowbird, Utha, Estados Unidos en febrero de 2001, a través de una iniciativa de Kent Beck. Este grupo estaba conformado por 17 representantes de procesos de desarrollo livianos. Las 17 personas fueron Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Hihsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Stephen J. Mellor, Ken Schwaber, Jeff Sutherland, Dave “Pragmatic” Thomas. Otras personas fueron invitadas pero estas 17 asistieron.

La razón de la reunión fue ver si había algo en común entre las diferentes metodologías livianas de ese momento: Adaptative Software Development, XP, Scrum, Crystal, FDD, DADM y “Pragmatic Programming”. Tal como lo aclara Cockburn [Cockburn 2007], ninguno estaba interesado en combinar las prácticas para crear una Metodología Liviana Unificada (Unified Light Methodology – ULM). El objetivo de la reunión era discutir los valores y principios que facilitarían desarrollar software más rápidamente y respondiendo a los cambios que surjan a lo largo del proyecto. La idea era ofrecer una alternativa a los procesos de desarrollo tradicionales, caracterizados por su rigidez y dirigidos por la documentación que se genera en cada etapa.

En esta reunión se decidió adoptar el término *Ágil* y como resultado se obtuvo un documento conocido como el Manifiesto Ágil [ManifiestoAgil web]. El Manifiesto Ágil incluye cuatro postulados y una serie de principios asociados. Tal como hace observar Alistair CockBurn [Cockburn 2007], el propio manifiesto comienza diciendo que se están sacando a la luz mejores formas de desarrollar software, no las están inventando. Además, que el resultado se obtuvo de la experiencia directa y la reflexión sobre la experiencia y que se reconoce valor a las herramientas, procesos, documentación, contratos y planes, si bien ponen mayor énfasis en otros valores. Por esto es que las metodologías ágiles cambian significativamente algunos de los énfasis de las metodologías tradicionales, lo que puede apreciarse en los postulados del manifiesto ágil.

Sus postulados son:

1) *Valorar al individuo y a las interacciones del equipo de desarrollo por encima del proceso y las herramientas*. Este postulado enuncia que las personas son componentes primordiales en cualquier desarrollo [Cockburn 1999]. Tres premisas sustentan este postulado:

a) los integrantes del equipo son el factor principal de éxito;

b) es más importante construir el equipo de trabajo que construir el entorno; y

c) es mejor crear el equipo y que éste configure el entorno en base a sus necesidades. [MendesCalo 2010]

Se presta atención a las personas en el equipo como opuesto a los roles en un diagrama de proceso, donde las personas son reemplazables. Y por otro lado se presta atención a la interacción de los individuos. Es preferible un proceso no documentado con buenas interacciones que un proceso bien documentado con interacciones hostiles [Cockburn 2007]

2) *Valorar el desarrollo de software que funcione por sobre una documentación exhaustiva*. El postulado se basa en la premisa que los documentos no pueden sustituir ni ofrecer el valor agregado que se logra con la comunicación directa entre las personas a través de la interacción con los prototipos. Se debe reducir al mínimo indispensable el uso de documentación que genera trabajo y que no aporta un valor directo al producto [MendesCalo 2010].

El sistema funcionando es lo único que muestra lo que el equipo *ha desarrollado*. Correr código es honestidad implacable. La documentación la utiliza el equipo para reflexionar, como pistas de lo que se debería realizar. El acto completo de reunir requerimientos, diseñar, codificar, probar el software revela información del equipo, el proceso, y la naturaleza del problema a resolver. Esto, junto con la posibilidad de correr el resultado final, provee la única medida confiable de la velocidad del equipo, y un vistazo de lo que el equipo debería estar desarrollando. Los documentos pueden ser útiles pero deben usarse en la “medida justa” o “a penas suficiente” [Cockburn 2007]

3) *Valorar la colaboración con el cliente por sobre la negociación contractual*. En el desarrollo ágil el cliente se integra y colabora con el equipo de trabajo. Se asume que el contrato en sí, no aporta valor al producto, sino que es sólo un formalismo que establece líneas de responsabilidad entre las partes. [MendesCalo 2010]

Es muy importante la relación entre la persona que quiere el software y los que la construyen. La *Colaboración* se refiere a amigabilidad, toma de decisiones conjuntas, comunicación rápida, y conexiones de interacción entre individuos. Aunque a veces los contratos son útiles, la colaboración fortalece el desarrollo tanto cuando hay un contrato como cuando no existe el contrato. [Cockburn 2007]

4) *Valorar la respuesta al cambio por sobre el seguimiento de un plan*. La evolución rápida y continua deben ser factores inherentes al proceso de desarrollo. Se debe valorar la capacidad de respuesta ante los cambios por sobre la capacidad de seguimiento y aseguramiento de planes pre-establecidos. [MendesCalo 2010]

El valor final se refiere a la rapidez para ajustarse a los cambios del proyecto. Construir un plan es útil y cada metodología ágil contiene actividades de planificación específicas, pero también tienen mecanismos para manejar cambios de prioridades.

Scrum, DSDM, Adaptative Software Development convocan a un desarrollo en períodos fijos (timebox) con repriorización posterior (no durante) a cada periodo fijo (timebox). XP permite repiorización durante el timebox. Los periodos son de 2 a 4 semanas. El trabajar de esta forma garantiza que el equipo tenga el tiempo y tranquilidad mental para desarrollar *software que funcione*. Al trabajar con iteraciones cortas, los sponsors del proyecto pueden cambiar las prioridades para que reflejen sus necesidades. [Cockburn 2007]

Los postulados descriptos anteriormente, inspiraron los doce principios del Manifiesto Ágil. Estos principios son las características que diferencian un proceso ágil de uno tradicional. Dos de ellos hacen referencias a generalidades, luego hay seis que tienen que ver directamente con el proceso de desarrollo de software a seguir y cuatro que están más directamente relacionados con el equipo de desarrollo, en cuanto metas a seguir y organización del mismo.

1) la prioridad es satisfacer al cliente mediante entregas de software tempranas y continuas;
2) los cambios en los requerimientos son aceptados;

3) software que funcione se entrega frecuentemente, con el menor intervalo posible entre entregas;
4) el cliente y los desarrolladores deben trabajar juntos a lo largo del proyecto;
5) el proyecto se construye en base a individuos motivados;
6) el dialogo cara a cara es el método más eficiente y efectivo para comunicar información dentro del equipo;
7) el software que funcione es la medida principal del progreso;
8) los procesos ágiles promueven el desarrollo sostenido;
9) la atención continua a la excelencia técnica y al buen diseño mejora la agilidad;
10) la simplicidad es esencial;
11) las mejores arquitecturas, requerimientos y diseños surgen de equipos auto-organizados, y
12) el equipo reflexiona en cómo ser más efectivos, y ajusta su comportamiento en consecuencia.

Aunque los creadores e impulsores de las metodologías ágiles más populares han suscrito el manifiesto ágil y coinciden con los principios enunciados anteriormente, cada metodología tiene características propias y hace hincapié en algunos aspectos más específicos. En el manifiesto no se encuentra la necesidad de diferentes formas de trabajar en diferentes situaciones, pero Jim Highsmith y Alistar Cockburn [Cockburn 2007] aconsejan tener esta idea en mente. Ser ágil es diferente, por ejemplo, para un proyecto de 100 personas que para uno de 10 personas. El punto es que las metodologías deben adaptarse o afinarse para el proyecto que se tiene en frente. Esta idea no se capturó en el manifiesto.

Algunos de los autores recomiendan metodologías ágiles para situaciones muy fluctuantes pero por ejemplo para Cockburn, en su propia experiencia comenta que aún sirvió esta metodología en proyectos supuestamente estables.

2.1.3 Definición de Metodologías Ágiles

De acuerdo con lo que menciona Damon B. Poole, en su libro "*Do it yourself agile",* dar una definición concisa de Metodología Ágil no es nada fácil, probablemente porque Ágil es realmente un paraguas de una variedad amplia de metodologías y porque Ágil está definido oficialmente como los 4 valores en el Manifiesto Ágil.

- Individuos e interacciones sobre procesos y herramientas
- Software funcionando sobre documentación extensiva
- Colaboración con el cliente sobre negociación contractual
- Respuesta ante el cambio sobre seguir un plan

Dado que hay tantas prácticas asociadas al desarrollo ágil, una forma más simple de definir Ágil es hacerlo en término de beneficios. Poole define el desarrollo ágil como aquel que, en comparación con el desarrollo tradicional, provee beneficios de mayor flexibilidad, retorno de inversión más alto, realización más rápida del retorno de Inversión, más alta calidad, mayor visibilidad, y paz sostenible. [Poole 2009]

Poole hace notar que se habla de Metodologías Ágiles como un solo paquete: si se adhiere a los principios del manifiesto ágil, se obtienen los beneficios. Pero, hay otra forma de mirarlas y es pensar que las metodologías ágiles introducen un conjunto nuevo y completo de prácticas al conjunto de herramientas de desarrollo. Estas prácticas incluyen: product backlog, programación de a pares, cliente en el lugar, integración continua, refactorización, desarrollo dirigido por pruebas (TDD) y muchas otras. Mientras que todas estas prácticas han estado asociadas con las Metodologías Ágiles o fueron creadas como resultado de las Metodologías Ágiles, su aplicación y beneficios resultantes pueden aplicarse completamente independientes de cualquier metodología específica. [Poole 2009]

Por otro lado, según [Giro 2002] [Ferrer 2003] [Canós 2003], se considera que un modelo es ágil o liviano cuando se emplea para su construcción una herramienta o técnica sencilla, que apunta a desarrollar un modelo aceptablemente bueno y suficiente en lugar de un modelo perfecto y complejo. Un modelo es suficientemente bueno cuando cumple con los objetivos para los que fue creado, esto es, logra principalmente el propósito de la comunicación; es entendible por la audiencia para la que fue concebido; no es perfecto y puede presentar algunos errores e inconsistencias no esenciales; posee un grado de detalle adecuado; suma valor al proyecto; y es suficientemente simple de construir [Ambler web].

2.1.3.1 Ciclo de las metodologías ágiles [MendesCalo 2008]

Si bien el ciclo de desarrollo que aplican las Metodologías Ágiles es iterativo e incremental, tal como se referencia en varios trabajos relacionados, el factor humano es fundamental para el éxito del proyecto [Cockburn 1999] [McConnell 2003] [Glass 2002]. Este modelo permite entregar el software en partes pequeñas y utilizables, conocidas como incrementos. Estas metodologías, adhiriendo a lo expuesto por Mendes Calo, aportan nuevos métodos de trabajo que requieren una cantidad de procesos, que no se desgastan con cuestiones administrativas – tales como planificación, control, documentación - ni tampoco defienden la postura extremista de la total falta de proceso. Debido a que se tiene conciencia de que los cambios indefectiblemente se producirán, el objetivo es reducir el costo de rehacer parte del producto por los cambios introducidos. Se intenta guiar el desarrollo hacia un objetivo que puede no permanecer constante en el tiempo a medida que aumenta el conocimiento de la aplicación a ser construida. Cada iteración se puede considerar como un mini-proyecto en el que las actividades de análisis de requerimiento, diseño, implementación y testing son llevadas a cabo con el fin de producir un subconjunto del sistema final. El proceso se repite varias veces produciendo un nuevo incremento en cada ciclo. Dicho proceso concluye cuando se haya elaborado el producto completo. Si bien todas las metodologías ágiles adoptan este ciclo, cada una presenta sus propias características. Aunque la mayoría de las metodologías ágiles satisfacen los postulados y principios del Manifiesto Ágil, no todas lo hacen de la misma manera. Más aún, el proceso de seleccionar la metodología que mejor se adapta a un problema en particular, se torna dificultoso.

2.1.3.2 Lo nuevo de las Metodologías Ágiles [Abrahamsson 2002]

Según Highsmith y Cockburn [Highsmith 2001], "lo que es nuevo sobre los métodos ágiles no son las prácticas que usan, sino el reconocimiento de personas como los conductores primarios al éxito del proyecto, junto con un foco intenso en la efectividad y mantenibilidad. Esto produce una nueva combinación de valores y principios que definen una vista *ágil* del mundo". Bohem en [Bohem 2002] ilustra el espectro de diferentes métodos de planificación como se muestra en la Figura 2.1.1., donde los hackers están ubicados en un extremo y el enfoque contractual pormenorizado con límites de acero en el extremo opuesto.

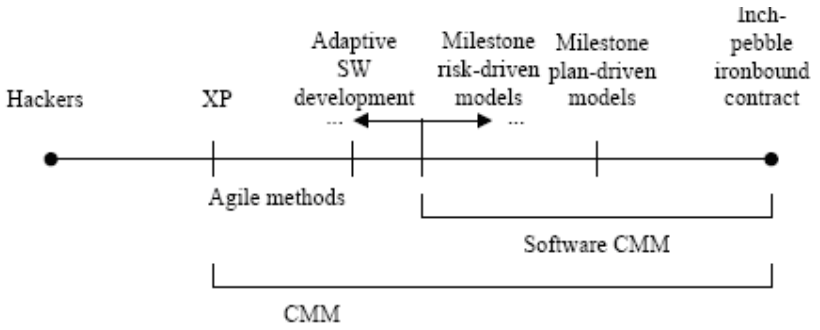


Figura 2.1.1. El espectro de la planificación [Bohem 2002]

Hawrysh y Ruprecht, [Hawrysh 2002], declaran que una sola metodología no puede funcionar para todo un espectro de proyectos diferentes, pero en su lugar el administrador de proyectos debe identificar la naturaleza específica del proyecto en mano y luego seleccionar la mejor

metodología de desarrollo aplicable. Para remarcar más ese punto, según [McCourley 2001] hay una necesidad tanto de métodos ágiles como de métodos orientados a procesos, así como no hay un modelo de desarrollo de software que le sea apropiado a todos los propósitos imaginables. Esta opinión es compartida por varios expertos en el campo [Glass 2000].

Cockburn, [Cockburn 2002], define el corazón de los métodos de desarrollo de software ágil como el uso de reglas livianas pero suficientes sobre el comportamiento de un proyecto y el uso de reglas orientadas a las personas y a la comunicación. El proceso ágil es tanto liviano como suficiente; liviano significando mantenerse maniobrable y suficiente como un asunto relacionado a mantenerse en el juego.

Según Abrahamsson Pekka, Salo Outi, Ronkainen Jussi & Juhani Warsta [Abrahamsson 2002], estudios han demostrado que las metodologías de desarrollo de software tradicionales dirigidas por planes no se usan en la práctica. Se ha argumentado que las metodologías tradicionales son demasiado mecánicas para usarse en detalle. Como resultado, los desarrolladores de software industriales se han vuelto escépticos sobre "nuevas" soluciones que son difíciles de comprender y por lo tanto permanecen sin usar. Los métodos de desarrollo de software ágil, que "oficialmente" comenzaron con la publicación del manifiesto ágil, hacen un intento por hacer un cambio en el campo de la ingeniería de software. Los métodos ágiles claman por colocar más énfasis en las personas, la interacción, el software funcionando, la colaboración del cliente, y el cambio más que en los procesos, las herramientas, los contratos y los planes.

El pensar de manera ágil es una vista centrada en las personas para desarrollar software [Abrahamsson 2002]. Las estrategias centradas en las personas son una fuente importante de ventaja competitiva, porque, a diferencia de la tecnología, los costos o las nuevas tecnologías, estas estrategias humanas son difíciles de imitar ([Pfeffer 1998];[Miller 2001]). Eso, sin embargo, no es una nueva creación. Una edición de verano de 1990 de American Programmer (Ed Yourdon's Software Journal, Vol3. No 7-8) estuvo dedicada exclusivamente a "Peopleware". El editor señala que "*todos saben que la mejor forma de mejorar la productividad y calidad software es enfocarse en las personas*". Por lo tanto, la idea que traen las metodologías ágiles no son nuevas, ni tampoco los agilistas se adjudican esto. Ellos, sin embargo, creen que los métodos ágiles de desarrollo de software proveen una forma novedosa de aproximarse a los problemas de ingeniería de software, así como también sostienen que los métodos no son de ninguna forma exhaustivos o capaces de resolver todos los problemas.

2.1.4 Metodologías a describir

El desarrollo ágil requiere innovación y mantenerse receptivo. Está basado en generar y compartir conocimiento entre el grupo de desarrollo y con el cliente. Los desarrolladores de software ágil hacen uso de las fortalezas de clientes, usuarios y desarrolladores, encontrando solo un proceso suficiente para balancear calidad y agilidad [Cockburn 2007].

Existe gran variedad de metodologías ágiles, pudiendo complementarse unas con otras dado que el enfoque en cada una puede ser diferente. Por ejemplo, XP se centra en la programación y Scrum en la administración. Muchas organizaciones están utilizando estas metodologías, como por ejemplo: Google, Canon, NEC, Seros, Fuji, Oracle, Toyota, Honda, Nokia, Yahoo!, Microsoft, HP, 3M, Sun, Epson [Programación-extrema].

Los resultados de la octava encuesta del "Estado del Desarrollo Ágil" realizada por VersionOne entre el 4 de agosto al 23 de octubre de 2013 fue publicada en 2014. Esta encuesta fue respondida por 3501 empresas. Entre los datos obtenidos, se puede mencionar que hay un 11% de incremento respecto de hace dos años en el número de personas que dicen que las metodologías ágiles ayudan a las organizaciones a completar más rápidamente los proyectos. A través de los datos de la encuesta, también se ve que quienes fallan al intentar adoptar una metodología ágil es a menudo por aspectos relacionados con la cultura organizacional y la resistencia al cambio. Específicamente las personas sienten una falta de planificación previa y pérdida de control en la administración del proyecto. Scrum y sus variantes siguen siendo las metodologías ágiles más difundidas. Kanban sigue ganando popularidad, aumentando un 7%. También aparecieron líneas nuevas e interesantes:

- Más equipos distribuidos están practicando metodologías ágiles. Este año pasó del 35% del 2012 al 75%, más del doble en un año.
- Más personas usan o planean utilizar herramientas para la administración de un proyecto ágil. (76% en 2013 respecto de un 66% en 2011)
- Mayor uso de retrospectiva, un aumento de un 10% en los dos últimos años

El 88% de los encuestados ha practicado desarrollo ágil durante el 2012 y un 53% viene trabajando con ágiles de 2 a 5 años y un 19% por más de 5 años. Los niveles donde se toman las decisiones son de un 61% en el nivel de administración, solo un 17% dentro de los equipos de desarrollo. El número de equipos dentro de las organizaciones que utiliza métodos ágiles sigue aumentando.

A continuación se muestran algunos de los datos extraídos de las encuestas [VersionOne web] y posteriormente se presentan algunos gráficos asociados a estas encuestas (Figuras 2.1.2 a Figura 2.1.8):

- **Razones para adoptar una metodología ágil**: Acelerar el tiempo de comercialización es de la razón número uno. La siguiente fue administrar cambio de prioridades.
- **Beneficios obtenidos de una implementación ágil**: Los tres beneficios más votados fueron: capacidad de administrar cambios en las prioridades con un 92%, aumentar la productividad con un 87% y mejorar la visibilidad del proyecto con un 86%.
- **Metodologías ágiles utilizadas**: se destaca Scrum y sus variantes que conforman más del 70% y se registra un aumento del uso de Kanban respecto a encuestas anteriores.
- **Técnicas ágiles empleadas**: Las más destacadas son Reunión Diaria de Pie (Daily Standup) con un 85%, planificación de la iteración 75% y retrospectiva 74%. Las pruebas de unidad con un 72% bajaron 2 puntos su porcentaje respecto del 2012. Los burn down charts son mantenido por un 70%. Hubo un aumento de un 10% en los dos últimos años respecto al uso de retrospectiva. Las pruebas de aceptación, y propiedad colectiva del código tienen bajas proporciones de uso y además sufrieron disminuciones respecto del 2012.
- **Causas de fracasos en los proyectos ágiles:** la mayoría lo atribuye a la filosofía de la compañía o bien alguna otra resistencia cultural.
- **Barreras para adoptar a futuro metodologías ágiles:** la misma razón que hace fallar los proyectos es la que hace que la organización no quiera volver a utilizarla. El 52%respondió que la razón es la imposibilidad de cambiar la cultura de la organización y un 42% la resistencia al cambio.
- **Éxito en los proyectos:** más de la mitad de los encuestados manifestaron que la mayoría, sino todos, sus proyectos ágiles resultaron ser exitosos.
- **Preferencias y usos de herramientas ágiles:** bug trakers y pizarras son las herramientas más utilizadas con porcentajes superiores al 80%. Le sigue el uso de wikis, herramientas de construcción automatizadas y planillas de cálculo.

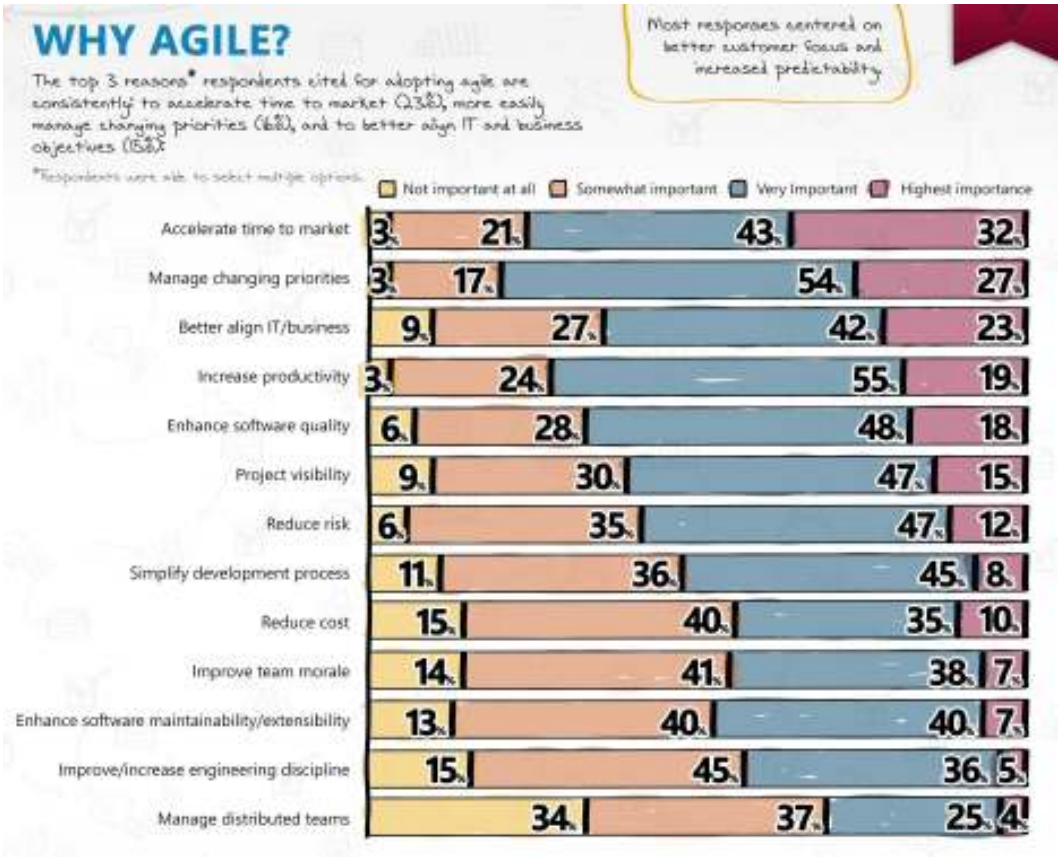


Figura 2.1.2. Razones para adoptar una metodología ágil [VisionOne]

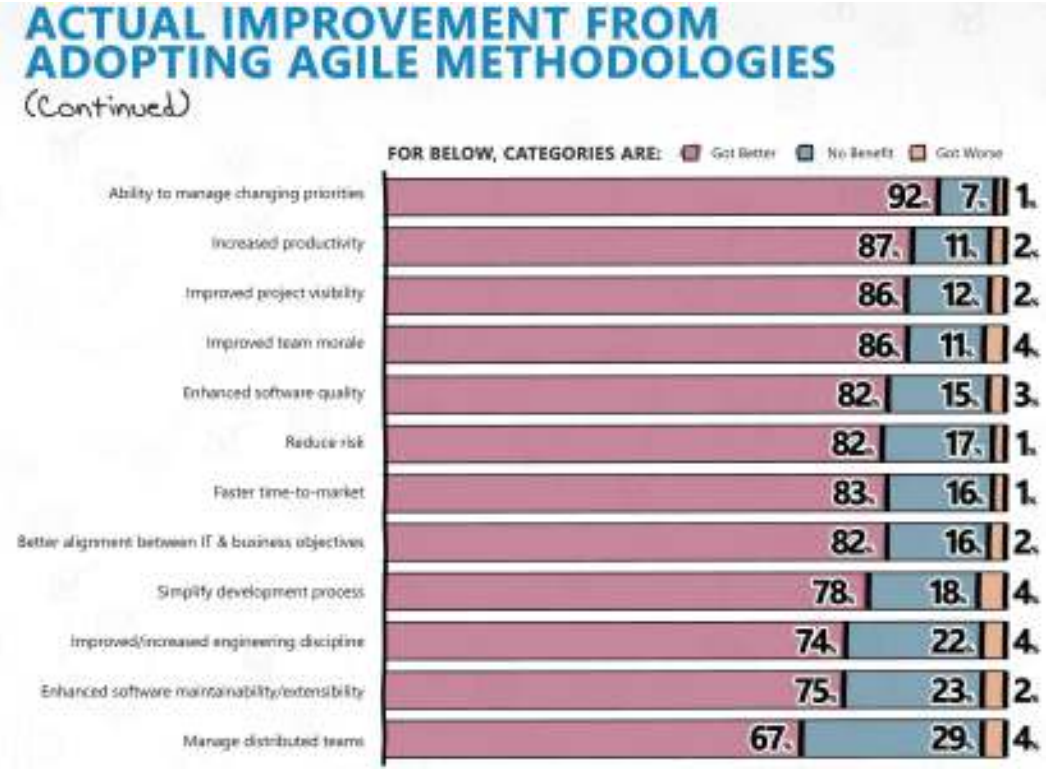


Figura 2.1.3. Mejoras obtenidas por implementar una metodología ágil [VisionOne]

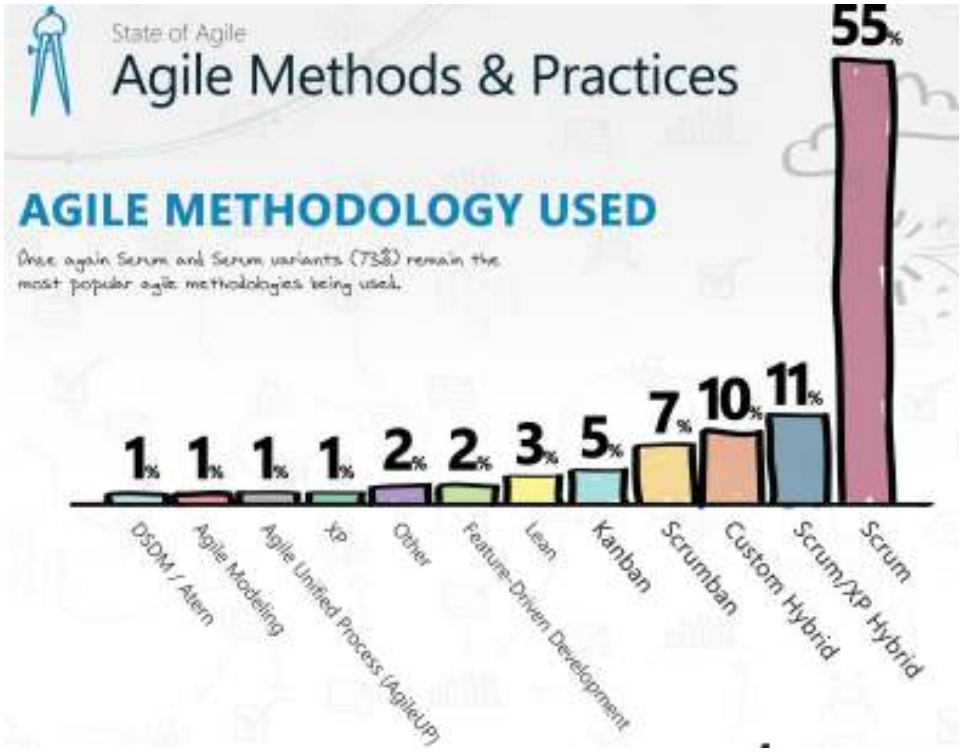


Figura 2.1.4. Metodologías y prácticas ágiles utilizadas [VisionOne]

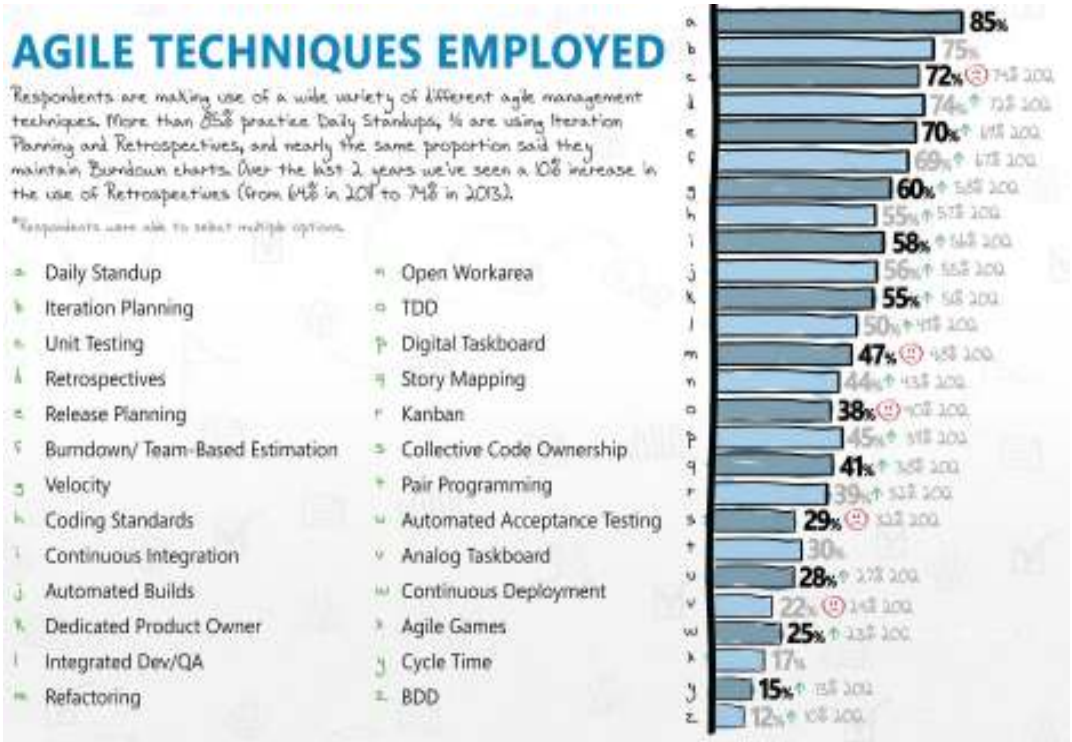


Figura 2.1.5. Técnicas ágiles empleadas [VisionOne]

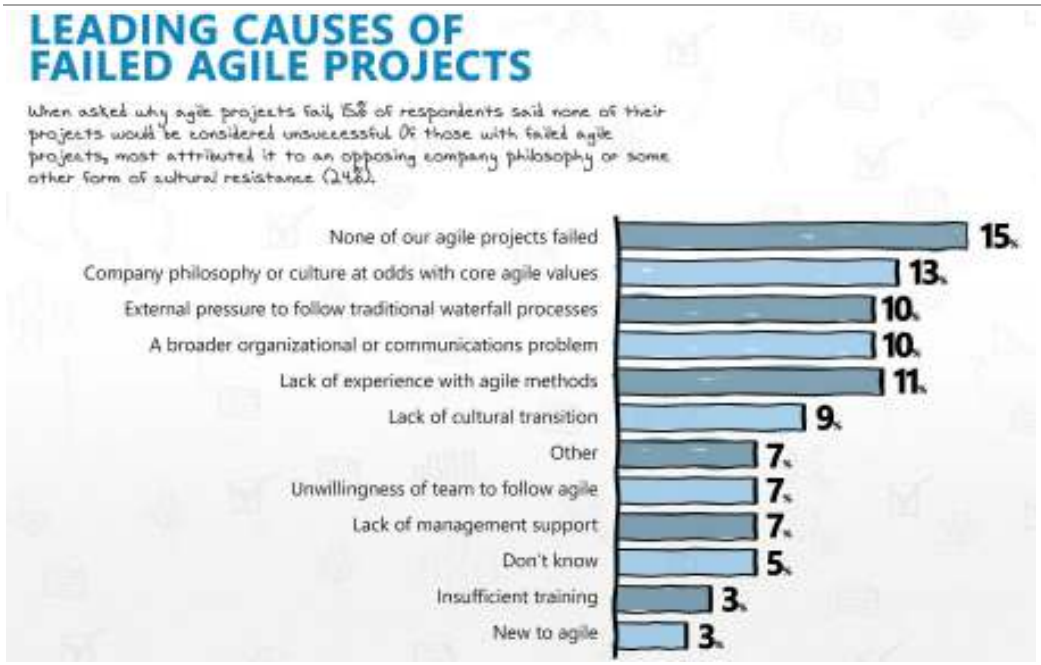


Figura 2.1.6. Causas de fracaso en los proyectos ágiles [VisionOne]

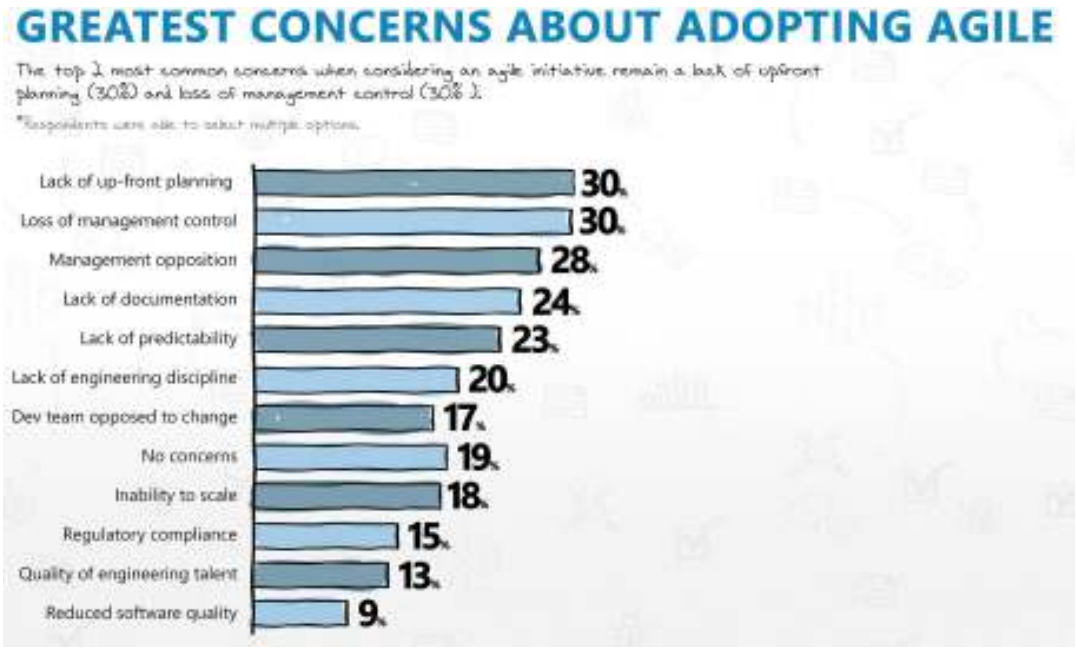


Figura 2.1.7. Preocupaciones sobre adoptar desarrollo ágil [VisionOne]

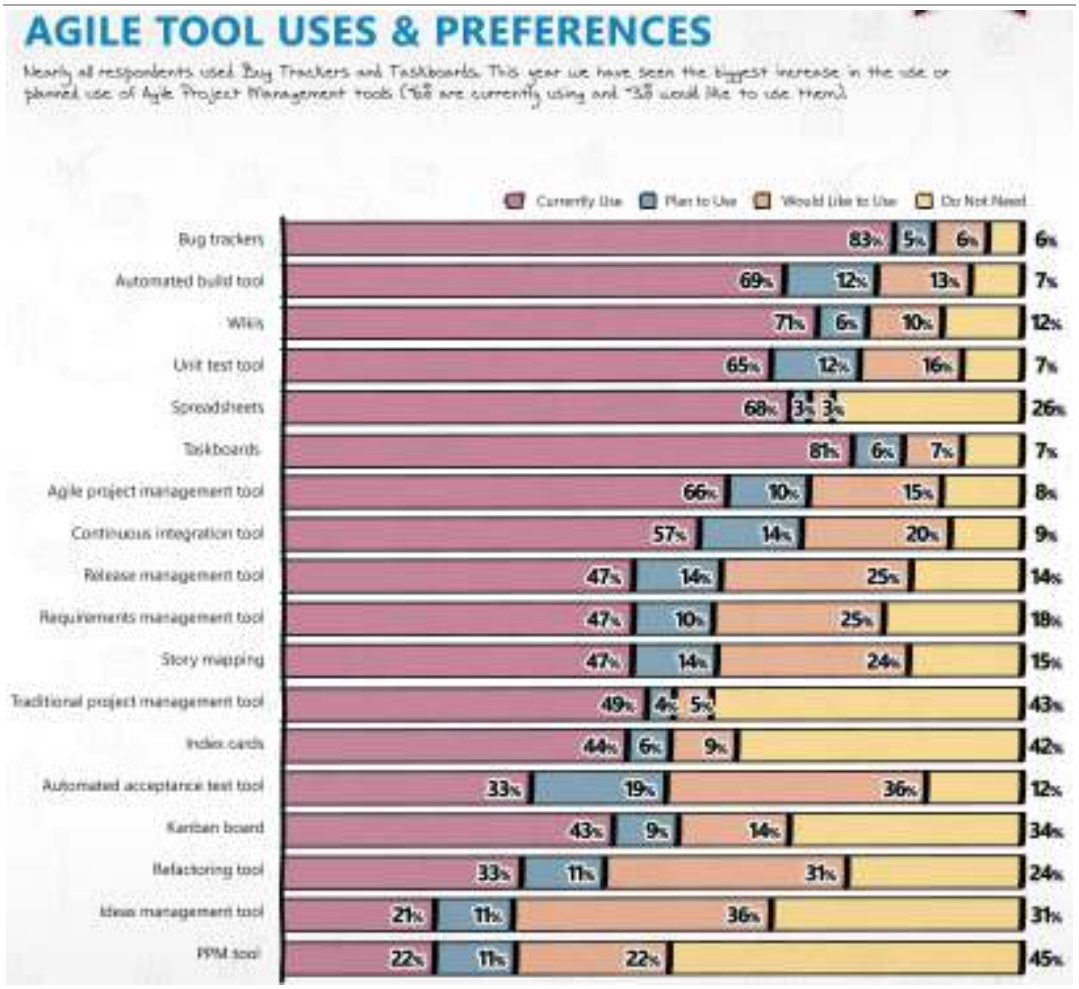


Figura 2.1.8. Preferencias y usos de herramientas ágiles [VisionOne]

En Argentina, un estudio realizado por M. Alvarez, F. Escobar, A. Nardin, E. Ricci Aparicio, G. Bioul, denominado Análisis de la implementación de prácticas ágiles en Argentina publicado en AgilSpain en diciembre de 2011 afirma que en Argentina de un grupo heterogéneo de empresas consultadas, el 85% usa Scrum aunque no en forma completa sino adaptándolo las técnicas que consideran más apropiadas. Algunas usan otras prácticas ágiles, XP y TDD, sin especificar porcentajes. [AgileSpain web]

Las metodologías a describir en este trabajo son XP, Scrum, Kanban, OpenUP, Cristal, TDD y DSDM. Las prácticas de ingeniería de XP evolucionaron junto con Scrum y los dos principales procesos ágiles de desarrollo trabajan bien juntos. Scrum y XP son las metodologías ágiles más utilizadas en todo el mundo y sus creadores son co-autores del Manifiesto Ágil [Sutherland 2011]. Debido a esto, se detallarán un poco más que el resto de las metodologías aquí mencionadas. Kanban está surgiendo con gran adhesión, por lo que es indispensable considerarla. Open UP presenta una metodología más cercana a lo que sería una metodología tradicional como es el Proceso Unificado. Crystal Clear, por el contrario de Open UP, va al extremo opuesto dejando la mayor libertad a la hora de definir un proceso de desarrollo. A TDD algunos la consideran más bien una técnica dentro de otra metodología, como por ejemplo dentro de XP, pero su planteo es sumamente provechoso a la hora de encarar un proceso de desarrollo ágil. DSDM fue la primera metodología ágil y sigue siendo utilizada mayormente en Europa. La intención final es tener un panorama general de diferentes propuestas ágiles actuales.

2.2 PROGRAMACIÓN EXTREMA (XP)

2.2.1 Introducción

La expresión Programación eXtrema es la traducción del inglés de "Extreme Programming" (XP) acuñada por Kent Beck quien ahora se considera como una de las principales figuras de este modelo de programación, junto a Ron Jeffries y Ward Cunningham quienes fueron partícipes de la conformación y divulgación de una metodología mucho más arriesgada, versátil y flexible para el desarrollo de software [Fowler 2003b]. Esta propuesta metodológica fue denominada extrema porque lleva a límites extremos algunos elementos y actividades comunes de la forma tradicional de programar, de ahí proviene su nombre.

Si bien la programación extrema surgió en la década de los 90's con el desarrollo de un proyecto de elaboración de software, XP hizo explosión en la escena del desarrollo de software en 2000-2001. Varias de las otras metodologías ágiles estuvieron prácticamente en la oscuridad antes de que XP apareciera en escena. Según [Highsmith], XP ayudó a llevar a la luz a toda la categoría "Ágil".

XP deriva de buenas prácticas que han estado alrededor por largo tiempo. Son prácticas simples, que podrían considerarse ingenuas o extrañas al principio, fácilmente adoptadas luego, apoyadas unas en otras, con reducción de actividades improductivas. El propio Kent afirma que "*ninguna de las ideas en XP son nuevas. Muchas son tan viejas como la programación*". Aunque, adhiriendo al comentario de Jim Highsmith, *"Mientras las prácticas que usa XP no son nuevas, las bases conceptuales y como se mezclan entre sí son nuevas y mejoran grandemente estas "viejas" prácticas*".

2.2.1.1 ¿Qué es XP?: Características generales

XP es una metodología que sigue la filosofía de las metodologías ágiles, cuyo objetivo es conseguir la máxima satisfacción del cliente en forma rápida y eficiente ante los cambios de requisitos. XP según [Beck 2000] es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. Propone realizar diseños simples, códigos simples y proporcionar rápida respuesta de lo requerido y lograr un cliente contento.

Se sustituye la documentación escrita por la comunicación directa entre clientes y desarrolladores o entre los propios desarrolladores. Propone un desarrollo iterativo a través de cuatro pasos, planificación, diseño, codificación y prueba. En cada iteración se añaden nuevas funcionalidades al software. Es especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

Se basa en una serie de prácticas y principios que se han ido gestando a lo largo de toda la historia de la ingeniería del software, son de sentido común pero llevadas al extremo. Usadas conjuntamente proporcionan una nueva metodología de desarrollo software que se puede englobar dentro de las metodologías ágiles o ligeras. Kent Beck describe la filosofía de XP en [Beck 2000] sin cubrir los detalles técnicos y de implantación de las prácticas. Posteriormente, otras publicaciones de experiencias se han encargado de dicha tarea.

"*La programación extrema se basa en la simplicidad, la comunicación y el reciclado continuo de código, para algunos no es más que aplicar una pura lógica"* [Calero]

2.2.2 Valores y Principios de XP

El proceso de desarrollo en XP está fundamentado en una serie de valores y principios que lo guían. Los valores representan aquellos aspectos que los autores de XP han considerado como fundamentales para garantizar el éxito de un proyecto de desarrollo de software. Un valor es una descripción de cómo debe enfocarse el desarrollo de software.

Los partidarios de la programación extrema dicen que estos cuatro valores son los necesarios para conseguir diseños y códigos simples, métodos eficientes de desarrollo software y clientes contentos. Estos valores le brindan consistencia y solidez al equipo de trabajo [Calero]. Los valores deben ser intrínsecos al equipo de desarrollo. Los cuatro valores de XP son:

- **Comunicación** que prevalece en todas las prácticas de XP. La comunicación cara a cara es la mejor forma de comunicación, entre los desarrolladores y el cliente. Es un método muy ágil. También apoya agilidad con la extensión del conocimiento tácito dentro del equipo del desarrollo, evitando la necesidad de mantener la documentación escrita.
- **Simplicidad** o sencillez que ayuda a que los desarrolladores de software encuentren soluciones más simples a problemas, según el cliente lo estipula. Los desarrolladores también crean características en el diseño que pudieran ayudar a resolver problemas en un futuro.
- **Retroalimentación continua** o feedback del cliente que permite a los desarrolladores llevar y dirigir el proyecto en una dirección correcta hacia donde el cliente decida. Apunta a la respuesta rápida, constante e iterativa que se le ofrece al cliente.
- **Coraje** que requiere que los desarrolladores vayan a la par con el cambio, ya que el cambio es inevitable, pero el estar preparado con una metodología ayuda a ese cambio.

De los cuatro valores, quizás el que llame más la atención es el de coraje. Detrás de este valor se encuentra el lema "si funciona, mejóralo", que va en contra de la práctica habitual de no tocar algo que funciona, por si acaso. Aunque también es cierto que al tener las pruebas unitarias, no se pide a los desarrolladores una heroicidad, sino sólo coraje. Otro punto de vista del coraje, es el del administrador del proyecto quien debe estar lo suficientemente seguro de sí mismo para abandonar un buen trato de control, ya que no tendrá más la autoridad para determinar qué quiere y cuándo lo quiere.

Los principios suponen un puente entre los valores, algo intrínseco al equipo de desarrollo y las prácticas, que se verán posteriormente, y que están más ligadas a las técnicas que se han de seguir. Los principios fundamentales se apoyan en los cuatro valores previamente definidos y también son cuatro: retroalimentación veloz, modificaciones incrementales, trabajo de calidad y asunción de simplicidad.

2.2.3 Instrumentos usados en XP

Dentro de los instrumentos usados en XP resaltan las historias de usuarios como el instrumento principal de la metodología. Además, se utilizan otros artefactos como las tareas de ingeniería o programación y las tarjetas CRC (Tarjetas de Clases, Responsabilidades y Colaboración).

2.2.3.1 Las Historias de Usuario

Las historias de usuario son la técnica utilizada en XP para especificar los requisitos del software. Representan una breve descripción del comportamiento del sistema. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. Se emplea terminología del cliente sin lenguaje técnico, se realiza una historia de usuario por cada característica principal del sistema. Se emplean para hacer estimaciones de tiempo y para el plan de lanzamientos. [Jefrries web]

El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento las historias de usuario pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas. [Letelier 2006] Para efectos de planificación, las historias pueden ser de una a tres semanas de tiempo de programación (para no superar el tamaño de una iteración).

Las historias de usuario tienen tres aspectos: tarjeta (se almacena suficiente información para identificar y detallar la historia), conversación (cliente y programadores discuten la historia para ampliar los detalles) y pruebas de aceptación (permite confirmar que la historia ha sido implementada correctamente). Las historias de usuario se descomponen en tareas de programación y son asignadas a los programadores para ser implementadas durante una iteración.

Respecto de la información contenida en la historia de usuario propiamente dicha, existen varias plantillas sugeridas pero no existe un consenso al respecto. Beck en su libro [Beck 2000] presenta un ejemplo de ficha (*customer story* y *task card*) en la cual pueden reconocerse los siguientes contenidos: fecha, tipo de actividad (nueva, corrección, mejora), prueba funcional, número de historia, prioridad técnica y del cliente, referencia a otra historia previa, riesgo, estimación técnica, descripción, notas y una lista de seguimiento con la fecha, estado cosas por terminar y comentarios.

2.2.4 Roles XP

En este apartado se describen los roles de acuerdo con la propuesta original de Beck, si bien en otras fuentes de información aparecen algunas variaciones y extensiones de roles en XP. Los roles originales fueron: Programador, Cliente, Encargado de Pruebas, Encargado de Seguimiento, Entrenador, Consultor y Gestor.

1. **Programador:** es el encargado de escribir las pruebas unitarias y producir el código del sistema. Es responsable de las decisiones técnicas y de construir el sistema. Debe existir una comunicación y coordinación adecuada entre los programadores y otros miembros del equipo.

2. **Cliente:** escribe las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración buscando aportar mayor valor al negocio. El cliente es sólo uno dentro del proyecto pero puede corresponder a un interlocutor que está representando a varias personas que se verán afectadas por el sistema.

3. **Encargado de pruebas (Tester):** ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.

4. **Encargado de seguimiento (Tracker):** proporciona realimentación al equipo en el proceso XP. Su responsabilidad es verificar el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, comunicando los resultados para mejorar futuras estimaciones. También realiza el seguimiento del progreso de cada iteración y evalúa si los objetivos son alcanzables con las restricciones de tiempo y recursos presentes. Determina cuándo es necesario realizar algún cambio para lograr los objetivos de cada iteración.

5. **Entrenador (Coach):** es responsable del proceso global. Es el líder del equipo y quien toma las decisiones importantes. Es necesario que conozca a fondo el proceso XP para proveer guías a los miembros del equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente y tiende a estar en segundo plano cuando el equipo madura.

6. **Consultor:** es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto. Guía al equipo para resolver un problema específico.

7. **Gestor (Big boss):** es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es de coordinación.

2.2.5 El Proceso de desarrollo en XP

El desarrollo es la pieza clave de todo el proceso XP. Todas las tareas tienen como objetivo realizar el desarrollo a la máxima velocidad, sin interrupciones y siempre en la dirección correcta. A grandes rasgos, el ciclo de desarrollo se puede simplificar en los siguientes pasos [Letelier 2006]:

1. El cliente define el valor de negocio a implementar.
2. El programador estima el esfuerzo necesario para su implementación.
3. El cliente selecciona qué construir, de acuerdo con sus prioridades y las restricciones de tiempo.
4. El programador construye ese valor de negocio.
5. Vuelve al paso 1.

En todas las iteraciones de este ciclo tanto el cliente como el programador aprenden. No se debe presionar al programador a realizar más trabajo que el estimado, ya que se perderá calidad en el software o no se cumplirán los plazos. De la misma forma el cliente tiene la obligación de manejar el ámbito de entrega del producto, para asegurarse que el sistema tenga el mayor valor de negocio posible con cada iteración.

El ciclo de vida ideal de XP consiste de seis fases [Beck 2000]: Exploración, Planificación de la Entrega (Release), Iteraciones, Producción, Mantenimiento y Muerte del Proyecto:

1. **Exploración**: en esta fase, el cliente plantea a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto ya que se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. Esta fase toma pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.

2. **Planificación de la Entrega**: en esta fase el cliente establece la prioridad de cada historia de usuario, y los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses. Esta fase dura unos pocos días.

   Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores utilizando como medida el punto. Un punto, equivale a una semana ideal de programación. Las historias generalmente valen de uno a tres puntos. Por otra parte, el equipo de desarrollo mantiene un registro de la "velocidad" de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración.

3. **Iteraciones**: esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El plan de entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fuercen la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que *es el cliente quien decide* qué historias se implementarán en cada iteración. Al final de la última iteración el sistema estará listo para entrar en producción.

   Los elementos que deben tomarse en cuenta durante la elaboración del plan de la iteración son historias de usuario no abordadas, velocidad del proyecto, pruebas de aceptación no superadas en la iteración anterior y tareas no terminadas en la

iteración anterior. Todo el trabajo de la iteración es expresado en tareas de programación, cada una de ellas es asignada a un programador como responsable, pero llevadas a cabo por parejas de programadores.

Será tarea del cliente retroalimentar al equipo de desarrolladores después de cada iteración con los problemas con los que se ha encontrado, mostrando sus prioridades, expresando sus sensaciones. El cliente puede cambiar de opinión sobre la marcha y a cambio debe encontrarse siempre disponible para resolver dudas del equipo de desarrollo y para detallar los requisitos especificados cuando sea necesario. Es el cliente el que tiene que escribir lo que quiere, no se permite que alguien del equipo de desarrolladores lo escriba por él.

Antes de codificar se debe hacer un diseño. Como XP es una metodología simple, el diseño debe ser simple. Aunque en general el diseño es realizado por los propios desarrolladores en ocasiones se reúnen aquellos con más experiencia o incluso se involucra al cliente para diseñar las partes más complejas. En estas reuniones se suelen emplean las tarjetas CRC cuyo objetivo es facilitar la comunicación y documentar los resultados.

4. **Producción**: la fase de producción requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase.

   Es posible que se disminuya el tiempo que toma cada iteración, de tres a una semana. Las ideas que han sido pospuestas y las sugerencias son documentadas para su posterior implementación (por ejemplo, durante la fase de mantenimiento).

5. **Mantenimiento**: mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para realizar esto se requiere de tareas de soporte para el cliente. De esta forma, la velocidad de desarrollo puede bajar después de la puesta del sistema en producción. La fase de mantenimiento puede requerir nuevo personal dentro del equipo y cambios en su estructura.

6. **Muerte del Proyecto**: es cuando el cliente no tiene más historias para ser incluidas en el sistema. Esto indica que se satisfacen todas las necesidades del cliente en aspectos como rendimiento y fiabilidad del sistema. Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura. La muerte del proyecto también ocurre cuando el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo.

En la Figura 2.2.1 se puede observar cuales son los ciclos de planificación y retroalimentación dentro del proceso de desarrollo según XP, con las duraciones de cada uno.

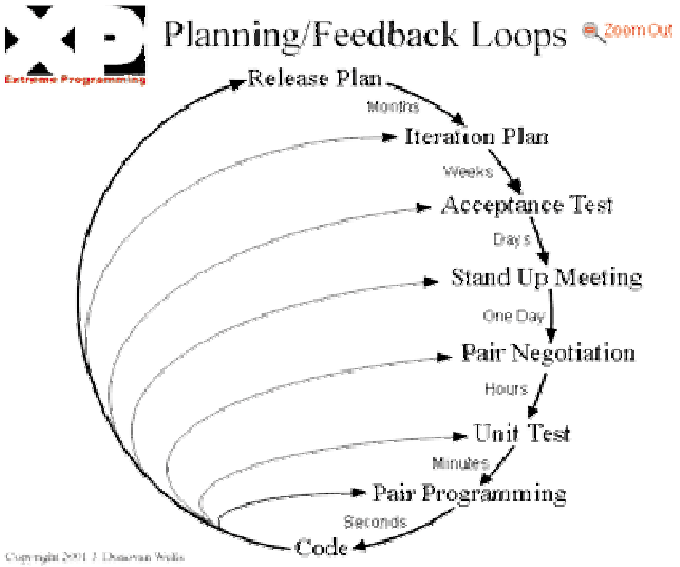


Figura 2.2.1. Proceso XP – Ciclos de planificación y retroalimentación [XP web]

2.2.6 Prácticas XP

La principal suposición que se realiza en XP es la posibilidad de disminuir la mítica curva exponencial del costo del cambio a lo largo del proyecto, lo suficiente para que el diseño evolutivo funcione. XP apuesta por un crecimiento lento del costo del cambio y con un comportamiento asintótico. Esto se consigue gracias a las tecnologías disponibles para ayudar en el desarrollo de software y a la aplicación disciplinada de ciertas prácticas [Letelier 2006].

XP especifica ciertas prácticas concretas de programación que deben llevarse a cabo al implementar este modelo. Las mismas tienen su origen en prácticas bien conocidas en la ingeniería del software, tal como se mencionó previamente. Las prácticas de XP facilitan el desarrollo de una aplicación y permiten a los desarrolladores obtener software de calidad. Las doce prácticas que caracterizan a la metodología son:

- **Juego de la Planificación:** es un espacio frecuente de comunicación entre el cliente y los programadores. El juego de la planificación define el límite o el punto de interacción entre clientes y desarrolladores, especificando las responsabilidades de ambas partes. El equipo técnico realiza una estimación del esfuerzo requerido para la implementación de las historias de usuario y los clientes deciden sobre el ámbito y tiempo de las entregas y de cada iteración. La duración de cada iteración se suele estimar en 2-3 semanas de trabajo.
- **Entregas pequeñas**: la idea es producir rápidamente versiones del sistema que sean operativas, aunque obviamente no cuenten con toda la funcionalidad pretendida para el sistema pero sí que constituyan un resultado de valor para el negocio. Kent Beck enuncia en su libro *Extreme Programming Explained: Embrace Change* [Beck 2000] *"cada entrega debe ser tan pequeña como sea posible, conteniendo los requerimientos con mayor valor de negocio".* Las entregas pequeñas proveen un sentido de logro o éxito, que a menudo falta en proyectos largos, y retroalimentación frecuente y relevante. Una entrega no debería tardar más de 3 meses.
- **Metáfora:** en XP no se enfatiza la definición temprana de una arquitectura estable para el sistema. Dicha arquitectura se asume evolutiva y los posibles inconvenientes que se generarían por no contar con ella explícitamente en el comienzo del proyecto se solventan con la existencia de una metáfora. El sistema es definido mediante una metáfora o un conjunto de metáforas compartidas por el cliente y el equipo de desarrollo. Una metáfora es una historia compartida que describe cómo debería funcionar el sistema. La práctica de la metáfora consiste en formar un conjunto de nombres que actúen como vocabulario para hablar sobre el dominio del problema. Este conjunto de nombres ayuda a la nomenclatura de clases y métodos del sistema [Letelier 2006]. El principal objetivo de la metáfora es

mejorar la comunicación entre todos los integrantes del equipo al crear una visión global y común del sistema que se pretende desarrollar.

- **Diseño simple:** El diseño simple tiene dos partes: (1) diseñar para la funcionalidad que se definió, no para potenciales funcionalidades futuras y (2) crear el mejor diseño y más simple que puede proveer esa funcionalidad. Se debe diseñar la solución más simple que pueda funcionar y ser implementada en un momento determinado del proyecto. La complejidad innecesaria y el código extra debe ser removido inmediatamente. Según Kent Beck, en cualquier momento el diseño adecuado para el software es aquel que supera con éxito todas las pruebas, no tiene lógica duplicada y tiene el menor número posible de clases y métodos.
- **Pruebas:** XP usa dos tipos de pruebas: de unidad y funcionales (o de aceptación). La producción de código está dirigida por las pruebas unitarias. Las pruebas unitarias son establecidas antes de escribir el código y son ejecutadas constantemente ante cada modificación del sistema. Por lo tanto, la práctica de pruebas de unidad involucra desarrollar las pruebas para la característica deseada antes de escribir el código. Una vez que se escribe el código, éste está sujeto a las suites de prueba con retroalimentación inmediata. Las pruebas unitarias se llevan a cabo por los desarrolladores cada vez que se añade código, y además sobre cada historia por separado antes de ser integrado al software. Los clientes escriben las pruebas funcionales para cada historia de usuario que deba validarse y son las que él mismo ejecuta para validar el software, asistidos por el tester (Ver Figura 2.2.2). En este contexto de desarrollo evolutivo y de énfasis en pruebas constantes, es crucial la automatización para apoyar esta actividad [Letelier 2006].

Caso de Prueba de Aceptación	
Código:	**Historia de Usuario (Nro. y Nombre):**
Nombre:	
Descripción:	
Condiciones de Ejecución:	
Entrada / Pasos de ejecución:	
Resultado Esperado:	
Evaluación de la Prueba:	

Figura 2.2.2. Modelo propuesto para una prueba de aceptación [Canós 2003]

- **Refactorización (Refactoring):** la refactorización es una actividad constante de re-estructuración del código con el objetivo de remover duplicación de código, mejorar su legibilidad, simplificarlo y hacerlo más flexible para facilitar los posteriores cambios. La refactorización mejora la estructura interna del código sin alterar su comportamiento externo [Letelier 2006]. Tiene su justificación principal en que el código no sólo tiene que funcionar, también debe ser simple. Esto hace que, a la larga, refactorizar ahorre mucho tiempo y suponga un incremento de calidad. Por cierto, tal es el énfasis que se pone en la refactorización que, inclusive se deben refactorizar las pruebas unitarias. Según Highsmith,

la refactorización distingue a XP de los demás enfoques – el continuo rediseño de software para mejorar su respuesta al cambio. Debería pensarse a XP como un rediseño continuo. En tiempos de cambios rápidos y constantes se necesita prestar más atención en refactorización. Debe considerarse a la refactorización como rediseño incremental y no rediseño monumental.

- **Programación en parejas:** "*Programación de a pares es un diálogo entre dos personas tratando de programar simultáneamente y entender cómo programar mejor*" [Beck 2000]. Toda la producción de código debe realizarse con trabajo en parejas de programadores frente a una computadora, con un sólo mouse y un sólo teclado. Cada miembro de la pareja juega su papel: uno codifica en la computadora y piensa la mejor manera de hacerlo, el otro piensa más estratégicamente. Tener dos personas sentadas frente a una misma terminal, una ingresando código o casos de prueba y otra revisando y pensando, crea un intercambio dinámico y continuo. El principal objetivo es realizar de forma continua y sin parar el desarrollo, una revisión de diseño y de código. El emparejamiento es dinámico. Cualquier miembro del equipo se puede emparejar con cualquiera. Las parejas deben ir rotando de forma periódica para hacer que el conocimiento del sistema se vaya difundiendo por el equipo, a la vez que se fomenta el entrenamiento cruzado [Jeffries 2007]. Existen estudios que concluyen que esta práctica es eficaz, justificándola con aspectos psicológicos y sociológicos [Cockburn 2004].
- **Propiedad colectiva del código:** Brinda a cualquier programador en el proyecto el "permiso" para cambiar cualquier parte del código en cualquier momento. Esta práctica motiva a todos a contribuir con nuevas ideas en todos los segmentos del sistema, evitando a la vez que algún programador sea imprescindible para realizar cambios en alguna porción de código. Por otro lado, si bien nadie conoce cada parte igual de bien, todos conocen algo sobre cada parte, esto ayuda a preparar para la sustitución no traumática de cada miembro del equipo.
- **Integración continua**: cada pieza de código se integra en el sistema una vez que está lista. Así, el sistema puede llegar a ser integrado y construido varias veces en un mismo día. El código se debe integrar como mínimo una vez al día, y realizar las pruebas sobre la totalidad del sistema. Una pareja de programadores se encargará de integrar todo el código en una máquina y realizar todas las pruebas hasta que éstas funcionen al 100%. para que el nuevo código sea incorporado definitivamente. La integración continua a menudo reduce la fragmentación de los esfuerzos de los desarrolladores por falta de comunicación sobre lo que puede ser reutilizado o compartido. El equipo de desarrollo está más preparado para modificar el código cuando sea necesario, debido a la confianza en la identificación y corrección de los errores de integración [Letelier 2006].
- **40 horas por semana**: se debe trabajar un máximo de 40 horas por semana. Las horas extras son síntoma de serios problemas en el proyecto. Si esto ocurre, probablemente está ocurriendo un problema que debe corregirse. El trabajo extra desmotiva al equipo. Los proyectos que requieren trabajo extra para intentar cumplir con los plazos suelen al final ser entregados con retraso. En lugar de esto se puede realizar el juego de la planificación para cambiar el ámbito del proyecto o la fecha de entrega.
- **Cliente in-situ**: el cliente tiene que estar presente y disponible todo el tiempo para el equipo. Gran parte del éxito del proyecto XP se debe a que es el cliente quien conduce constantemente el trabajo hacia lo que aportará mayor valor de negocio y los programadores pueden resolver de manera inmediata cualquier duda asociada. La comunicación oral es más efectiva que la escrita, ya que esta última toma mucho tiempo en generarse y puede tener más riesgo de ser mal interpretada. En [Jeffries2001] Jeffries indica que se debe pagar un precio por perder la oportunidad de un cliente con alta disponibilidad.
- **Estándares de programación:** XP enfatiza la comunicación de los programadores a través del código, con lo cual es indispensable que se sigan ciertos estándares de programación (del equipo, de la organización u otros estándares reconocidos para los lenguajes de programación utilizados). Los estándares de programación mantienen el código legible para los miembros del equipo, facilitando los cambios. Son un apoyo a la metáfora ya que consiste en una correcta elección de los nombres que se escojan durante

el proyecto para las clases, métodos, variable, procesos, etc. Nombres bien puestos implican claridad, reusabilidad y simplicidad.

El mayor beneficio de las prácticas se consigue con su aplicación conjunta y equilibrada puesto que se apoyan unas en otras. Las prácticas de XP son realmente un sistema de prácticas diseñadas para interactuar, hacer contrapeso y reforzar cada una, por eso a su vez elegir algunas para usar y descartar otras puede ser delicado. [Highsmith]

2.2.7 Conclusiones

De todas las metodologías ágiles, ésta es la que ha recibido más atención [Fowler2003]. Los defensores de XP son cuidadosos en expresar cuándo consideran que es apropiado XP y cuándo no lo es. Defensores como Kent Beck y Ron Jeffries pueden imaginar que XP tiene una aplicabilidad más amplia, generalmente son cautelosos sobre sus afirmaciones. Por ejemplo, ambos son claros respecto de la aplicabilidad de XP en equipos pequeños (menos de 10 personas) en un mismo lugar, una situación donde ellos han tenido experiencia. Ellos no tratan de convencer a las personas que las prácticas funcionarán en equipos de 200. [Higsmith 2002]

Sin embargo, como se resalta en [Letelier 2006], hay que tener presente una serie de inconvenientes y restricciones para su aplicación, tales como: están dirigidas a equipos pequeños o medianos (Beck sugiere que el tamaño de los equipos se limite de 3 a 20 miembros como máximo, otros dicen no más de 10), el entorno físico debe ser un ambiente que permita la comunicación y colaboración entre todos los miembros del equipo durante todo el tiempo, cualquier resistencia del cliente o del equipo de desarrollo hacia las prácticas y principios puede llevar el proceso al fracaso (el clima de trabajo, la colaboración y la relación contractual son claves), el uso de tecnologías que no tengan un ciclo rápido de retroalimentación o que no soporten fácilmente el cambio, etc.

Aunque en la actualidad ya existen libros asociados a cada una de las metodologías ágiles existentes y también abundante información en internet, es XP la metodología que resalta por contar con la mayor cantidad de información disponible y es con diferencia la más popular.

2.3 SCRUM

2.3.1 Introducción

Scrum no es una metodología es un camino" Ken Schwaber, Co, Nov 2005. [Higsmith 2002]

El concepto de Scrum tiene su origen a principios de los 90's. Está basado en un estudio de gestión de equipos de 1986 desarrollado por Hirotaka Takeuchi e Ikujiro Nonaka llamado *The New New Product Developement Game*. Era un estudio sobre los nuevos procesos de desarrollo utilizados en productos exitosos en Japón y los Estados Unidos (cámaras de fotos Canon, fotocopiadoras Xerox, automóviles Honda, ordenadores HP y otros). En este estudio se comparaba la forma de trabajo de equipos de desarrollo, altamente productivos y multidisciplinares, con la colaboración entre los jugadores de Rugby y su formación de Scrum. [Sutherland 2011].

Scrum es una metodología para desarrollo ágil de productos software. Esta metodología, si bien ha sido aplicada principalmente a proyectos de software, un número de proyectos que no están relacionados con el software han sido administrados con Scrum – los principios son aplicables a cualquier proyecto [Highsmith]. Actualmente SCRUM es uno de los métodos ágiles que está creciendo más y ya en 2011 lo usaba el 75% de equipos ágiles alrededor del mundo. [Sutherland web] [Sutherland 2011]. La encuesta de 2013 de VisionOne obtuvo porcentajes similares, tal como se mostró anteriormente. La Scrum Alliance [ScrumAlliance] es la organización sin ánimo de lucro que se encarga de difundir Scrum.

Está siendo utilizado por grandes y pequeñas compañías, incluyendo Yahoo, Microsoft, Google, Lockheed Martin, Motorola, SAP, Cisco, GE Medical, Capital One y la Reserva Federal de los Estados Unidos [Deemer]. Muchos equipos que están usando Scrum reportan mejoras importantes, y en algunos casos transformaciones completas, tanto en productividad como en la moral. Scrum es simple, poderoso y cimentado en el sentido común. [Highsmith]

"El corazón del enfoque Scrum es la creencia que la mayoría de los desarrollos de sistemas tienen las bases filosóficas erróneas ". Ken Schwaber afirma que el desarrollo de sistemas no es un "proceso definido" como asumen las metodologías rigurosas, sino un "proceso empírico". [Higsmith 2002]. Un proceso empírico requiere un estilo completamente diferente de administración. Los procesos empíricos, a diferencia de los rigurosos, no pueden repetirse consistentemente, requieren por lo tanto monitoreo y adaptación constante. "*Los desarrolladores y administradores de proyectos son forzados a vivir una mentira – deben pretender que pueden planificar, predecir y entregar*". Eso dice Ken Schwaber.

Scrum comienza con la premisa que el mundo es complicado y por lo tanto "*no se puede predecir o planear definitivamente lo que se entregará, cuando se entregará y que calidad y costo tendrá"* dice Ken Schwaber. No se basa en seguir un plan sino en la adaptación continua a las circunstancias de la evolución de proyecto. Mientras XP tiene un sabor definido a programación (programación de a pares, estándares de codificación, refactorización), Scrum tienen un énfasis en la administración de proyecto.

2.3.2 Principales elementos

Estos son los principales elementos de Scrum, si bien se utilizan otros que se irán describiendo más adelante.

- **Product Backlog:** Lista priorizada de funcionalidades técnicas y de negocio. Estas funcionalidades son requisitos a muy alto nivel de lo que debe hacer la aplicación, donde se listan características, funciones, tecnología, mejoras, bugs, etc. que serán aplicadas al producto. El Product Backlog es el punto de inicio.

- **Sprint Backlog:** Lista de tareas de un Sprint. El Sprint Backlog identifica y define el trabajo a ser alcanzado por el equipo de desarrollo durante un Sprint. A un nivel el Sprint Backlog identifica las características mientras que a otro nivel, identifica las tareas requeridas para implementar esas características.

- **Incremento:** Parte de un sistema desarrollado en un Sprint. Este producto desarrollado es potencialmente entregable al final de cada Sprint, implica que todo está completamente terminado en cada Sprint; y se podría realmente empaquetar o desplegar inmediatamente después de la Revisión del Sprint con mínimas tareas, si bien a veces se necesitan ciertos trabajos de acabado tales como pruebas o documentación.

2.3.3 Roles

En Scrum se definen varios roles, estos están divididos en dos grupos. Por un lado figuran los que están comprometidos con el proyecto y el proceso Scrum. Por otro lado, aquellos que en realidad no forman parte del proceso Scrum, pero que deben tenerse en cuenta y cuya participación es importante. [CaraballoMaestre 2009]

2.3.3.1 Roles comprometidos con el Proyecto

- **Product Owner:** Representa la voz del cliente. Escribe historias de usuario, las prioriza, y las coloca en el Product Backlog.

- **Scrum Master:** Protege al equipo de distracciones y de otros elementos externos y lo mantiene enfocado. Elimina obstáculos que alejen al grupo de la consecución de objetivos del Sprint. No es el líder del grupo, ya que el grupo se autogestiona. Dirige los scrums diarios. Realiza el seguimiento del avance.

- **Equipo:** Conformado por no más de 8 personas (si hay más se organizan varios equipos que trabajan sobre el mismo product backlog). Tiene la responsabilidad de entregar el producto. Son autónomos y auto-organizados. Deben entregar un conjunto de ítems del Backlog al final del Sprint.

2.3.3.2 Roles involucrados en el proceso

- **Usuarios:** Son los destinatarios finales del producto.

- **Stakeholders o Interesados (Clientes, Proveedores**): Se refiere a la gente que hace posible el proyecto y para quienes el proyecto producirá el beneficio acordado que lo justifica.

- **Managers:** Son aquellos que establecen el ambiente para el desarrollo del producto.

2.3.4 El proceso Scrum

Scrum define un marco de administración de proyecto donde las actividades de desarrollo – recolección de requerimientos, diseño, programación – tienen lugar. El período de desarrollo es de una iteración de 30 días llamada Sprint, si bien podría trabajarse con sprints de menor duración. El marco de trabajo de Scrum tiene tres componentes o cuatro según como se los mire.

- **pre-Sprint**: planificación del sprint

- **Sprint**: ciclo de trabajo

- **post-Sprint**: revisión y retrospectiva del sprint (pueden considerarse como dos componentes diferentes posteriores al sprint)

El punto focal es el Sprint, donde se desarrolla software que funcione. (Ver Figura 2.3.1.)

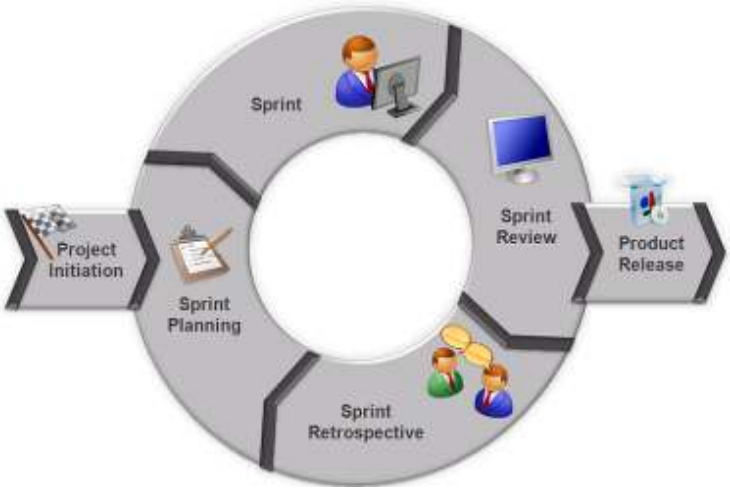


Figura 2.3.1. Ciclo de vida de un proyecto Scrum [Parasuraman 2011]

2.3.4.1 Planificación pre-sprint

Antes de iniciar esta actividad el Product Owner ya tiene que haber creado y priorizado el Product Backlog, posiblemente en una reunión previa para no extender más la duración de esta reunión. En esta reunión están presentes el equipo de desarrollo, el administrador y el Product Owner. Aquí, el Product Owner define las características a incorporar en el siguiente Sprint y el equipo define las tareas necesarias para entregar esas características. El equipo determina la lista de tareas y estima cuanto puede cumplirse en el Sprint. El equipo y el Product Owner iteran en el proceso hasta que las características planeadas están acordes con los recursos disponibles para el Sprint, quedando definido entonces el Sprint Backlog.

Una pieza final del proceso de planificación es desarrollar un Objetivo del Sprint, un propósito de negocio para el Sprint. El objetivo del Sprint puede alcanzarse aún cuando algunas características o tareas fueron demoradas o dejadas de lado. "*La razón de un Objetivo de Sprint es dar al equipo cierta flexibilidad respecto a la funcionalidad"* tal como Ken Schwaber y Mike Beedle lo enuncian en *Agile Software Development with Scrum*. El objetivo de un Sprint recuerda constantemente al equipo cuales son las tareas detalladas que deben alcanzarse. Sin este objetivo, el equipo puede enfocarse demasiado en las tareas y perder la razón por la que se están realizando. Además, mantener el objetivo en mente anima al equipo a adaptarse a condiciones que pudieran surgir durante el transcurso del Sprint.

Podría decirse que la Planificación del sprint consiste de dos sub-reuniones donde se logra:

- Determinar las funcionalidades y el objetivo
- Desglosarlas en tareas y estimar el esfuerzo

En la Figura 2.3.2 se puede observar cuáles son las entradas necesarias para realizar el pre-sprint y las salidas que esta actividad debe generar.

Figura 2.3.2. Pre–sprint o Planificación del sprint [PalacioBañeres 2008]

2.3.4.2 Sprint

Es un ciclo de producción dentro de un desarrollo iterativo e incremental, la duración estándar es de 30 días (si bien algunos aducen que el Sprint puede variar de una semana a un mes). Durante un Sprint los miembros del equipo eligen tareas a realizar, cada uno se esfuerza para alcanzar el Objetivo del Sprint y todos participan de la reunión diaria de Scrum. No hay planes elaborados durante un Sprint – se espera que el equipo use sus talentos para entregar resultados.

Algo que hace de Scrum un poderoso enfoque es que reduce la fragmentación de tiempo y el cambio constante de las prioridades al "bloquear" las características durante el Sprint. Excepto en situaciones extremas, las prioridades no se cambian durante el sprint por nadie, ni siquiera por el Product Owner, ni el administrador. Al final de Sprint, el Product Owner podría elegir descartar todas las características desarrolladas o reorientar el proyecto, pero durante un Sprint las prioridades se mantienen constantes. Esta es la parte de "control del caos controlado". Esto es crítico, porque en un entorno de constante cambio debe haber un cierto punto de estabilidad. Es necesaria cierta estabilidad para que el equipo pueda sentir que pueden lograr algo. Si aparece una circunstancia externa que cambie significativamente las prioridades, e implique por lo tanto que el equipo estaría perdiendo el tiempo si continúa trabajando, el Product Owner puede terminar el Sprint, significando que el equipo deja todo el trabajo que estuvieran haciendo y comienza una reunión de planificación de un nuevo Sprint. La cancelación de un Sprint es un costo, lo que sirve para desalentar al Product Owner a realizarlo salvo circunstancias extremas.

Hay una influencia positiva poderosa que proviene de proteger al equipo de cambios de objetivos durante el Sprint. Primero, el equipo trabaja sabiendo con absoluta certeza que el compromiso asumido no cambiará, lo que refuerza el foco del equipo en asegurar cumplir con ese objetivo. Segundo, ayuda a que el Product Owner realmente piense a conciencia los ítems que prioriza en el Product Backlog. Saber que el compromiso es para toda la duración el Sprint hace que el Product Owner sea más diligente al decidir lo que va a solicitar desde el principio.

2.3.4.2.1 SCRUM BOARD: TABLERO DE TRABAJO

Durante el Sprint es conveniente organizar un Tablero de Trabajo con las Historias ordenadas por su estado, en su forma más básica, el tablero tiene 3 columna: *Tareas a realizar, Tareas en Ejecución, Tareas Finalizadas*, pero puede (y es recomendable) agregar estados intermedios como por ejemplo *análisis, codificación y pruebas.* En la Figura 2.3.3 puede observarse un ejemplo de ScrumBoard.

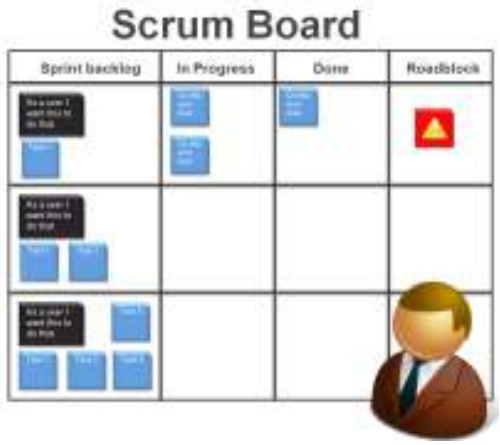


Figura 2.3.3: Ejemplo de ScrumBoard [Parasuraman 2011]

Cada tarea está representada por una tarjeta que pueda moverse por el tablero (post-it o papeles con chinches). Cada tarjeta de tarea debe contener al menos un identificador (numero único), nombre de la tarea, breve descripción y responsable (si es aplicable). En algunos casos también se usan carriles horizontales para identificar el proyecto o el responsable de la tarea.

Al inicio del Sprint todas las tareas están en la primera columna de la izquierda, es decir en estado de planificadas. Al finalizar el Sprint todas deberían estar en la última columna es decir completadas.

2.3.4.3 Reunión diaria (Scrum Diario)

Dentro de un Sprint, la reunión diaria (Daily Standup Scrum) brinda energías al propio Sprint. Existen principios muy viejos que enuncian “aumentar la comunicación” o “involucrar al cliente”. Los administradores no pueden controlar las comunicaciones o el grado de participación del cliente pero puede influenciar y crear un entorno que fomente trabajar juntos. [Highsmith]

Dado que los procesos empíricos se definen por su incertidumbre y cambio, es importante que se ejecuten chequeos diarios. La reunión diaria de Scrum permite al equipo monitorizar el estado, enfocarse en el trabajo a realizar y detectar problemas y cuestiones. Las reuniones se realizan de pie, frente al Scrum Board y a la misma hora. En promedio debe durar 15 minutos y debe ser moderada por el Scrum Master. Todos los miembros del equipo de desarrollo deben participar. También los administradores deben asistir para tener un seguimiento del estado del proyecto pero no para participar. Las reuniones se usan para detectar ciertas cuestiones y obstáculos pero no para proponer soluciones ya que eso implicaría una duración mucho mayor. Posteriormente el Scrum Master tratará de solucionar las cuestiones que generan obstáculos en el equipo. Cada participante debe comentar qué hizo, que hará y qué obstáculos tuvo en el camino para poder realizar su trabajo. Cada miembro se debe dirigir al Equipo, no al Scrum Master ni a ningún otro miembro en concreto.

2.3.4.3.1 MONITOREO DEL PROGRESO

Un Sprint terminando tardíamente es un indicador que el proyecto no está marchando correctamente. Sin embargo sería muy tarde esperar que un Sprint termine para ver que no todas las tareas se completaron.

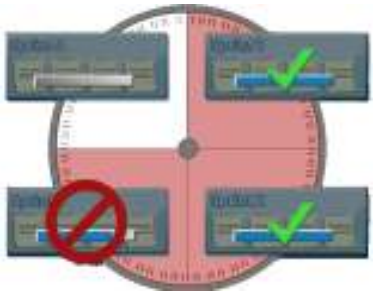


Figura 2.3.4 Progreso en un sprint [Cockburn web]

Jeff Sutherland ha refinado el monitoreo de Scrum con un razonable aumento diario de actividad administrativa – un minuto por día para los desarrolladores y 10 minutos por día para del administrador de proyectos (Figura 2.3.4). Después de la reunión de scrum diario, cada miembro del equipo actualiza la cantidad de tiempo que le falta para completar cada tarea que asumieron realizar durante el Sprint en el Sprint Backlog (ver Figura 2.3.5). Posteriormente el administrador del proyecto suma las horas restantes de trabajo que figuran en el Sprint Backlog y las refleja en una herramienta gráfica (ver Figura 2.3.6). Jeff Sutherland propone utilizar una herramienta simple pero a su vez poderosa para monitorear el progreso del proyecto: el Burndown chart del Sprint Backlog. Lo importante es que se muestre al equipo progreso hacia su objetivo, no en términos de cuanto hicieron en el pasado (un hecho irrelevante en término de progreso), sino en término de cuanto trabajo queda en el futuro –lo que separa al equipo de su objetivo.

El Burndown chart muestra en el eje horizontal los días y el trabajo restante expresado en horas en el eje vertical; al final de los 30 días, el trabajo faltante debería ser cero. Al día cero, el trabajo restante debe ser igual a la cantidad de horas estimadas para el Sprint. Cada día los desarrolladores registran los días (horas) invertidos en una tarea y su porcentaje de completitud usando alguna herramienta automática de Backlog que calcula el trabajo completado y el trabajo restante. Al mirar el gráfico de Sprint Backlog, el líder de proyecto tiene retroalimentación diaria del progreso (o falta de progreso) y cualquier estimación que resultó ser inadecuada.

				Nuevo esfuerzo estimado Lo que queda al final del día...					
Elemento de la Pila de Producto	**Tarea del Sprint**	**Voluntario**	**Esfuerzo estimado inicial**	**1**	**2**	**3**	**4**	**5**	**6**
Como comprador quiero poner un libro en el carrito de la compra	modificar base de datos	Sanjay	5	4	3	0	0	0	
	crear página web (interfaz de usuario)	Jing	3	3	3	2	0	0	
	crear página web (lógica Javascript)	Tracy & Sam	2	2	2	2	1	0	
	escribir pruebas de aceptación automáticas	Sarah	5	5	5	5	5	0	
	actualizar la página de ayuda del comprador	Sanjay & Jing	3	3	3	3	3	0	
	. . .								
Mejorar el rendimiento de procesamiento de transacciones	juntar el código DCP y completar los test del nivel de capa		5	5	5	5	5	5	
	completar la orden de máquina para pRank		3	3	8	8	8	8	
	Cambiar el DCP y el lector para usar el API http de pRank		5	5	5	5	5	5	
			...						
		Total (personas-hora)		50	49	48	44	43	34

Figura 2.3.5. Actualizaciones diarias de Trabajo Restante en la Pila del Sprint [DeemerEs]

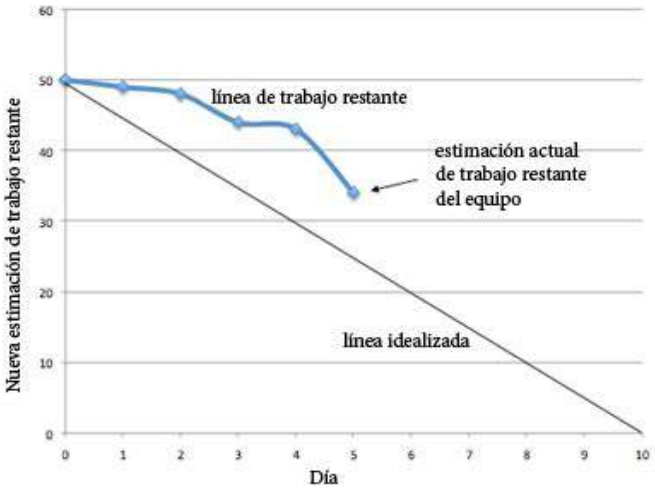


Figura 2.3.6. Gráfica de Trabajo Restante del Sprint [DeemerEs]

Idealmente el trabajo remanente siempre va disminuyendo, pero es normal (aunque no debe suceder a menudo) que el trabajo remanente aumente debido a tareas que consumen más horas de las planeadas.

Mike Beedle y Ken Schwaber señalan que "*esto <el monitoreo> no debe confundirse con reporte de tiempo que no es parte de Scrum Los equipos se miden por alcanzar objetivos no por cuantas horas toman para alcanzarlo" (Schwaber & Beedle 2002).* Los administradores están mirando la tendencia general en el gráfico del Sprint Backlog y no estimaciones y completitud de tareas de micro-administración.

2.3.4.4 Reunión Post Sprint – Revisión del Sprint

Al final de la iteración del Sprint, se realiza una reunión post-sprint para revisar el progreso, mostrar al cliente la demo de lo que se estuvo construyendo y revisar el proyecto desde una perspectiva técnica. Es una presentación del incremento desarrollado y cualquiera de los presentes es libre de realizar preguntas y hacer comentarios. Asisten el Product Owner, el Scrum Master, el Equipo y personas que podrían estar involucradas en el proyecto. El Equipo es quién muestra los avances realizados en el Sprint. Aquí se obtiene el feedback del cliente para el siguiente Sprint y se define la fecha para la reunión del siguiente sprint.

El análisis de Retrospectiva podría realizarse fuera de la reunión de Revisión realizando una reunión informal del equipo al terminar el Sprint donde se plantean los inconvenientes tenidos durante el desarrollo del Sprint, así como las mejoras posibles para trabajos futuros. Esta es una práctica que muchos equipos saltean, y esto es desafortunado ya que es una de las herramientas más importantes para hacer un Scrum exitoso. Es una oportunidad del equipo de discutir lo que está funcionando y lo que no y consensuar los cambios a probar.

2.3.4.5 Finalizando el Sprint

Uno de los principios fundamentales de Scrum es que nunca se prolonga la duración del Sprint, termina en la fecha asignada aunque el equipo no haya terminado el trabajo comprometido. Si el equipo no completó el objetivo del Sprint, deben reconocer al final del Sprint que no cumplieron lo que se habían comprometido a alcanzar. Esto crea un ciclo de retroalimentación visible y el equipo se ve forzado a mejorar en la estimación de lo que es capaz de lograr en un Sprint y entonces entregarlo sin fallar.

Si bien la duración sugerida por los autores es de 30 días, a los equipos se les recomienda que escojan una duración para el Sprint (por ejemplo dos semanas o 30 días) y que no la cambien. Una duración consistente ayuda al equipo a saber cuánto pueden hacer, lo que les ayuda en la

estimación y en la planificación de la entrega a más largo plazo. También ayuda a que el equipo adquiera un ritmo de trabajo; en Scrum a esto se le llama el "pulso" del equipo.

2.3.5 Comenzando el próximo Sprint

Después de la Revisión del Sprint, el Product Owner puede actualizar el Product Backlog con nuevas ideas. Puede presentar todas las nuevas prioridades que hayan surgido durante el Sprint e incorporarlas al Product Backlog. También puede modificar algunos ítems del Product Backlog, reordenarlos o eliminarlos. En este punto, el Product Owner y el equipo están listos para empezar otro ciclo de Sprint. No hay tiempo de descanso entre Sprints, los equipos normalmente van de la Retrospectiva del Sprint una tarde y a la Planificación del próximo Sprint la mañana siguiente (o después del fin de semana).

Uno de los principios de desarrollo ágil es "ritmo sostenible", y solo trabajando dentro de las horas delimitadas por el horario laboral a un ritmo razonable permite que los equipos continúen este ciclo indefinidamente.

2.3.6 Conclusiones

Esta metodología, SCRUM, se centra en la gestión del proyecto y puede aplicarse a otros proyectos que no tienen que ver con el desarrollo de software. Busca el trabajo cooperativo de equipos multidisciplinares altamente productivos. Define sprints de duración fija y determina una serie de reuniones a realizar a lo largo del proyecto. Como resultado de cada sprint se entrega un incremento. Plantea realizar un monitoreo del proceso y una adaptación constante. Actualmente es uno de los métodos ágiles que más está creciendo. En la Figura 2.3.7 se puede tener una visión general de esta metodología.

En el primer Scrum se usaron todas las prácticas ingenieriles de XP y sentaron las bases de la ingeniería actual. La mayoría de los equipos con alta performance usan conjuntamente SCRUM y XP. Es difícil obtener un Scrum con velocidad extrema sin las prácticas ingenieriles de XP. Por otro lado no se puede escalar XP sin Scrum. [Sutherland web]

.

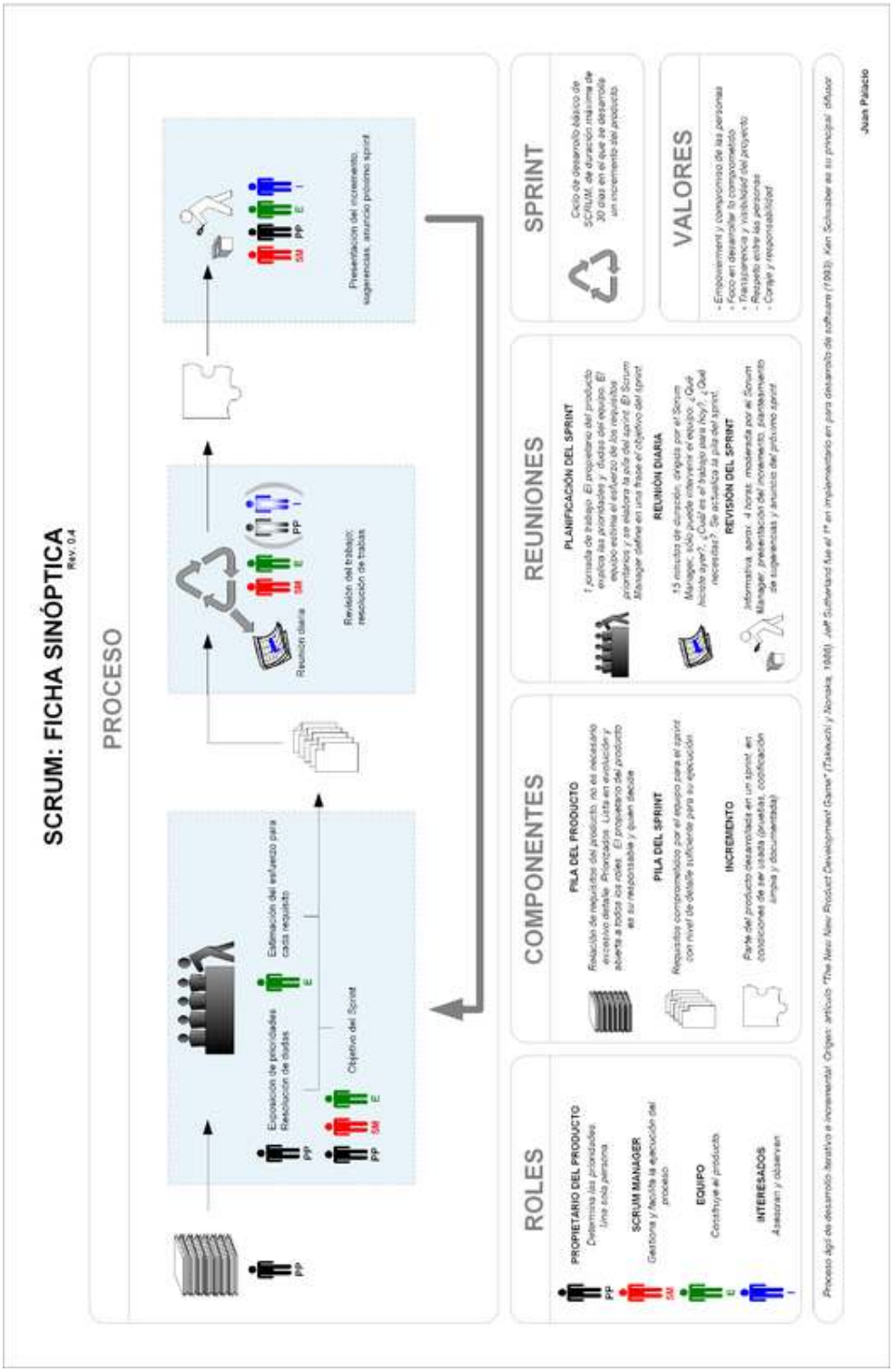


Figura 2.3.7: Scrum – Ficha sinóptica [PalacioBañeres 2008]

2.4 KANBAN

2.4.1 Introducción

Kanban se considera el enfoque de Lean al desarrollo de software [Crisp web], por lo tanto antes de introducir los conceptos de Kanban propiamente dichos se presentará brevemente Lean.

2.4.1.1 Lean

Lean está definido como un C*onjunto de Principios*, no es un proceso que pueda ser directamente adaptado a distintos ambientes. Lean Development fue iniciado por Bob Charette y se inspira en el éxito del proceso industrial Lean Manufacturing, bien conocido en la producción automotriz y en manufactura desde la década de 1980. Este proceso tiene como precepto la eliminación de residuos a través de la mejora constante, haciendo que el producto fluya a instancias del cliente para hacerlo lo más perfecto posible.

Las practicas Lean fueron implementadas y perfeccionadas por *Toyota Production System* hace más de 40 años y luego fueron copiadas por otras empresas automotrices con mucho éxito. También fueron llevadas a otras parte de la organización que no tienen mucho que ver con la producción, es allí donde estas prácticas debieron ser adaptadas.

La diferencia entre la producción y otras áreas es que en estas últimas no es posible combatir la variabilidad, sistematizar y estandarizar el trabajo como se hace en la producción. Sin embargo la adaptación de las practicas Lean en otras áreas fue tan exitosa o más que en la producción. [Modezki 2011]

Los principios que definen un sistema Lean son los siguientes [DeSeta web][Rubio 2009]:

- **Calidad perfecta a la primera** - búsqueda de cero defectos, detección y solución de los problemas en su origen. Todo lo que se hace, debe de intentar hacerse bien, no rápido, ya que cuesta más tiempo hacer algo rápido y tener que arreglarlo después, que hacerlo bien desde el principio.

- **Minimización del desperdicio** - eliminación de todas las actividades que no son de valor añadido y la optimización del uso de los recursos escasos como capital, gente y espacio. Hacer lo justo y necesario y hacerlo bien, como se decía antes en el punto anterior, y no entretenerse en otras tareas secundarias o no necesarias (principio YAGNI).

- **Mejora continua** - reducción de costos, mejora de la calidad, aumento de la productividad y compartir la información. Ir mejorando continuamente los desarrollos, según los objetivos a lograr y alcanzar.

- **Procesos "pull"** - los productos deben ser solicitados por el cliente final, no empujados al mercado desde la fábrica hacia el cliente.

- **Flexibilidad** - producir rápidamente diferentes mezclas de gran variedad de productos, sin sacrificar la eficiencia debido a volúmenes menores de producción. Lo siguiente a realizar se decide del backlog pendiente, con lo que las tareas entrantes se pueden priorizar y condicionar según las necesidades puntuales.

- **Construcción y mantenimiento de una relación a largo plazo** con los proveedores tomando acuerdos para compartir el riesgo, los costos y la información**.**

Lean es básicamente todo lo concerniente a obtener las cosas correctas en el lugar correcto, en el momento correcto, en la cantidad correcta, minimizando el desperdicio, siendo flexible y estando abierto al cambio. [Ferreiras2010]. Las prácticas Lean tienen por objetivo generar el máximo valor lo más rápidamente posible, en forma sostenida y sustentable.

"Todo lo que hacemos es mirar la línea de tiempo y reducimos el tiempo reduciendo todo aquel desperdicio que no agrega valor" Taiichi Ohno, creador de Lean y Just In Time

2.4.2 Kanban

El sistema Kanban utilizado en Toyota es la inspiración detrás de lo que se conoce como *Kanban para Ingeniería de Software*. Kanban implementa completamente los principios Lean. Mary y Tom Poppendieck publican el primer libro, por el año 2003, sobre la aplicación de los principios de Lean al desarrollo de software llamado *Lean Software Development* [Poppendieck 2003], donde se refiere a Kanban. Aunque el contexto en que es usado no es técnicamente correcto (porque se refiere al tablero, pero no habla de los límites) el término *Tablero Kanban* se deslizó dentro del vocabulario Ágil. [Joyce 2009].

En los últimos años, los sistemas Kanban se han vuelto un tema de creciente interés en la comunidad ágil, gracias particularmente a la excelente introducción que Tom y Mary Poppendieck realizaron sobre los principios propuestos por Lean aplicados al desarrollo de software [Poppendieck 2003]. Dentro de esta línea se pueden destacar tres personas que han estado trabajando con sistemas Kanban dentro del marco de desarrollo ágil: David Anderson, Arlo Belshee y Kenji Hiranabe. [Shore web]

Kanban se basa en una idea muy simple: el trabajo en curso (Work In Progress, WIP) debería limitarse, y sólo se debería empezar con algo nuevo cuando un bloque de trabajo anterior haya sido entregado o ha pasado a otra función posterior de la cadena. El Kanban (o tarjeta señalizadora) implica que se genera una señal visual para indicar que hay nuevos bloques de trabajo que pueden ser comenzados porque el trabajo en curso actual no alcanza el máximo acordado. Según David J. Anderson, Kanban parece un cambio muy pequeño pero aun así cambia todos los aspectos de una empresa. [Kniberg 2010]

El sistema Kanban es la última tendencia en el desarrollo de software de Lean, enfatizando un enfoque visual para maximizar la fluidez y detectar cuellos de botella y otros tipos de aspectos [Anderson 2010]. Uno de los conceptos claves de Kanban es todo el desarrollo debe optimizarse, evitando optimizaciones locales y esforzándose por alcanzar una optimización global.

2.4.2.1 Objetivos

- Balancear la demanda con la capacidad.
- Limitar el trabajo en proceso, mejorar el “flujo” del trabajo, descubrir los problemas tempranamente y lograr un ritmo sostenible.
- Controlar el trabajo (no la gente), coordinar y sincronizar, descubrir los cuellos de botella y tomar decisiones que generen valor.
- Equipos auto-organizados.
- Lograr una cultura de optimización incremental.

2.4.2.2 Beneficios

- Ajuste de cada proceso y flujo de valor a medida.
- Reglas simples que permiten optimizar el trabajo en función del valor que genera.
- Mejor manejo del riesgo.
- Tolerancia a la experimentación,
- Difusión “viral” a lo largo de la organización con mínima resistencia.
- Incremento de la colaboración dentro y fuera del equipo.
- Mejora de la calidad del trabajo.
- Ritmo de trabajo sostenido y sustentable.

2.4.2.3 ¿Qué es Kanban?

Kanban es una palabra que proviene del japonés. Kan significa “visual” y ban “tarjeta” o tablero. O sea, Kanban significa “tarjeta visual” o “tarjeta indicadora”, en la Figura 2.4.1 se puede observar un ejemplo de Tarjeta Kanban. Cada tarjeta representa un ítem de trabajo. El ítem de trabajo puede ser “escribir la documentación” o “agregar comentarios a una función”. La intención de la tarjeta Kanban es balancear la demanda con la capacidad y priorizar todo lo que mejore el valor del negocio. La limitación del número de tarjetas en el tablero (WIP) impide sobrecargar el sistema. [Modezki 2011]

Figura 2.4.1: Tarjeta Kanban como se usa en Toyota [Modezki 2011]

2.4.3 Sistema Kanban

Los sistemas Kanban son un enfoque para planificación del trabajo. Los proyectos ágiles típicos usan iteraciones de tiempo prefijadas o perentorias (time-boxed). Es decir, al inicio de la iteración el equipo se reúne y elige las historias del backlog que pueden entregarse al final de la misma, tal como es el caso de Scrum. Durante la iteración, desarrollan esas historias y al final de la misma la entregan.

Estos sistemas son diferentes. En vez de utilizar iteraciones de tiempo prefijado, se enfocan en el flujo continuo de trabajo. El equipo toma una historia del backlog, la desarrolla y entrega tan pronto como se finaliza. Luego toman la siguiente historia del backlog, la desarrollan y entregan. El trabajo se entrega tan pronto esté listo y el equipo solo trabaja en una historia a la vez. Este es el ideal, los enfoques varían .El principio de Kanban es que se comienza con lo que se está haciendo actualmente. Se comprende el proceso actual mediante la realización de un mapa del flujo de valor y entonces se acuerda los límites de trabajo en curso (WIP) para cada fase del proceso. A continuación se comienza a hacer fluir el trabajo a través del sistema comenzándolo ("pull") cuando se van generando las señales Kanban. [Shore web]

2.4.3.1 Flujo de valor

El flujo de valor de un determinado trabajo está compuesto por todos los procesos y subprocesos que van desde la planificación hasta la entrega. En el siguiente tablero los procesos aparecen en columnas (Ver Figura 2.4.2).

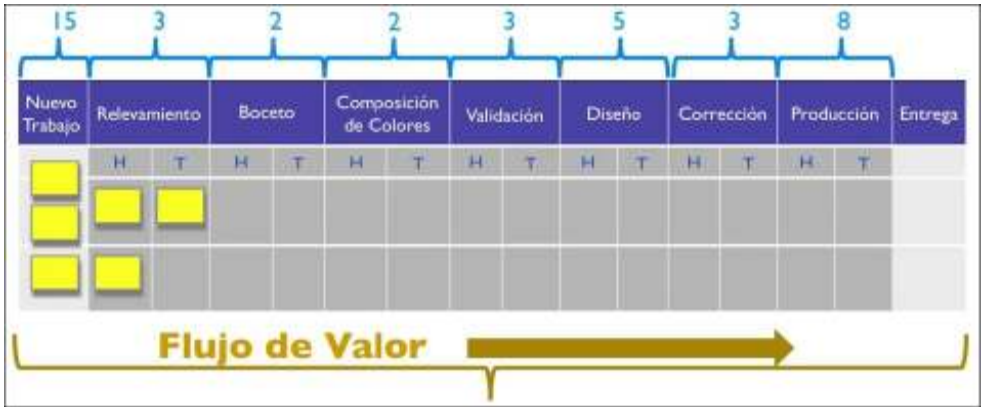


Figura 2.4.2: Flujo de Valor [Modezki 2011]

Cada proceso tiene dos sub-columnas; H para el trabajo que se está haciendo y T para los terminados. Los números que aparecen encima de cada columna son los límites a la cantidad de ítems que se puede tener en cada proceso. [Modezki 2011]

2.4.3.2 Límites

Para el sistema Kanban no solo es importante balancear demanda y capacidad sino también poner límite en cada proceso y subproceso. Empíricamente se comprueba que cuando se limita la cantidad de trabajo a la capacidad, la calidad aumenta y eso reduce los costos, aumenta la velocidad y reduce los re trabajos por errores. [Modezki 2011]. Eso es porque:

- Las cosas se hacen con más velocidad cuando menos tareas paralelas hay. Esta característica que cuando los sistemas están cargados de trabajos funciona más lentamente se demuestra prácticamente en cada trabajo que se pueda hacer.

- La multitarea genera una pérdida de foco y de tiempo al pasar de una tarea a otra. Estas pérdidas son un desperdicio.

- Al terminar más rápidamente un trabajo y pasar al siguiente proceso se tiene una retroalimentación más rápida y se descubren los problemas con mayor velocidad. Esto genera menos daños y por tanto menos costos. Los errores tardíos se acumulan y se hace más difícil descubrir dónde y por qué se generaron, incrementando los costos.

Un aspecto importante es que la cantidad asociada a Nuevo Trabajo, es decir, aquello planificado para hacer también está limitado. Esto es porque muchas veces se planifica con anticipación aquello que después se tiene que cambiar, esto conlleva un costo porque es perjudicial a la empresa. Perder tiempo y esfuerzo en una planificación que no va a ser realizada o cuyas prioridades y objetivos cambiarán también es un desperdicio. [Modezki 2011]

En el ámbito de la ingeniería de software es más práctico pensar el límite no como la cantidad de ítems, sino como la suma de *story points* o algún otro equivalente. [Joyce 2009]

2.4.3.3 Arrastrar vs Empujar

El proceso de Relevamiento en la Figura 2.4.2 tiene un límite de 3, esto es así básicamente para no generar colas que por último resultan en desperdicio.

Kanban es un sistema de Pull (o Arrastre), no de Push (o Empuje). De esta manera produce solo el trabajo estrictamente necesario y esto es igualmente válido para la planificación. La siguiente Figura intenta mostrar, de manera gráfica, la diferencia entre ambos conceptos: arrastrar y empujar)

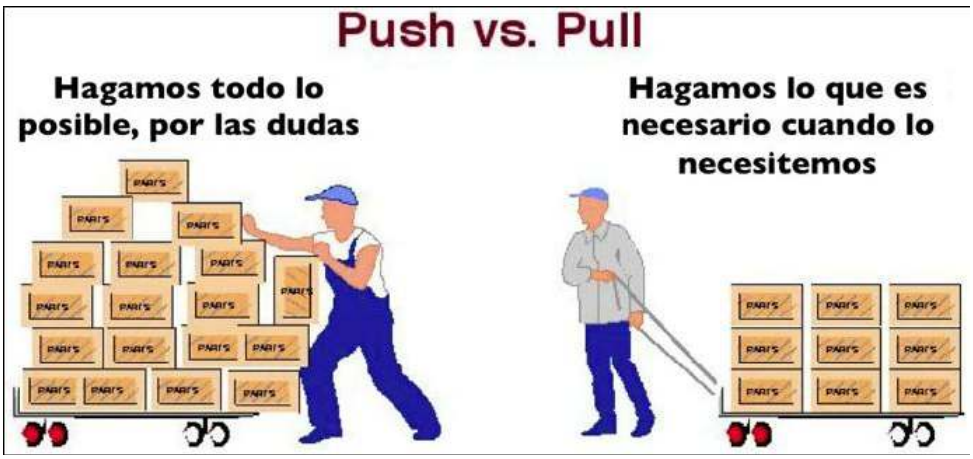


Figura 2.4.3: Empujar vs. Arrastrar [Modezki 2011]

2.4.3.4 Entrega rápida

Limitar el trabajo en curso evita los olvidos, mala información, malos entendidos, duplicación, pérdida de calidad y retrasos. El objetivo de Kanban al limitar el trabajo en curso es que cada tarea se haga lo más rápidamente posible. Los límites deben ser fijados empíricamente, con prueba y error. [Modezki 2011]

2.4.3.5 Tablero

En el desarrollo del software, el sistema Kanban se puede resumir como la visualización de las tareas mediante un tablero, en el que se van moviendo entre los sectores delimitados, con el objetivo de tener siempre presente el trabajo a realizar y lo que hace cada uno. Que nadie se quede sin trabajo y que todas las tareas importantes se realicen primero. [Rubio 2009]

El tablero Kanban, el cual debe estar visible a todo el equipo de trabajo, tiene la peculiaridad de ser un tablero continuo. Esto quiere decir que, no se rellena con tarjetas y se van desplazando hasta que toda la actividad ha quedado realizada (como pasa en Scrum), sino que a medida que se avanza, las nuevas tareas (mejoras, fallos o tareas del proyecto) se van acumulando en la sección inicial, en las reuniones periódicas con el dueño de producto (o el cliente) se priorizan las más importantes, y se encolan en las siguientes zonas [Rubio 2009]. El tablero Kanban debe contener todo lo necesario para que el equipo tome las decisiones más convenientes sin necesidad de supervisión.

- Las columnas representan los procesos, con los números que son los límites de trabajo.
- Las tarjetas blancas son productos en proceso que se deben hacer.
- Las tarjetas amarillas son tareas que deben realizarse para terminar un producto o subproducto.
- Los avatares (personas en dibujos) son los responsables de cada tarea. No es conveniente poner el nombre del responsable en cada tarjeta porque esta tarea va siendo trabajada por distintas personas en distintos procesos.
- Los marcadores verdes son trabajos completados.
- Los marcadores rojos son trabajos bloqueados.

Existen reglas de decisión (ver dentro de Figura 2.4.4 - abajo a la derecha) que indican el procedimiento para decidir cómo arrastrar el trabajo. Debajo de cada proceso está su definición de “terminado”, esta es una característica muy importante, ya que define la calidad con la que se termina una determinada tarea.

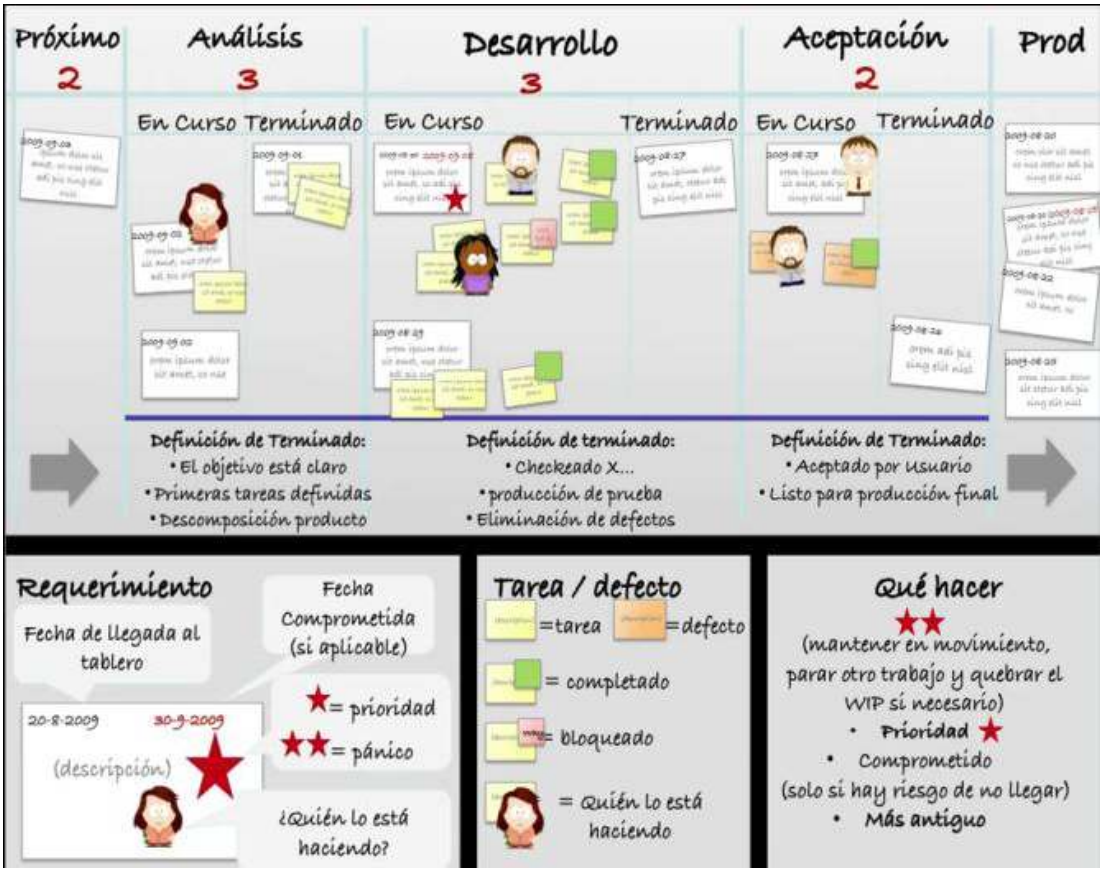


Figura 2.4.4: Tablero Kanban [Modezki 2011]

Cada tarjeta tiene: Fecha de ingreso, Fecha comprometida (si es aplicable), Descripción y Prioridad. Las tarjetas deben facilitar el sistema de pull (tirar, arrastrar) y alentar la toma de decisiones autónoma del equipo. [Modezki 2011]. Las tarjetas de Kanban suelen tener bastante más información, ya que el método consiste en tener todo visual, para saber de forma rápida la carga total de trabajo, ya sea de los grupos, como del departamento, etc. Es importante destacar que las tarjetas deben de tener la estimación de tiempo que tiene asignada la tarea, así como se pueden anotar las fechas de entrada en cada cuadrante, para tener información, al término de la tarea, de si ha sido una buena estimación, así como obtener el rendimiento del equipo de trabajo.

El sistema Kanban, no se dedica a llevar información de un solo proyecto, sino que se pueden entremezclar tareas y proyectos. El método se basa en tener a los trabajadores con un flujo de trabajo constante, las tareas más importantes en cola para ser realizadas y un seguimiento pasivo, a modo de no tener que interrumpir al trabajador para saber qué hace en cada momento. El sistema Kanban tiene ciertas ventajas con relación a otras metodologías, ya que permite no solo llevar el seguimiento de un proyecto de forma individual, sino también de las incidencias que se van sucediendo, así como otros proyectos paralelos que tenga que hacer el mismo equipo de desarrollo, por lo que se puede deducir es que no se lleva pista de los proyectos, sino de los equipos de trabajo.

2.4.3.6 Indicadores

Kanban provee indicadores muy claros de la salud del proyecto. El límite de trabajo es fácil de controlar, ayuda a decidir si se debe comenzar o no un nuevo trabajo. Es muy importante indicar la cantidad y fecha en que se fijó el límite. Regular los límites reporta los problemas en los procesos, mientras estos están sucediendo. Las colas acumulando ítems están seguidas por un proceso bloqueado. Las colas no permiten dobles interpretaciones.

Las consecuencias son altamente predecibles. Antes que cualquier parte colapse el sistema completo responde bajando la velocidad. Entonces se liberan recursos que deben acudir a responder el problema. [Joyce 2009]

2.4.3.7 Costos

Lean identifica 7 tipos de desperdicio en la manufacturación. La metáfora de "desperdicio" no es muy útil en la Ingeniería de Software. El "desperdicio" en las actividades puede ser una forma de descubrir información, y por lo tanto no siempre es un desperdicio. El desperdicio se trata más que nada del *costo de retraso*: *maximizar el valor* y *disminuir los riesgos*. En la ingeniería de software el costo de retraso (desperdicio) proviene de 3 tipos [Joyce 2009]:

- Re-trabajo – Defectos, fallas en la demanda
- Costos de Transacción – Planeación, liberación, entrenamiento
- Costos de coordinación – Planificación, recursos

2.4.3.8 Funcionamiento

Ya en funcionamiento la limitación del trabajo en curso debería permitir lograr un ritmo de trabajo suave, constante, armónico y sostenido. Ese debe ser el foco para lograr todos los beneficios que el sistema Kanban provee.

Uno de los aspectos que se debe evitar es la formación de colas o cuellos de botella, no solo porque interrumpe el flujo de trabajo sino porque son una fuente enorme de defectos, errores y olvidos que se deben evitar. Puede producirse debido a problema de definición de límites, problemas de capacidad o una oportunidad de mejorar algunos procesos. En caso de producirse los cuellos de botella, lo ideal es usar los recursos libres para liberar las colas y luego mejorar los procesos. De esta manera sucede una mejora incremental del flujo de valor.

Otra situación indeseable es la hambruna, una persona tiene todas sus tareas todas terminadas pero no puede continuar porque no tiene de dónde "arrastrar", ya que la persona encargada de realizar la tarea anterior no tiene ninguna tarea finalizada. Nuevamente puede ser problema de definición de límites, problemas de capacidad o una oportunidad de mejorar algunos procesos.

Una tercera situación se presenta si un recurso dentro de un proceso queda liberado y tiene un compañero que no finalizó su tarea. La persona liberada podría arrastrar un trabajo de la cola de tareas o bien ayudar a su compañero para que el trabajo en proceso sea terminado lo más rápido posible. La regla es que el recurso libre debe ayudar a su compañero a terminar el trabajo siempre que esto sea posible. Una vez que ambos terminen arrastran 2 trabajos de la cola. Esto es otra muestra de colaboración y coordinación autónoma, sin necesidad de consultar a un supervisor.

Por todo esto, si una persona terminó su trabajo, lo primero que debe tratar de hacer es ayudar a un compañero a terminar su trabajo. En caso de no poder ayudarlo debería buscar un cuello de botella y liberarlo. Y si no es posible liberar un cuello de botella, debería arrastrar un trabajo de la cola si es posible. En caso de no ser posible, buscaría otra cosa interesante para trabajar.

Kanban muestra que la utilización máxima de los recursos no es su objetivo, sino por el contrario, si todos trabajaran al máximo no podría haber mejoras. Los únicos lugares que hacen máxima utilización de recursos son los cuellos de botella, que por definición debería ser uno solo. [Modezki 2011]

2.4.3.9 Políticas y clases

Las Clases de Servicio proveen de una manera conveniente de clasificar el trabajo para ofrecer al cliente más flexibilidad mientras se optimiza el resultado económico. Se definen en función del impacto en el negocio, o sea la generación de valor. Para entender el impacto en el negocio debe entenderse como juegan en combinación distintos factores entre ellos costos fijos, demanda y riesgo.

Las entregas, implementaciones y prioridades varían entre diferentes clases. La clasificación resultará en *Niveles de servicio* específicos a cada clase. [Modezki 2011]. Los trabajos deben fluir por el sistema *optimizando el riesgo*. El resultado debe ser una liberación optimizada en riesgo, o sea *máximo valor para el negocio* y *mínimo costo de retraso*. Se debe facultar a todos a tomar decisiones priorizando los lineamientos con los riesgos, o sea, decidir cualquier día, sin ninguna intervención gerencial o supervisión. [Joyce 2009]

Por otro lado, las Políticas son reglas asociadas a cada Clase de Servicio, que indican qué "arrastrar" primero y qué hacer cuando debemos tomar una decisión. [Modezki 2011]. Las Políticas pueden incluir: Prioridad de arrastre, Límites, Restricciones de Tiempo y Riesgo, Color de la tarjeta y Anotaciones complementarias. Se recomienda que las Clases de Servicio no excedan más de 6. Esto busca evitar hacer el sistema complejo. [Joyce 2009]

2.4.3.10Coordinación y sincronización

Lo conveniente es organizar una reunión al comienzo de cada día. En esta reunión la gente está parada y no debe durar más de 15 minutos. El objetivo de la reunión diaria es analizar el flujo de trabajo en el tablero desde la derecha (el final) hacia la izquierda. El foco es siempre terminar el trabajo en curso. Se analiza el flujo de trabajo, como se comporta y si se están formando colas o cuellos de botella para saber por qué esto está sucediendo y como resolverlo. Todos pueden entender que es lo que pasa visualmente. [Modezki 2011]

Los proyectos Kanban no tienen problemas en escalarse hasta 40 integrantes o más. La diferencia de las reuniones Kanban es que existe un facilitador que enumera el trabajo, no las personas. Durante la reunión todos los miembros pueden modificar la pizarra, que debe mantenerse permanentemente actualizada. Si hubiera problemas puntuales, existe una segunda reunión espontánea solo con las personas involucradas con el fin de introducir mejoras y evitar que los problemas se repitan. [Joyce 2009]

2.4.3.11Control

Kanban combina la flexibilidad de la producción artesanal con el control de flujo de una cañería.

- El trabajo en proceso es limitado
- El tiempo de ciclo es manejado
- Los procesos son altamente transparentes y repetibles
- Están dadas todas las condiciones necesarias para la mejora continua

2.4.4Conclusiones

Kanban es un sistema muy simple que se basa principalmente en: Visualización del flujo de trabajo, Limitación del trabajo en proceso, Medición y Gestión del Flujo y Mejora incremental. Tal como lo mencionan los autores de esta metodología, son justamente los métodos simples los que generan los menores trastornos y generalmente los más sustentables beneficios.

Esta metodología puede verse como un sistema transparente y visual de limitación del trabajo en curso y arrastre (pull) del valor. Ayuda a entregar valor promoviendo el flujo y exponiendo los cuellos de botella, las colas, el impacto de los defectos, los costos económicos y por lo tantos el desperdicio.

Kanban, tal como uno de sus mayores exponentes David Anderson afirma, ha demostrado ser útil en equipos que realizan desarrollo Ágil de software, pero también están ganando fuerza en equipos que utilizan métodos más tradicionales. Kanban se está introduciendo como parte de iniciativas Lean para transformar la cultura de las organizaciones y fomentar la mejora continua. [Kniberg 2010]

Kanban dentro de un proyecto o en un área tiene que ser adaptado a su contexto y a sus características específicas.

2.5 OPENUP

2.5.1 ¿Qué es Open UP?

Open Up es un proceso unificado ágil y liviano, que aplica un enfoque iterativo e incremental dentro de un ciclo de vida estructurado y contiene un conjunto mínimo de prácticas que ayuda al equipo a ser más efectivo desarrollando software. Open Up abraza una filosofía pragmática y ágil de desarrollo, que se enfoca en la naturaleza colaborativa del desarrollo de software. Se trata de un proceso con herramientas neutrales y de baja ceremonia que puede extenderse para alcanzar una amplia variedad de tipos de proyectos. [OPENUP web]

Es un modelo de desarrollo de software, es parte del Framework de modelo de proceso de Eclipse (Eclipse Process Framework), desarrollado por la fundación Eclipse. Mantiene las características esenciales de Rational Unified Process (RUP), en el cual se incluyen las siguientes características:

- Desarrollo incremental.
- Uso de casos de uso y escenarios.
- Manejo de riesgos.
- Diseño basado en la arquitectura.

OpenUp es un marco de trabajo para procesos de desarrollo de software. Fue propuesto por un conjunto de empresas de tecnología (IBM Corp., Telelogic AB., Armstrong Process Group, Inc., Number Six Software, Inc. & Xansa plc.) lo donaron en el año 2007 a la Fundación Eclipse. La fundación lo ha publicado bajo una licencia libre y lo mantiene como método de ejemplo dentro del proyecto *Eclipse Process Framework*.

Es mínimamente suficiente, es decir, que solamente incluye contenido fundamental. No provee guías en muchos aspectos que los proyectos podrían enfrentar como por ejemplo: equipos de desarrollo de gran tamaño, situaciones contractuales, aplicaciones críticas o de salud, guías específicas sobre tecnología, etc. Sin embargo, Open UP es completo en el sentido que puede verse como un proceso entero para construir un sistema.

Su proceso puede ser personalizado y extendido para distintas necesidades, que aparecen a lo largo del ciclo de vida del desarrollo de software, dado que su modelo de desarrollo es iterativo incremental, es capaz de producir versiones. Además, una de sus mayores ventajas es que puede ser acoplado para proyectos pequeños, dado que en su gráfica de roles aparecen 4 personas, que pueden trabajar bien manejando esta metodología.

Como mantiene las bases de RUP, aún maneja procesos tan importantes como *Manejo de Riesgos*, que es una parte del desarrollo de software que no se puede descuidar. Una desventaja es que se puede utilizar este modelo sin tanto formalismo y se puede caer en el desorden y perder la trazabilidad del proyecto. [OPENUP web]

Pueden distinguirse tres Niveles o Capas en OpenUP, tal como se aprecia en la Figura 2.5.1: Micro-incrementos, ciclo de vida de la iteración y ciclo de vida del proyecto.

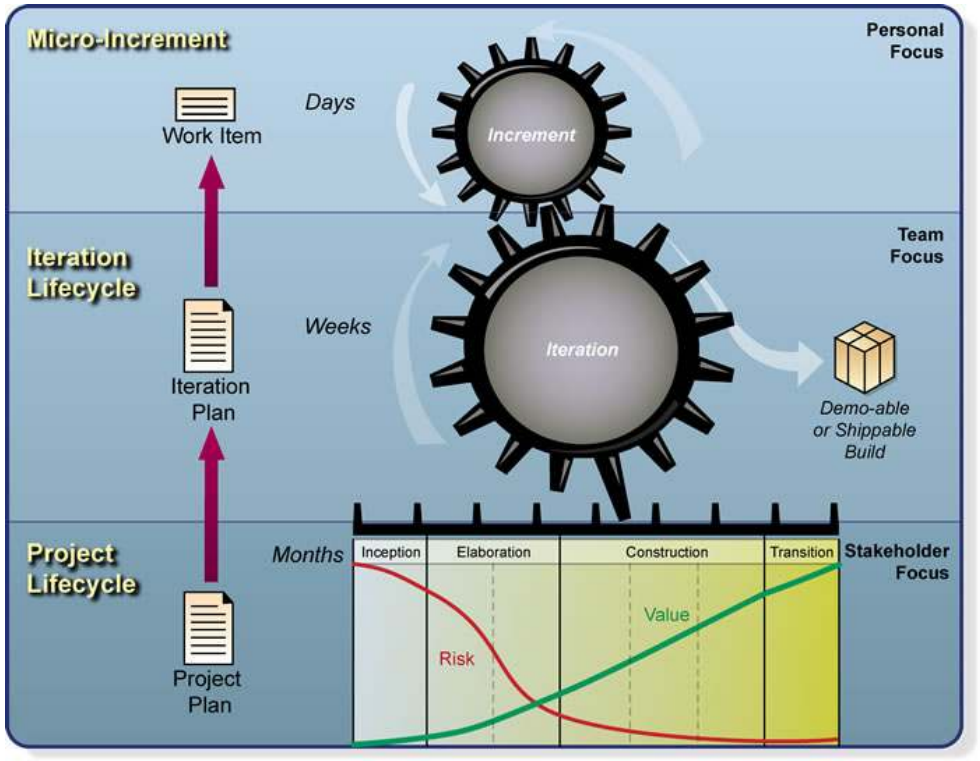


Figura 2.5.1: Capas de OpenUP: micro-incrementos, ciclo de vida de la iteración y del proyecto [OPENUP web]

El esfuerzo personal en un proyecto OpenUP se organiza en **microincrementos**. Estos representan unidades cortas de trabajo que producen un ritmo estable y medible de progreso del proyecto (generalmente medido en horas o unos pocos días). Este proceso aplica colaboración intensiva a medida que el sistema se va desarrollando incrementalmente por un equipo comprometido y auto-organizado. Estos microincrementos proveen un ciclo de retroalimentación extremadamente corto que conduce a decisiones de adaptación con cada iteración.

OpenUP divide el proyecto en iteraciones: intervalos de tiempo prefijados y planeados, generalmente expresados en semanas. Las iteraciones hacen que el equipo se focalice en entregar valor incremental a los stakeholders de una forma predecible. En el plan de la iteración se define lo que debe entregarse dentro de la iteración y el resultado es algo que puede mostrarse en una demo o entregarse. Los equipos OpenUP se auto-organizan según cómo lograr los objetivos de la iteración y se comprometen a entregar los resultados. Logran ésto definiendo y extrayendo tareas de granularidad fina de una lista de ítems. OpenUP aplica un **ciclo de vida iterativo** que estructura cómo se aplican los microincrementos para entregar porciones de desarrollo estables y unidas del sistema que progresa incrementalmente hacia los objetivos de la iteración.

Estructura el ciclo de vida del **proyecto** en cuatro fases: Inicio, Elaboración, Construcción y Transición. Cada fase puede tener tantas iteraciones como sean necesarias. El ciclo de vida del proyecto provee visibilidad y puntos de decisión a través del proyecto a stakeholders y miembros del equipo. Esto brinda una visión efectiva y permite tomar decisiones de continuar o no en los momentos apropiados. Un plan de proyecto define el ciclo de vida y el resultado final es una aplicación entregada.

2.5.2 Visión general de OpenUP

OpenUP es para equipos pequeños que trabajan juntos en la misma ubicación. Los equipos necesitan involucrarse en una interacción plena cara a cara diariamente. Los miembros del equipo incluyen a los stakeholders, desarrolladores, arquitectos, administradores de proyecto y

testers. Ellos toman sus propias decisiones sobre en qué necesitan trabajar, cuáles son las prioridades y cómo alcanzar las necesidades del stakeholder de la mejor manera.

Los miembros del equipo trabajan colaborativamente. La presencia de los stakeholders como miembros del equipo es crítica para la implementación exitosa de OpenUP. Además, los miembros del equipo participan en daily stand-up meetings para comunicar los estados y situaciones. Los problemas se resuelven fuera de estas reuniones, similar a Scrum.

OpenUP se enfoca en reducir el riesgo de manera temprana en el ciclo de desarrollo. Esto requiere de reuniones regulares para revisar los riesgos y una implementación rigurosa de estrategias de mitigación.

Todo el trabajo se lista, rastrea y asigna a través de la lista de ítems de trabajo (Work Items List). Los miembros del equipo usan este simple repositorio para todas las tareas que necesitan registrarse y rastrearse. Esto incluye todas las solicitudes de cambio, bugs y solicitudes del stakeholder.

Los casos de usos se usan para elicitar y describir los requerimientos. Si bien los describen los desarrolladores, son los Stakeholders responsables de revisar y certificar que los requerimientos sean correctos. Los casos de uso se desarrollan de manera colaborativa.

Los requerimientos importantes arquitectónicamente deben identificarse y estabilizarse en la fase de Elaboración de tal manera que se pueda crear una arquitectura robusta como el corazón del sistema. Un cambio en un requerimiento importante desde el punto de vista de la arquitectura que deba realizarse, puede traer retrasos en el desarrollo, pero, el riesgo de que esto suceda se reduce significativamente en la fase de elaboración.

Las pruebas (Testing) se realizan muchas veces en cada iteración, cada vez que se incrementa la solución con el desarrollo de un requerimiento, cambio o arreglo de un bug. Estas pruebas ocurren luego de que los desarrolladores han desarrollado la solución (a la que debe haberse realizado la prueba de unidad) y la integren al código base. Estas pruebas ayudan a garantizar que se está creando una solución estable y siempre está disponible a medida que progresa el desarrollo.

2.5.3 Principios OpenUP

OpenUP está gobernada por cuatro principios fundamentales

- *Balancear las prioridades involucradas para maximizar el valor para el stakeholder*

Fomentar prácticas que permitan a los participantes del proyecto y los stakeholders desarrollar una solución que maximice los beneficios de los stakeholders y sea compatible con restricciones impuestas en el proyecto (de costes, plazos, recursos, normas, etc).

- *Colaborar para alinear los intereses y compartir entendimiento*

Promover las prácticas que fomenten un ambiente de equipo saludable y que permitan la colaboración y desarrollo de un entendimiento compartido del proyecto.

- *Enfocarse en la arquitectura tempranamente para minimizar riesgos y organizar el desarrollo*

Promover prácticas que permitan al equipo enfocarse en la arquitectura para minimizar los riesgos y organizar el desarrollo

- *Evolucionar para obtener constantemente retroalimentación y mejorar*

Promover prácticas que permitan al equipo obtener retroalimentación temprana y continua de los stakeholders, y demostrarles el valor incremental.

Cada principio de OpenUP apoya una declaración del Manifesto Agil, como se ve en la siguiente tabla [Baudino]

Principio de OpenUP	**Declaración del Manifesto Agil**
Colaborar para alinear los intereses y compartir entendimiento	*Al individuo y a las interacciones del equipo de desarrollo por encima del proceso y las herramientas*
Balancear las prioridades involucradas para maximizar el valor para el stakeholder	*Colaboración con el cliente por sobre la negociación contractual*
Enfocarse en la arquitectura tempranamente para minimizar riesgos y organizar el desarrollo	*Software que funcione por sobre una documentación exhaustiva*
Evolucionar para obtener constantemente retroalimentación y mejorar	*Respuesta al cambio por sobre el seguimiento de un plan*.

Tabla 2.5.1: Mapping between OpenUP principles and Agile Manifesto [Baudino]

2.5.4 Roles

OpenUp describe un conjunto mínimo de seis roles que deben cubrirse al realizar un desarrollo. Ver Figura 2.5.2.

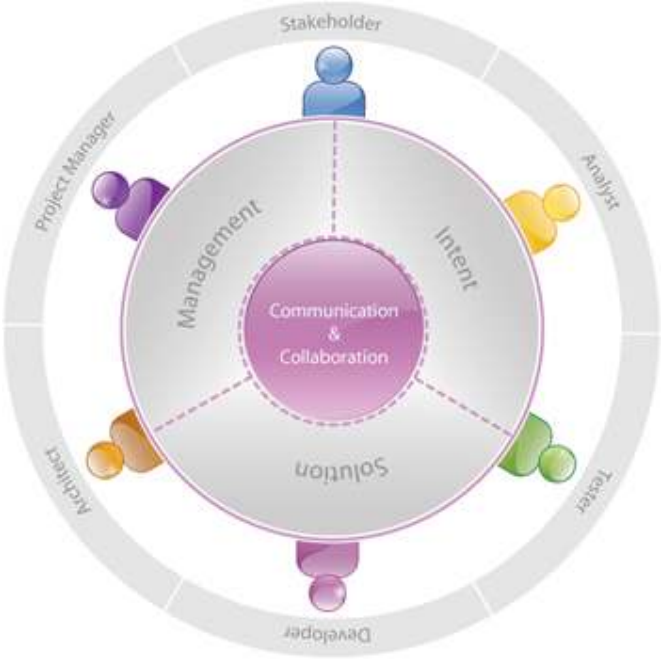


Figura 2.5.2: Roles principales en OpenUp y su interacción [OPENUP web]

- **Administrador del proyecto (Project Manager):** lidera la planificación del proyecto, coordina las interacciones con los stakeholders, y mantiene el equipo enfocado en alcanzar los objetivos del proyecto.

- **Analista (Analist):** representa las preocupaciones de los clientes y usuarios finales mediante la recopilación de aportes de los stakeholders para entender el problema a resolver y mediante la captura y establecimiento de prioridades para los requisitos.

- **Arquitecto (Arquitect):** es responsable de definir la arquitectura de software, que incluye la toma de decisiones técnicas claves que limitan el diseño y la implementación del sistema.

- **Tester:** es el responsable de las actividades principales del esfuerzo de la prueba. Esas actividades incluyen la identificación, definición, implementación y realización de las pruebas necesarias, así como del registro de los resultados de las pruebas y el análisis de los resultados.

- **Stakeholders:** representa a grupos cuyas necesidades deben ser satisfechas por el proyecto. Es un papel que puede jugar por cualquier persona que es (o será potencialmente) afectados por el resultado del proyecto. Son aquellos que se verán afectados directamente con el resultado del proyecto.

- **Desarrollador (Developer):** es responsable de desarrollar una parte del sistema, incluyendo diseñarlo de tal forma que se ajuste a la arquitectura, posiblemente crear prototipos de la interfaz de usuario y luego implementar pruebas de unidad, e integrar los componentes que forman parte de la solución.

2.5.5 Disciplinas OpenUP

Así como Proceso Unificado plantea una serie de disciplinas para ordenar los flujos de trabajo (Ver Figura 2.5.3), OpenUP plantea seis disciplinas para agrupar las tareas involucradas en el desarrollo.

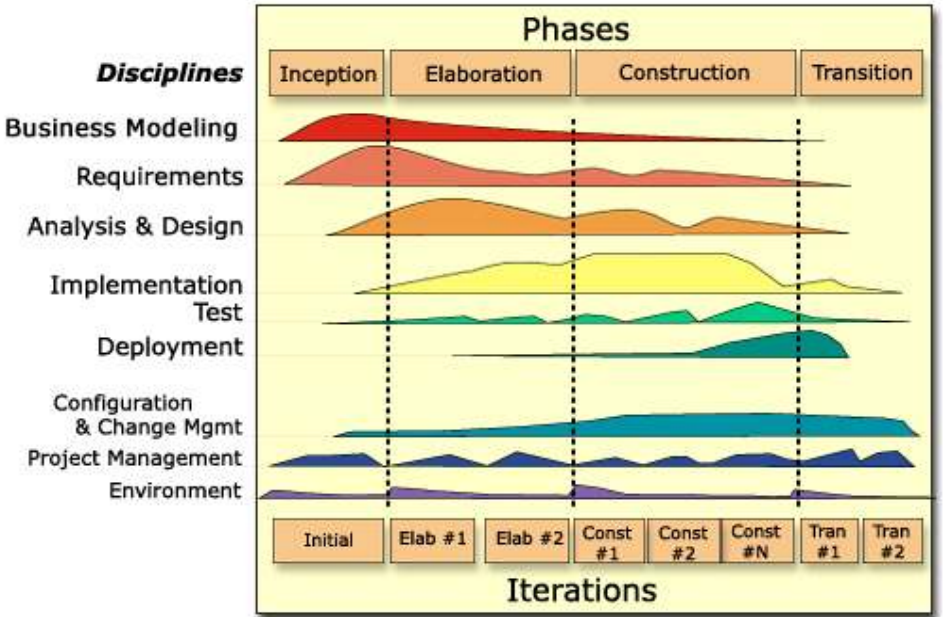


Figura 2.5.3: Disciplinas de Proceso Unificado [Baudino]

OpenUP se enfoca en las siguientes disciplinas: Requerimientos, Arquitectura, Desarrollo, Prueba, Administración de Configuración y Cambio y Administración de Proyecto. [Baudino]

- **Requierimientos (Requirements)**: explica cómo elicitar, analizar, validar y manejar los requerimientos para el sistema a desarrollar. Es importante entender la definición y alcance del problema que se intenta resolver, identificar los stakeholders y definir el problema a resolver. Durante el ciclo de vida se administran los cambios de los requerimientos.

- **Arquitectura (Arquitecture):** El propósito es lograr evolucionar a una arquitectura robusta para el sistema. Explica cómo crear una arquitectura a partir de requerimientos arquitectónicamente (estructuralmente) importantes. Es en la Disciplina de desarrollo donde se construye la arquitectura.

- **Desarrollo (Development):** explica cómo diseñar e implementar una solución técnica que esté conforme a la arquitectura y sustente los requerimientos

- **Prueba (Test):** explica cómo proveer retroalimentación sobre la madurez del sistema diseñando, implementando, ejecutando y evaluando pruebas. Es iterativa e incremental. Aplica la estrategia de "prueba lo antes posible y con la mayor frecuencia posible" para replegar los riesgos tan pronto como sea posible dentro del ciclo de vida del proyecto.

- **Administración de configuración y cambio (Configuration and change management):** explica cómo controlar los cambios de los artefactos, asegurando una evolución sincronizada del conjunto de productos (Work Products) que compone un sistema software. Esta disciplina se expande durante todo el ciclo de vida

- **Administración del Proyecto (Project Management):** explica cómo entrenar, ayudar y apoyar al equipo, ayudándolo a manejar los riesgos y obstáculos que se encuentren al construir el software.

En OpenUP se omitieron otras disciplinas y áreas de incumbencia, como por ejemplo el modelado de negocio, entorno, administración avanzada de requerimientos y administración de la configuración. Estos aspectos se consideran innecesarios para proyectos pequeños o bien se manejan dentro de otras áreas de la organización, fuera del equipo del proyecto [Baudino]

2.5.5.1 Tareas

Una tarea es una unidad de trabajo que se puede solicitar que lo realice un rol de trabajo. En OpenUP hay 19 tareas que los roles pueden realizar como actor principal (teniendo la responsabilidad de ejecutar esas tareas) o adicionales (ayudando y proveyendo información que se usa al ejecutar la tarea). La naturaleza colaborativa de OpenUP se ve manifiesta al tener los actores principales de las tareas interactuando con otros individuos al realizar las tareas. Las tareas se enumeran en la siguiente tabla.

Disciplina	**Tarea**
Arquitectura	• Refinar la arquitectura • Visualizar (Preveer) la arquitectura
Desarrollo	• Implementar las pruebas de desarrollo • Implementar la solución • Correr las pruebas de desarrollo • Integrar y crear el desarrollo • Diseñar la solución
Administración de Proyecto	• Evaluar resultados • Administar la iteración • Planear la iteración • Planear el proyecto • Solicitar cambios
Requerimientos	• Identificar y resaltar requerimientos • Detallar Escenarios de Caso de Uso • Detallar requerimientos generales del sistema • Desarrollar una visión técnica
Prueba	• Crear Casos de Prueba • Implementar pruebas • Correr pruebas

Tabla 2.5.2: Tareas de OpenUP (armado con información de [OPENUP web])

2.5.6 Artefactos

Un artefacto (work product) es algo producido, modificado o utilizado por una tarea. Los Roles son responsables de crear y actualizar los artefactos. Los artefactos están sujetos a control de versión a través del ciclo de vida del proyecto. Los 17 artefactos de OpenUP se consideran esenciales y son los que debe utilizar un proyecto para capturar información relacionada con el producto y el proyecto. No hay obligación de capturar la información en artefactos formales. Se puede capturar la información informalmente en pizarras (por ejemplo para el diseño y la arquitectura), notas de reuniones (por ejemplo para la evaluación del estado), etc. Los proyectos pueden utilizar los artefactos de OpenUP o reemplazarlos con los propios. En la siguiente tabla se describe brevemente cada uno de los artefactos.

Disciplina	**Artefacto**	**Tipo de Producto**	**Descripción General**
Arquitectura	Architecture Notebook	Especificación	Describe el contexto para el desarrollo del software. Contiene las decisiones, razones, supuestos, explicaciones e implicancias de armar la arquitectura
Desarrollo	Diseño	Solución	Describe la realización de la funcionalidad del sistema requerida en términos de componentes y sirve como una abstracción del código fuente.
Desarrollo	Construcción	Solución	Versión operacional de un sistema o parte del sistema que muestra un subconjunto de las funcionalidades que se proveerán en el producto final.
Desarrollo	Prueba	Solución	Instrucciones que validan que los componentes individuales del software se comportan como se especificó.
Desarrollo	Implementación	Solución	Archivos de código software, archivos de datos y archivos secundarios como archivos de ayuda en línea que representan las partes gruesas de un sistema.
Administración de Proyecto	Plan de iteración	Plan	Un plan con gran nivel de detalle describiendo los objetivos, asignaciones de trabajo y criterios de valuación para la iteración
Administración de Proyecto	Plan de Proyecto	Plan	Reúne toda la información requerida para administrar el proyecto. Sus partes principales están compuestas por un plan poco detallado que contiene las fases del proyecto y los hitos
Administración de Proyecto	Lista de ítems de trabajo	Datos del proyecto	Contiene una lista de todo el trabajo calendarizado a realizarse dentro del proyecto, así como el trabajo propuesto que puede afectar al producto en este o futuros proyectos. Cada ítem de trabajo puede contener referencias a información relevante para llevar a cabo el trabajo descripto dentro de él.

Administración de Proyecto	Lista de riesgo	Datos del proyecto	Lista abierta de riesgos conocidos, ordenados según importancia y asociados con acciones específicas de mitigación o contingencia.
Requerimientos	Especificación de requerimientos secundario	Especificación	Captura los requerimientos globales del sistema no capturados en escenarios o casos de uso, incluyendo requerimientos de calidad y requerimientos de funcionalidad global.
Requerimientos	Visión	Concepto	Contiene la definición de la vista que tienen los stakeholders sobre el producto a desarrollar, especificado en términos de necesidades y características claves de los stakeholders. Contiene una idea general del corazón de requerimientos imaginados para el sistema
Requerimientos	Caso de Uso	Especificación	Captura la secuencia de acciones que un sistema puede realizar que produce un resultado observable de valor para los que interactúan con el sistema.
Requerimientos	Glosario	Elemento de modelo, Especificación	Define los términos importantes usados en el proyecto. Estos términos son la base para la colaboración efectiva con los stakeholders y otros miembros del equipo.
Requerimientos	Modelo de Caso de Uso	Modelo	Captura un modelo de las funcionalidades pretendidas en el sistema y su entorno, y sirve como un contrato entre el cliente y los desarrolladores.
Pruebas	Caso de Prueba	Especificación	Es la especificación de un conjunto de entradas de prueba, condiciones de ejecución y resultados esperados, identificados con el propósito de realizar la evaluación de algún aspecto particular de un escenario
Pruebas	Log de Prueba	Dato de Proyecto	Recolecta salidas sin refinar capturadas durante una única ejecución de una o más pruebas para una sola corrida del ciclo de prueba.
Pruebas	Script de Prueba	Solución	Contiene las instrucciones paso a paso que realiza una prueba, permitiendo su ejecución. Estas pueden tomar forma ya sea de instrucciones textuales documentadas que se ejecutan manualmente o bien instrucciones que una computadora puede entender y permite la ejecución automática de la prueba.

Tabla 2.5.3: Artefactos de OpenUP (armado con información de [OPENUP web])

2.5.7 Ciclo de vida de la iteración

El ciclo de vida de la iteración provee un conjunto de prácticas basadas en el equipo que describen cómo utilizar las iteraciones para enfocar al equipo en la entrega de valor incremental a los stakeholders de una forma predecible. Las iteraciones en OpenUP mantienen al equipo enfocado en brindar al cliente valor incremental cada pocas semanas entregando un incremento de producto completamente probado que puede ser mostrado o entregado al cliente. Esto crea un enfoque saludable de asegurarse de estar trabajando en algo que agregue valor a los stakeholders. La toma de decisiones debe realizarse rápidamente ya que no hay tiempo para debates interminables. El desarrollo iterativo se enfoca en producir código que funcione reduciendo el riesgo de la fase de "parálisis-análisis" [OPENUP web]. La demostración frecuente de código funcionando provee mecanismos de retroalimentación que permiten realizar las correcciones en curso según sean necesarias.

La planificación, estimación y rastreo de progreso en la iteración está centrada en los ítems de trabajo (work ítems). Se crea el plan de la iteración seleccionando los ítems de trabajo con mayor prioridad. Se usan técnicas ágiles de estimación para comprender cuantos ítems de trabajo pueden completarse en el tiempo prefijado de la iteración, y se filtran los work ítems para asegurar que los ítems de trabajo elegidos pemitirán al equipo cumplir con los objetivos de la iteración expresados por los stakeholders. El progreso se demuestra a través de la cantidad de ítems de trabajo que se completan.

Así como el proyecto tiene un ciclo de vida, las iteraciones tienen su ciclo de vida. Este ciclo de vida de las iteraciones tiene un enfoque diferente para los equipos dependiendo si es la primera o la última semana de la iteración, ver Figura 2.5.4. Una iteración comienza con una reunión de planificación de unas pocas horas de duración. Los primeros días el enfoque está puesto en seguir planificando y definir la arquitectura, entre otras cosas, para entender las dependencias y orden lógico de los ítems de trabajo y de los impactos en la arquitectura del trabajo a realizar. La mayor parte del tiempo en una iteración se emplea para ejecutar los microincrementos. Cada microincremento debe entregar código probado al incremento así como artefactos validados. Para dar mayor disciplina, al final de cada semana se produce un incremento estable. Se emplea mayor atención en ese incremento para asegurar que no se está socavando la calidad y problemas son resueltos tempranamente de tal forma que no peligra el éxito de la iteración. La última semana o últimos días de la iteración, generalmente tienen un énfasis mayor en pulir y arreglar errores que en semanas anteriores, aunque se agregan nuevas características según sea apropiado. El objetivo es nunca socavar la calidad, y así producir un incremento de producto útil de alta calidad al final de la iteración. A iteración termina con una valoración (con los stakeholders) de lo que se construyó, y una retrospectiva para entender cómo mejorar el proceso para la siguiente iteración.

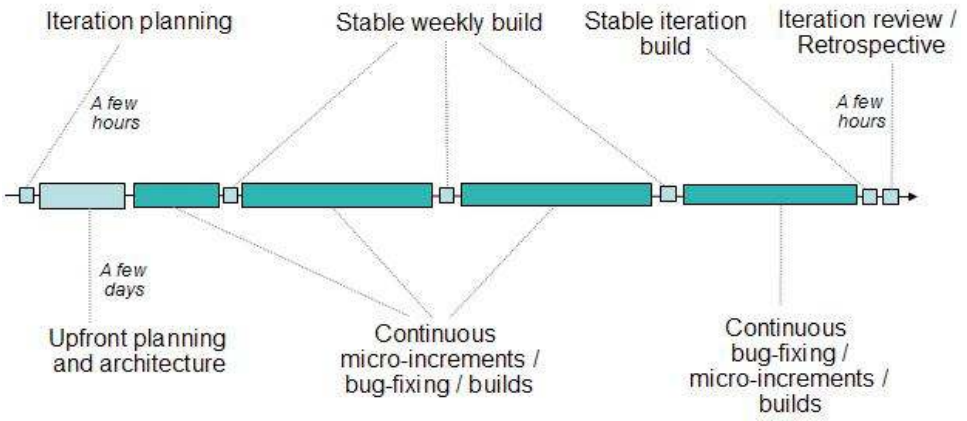


Figura 2.5.4: Énfasis dentro de una iteración. [OPENUP web]

Dar al equipo la posibilidad de organizar su trabajo y determinar la mejor forma de alcanzar los objetivos motiva a los miembros del equipo a hacer lo mejor que pueden. También, los ayuda a colaborar entre sí para asegurar que se aplican las habilidades apropiadas. La auto organización impacta en muchas áreas, incluyendo cómo planificar o asumir los compromisos (por el equipo, no individuos), cómo se asigna el trabajo (no se asigna a alguien sino que cada uno elige qué hacer), y cómo los miembros del equipo ven los roles en el proyecto (en primer lugar miembros del equipo, luego la función del trabajo).

2.5.8 Microincrementos

Un microincremento es un paso pequeño, medible hacia los objetivos de una iteración. Representa unas pocas horas hasta un día de trabajo realizado por una o unas cuantas personas colaborando para alcanzar el objetivo

El concepto de microincremento ayuda al miembro individual del equipo a partir el trabajo en pequeñas unidades que entregan algo de valor medible al equipo. Los microincrementos proveen una retroalimentación extremadamente corta que conduce a decisiones adaptativas dentro de una iteración.

Un microincremento debe estar bien definido y se debe poder rastrear diariamente su progreso. En un ítem de trabajo se especifican y rastrean los microincrementos. Ejemplo de microincrementos:

- **Identificar Stakeholders**. Dentro de una tarea que puede llevar unas semanas como es Definir la Visión, esta actividad constituye un microincremento.
- **Desarrollar un incremento de solución**. Definir, diseñar e implementar un Caso de uso puede llevar varias semanas, se puede dividir el trabajo y considerar subflujos de un caso de uso.
- **Definir el plan de la iteración**. Podría incluir una reunión para crear el plan y realizar alguna preparación previa para la misma.

La aplicación evoluciona en microincrementos a través de la ejecución simultánea de un número de ítems de trabajo. Se logra la transparencia y posibilidad de comprender el trabajo de cada uno a través de compartir abiertamente el progreso a través de los microincrementos.

2.5.9 Ciclo de vida del desarrollo de software mediante OpenUP

Open UP es un proceso iterativo con iteraciones distribuidas en cuatro fases ya mencionadas anteriormente: Inicio, Elaboración, Construcción y Transición. Cada fase puede tener tantas iteraciones como sean necesarias (dependiendo del grado de novedad, del dominio del negocio, la tecnología usada, la complejidad de la arquitectura, el tamaño del proyecto y de otros factores). Las iteraciones pueden tener longitud variable dependiendo de las características del proyecto, si bien las iteraciones de un mes son las que se recomiendan en general.

Para ofrecer un comienzo rápido a los equipos para planear sus iteraciones, OpenUp propone desglosar el trabajo para cada iteración en un WBS (Work Breakdown Structure - en castellano Estructura de Desglose de Tareas, EDT), y un WBS para todo el proceso del proyecto (de extremo a extremo).

En cada fase se desarrolla y entrega versiones del software estables y que funcionan. La finalización de cada iteración representa un hito menor del proyecto y contribuye a lograr el éxito del hito mayor de la fase donde se alcanzan los objetivos de la fase. Cada fase termina con un hito esperado proveyendo una revisión haciendo surgir y respondiendo un conjunto de preguntas que son comúnmente críticas para los stakeholders:

- **Inicio** (Inception). ¿Se está de acuerdo con el alcance y los objetivos? ¿Se debe continuar o no con la realización del proyecto?

- **Elaboración** (Elaboration). ¿Se está de acuerdo con la arquitectura ejecutable a utilizarse para desarrollar la aplicación? ¿Es aceptable el valor entregado hasta el momento y el riesgo remanente?

- **Construcción** (Construction).¿La aplicación está suficientemente cercana a ser entregada de tal forma que se deba cambiar el foco primario del equipo a refinar, pulir y asegurar un exitoso despliegue?

- **Transición** (Transition). ¿La aplicación está lista para ser liberada?

Si la respuesta es SI a las preguntas anteriores en la fase de revisión, el proyecto continúa. En caso de ser negativa la respuesta, la fase es demorada (generalmente agregando una nueva iteración) hasta que se reciba una respuesta satisfactoria, o bien los stakeholders determinen que debe cancelarse el proyecto.

La siguiente figura muestra el ciclo de vida de Open UP. Este ciclo de vida del proyecto provee a los stakeholders y miembros del equipo con visibilidad, sincronización, puntos de decisión a través del proyecto permitiendo momentos de revisión y decisiones de continuar o no continuar.

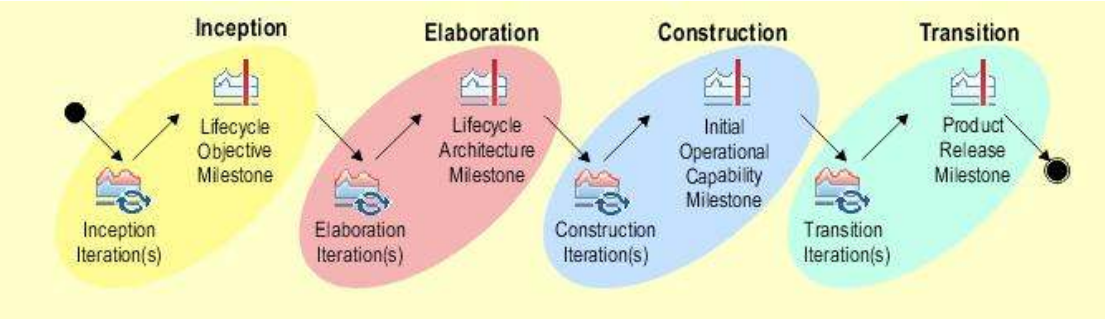


Figura 2.5.5. Ciclo de vida de OpenUP [OPENUP web]

El ciclo de vida del proyecto provee a los stakeholders con revisión, transparencia y mecanismos de dirección para controlar las bases del proyecto, el alcance, la exposición a los riesgos, el valor provisto y otros aspectos del proceso.

Cada iteración entrega un incremento del producto, que provee a los stakeholders la oportunidad de entender el valor que ha sido entregado y qué tan bien está marchando el proyecto. También le brinda al equipo de desarrollo la oportunidad de realizar cambios al proyecto para optimizar la salida.

Uno de los objetivos del ciclo de vida del proyecto es enfocarse en dos conductores claves de los stakeholders: reducción de riesgo y creación de valor. Las fases de OpenUP enfocan al equipo en la reducción del riesgo relacionado con preguntas a ser contestadas al final de la fase, mientras se rastrea la creación de valor, ver Figura 2.5.6.

El riesgo es la posibilidad de que ocurran cosas inesperadas en el proyecto, y los riesgos se interponen en la creación de valor. El riesgo es directamente proporcional a la incertidumbre estimada, y los stakeholders generalmente quieren saber más temprano y no más tarde qué valor del proyecto se puede entregar en el tiempo estipulado. En muchos casos, se reduce el riesgo cuando se crea valor implementando y probando las características más críticas. Sin embargo, hay situaciones donde la reducción del riesgo y la inmediata creación de valor están contrapuestas.

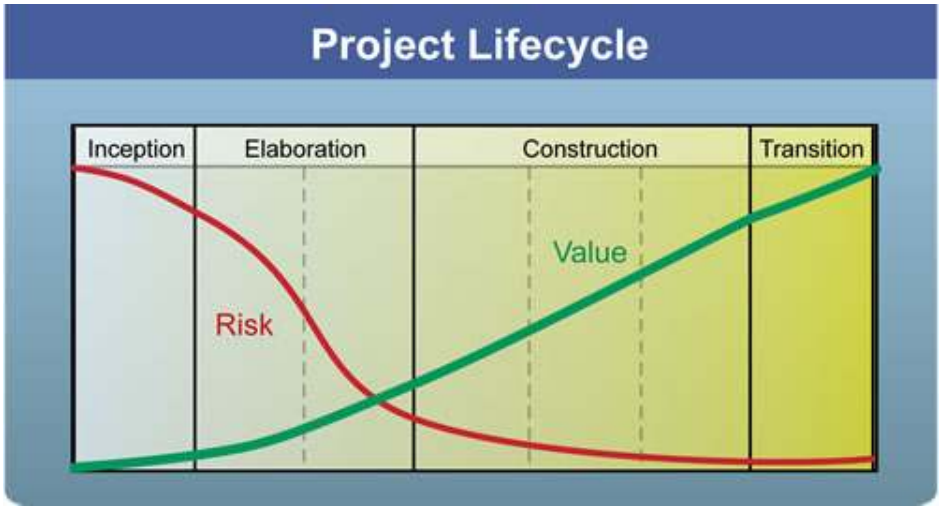


Figura 2.5.6: Reducción de Riesgo (curva roja) y creación de valor (curva verde) durante el ciclo de vida del proyecto.[OPENUP web]

2.5.9.1 Fase de Inicio

Al ser la primera de las cuatro fases, esta fase busca comprender el alcance del proyecto y sus objetivos y también obtener suficiente información para confirmar que el proyecto debe continuar o convencer que debe cancelarse. El propósito es alcanzar un acuerdo entre todos los stakeholders sobre los objetivos del proyecto.

Hay cuatro objetivos en esta fase que clarifican el alcance, objetivos del proyecto y factibilidad de la solución pretendida: [Kroll 2003]

- **Entender lo que se va a construir.** Determinar una visión general, incluyendo alcance del sistema y sus límites. Identificar stakeholders.

- **Identificar las funcionalidades claves del sistema.** Decidir qué requerimientos son más críticos.

- **Determinar por lo menos una posible solución.** Evaluar si la visión es técnicamente factible. Esto puede involucrar identificar una arquitectura candidata de alto nivel o realizar prototipos técnicos, o ambas cosas.

- **Entender** a un alto nivel la estimación de costos, calendario y riesgos asociados al proyecto.

2.5.9.2 Fase de Elaboración

Esta es la segunda fase donde se tienen en cuenta los riesgos estructuralmente significativos. El propósito de esta fase es establecer la línea base de la arquitectura del sistema y proveer una base estable para la mayor parte del esfuerzo de desarrollo de la siguiente fase. La mayoría de las actividades durante la fase de elaboración se hace en paralelo.

Hay tres objetivos que ayudan a tratar los riesgos asociados con los requerimientos, arquitectura, costos y calendario [Kroll 2003]:

- Obtener un entendimiento más detallado de los requerimientos. Asegurarse de tener un conocimiento profundo de los requerimientos más críticos**.**

- Diseñar, implementar, validar y establecer la línea base de la arquitectura (para el esqueleto de la estructura del sistema).

- Mitigar riesgos esenciales y producir un calendario apropiado y estimación de costos.

Esencialmente, los principales objetivos para la elaboración están relacionados con el mejor entendimiento de los requerimientos, creando y estableciendo una línea base de la arquitectura para el sistema, y mitigando los riesgos de mayor prioridad.

2.5.9.3 Fase de construcción

La tercera fase se enfoca en diseñar, implementar y probar funciones para desarrollar un sistema completo. El propósito es completar el desarrollo del sistema tomando como base la arquitectura definida en la fase anterior.

Hay ciertos objetivos que ayudan a tener un desarrollo costo efectivo para un producto completo, una versión operativa del sistema, que puede entregarse al usuario [Kroll 2003]:

- Desarrollar iterativamente un producto completo que esté listo para la fase de transición.

- Minimizar los costos de desarrollo y lograr cierto grado de paralelismo.

2.5.9.4 Fase de transición

La última de las fases se enfoca en desplegar el software a los usuarios y asegurarse que se alcanzaron sus expectativas sobre el software. El propósito es asegurar que el software está listo para entregarse al usuario.

Hay tres objetivos que ayudan a refinar la funcionalidad, performance, y calidad global del producto beta completado en la fase anterior [Kroll 2003]:

- Realizar una Prueba Beta para validar que se cumplen las expectativas del usuario.

- Lograr constancia por parte de los stakeholders para asegurar que el despliegue esté completo. Hay varios niveles de pruebas para la aceptación del producto.

- Mejorar el rendimiento de proyecto futuro a través de lo aprendido.

2.5.10 Como comenzar

El cuarto principio fundamental de OpenUP, "*Evolucionar para obtener retroalimentación continua y mejorar* ", sugiere un enfoque iterativo e incremental para adoptar OpenUP.

- Comenzar con los principios fundamentales y comprender las intenciones detrás de OpenUP.

- Dentro de procesos existentes, elegir una o dos áreas claves a mejorar.

- Comenzar usando OpenUP para mejorar esas áreas primero.

- En iteraciones ciclos de desarrollos posteriores, realizar mejoras incrementales en otras áreas.

- En caso de no tener experiencia usando proceso unificado u otro proceso iterativo, usar OpenUP en un pequeño proyecto piloto.

En OpenUP puede extenderse el proceso (como por ejemplo auditorías de sistemas de seguridad críticos) o modificarse las plantillas de los productos de trabajo (work product) para adaptarse a necesidades específicas, ya sea que se necesite mayor precisión en los work products, la organización tenga ciertos protocolos en los procesos, añadir experiencia, etc.

2.5.11 Conclusiones

Open UP es una metodología de desarrollo completa, en el sentido que puede verse como un proceso entero para construir un sistema. Mantiene las características esenciales de RUP y realiza un manejo de Riesgos pero, a su vez, define cuatro principios que lo rigen y que pueden vincularse con los cuatro valores del Manifiesto ágil.

Las iteraciones son de tiempo prefijado y su proceso puede ser personalizado y extendido para distintas necesidades. Si bien se proponen una serie de artefactos esenciales, éstos se pueden reemplazados con los propios del equipo de desarrollo. Cada iteración entrega un incremento del producto, que provee a los stakeholders la oportunidad de entender el valor que ha sido entregado y comprender qué tan bien está marchando el proyecto para, de esta forma, brindar al equipo de desarrollo la oportunidad de realizar cambios al proyecto para optimizar la salida.

Open UP es para equipos pequeños que trabajan colaborativamente en la misma ubicación. Los equipos necesitan involucrarse en una interacción plena cara a cara diariamente.

2.6 CRYSTAL CLEAR

2.6.1 Introducción

Según Alistair Cockburn, el desarrollo de software puede caracterizarse como un juego cooperativo de invención y comunicación con restricciones económicas, descripto en Agile Software Development [Cockburn 2002]. La forma en que cada equipo juega cada partido tiene mucho que ver con la salida del proyecto y el software resultante. Crystal Clear ataca directamente el juego económico-cooperativo, apuntando dónde prestar atención, dónde simplificar y cómo variar las reglas.

Crystal Clear no aspira a ser la "mejor" metodología; aspira a ser "suficiente", de tal manera que el equipo lo amolde a sus necesidades y lo use. Alistair Cockburn [Cockburn 2002]

Crystal Clear es una metodología que funciona con personas, descripta de la manera más simple como sigue [Cockburn 2002]:

El diseñador líder y otros dos a siete desarrolladores en una gran habitación o habitaciones contiguas, con radiadores de información, como pizarrones y rotafolios en la pared, teniendo acceso a usuarios claves, distracciones mantenidas al margen, entregando y corriendo código usable y probado cada mes o dos (a lo sumo tres), reflexionando periódicamente y ajustando su propio estilo de trabajo.

2.6.1.1 ¿Cuál es la base para Crystal?

La primera base para Crystal es empírica. En 1991 Alistair Cockburn fue contratado por IBM Consulting Group para crear una metodología para los próximos proyectos con tecnología de objetos. Comenzó a visitar y tomar notas de diferentes equipos en todo el mundo. Después de muchos años de entrevistas y trabajando directamente en proyectos, determinó que las técnicas y productos de trabajo detallados en las "metodologías" no formaban parte del corazón de propiedades de éxito, más bien, estaban en un segundo orden de factor de éxito. Al hacer que las personas trabajen muy juntas, comunicándose de manera bien predispuesta, entregando a menudo y obteniendo retroalimentación rápidamente de los usuarios, probablemente el equipo logre resolver el resto por su propia cuenta, usando cualquier tecnología y técnicas que conozca. Posteriormente se dio cuenta que estas recomendaciones formaban una metodología, que contenía pocas reglas pero permitía a las personas actuar como profesionales y obtener el mejor camino al éxito por su propia cuenta

La segunda base consiste en una serie de 10 principios y de literatura sobre la comunicación humana y el proceso de diseño. Los principios resumidos son los siguientes:

- **Diferentes proyectos necesitan diferentes compensaciones (trade-offs).** de metodología. Se debe tener en cuenta el número de personas a ser coordinadas y el grado de daño que podría causar el mal funcionamiento del sistema para considerar la metodología a adoptar.

- **Equipos más grandes necesitan mayores elementos de comunicación**. Por ejemplo, Crystal Clear está recomendado para equipos que pueden lograr comunicación osmótica.

- **Proyectos que se enfrentan a daños potenciales mayores necesitan más elementos de validación**. Por ejemplo, un grupo pequeño que trabaja en un sistema que manipula barras de boro en un reactor nuclear tendrá más cuidado en su trabajo que un sistema organizador de recetas de cocina. La diferencia está en la dimensión de verificación y validación y no en la dimensión de comunicación y coordinación.

- **Una pequeña metodología produce muchos beneficios, después de eso el peso es muy costoso**. A diferencia del pensamiento "más metodología es mejor" el autor propone "menos es generalmente mejor, mientras se cubra el resto con comunicación personal". Jim Highsmith da el consejo de comenzar con menos de lo que se piensa

necesitar y probablemente eso sea todo lo necesario; es mas fácil agregar algo después que quitar algo.

- **La formalidad, el proceso y la documentación no son sustitutos de disciplina, habilidad y entendimiento**. Este principio de Jim Highsmith del año 2003 articula la diferencia entre metodologías ágiles y tradicionales. El software lo construyen personas y la disciplina, la habilidad y el entendimiento son propiedades internas de una persona y no pueden remplazarse por una forma externa.

- **Comunicación interactiva, cara a cara es el canal más barato y rápido para intercambiar información**. Por ejemplo, dos o tres personas paradas frente a una pizarra, diagramando y hablando pueden tomar ventaja de saludables canales de comunicación y obtener retroalimentación casi instantánea.

- **Aumentar la retroalimentación y la comunicación reduce la necesidad de entregables intermedios**. Si el equipo desarrolla algo y lo muestra al usuario, por ejemplo cada mes, el tiempo de demora es relativamente pequeño y no es necesario hacer promesas elaboradas sino mostrar el resultado del mes y aprender directamente lo que está bien y corregir lo que es incorrecto.

- **El costo de desarrollo concurrente y serial se entrecruzan por velocidad y flexibilidad**. Desarrollar de manera concurrente puede hacer más rápido el desarrollo a un costo posiblemente superior comparado con un desarrollo serial ejecutado correctamente. El problema del costo más bajo es que cualquier causa de mala interpretación implica rehacer trabajo, que es muy caro. El desarrollo concurrente permite descubrir errores de interpretación en tiempo real, pero al mismo tiempo requiere mejor comunicación entre las personas.

- **La eficiencia es prescindible en actividades que no son cuello de botella**. El libro, *The Goal* [Goldratt 1992], identifica que cada proceso tiene una estación “cuello de botella”, que condiciona la velocidad de toda la empresa. La familia Crystal agrega el corolario que las estaciones que no son cuello de botella pueden ayudar en ciertas formas, operando con menos eficiencia. Si las personas tienen capacidades libres pueden ayudar en otras actividades.

- **"Puntos dulces (sweet spots)" aceleran el desarrollo**. El mejor de los mundos es tener personas (1) dedicadas, (2) expertas que (3) se sientan de manera que puedan oírse, (4) usan pruebas de regresión automatizadas, (5) tienen acceso fácil a los usuarios, y (6) entregan sistemas corriendo, ya probados a esos usuarios cada mes o dos. Tal proyecto está en una posición mayor para tener éxito que otro donde falten estas características. Crystal Clear se construye sobre las cuatro últimas ya que no se puede contar con todo el personal capacitado y dedicado.

Ya que es difícil recordar los 10 principios, ellos se derivan de una idea, la del juego cooperación – económico, que reencuentra descripto en [Cockburn 2004]. El manifiesto del juego cooperativo dice:

El desarrollo de software es una serie de recursos limitados, juegos cooperativos de objetivos directos de invención y comunicación. El objetivo primario de cada juego es la producción y liberación de un sistema de software; el residuo del juego es un conjunto de marcas para ayudar a los jugadores en el próximo juego. El siguiente juego es una alteración al sistema o la creación de sistemas vecinos. Cada juego por lo tanto tiene como segundo objetivo crear posiciones ventajosas para el siguiente juego. Dado que cada juego tiene recursos limitados, el objetivo principal y el secundario compiten por los recursos. [Cockburn 2004]

La ventaja de esta expresión es que resalta un par de aspectos importantes del desarrollo de software:

- Son las *personas* y su *invención* y *comunicación* que hacen que surja el software.

- Hay dos objetivos en juego en cada instante: entregar el sistema actual y definir las bases para el siguiente juego. Cada decisión, realizadas por cualquier persona involucrada, tiene consecuencias económicas. La mayoría de las recomendaciones de esta metodología están basadas en este manifiesto del juego cooperativo.

El desarrollo de software es un juego de personas trabajando con y contra personas. Todos los motivos y emociones humanas se aplican y deben tenerse en cuenta, aún en Crystal.

2.6.1.2 La Familia Crystal

Crystal es una familia de metodologías con un código genético común, uno que enfatiza entregas frecuentes, comunicación cercana y mejora reflexiva. No hay una metodología Crystal. Hay diferentes metodologías Crystal para diferentes tipos de proyectos. Cada proyecto u organización usa el código genético para generar nuevos miembros de la familia.

El nombre "Crystal" viene de la caracterización de Alistair Cockburn de los proyectos en dos dimensiones, tamaño y criticidad, comparando con los minerales, color y dureza (ver Figura 2.6.1). Grandes proyectos, que requieren más coordinación y comunicación, se mapean a colores más oscuros (clear, amarillo, naranja, rojo, y así siguiendo). Proyectos para sistemas que pueden causar más daño necesitan agregar dureza o rigidez a la metodología, así como más reglas de validación y verificación. Una metodología de cuarzo es apropiada para pocos desarrolladores creando un sistema de facturación. El mismo equipo controlando los movimientos de barras de boro en un reactor nuclear necesita una metodología de diamante, que establezca chequeos repetidos tanto en el diseño como en la implementación de sus algoritmos.

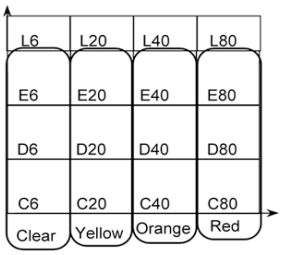


Figura 2.6.1: Cobertura de diferentes tipos de proyectos dentro de Crystal. [Cockburn 2004]

El autor caracteriza a las metodologías Crystal por color, según el número de personas que son coordinadas: *Clear* es para equipos de 8 personas o menos ubicadas en un mismo lugar, *Amarillo* es para equipos de 10 - 20 personas, *Naranja* es para equipos de 20 - 50 personas, *Rojo* para equipos de 50 - 100 personas, y así sucesivamente, pasando por el *Marrón*, *Azul*, *Violeta*.

2.6.1.3 El código genético de Crystal

El código genético de Crystal está compuesto por:

- El modelo de juego económico-cooperativo (The economic-cooperative game model)
- Prioridades Seleccionadas
- Propiedades Seleccionadas
- Principios Seleccionados
- Técnicas de Muestreo Seleccionadas
- Ejemplos de Proyecto

El *modelo de juego económico-cooperativo,* ya explicado en páginas anteriores, conduce a las personas en un proyecto a pensar sobre su trabajo de una manera muy específica, enfocada y efectiva.

Las *prioridades* comunes a la familia Crystal son:

- Seguridad en el resultado del proyecto: obtener un resultado de negocio razonable, dadas las prioridades del proyecto y las restricciones de recurso.
- Eficiencia en el desarrollo es una prioridad alta porque muchos proyectos están económicamente sobre-exigidas.
- Habitabilidad (*habitability*) de las convenciones. Las reglas necesitan ser tales que las personas dentro del equipo puedan vivir con ellas y no las ignoren.

Las prioridades interactúan entre sí. La *habitablilidad* significa que se acepta que un conjunto de reglas dado puede no ser lo más *eficiente* posible. La Prioridad de *seguridad* significa que las reglas elegidas deben ser lo suficientemente buenas para obtener resultados adecuados. Parte de la eficiencia y habitabilidad es enfrentarse al hecho de que cada proyecto es levemente diferente y necesita su propia metodología hecha a la medida. Cada equipo adapta la metodología base según sus requerimientos, experiencia, y características tecnológicas. Esto debe hacerse rápidamente, para que no sea costoso. Para poder definir un conjunto de reglas básicas para un proyecto en particular, Alistair Cockburn construye una tabla de dos entradas, como se mostró en la Figura 2.6.1.

Crystal conduce al equipo del proyecto hacia siete *propiedades* de seguridad, las tres primeras son el corazón de Crystal. Las otras pueden agregarse para aumentar el margen de seguridad. Las propiedades son:

- Entregas Frecuentes
- Mejora reflexiva
- Comunicación cercana
- Seguridad del Personal, el primer paso en la confianza
- Foco
- Facilidad de acceso a usuarios expertos
- Entorno técnico con pruebas automatizadas, administración de configuración e integración frecuente

Los *principios* de Crystal se describen en detalle en Agile Software Development [Cockburn 2002]. Entre ellos hay unas cuantas ideas centrales:

- El nivel de detalle necesario en los documentos de requerimientos, diseño y planificación varía con las circunstancias del proyecto, específicamente la extensión del daño que podría causarse por no detectar defectos y la frecuencia de la colaboración del personal aportada al equipo.
- Podría no ser posible eliminar todos los productos intermedios y documentos legales como documentos de requerimientos, diseño y planes de proyecto; pero pueden reducirse ya que hay caminos de comunicación cortos, ricos e informales en el equipo y se entrega de manera temprana y frecuente software funcionando y probado.
- El equipo continuamente ajusta sus convenciones de trabajo para adaptarse a las personalidades particulares en el equipo, el entorno actual de trabajo local y las peculiaridades de la asignación específica.

Crystal no requiere ninguna técnica específica a utilizar, solo desarrollo incremental. Por lo tanto, el conjunto permitido de *estrategias y técnicas* es muy amplio. Sin embargo, el autor presenta un conjunto de técnicas que se incluyen como un conjunto inicial.

2.6.1.4 Crystal Clear y la familia Crystal

Cada miembro de la familia Crystal se genera al inicio del proyecto dándole forma a una metodología base según el código genético. Como la situación cambia a través del tiempo, la metodología se afina durante el transcurso del proyecto. Tanto las tareas para darle forma como para afinarla se realizan lo suficientemente rápidas de tal forma que el tiempo empleado se recupera dentro del marco del tiempo del proyecto.

Crystal Clear es una optimización de Crystal que puede aplicarse cuando el equipo consiste de 2 a 8 personas ubicadas en la misma habitación u oficinas adyacentes. La propiedad de comunicación cercana se refuerza a comunicación "osmótica", significando que las personas escuchan a los otros discutiendo prioridades, estado, requerimientos, y diseño de proyecto diariamente. Esta comunicación reforzada permite al equipo trabajar más desde una comunicación tácita y notas pequeñas que de otra forma no sería posible.

Crystal Clear comparte algunas características con XP pero es generalmente menos demandante. Es una alternativa más relajada, un lugar para volver si XP no está funcionando para el grupo, o un trampolín para obtener algunas prácticas ágiles antes de saltar a XP.

2.6.2 Roles en Crystal Clear

Hay ocho roles definidos para Crystal Clear: sponsor ejecutivo, usuario experto, diseñador-líder, diseñador-programador, experto del negocio, coordinador, tester, writer. Los cuatro primeros deberían ser personas diferentes. Los otros cuatro pueden ser roles adicionales asignados a personas del proyecto.

- **Sponsor Ejecutivo:** Es la persona que provee el dinero al proyecto o quien representa a esa persona. Debe mantener la visión a largo plazo en mente, balanceando las prioridades a corto plazo con aquellas de entregas posteriores o evoluciones del equipo o mantenimiento del sistema. Esta persona creará la visibilidad externa para el proyecto y proveerá al equipo con decisiones cruciales a nivel del negocio. Parte de la metodología involucra entregar al sponsor ejecutivo buena información para tomar esas decisiones.

- **Usuario Experto:** Es la persona que se supone está familiarizada con los procedimientos operacionales y el sistema en uso, si es que hay alguno, conociendo cuáles son modos frecuentes e infrecuentes de operación, qué atajos se necesitan, y qué información debe verse junta en la pantalla al mismo tiempo.

- **Diseñador líder:** Es la persona técnica líder, la persona que se supone tiene experiencia con el desarrollo de software, capaz de realizar el diseño mayor del sistema, que avisa cuando el equipo de proyecto está o no en cronograma, y si no lo está como volver al cronograma. El líder de diseño suele tener mayor influencia en el taller para darle forma a la metodología. Alistair usa la palabra "diseñador" para este rol para cortar el nombre "líder diseñador-programador." El diseñador líder debe diseñar y programar como los otros diseñadores-programadores.

- **Diseñador-Programador:** Al definir este rol, Alistair combina ambas palabras “diseñador” y “programador” para resaltar que cada persona programa y diseña. Ni diseñador ni programador se mantienen como nombre de un rol separado.

- **Coordinador:** La coordinación es probablemente una ocupación parcial de algún miembro del equipo. La persona que ocupa el rol de coordinador debe, como mínimo, tomar nota en la planificación del proyecto y las sesiones de estado del proyecto, así rastrear la información para enviar o presentar. El coordinador es responsable de dar, a los sponsors del proyecto, visibilidad sobre la estructura y el estado del proyecto.

- **Experto de negocio:** Es el experto sobre cómo funciona el negocio, qué estrategias o políticas son fijas, cuales pueden variar pronto, a menudo o de vez en cuando. La información que provee es diferente a la que típicamente puede proveer un usuario experto, aunque en algunos casos puede ocurrir que el usuario experto sea también el experto del negocio.

- **Tester y writer:** suelen ser asignaciones temporales que van rotando. Algunos equipos pueden contratar un writer por períodos de tiempo o tener un tester dedicado trabajando e inclusive, sentado con ellos.

2.6.3 Los productos de trabajo

Los productos de trabajo no son ni completamente requeridos ni tampoco completamente opcionales. Pueden usarse uno a la vez o todos juntos. Se permite una sustitución equivalente, ya que es una forma de ajustarlo y variarlo. En una metodología pequeña como Clear, el número y la formalidad de productos de trabajo intermedios se reduce significativamente. El equipo vive de la comunicación personal, notas, pizarras o posters alrededor del cuarto, y las demos o entregables en su comunicación con el usuario.

Los productos de trabajo listados aquí son un conjunto por defecto para un proyecto típico de Crystal Clear porque han demostrado, según Cockburn, su valor en muchos proyectos. Pero, depende del equipo del proyecto agregar, sacar o modificar la lista basados en su situación. Cada ítem sirve a un propósito de comunicación en el juego económico-cooperativo. Hay casi 24 productos de trabajo, dependiendo de cómo se los cuente. Los productos de trabajo aquí son livianos y están distribuidos a través de los roles, así el equipo no lo encuentra agobiante en la práctica.

No se puede definir un grupo de productos que sea el corazón. Si se saltean muchos productos de trabajo se pierde alineación y visibilidad. La dirección y apoyo pueden sufrir por ello. Por otro lado, no sería lo más apropiado insistir en realizar todos ya que los productos de trabajo específicos que necesita el equipo varían según técnicas, tecnologías, hábitos de comunicación y moda que todas son cambiantes. El desacuerdo dentro del equipo y la falta de comunicación con el mundo exterior aumenta con cada omisión. Los riesgos se acumulan, lentamente al principio, hasta que el proyecto no está en zona segura, y nunca está claro cuándo pasa de zona segura a no segura.

A continuación se presenta una tabla indicando que rol es el último responsable de cada producto. Muchos productos tienen una forma informal y otra formal de utilizarlos. Los productos de Crystal se describen en detalle en [Cockburn 2004] y para no exceder con información no se incluyen en el informe.

Rol- último responsable	Productos
El sponsor (patrocinador, quien financia)	La declaración de la Misión con el Trade-off de prioridades.
El equipo	La estructura y las convenciones del equipo Los resultados del trabajo de reflexión.
El coordinador, con ayuda del equipo	El Mapa del Proyecto, El Plan de Entrega, El Estado del Proyecto, La Lista de Riesgo, El plan y Estado de la Iteración La visualización del Calendario -Cronograma.
El experto del negocio y usuario experto juntos	La lista de objetivos por actor: Los Casos de Uso, El archivo de Requerimientos: El modelo del rol del usuario
El líder de diseño (diseñador líder)	La descripción de la Arquitectura.

Los diseñadores-programadores (incluyendo al líder de diseño)	Borradores de pantalla, Modelo de Dominio Común, Esquemas y notas de diseño, Código fuente, Código de Migración, Las Pruebas El sistema empaquetado.
EL tester	Reporte de errores en ese momento
El writer	El texto de la ayuda al usuario.

Tabla 2.6.1: Roles y productos en Crystal (basado en [Cockburn 2004])

2.6.4 Proceso Crystal Clear

2.6.4.1 Tipos de Procesos Involucrados

Crystal Clear usa procesos cíclicos anidados de varias longitudes: el episodio de desarrollo, la iteración, el período de entrega y el proyecto completo. Lo que las personas realizan en cada momento depende en qué ciclo están.

Se da el nombre de *Desarrollo concurrente* cuando el trabajo se realiza en paralelo en productos de trabajo dependientes. Muchas o todas las actividades se realizan al mismo tiempo. Las personas intercambian notas continuamente para mantenerse al día con el estado cambiante de cada parte.

Se definen siete ciclos que se encuentran en la mayoría de los proyectos:

1. El proyecto (una unidad de fondos, que puede ser de cualquier duración)
2. El ciclo de entrega (una unidad de entrega, una semana a tres meses)
3. La iteración (una unidad de estimación, desarrollo y celebración, una semana a tres meses)
4. La semana de trabajo (ritmo calendario por ejemplo reuniones de los lunes)
5. El período de integración (una unidad de desarrollo, integración y prueba de sistema, 30 minutos a tres días)
6. El día de trabajo (ritmo calendario por ejemplo daily stand up meeting)
7. El *episodio* de trabajo (desarrollando y chequeando en una sección de código, tomando de unos minutos a unas horas)

Crystal Clear requiere múltiples entregas por proyecto, pero no múltiples iteraciones por entrega. Cada ciclo tiene su propia secuencia y su propio ritmo. En un día dado se realizan diferentes actividades de diferentes ciclos.

Las Figuras 2.6.2 a 2.6.5 muestran diferentes formas de desenrollar estos ciclos. Se puede notar que episodios, días y períodos de integración pueden anidarse de diferente forma dentro de una iteración. Las Figuras 2.6.3 y 2.6.4 muestran dos formas posibles.

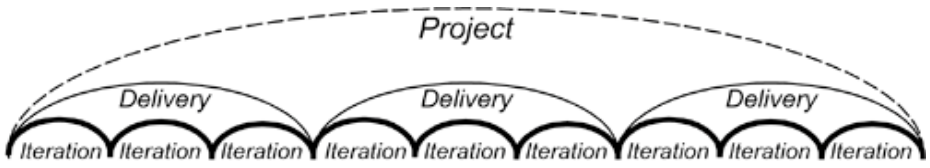


Figura 2.6.2: Iteración y ciclos de entrega dentro de un proyecto [Cockburn 2004]

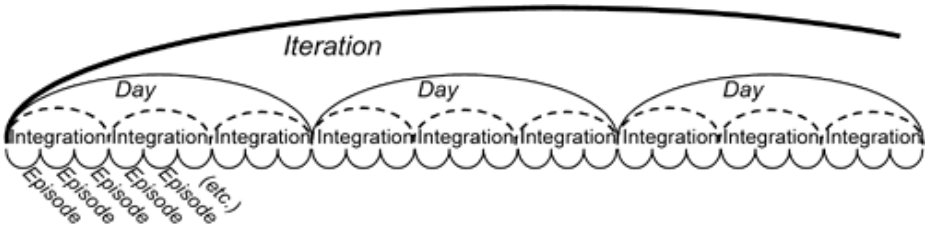


Figura 2.6.3: Un ciclo de iteración, integrando varias veces por día [Cockburn 2004]

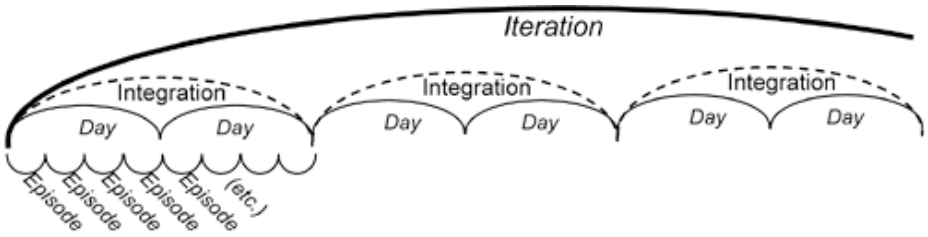


Figura 2.6.4: Un ciclo de iteración con varios días por integración [Cockburn 2004].

La Figura 2.6.5 muestra la expansión de cada ciclo de forma separada, listando las actividades específicas que ocurren para ese tipo de ciclo

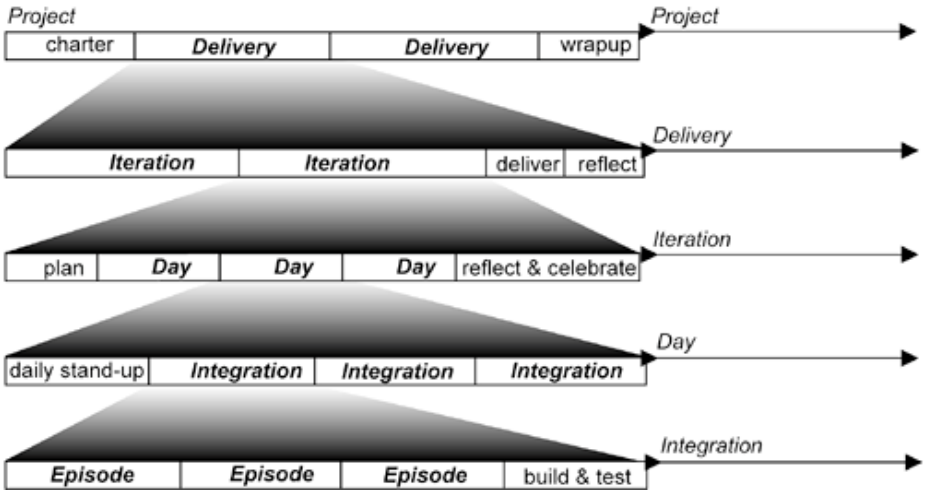


Figura 2.6.5: Los ciclos expandidos para mostrar actividades específicas. [Cockburn 2004]

De los ciclos antes mencionados se ampliarán solo algunos de ellos que requieren mayor explicación.

2.6.4.2 El ciclo del Proyecto

Aunque un proyecto se crea una vez para todas las actividades, es seguido por otro proyecto, en un ciclo que se repite. Tiene tres partes:

- **Una actividad de caracterización (*chartering*):** Esta actividad toma de unos pocos días a unas semanas. Consiste de cuatro pasos: Definir el corazón del equipo, realizar la exploración de *360°* (pudiendo resultar en la cancelación del proyecto), dar forma y afinar las convenciones de la metodología y construir el plan inicial del proyecto.

- **Dos o más ciclos de entrega:** Este ciclo tiene tres o cuatro partes: una re-calibración del plan de entregas, una serie de una o más iteraciones, cada una resultando en código integrado y probado, entrega a usuarios reales y un ritual de finalización,

incluyendo reflexión tanto en el producto que se está creando como en las convenciones que se usan. Se resalta que el tener solo un ciclo de entrega es una violación a Crystal Clear.

- **Un ritual de finalización:** el empaquetado del proyecto Es por eso que después de una entrega se debe proporcionar un tiempo de relajación por las presiones del ciclo. Después de la entrega el equipo tiene dos aspectos más para reflexionar:

  1. *¿Cómo fue la distribución? ¿Qué se podría hacer para reducir los padecimientos a la hora de distribuirlo a los usuarios y entrenarlos?*
  2. *¿Qué piensan los usuarios sobre el sistema? ¿Cuáles son los puntos fuertes y débiles?¿Se puede aprender algo de lo que realmente necesitan los usuarios respecto de lo solicitado originalmente?*

  La reflexión en el proceso de entrega es la misma que para cualquier otro workshop de reflexión. El grupo se pregunta qué es lo que quieren mantener o hacer diferente. El punto a destacar es que se revisa el *producto*, no el proceso.

2.6.4.3 El Ciclo de Iteración

La duración y el formato de las iteraciones varían según los equipos. Podría considerarse una duración desde 1 semana a 2 meses. Una iteración tiene tres partes: (1) Planificación de la iteración, (2) Actividades diarias y de ciclo de integración y (3) Ritual de finalización que incluye workshop de reflexión y celebración. Durante el período de la iteración, el equipo estará agregando requerimientos, evaluando diseños de interfaz de usuario, entendiendo la arquitectura del sistema, agregando funcionalidad, mostrando el sistema al usuario, agregando pruebas.

En la *planificación* se definen las prioridades de la iteración. Se divide el trabajo en pequeñas partes. La duración variará si es una iteración de una semana o de dos meses. Cada día el equipo tiene reunión diaria de *check-in* or *stand-up* meeting. La esencia es mantener la reunión corta (a lo sumo 10 minutos) y permitir a cada miembro saber qué le está sucediendo a los otros miembros del equipo.

Luego, el equipo simplemente realiza sus actividades normales de desarrollo. Los *episodios de desarrollo* consisten en simplemente tomar una asignación de trabajo, desarrollarla y chequearla respecto al sistema de administración de configuración, más realizar la integración y pruebas de sistema, si se usa esa convención en el equipo. Probablemente discutan y muestren su trabajo al sponsor o usuario experto.

Si se trabaja con períodos muy cortos, el equipo suele olvidarse de llevar el sistema a usuarios reales. Se puede agregar un súper ciclo, que incluya visitas del usuario o sino crear explícitamente una agenda de Visitas del usuario. Si por el contrario se trabaja en ciclos muy largos, es recomendable tener un workshop de reflexión intermedio ya que permite descubrir si se ha detectado algo que pudiera poner en riesgo el éxito de la iteración.

Dentro del *ritual de finalización*, se realiza el workshop de reflexión. Se debe revisar cada aspecto de sus trabajos, desde sus relaciones con el sponsor y usuarios, hasta patrones de comunicación, hostilidad, la forma de recolectar requerimientos, convenciones de codificación, el entrenamiento que obtienen o no obtienen, nuevas técnicas que quisieran probar, etc. En iteraciones muy cortas es mejor realizarlos mensualmente o cada dos meses, para que se reflexione sobre un periodo mayor (Notar que esto crea el súper ciclo mencionado anteriormente). El workshop de fin de ciclo brinda un tiempo para reflexionar en lo que consideran positivo y negativo a cerca de sus hábitos de trabajo.

Después de la primera iteración o ciclo de entrega, muy a menudo los equipos ajustan sus estándares, obtienen más entrenamiento, hacen más eficientes sus flujos de trabajo, aumentan las pruebas, encuentran un “usuario amigable”, definen las convenciones de administración de configuración. Al final de ciclos subsiguientes, sus cambios tienden a ser mucho menores, a menos que estén experimentando con un proceso radicalmente nuevo.

El punto de esta reflexión periódica es atrapar errores a tiempo para poder repararlos dentro del proyecto y brindar la oportunidad a las personas de notar patrones ineficientes.

Se suelen realizar también actividades que logren descomprimir la saturación que tiene el equipo. Se pude utilizar una serie de juegos de computadora, juegos de ping pong, una discusión ligera sobre algún tópico profesional, un paseo en bicicleta, una salida a una confitería.

2.6.4.4 El episodio de distribución

Ward Cunningham acuño el término *episodio* para describir la unidad básica de trabajo de un programador en un desarrollo ágil. Durante un episodio, una persona toma una tarea de diseño pequeña, la programa hasta su finalización (idealmente con pruebas de unidad), y chequea respecto al sistema de administración de configuración. Esto puede tomar de 15 minutos a varios días, dependiendo del programador y las convenciones del proyecto. Funciona mejor mantener los episodios con una duración menor a un día.

Para Crystal Clear programar y diseñar no vale la pena discriminarlos, van juntos. La frase "tomar una tarea de diseño pequeña, programarla hasta su finalización" significa tomar una tarea pequeña que requiere diseño y programación, luego diseñar, programar, depurar y probarla hasta la finalización.

2.6.4.5 Reflexión sobre el Proceso

La vista de ciclos anidados permite la discusión de progreso y actividades de operaciones. Ambas son importantes para el equipo y forman parte del "proceso". En Crystal Clear, está permitido tener una iteración por ciclo de entrega, pero si se realiza esto, se deben tener *visitas* intermedias de usuarios reales. Si se tienen múltiples iteraciones por entrega, algunas de ellas deben incluir *visitas* de usuarios reales. Si no se realiza esto se está armando un equipo efectivo que produce el software equivocado muy eficientemente.

2.6.5 Conclusiones

Esta metodología se centra en las personas, la interacción, comunicación directa, habilidades, talentos con la convicción que son éstos quienes tienen el efecto mayor en el desempeño, dejando al proceso en un segundo lugar [Highsmith]. Cada equipo tiene un conjunto distinto de talentos y habilidades y por lo tanto cada equipo debe utilizar un proceso de desarrollo adaptado a él. Crystal Clear minimiza el proceso significativamente, una de las razones para realizar esto es que exige que la ubicación del equipo esté en un solo lugar físico.

Es una metodología ágil con la cual el equipo no solo va haciendo evolucionar los entregables sino el proceso mismo de desarrollo. Tal vez la contribución más importante es ofrecer las *Siete Propiedades de los Proyectos Exitosos*. Cockburn ha estudiado proyectos ágiles exitosos y ha identificado rasgos comunes entre ellos. Estas propiedades llevan al proyecto al éxito; en cambio su ausencia pone en riesgo el proyecto. [Cockburn 2004]

A diferencia de XP, para el autor de esta propuesta, si todos agregan y modifican código puede quedar desprolijo y llevar tiempo sacar mucha basura. Es mejor tener dueños de clases o casos de uso. Cualquiera puede hacer un cambio a corto plazo pero debe notificar al dueño para que lo revise.

2.7 TEST DRIVEN DEVELOPMENT)

Test Driven Development, TDD o Desarrollo Dirigido por Pruebas, es la técnica por antonomasia de XP, como se detalló en la sección sobre XP. Se encuadra dentro del ciclo de vida de la metodología XP, en las fases de iteración, producción y mantenimiento. Según Fowler, [Fowler 2006d], TDD es una técnica que conduce el proceso de desarrollo a través de pruebas. Pero, debido a su importancia, mucha gente ya la considera una metodología independiente de XP. Tal como lo afirma Carvajal Riola, se puede considerar a TDD una metodología ya que presenta un conjunto de prácticas y métodos de desarrollo, además de que condiciona la mentalidad de los desarrolladores guiándolos a través del desarrollo y aumentando la calidad del producto final. Puede ser aplicada independientemente de XP, en proyecto con Scrum, FDD o metodologías tradicionales. Debido a su radical planteamiento a la hora de escribir código, tal como se afirma en [Carvajal 2008], cambia drásticamente la mentalidad de cualquier equipo de desarrollo, generalmente agilizando los resultados y aumentando la calidad del sistema.

TDD [Beck 2003] [Astels 2003], es un enfoque de desarrollo evolutivo que combina *TFD (Test-First Development)* donde se escribe una prueba justo antes de escribir el código de producción que cumpla con esa prueba y *Refactorización*. Para algunos el objetivo de TDD es la especificación y no la validación [Martin 2003]. Es una manera de pensar a través de los requerimientos o diseño antes de escribir código funcional, implicando que TDD es una técnica importante para requerimientos ágiles y para diseño ágil. Para otros, TDD es una técnica de programación. Ron Jeffries afirma que el objetivo de TDD es escribir código limpio que funcione. Scott Amber adhiere a la primera línea de pensamiento. [AgileData web]

2.7.1 Procesos.

En esencia se siguen tres simples pasos que se repiten dentro de TDD:

- Escribir una prueba para la siguiente porción de funcionalidad que se pretende incorporar
- Escribir el código funcional hasta que pase la prueba
- Refactorizar tanto el código viejo como el nuevo para que esté bien estructurado

Se sigue iterando paso a paso, con una prueba por vez, construyendo la funcionalidad del sistema.

Programar la prueba primero (Test First Programming) provee dos beneficios principales. En primer lugar es una forma de tener código auto-probado ya que solo se escribe código funcional para que una prueba sea superada. El segundo beneficio es que al pensar en las pruebas primero fuerza a pensar acerca de la interfaz con el código primero. Esto hace focalizarse en la interfaz y en cómo usar una clase ayudando a separar la interfaz de la implementación [Fowler 2006d]

Según David Astels [Astels 2003] TDD es un estilo de desarrollo donde mantienes un juego de pruebas del programador exhaustivo, ninguna parte del código pasa a producción a no ser que pase sus juegos asociados, primero se escriben las pruebas y las pruebas determinan el código que se necesita escribir para que puedan pasarse las pruebas.

Podrían simplificarse los pasos con el siguiente gráfico, ver Figura 2.7.1.

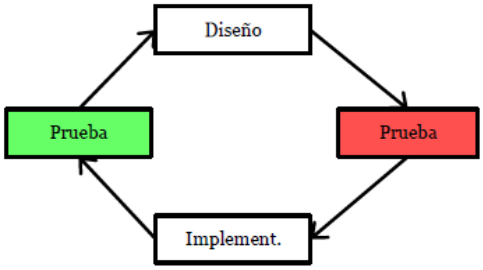


Figura 2.7.1: Proceso simplificado TDD [Kirrily]

Amber utiliza una simple fórmula para explicar en qué consiste TDD, y es la siguiente:

TDD = Refactoring + TFD

TDD cambia completamente el desarrollo tradicional. Al tratar de implementar una nueva funcionalidad primero se pregunta si el diseño actual es el mejor diseño posible para permitir implementar la funcionalidad. En caso afirmativo se procede con un enfoque TFD. En caso contrario se realiza una refactorización localmente para cambiar la porción de diseño afectada por la nueva característica, permitiendo agregar la característica de la manera más simple posible. Como resultado, siempre se mejorará la calidad del diseño, haciendo más fácil el trabajo futuro. [Ambler en AgileData]

Los pasos para TFD pueden verse en un diagrama de actividad, en la siguiente figura (Ver Figura 2.7.2. El primer paso es agregar rápidamente una prueba, básicamente código suficiente para que falle. Luego se corre la prueba para asegurar que falle realmente. Luego se actualiza el código funcional para que pueda pasar las nuevas pruebas. El cuarto paso es correr las pruebas nuevamente. Si fallan se debe actualizar el código funcional y volver a probar. Una vez que las pruebas pasan, se comienza nuevamente, posiblemente se deba re-factorizar cualquier duplicación del diseño, volviendo a TFD en TDD.

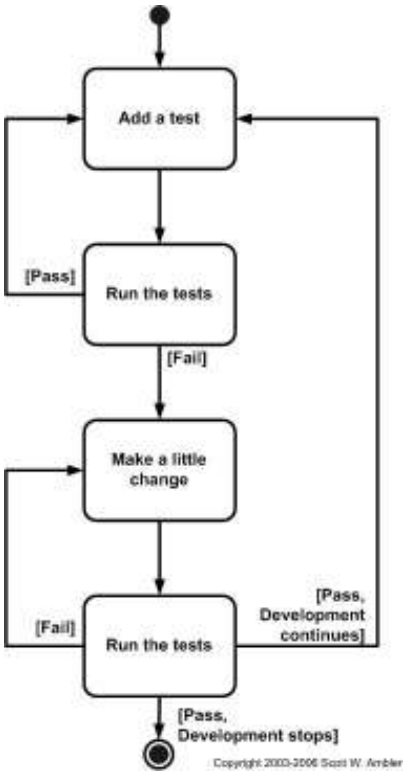


Figura 2.7.2: Proceso de TFD [AgileData web]

2.7.2 Principios de TDD

Existen cuatro principios subyacentes en TDD:

- Mantener un juego exhaustivo de pruebas del programador.

Las pruebas del programador prueban que las clases se comporten de la forma esperada, son escritas por los desarrolladores que escriben el código que será probado. Se llaman pruebas del programador porque aunque se parecen mucho a las pruebas unitarias, se escriben por diferentes motivos. Las pruebas unitarias se escriben para demostrar que el código escrito funciona, en cambio, las pruebas del programador son escritas para definir qué significa que el código funcione. También se llaman así para diferenciarlas de las pruebas escritas por los usuarios.

En TDD se dispone de un gran juego de pruebas, esto es así porque no puede haber código si no existen las pruebas que los prueben. Primero se escriben las pruebas y luego el código que será probado, no hay por tanto código sin pruebas, concluyendo que las pruebas son por tanto exhaustivas.

- Todo código que pasa a producción tiene sus pruebas asociadas.

Esta característica recuerda una propiedad fundamental en TDD y es que una funcionalidad no existe hasta que existe un juego de pruebas que vaya con él. Esta propiedad brinda la oportunidad de utilizar un par de técnicas muy comunes como son el refactoring y la integración continua. Ambas técnicas solo pueden ser ejecutadas si realmente se está seguro de que el código sigue cumpliendo las características definidas en las pruebas.

- Escribir las pruebas primero.

Cuando se tiene una nueva funcionalidad, antes de implementarla se deben escribir sus pruebas. Esta es una de las características más “extremas” de TDD. La manera de proceder es escribir unas pruebas pequeñas y entonces, escribir un poco de código que las ejecute y las pase, luego otra vez se amplían las pruebas, y se vuelve a escribir código, y así sucesivamente.

- Las pruebas determinan el código que tienes que escribir.

Se está limitando la escritura de código a tan solo la prueba que ya se tiene implementada. Solo se escribe lo necesario para pasar la prueba, esto quiere decir que se hace la cosa más sencilla para que funcione.

2.7.3 Roles y responsabilidades.

Los dos roles que como mínimo debe disponer todo proyecto con TDD son clientes y desarrolladores.

- Cliente: Desarrolla las historias con los desarrolladores, las ordena según la prioridad y escribe las pruebas funcionales.

- Desarrollador: Desarrolla las historias con el usuario, las estima, y entonces toma responsabilidad de su desarrollo utilizando TDD, lo que quiere decir que ejecuta las pruebas, implementa y refactoriza.

2.7.4 Prácticas.

Existen dos prácticas muy vinculadas con TDD, que forman parte de XP y fueron mencionadas anteriormente. Sin estas prácticas TDD no podrían realizarse o sería casi una utopía. Las prácticas son: refactoring e integración continua. Ambas fueron descriptas en la sección de XP.

2.7.4.1 Refactoring.

Esta actividad está muy relacionada con TDD ya que muchas veces se duplica código y se introduce mucha "basura" cuando se está intentando programar algo para que pase unas pruebas específicas. Se suele hacer refactorización cuando existe código duplicado, cuando se percibe que el código no está lo suficientemente claro o cuando parece que el código tiene algún problema. Luego de refactorizar se debe correr un juego de pruebas para verificar que no se introdujeron cambios al comportamiento.

Fowler, [Fowler 2006d] recalca que la refactorización es clave en TDD ya que mantiene limpio al código, sino se terminaría con una mezcla de agregaciones de fragmentos de código

2.7.4.2 Integración continúa.

Del mismo modo que con la técnica de refactoring, está íntimamente ligada a TDD, ya que cada vez que se realiza una integración un juego de pruebas exhaustivo se ejecuta y valida el código introducido. Sin estas pruebas, la integración continua no tendría sentido, ya que no garantizaría ni la cohesión, ni la validez del código integrado.

2.7.5 Herramientas

Algunas herramientas fundamentales en el uso de TDD, ya que sin ellas sería imposible realizar las tareas que la metodología propone, son.

- **Pruebas unitarias:** En los últimos años TDD ha popularizado el uso de la familia de herramientas xUnit, que son herramientas que permiten la ejecución de pruebas unitarias de forma automática.

- **Repositorios de código:** Es una herramienta obligatoria para el uso de la metodología TDD. Las más conocidas son CVS o Subversion y facilitan el control de versiones, acceso e integración de código.

- **Software de integración continúa:** Existen diferentes aplicaciones web como Hudson, Apache Gump, Cruise Control, etc. La finalidad de ellas es construir la aplicación a partir del código fuente, normalmente situado en un repositorio de código (subversión, cvs, etc.) y ejecutar las pruebas especificadas. Si las pruebas son satisfactorias se despliega en el entorno de producción, en caso contrario no se incluyen las modificaciones. Este es un proceso que se puede ejecutar varias veces a lo largo del día y garantiza que se realiza la técnica de integración continua.

- **Herramientas de construcción automáticas:** Ayudan y mucho la utilización de herramientas como Maven o Ant para la compilación y ejecución de las pruebas automáticamente.

2.7.6 Conclusiones

Test Driven Development es una de las metodologías con mayor acogida en el campo profesional y que continúa expandiéndose debido a sus buenos resultados. La tendencia actual es integrar TDD independientemente en cualquier metodología ya sea ágil [Schmidkonz 2007] o tradicional [Letelier 2003] y aprovechar los beneficios de practicar una metodología que siempre permite deshacer los errores, asegurar una calidad del producto y protegerse de errores tanto malintencionados como humanos.

Dr. Hakan Erdogmus, editor jefe de IEEE Software [Hartmann 2008], comenta que TDD a veces es entendido como un procedimiento de aseguramiento de lo calidad, provocando que algunos administradores no lo utilicen porque creen que sus equipos ya tienen otros procedimientos que garantizan la calidad. Hakan hace hincapié en que originalmente TDD fue pensado como una técnica para mejorar la productividad y que el aumento de la calidad es un efecto secundario, si bien actualmente se reconoce a TDD como una metodología ágil en si misma.

2.8 DSDM (DYNAMIC SYSTEMS DEVELOPMENT METHOD)

2.8.1 Introducción

Unos años antes de la aparición de XP, en enero de 1994, se reunieron en Londres, Reino Unido, un grupo de profesionales para discutir sobre la creación de un proceso iterativo normalizado para el desarrollo RAD. RAD son las siglas de Rapid Application Development, un método de desarrollo creado por James Martin que se caracteriza por usar un ciclo de vida iterativo, prototipos y herramientas CASE (Computer Aided Software Engineering). Se formó un consorcio sin fines de lucro e independiente. Este consorcio se dedicó a entender las mejores prácticas en el desarrollo de aplicaciones y codificándola de tal forma que fuera ampliamente enseñado e implementado. [Stapleton 1997]. El objetivo era desarrollar y promover de manera conjunta un framework RAD independiente combinando sus mejores experiencias prácticas.

El resultado del trabajo de este consorcio, después de más de un año, fue DSDM (Dynamic Systems Development Method), una forma de desarrollar sistemas de aplicación que realmente satisfaga las necesidades del negocio. [Stapleton 1997]. La versión 1 fue publicada en febrero de 1995. Este resultado fue un método genérico que cubre personas, proceso y herramientas y que fue formado a partir de las experiencias de organizaciones de todo tipo de sector y tamaño. DSDM Consortium es dueña y administra el framework DSDM y solo sus miembros pueden emplearlo con fines comerciales. [Navegapolis web]. Habiendo empezado con 17 fundadores ahora tiene más de mil miembros y ha crecido fuera de sus raíces británicas. Siendo desarrollado por un consorcio, tiene un sabor diferente a muchos de los otros métodos ágiles. Tiene una organización de tiempo completo que lo apoya con manuales, cursos de entrenamiento, programas de certificación y demás.

DSDM contempla el ciclo de vida iterativo e incremental, involucrar continuamente al usuario y la adaptación al cambio. [Frankel 2004]. Tal como lo expone claramente [Caine 2012], es la única metodología ágil que cubre el ciclo de vida del proyecto entero. Incorpora disciplinas de la administración de proyecto y provee mecanismos para asegurar que los beneficios del proyecto sean claros, que sea factible la solución propuesta y que haya sólidas bases antes de comenzar con el trabajo detallado.

Se considera a DSDM como el primer método ágil [Frankel 2004]. Estuvo representada en la firma del Manifiesto Ágil y recientemente ha tenido un despertar en su popularidad, especialmente en el Reino Unido. Esto es porque la APMG-International, los fundadores de Prince2, eligieron combinar DSDM Atern y PRINCE2 para producir un nuevo título: Administración de Proyecto Ágil (Agile Project Management - APM). El gobierno del Reino Unido está buscando adoptar APM para todos sus proyectos IT. Se volvió popular en Europa, si bien actualmente tiene presencia en Inglaterra, Estados Unidos, Benelux, Dinamarca, Francia y Suiza; y con interés y contactos para futuras representaciones en Australia, India y China [Navegapolis web].

Como todo enfoque ágil, DSDM ha ganado a través de la experiencia de los años. Dane Falker, director de DSDM Consortium de Estados Unidos en 2001 afirma que si bien el acrónimo se mantiene igual, las palabras y significado han cambiado, ver Figura 2.8.1. [Highsmith]

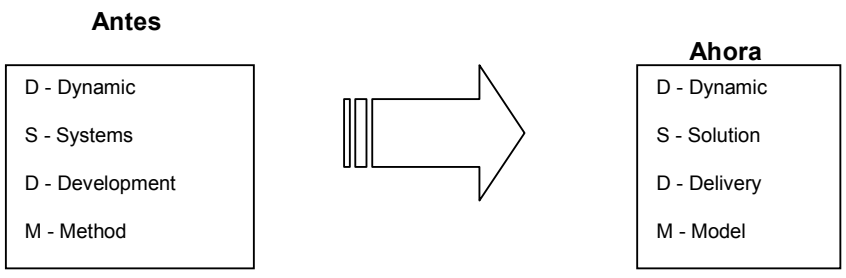


Figura 2.8.1. Acrónimo de DSDM. Elaborado según [Highsmith]

La primera D del acrónimo por “Dinámico” refleja la habilidad para adaptarse a los cambios. La S ahora refleja el enfoque en las “Soluciones” del negocio más que “Sistemas”. DSDM se enfoca en las soluciones al cliente y el valor de negocio. La segunda D refleja “Entrega” (Delivery), un concepto más amplio que “Desarrollo” e indica la importancia de los productos entregables (características o historias) más que las tareas tradicionales. Finalmente la M representa más un “Modelo” que un “Método”, reflejando una perspectiva de negocio en los proyectos. Por lo tanto, podría decirse según Falker que DSDM es un Modelo de entrega de soluciones dinámicas. [Highsmith]

DSDM enfatiza el uso de Workshops facilitadores tanto en las fases de Estudio de Negocio como del Modelo Funcional para involucrar los clientes y usuarios claves en tener el proyecto iniciado apropiadamente. [Highsmith]

2.8.2 Contexto

El modelo ágil que más áreas comprende es DSDM. Es la más veterana de las metodologías ágiles, y la más próxima a los métodos formales; de hecho, la implantación de un modelo DSDM en una organización, la lleva a alcanzar lo que en CMM (modelo no ágil) sería un nivel 2 de madurez, dejando por tanto fuera de su ámbito los objetivos y práctica de los niveles 3, 4 y 5. Surgió en 1994 de los trabajos de Jennifer Stapleton, la actual directora del DSDM Consortium. [Navegapolis 2010]

La idea fundamental de DSDM se basa en que en vez de fijar las funcionalidades de un producto primero y después el tiempo y el coste, fija primero el tiempo y el coste y con esto fijado, determina las funcionalidades que se pueden implementar en el producto.

En común con los métodos ágiles, DSDM considera imprescindible una implicación y una relación estrecha con el cliente durante el desarrollo, así como la necesidad de trabajar con métodos de desarrollo incremental y entregas evolutivas.

DSDM cubre los aspectos de gestión de proyectos, desarrollo de los sistemas, soporte y mantenimiento y se autodefine como un marco de trabajo para desarrollo rápido más que como un método específico para el desarrollo de sistemas. [Navegapolis 2010]

Es frecuente que DSDM se implante en combinación con XP o Prince2, siendo este último un modelo predictivo tal como lo es PMBOK utilizado por el gobierno del Reino Unido como el estándar de administración de proyectos para proyectos públicos. [Navegapolis web]

2.8.2.1 DSDM Atern [DSDM web]

DSDM ha evolucionado a través de los años, desde aquella primera versión en 1995. En 2001, año del Manifiesto Ágil, DSDM publicó la versión 4.1 de su modelo, y se consideró una metodología ágil; y aunque mantuvo las siglas, cambió la denominación original Dynamis Systems Development Method por *Framework for Business Centred Development*. En mayo de 2003 se lanzó la versión 4.2 de DSDM que sigue siendo ampliamente utilizada y sigue siendo válida. En 2007 surge la versión más reciente de DSDM que se llama DSDM *Atern*. El nombre Atern es una abreviación de Arctic Tern - un pájaro de colaboración que puede viajar grandes distancias y resume muchos aspectos de la metodología que es su forma natural de trabajar, por ejemplo establecimiento de prioridades y la colaboración. [Navegapolis web]

DSDM Atern es un framework ágil, robusto y probado para la efectiva administración del proyecto y entrega [DSDM web]. Se lo puede ver en forma gratuita y está complementado por un conjunto completo de materiales, handbook y plantillas. Además el DSDM Consortium brinda el soporte.

El Atern Agile framework y sus prácticas han estado largo tiempo a la cabeza de la entrega de proyectos exitosos; experiencias del mundo real que asegura que los objetivos de tiempo, calidad y costos están siempre administrados y son alcanzables. Se diseñó Atern para que sea fácilmente adaptado y usado en conjunción con métodos de entrega (delivery methods) como PRINCE2® y demostró que complementa y mejora apropiadamente sistemas de administración de calidad trabajados incluyendo aquellos que cumplen CMMI e ISO9001.

Atern es un framework de entrega de proyecto ágil que entrega la solución adecuada en el tiempo adecuado. Se entrega la solución de negocio correcta porque [Caine web]:

- El equipo y los stakeholders importantes se mantienen enfocados en el resultado del negocio.
- Se entrega a tiempo asegurando un retorno temprano en la inversión.
- Todas las personas involucradas en el proyecto trabajan colaborativamente para entregar la solución óptima.
- Se prioriza el trabajo según la necesidad del negocio y la habilidad de los usuarios para acomodar los cambios en la escala de tiempo acordada
- Atern no compromete la calidad, por ejemplo la solución no está ni subdiseñada ni sobrediseñada

El DSDM Consortium brinda el soporte y certifica Atern; la misión del DSDM Consortium sin fines de lucro es promover las mejores prácticas en la entrega de un proyecto ágil

2.8.3 Método de Desarrollo de Sistema Dinámico (DSDM)

Como se viene mencionando, DSDM se caracteriza por su rapidez de desarrollo atendiendo a las demandas de tecnología de forma eficaz y eficiente previendo que transcurra mucho tiempo y la tecnología cambie. Es una metodología ágil situada dentro de las RAD (Rapid Aplication Development) y es ideal para proyectos de sistemas de información cuyos presupuestos y agendas son muy apretados.

DSDM trata de evitar la falta de participación de los usuarios, la limitación en las oportunidades de cooperación y colaboración, sistemas de baja calidad que no cumplen con los requisitos de los usuarios, etc., todos estos son problemas que los grupos han encontrado. DSDM consiste en técnicas de desarrollo y gestión del proyecto en la misma metodología.

2.8.3.1 Variables de un proyecto [DSDM web]

La mayoría de los proyectos tienen cuatro parámetros: tiempo, costo, características y calidad. Tratar de ajustar todos los parámetros en la salida del proceso es imposible y es la causa de muchos problemas comunes.

En enfoques tradicionales de administración de proyecto (Figura 2.8.2 Diagrama de la izquierda) se fijan las características de la solución y el tiempo y costo están sujetos a variación. Si el proyecto se sale de su curso generalmente se agregan más recursos o se extiende la fecha de entrega. Pero agregar recursos a un proyecto ya atrasado, lo atrasa más. No cumplir con un plazo es desastroso desde una perspectiva de negocio y puede dañar la credibilidad fácilmente. La calidad frecuentemente es una casualidad y se vuelve también una variable acompañada de entrega tardía y aumento de costos.

El enfoque Atern para administración de proyectos (Figura 2.8.2 Diagrama de la derecha) fija el tiempo, el costo y la calidad en la Fase de fundamentación mientras la contingencia se maneja variando las características a entregar. Si es necesario, se descartan o difieren características de menor prioridad con el consentimiento de todos los stakeholders. Así, un proyecto Atern siempre entrega una solución viable y se garantiza por lo menos la entrega de un conjunto mínimo de características en tiempo y presupuesto

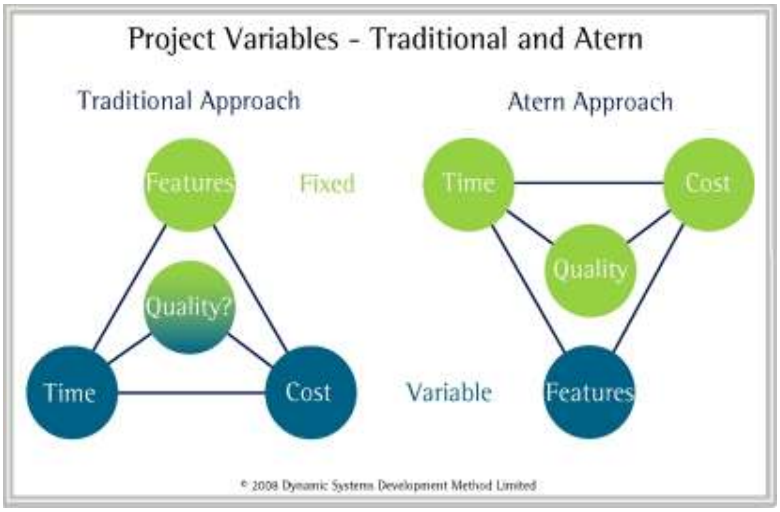


Figura 2.8.2: Variables de un Proyecto [DSDM Consortium]

En un proyecto Atern, se fija la calidad porque los criterios de aceptación se consensuan y establecen antes de comenzar el desarrollo A diferencia de los proyectos tradicionales, los costos, tiempos y calidad se fijan en etapas tempranas del proyecto. Contingencias, en la forma de características de menor prioridad, aseguran que se puede alcanzar a realizar la entrega a tiempo de una solución viable protegiendo un subconjunto mínimo de usabilidad y descartando o posponiendo características de menor prioridad, si es necesario.

2.8.4Los principios de DSDM

DSDM tiene principios subyacentes que incluyen una interacción activa del usuario, entregas frecuentes, equipos autorizados, pruebas a lo largo del ciclo. Como otros métodos ágiles, usan ciclos de plazos cortos de entre dos y seis semanas. Hay un énfasis en la alta calidad y adaptabilidad hacia requisitos cambiantes. DSDM es notable por tener mucha de la infraestructura de las metodologías tradicionales más maduras, al mismo tiempo que sigue los principios de los métodos ágiles. DSDM claramente evoca a las ideas de un enfoque de desarrollo exploratorio, remarca que los usuarios del sistema no pueden visualizar todos sus requerimientos al principio y recomienda un enfoque iterativo “el paso actual solo necesita completarse lo suficiente para pasar al siguiente paso” [Highsmith]

En Atern se usan los principios para proveer una guía a través del proyecto. Hay 8 principios y todo el framework completo puede derivarse a partir de ellos. En Atern existe una diferencia respecto a DSDM 4.2 ya que para la versión DSDM 4.2 se definieron 9 principios. Los principios están basados en las mejores prácticas en el verdadero sentido. Definen “la forma en que deben hacerse las cosas” [DSDM web]. No cumplir con uno de ellos podría llevar al fracaso ya que son los bloques básicos sobre los que se apoya DSDM Atern [DSDM web].

En la siguiente tabla se detallan los principios de Atern y una breve descripción.

Icono	Principio	Descripción
	Enfocarse en las necesidades del negocio	Entregar lo que el negocio necesita cuando lo necesita. Las verdaderas prioridades del negocio deben entenderse.
	Entregar a tiempo	Planear los tiempos de duración anticipadamente y definir el marco de tiempo. Las fechas nunca se cambian; se varían las características dependiendo de las prioridades del negocio para cumplir con los plazos.
	Colaborar	Los equipos trabajan en un espíritu cooperativo y comprometido. La colaboración alienta a entender, ir más rápido y compartir la propiedad. El equipo debe tener poder de decisión e incluir representantes del negocio.
	Nunca comprometer la calidad	Una solución debe ser “suficientemente buena”. El nivel de calidad se define al principio. Los proyectos se deben testear temprana y continuamente y revisar constantemente.
	Construir incrementalmente a partir de base sólidas	Los incrementos permiten que el negocio tome ventaja del trabajo antes de que el producto final esté completo, dándole confianza a los stakeholders y brindando retro-alimentación. Esto está basado en hacer solo el análisis suficiente para proceder y aceptar que pueden surgir detalles más tarde.
	Desarrollar iterativamente	Aceptar que el trabajo no siempre está bien la primera vez. Usar plazos de tiempo fijos para permitir cambios y continuamente confirmar que la solución es la correcta.
	Comunicarse continua y claramente	Usar talleres, reuniones daily standups, modelado de prototipos, presentaciones y promover comunicación informal cara a cara.
	Demostrar control.	El equipo necesita ser proactivo al monitorizar y controlar el progreso respecto a las bases definidas. Constantemente necesitan evaluar la viabilidad del proyecto basado en los objetivos del negocio.

Tabla 2.8.1. Principios de DSDM Atern. Traducido de [Caine web]

2.8.5 Roles.

DSDM define tres grupos de roles: roles del proyecto, roles del desarrollo de la solución y otros roles El siguiente diagrama (Figura 2.8.3), conocido como “bebé alienígena” o “alien baby” es el diagrama estándar de DSDM Atern que ilustra estos tres grupos de roles.

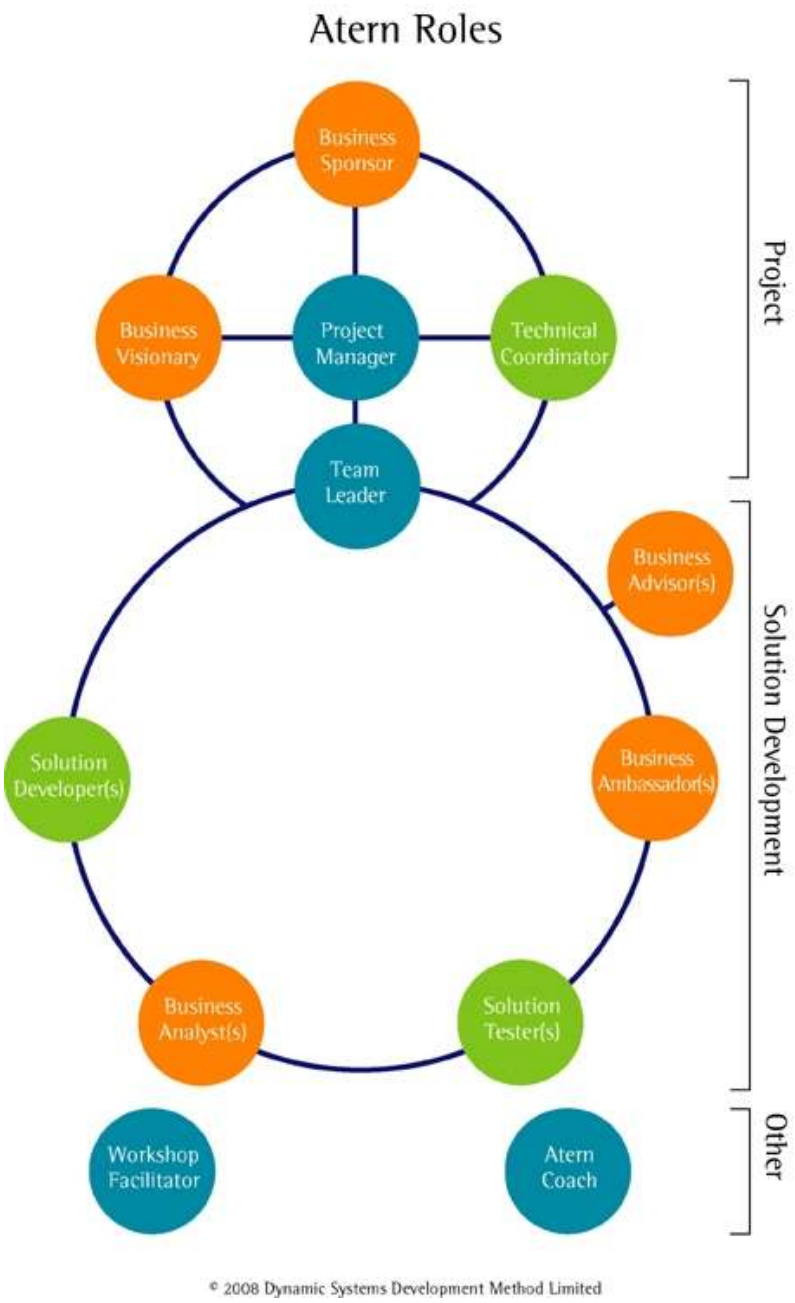


Figura 2.8.3. Roles en DSDM [Caine web]

En la siguiente tabla (ver Tabla 2.8.2) se describe cada uno de ellos.

Rol	Responsabilidades Claves
Sponsor de negocio (Business Sponsor)	Es dueño del negocio. Asegura los fondos y recursos. Garantiza la toma de decisiones efectiva y se ocupa de las priorizaciones rápidamente
Administrador de Proyecto (Project Manager)	Gobierna el proyecto. Planifica a alto nivel. Monitorea el progreso, recursos disponibles, configuración del proyecto, administra los riesgos y situaciones que puedan surgir.
Visionario del negocio (Business Visionary)	Tiene la visión del negocio y el impacto en cambios del negocio más amplios. Monitorea el progreso frente a la visión. Contribuye en las sesiones de los requerimientos claves, diseño y revisión.
Coordinador Técnico (Technical Coordinator)	Acuerda y controla la arquitectura técnica. Aconseja y coordina a los equipos. Identifica y administra riesgos técnicos. Asegura que se alcancen los requerimientos no funcionales.
Jefe del equipo (Team Leader)	Enfoca al equipo para entregar a tiempo. Alienta a la participación de todo el equipo. Maneja los tiempos fijados para actividades detalladas y actividades día a día. Garantiza que las actividades de prueba y revisión sean programadas y completadas.
Embajador de Negocio (Business Ambassador)	Contribuye en las sesiones de requerimientos, diseño y revisión. Provee la visión del negocio para realizar la toma de decisiones día a día. Describe escenarios de negocio para ayudar a diseñar y probar la solución. Provee la seguridad de que la solución es correcta. Coordina la aprobación del negocio.
Desarrollador de la solución (Solution Developer)	Crea la solución y participa por completo en todas las actividades de QA apropiadas.
Probador de la solución (Solution Tester)	Trabaja con otros roles del negocio para definir los escenarios de prueba para la solución. Realiza reportes completos de resultados de pruebas técnicas al líder de proyecto y al Coordinador técnico.
Analista del Negocio (Business Analyst)	Apoya la comunicación entre miembros técnicos y del negocio del equipo. Administra todos los productos requeridos relacionados a los requerimientos del negocio. Asegura que las implicancias del negocio de las decisiones diarias sean pensadas apropiadamente.
Asesor de Negocio (Business Advisor)	Provee entradas especializadas, por ejemplo un contador. Generalmente un futuro usuario de la solución.
Entrenador Atern (Atern Coach)	Ayuda a los equipos nuevos con Atern a sacarle el mayor provecho. Adapta Atern a las necesidades del proyecto.
Facilitador de taller (Workshop Facilitator)	Maneja y organiza talleres. Responsable del contexto no del contenido. Independiente.
Otros especialistas	Expertos requeridos por poco tiempo, posiblemente por cuestiones técnicas. Por ejemplo Especialistas en load-test, etc.

Tabla 2.8.2: Roles en DSDM Atern [DSDM web]

Los principales beneficios del enfoque DSDM se derivan de los usuarios del negocio (clientes) que participan activamente en los equipos de desarrollo.

2.8.6Proceso Atern [Caine web]

Atern difiere como se mencionó anteriormente de otros enfoques ágiles en que cubre todo el ciclo de vida del proyecto. Incorpora disciplinas de la administración de proyectos, y provee mecanismos para asegurar que los beneficios del proyecto sean claros, la solución propuesta sea factible y haya bases sólidas antes de comenzar el trabajo detallado. En Atern existe una diferencia respecto a DSDM 4.2 sobre como mirar el proceso de desarrollo. Hay 7 fases en un proyecto Atern (ver Figura 2.8.4 y descripción en Tabla 2.8.3).

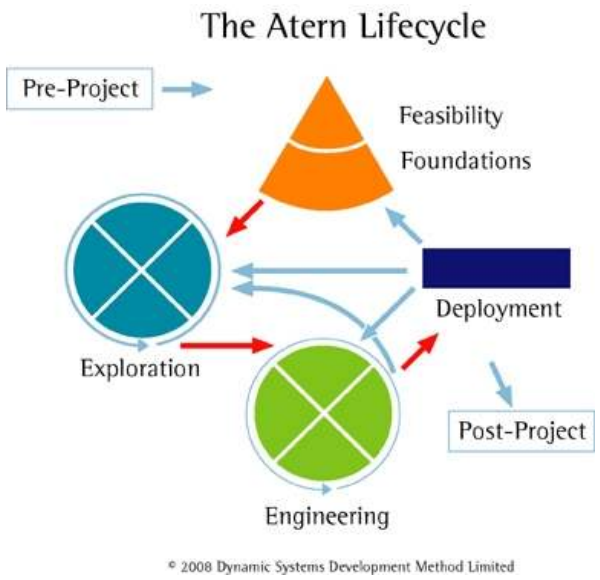


Figura 2.8.4: Fases de Atern [Caine web]

2.8.6.1.1 FASES	2.8.6.1.2 DESCRIPCIÓN
Pre-proyecto	Iniciación del proyecto, acordando los Términos de Referencia para el trabajo.
Factibilidad	Generalmente una fase corta para evaluar la viabilidad y realizar un bosquejo del caso de negocio (justificación).
Bases (fundamentación)	Fase clave para asegurar que se entiende el proyecto y está definido lo suficiente para delinear el alcance a un alto nivel y los componentes tecnológicos y estándares acordados antes de que comiencen las actividades de desarrollo.
Exploración	Fase de desarrollo iterativo durante la que el equipo expande los requerimientos de alto nivel para mostrar la funcionalidad.
Ingeniería	Fase de desarrollo iterativo donde se elabora la solución que puede usarse para la entrega.
Despliegue	Por cada incremento (conjunto de plazos prefijados) del proyecto se obtiene una solución disponible.
Post-proyecto	Evalúa los beneficios alcanzados.

Tabla 2.8.3 Fases de DSDM Atern. Traducido de [DSDM web]

Las fases de exploración e ingeniería a menudo se combinan ya que el método es flexible, permitiendo a los equipos organizarse de la mayor forma para alcanzar la solución. Algunos ejemplos, presentados en [DSDM web] se muestran a continuación (Ver Figura 2.8.5).

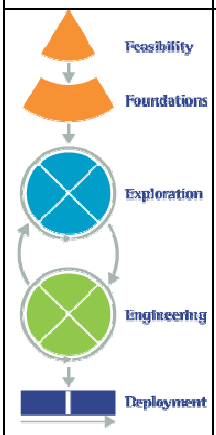
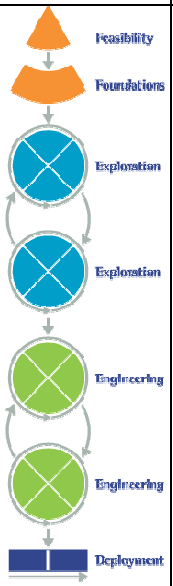
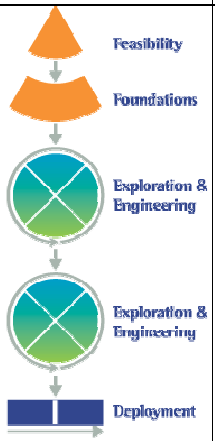
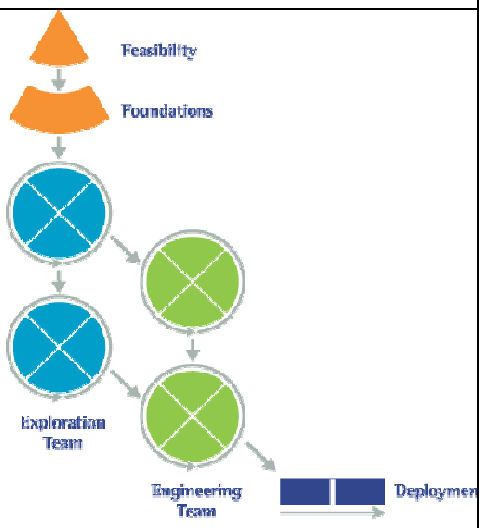
Ejemplo1	Ejemplo2	Ejemplo3	Ejemplo4
			
Realiza iteraciones entre las fases de exploración e ingeniería antes de producir un incremento	Completa todas las actividades exploratorias antes de comenzar con las de ingeniería	Se combina el trabajo de Exploración e Ingeniería	Escenario más complejo con dos equipos involucrados. Un equipo se centra en fase exploratoria y otra en la de ingeniería

Figura 2.8.5: Ejemplos de adaptaciones de DSDM Atern a un proyecto específico [DSDM web]

2.8.7 Entregables

Los entregables están asociados con cada fase del ciclo de vida y se denominan productos (ver Figura 2.8.6). No se requieren todos los productos para cada proyecto y la formalidad dependerá del proyecto y la organización. Algunos productos son específicos de una fase particular y otros pueden seguir evolucionando a través de fases subsecuentes.

El flujo básico de los productos a través del ciclo de vida se muestra a continuación. Por ejemplo, el Estudio de Factibilidad (Feasibility Assessment) se completa con las Bases del negocio (Business Foundations) y la Lista priorizada de Requerimientos (PRL). De manera similar el Similarly, bosquejo de plan (Outline Plan) se refina en el Plan de Entrega (Delivery Plan) para el proyecto que el equipo va refinando por turnos para crear el Plan de Tiempo Prefijado individual y el plan de Despliegue para un incremento.

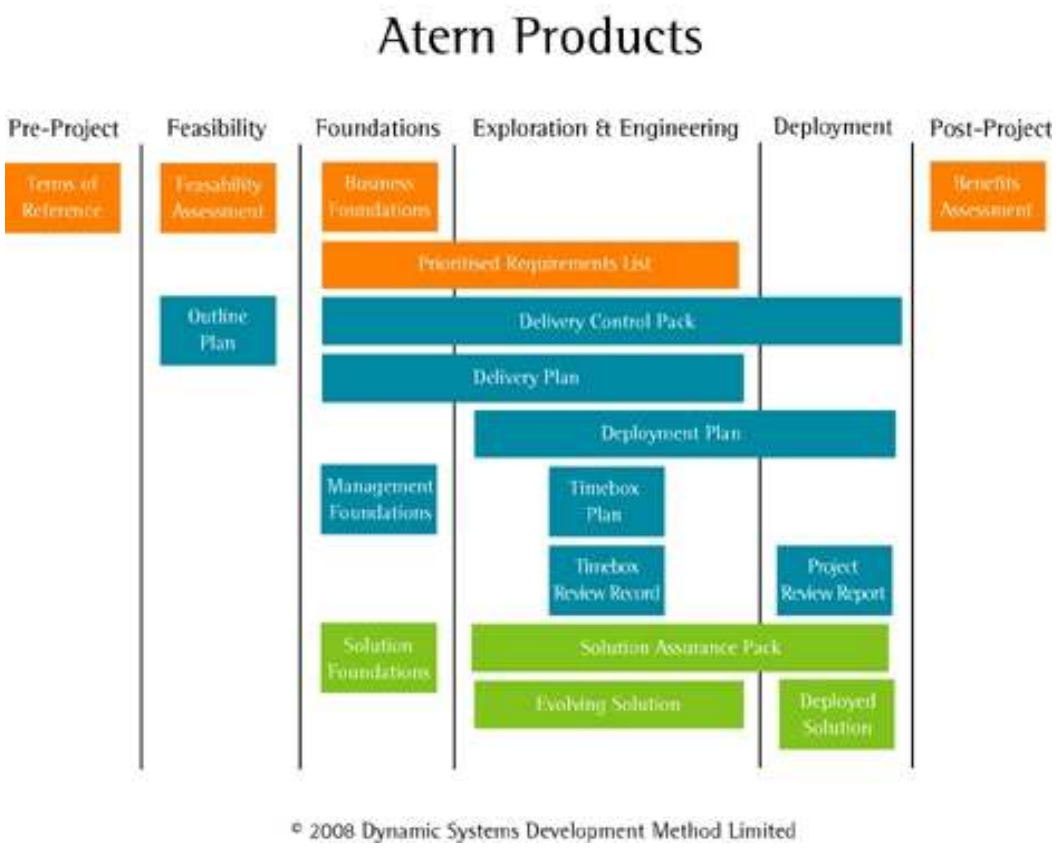


Figura 2.8.6: Productos Atern [DSDM web]

Atern permite al equipo decidir qué productos construir o como deben verse, permitiendo adaptar los productos a la mayoría de los entornos. Algunos entornos podrían requerir todos los productos y otros solo el PRL y la Solución que va evolucionando (similar a Scrum).

2.8.8Conclusiones

DSDM es una metodología ágil que abarca todo el cido de vida de un proyecto de desarrollo. Usa un ciclo iterativo para hacer evolucionar la solución apropiada para satisfacer los objetivos del proyecto. Al dividir el proyecto en períodos cortos de tiempo (timeboxes), cada uno con salidas esperadas muy claras, el Administrador del proyecto y los propios miembros del equipo pueden ejercer el control.

Se definen claramente los roles y se divide el trabajo en time-boxes con fechas de entrega inamovibles y resultados preacordados. El tamaño de los equipos en DSDM varía desde un mínimo de dos integrantes hasta seis, pudiendo coexistir múltiples equipos en un mismo proyecto. [Carvajal 2008]

Atern puede usarse para complementar otras disciplinas de administración de proyecto como PRINCE2™ y PMI (Project Management Institute) sin duplicar esfuerzos. Atern puede incorporar otros enfoques ágiles, como XP y Scrum para proveer el framework de agilidad necesario para permitir una entrega controlada de proyectos ágiles.

2.9 CONCLUSIONES METODOLOGIAS AGILES

No existe una metodología universal para hacer frente con éxito a cualquier proyecto de desarrollo de software. Toda metodología debe ser adaptada al contexto del proyecto: recursos técnicos y humanos, tiempo de desarrollo, tipo de sistema, etc. [Letelier 2006]

Históricamente, las metodologías tradicionales han intentado abordar la mayor cantidad de situaciones de contexto del proyecto, exigiendo un esfuerzo considerable para ser adaptadas, sobre todo en proyectos pequeños y con requisitos muy cambiantes. Las metodologías ágiles ofrecen una solución casi a medida para una gran cantidad de proyectos que tienen estas características. Una de las cualidades más destacables en una metodología ágil es su sencillez, tanto en su aprendizaje como en su aplicación, reduciéndose así los costos de implantación en un equipo de desarrollo. Esto ha llevado hacia un interés creciente en las metodologías ágiles. [Letelier 2006]

Falta aún un cuerpo de conocimiento consensuado respecto a los aspectos teóricos y prácticos de la utilización de metodologías ágiles, así como una mayor consolidación de los resultados de aplicación. La actividad de investigación está orientada hacia líneas tales como: métricas y evaluación del proceso, herramientas específicas para apoyar prácticas ágiles, aspectos humanos y de trabajo en equipo.

3 DESARROLLO BASADO EN CONOCIMIENTO

3.1 INTRODUCCIÓN AL DESARROLLO BASADO EN CONOCIMIENTO

3.1.1 Nuevo Paradigma: Describir en vez de Programar

> *Se pretende "describir" en vez de "programar". Se pretende maximizar las descripciones declarativas y minimizar las especificaciones procedurales. [Gonda 2010]*

Según Gonda Breogán y Jodal Nicolás, hasta el 2006 la complejidad de los sistemas había aumentado en un 2000% (datos al 2006) la productividad aunque los lenguajes de programación solo aumentaron un 150%. Esta situación hace que los costos crezcan de una manera desmesurada, considerando los costos como una composición de tiempo y dinero.

En la actualidad muchas empresas internacionales contratan personal para el desarrollo y mantenimiento de sus sistemas en países de bajos salarios, así influyen en la variable *dinero*. Pero, en realidad, el esfuerzo de realización de las tareas sigue siendo el mismo ya que el trabajo involucrado sigue siendo el mismo. Se podría considerar que simplemente se cambió la unidad de medida, en estas circunstancias el precio de la hora-hombre es mucho menor. Por lo tanto si se mide el costo en función del dinero, se produce una disminución. Pero, si se considera la dimensión tiempo, ésta no se modifica.

El *tiempo* es crucial a la hora de desarrollar sistemas y mantenerlos. Por otro lado, los sistemas de software necesitan responder a nuevas necesidades: nuevos dispositivos, nuevos usuarios, nuevas modalidades de uso, nuevas posibilidades de integración y, por todo esto, se hacen cada día más y más complejos y los negocios están sujetos a un “time to market” cada vez más crítico y que no pueden cambiar. Como consecuencia, cada día más negocios se están perjudicando por los tiempos inadecuados de desarrollo de las nuevas soluciones y de mantenimiento de las actuales. Debido a todo esto es evidente que es necesario un aumento de la productividad a través de tecnologías de muy alto nivel.

La solución no está relacionada tanto en mejorar más todavía los lenguajes de programación sino en la programación en sí. Hoy la enorme mayoría de los sistemas se desarrollan y mantienen con programación manual. Si se "describe" en vez de "programar", se pueden maximizar las descripciones declarativas y minimizar las especificaciones procedurales, haciendo *desarrollo basado en conocimiento* y no en programación. Esta pretensión constituye un cambio esencial de paradigma e implica un choque cultural.

3.1.1.1 Metodologías tradicionales de desarrollo y problemas asociados [Artech 2008]

La forma tradicional de desarrollar aplicaciones parte de una premisa básica: es posible construir un modelo de datos estable de la empresa. Basándose en esa premisa, la primera tarea que se encara es el análisis de datos, donde se estudia la realidad en forma abstracta y se obtiene como producto el modelo de datos de la empresa. La segunda tarea es diseñar la base de datos. Es muy sencillo diseñar la base de datos partiendo del modelo de datos ya conocido. Una vez que se ha estudiado la realidad desde el punto de vista de los datos, se hace lo propio desde el punto de vista de las funciones (análisis funcional). Sería deseable que el estudio de la realidad tuviera como producto una especificación funcional que dependiera sólo de dicha realidad. Lo que se hace en las metodologías más usadas, sin embargo, es obtener una especificación funcional que se refiere a los archivos de la base de datos (o bien a las entidades del modelo de datos, lo que es esencialmente equivalente). Una vez que se tiene la base de datos y la especificación funcional, se

pasa a la implementación de las funciones, existiendo tradicionalmente para ello varias opciones (lenguajes de 3ª o 4ª generación, generadores, interpretadores). En la siguiente figura se planta un bosquejo de una metodología tradicional. Sin embargo, todas las formas de implementación vistas tienen un problema común y es que parten de la enunciada premisa: es posible construir un modelo de datos estable de la empresa, y esta premisa es falsa.

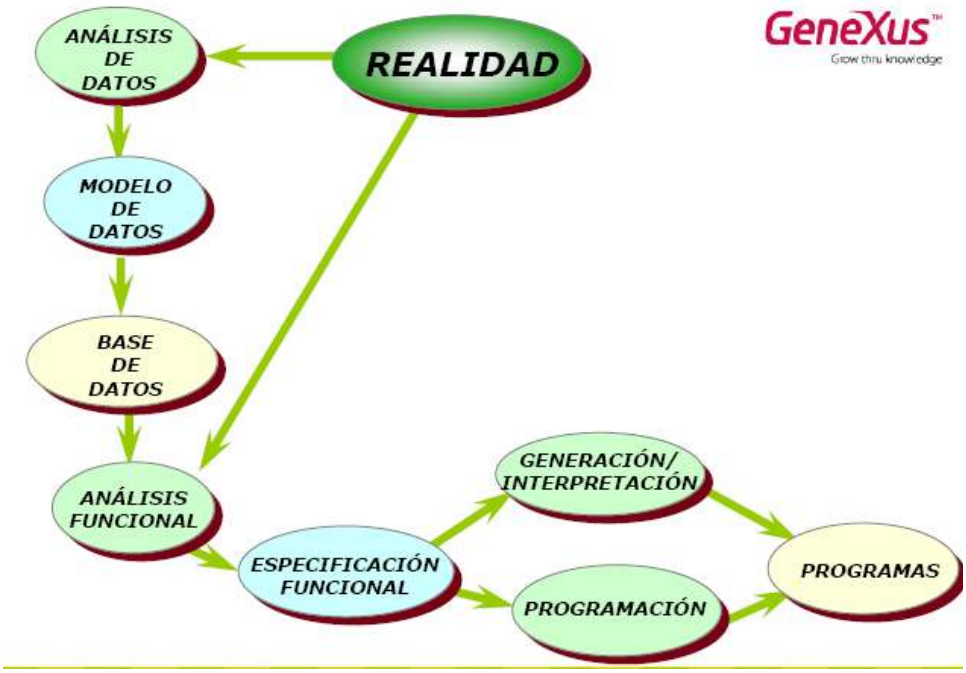


Figura 3.1.1: Metodología Tradicional [Genexus web]

Realmente es imposible hacer, de una forma abstracta, un modelo de datos detallado de la empresa y con el suficiente nivel de detalle y objetividad, porque nadie la conoce como un todo. Por ello es necesario recurrir a múltiples interlocutores, y cada uno de ellos proyecta sobre el modelo, su propia subjetividad. Una consecuencia de esto es que, durante todo el ciclo de vida de la aplicación, se producen cambios en el modelo. Pero aún si se considerara la situación ideal, donde se conocen exactamente las necesidades y, entonces, es posible definir la base de datos óptima, el modelo no podrá permanecer estático porque deberá acompañar la evolución de la empresa. Todo esto sería poco importante, si la especificación funcional y la base de datos fueran independientes. Sin embargo, dado que la especificación funcional se refiere a la base de datos, las inevitables modificaciones en ésta implican la necesidad de modificaciones (manuales) en aquella.

La mayor consecuencia de lo anterior está constituida por los muy altos costos de mantenimiento: en la mayoría de las empresas que trabajan de una manera convencional se admite que el 80% de los recursos que teóricamente están destinados al desarrollo, realmente se utilizan para hacer mantenimiento de las aplicaciones ya implementadas. Cuando se trata de aplicaciones grandes la situación es aún peor: este mantenimiento comienza mucho antes de la implementación, lo que hace que los costos de desarrollo crezcan en forma hiperlineal con respecto al tamaño del proyecto.

Dado que es muy difícil, en este contexto, determinar y propagar las consecuencias de los cambios de la base de datos sobre los procesos, es habitual que, en vez de efectuarse los cambios necesarios, se opte por introducir nuevos archivos redundantes, con la consiguiente degradación de la calidad de los sistemas y el incremento de los costos de mantenimiento.

3.1.1.2 ¿Paradigma orientado al conocimiento? [Gonda 2003]

En los últimos años se han solidificado los sistemas de gestión de base de datos relacionales: hoy es impensable hablar de otro tipo de Sistema de Gestión de Base de Datos. Se han configurado estándares robustos que, utilizados estrictamente, permiten la portabilidad de las aplicaciones. La existencia de esos estándares dificulta la adopción de modificaciones sustanciales a los mismos. En particular, es muy poco probable que se tenga en un futuro previsible bases de datos cualitativamente mucho más evolucionadas.

Es previsible, en cambio, que se siga dando cada vez más solidez: disponibilidad, seguridad, eficiencia, escalabilidad, optimización de recursos y que se neutralicen las nuevas amenazas como las representadas por virus innovadores.

Paralelamente se ha generalizado la programación orientada a objetos (en particular fuertemente impulsada por el previsible auge de las plataformas Java y .net) y es vital que se resuelva rápidamente el problema de la vinculación natural de los programas con la base de datos (Object Relational Mapping).

Hay grupos de informáticos que creen en “bases de datos inteligentes” que permitan alimentar en forma declarativa todo el conocimiento necesario (fundamentalmente “reglas del negocio”, “reglas de precedencia y/o de flujo” y “reglas de autorización” con toda la generalidad que esos conceptos representan) de manera que cualquier usuario sin la necesidad de conocimiento técnico alguno, de una manera simple, pueda hacer en cualquier momento todo aquello que quiera y esté autorizado a hacer, sin necesidad de programar. Este es un nuevo paradigma.

3.1.1.3 Nuevo Paradigma: desarrollo basado en conocimiento [Gonda 2010]

En los últimos años se ha hablado mucho en la industria de la administración del conocimiento (Knowledge Management**)** y, dentro de este rótulo se han colocado muchas cosas que están bien distantes del Desarrollo Basado en Conocimiento al que se pretende referir en este trabajo. La ingeniería del conocimiento se la considera frecuentemente como una mezcla de Inteligencia Artifical e Ingeniería de software. La ingeniería del conocimiento busca definir metodologías que permitan construir sistemas basados en conocimiento de manera sistemática y controlable [Knublauch 2002]. En los últimos años, la definición de Ingeniería del Conocimiento (knowledge engineering - KE) evolucionó, de considerarse un proceso de transferencia de conocimiento experto en un formato computacional que pueda ser usado por agentes inteligentes, hacia una perspectiva basada en modelos de la ingeniería del software. Estos modelos pueden usarse para estructurar el conocimiento de tal forma que las aplicaciones pueden emular efectivamente las capacidades de un dominio de experto. [Borlawsky 2009]. Estas definiciones están bien distantes del Desarrollo Basado en Conocimiento al que se pretende referir en este trabajo.

Generalmente la industria se ha referido a maneras de organizar y/o acceder el conocimiento para ser utilizado de una forma tradicional por los seres humanos. Provee los medios para la recolección, organización y recuperación computarizada de conocimiento Se trata de una versión actualizada, utilizando la tecnología actualmente disponible, de los libros (y que es de enorme utilidad para toda la humanidad): se accede a un cierto conocimiento leyendo un libro y, en nuestra mente, se hacen razonamientos sobre ese conocimiento lo que, eventualmente, determina acciones. Los buscadores de texto inteligentes que están disponibles desde hace pocos años hacen que este conocimiento sea cada vez más accesible a los seres humanos.

Como característica general, este conocimiento no es “entendible” por una máquina y, en consecuencia, no es operable. Adicionalmente, como el razonamiento de los seres humanos puede lidiar razonablemente (dentro de ciertos límites) con la ambigüedad y, aún, con la inconsistencia, este conocimiento muchas veces no es riguroso. [Artech 2008].

Los sistemas basados en conocimiento (SBC) tratan de mantener una gran cantidad de conocimiento y aportan mecanismos para manejarlo. La representación es declarativa: se separa

el conocimiento del dominio de los mecanismos de deducción. Esto permite reutilizar tanto la Base del Conocimiento como los mecanismos de razonamiento [Alonso 2004]. En este trabajo de tesis, el paradigma de desarrollo basado en conocimiento al que se hace referencia, consta de dos partes principales: la Base de conocimiento y el motor de inferencia o proceso de razonamiento, donde el objetivo es obtener por inferencia la base de datos y los programas para manejar los objetos descritos por el usuario. Pero, no busca emular capacidades de un dominio experto. Este paradigma está orientado más a los sistemas de gestión. Permiten partir de la descripción de las visiones de todos los usuarios del sistema a desarrollar y genera la base de datos y los programas necesarios para satisfacer dichas visiones.

En el trabajo de Holger Knublauch [Knublauch 2002], se planteó un proceso ágil, que busca aplicar las prácticas principales de XP para el modelado de conocimiento. El objetivo principal fue buscar realizar el desarrollo de sistemas basados en conocimiento de manera más eficiente. Pero si bien en ese trabajo se plantea combinar metodologías ágiles y desarrollo basado en conocimiento, el concepto de desarrollo basado en conocimiento no es el mismo, en el caso del trabajo de Knublauch está relacionado con la inteligencia artificial y la posibilidad de emular capacidades de un dominio experto, mientras que en el trabajo planteado en esta tesis, el concepto de desarrollo basado en conocimiento es diferente, por lo especificado en el párrafo anterior.

Ya en el año 1983 Blazer y sus colegas [Barler] habían propuesto un modelo operacional que soporte la implementación automática a través de un CASE (asistente de desarrollo de software computarizado). La especificación de software operacional es obviamente una parte integral de la base de conocimiento de ingeniería de software. Crear este tipo de base de conocimiento, según los mencionados autores, requiere que:

- Exista un modelo de representación de base del conocimiento y tenga asociado un mecanismo de almacenamiento.
- La representación de la base del conocimiento esté embebida en un entorno de sistema de conocimiento que pueda ser accedido, modificado y ejecutado.
- La base del conocimiento está configurada con la versión inicial de un modelo operacional del ciclo de vida.

Un sistema basado en conocimiento útil debe ser capaz de almacenar conocimiento de una manera persistente, una que pueda ser administrada y mantenida como se mantiene una base de datos. Una base de conocimiento persistente puede construirse acoplando sistemas de conocimiento y un administrador de base de datos relacional. [IEEE web]

Entonces, es bueno restringir el concepto de conocimiento que se utilizará al mencionar el término Desarrollo Basado en Conocimiento adhiriendo a lo especificado por Artech [Artech 2008]. Se trata de conocimiento que cumple las siguientes condiciones:

- Riguroso
- Representable en forma objetiva
- Operable

Una nueva manera de resolver el problema del desarrollo de sistemas pasa por la sustitución de la premisa básica enunciada anteriormente: asumir que no es posible construir un modelo de datos estable de la empresa y, en cambio, utilizar una filosofía incremental y hacer un Desarrollo Basado en Conocimiento. Un esquema incremental parece muy natural: no se encaran grandes problemas, sino que se van resolviendo los pequeños problemas a medida que se presentan. [Artech 2008].

Este paradigma de desarrollo basado en conocimiento es completamente diferente a los usuales paradigmas de desarrollo de sistemas: no parte de un modelo de datos pre-existente ni de concepciones abstractas sobre lo que es importante para la empresa y lo que no lo es. [Gonda 2010]

En todas las organizaciones hay múltiples usuarios (desde el Gerente General al cargo más bajo en el escalafón de la empresa). Pero, difícilmente exista alguien entre todos ellos que tenga el conocimiento suficiente sobre los datos de la organización. Tampoco es muy probable que alguno de ellos conozca estos datos con la adecuada objetividad y el suficiente detalle. Y este no es un problema que afecte exclusivamente a las grandes empresas, ocurre en empresas de todo tamaño. Por ello *se debe privilegiar lo concreto sobre lo abstracto*, porque los usuarios, que serán múltiples y tendrán los perfiles y roles más diversos en la empresa, se manejan bien con elementos concretos de su ambiente y mal con conceptos abstractos: la representación del conocimiento debe ser simple, detallada y objetiva.

La idea de este paradigma es partir de las diferentes visiones de sus usuarios. Cada usuario, perteneciente a cualquier nivel de la empresa, conoce bien la visión de los datos con los que trabaja a diario. Así, tomando como base este conjunto de visiones de datos, se encuentra el modelo de datos ideal derivado de ellas (puede probarse rigurosamente que, dado un número de visiones de usuarios, existe solo un modelo relacional mínimo que las satisface [Gonda 2007]).

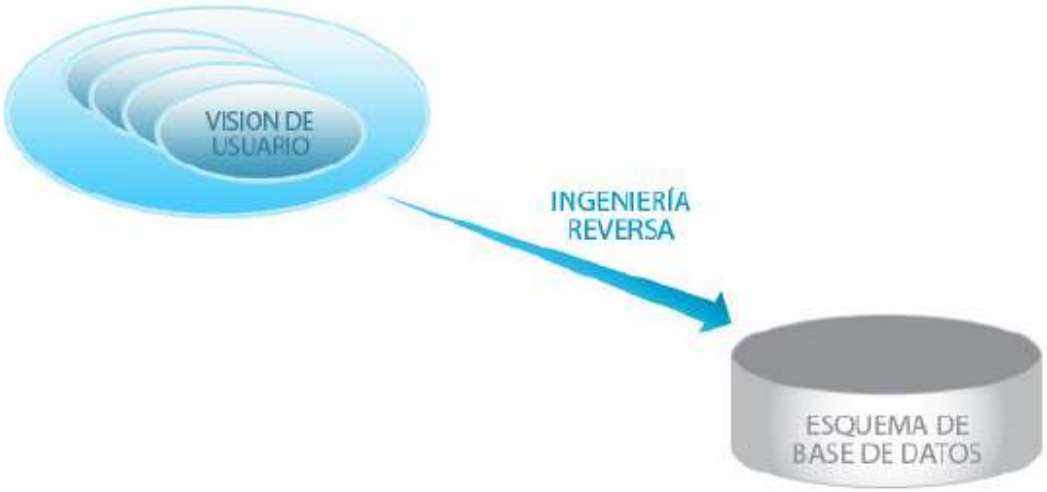


Figura 3.1.2: Proceso de Ingeniería Inversa [Gonda 2007]

Se puede pensar entonces en un proceso de ingeniería inversa (ver Figura 3.1.2) que, a partir de una serie de visiones de datos de diferentes usuarios, desarrolla el modelo ideal y la base de datos relacional correspondiente. Todo este conocimiento se puede sistematizar y todo este conocimiento es lo que Gonda denomina una Base de Conocimiento. Además, como subproducto, también se puede sistematizar una buena descripción de las visiones de los usuarios y, partiendo de esto, se pueden generar, por ejemplo, los programas requeridos para operar con ellas.

Por lo tanto, resumiendo, este nuevo paradigma pretende capturar el conocimiento que existe en las visiones de los usuarios, y sistematizarlo en una base de conocimiento (todo ello en forma automática). La característica fundamental de esta base de conocimiento, que la diferencia de los tradicionales diccionarios de datos, es su capacidad de inferencia: se pretende que, en cualquier momento, se puedan obtener de esta base de conocimiento, tanto elementos que se han colocado en ella, como cualquier otro que se pueda inferir a partir de ellos. Si este objetivo se logra, la base de datos y los programas de aplicación pasan a ser transformaciones determinísticas de dicha base de conocimiento y ello permite [Artech 2008]:

- Generarlos automáticamente
- Ante cambios en las visiones de los usuarios determinar el impacto de dichos cambios sobre datos y procesos y propagar esos cambios generando:

- los programas necesarios para convertir los datos;
- los programas de la aplicación afectados por los cambios;
- aquellos programas de aplicación que no han sido afectados por los cambios pero que, ahora, podrían ser sustituidos por otros más eficientes

3.1.1.4 Modelo Externo y Base de Conocimiento [Gonda 2007]

Históricamente la comunidad informática comenzó trabajando sólo con un modelo físico, que tenía muchas rigideces. Luego fue introduciendo otros modelos a los efectos de tener más flexibilidad y obtener cierta permanencia de las descripciones. Por ejemplo ANSI SPARC [ANSI-SPARC web] introduce tres modelos:

- **Modelo Externo** donde se representan las visiones externas.
- **Modelo Lógico o Conceptual**: Es otro modelo (generalmente un modelo E-R) que se pretendía obtener por abstracción de la realidad.
- **Modelo Físico**: Se refería fundamentalmente al esquema de la base de datos.

La idea era desarrollar paralelamente los tres modelos para que los programas sólo interactuaran con el Modelo Externo y, a partir del Modelo Lógico se hiciera un mapeo entre esos programas y la base de datos (Modelo Físico) y que ello hiciera independientes a los programas de la base de datos.

El informe fue muy elogiado en su momento y, luego, casi olvidado. Más allá de todo esto, el informe ANSI SPARC sigue siendo válido hoy. La mayor diferencia es la tecnología disponible. En los años transcurridos, los cambios tecnológicos han sido muy importantes.

Analizando el asunto a la luz de la tecnología actual, los desarrolladores que trabajan con el nuevo paradigma de desarrollo basado en conocimiento, concluyen que pueden existir varios modelos, pero *el modelo realmente importante para los usuarios y los desarrolladores, es el Modelo Externo:* en él se recoge el conocimiento exterior y todo lo demás como, otros modelos auxiliares que pudieran ayudar, debe y puede inferirse automáticamente a partir de dicho Modelo Externo.

Ellos pone el énfasis en el *Modelo Externo*, porque es allí donde está el conocimiento genuino y donde reposa la esencia (el "*qué*") y es el realmente importante para los usuarios y los desarrolladores. En él se recoge el conocimiento exterior y todo lo demás, como otros modelos auxiliares que pudieran ayudar, puede inferirse automáticamente a partir de ese Modelo Externo. Un "Modelo Externo" no puede contener ningún elemento físico o interno como: archivos, tablas, entidades, relaciones entre entidades, índices o cualquier otro que pueda deducirse automáticamente del mismo. Ante nuevas posibilidades de la tecnología y necesidades de los clientes, lo primero a hacer es ampliar dicho Modelo Externo.

El Modelo Relacional está orientado a obtener una buena representación de los datos en una Base de Datos, obedeciendo algunas condicionantes muy deseables: eliminar la redundancia e introducir un pequeño conjunto de reglas, de manera de evitar las mayores fuentes de inconsistencia de los datos e introducir un conjunto de operadores que permitan manipular los datos a buen nivel.

El Modelo Externo se utiliza, entonces, para obtener y almacenar el conocimiento. Este modelo pretende ser la representación más directa y objetiva posible de la realidad, por ello, se toman las visiones de los diferentes usuarios. Estas visiones son almacenadas en el modelo. Luego, se captura todo el conocimiento contenido en ellas y se lo sistematiza para maximizar las capacidades de inferencia. Con eso se logra la independencia de la implementación, porque se tiene una descripción del sistema en alto nivel, descripción que solo cambiará si cambian las visiones sobre el mismo. El Modelo Relacional se utiliza para representar y manipular los datos: es el modelo interno o físico, pero será inferido por alguna herramienta que trabaje con este paradigma a través de un procedimiento de ingeniería inversa. [MárquezLisboa 2007]

Para definir el Modelo Externo, de manera que pueda aplicarse ingeniería inversa sobre ellas se realiza una descripción formal y rigurosa para que no haya ambigüedades y que un programa pueda "inferir" todo eso que los programadores hacen todavía a mano.

La Base de Conocimiento (Knowledge Base) inicialmente tiene asociado un conjunto mecanismos de inferencia y contiene reglas generales [MárquezLisboa 2007], como por ejemplo las que aseguran la consistencia, que son independientes de cualquier aplicación particular. Luego, el analista comienza a describir la realidad del usuario a través de objetos, estas descripciones (el Modelo Externo) son sistematizadas automáticamente y pasan a estar en la base de conocimiento. Además, sobre ese conocimiento, obtiene un conjunto de resultados que le ayudan a mejorar la eficiencia de las inferencias posteriores. Adicionalmente, sobre el conocimiento anterior, el sistema infiere lógicamente un conjunto de resultados que ayudan a mejorar la eficiencia de las inferencias posteriores. Esto se refleja en la siguiente Figura.

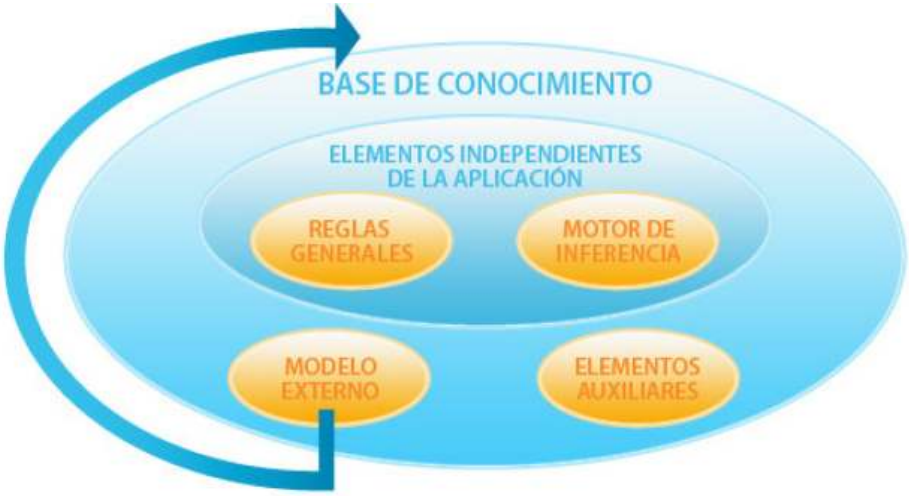


Figura 3.1.3: Base de conocimiento [Gonda 2007]

Todo el conocimiento de la Base de Conocimiento es equivalente al contenido en el Modelo Externo (subconjunto de ella), ya que consiste en el propio Modelo Externo más reglas y mecanismos de inferencia independientes de dicho Modelo Externo y un conjunto de otros elementos que son automáticamente inferidos a partir del mismo.

El desarrollador puede alterar, modificando objetos de la realidad del usuario, el Modelo Externo y las modificaciones se propagarán automáticamente a todos los elementos que lo necesiten: otros elementos de la Base de Conocimiento, Base de Datos y programas de la aplicación. De la misma manera, el desarrollador no puede alterar directamente ningún elemento que no pertenezca al Modelo Externo.

A través de Herramientas Case Específicas creadas para tal fin se pretende dedicar la atención únicamente al Modelo Externo (el "qué") y abstenerse de tratar la Base de Conocimiento, que lo contiene y lo mantiene, (y que forma parte del "cómo": un sofisticado conjunto de herramientas matemáticas y lógicas sobre las que se basan los procesos de inferencia y resultados intermedios utilizados para aumentar la eficiencia de dichas inferencias). Si bien el "qué" y el "cómo" no funcionan uno sin el otro, si el "qué" no es correcto, todo está equivocado. De lo anterior se puede inferir algo muy importante: *todo* el conocimiento está contenido en el Modelo Externo y, por ello, mañana se podría soportar la Base de Conocimiento de una manera totalmente diferente y el conocimiento de los clientes seguiría siendo utilizable sin problema alguno.

Se intenta, entonces, hacer un modelo que tenga las siguientes características:

- **Conocimiento adecuado para su tratamiento automático**. Modelo con el cual un software de computador pueda trabajar automáticamente. Esto implica que el conocimiento del modelo debe ser "entendible" y "operable" automáticamente.
- **Consistencia**. Modelo siempre consistente
- **Ortogonalidad**. Los objetos que en un determinado momento constituyen el modelo deben ser independientes entre sí. La adición de un nuevo objeto o su modificación o eliminación no implicarán la necesidad de modificar ningún otro objeto.
- **Proyecto, generación y mantenimiento automáticos de la base de datos y los programas**. Entre las cosas que un software de computador puede hacer automáticamente con el modelo, son esenciales el proyecto, generación y mantenimiento automáticos de la base de datos y los programas.
- **Escalabilidad**. Los ítems anteriores expresan las características cualitativas que debe tener el modelo. Pero se pretende resolver problemas grandes. Para ello se necesita que los mecanismos de almacenamiento, acceso e inferencia del modelo actúen eficientemente con independencia del tamaño del problema.

En resumen, la Base de Conocimiento y los mecanismos de inferencia asociados constituyen una gran "máquina de inferencia". Un buen ejemplo de esto es: dada una visión de datos se puede inferir en forma totalmente automática el programa necesario para manipularla.

3.1.2 Genexus

> *"El objetivo de Genexus es [a través de la descripción de las visiones de los usuarios] conseguir un muy buen tratamiento automático del conocimiento de los sistemas de negocios." Breogán Gonda & Nicolás Jodal*

Genexus es una herramienta de desarrollo de software basada en conocimiento (Knowledge-based Development Tool), orientada principalmente a aplicaciones de clase empresarial para la web, plataformas Windows y Smart Devices producida en Uruguay por la empresa Artech. Su primera versión fue liberada al mercado para su comercialización en 1989. Genexus X Evolution 3, la versión actual, fue lanzada en el 2014. Sus creadores, Breogán Gonda y Nicolás Jodal, han sido galardonados con el Premio Nacional de Ingeniería en 1995 otorgado por el "Proyecto Genexus" [Gonda 1995] [Artech 2008].

Según los propios autores Genexus es, esencialmente, un sistema que permite una buena administración automática del conocimiento de los sistemas de negocios [Gonda 2010]. Genexus es una herramienta que parte de las visiones de los usuarios; captura su conocimiento y lo sistematiza en una *base de conocimiento*. A partir de esta última, Genexus es capaz de diseñar, generar y mantener de manera totalmente automática la estructura de la base de datos y los programas de la aplicación, es decir, los programas necesarios para que los usuarios puedan operar con sus visiones. [Genexus web] El desarrollador describe sus aplicaciones en alto nivel (de manera mayormente declarativa), a partir de lo cual se genera código para múltiples plataformas. Genexus incluye un módulo de normalización, que crea y mantiene la base de datos óptima (estructura y contenido) basada en las visiones de la realidad descriptas por los usuarios utilizando un lenguaje declarativo, ver Figura 3.1.4 y Figura 3.1.5. [Artech 2008]

Figura 3.1.4: Base de Conocimiento [Gonda 2007]

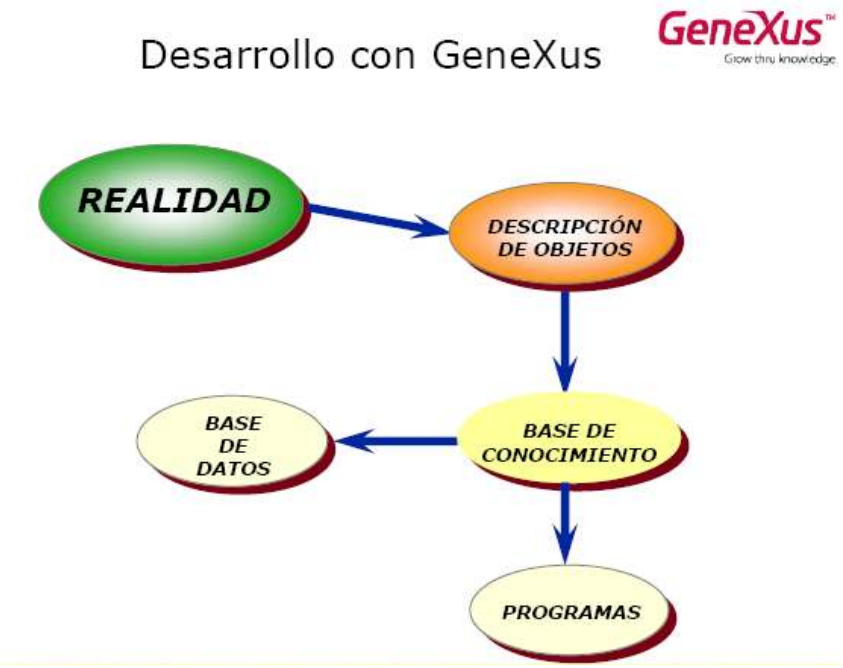


Figura 3.1.5: Desarrollo con Genexus[Gonda 2007]

GeneXus [Genexus web] es una herramienta de especificación de sistemas de información basada en la aplicación de un modelo con rigurosa fundación matemática que permite capturar e integrar las visiones de los usuarios en bases de conocimiento a partir de las cuales genera, mediante ingeniería inversa y procesos de inferencia, bases de datos normalizadas y aplicaciones completas; para diferentes plataformas de destino a partir de una misma especificación básica, pudiendo mantenerlas en forma automatizada ante cambios en los requerimientos [MárquezLisboa 2007] [ANSI-SPARC web] [Latorres 2003] [Salvetto 2006]. GeneXus modela la realidad mediante un conjunto de instancias de "objetos tipos Genexus", almacenados en las bases de conocimiento (Ver Figura 3.1.6). Produce el código necesario para implementar las aplicaciones en diferentes lenguajes y plataformas de destino mediante la utilización de programas generadores. Se apoya en

una metodología incremental e iterativa y trabaja sobre especificaciones, lo cual permite independencia tecnológica.

Figura 3.1.6: Objetos Genexus

La base de conocimiento de Genexus tiene una gran capacidad de inferencia lógica: en cualquier momento es capaz de proporcionar cualquier conocimiento que se ha almacenado en ella o que puede inferirse lógicamente de aquellos almacenados en ella. Basado en esta capacidad de inferencia es capaz de proyectar, generar y mantener, en forma 100% automática, la base de datos y los programas necesarios para satisfacer todas las visiones de usuarios conocidas en un determinado momento.

La versión X posee una arquitectura extensible, por lo cual pueden agregarse al producto paquetes de *software*, conocidos como extensiones, capaces de interactuar con la base de conocimiento, pudiendo ser desarrolladas por terceros [Gonda 2010].

Una parte considerable del total de *software* producido en Uruguay se elabora con esta herramienta creada y mantenida por una empresa uruguaya, Artech y en Salta existen varias empresas del ámbito público y privado que también las utilizan.

3.1.2.1 Orígenes de Genexus [Gonda 2007]

El primer propósito de Genexus era establecer un robusto modelo de datos. Una vez construido con todo rigor dicho modelo, se pudo pensar en una ampliación del mismo incorporándole elementos declarativos como reglas, fórmulas, pantallas, etc., tratando de describir las visiones de datos de los usuarios y, de esta manera, abarcar todo el conocimiento de los sistemas de negocios. Genexus trató de abarcar todo el conocimiento de los sistemas de negocios porque es una condición necesaria para poder proyectar, generar y mantener en forma 100% automática los sistemas computacionales de una empresa cualquiera. Pero el propósito de Genexus actualmente *es conseguir un muy buen tratamiento automático del conocimiento de los sistemas de negocios.* Ésta es la esencia. Cumplido ese objetivo existe un gran conjunto de subproductos como, por ejemplo:

- Proyecto, generación y mantenimiento automáticos de la base de datos y los programas de aplicación necesarios para la empresa.
- Generación, a partir del mismo conocimiento, para múltiples plataformas.
- Integración del conocimiento de diversas fuentes para atender necesidades muy complejas con costos en tiempo y dinero muy inferiores a los habituales.
- Generación para las tecnologías futuras, cuando esas tecnologías estén disponibles, a partir del conocimiento que, hoy, atesoran los clientes Genexus en sus Bases del

Conocimiento: Genexus trabaja con el conocimiento puro, cuya validez es totalmente independiente de las tecnologías de moda.

3.1.2.2 Tratamiento Automático del Conocimiento por Genexus [Gonda 2010]

"Genexus, al apoyarse en el paradigma de desarrollo basado en conocimiento, trabaja con conocimiento puro, cuya validez es totalmente independiente de las tecnologías de moda." [MárquezLisboa 2007]

Genexus es una herramienta que parte de las "visiones de los usuarios"; captura su conocimiento y lo sistematiza en una base de conocimiento. Genexus almacena en una Base de Conocimiento todos los elementos necesarios para construir la aplicación, y luego la utiliza para el desarrollo del sistema, construyendo automáticamente el modelo de datos en forma normalizada, y también utilizando el lenguaje de programación y base de datos que se le indique. A partir de su base de conocimiento, Genexus es capaz de diseñar, generar y mantener de manera totalmente automática la estructura de la base de datos y los programas de la aplicación (los programas necesarios para que los usuarios puedan operar con sus visiones). Así, permite obtener un proyecto con el conocimiento de la empresa. Todo ese conocimiento -reglas de negocio, diálogos, controles, reportes- que hoy están en un lenguaje de programación determinado, pueden ser convertidos a otros lenguajes sin necesidad de arrancar de cero. O sea que se reutiliza el conocimiento porque la Base de Conocimiento es conocimiento independiente de la plataforma.

Algunas características de esta herramienta CASE

- **Conocimiento puro**. Genexus trabaja con conocimiento puro, y este conocimiento es independiente de la tecnología utilizada.

- **Mantenimiento 100% automático**. Genexus *"conoce realmente"* la base de datos y los programas (porque posee el conocimiento para generarlos). Como consecuencia, es capaz de inferir un informe sobre el impacto causado por los cambios efectuados a los programas y a la base de datos, automáticamente y en cualquier momento. Y una vez que dicho reporte es aceptado, puede propagar automáticamente todos esos cambios a los datos y a los programas.

- **Independencia de plataforma, arquitectura y tecnología**. El conocimiento puro tiene un valor permanente, y es independiente de elementos de menor nivel tales como la plataforma (hardware, sistema operativo, servidor de base de datos, servidor de aplicaciones, etc.), la arquitectura (centralizada, cliente servidor de dos capas, cliente servidor de tres capas, multiservidor orientado a la red como Java o Microsoft .NET) y la tecnología disponible. Como consecuencia, el conocimiento que ha sido compilado en el desarrollo de un sistema con una plataforma y una arquitectura específicas y en un contexto tecnológico específico, puede usarse para generar sistemas para otras plataformas, arquitecturas y contextos tecnológicos (por ejemplo, las aplicaciones que hayan sido desarrolladas hace diez años para una plataforma centralizada y pantallas de formato texto, pueden ser tomadas ahora para plataforma Microsoft .NET o Java). Debido a esto es que se dice que Genexus protege el conocimiento de todos los usuarios, independientemente de la tecnología utilizada. Cualesquiera que sean las tecnologías usadas en el futuro, el conocimiento será el mismo; por lo tanto, construyendo los generadores necesarios, este conocimiento será reutilizado para generar sistemas para las nuevas tecnologías.

- **El Negocio del Conocimiento**. Otra consecuencia del tratamiento automático del conocimiento mencionado anteriormente es que este conocimiento puede ser "fácilmente integrado" y, por lo tanto, comprado y vendido para facilitar y optimizar el desarrollo de sistemas.

3.1.2.3 Ciclo de desarrollo incremental Genexus [Artech 2008]

Cuando una aplicación se desarrolla con Genexus la primera etapa consiste en hacer el Diseño de la misma registrando las visiones de usuarios (a partir de las cuales el sistema captura y sistematiza el conocimiento). Posteriormente se pasa a la etapa de Prototipación en donde Genexus genera la base de datos (estructura y datos) y programas para el ambiente de prototipo. Una vez generado el Prototipo, debe ser puesto a prueba por el analista y los usuarios. Si durante la prueba del Prototipo se detectan mejoras o errores se retorna a la fase de Diseño, se realizan las modificaciones correspondientes y se vuelve al Prototipo. Se denomina a este ciclo Diseño/Prototipo, ver Figura 3.1.7. Una vez que el Prototipo está aprobado, se pasa a la etapa de Implementación, en donde Genexus genera, también automáticamente, la base de datos y programas para el ambiente de producción.

En resumen, una aplicación comienza con un diseño, luego se realiza el prototipo, posteriormente se implementa o pone en producción y en cualquiera de los pasos anteriores se puede regresar al diseño para realizar modificaciones.

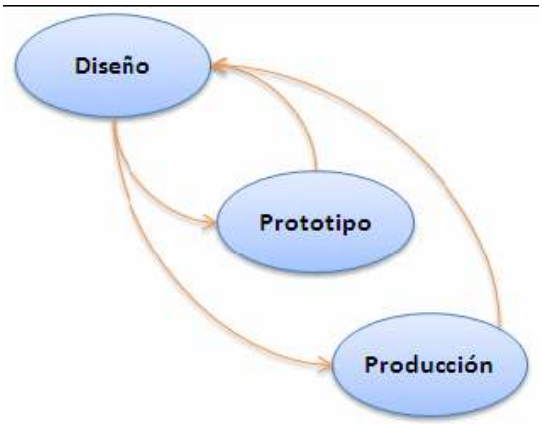


Figura 3.1.7: Ciclos Diseño-Prototipación y Diseño-Producción [Artech 2008]

A continuación se describe cada una de estas tareas.

- **Diseño**

Esta tarea es realizada conjuntamente por el analista y el usuario, y consiste en identificar y describir las visiones de datos de los usuarios. El trabajo se realiza en el ambiente del usuario. Este esquema permite trabajar con un bajo nivel de abstracción, utilizando términos y conceptos que son bien conocidos por el usuario final.

Una consecuencia muy importante, es que la actitud del usuario se transforma en francamente participativa. El sistema pasa a ser una obra conjunta y, como el usuario sigue permanentemente su evolución, su calidad es mucho mejor que la habitual.

De acuerdo a lo visto, Genexus captura el conocimiento por medio de visiones de objetos de la realidad del usuario. Los tipos de objetos soportados por Genexus son, entre otros: Transacciones, Reportes, Procedimientos, Work Panels, Web Panels, Data Views, Transacciones de BI.

La tarea de diseño consiste, fundamentalmente, en identificar y describir estos objetos. A partir de estas descripciones, y automáticamente, Genexus sistematiza el conocimiento capturado y va construyendo, en forma incremental, la Base de Conocimiento. Esta Base de Conocimiento es un repositorio único de toda la información del diseño, a partir de la cual Genexus crea el modelo de datos físico (tablas, atributos, índices, redundancias, reglas de integridad referencial, etc.), y los programas de aplicación. Así, la tarea fundamental en el análisis y diseño de la aplicación se centra en la descripción de los objetos Genexus.

Desarrollo basado en el conocimiento

Partiendo de los objetos descritos, el modelo de datos físico es diseñado con base en la Teoría de Bases de Datos Relacionales, y asegura una base de datos en tercera forma normal (sin redundancia). Esta normalización es efectuada automáticamente por Genexus. El analista puede, sin embargo, definir redundancias que, a partir de ello, pasan a ser administradas (controladas o propagadas, según corresponda), automáticamente por Genexus.

El repositorio de Genexus mantiene las especificaciones de diseño en forma abstracta, o sea que no depende del ambiente objeto, lo que permite que, a partir del mismo repositorio, se puedan generar aplicaciones funcionalmente equivalentes, para ser ejecutadas en diferentes plataformas.

- **Prototipado**

En las tareas de diseño están implícitas las dificultades de toda comunicación humana:

- El usuario olvida ciertos detalles.
- El analista no toma nota de algunos elementos.
- El usuario se equivoca en algunas apreciaciones.
- El analista interpreta mal algunas explicaciones del usuario.

Pero, además, la implementación de sistemas es, habitualmente, una tarea que insume bastante tiempo, por lo que:

- Como muchos de estos problemas sólo son detectados en las pruebas finales del sistema, el costo (tiempo y dinero) de solucionarlos es muy grande.
- La realidad cambia, por ello, no es razonable pensar que se pueden congelar las especificaciones mientras se implementa el sistema.
- La consecuencia de la congelación de las especificaciones, es que se acaba implementando una solución relativamente insatisfactoria.

El impacto de estos problemas disminuiría mucho si se consiguiera probar cada especificación, inmediatamente, y saber cuál es la repercusión de cada cambio sobre el resto del sistema.

Una primera aproximación a esto, ofrecida por diversos sistemas, es la posibilidad de mostrar al usuario formatos de pantallas, informes, etc. animados por menús. Esto permite ayudar al usuario a tener una idea de qué sistema se le construirá pero, posteriormente, siempre se presentan sorpresas.

Una situación bastante diferente sería la de poner a disposición del usuario para su ejecución, inmediatamente, una aplicación funcionalmente equivalente a la deseada, hasta en los mínimos detalles. Esto es lo que hace Genexus: *Un prototipo Genexus es una aplicación completa, funcionalmente equivalente a la aplicación de producción*.

La diferencia entre prototipación y producción consiste en que la primera se hace en un ambiente de microcomputador, mientras que la producción se realiza en el ambiente objeto del usuario (IBM iSeries, servidor Linux, Cliente / Servidor, JAVA, .NET, etc.). El prototipo permite que la aplicación sea totalmente probada antes de pasar a producción. Durante estas pruebas, el usuario final puede trabajar con datos reales, o sea que prueba, de una forma natural, no solamente formatos de pantallas, informes, etc. sino también fórmulas, reglas del negocio, estructuras de datos, etc.

La filosofía de GeneXus está basada en el concepto conocido como *desarrollo incremental*. Cuando se trabaja en un ambiente tradicional, los cambios en el proyecto hechos durante la implementación y, sobre todo, aquellos que son necesarios luego de que el sistema está implantado, son muy onerosos (y raramente quedan bien documentados).

GeneXus resuelve este problema: construye la aplicación con una metodología de aproximaciones sucesivas que permite, una vez detectada la necesidad de cambios, prototiparlos y probarlos inmediatamente por parte del usuario, sin costo adicional.

- **Implementación**

Genexus genera automáticamente el código necesario para:

  - Crear y mantener la base de datos;
  - Generar y mantener los programas para manejar los objetos descritos por el usuario.

El proceso de generación puede ser considerado en dos etapas: *especificación y generación*. La especificación es totalmente independiente del ambiente objeto, pero la generación no. Esto significa que se puede ejecutar el mismo modelo en las diferentes plataformas de ejecución para las que se ha generado y cada una de estas versiones generadas puede ser optimizada de acuerdo con el ambiente en el cual correrá.

Recientemente en abril de 2014 se lanzó Genexus X Evolution 3 que acompaña los nuevos avances en la tecnología, incorporando un generador web basado en HTML5 y CSS3, Integración Automática en la Nube, y el generador de aplicaciones nativas para dispositivos móviles, con soporte para las plataformas más populares del mercado: Android, Blackberry e iOS.

Genexus genera código: para múltiples lenguajes, incluyendo: Cobol, RPG, Visual Basic, Visual FoxPro, Ruby, C#, Java; para múltiples plataformas nativas para dispositivos móviles, web compatibles con todos los browsers, y para servidores IBM, Apache y Windows. Los DBMSs más populares son soportados, como Microsoft SQL Server, Oracle, IBM DB2, Informix, PostgreSQL y MySQL

Los ambientes y lenguajes más importantes actualmente soportados hasta la fecha de edición de este documento se detallan en la siguiente tabla:

Plataformas	• *Plataformas de ejecución: JAVA, Microsoft .NET, Microsoft .NET .NET Mobile, Ruby* • *Sistemas Operativos: IBM OS/400, LINUX, UNIX, Windows NT/2000/2003 Servers, Windows NT/2000/XP/Vista, Windows 7, y Windows Mobile* • *Internet: JAVA, ASP.NET, Visual Basic (ASP), C/SQL, HTML, Web Services*
Bases de Datos	• *IBM DB2 , DB2 para iSeries - System i, Informix, Microsoft SQL Server, MySQL, Oracle, PostgreSQL, SQL Server CE, SQL Lite*
Lenguajes	• *Java para Android, Java para BlackBerry, Objective-C para iOS, WinJS, HTML5 & CSS 3 para Windows 8; Java, .Net, Ruby y Javascript.* • *También leguajes legados: COBOL, RPG, Visual FoxPro*
Servidores Web	• *Microsoft IIS, Apache, WebSphere, etc*
Múltiples Arquitecturas	• *Arquitecturas de múltiples capas, basadas en web, Cliente/Servidor, centralizadas (iSeries).*
Mobile	• *Android, iPhone, iPad, iPod Touch, BlackBerry y Windows 8*

Tabla 3.1.1: Ambientes y lenguajes soportados por Genexus [Genexus web]

Una característica recientemente agregada a la nueva versión de Genexus, y que va la pena destacar, es la programación modular. Permite dividir las Bases de Conocimiento Genexus en

módulos que vuelven más sencillo desarrollar y mantener aún las soluciones IT más complejas. Al partir los sistemas monolíticos en componentes manejables cada miembro del equipo puede trabajar de manera simultánea en un módulo de aplicación lógicamente distinto.

También en la nueva versión, se ha mejorado el lenguaje del corazón de Genexus que brinda mayor flexibilidad a los desarrolladores.

Además Genexus ofrece un conjunto de herramientas complementarias, suite Genexus, que facilitan el desarrollo y se detallan en la siguiente tabla:

Herramienta	Funcionalidad principal
GXflow	(http://genexus.es/gxflow/): modelar, optimizar, y administrar procesos de negocio
GXquery	crear reportes a partir de bases de datos operacionales. (http://genexus.es/gxquery/)
GXportal	construir sitios web dinámicos y sencillos. (http://www.genexus.com/productos/cree-sitios-a-medida?es)
GxServer	coordinar el trabajo de desarrollo de aplicaciones Genexus. (http://genexus.es/gxserver/)
GXbpm	para modelar gráficamente procesos de negocio y optimizarlos. (http://www.genexus.com/productos/business-process-management-suite?es)
GxTest	automatizar pruebas funcionales. (http://genexus.es/gxtest/)
GXplorer	construir, administrar y explorar Data warehouses basado en datos de negocio y registrado por Genexus. (www.gxplorer.com)

Tabla 3.1.2: Suite de herramientas Genexus [Genexus web]

- **Mantenimiento**

Esta es una de las características más importante de Genexus, y la que lo diferencia de manera más clara de sus competidores: el mantenimiento, tanto de la base de datos (estructura y contenido) como de los programas, es totalmente automático.

A continuación se explicará el proceso de mantenimiento ante cambios en la descripción de algún objeto GeneXus (visión del usuario):

- Impacto de los cambios sobre la base de datos

  Análisis de impacto: Una vez descritos los cambios de las visiones de usuarios, Genexus analiza automáticamente cual es el impacto de los mismos sobre la base de datos y produce un informe donde explica cómo debe hacerse la conversión de los datos y, si cabe, qué problemas potenciales tiene esa conversión (inconsistencias por viejos datos ante nuevas reglas, etc.). El analista decide si acepta el impacto y sigue adelante o no.

  Generación de programas de conversión: Una vez que los problemas han sido solucionados o bien se ha aceptado la conversión que Genexus sugiere por defecto, se

generan automáticamente los programas para hacer la conversión (estructura y contenido) de la vieja base de datos a la nueva.

Ejecución de los programas de conversión: A continuación, se pasa al ambiente de ejecución que corresponda (prototipo, producción Internet, producción Cliente / Servidor, etc.) y se ejecutan los programas de conversión.

- Impacto de los cambios sobre los programas

  Análisis de impacto: Genexus analiza el impacto de los cambios sobre los programas, y produce un diagnóstico informando qué programas deben generarse o re-generarse y proporcionando también, para el nuevo programa, o bien el diagrama de navegación o bien un pseudo-código, a elección del analista.

  Generación de nuevos programas: A continuación el sistema genera o regenera automáticamente todos los programas.

- **Documentación**

Todo el conocimiento provisto por el analista, o inferido por GeneXus, está disponible en un repositorio activo, que constituye una muy completa documentación online, permanentemente actualizada.

La documentación incluye la descripción de objetos específicos e información sobre la base de conocimiento resultante y sobre la base de datos diseñada. La base de conocimiento de GeneXus no solamente le permite acceder al conocimiento que almacena siempre que el desarrollador lo desee sino que también le habilita el acceso a toda la información inferida lógicamente (una regla de integridad referencial, un mapa de navegación en la base de datos, un análisis de impacto de cambios, referencias cruzadas, diagramas de Entidad – Relación inferidos a partir del conocimiento almacenado, etc.).

3.1.2.4 Tendencias de Genexus

Genexus es un producto con una permanente evolución. Las tendencias de esta evolución se las pueden representar sobre cuatro dimensiones: *completitud, productividad, universalidad y usabilidad.*

- **Completitud**: Mide el porcentaje de código del sistema del usuario que es generado y mantenido automáticamente por Genexus. Desde 1992 la completitud alcanzada por Genexus es siempre de un 100%, Éste es un compromiso fundamental porque implica una propagación automática de los cambios, que implica una disminución dramática de los costos de desarrollo y mantenimiento.

- **Universalidad**: Mide la cobertura de las diferentes plataformas importantes disponibles en el mercado. Hoy soporta todas las plataformas "vivas", entendiéndose por plataformas vivas aquellas que tienen una importante y creciente base instalada. De esta manera, brinda a los clientes una gran libertad de elección: si una aplicación es desarrollada con Genexus, el cliente puede siempre pasarla a otra plataforma soportada sin costos importantes.

- **Productividad**: Mide el aumento de productividad del desarrollo con Genexus sobre el desarrollo con programación manual. El objetivo de aumento de productividad de Genexus fue, durante muchos años, de un 500%, lo que era suficiente. Pero los sistemas que necesitan los clientes son cada día más complejos porque existen nuevos dispositivos y nuevas necesidades, sobre todo de sistemas que cumplan con las necesidades, por complejas que sean, pero se mantengan simples para los usuarios que, en general, no serán entrenados. Adicionalmente, esos sistemas deben ser desarrollados cada vez en

menos tiempo: el “time to market” es cada vez más crítico. En el año 2004 se modificaron esos objetivos pasando al 1000% para la versión 9 y el 2000% para la versión Rocha.

- **Usabilidad**: Mide la facilidad de uso. Y pretende hacerlo con carácter general: facilitar el uso a los usuarios técnicos, pero también aumentar el alcance permitiendo que cada vez más usuarios no técnicos puedan utilizar Genexus con provecho. El solo hecho de utilizar Genexus implica un gran aumento de la usabilidad: un desarrollador Genexus puede desarrollar excelentes aplicaciones para una determinada plataforma sin necesitar de un conocimiento profundo de esa plataforma lo que significa un nivel mínimo importante de usabilidad.

3.1.2.5 ¿Por qué elegir este nuevo paradigma?

La razón principal aducida por Breogán Gonda y Nicolás Jodal para optar por este nuevo tipo de paradigma, desarrollo basado en conocimiento, es que en las grandes organizaciones no existe nadie que conozca los datos de la empresa con la adecuada objetividad y el suficiente detalle. Por lo tanto, el paradigma introducido por Genexus, que consiste en tomar el conocimiento partiendo de las visiones de los usuarios (de alguna manera, realizando un desarrollo isomorfo con la perspectiva), es mucho mejor que los paradigmas tradicionales, tal como se describió anteriormente.

Usualmente, los clientes de Genexus lo utilizan para desarrollar y mantener grandes aplicaciones de Misión Crítica. [Gonda 2010]

3.1.3 Proyecto Altagracia

El gobierno de Venezuela prepara un generador de software libre que remplazará a Genexus denominado Altagracia. Este será un programa cuyo desarrollo es dirigido por el Centro nacional de Tecnología de información del gobierno bolivariano de Venezuela

3.1.3.1 Motivación del proyecto

Altagracia nace tras denuncias de las comunidades de Software Libre, luego de que el Estado venezolano firmara un convenio bilateral con Uruguay para la adquisición de licencias de Genexus que, por el decreto N° 3.390, debía ser totalmente libre, cumpliendo las cuatro libertades del software GPL. Genexus, que tiene muchas propiedades, pero que no es software libre.

El funcionario Carlos Figueira, presidente del Centro Nacional de Tecnologías de la Información de Venezuela, argumenta que de la reunión con directivos de Artech surgió el proyecto de crear un generador de código alternativo a Genexus que lo han llamado Altagracia, que será compatible con Genexus, tendrá sus cualidades, el mismo nivel de desarrollo de suite de aplicaciones, la misma potencia, con cambios mínimos. Según Figueira, Altagracia será una herramienta totalmente libre. Quien quiera tenerla, la adapta y la utiliza; serán las cuatro libertades puestas en su máxima expresión. Además estima que ahora la empresa Artech puede tener aplicaciones para ambos mundos: el libre y el propietario.

> *"Genexus tendrá algunas cosas distintas, en particular su capacidad para producir código para software propietario, Altagracia no hará nada para sistemas propietarios: base de datos, ERP y demás aplicaciones; esto permite que ellos mantengan su línea de negocios clásica del mundo propietario. Además tienen un esquema en el que ellos también se benefician, porque entran en el mundo del software libre; lo importante es que participarán empresas venezolanas, que serán formadas y participarán en el proceso". Carlos Figueira presidente del Centro Nacional de Tecnologías de la Información de Venezuela [Alvarado 2010]*

Para el desarrollo de Altagracia, se tomó como base el generador Genexus. Altagracia, será compatible con Genexus, tendrá sus cualidades, el mismo nivel de desarrollo de suite de

aplicaciones, la misma potencia, con cambios mínimos, pues el interés tampoco es violentar a la empresa uruguaya".

3.1.3.2 Altagracia

Altagracia apunta a obtener, a partir del trabajo en conjunto entre profesionales de la República Oriental del Uruguay y la República Bolivariana de Venezuela, una plataforma para la generación de códigos, construido bajo estándares abiertos. El nuevo software deberá cumplir con las condiciones sobre software libre, inter-operable e independiente de la tecnología englobada en Genexus. Altagracia será un generador capaz de soportar el diseño e implementación de nuevas aplicaciones en software libre, así como la adaptación y el mantenimiento de las soluciones desarrolladas en estas. Con el generador de códigos libre, se podrán hacer todas las aplicaciones basadas en Genexus, como sistemas administrativos, gestión de documentación, logísticos, todas serán hechas en fuentes libres.

> *"Altagracia nos dará total autonomía sobre las aplicaciones y nos brindará una herramienta para el desarrollo de aplicaciones sofisticadas. El Estado se reserva el derecho de reservar la no liberación del código fuente creado en sectores críticos, si esto ocurre, se haría por razones estratégicas." Carlos Figueira presidente del Centro Nacional de Tecnologías de la Información de Venezuela [Alvarado 2010]*

A inicio del año 2008 a través de diversos medios el Estado venezolano informó sobre este proyecto y anunció que tendría listo en un año el generador de código venezolano, de nombre código "Altagracia": un programa de computadora capaz de crear otros programas de computadora, ello a partir de diagramas introducidos por los programadores. *"Esto facilitará enormemente el desarrollo de aplicaciones de software libre para el Estado"*. Así lo había informado Carlos Figueira, presidente del Centro Nacional de Tecnologías de Información (Cnti, ente adscrito al Ministerio de Telecomunicaciones e Informática) en entrevista a Últimas Noticias.

El propio Figueira explicó en julio de 2010 que el proyecto estaba a punto de empezar y que "tardó el arranque porque involucra dos países y sin dudas hay complicaciones burocráticas, todo el tema tomará calor en pocas semanas y de aquí a un año estará el proyecto culminado" [Alvarado 2010]. En entrevista realizada por Heberto Alvarado Vallejo, también, Carlos Figueira informó que en el 2011 se tendrá listo el generador de código libre Altagracia basado en Genexus. A más de tres años de iniciado el acuerdo con la empresa uruguaya Genexus, creadora del generador de código del mismo nombre, que permitiría la creación de un sistema con las mismas prestaciones, llamado Altagracia, pareciera que ahora sí hay fecha definitiva para adopción definitiva de Genexus. Tal como se puede ver en la página de administración de proyectos del Área Estratégica: CIENCIA, TECNOLOGÍA E INDUSTRIAS INTERMEDIAS de la Embajada de la República Bolivariana de Venezuela en la República Oriental del Uruguay [GenexusLibre web]. El proyecto cuyo nombre completo es GENEXUS LIBRE. Proyecto Altagracia CNTI – Langecor figura que se terminaría a mediados de 2011.

Las Instituciones involucradas son: el Ministerio del Poder Popular para la Ciencia, la Tecnología y las Industrias Intermedias (http://www.mcti.gob.ve/), el Ministerio de Energía y Minería de la República Oriental del Uruguay (http://www.miem.gub.uy/portal/hgxpp001?5) y la empresa uruguaya Langecor, S.A. (Genexus Internacional). [CNTI web]

3.2 CONCLUSIONES DESARROLLO BASADO EN CONOCIMIENTO

La idea de trabajar con bases de conocimiento para el desarrollo de software parece haber resultado positiva y ahorrar tiempo y esfuerzo de desarrollo concentrando el trabajo en aspectos claves como entender lo que el cliente necesita.

Al trabajar con este tipo de desarrollo como Genexus no se plantea una metodología ágil. Si bien se menciona en [Genexus web] como una de sus fortalezas que ésta permite el desarrollo ágil, no hay que entenderlo como un desarrollo siguiendo una metodología ágil, cumpliendo con el

manifiesto ágil. Hay que entender la expresión simplemente como la posibilidad de acelerar los ciclos de producción y permitir responder rápidamente a los cambios del negocio, no implicando con esto que busca cumplir con los valores y principios enunciados en el manifiesto ágil.

Genexus es un producto con una permanente evolución. Tal como se mencionó anteriormente recientemente salió al mercado una nueva versión de Genexus que incorpora tecnología para Smart Devices. Una gran desventaja de Genexus es su característica de ser propietario pero el proyecto Altagracia reivindica la idea detrás de Genexus, buscando brindar las mismas características a través de una herramienta totalmente libre.

4 PROPUESTA DE PRÁCTICAS ÁGILES PARA EL DESARROLLO BASADO EN CONOCIMIENTO

4.1 INTRODUCCIÓN

Como primer paso para poder elaborar una propuesta de prácticas ágiles para el desarrollo basado en conocimiento se buscó conocer el contexto de esta forma de desarrollo en Salta Capital para luego presentar una forma de trabajo que logre que las organizaciones al seguirla se acerquen a los postulados enunciados en el manifiesto ágil. Por lo tanto, este capítulo muestra un estudio de campo de la situación en zona, luego se plantea una propuesta concreta de prácticas ágiles, posteriormente se muestran datos de una experiencia de aplicación práctica de esta propuesta y se exponen las conclusiones arribadas.

4.2 ESTUDIO DE CAMPO EN SALTA SOBRE DBC

En esta sección se muestra la información relevada sobre la forma de trabajo de empresas del sector público y privado de Salta que realizan desarrollo basado en conocimiento (DBC). Se pretendió conocer principalmente la experiencia de la organización, la conformación de los grupos de trabajo, y forma de trabajo en los proyectos a los que se aplican, como así también detectar aplicación de prácticas ágiles al desarrollo. Para finalizar este estudio de campo, se presentan las conclusiones a las que se pudo arribar sobre cada uno de los temas mencionados, luego de analizar las respuestas obtenidas.

Para obtener el listado de empresas y organismos del medio que trabajan con Bases de Datos del Conocimiento, lo primero que se realizó fue una entrevista con el representante de Genexus en Salta Capital. Como resultado se obtuvo un listado de las 11 organizaciones que lo utilizan. Seis corresponden a empresas privadas y cinco, a organismos del sector público. Estos números corresponden a quienes trabajan con las licencias comerciales, sabiendo por supuesto que existe otro grupo de empresas que lo utilizan de manera ilegal. Varias empresas privadas, alrededor de 10 que antes utilizaban desarrollo basado en conocimiento, por la crisis que ha vivido últimamente el país, dejaron de trabajar con él.

A la hora de realizar el relevamiento se informó a las organizaciones, que los datos obtenidos serían tratados de manera anónima, por lo que no se hacen referencias directas a los nombres de las empresas u organismos para respetar la confidencialidad. Pero, se puede decir que todos tienen sede en Salta capital y una empresa es multinacional.

Tanto en organismos públicos como en empresas privadas, fueron las áreas de desarrollo de software las que fueron analizadas. Es por eso que de ahora en adelante se hará referencia a las áreas, grupos o sectores de desarrollo y no a organizaciones, salvo que sea necesario hacer una referencia específica.

4.2.1 Muestra Obtenida para estudio

Se realizó un primer contacto con cada uno de los sectores de desarrollo que trabajan con desarrollo basado en conocimiento en Salta Capital. Esto se realizó, principalmente, para saber si se contaba con su colaboración para este estudio de campo. En esta instancia se explicó brevemente los datos que se recabarían y se intentó definir una fecha para la entrevista con un responsable idóneo en el área en cada grupo de desarrollo, sin éxito en algunos casos. Todos se mostraron predispuestos a colaborar.

Para realizar el relevamiento, la idea inicial fue realizar entrevistas personales a cada responsable de área de desarrollo. Lamentablemente, después del primer contacto con las organizaciones, se pudieron concretar un 45.45% de entrevistas. En el resto de los casos, los responsables adujeron

no tener tiempo disponible para concretar una cita para la entrevista, por lo que se debió elaborar una encuesta. Esta encuesta fue enviada por correo electrónico para ser contestada. Dichas encuestas tenían una serie de preguntas cerradas y otras abiertas para poder obtener suficiente información. Para hacer las entrevistas y encuestas comparables, se tomó el modelo de las encuestas como base para realizar las entrevistas presenciales. Un modelo de las encuestas puede consultarse en el Anexo 1.

La estructura de la encuesta contaba con un grupo de preguntas que buscaban contextualizar la organización, pública o privada, tiempo en el mercado, tiempo de uso de Genexus, cantidad de personas en el área desarrollo, etc. La segunda parte de la encuesta, profundizaba sobre el trabajo con desarrollo basado en conocimiento utilizando Genexus, forma de trabajo, gestión de requerimientos, prácticas ágiles, herramientas, riesgos, entre otros aspectos

De las encuestas enviadas, se obtuvieron las respuestas del 66.66%. Las dos encuestas sin responder corresponden a organismos públicos, que adujeron no tener tiempo para responderla, si bien previamente se habían comprometido a realizarla. Pero en total, con encuestas y entrevistas se logró obtener información del 81.82% de quienes actualmente están trabajando con desarrollo basado en conocimiento en Salta Capital.

En las siguientes secciones se detallan los resultados obtenidos de los sectores de desarrollo de software de las empresas y organismos que colaboraron con información.

4.2.2 Presentación de datos obtenidos

4.2.2.1 Datos generales

El objetivo de las primeras preguntas fue identificar características de cada organización al que pertenecía el sector de desarrollo de software relevado: tipo de organismo/empresa, tiempo en el mercado, experiencia en desarrollo específicos con Bases de Datos del Conocimiento, tamaño del sector de desarrollo, distribución de proyectos y organización interna a la hora de afrontar proyectos de desarrollo.

4.2.2.2 Características de las empresas/organismos

- ***Tipo de organización:***

Se buscó poder contextualizar la organización donde se encuentra el sector de desarrollo que trabaja con Desarrollo Basado en conocimiento. Se permitía contestar: SI / NO. Del total de organizaciones relevadas que trabajan en Salta con Desarrollo Basado en conocimiento, existe una mayor proporción, poco más del 10%, de empresas del sector privado respecto del sector público (Ver figura 4.2.1).

Figura 4.2.1. Tipo de organizaciones que utilizan desarrollo basado en conocimiento

- ***Tiempo en el mercado:***

Para determinar la experiencia de la organización, se consultó el tiempo en el mercado. El 56% de las organizaciones relevadas tienen más de 10 años en el mercado. Muchos organismos públicos tienen inclusive más de 50 años de existencia y la variación se presenta en las empresas privadas. En promedio el tiempo en mercado de las empresas privadas es de poco más de 11 años (11,83). (Ver figura 4.2.1).

Figura 4.2.2. Tiempo en el mercado de las organizaciones

4.2.2.3 Experiencia con Desarrollo basado en Conocimiento

- ***Tiempo de uso de DBC:***

Un dato útil para medir la idoneidad del encuestado respecto al desarrollo basado en conocimiento es justamente saber cuánto tiempo viene trabajando con él. Se obtuvo que el tiempo medio de uso de esta forma de desarrollo es de 5 años y medio. Claramente se puede observar en el gráfico que el 55.56% de las empresas consultadas tiene entre 5 y 10 años de utilización de desarrollo basado en conocimiento y el 33.33% más de 10 años por lo que se puede concluir que el 88.98% de las organizaciones consultadas viene utilizándolo por más de 5 años. (Ver figura 4.2.3).De esta forma se considera que por lo menos en este porcentaje de organizaciones ya se tiene un conocimiento sólido de la forma de Desarrollo Basado en Conocimiento. En promedio el tiempo de uso es de poco más de 9 años y medio (9.56).

Figura 4.2.3. Tiempo de empleo de Desarrollo basado en conocimiento

- ***Cantidad de proyectos desarrollados con desarrollo basado en Conocimiento:***

Para continuar con el análisis de la experiencia con el desarrollo basado en conocimiento, se consultó la cantidad de proyectos desarrollados hasta el momento. Aquí se obtuvo una respuesta variada (Ver figura 4.2.4). El porcentaje de quienes que no han desarrollado más de 25 proyectos, es de un 55.56% de las organizaciones relevadas.

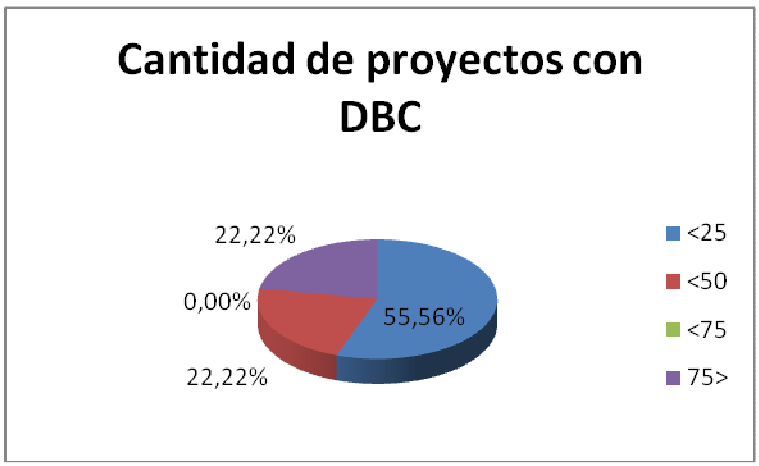


Figura 4.2.4. Cantidad de proyectos siguiendo Desarrollo basado en conocimiento

Por supuesto, solo cantidades no debería ser comparables de manera directa, dado que hay algunos proyectos de desarrollo de sistemas que realmente fueron muy importantes y de una larga duración, por lo que se pudo deducir de las entrevistas. Sin embargo, estos datos sirven para contextualizar la información.

- ***Proporción de Proyectos para terceros:***

Se consultó como dato adicional la proporción de proyectos realizados para terceros, respecto de los internos. Un 55.55% trabaja mayoritariamente para terceros y el 44.44% restante para proyectos internos de la organización(Ver figura 4.2.5). Aquellos que desarrollan mayoritariamente para terceros han desarrollado algunos proyectos internos para crear herramientas que faciliten y agilicen el proceso de desarrollo.

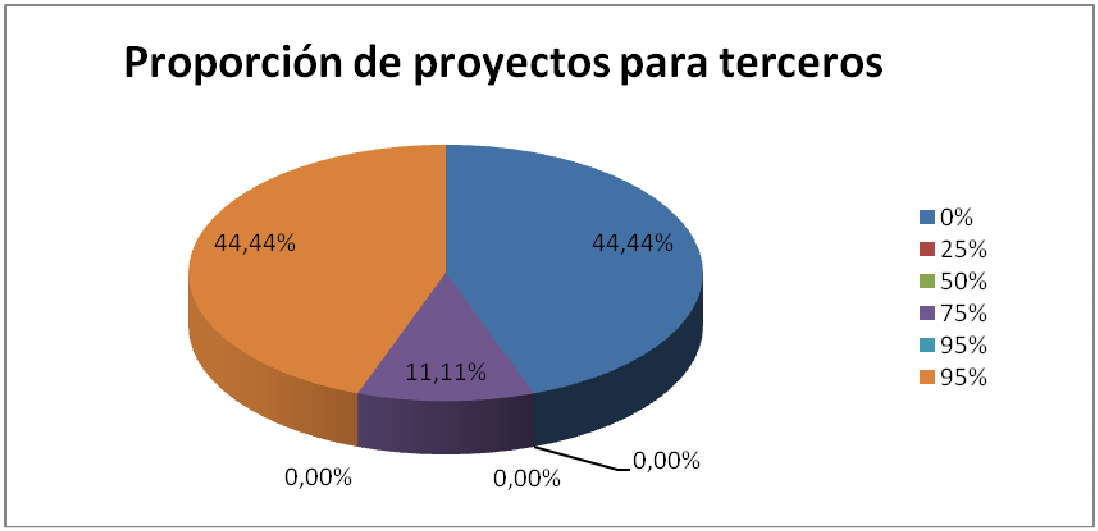


Figura 4.2.5. Proporción de proyectos desarrollados para terceros

4.2.2.4 Personas involucradas

- ***Cantidad de personas involucradas:***

La cantidad de personas influye en la forma de trabajo y organización de los equipos, por eso se consideró realizar esta pregunta. La media de la cantidad de personas involucradas en el sector de desarrollo de software obtenida es de aproximadamente 8 personas. Se plantean situaciones extremas, con 25 personas en el equipo de desarrollo por un lado, así como 2 personas en el otro extremo. El 66.66% de los sectores está integrado con más de 5 personas (Ver figura 4.2.6).

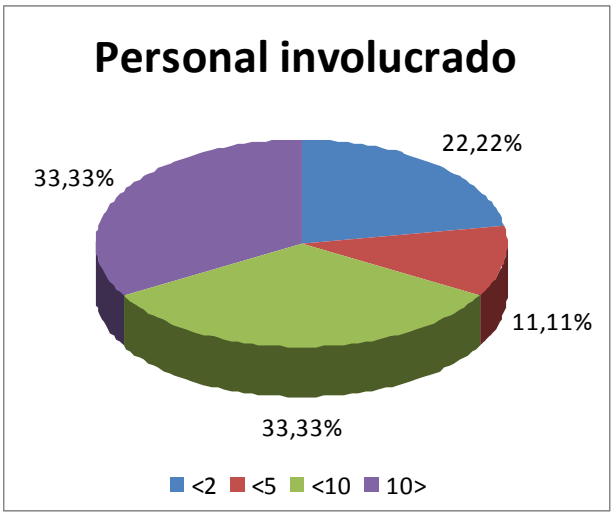


Figura 4.2.6. Cantidad de personal involucrado en el sector de desarrollo.

- ***Equipos de desarrollos internos***

Dentro de un sector de desarrollo existen equipos de trabajo internos y conocer su tamaño también es algo a considerar. El 66.67% de las áreas de desarrollo de software cuentan con más de un equipo de desarrollo (Ver figura 4.2.7).

Figura 4.2.7. Existencia de varios equipos dentro del sector de desarrollo.

Los equipos de desarrollo integrados con una cantidad entre 3 y 5 personas cubren el 44% de los relevados. Si se los agrupa y considera a los equipos conformados entre 3 y 7 personas, es de 77.77%.

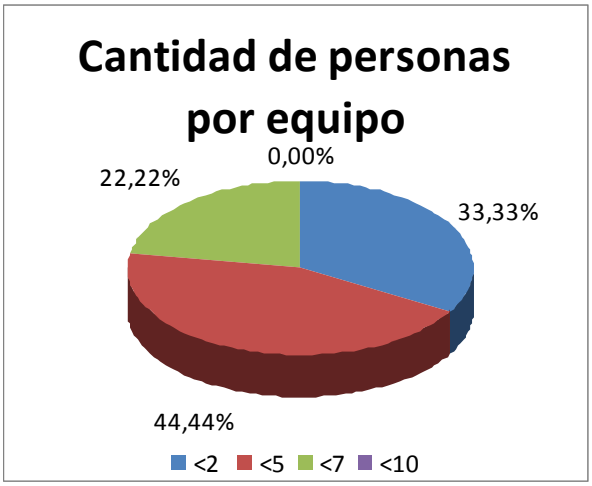


Figura 4.2.8. Cantidad de personas por equipo dentro del sector de desarrollo.

4.2.2.5 Horarios y lugar de trabajo del personal

- ***Horario del personal***

Conocer el horario del personal es un dato importante si se piensa aplicar una metodología ágil. Por esa razón, se preguntó si las personas comparten el horario de trabajo y se brindaron tres alternativas: SIEMPRE / SE SOLAPAN / NUNCA. Las personas involucradas en el desarrollo comparten el horario de trabajo en el 50% de los casos (Ver figura 4.2.9). Si se considera solo dentro del sector público, el 100% comparte el horario de trabajo siempre. En el caso de empresas privadas, existe variación.

Figura 4.2.9. Personal que comparte horario de trabajo dentro del sector de desarrollo.

Dentro de las empresas privadas, aquella que realiza solo desarrollos internos, el personal del sector de desarrollo comparte el horario de trabajo de todos miembros del equipo. El 60% nunca comparten el horario de trabajo de todos los miembros del equipo. Esto se debe a que en la gran mayoría hay diferentes turnos de trabajo, y, si bien se trata de armar los grupos con personas que estén en el mismo horario, esto no es factible la mayoría de las veces.

- ***Lugar de trabajo del personal***

Otra característica que influye en la forma que trabaja y se comunica el equipo de desarrollo es compartir o no el lugar de trabajo. Se solicitó contestar SI / NO a esta pregunta. El 88.89% comparte el lugar de trabajo, si bien esto no implica necesariamente que compartan el horario de trabajo. Como se puede apreciar, el porcentaje de personas que comparten el lugar de trabajo es mucho mayor al porcentaje mencionado en el punto anterior que comparte el horario de trabajo.

Dentro del sector de desarrollo de empresas públicas, todos comparten el lugar de trabajo, mientras que en las privadas solo el 83.33% lo hace (Ver figura 4.2.10).

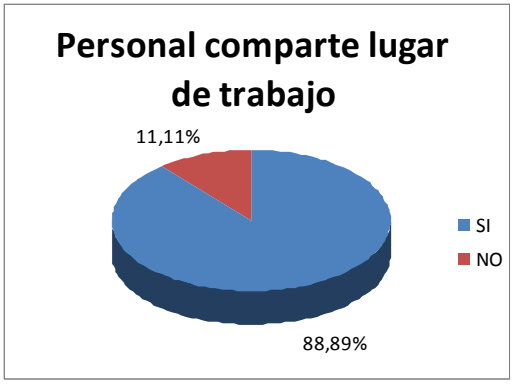


Figura 4.2.10. Personal que comparte lugar de trabajo dentro del sector de desarrollo.

El no disponer de un horario común y/o lugar de trabajo común hace indispensable definir instrumentos que permitan mantener al equipo de desarrollo comunicado. Se propuso, a la hora de realizar el relevamiento, una serie de vías de comunicación para que se indique cuales se emplean: VIDEO CONFERENCIA / CHAT / TELEFONO / CORREO / MEMO / REUNIONES PACTADAS / OTRAS.

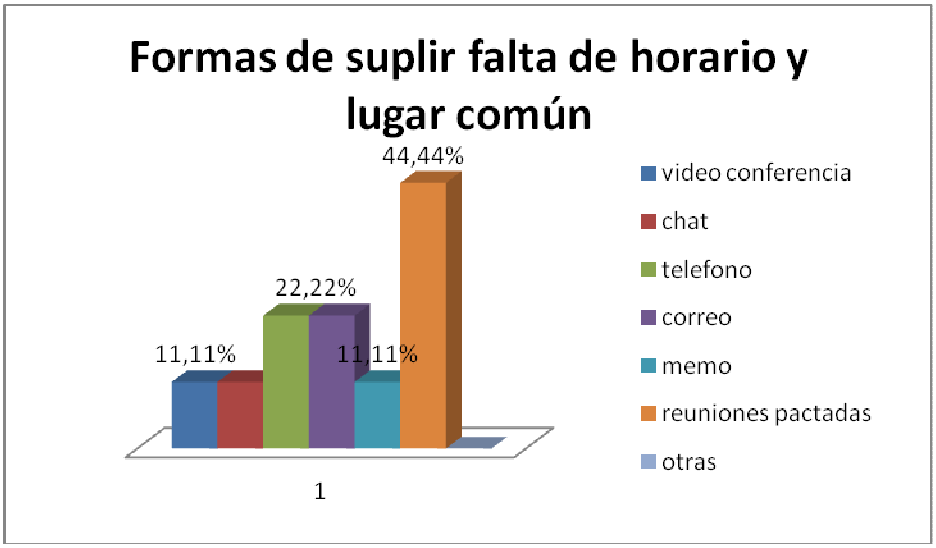


Figura 4.2.11. Formas de suplir la falta de un horario y lugar común de trabajo.

Se puede observar en Figura 4.2.11 que el recurso más utilizado es el de las reuniones pactadas. Le sigue el correo y teléfono para mantener la comunicación del equipo. Y finalmente, las video-conferencias, el chat y los memos son otros recursos que algunos sectores de desarrollo utilizan. No surgió otro recurso que sea utilizado para suplir la distancia física y temporal de los miembros del equipo que no fuera alguna de las propuestas presentadas.

4.2.2.6 Proyectos simultáneos

Todas áreas de desarrollo consultadas trabajan en más de un proyecto a la vez. A su vez, todos los equipos dentro del área de desarrollos de cada organización trabajan en más de un proyecto al mismo tiempo.

4.2.2.7 El rol del líder de proyecto

En todos los sectores relevados existe un líder de proyecto. Por otro lado, el líder de proyecto es responsable de más de un proyecto a la vez en el 100% de las áreas de desarrollo consultadas. La cantidad de proyectos que suele manejar al mismo tiempo es entre 2 y 5 proyectos (Ver Figura 4.2.12). Dentro de estos proyectos, no solamente se cuenta los proyectos de desarrollo siguiendo Desarrollo Basado en Conocimiento, sino otros proyectos de la empresa que se encuentran asignadas al líder. Además, el líder puede tener a su cargo más de un grupo de desarrollo, situación que vuelve más complejo poder coordinar y supervisar los grupos.

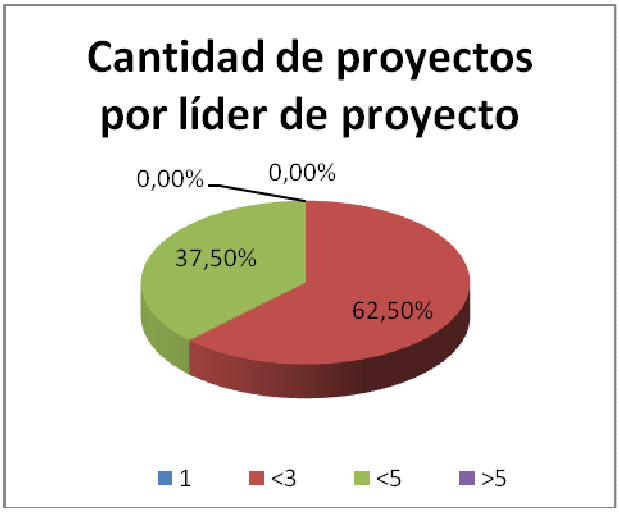


Figura 4.2.12. Cantidad de proyectos por líder de proyecto

En todos los sectores que pertenecen a empresas privadas de desarrollo de software para terceros, se presenta el caso que el rol del líder de proyecto lo realiza el dueño o los socios gerentes de la empresa. Los dueños no delegan responsabilidades al equipo. En uno de los casos, el entrevistado quien era uno de los socios de una empresa privada, hizo referencia explícita a un intento por delegar a otra persona el rol de líder de proyecto pero declaró no haber tenido buenos resultados. Debido a ello, se sienten sobrecargados de tareas y responsabilidades pero consideran que por el momento es la única forma de que se logren alcanzar los objetivos.

4.2.2.8 Conclusiones de sección

Queda claro que los sectores de desarrollos relevados tienen bastante trayectoria en el desarrollo con esta metodología. Lo que permite obtener datos valiosos para analizar.

Los recursos humanos que conforman las áreas de desarrollo de los diferentes organismos/empresas si bien son variables, en todos los casos permiten poder aplicar una metodología ágil o en su defecto ciertas prácticas ágiles. El personal de estos sectores no es numeroso, y en ninguno de los casos supera las 10 personas por equipo. Esto permite una comunicación e interacción fluida entre los miembros del equipo.

En el 50% de los sectores de desarrollo analizados, las personas que lo conforman comparten el horario de trabajo. La falta de un horario compartido podría ser un impedimento a la hora de querer aplicar una metodología ágil, pero se puede intentar suplir esta ausencia considerando que el 89% trabaja en el mismo lugar. Compartir el lugar de trabajo aunque sea en diferentes horarios permite dejar ciertos recursos a la vista y disponibles para quienes estén trabajando en el sector, pizarras, etc. Además, un 12% tiene personal con horario solapado, permitiendo en algunos momentos el diálogo cara a cara.

Relacionado con este aspecto, los sectores de desarrollo que expresaron la necesidad de suplir la falta de comunicación cara a cara por no compartir el horario y/o lugar de trabajo, todos expresan que utilizan reuniones pactadas. Por lo que en situaciones especiales se puede contar con la comunicación directa entre los miembros del equipo. Esto implica que existen mecanismos internos que permiten reunir a todo el personal del equipo para alguna situación especial.

Todos los equipos dentro del área de desarrollos de cada organización trabajan en más de un proyecto al mismo tiempo. Esto puede causar situaciones no deseables en algunos de los proyectos. Además, no es una característica deseable a la hora de trabajar con metodologías ágiles si se trabaja con iteraciones. La alternativa podría ser ver el proyecto como flujo de trabajo

continuo donde pueden entremezclarse varios proyectos sin demasiadas complicaciones. Para cada funcionalidad o tarea a realizar se define una historia de usuario y se coloca en la lista de manera priorizada. Para poder visualizar este flujo de trabajo continuo sería conveniente incorporar el uso de un Kanban board que puede ser una pizarra tradicional o una pizarra web.

Dentro del sector privado, existen empresas con áreas de desarrollo que utilizan Genexus exclusivamente para desarrollo interno y otras que lo utilizan para proveer servicios de desarrollo a terceros. En sector público lo utilizan para desarrollo interno.

En todos los sectores relevados por lo menos existe un líder de proyecto. El líder de proyecto siempre es responsable de más de un proyecto a la vez y suele manejar entre 2 y 5 proyectos. Además, el líder puede tener a su cargo más de un grupo de desarrollo. En todos los sectores que pertenecen a empresas privadas de desarrollo de software para terceros, se presenta el caso que el rol del líder de proyecto lo realiza el dueño o los socios gerentes de la empresa. Los dueños no delegan responsabilidades al equipo. Esto demuestra poca confianza hacia ellos. Esta tampoco es una característica deseable en las metodologías ágiles, va en contra del propio manifiesto ágil, que habla de equipos auto-organizados. El dueño de la empresa o líder de proyecto debe confiar en el equipo y darles su espacio. La alternativa para poder intentar cambiar esto es proponer varios roles diferentes que vayan desdibujando el rol del líder de proyecto: un analista que represente las preocupaciones del cliente y defina prioridades, un diseñador líder que guíe en el diseño mayor del sistema, un coordinador que realice el seguimiento del avance y proteja de interrupciones al equipo.

4.2.3 Forma de trabajo con Desarrollo Basado en Conocimiento

Este grupo de preguntas buscaba poder relevar información en diferentes aspectos relacionados con la forma de trabajo del grupo de desarrollo al emplear DBC. Se consideró necesario consultar sobre: el procedimiento para el desarrollo software, la participación del cliente, los requerimientos en los proyectos desarrollados, la planificación y seguimiento del proyecto, el desarrollo con reutilización, los resultados obtenidos en los proyectos realizados, los riesgos en proyectos de desarrollo, la utilización de metodologías o prácticas ágiles, la razón de la elección de Bases del Conocimiento, uso de herramientas relacionadas con este desarrollo. Además, se brindó la posibilidad de realizar cualquier otro comentario adicional que se considerara conveniente compartir y no hubiere sido recabado anteriormente.

4.2.3.1 Procedimiento para el desarrollo software utilizando DBC

- ***Existencia de un procedimiento fijo a seguir para cada desarrollo***

A la hora de desarrollar software, muchos grupos de desarrollo tienen una serie de pasos definidos, ya sea formal o informalmente para realizar la actividad. Se consideró importante poder determinar si existía algún procedimiento preestablecido para realizar el desarrollo del software basado en conocimiento. Se utilizó el término procedimiento para evitar que se nombre directamente una metodología y se pueda relevar la forma de trabajo real. En el 77.78% existe un procedimiento predefinido (Ver Figura 4.2.13)

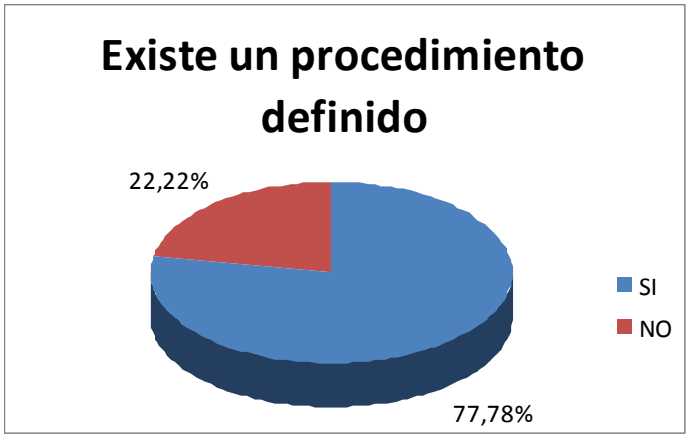


Figura 4.2.13. Existencia de un procedimiento predefinido para el desarrollo.

Analizando de manera separada, para sectores de desarrollo pertenecientes a empresas privadas y organismos públicos, el 83,33% de los grupos de desarrollo pertenecientes al sector privado lo tiene definido y mientras que, en los organismos públicos, el porcentaje es menor, solo el 66,67% de ellos (Ver Figura 4.2.14)

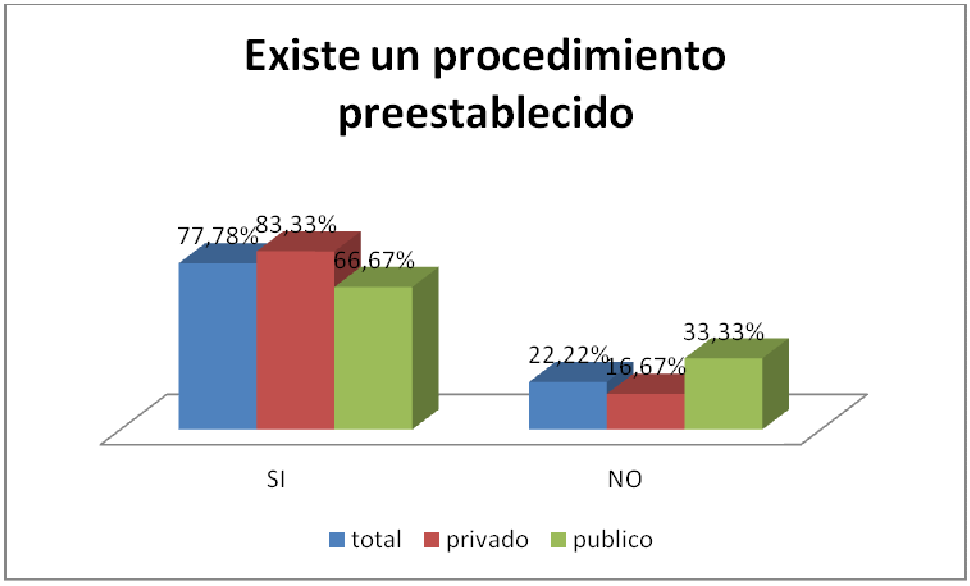


Figura 4.2.14. Existencia de un procedimiento predefinido para el desarrollo, discriminado por tipo de organización.

- ***Descripción de los procedimientos para el desarrollo de software con Desarrollo Basado en Conocimiento***

Se solicitó a cada responsable que describiera la forma de encarar un desarrollo, de manera abierta. Los detalles brindados por cada uno pueden consultarse en el Anexo 1.

Realizando un análisis de los procedimientos, se puede observar a simple vista que algunos tienen procedimientos detallados y otros muy generales. Analizando la forma de trabajo según el desarrollo basado en conocimiento, solo tres sectores de desarrollo plantean un procedimiento que contempla éstas características, el resto plantea tareas o actividades que están más relacionadas con un desarrollo más tradicional. Algunos definen dos procedimientos dependiendo del tamaño del proyecto. Se pueden realizar las siguientes apreciaciones:

- Los instrumentos que utilizan para describir los requerimientos son bastante heterogéneos: Historias de usuario (HU), minutas, modelos conceptuales para describir las necesidades. Algunos utilizan DER o diagramas de clases.
- Solo un 36.36% mencionan alguna actividad de pruebas ya sean unitarias y/o de aceptación.
- Solo una empresa permite que cada miembro del equipo interactúe y/o visite al cliente
- Una sola empresa tiene registros muy formales del proceso de desarrollo, si bien ellos mismos manifiestan que han flexibilizado un poco dicho proceso.
- Solo una empresa, en proyectos grandes, divide en fases menores y solo analiza con detenimiento esa fase.

- ***Adaptación del procedimiento según el proyecto***

Así como se consideró oportuno preguntar sobre la existencia de un procedimiento a seguir para el desarrollo de software, se consultó sobre la posibilidad de adaptarlo según las características del proyecto a desarrollar. Se permitía como respuesta: SI / NO / DEPENDE. En caso de esta última opción se requería mayores detalles.

Según lo relevado, el 62.50% permite al equipo adaptar el procedimiento según el proyecto (Ver Figura 4.2.15). Esto haría pensar que las empresas permiten a los equipos organizarse para encarar el proyecto de manera autónoma e independiente. Lo que sucede es que, los dueños de empresas de desarrollo son a su vez líderes del proyecto y miembros del equipo y son ellos los que adaptan el proceso. No es una decisión del equipo en su conjunto. Esto se podrá observar en el siguiente ítem analizado.

Algo a considerar es que uno de los encuestados respondió que no tiene un procedimiento predefinido, pero en esta pregunta contestó que el procedimiento lo pueden adaptar según el proyecto. Esto no es incongruente, porque se define el procedimiento para cada proyecto en particular.

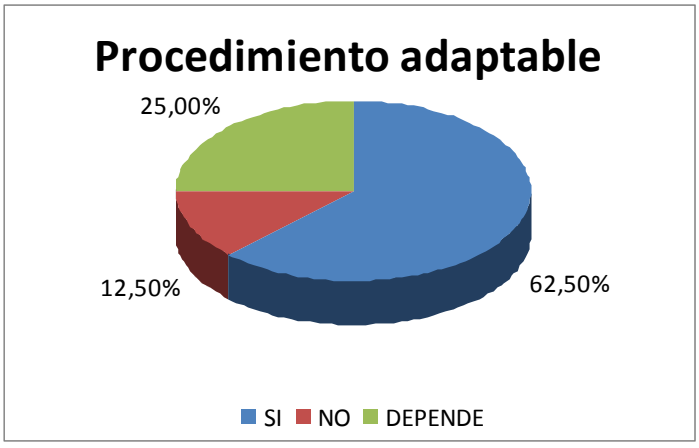


Figura 4.2.15. Adaptabilidad del procedimiento de desarrollo.

- ***Responsable de definir el procedimiento a seguir en un proyecto***

Exista o no un procedimiento preestablecido, alguien define los pasos a seguir en un proyecto. Se consultó sobre quién define el procedimiento a seguir en cada proyecto de desarrollo. Se brindaron tres opciones: LA DIRECCIÓN / EL LÍDER DE PROYECTO / EL EQUIPO.

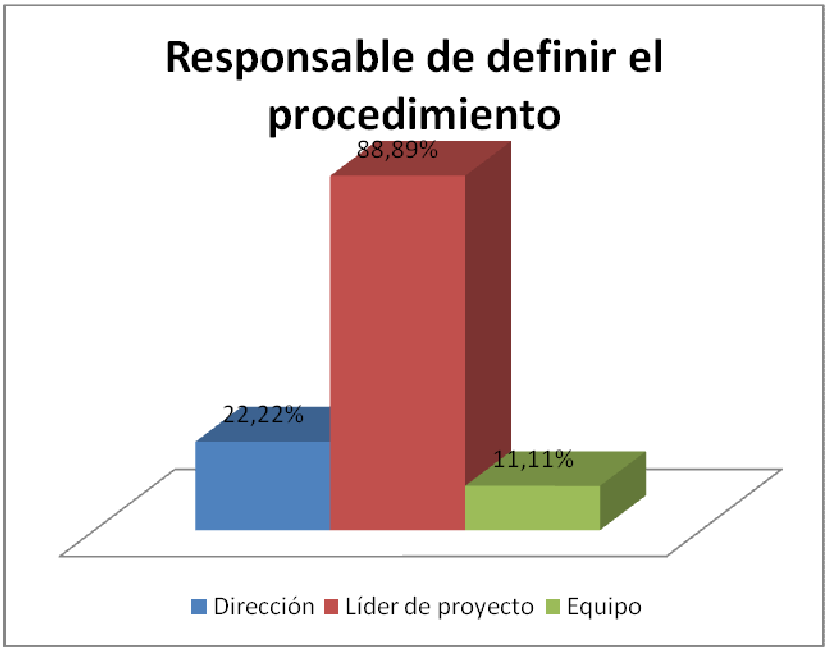


Figura 4.2.16. Responsable de definir el procedimiento de desarrollo.

El 88.89% mencionó al líder como responsable de esta actividad (Ver Figura 4.2.16). El 11% que consideró al equipo, señaló que lo define junto con el líder de proyecto. Por otro lado, el 22.22% que corresponde a grupos de desarrollos donde el dueño es el que realiza la definición del procedimiento a seguir indicó que los dueños son a su vez los líderes del proyecto. Entonces, para este caso, si bien respondieron con una sola opción, los dueños, debería considerarse también como parte de su respuesta no solo al dueño sino también al líder del proyecto. Si se analizan los datos, teniendo en cuenta estas consideraciones, en el 100% de los casos el líder de proyecto está involucrado en la definición del procedimiento de desarrollo.

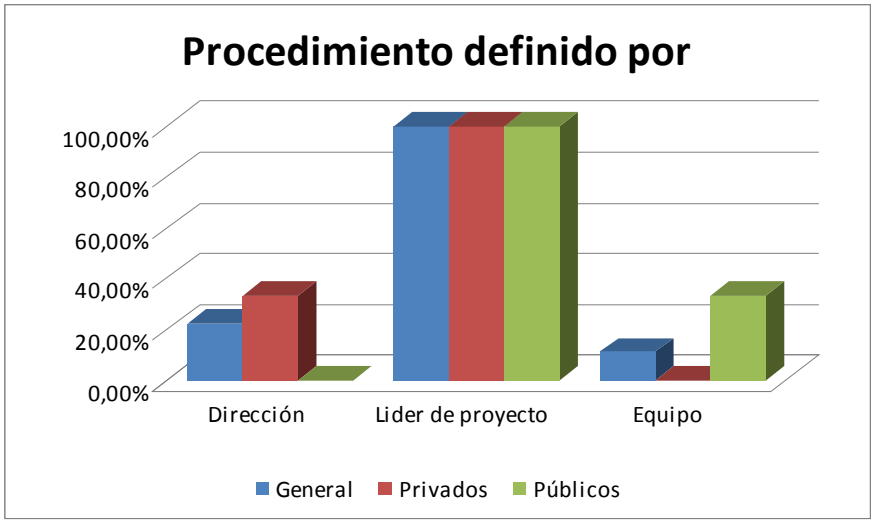


Figura 4.2.17. Responsable de definir el procedimiento de desarrollo por tipo de organización.

Entonces, se observa que, en 77,78% de los sectores de desarrollo, es sólo el líder el responsable de esta decisión (Ver Figura 4.2.17 y Figura 4.2.18). En un 22,22% es el líder y la dirección quienes tiene esta responsabilidad (pero la misma persona es la que desempeña ambos roles).

Solo el 11,11% restante define el procedimiento de manera conjunta entre el líder del equipo y equipo de desarrollo.

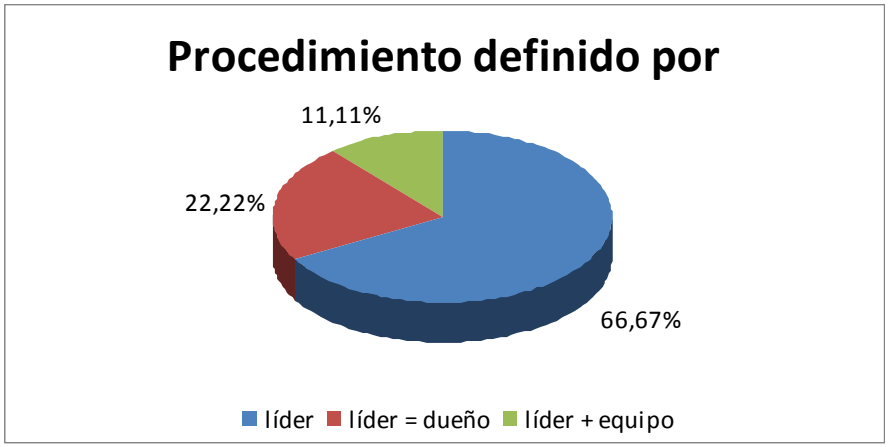


Figura 4.2.18. Responsable de definir el procedimiento de desarrollo. Vista agrupada.

- ***Desarrollo iterativo***

Se consultó si la empresa realizaba el desarrollo de software de manera iterativa. Se permitía contestar: SI / NO. Luego se pedía definir la duración de la iteración en caso de haber contestado afirmativamente. El 66.67% de las empresas contestó que desarrolla de manera iterativa (Ver Figura 4.2.2.19).

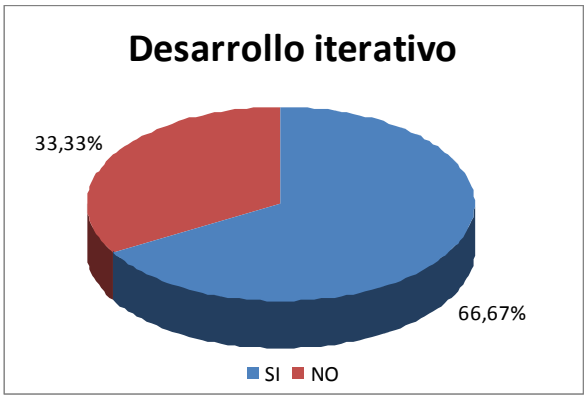


Figura 4.2.19. Sectores que trabajan con un desarrollo iterativo.

Las respuestas obtenidas sobre las duraciones de las iteraciones se pudieron organizar en 3 grupos: DURACIÓN DE ITERACIONES VARIABLES / ITERACIONES DE 7 A 14 DÍAS / NO HAY ITERACIONES.

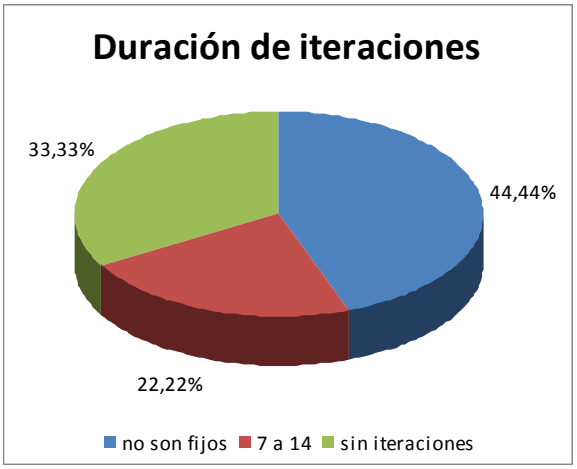


Figura 4.2.20. Duración de las iteraciones.

Analizando los datos según estas agrupaciones, el 44.44% no trabaja con iteraciones de duración fija, generalmente las definen por los productos que deben realizar, son por lo tanto duraciones variables (Ver Figura 4.2.20). Solo un 22.22% trabaja con duraciones fijas. El 33.33% ni siquiera trabaja con iteraciones, uno de los grupos que no trabaja con iteraciones aclaró que trabajan entregando al cliente los productos que le son más importantes al cliente a medida que los van desarrollando

- ***Uso de UML***

Para tratar de entender un poco más cómo realizan el diseño de las aplicaciones a desarrollar, y complementar la definición brindada por cada grupo de desarrollo sobre la forma de encarar un desarrollo software, se consultó sobre el uso de diagramas UML. Se permitió una contestación simple: SI – NO. Además se solicitó a quienes contestaron afirmativamente que mencionen cuáles utiliza. El 55.56% utiliza algún diagrama de UML (Ver Figura 4.2.21).

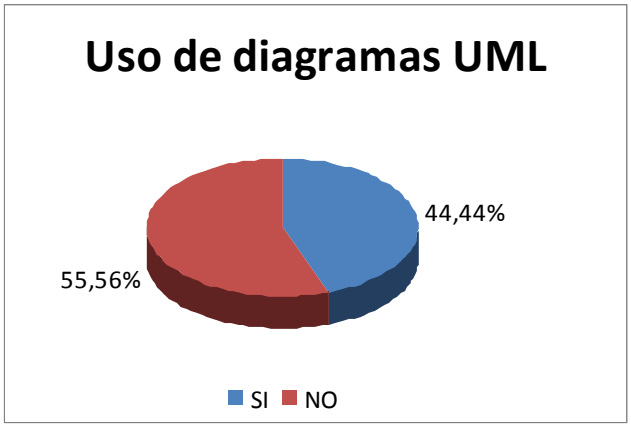


Figura 4.2.21. Sectores que utilizan diagramas UML.

Analizando los diagramas utilizados, el 33.33% utiliza Diagramas de Casos de Uso y el 22.22% Diagramas de Clases. Es lógica la utilización de Casos de Uso para poder especificar las necesidades del proyecto y los Diagramas de Clases, para poder trabajar directamente definiendo la Base de Datos del Conocimiento. (Ver Figura 4.2.22). Pero, ninguno hizo mención a algún otro de los diagramas, ni siquiera para situaciones excepcionales.

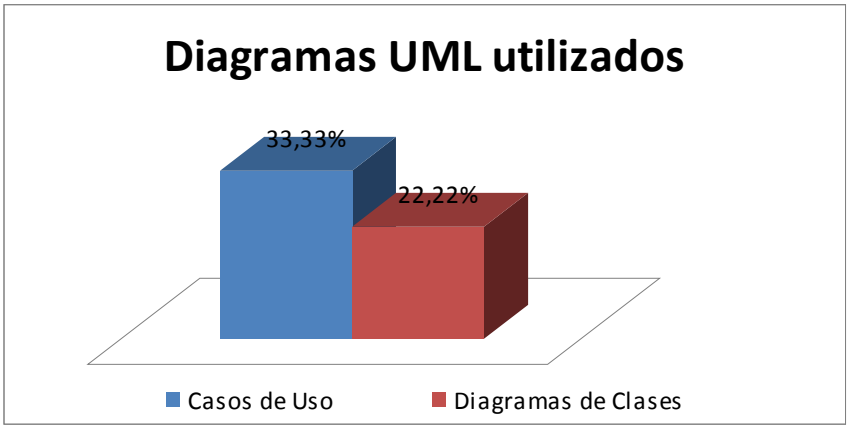


Figura 4.2.22. Diagramas UML utilizados por los sectores de desarrollo.

4.2.3.2 Participación del cliente

- ***Participación del cliente en el proceso***

Cómo se incluye al cliente a lo largo del proceso de desarrollo es muy importante, sobre todo si se está pensando incorporar una metodología ágil. Se consultó como era la participación del cliente en el proyecto. Las opciones que se brindaban fueron: SOLO AL INICIO / DURANTE TODO EL PROCESO / AL FINAL. El 100% contestó que el cliente está involucrado durante todo el proceso de desarrollo. Este punto además solicitaba especificar la forma en que obtenía la participación del cliente, permitiendo realizar una respuesta abierta. Los detalles brindados por cada uno pueden consultarse en el Anexo 1.

Todos de alguna manera buscan tener contacto con el cliente. Esto muestra la importancia que tiene para cada equipo de desarrollo. El diálogo y la comunicación, es lo que podría encontrarse en común en la mayoría de las respuestas.

Se observa que uno solo de ellos considera presentaciones de informes de avance, documentos donde se describe lo que están realizando más que software funcionando. Y también, uno solo hace uso de la presentación de prototipos de manera rápida al cliente para corroborar los requerimientos, siendo ésta una de las características que permite fácilmente Genexus. Esto podría demostrar que no lo consideran como una forma de participación del cliente de manera intuitiva.

- ***Entregas frecuentes al cliente***

Otro aspecto a analizar es sobre, si se realizan o no, entregas frecuentes (a lo sumo cada 2 meses) de producto desarrollado al cliente y el período de tiempo entre entregas ya que esto también es un punto relevante dentro de las metodologías ágiles. El 88.89% contestó que sí realiza entregas frecuentes. Y solo una de ellas tiene preestablecido un período de tiempo para las entregas que va de 15 días a un mes. El resto se organiza por módulos o porciones de software a entregar (Ver Figura 4.2.23).

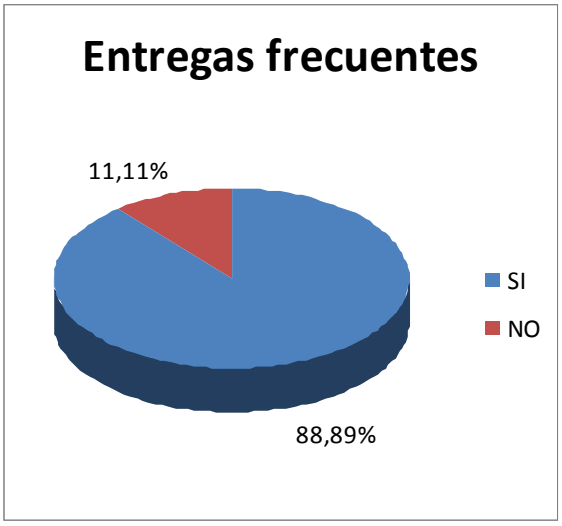


Figura 4.2.23. Sectores que trabajan realizando entregas frecuentes.

En el 22.22% de los casos, inclusive varía la frecuencia, incrementándose a medida que avanza el proyecto. Uno solo hace mención que buscan entregar pequeñas funcionalidades que vayan despertando el interés de los usuarios. La mayoría entrega módulos completos, pero no se fija plazos máximos para ellos, dependiendo de lo que se está implementando.

- ***Colaboración del cliente***

En este punto, colaboración permanente del cliente, se solicitaba responder SI / NO y que se explique cómo se concretaba la colaboración, si la respuesta era afirmativa. Se obtuvo un 100% de respuestas afirmativas.

A la hora de analizar los datos complementarios que explicaban la respuesta afirmativa, se observa que no todos tienen la misma idea de colaboración del cliente en el proceso de desarrollo. Para algunos es trabajar con ellos para probar el sistema, para otro es brindarles información del avance del proyecto y para otros la colaboración del cliente en el proyecto es poder realizarles consultas cuando sea necesario, es decir tener una comunicación directa con él en diferentes momentos. Teniendo en cuenta estos tres grupos, se clasificaron las respuestas de cada organización y se pudo realizar la siguiente gráfica, (Ver Figura 4.2.24).

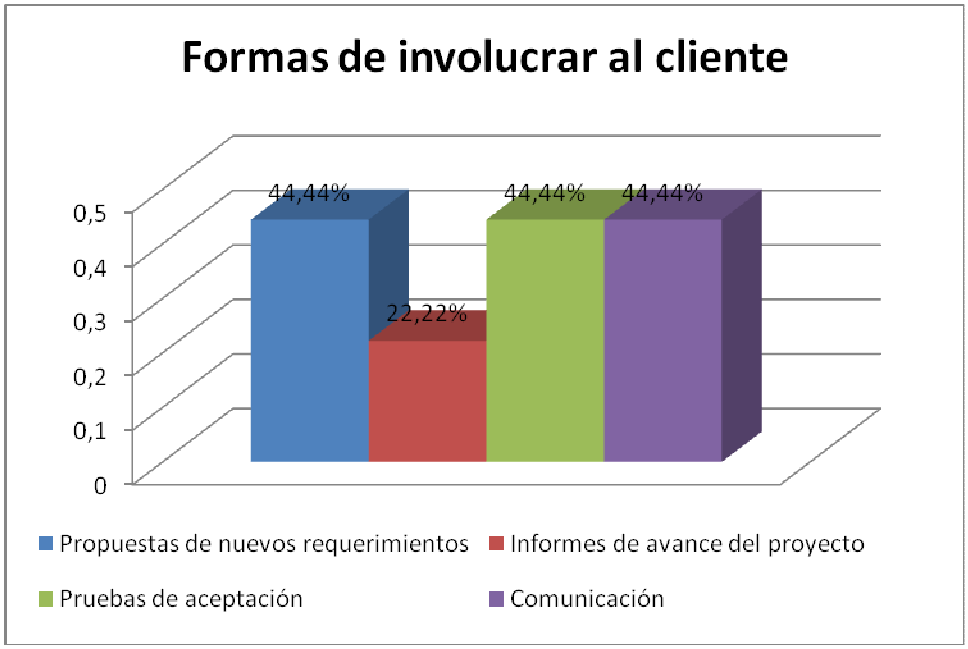


Figura 4.2.24. Formas en que los sectores de desarrollo involucran al cliente.

Cada sector relevado trabaja de manera diferente, algunos combinan propuestas de informes de avance con propuestas de nuevos requerimientos, otros solo consideran propuestas de requerimientos, otros combinan la propuesta de requerimientos con comunicación y otros incluso le agregan las pruebas de aceptación. Pero las combinaciones son todas diferentes entre los datos relevados. No se puede determinar un patrón de combinación de la forma de participación. No hay una tendencia en cómo combinarlos, siendo las de mayor proporción: la participación del cliente para proponer nuevos requerimientos, realizar pruebas de aceptación y mantener cierta comunicación con el equipo de desarrollo.

4.2.3.3 Requerimientos en los proyectos desarrollados con DBC

- ***Definición de requerimientos***

Respecto al momento de definir los requerimientos del proyecto, todas las empresas relevadas respondieron que los requerimientos se definen al inicio del proyecto. Se solicitó, además, que especifiquen la forma en que se define los requerimientos. En base a las especificaciones obtenidas se puede desglosar la siguiente formación:

- La forma de obtener los requerimientos es a través de entrevistas, reuniones, encuestas.
- En la mayoría de los casos, 88.89%, se realiza detalle completo de todos los requerimientos del cliente. En solo un caso, que representa el 11.11%, se aclaró que no se realizan especificaciones en detalle de todo el producto software a desarrollar, solo generales y únicamente se realiza una especificación completa de la porción a entregar en primer lugar.
- Un 22.22%, comparten el documento realizado para que el cliente realice modificaciones.
- Un 11.11%, hace énfasis en los puntos de verificación y en algunos detalles específicos de tablas/objetos/estructuras.

- Los instrumentos mencionados en algunos casos para definir los requerimientos son los siguientes: mapas conceptuales, Historias de usuario, minutas de reuniones y documento BluePrint BBP.

- ***Cambios en los requerimientos***

Esta pregunta pretendía evaluar cuan frecuente es el cambio en los requerimientos del proyecto. Se brindaban las siguientes opciones: SI SIEMPRE / ALGUNAS VECES / CASI NUNCA / NO CAMBIAN. El 77.78% respondió que siempre cambian los requerimientos y los restantes que algunas veces, (Ver Figura 4.2.25). No existió ningún sector de desarrollo de organismo público o empresa privada que considere que los requerimientos no cambian o cambian muy pocas veces.

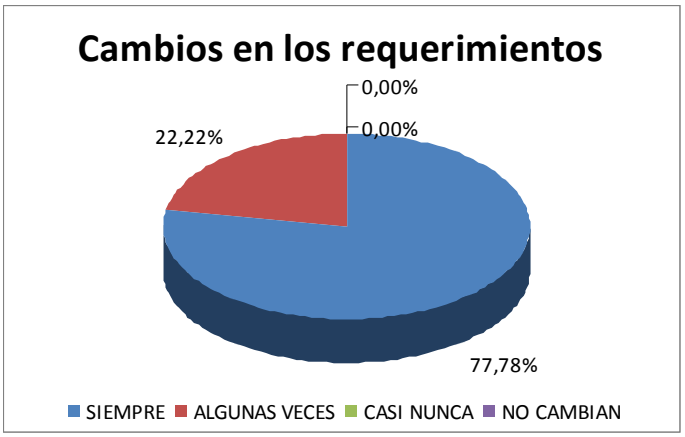


Figura 4.2.25. Frecuencia con que se producen cambios de los requerimientos.

4.2.3.4 Planificación y seguimiento del proyecto

- ***Planificación del desarrollo***

Se evaluó el momento, dentro del proyecto, cuando se realiza la planificación del desarrollo. Se presentaban las siguientes opciones: AL INICIO / AL INICIO PERO SE VA MODIFICANDO ENCADA ITERACION / OTRA OPCION (especificar). Esta fue la respuesta obtenida: el 11% planifica solo al inicio, el 44.44% si bien planifica al inicio puede modificar la planificación en cada iteración y otra opción con el 44.44% restante (Ver Figura 4.2.26).

Merece la pena analizar qué especificaron las empresas como otra opción. A continuación se enumeran las 4 respuestas obtenidas:

- al inicio pero se va modificando en el transcurso (no hay iteraciones)
- al inicio pero se va modificando en el transcurso
- al inicio pero se va modificando a lo largo de la implementación de los módulos
- al inicio pero se puede planificar en algunas iteraciones cuando se encuentre desviaciones del proyecto.

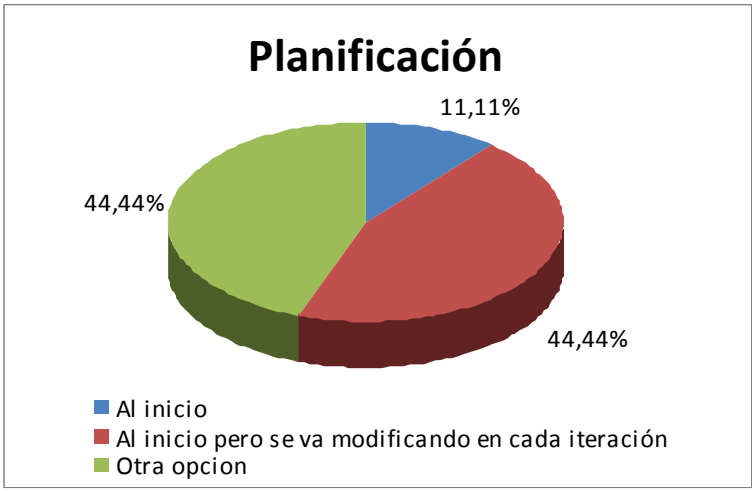


Figura 4.2.26. Momento en que se realiza la planificación.

Puede observarse que las tres primeras opciones detalladas podrían englobarse en la opción de *definir al inicio pero se puede ir modificando a lo largo del transcurso del proyecto*, sin hacer mención a las iteraciones. A este grupo correspondería un 33.33%. Y la última correspondería al grupo de "al inicio pero se va modificando en la iteración ya que permite modificar la planificación en una iteración, aumentando su porcentaje a 55.55%. Avanzando un poco más en este sentido, se puede agrupar a quienes realizan una planificación inicial y luego lo van modificando ya sea que lo realicen o no por iteración a esta modificación. Por otro lado quedan quienes realizan solo una planificación inicial. El 88.88% planifica al inicio pero a lo largo del proyecto va modificando su planificación, mientras que solo un 11.11% realiza la planificación al inicio y no la modifican.

Relacionado con la planificación se solicitaba que especifiquen la forma en que planifican, solo un 45.45% realizó esta especificación. Se destacan las siguientes respuestas:

Forma de planificar:

- Fijar una cota no rígida de plazos.
- Determinar el tiempo de realización de los módulos y determina los responsables de las tareas asociadas, los recursos necesarios y se delega a cada desarrollador la tarea a realizar, todos siguen la planificación a través de una herramienta de panificación.
- Planificar al inicio las grandes etapas del proyecto (se divide en fases o etapas). Se planifica con detalle solo la primera fase. Finalizada esta, se planifica la siguiente.
- Planificar a *ojo de buen cubero*, por similitud de complejidad de proyectos anteriores, analizando módulo por módulo

Existen algunas formas de planificar más formales que otras. Debido a la gran cantidad que no completó este punto no es representativo intentar obtener porcentajes de cuántos trabajan de manera más liviana que otros en este aspecto

- ***Herramientas utilizadas para el seguimiento de un proyecto***

En un proyecto de desarrollo de software existen muchas herramientas que pueden facilitar la gestión del mismo. Debido a esto, se recabó información de las herramientas utilizadas por los grupos de desarrollo para el seguimiento de un proyecto. Solo dos de los sectores relevados no indicó ninguna, por lo que puede considerarse entones que no utilizan ninguna herramienta para el

seguimiento del proyecto, y representa un 22.22%. El 77.78% restante, por lo tanto, utiliza alguna herramienta para realizar el seguimiento del proyecto, (Ver Figura 4.2.27).

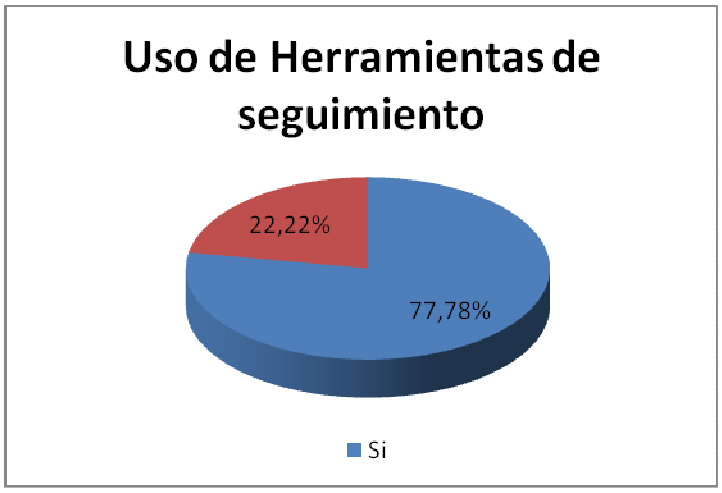


Figura 4.2.27. Sectores que utilizan herramientas para realizar el seguimiento del proyecto.

En esta pregunta se buscaba que cada grupo relevado especifique su herramienta por lo tanto se obtuvo un listado variado de herramientas: sistema de tickets, Google desktop, Microsoft Proyect, dot Project, Kanban board, Aplicativo de gestión de tareas, Correo, Excel. Los porcentajes con que fueron mencionados se pueden observar en la siguiente Figura.

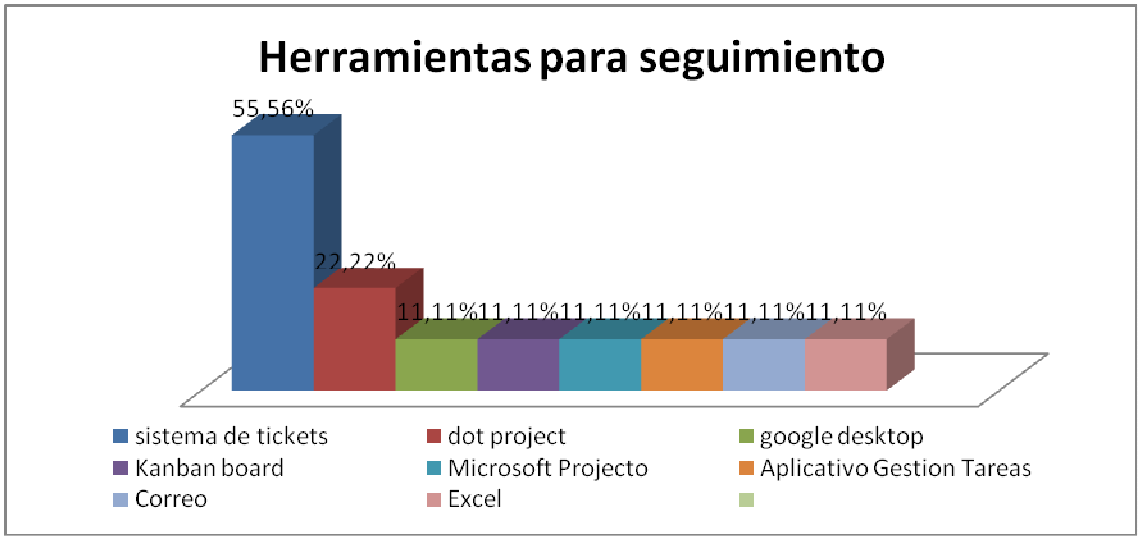


Figura 4.2.28. Herramientas utilizadas para realizar el seguimiento del proyecto.

Tres herramientas, dot Project, Microsfot Project y Aplicativo para Gestión de tareas puede englobarse en una herramienta que justamente ayuda a la gestión de tareas, definir calendario, responsables, tiempos, recursos, etc. Google desktop ayuda a la búsqueda de información y tener organizado archivos, marcadores y mensajes. Sistema de tickets, previamente descripto, permite al cliente solicitar servicios al equipo de desarrollo (modificaciones o requerimientos nuevos). Kanban board, también mencionado anteriormente, que es una pizarra para visualizar el proyecto, y permite ver el progreso del trabajo, asignación de tareas, que es lo que necesita realizarse luego, entre otras funcionalidades y que se utiliza en metodologías ágiles.

- ***Uso de BurnDown Chart***

Específicamente, como complemento a la pregunta anterior, se buscó determinar si los grupos de desarrollo de software utilizan un diagrama Burn Down para complementar el seguimiento del proyecto. Se debía contestar simplemente, SI / NO. Y se obtuvo que un 22.22% utiliza este diagrama (Ver Figura 4.2.29). Uno de ellos solo lo usa para proyectos grandes. Otro de los grupos de desarrollo planea usar a futuro pero actualmente no utiliza este diagrama.

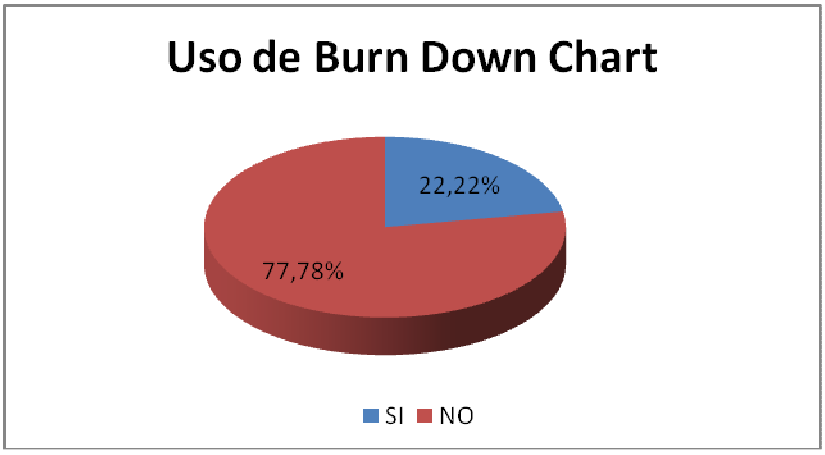


Figura 4.2.29. Sectores que utilizan Burn Down chart.

Ninguno hizo mención a este diagrama como herramienta de seguimiento del proyecto, por lo que surge la pregunta, si realmente se utiliza o no.

- ***Definición de costos y alcances***

A través de esta pregunta se buscaba determinar el momento en que los alcances y costos se pactan con el cliente. Se brindaban tres opciones: AL INICIO / PARA CADA ITERACION / OTRA OPCION (especificar) y luego se solicitaba describir brevemente como lo realizaban. Pero las respuestas merecieron dividir esta pregunta en dos partes separadas, para costos y para alcances, debido a que algunos manejaban de manera diferente costos y alcances y todos los grupos de desarrollos pertenecientes a organismos públicos no definen los costos.

Entonces, analizando primeramente la información obtenida sobre la forma de realizar la definición de alcances, se encontraron tres que eligieron la tercera opción como su respuesta. Los tres adujeron comentarios similares que podrían englobarse en una opción: *al inicio pero se va delimitando en forma más precisa durante las iteraciones*. Por lo tanto se obtuvieron los siguientes porcentajes (Ver Figura 4.2.30).

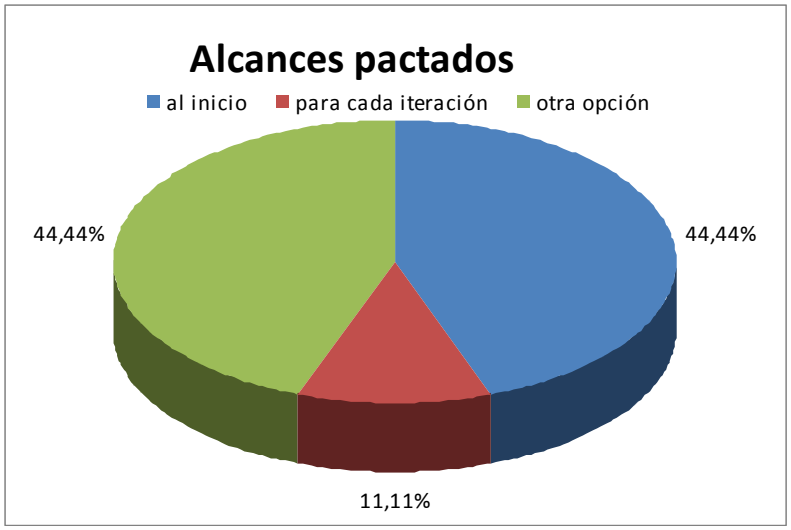


Figura 4.2.30. Momento en que se pactan los alcances del proyecto.

Respecto a los costos, en base a las respuestas y comentarios adicionales, se pudo considerar que, en este caso, otra opción era equivalente a: *no se manejan costos*. Y los que no manejan costos, como se mencionó anteriormente pertenecen a organismos públicos. Los que consideraron los costos al inicio, comentaron de diferentes formas que, en caso de variar los alcances, tratan de renegociar los costos pero esto es sumamente difícil con el cliente y deben mantener los costos definidos, si bien pueden modificar algunos plazos de entrega. En la siguiente Figura se pueden observar los porcentajes obtenidos.

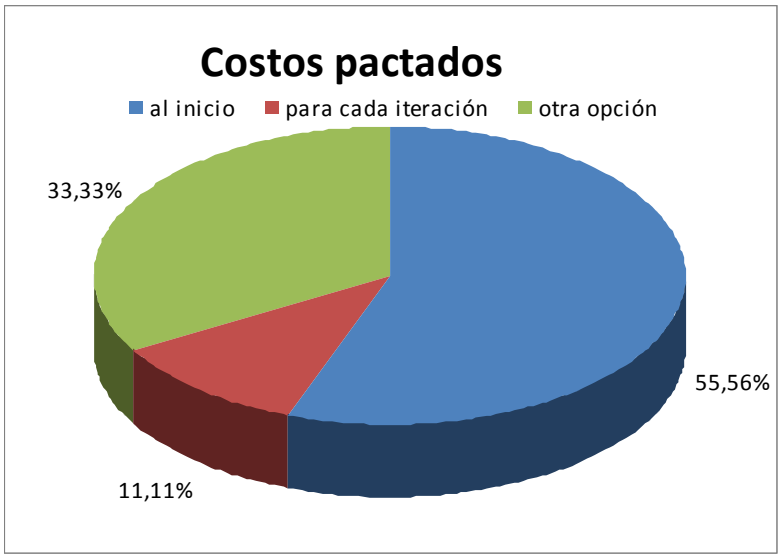


Figura 4.2.31. Momento en que se definen los costos del proyecto.

Un solo grupo de desarrollo plantea la posibilidad de definir en forma general todo el proyecto y solo en detalle una primera etapa, tanto en alcances como en costos. Esto les permite focalizarse en una parte del producto a desarrollar, definirlo con la mayor granularidad posible y establecer un precio proporcional a ese trabajo. Este grupo brinda la posibilidad al cliente de ofrecerle un análisis detallado del sistema global pero cobrándoles un costo adicional y ningún cliente se los solicitó hasta el momento

Una consideración especial que vale la pena destacar es que, como se mencionó antes, todos los grupos de desarrollo que pertenecen a organismos públicos no definen los costos. La razón aducida es que ya existen sueldos para el personal y esos son los costos. Pero, esto hace pensar que tal vez no se realiza un análisis costo - beneficio del proyecto de desarrollo a implementar, la sola justificación que existe un monto del organismo destinado a pagar sueldos, no justifica tal vez la realización de un proyecto. Se debería iniciar el desarrollo de acuerdo a un análisis costo - beneficio, indicando el tiempo de recupero de la inversión aunque sea del Estado. Así también, se debería considerar la innovación y cambio continuo del mercado.

4.2.3.5 Desarrollo con reutilización

- ***Desarrollo con Reutilización***

En el desarrollo de software la reutilización de lo que se tiene desarrollado es muy importante, por tanto se consultó si se desarrolla con reutilización. Pregunta al que el 100% contestó afirmativamente. Luego, se pedía que se especifique el porcentaje de reutilización de proyectos anteriores, siendo aproximadamente un 56% aquellos que reutilizan entre el 20% y 50%, le sigue luego los que reutilizan más del 70% y uno solo que representa al 11% reutiliza menos del 20% (Ver Figura 4.2.32).

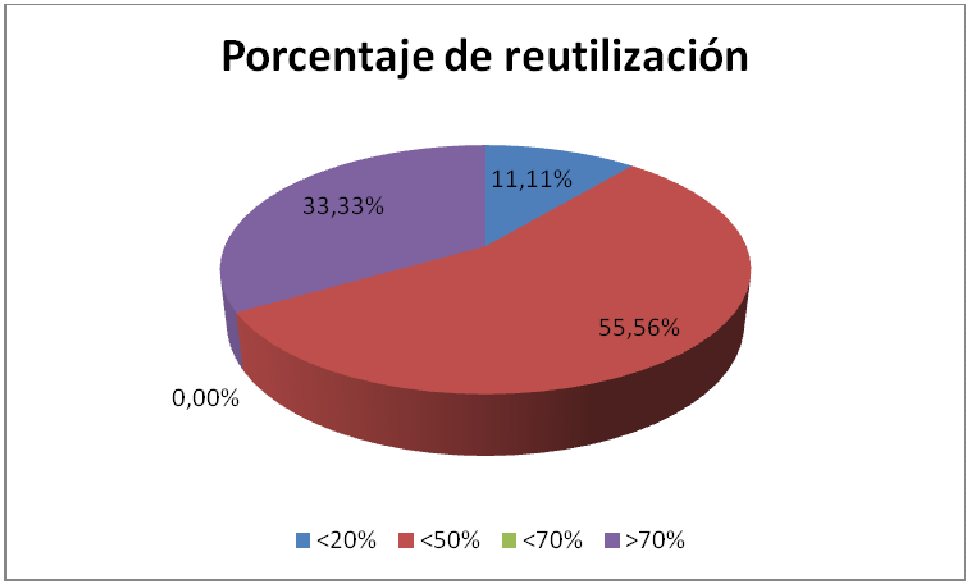


Figura 4.2.32. Porcentaje de reutilización en los proyecto.

Cómo se seleccionan los artefactos a reutilizar, es una tarea no muy simple y fue consultada también a los diferentes grupos de desarrollo. La gran mayoría, el 66,66% delega en el líder del proyecto la búsqueda y selección de artefactos a reutilizar, el resto no especifica quien realiza la tarea, pudiendo ser el líder o tal vez el equipo o un trabajo en conjunto.

Por otro lado, el 22.22% hace mención a un catálogo o librería donde tiene organizados los artefactos o módulos de proyectos anteriores que pueden reutilizarse y en donde debe realizarse la búsqueda. Un grupo explícitamente declara no utilizar ningún catálogo para realizar esta selección. El resto no explicita nada al respecto.

Aquellos que respondieron este ítem y no hicieron mención a ningún instrumento para organizar los artefactos reutilizables, se limitaron a responder que la reutilización es por similitudes de funcionalidades. Algo que es completamente lógico, dado que no se van a reutilizar objetos que no tengan funcionalidades similares a las requeridas.

4.2.3.6 Resultados obtenidos en los proyectos realizados

- ***Cumplimiento de Objetivos***

Se solicitó a los representantes de los grupos de desarrollo de software, tratar de definir un porcentaje de proyectos donde se cumplieron con los objetivos pactados con el cliente. Se dio libertad para que cada uno exprese su valor. En promedio se obtuvo que el 88.11% de los proyectos cumplió con los objetivos del cliente.

Agrupando los valores se obtiene la siguiente gráfica:

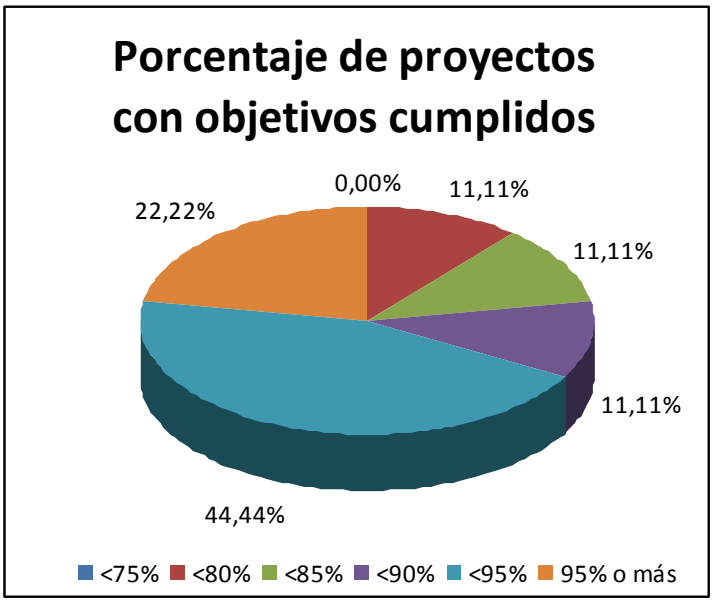


Figura 4.2.33. Porcentaje de proyectos con objetivos cumplidos

Un 66.67% cumplió el 95% de proyectos o más. Algunos al indicar el valor aclararon que entendían cumplir los objetivos en el tiempo estipulado, y que el resto de los proyectos cumplió con los objetivos del cliente pero se demoró un poco más en el desarrollo.

- ***Reacciones ante dificultades para lograr los objetivos***

Cuando surgen dificultades, y no se logran los objetivos de la iteración pactados, se deben tomar medidas. Esto es lo que se consultó en este punto, brindando cuatro opciones para elegir como posibles reacciones ante esta situación: SE TRABAJAN HORAS EXTRAS / FINALIZA IGUALMENE LA ITERACION EN FECHA PACTADA / SE EXTIENDEN LA ITERACION / OTRAS ACCIONES (especificar). Los grupos relevados podían elegir si quisieran más de una opción.

En el 77.78% de los grupos de desarrollo, trabajar horas extras es la forma de tratar de alcanzar los objetivos de la iteración (o proyecto, si lo tiene iteraciones). Para un 55.56%, también se utiliza extender la iteración o plazo de entrega (si no hay iteraciones). Un solo caso planteó otra opción que es la de agregar recursos si es que hay disponibles. Uno solo eligió la opción de finalizar la iteración en el tiempo pactado, en combinación con otras opciones según la situación que, por su forma de describir el uso de cada uno se transcribe a continuación:

- Debido al seguimiento con tickets, se detecta rápidamente las demoras. Sobre todo se hace seguimiento de tickets críticos. Si la demora es por inexperiencia del programador, este debe trabajar horas extras. Pero si es por una mala estimación se conversa con el cliente y se le explica que de cumplirse con la fecha de entrega, no funcionará de manera

completa, y se plantea la opción de correr el plazo de entrega para lograr con el alcance definido, por lo que en ese caso se extiende la iteración. En la siguiente Figura se pueden observar estos resultados de manera gráfica.

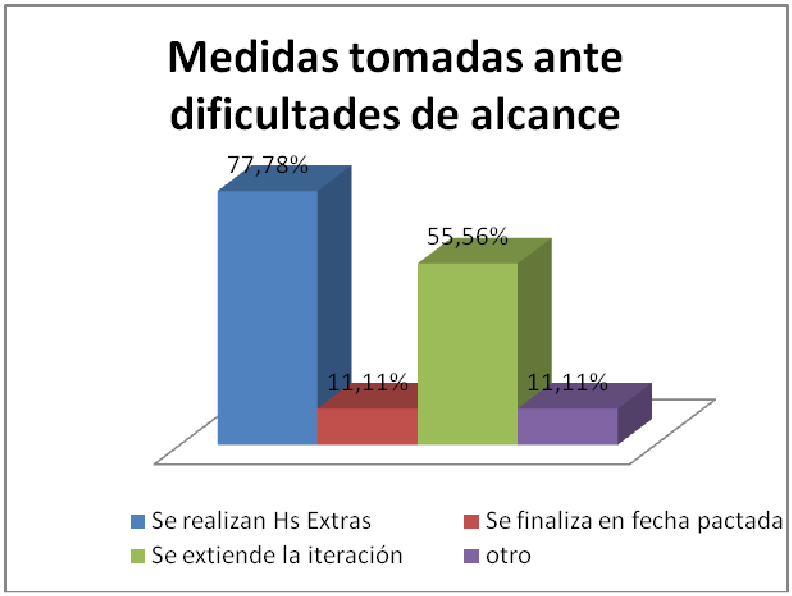


Figura 4.2.34. Medidas tomadas por los sectores ante dificultades en los alcances

- ***Nivel de satisfacción del cliente***

Se solicitó especificar el nivel de satisfacción que en general tuvieron sus clientes con los proyectos realizados frente al producto desarrollado. Se brindaron 5 opciones: COMPLETAMENTE SATISFECHO / MUY SATISFECHO / CONFORME / CON LEVES DISCONFORMIDADES / TOTALMENTE DISCONFORME. Para algunos expresar el nivel de conformidad en una sola opción no fue posible y eligieron más de una. Esto se dio en tres casos, por lo que se decidió trabajar un poco con valores ponderados para reflejar lo más fielmente posible la respuesta obtenida.

Debido a esto se consideró un peso igual a seis para indicar que todos los clientes pertenecen a este nivel de conformidad o satisfacción. Si la respuesta involucra más de una opción este peso se divide proporcionalmente, según los comentarios realizados. En caso de no tener comentario, se asigna el mismo peso a cada opción.

Tres fueron los casos analizados de esta forma:

- Eligió tres opciones aclarando que muchos están completamente satisfecha, algunos, muy conformes y unos pocos conformes. Se asignaron los valores 3, 2 y 1 respectivamente. A la hora de hacer la entrevista el propio encuestado mencionó asignar tres estrellas, dos y una a cada opción.

- Eligió dos opciones y aclaró que su elección era muy satisfecho en su mayoría y algunos estaban conformes, por lo que se asignó un valor de 5 y 1 respectivamente para reflejar la proporción.

- No brindó diferencias en cada uno por lo que se asigno un valor de 3 a cada uno.

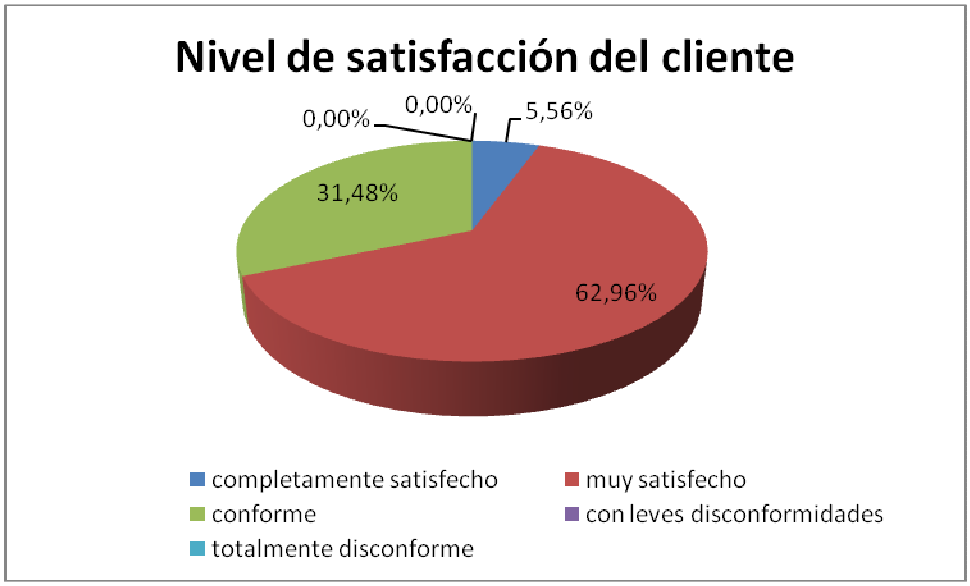


Figura 4.2.35. Niveles de satisfacción del cliente por proyecto

En ningún caso se eligieron las opciones con leves disconformidades o totalmente disconforme. Esto implica que en cada proyecto desarrollado cada grupo logró satisfacer las necesidades del cliente. Por otro, lado el porcentaje de clientes completamente conformes es muy bajo, 5.56% (Ver Figura 4.2.35). Generalmente es muy difícil poder cumplir con todas las expectativas del cliente y por eso no es sencillo lograr alcanzar este nivel de satisfacción. Pero tener un 62.96% de clientes muy conformes es un buen indicativo de que se está entendiendo bien las necesidades del cliente y se está logrando implementarlas.

4.2.3.7 Riesgos en proyectos de desarrollo siguiendo DBC

- ***Situaciones de Riesgo más comunes***

Se pidió identificar las situaciones de riesgo más comunes que ellos hubieren detectado en sus proyectos y que las numeren según su importancia, siendo 1 el riesgo más importante:

1. Imprecisiones del cliente
2. Variaciones de los requerimientos del cliente
3. Poca participación del cliente
4. Equipo de desarrollo desmotivado
5. Equipo de desarrollo involucrado en varios proyectos
6. Otras (especificar)

Si bien se solicitó ordenarlos por importancia, siendo 1 el riesgo más importante, a la hora de ebaborar la gráfica se invirtieron los valores para que a simplevista se pueda observar la barra más larga como la más importante. Para esto simplemente se consideró el valor mas grande que podría haberse asignado a un riesgo planteado y se restó el valor asignado. De esta manera se transformaron los datos, obteniendo un valor de 5 el riesgo más importante para el encuestado y 0 el ménos importante. Los puntajes medios obtenidos se muestran en la siguiente gráfica.

Figura 4.2.36. Puntaje asignado a los riesgos en los proyectos.

El riesgo al que claramente se le asignó mayor importancia fue la variación de los requerimientos de los clientes. Le sigue con cierta diferencia las imprecisiones del cliente, que podría asociarse también al riesgo anterior, dado que si los clientes no son precisos a la hora de definir que es lo que necesitan, lo más probable es que loas especificaciones de requeriientos varíen. En el orden de importancia le sigue el tener el equipo involucrado en más de un proyecto y otras causas, pero las diferencias en la valoración respecto al riesgo anterior son muy pequeñas. A continuación se encuentra el riesgo de tener un cliente con poca participación, que redunda en no tener la certeza de contar con requerimientos bien definidos. El riesgo con menor importancia es tener un equipo desmotivado. Este tuvo una valoración significativamente menor que hace notoria la diferencia respecto de todos los anteriores.

Ninguno de los grupos que mencionaron otras causas coincidieron en sus consideraciones. Se analizarán cada uno de los riesgos mencionados por los grupos de desarrollo relevados cuando aclararon su elección de la opción *otras causas*:

- Desmotivación del usuario, nunca conforme. Aquí se hace mención del usuario no del cliente, aunque a veces es la misma persona. Y lo consideró como el riesgo más importante. Esta persona eligió como nivel de satisfacción del cliente CONFORME

- Migración de datos por sus problemas asociados (cliente no entiende la inconsistencia de datos que podría haber tenido latente antes del nuevo software y que ahora es evidente). Eso tambien genera un cliente disconforme, porque tal como se aclaró en la entrevista aducen que los datos erróneos son producidos por el nuevo sistema y no reconocen que ya existían solo que no fueron detectados. Claramente esta situación genera riesgos grandes en un proyecto, y amerita dedicar tiempo a tratar de deostrar realmente cual es la situación.

- Falta de validación interna cuando todo es urgente, se deja de probar. Esta suele ser una práctica común cuando se deben cumplir fechas límites, pero llama la atención que no fue mencionado como un recurso más al momento de tomar medidas para cumplir con el alcance predefinido. Esto podría explicarse si se entiende que no es algo aconsejable, y por eso no se mencionó como una medida a tomar. Pero aquí se manifiesta que esta práctica existe, y se reconoce que puede poner en riesgo el éxito del proyecto.

- Problema del empalme de las partes reutilizadas. A veces la decisión de reutilizar algunos módulos termina necesitando demasiado re-trabajo que hubiera sido menos costoso desarrollarlo desde cero. Este riesgo está claramente explicado y no merece mayores comentarios

- El cliente no es organizado y piensa que el sistema lo va a organizar. Este riesgo hace referencia a que el cliente puede ver al sistema como una solución mágica que va a estructurarle toda la organización. Si bien es cierto que el sistema puede reorganizar ciertos procesos, facilitar otros, el cliente debe aportar su cuota.

- Planificación imprecisa. Este riesgo hace referencia a una actividad del equipo o del líder de proyecto. Dado que en todos los casos se consideró al líder involucrado directamente en la planificación del proyecto, recae en él este riesgo. Si bien la mayoría explicitó que se va ajustando la planificación en cada iteración, el responsable del equipo de desarrollo realiza la planificación al inicio y solo si hay desviaciones realiza modificaciones dentro de la iteración. Debido a esto es que se explica su mención en este punto.

4.2.3.8 Utilización de metodologías o prácticas ágiles

- ***Aplicación de metodología ágil***

Se consultó si los sectores de desarrollo de software trabajan actualmente siguiendo alguna metodología ágil, respondiendo concretamente: SI / NO. Y en caso de responder afirmativamente se solicitaba aclarar qué metodología se aplicaba. El único sector de desarrollo de software que utiliza metodologías ágiles pertenece a un organismo público y utiliza SCRUM, representando el 11.11%, ver en la siguiente Figura.

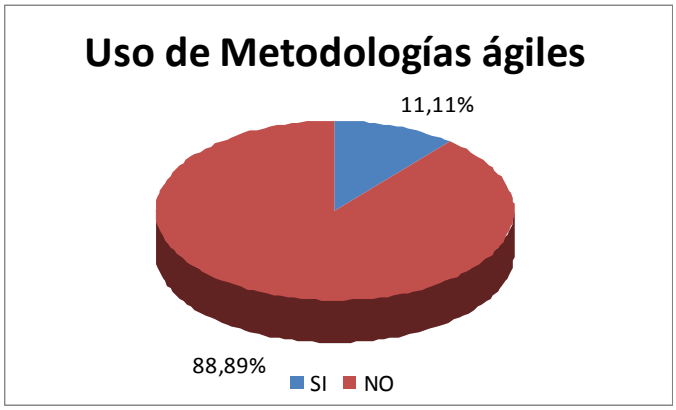


Figura 4.2.37. Sectores que utilizan metodologías ágiles actualmente.

También se consultó, en caso de no seguir actualmente una metodología ágil, si consideraban la posibilidad de utilizar alguna, contestando concretamente: SI / NO. La mitad contestó afirmativamente y la otra mitad negativamente (Ver Figura 4.2.38).

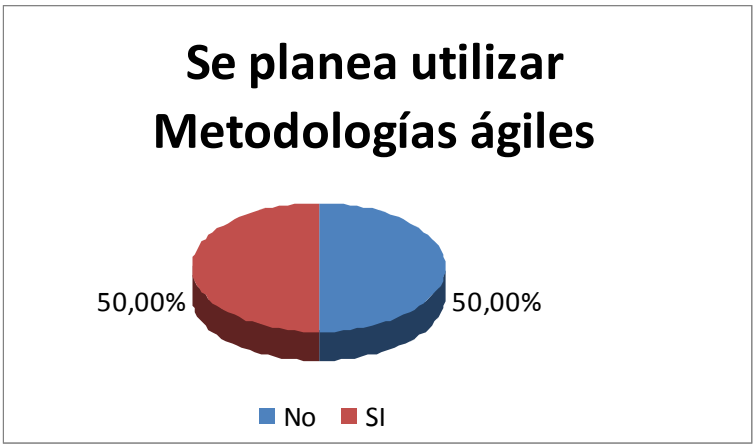


Figura 4.2.38. Sectores que consideran utilizan metodologías ágiles a futuro.

Dentro de los que contestaron negativamente, las razones aducidas fueron las siguientes:

- Por desconocimiento de casos de aplicación con éxito
- Porque se realizan prácticas ágiles pero no se sigue una metodología en particular. Se realizan reuniones entre líderes o reuniones diarias a través de skype.
- Por ser una sola persona

Dentro de los que contestaron afirmativamente, todos coinciden en utilizar Scrum, si bien una empresa contempla la posibilidad de incorporar DSDM como alternativa (Ver Figura 4.2.39).

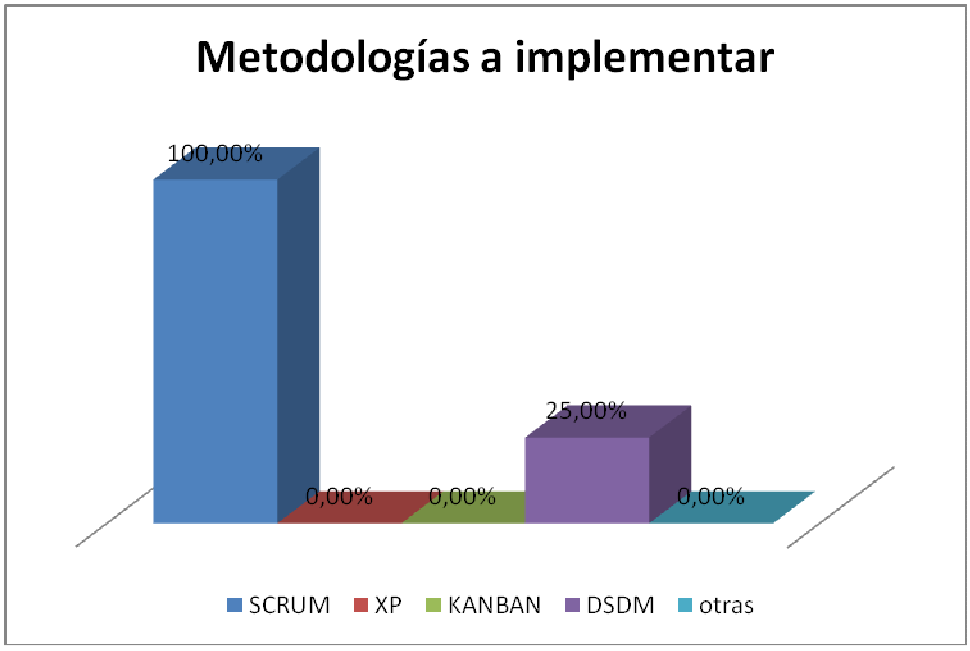


Figura 4.2.39. Metodologías que se planean utilizar a futuro.

Si se considera quienes utilizan y quienes tienen pensado incorporar las metodologías ágiles, se tiene el siguiente porcentaje expresado gráficamente en la siguiente Figura.

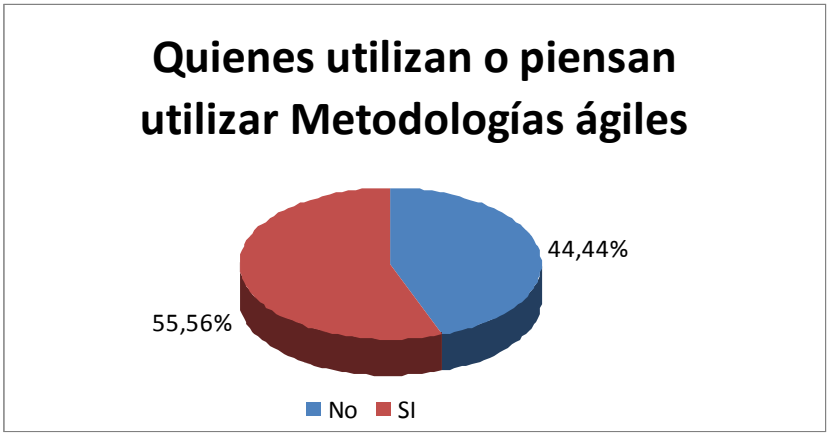


Figura 4.2.40. Sectores que utilizan o planean utilizar metodologías ágiles en el futuro.

- ***Aplicación de prácticas ágiles***

Si bien un grupo de desarrollo puede no adoptar una Metodología ágil puede estar utilizando prácticas ágiles, aún sin saber que se encuentran dentro de esta categoría. Se solicitó a cada representante del grupo de desarrollo que indique cuáles de las prácticas que se listaban eran llevadas a cabo por ellos y se les brindaba la posibilidad de indicar alguna más que no estuviera dentro del listado. El listado de prácticas ágiles presentado fue el siguiente.

- Programación de a pares
- Refactorización
- Diseño simple
- Reuniones diarias
- Pruebas automatizadas
- Iteraciones cortas
- Entregas frecuentes
- Otros

El 22.22% no indicó ninguna de las opciones por lo que se puede concluir que no sigue ninguna práctica ágil, dejando un 77.78 % que sí lo hace (Ver Figura 4.2.41).

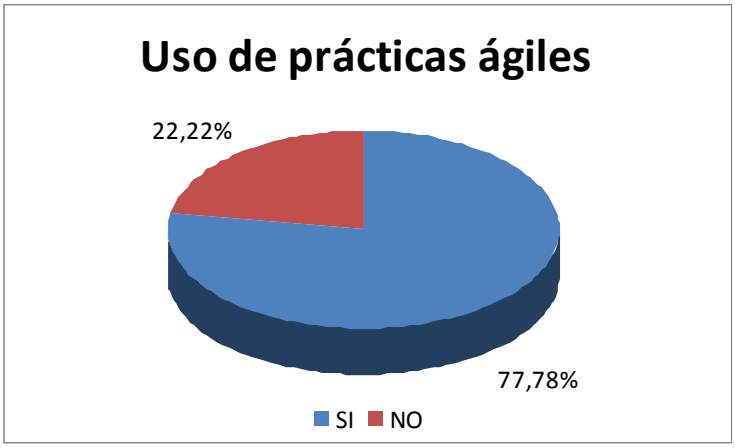


Figura 4.2.41. Sectores que utilizan prácticas ágiles.

Analizando ya las prácticas elegidas, no se dio el caso de que alguien especificara alguna práctica que no figurara en el listado. La práctica más indicada fue refactorización con 66.67% del total de los grupos de desarrollo relevados (Ver Figura 4.2.42), seguida por entregas frecuentes 55.56% y luego con un mismo porcentaje (44.44%) diseño simple y reuniones diarias. Ninguno realiza pruebas automatizadas.

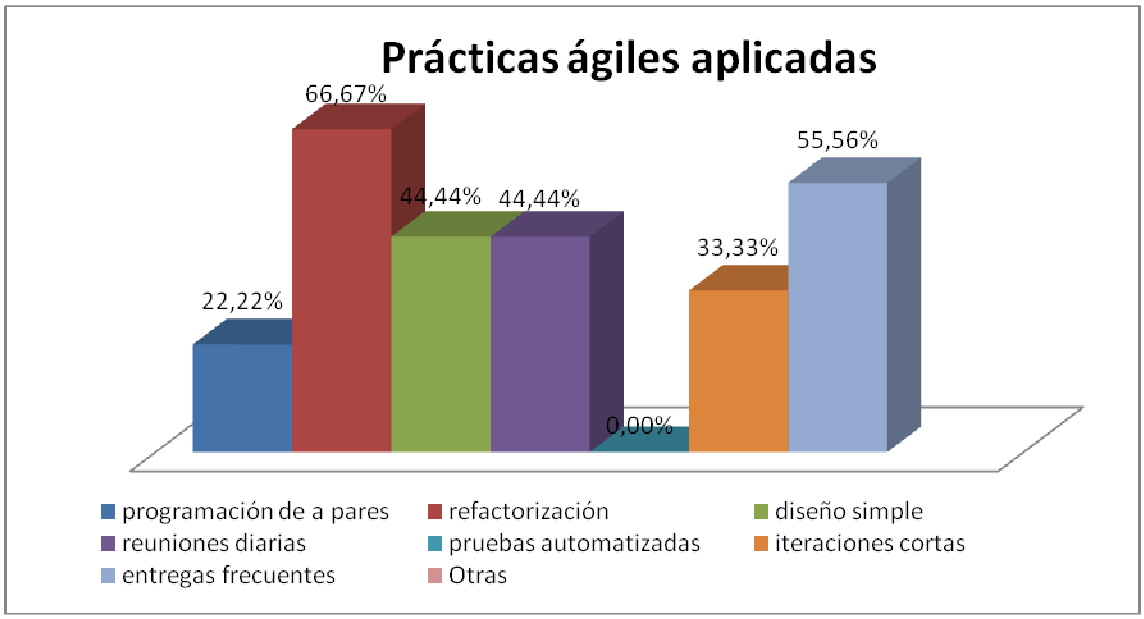


Figura 4.2.42. Prácticas ágiles utilizadas.

- ***Utilización de herramientas para realizar prácticas ágiles***

Se solicitó a cada representante del grupo de desarrollo que indique las herramientas utilizadas para poder llevar a cabo las prácticas ágiles, indicadas en el punto anterior. Se dio libertad para que cada uno indique aquellas que realmente utiliza y por lo tanto no se presentó un listado de opciones para elegir. Se obtuvieron mención de diferentes herramientas que pueden agruparse en las siguientes categorías:

- Herramienta para definir mapas conceptuales (Free Mind). Lo utilizan para modelar los requerimientos y es entendible por el cliente, quien fácilmente puede agregar “burbujas” a los gráficos.

- Herramienta que ayuda a interactuar con el cliente para que este pueda realizar solicitudes de modificación al sistema o informar errores (Sistema de tickets).

- Herramientas que ayudan a realizar teleconferencias (Team viewer / Skype). Asisten en reuniones virtuales del equipo o reuniones con el cliente.

- Las pizarras, donde Kanban board es un tipo especial de pizarra. Las Kanban board son pizarras para visualizar el proyecto, permiten ver el progreso del trabajo, asignación de tareas, que es lo que necesita realizarse luego, etc.

Los porcentajes obtenidos pueden observarse en la siguiente gráfica.

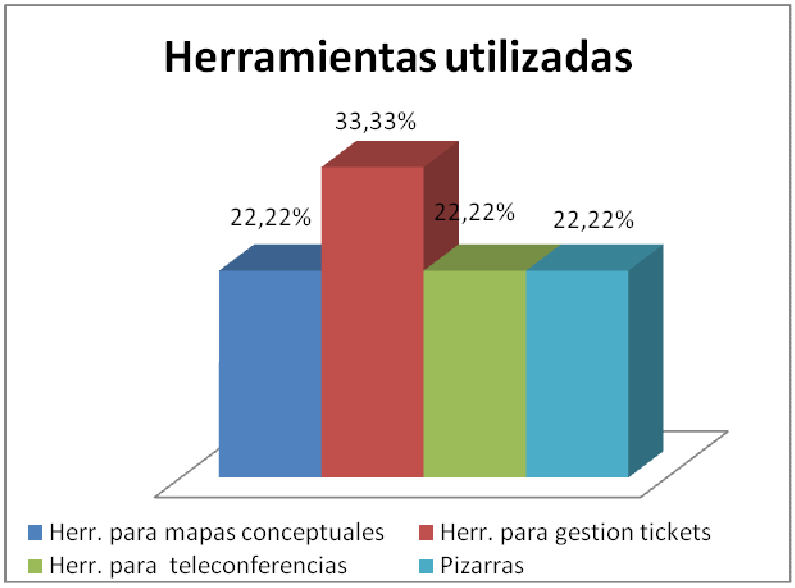


Figura 4.2.43. Herramientas utilizadas para facilitar prácticas ágiles.

Los dos primeros grupos de herramientas no asisten directamente a alguna práctica ágil listada en el punto anterior. Si bien los mapas conceptuales pueden servir como instrumento para definir los requerimientos con el cliente; también podría pensarse que ayudan a mantener un diseño simple, pero quienes utilizan esta herramienta no marcaron como una práctica seguida por el equipo al diseño simple. Por otro lado, los sistemas de tickets permiten al cliente hacer solicitudes de servicio al grupo de desarrollo que podrían ayudar a transmitir y gestionar cambios en los requerimientos.

En cambio, las herramientas para facilitar las teleconferencias ayudan a poder realizar reuniones diarias entre personas que no comparten el mismo espacio físico al mismo tiempo. Y las pizarras ayudan a visualizar el seguimiento del proyecto, compartiendo información con todo el equipo y es un radiador de información muy utilizado en metodologías ágiles, como por ejemplo Crystal, Kanban, Scrum. Estas pizarras son muy útiles también en las reuniones diarias, para tener en claro el estado actual del proyecto. Pero ninguna de los dos grupos de herramientas facilita las prácticas ágiles de: Programación de a pares, Refactorización, Iteraciones cortas, Diseño simple y Entregas frecuentes que según los grupos relevados, practican en el desarrollo de software. Es interesante, entonces, considerar que el 75% que dijo realizar refactorización, no utilizan ninguna herramienta, corriendo el riesgo de propagar errores al realizar esa acción.

4.2.3.9 Utilización de Bases del Conocimiento, y herramientas relacionadas

- ***Razones de adoptar Desarrollo Basado en Conocimiento con Genexus***

Se consideró importante preguntar los motivos por los que utiliza Desarrollo basado en Conocimiento, utilizando Genexus. Se permitió a cada uno responder ampliamente y se obtuvo una variedad de justificaciones.

Una de las justificaciones es simplemente porque fue comprado por el organismo, y por eso se utiliza. Otros mencionan haber realizado comparaciones con otras herramientas y haber optado por Genexus y otra menciona haber trabajado con otra herramienta y haber optado por Genexus debido a los costos. Dos mencionan que se encuentra en castellano. Se mencionan también varias características de Genexus como herramienta:

- Se encuentra en castellano.
- Es intuitivo.
- Tiene un ciclo de aprendizaje corto.
- Existe sinergia (retroalimentación). Los usuarios piden nuevas funcionalidades a Genexus (un requerimiento) y los desarrolladores de Genexus lo incorporan en un nuevo upgrade. Vocación orientada al desarrollo.
- Se utilizaba Progress que es similar pero tiene un costo mayor.
- Se encuentra muy difundido. Existe una comunidad y soporte.
- Hay evolución con la tecnología (comenzaron desarrollando para AS400, pasaron por C++, ahora móviles). Genexus se encarga de la base.
- Se cuenta con todos los ambientes y dispositivos contemplados en una sola herramienta (al igual que Progress) por lo que no es necesario contar con especialistas. Permite trabajar en diferentes plataformas y base de datos lo cual permite elegir la mejor solución tecnológica para el proyecto a encarar
- Es muy potente para el desarrollo de aplicaciones y brinda buenos resultados rápidos.
- Permite la generación de código.

Pero, analizando las características más destacables respecto del proceso de desarrollo, que es lo que se pretende considerar principalmente en este informe, se puede agrupar y mencionar los siguientes puntos:

- Requiere mínimo personal, debido a que permite generar código se ahora ese tiempo de codificación.
- Permite reutilización para desarrollos de sistemas a medida. Fácilmente se pueden tomar objetos para reutilizar.
- Permite desarrollo más rápido y dinámico, así es posible responder a las necesidades del cliente rápidamente y obtener en poco tiempo una funcionalidad que de otra forma varios meses.
- En poco tiempo permite evolucionar el prototipo. Esto a su vez permite realizar las interacciones con el cliente con mayor frecuencia. Este aspecto es lo que la gente de Genexus considera como ágil, no teniendo en mente una metodología ágil.

- Brinda facilidad de mantenimiento del sistema, debido a que se realizan las modificaciones sobre la base de datos del conocimiento y se genera nuevamente el código.

- ***Uso de otras herramientas complementarias de Genexus***

Dentro de Artech, empresa que comercializa Genexus se ofrecen otras herramientas complementarias: GXServer, GXFlow y GXQuery.

- GXServer "*es una herramienta que permite coordinar el trabajo de equipos para el desarrollo distribuido de aplicaciones GeneXus. Permite integrar distintos grupos de programadores en una o varias locaciones, distribuir las diferentes tareas, compartir bases de conocimiento y respaldar la información a distancia para llevar a buen término proyectos complejos"*. [sitioGenexus]

- GXFlow "*es una herramienta de workflow integrada a GeneXus que permite modelar, automatizar, administrar y optimizar los procesos de negocios de una empresa para la creación de aplicaciones críticas en forma simple y eficaz*". [sitioGenexus]

- GXQuery "*es un generador de informes orientado a los clientes que necesitan sustentar sus procesos de toma de decisiones en forma eficaz y económica."* [sitioGenexus]

Los resultados obtenidos respecto al uso de herramientas fueron los siguientes (Ver Figura 4.2.44):

- GXServer: El 66.67% considera la posibilidad de utilizar esta herramienta. De los que respondieron negativamente, uno ya lo ha utilizado y considera excesivo los costos se su licencia. Otro, descartan utilizarlo por lo costosa pero sobre todo por la forma de trabajo no lo consideran apropiado ya que trabajan mucho en el cliente.

- GXFlow: Uno solo de los grupos utiliza GFflow actualmente y representa un 11.11%, hay un 55.56% que tiene pensado incorporarlo. Se puede considerar que un 66.67% considera apropiado incorporarlo en su forma de trabajo. El 33.33% restante no lo utiliza, ni va a utilizarlo a futuro. Dentro de este grupo, uno de ellos indicó que ha trabajdo con WorkFlow para algunos clientes pero no volverán a utilizarlo, sin aducir más razones

- GXQuery: Solo el 44.44% piensa utilizarlo. Uno de ellos aclaró que sólo lo utilizará para un cliente específico. Ahora para otro están por considerar GXExplorer.

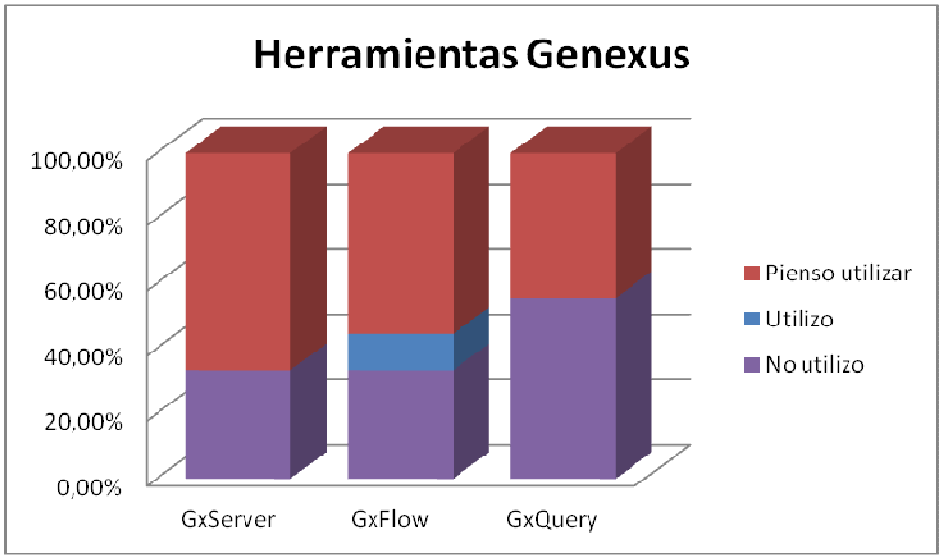


Figura 4.2.44. Opiniones sobre uso de herramientas complementarias a Genexus.

Existen además otras herramientas sobre las que no se consultó:

- GXTest que permite automatizar las pruebas funcionales y encontrar bugs en etapas tempranas. "*Fácil de configurar, utilizar y mantener. Permite correr las pruebas que son necesarias para asegurar la calidad del software generado, encontrando los errores cuando es más económico corregirlos. Creado sobre el modelo GeneXus. Los resultados se basan en conceptos de GeneXus (objetos, controles, etc.), facilitando la corrección de errores en su origen. Además, GXtest se adapta fácilmente a la evolución de la aplicación*" [sitioGenexus].

- GeneXus Business Process Modeler (BPM) "*permite modelar, optimizar y exportar los procesos de forma rápida para potenciar el desempeño de las empresas"* [Genexus web].

- GXplorer que es un "*Modelo de datos específico para el área de la salud. Abarca áreas administrativas, técnicas y clínicas. Satisface con un amplio margen todos los requerimientos propios, externos, y oficiales (SINADI-MSP). Se contempla la inclusión de datos externos para comparar. Esta totalmente integrado al producto de gestión para instituciones de salud MAGIK V4.0. Se puede anexar como módulo independiente a cualquier sistema de gestión".* [WikiGenexus web]

Comentarios adicionales.

El último punto de la encuesta era una oportunidad para que el encuestado pueda aportar cualquier dato que considerara relevante y que no hubiera sido contemplado en la encuesta, los detalles pueden consultarse en cada encuesta adjuntada en el Anexo1. La mayoría de los datos aquí mencionados fue incorporándose a los ítems evaluados anteriormente. La respuesta que puede resaltarse es simplemente que al trabajar con patterns se puede tener métricas y poder estimar complejidades y costos, se puede saber dónde realizar modificaciones debido a que todo se mantiene lo mas estándar posible y se puede tener un producto más estándar.

4.2.3.10Conclusiones de sección

- Procedimiento de desarrollo de software

La mayoría de los sectores de desarrollo tienen un procedimiento para el desarrollo de software (un 77,78%). La proporción de sectores con esta definición es mayor en sectores dentro del ámbito privado que en el ámbito público. El detalle con que se describen los pasos a seguir son de distinto nivel de granularidad, siendo algunos muy generales y otros extremadamente detallados. Aquí pudo influir también haber podido realizar algunas entrevistas, donde se explayaron más respecto de los que completaron la encuesta. Algunos sectores de desarrollo detallaron dos tipos de procedimientos según el tamaño del proyecto. Son más detallistas y rigurosos para los grandes y más permeables para los pequeños o con clientes ya conocidos.

Los pasos que plantean la mayoría de los procesos descriptos no reflejan las ventajas de trabajar con un desarrollo basado en conocimiento, y tampoco específicamente con Genexus. No se refleja el poder presentar rápidamente al cliente las funcionalidades solicitadas a través de un prototipo de DBC, por ejemplo.

Si se trabajara con un flujo continuo de trabajo, definiendo las tareas (HU) a realizar en un Kanban board, cada columna representaría una etapa o actividad dentro del proceso de desarrollo. Dichas etapas podrían ser adaptadas según la forma de trabajar del equipo y la organización, pudiendo agregar o eliminar etapas. Sería conveniente tener, mínimamente, una etapa de análisis, una de desarrollo, una de prueba y finalmente una de producción. Cuando una historia de usuario llegara a la columna producción, querría decir que el prototipo de DBC estaría listo para ser entregado al cliente, y que pasó inclusive la etapa de testing que debería haber incluido pruebas de aceptación en lo posible con participación del cliente. El incremento o prototipo de DBC se encontraría completamente operativo en su ambiente real y podría ser explotado por los usuarios.

Analizando la etapa de pruebas, menos del 40% mencionan alguna actividad de pruebas ya sean pruebas unitarias y/o de aceptación. Inclusive cuando se trabaja con un desarrollo tradicional las pruebas son muy importantes para tratar de detectar errores lo antes posible y tratar de validar si se está cumpliendo con los requerimientos especificados. Sería conveniente dejar explicitada la actividad de pruebas en el Kanban board, para que quede evidente la necesidad de probar el software y validarlo con el cliente para cada funcionalidad, independiente de la situación en que se encuentre el proyecto.

Un sector tiene registros muy formales del proceso de desarrollo, si bien ellos mismos manifiestan que han flexibilizado un poco dicho proceso. Tener procesos muy formales puede ir en contra del desarrollo ágil.

La definición del procedimiento, ya sea general o específico es responsabilidad del líder del proyecto. Solo uno de los sectores considera al equipo como co-responsable junto con el líder para definir el procedimiento de desarrollo a seguir. En metodologías ágiles es el equipo el que se auto-organiza y determina la forma de trabajar, si bien hay decisiones que toma el líder, Scrum master o cualquiera sea el nombre del rol que ejerza la supervisión. Se debería tener reuniones de reflexión donde el equipo pueda opinar y proponer modificaciones a la forma en que se encuentran trabajando para que el procedimiento se adapte al equipo.

Para el diseño dentro del proceso de desarrollo casi el 56% utiliza algún diagrama UML, pero solo utilizan Casos de Uso y Diagramas de Clases. Es lógica la utilización de casos de uso para poder especificar las necesidades del proyecto y los diagramas de clases, para poder trabajar directamente definiendo luego en Genexus la Base de Datos del Conocimiento. Pero, ninguno hizo mención del resto de los diagramas, ni siquiera para situaciones excepcionales. Otros, utilizan para el diseño los mapas conceptuales, diagrama de entidad relación, diccionario de datos, wiki con estructura de productos ya desarrollados. Uno define sus diseños con “técnicas ágiles”, pero no se aclara nada al respecto. La pregunta en realidad solo buscaba obtener información de los diagramas UML que se utilizan para diseñar, y la respuesta obtenida en varios no se alineó en tal sentido.

El 100% realiza un desarrollo con reutilización, variando sus porcentajes. La selección de los artefactos a reutilizar, es una tarea no muy simple y para uno de los encuestados puede ser un

gran riesgo dentro del proyecto si se eligen erróneamente debido a todo el trabajo de adaptación que se les debe realizar. Trabajar con patrones ayuda a estandarizar el trabajo y saber qué modificar. En el 66.67% de los sectores de desarrollo es el líder del proyecto el encargado de buscar y definir los artefactos a reutilizar. El resto no especifica quien realiza la tarea, pudiendo ser el líder o tal vez el equipo o un trabajo en conjunto. Para realizar la búsqueda, solo el 22,22% menciona el uso de un catálogo o librería donde tiene organizados los artefactos o módulos de proyectos anteriores que pueden reutilizarse y en donde debe realizarse la búsqueda. Un grupo explícitamente declara no utilizar ningún catálogo para realizar esta selección.

Respecto al tema de diseño, se podría considerar un rol de diseñador-líder quien sea responsable del diseño mayor del sistema y sea él quien determine los artefactos a reutilizar. No sería necesario establecer herramientas específicas para realizar los diseños, cada equipo utilizará los que crea convenientes en cada situación.

- La participación del cliente

Se puede observar que todos los sectores relevados, de alguna manera, buscan tener contacto con el cliente. Esto muestra la importancia que tiene para cada equipo de desarrollo esta participación. Todos afirman tener colaboración por parte del cliente pero se deduce claramente que no todos tienen la misma idea de colaboración del cliente en el proceso de desarrollo. Para algunos es trabajar con ellos para probar el sistema, para otro es brindarles información del avance del proyecto y para otros la colaboración del cliente en el proyecto es poder realizarles consultas cuando sea necesario, es decir tener una comunicación directa con él en diferentes momentos.

La comunicación es la primera forma de participación del cliente. Los instrumentos usados para lograr la comunicación, varían. El diálogo y la comunicación, es lo que podría encontrarse en común en la mayoría de las respuestas. En un solo sector de desarrollo se busca que cada miembro del equipo interactúe y/o visite al cliente. Para un sector la comunicación es simplemente realizar presentaciones de informes de avance, documentos donde se describe lo que están realizando más que software funcionando.

Solo uno menciona como instrumento para lograr la participación del cliente en el proceso de desarrollo, la presentación de prototipos de manera rápida a éste para corroborar los requerimientos implementados. La generación de prototipos es una característica muy destacada en Genexus, pero por lo que se analiza no es muy utilizada en los equipos para lograr que el cliente participe a lo largo del desarrollo. Tal vez realizan presentaciones de prototipos pero no consideran a esta actividad una forma de lograr la participación del cliente.

La mayoría manifiesta realizar entregas frecuentes al cliente, menores a 2 meses, pero solo un 25% tienen plazos periódicos de entregas definidos. El resto define los períodos de entregas según el producto a desarrollar. Inclusive a veces varía la frecuencia entre entregas según la etapa de desarrollo. Uno solo hace mención que buscan entregar pequeñas funcionalidades que vayan despertando el interés de los usuarios. Realmente las entregas frecuentes permiten tener una retroalimentación directa del cliente y/o usuario a lo largo del proyecto que permiten corroborar lo que se está realizando y lo que se planea realizar a futuro. También las entregas frecuentes, si presentan valor agregado para el cliente lo van a motivar para involucrarse más en proceso de desarrollo, colaborar más activamente en las pruebas del sistema y en utilización de aquellas porciones de producto que ya se encuentren en producción.

Sería de gran utilidad tener definidas ciertas reuniones con el cliente, para poder definir requerimientos, ajustes a requerimientos, presentar los prototipos de DBC y obtener retroalimentación frecuentemente. Como en un proyecto suelen haber varios stakeholders involucrados debería definirse un representante para participar en las reuniones ya que se volvería insostenible que todos asistan a todas las reuniones y en caso de asistir, llegar a un consenso entre todos llevaría mucho tiempo. Solo para situaciones específicas al inicio y finalización del proyecto sería necesaria la participación de todos.

- Los requerimientos en los proyectos desarrollados

En todas las áreas de desarrollo relevadas, se definen los requerimientos al inicio del proyecto a través de entrevistas, reuniones, encuestas. En la mayoría de los casos, 89%, se realiza detalle completo de todos los requerimientos del cliente. En el 11% restante, no se realizan especificaciones en detalle de todo el producto software a desarrollar cuando es muy grande; primeramente realiza descripciones generales de las porciones de producto a desarrollar y únicamente se realiza una especificación completa y detenida de la porción de funcionalidad que el cliente opta por desarrollar en primer lugar. Esta última forma de trabajo se asemeja un poco más a lo que son las prácticas de metodologías ágiles, pero aquí se eligen productos o porciones de productos más que funcionalidades.

Los instrumentos que fueron mencionados para definir los requerimientos son variados y con características diferentes: mapas conceptuales, historias de usuario, minutas de reuniones y documento BluePrint BBP. También algunos al describir el procedimiento de desarrollo utilizan diagramas de clase y diagramas de entidad relación para definirlos. Un 22% comparte el instrumento realizado con el cliente para que sea el cliente mismo quien modifique lo que considere necesario, antes de que queden establecidos. Algunos son muy formales, mientras que otros son más que nada diagramas simples de entender y mantener actualizados.

Uno solo, correspondiente al 11.11%, hizo énfasis que al definir los requerimientos considera además los puntos de verificación y añade luego algunos detalles específicos de tablas / objetos / estructuras para entregarle al equipo. Este sector es el mismo que realiza una definición general de todos los requerimientos y solo una definición detallada de la primera porción de producto a desarrollar. El líder al darle al equipo datos específicos adjuntos a los requerimientos, respecto a tablas / objetos / estructuras, en cierto modo está conduciendo el diseño del mismo y deja solo las tareas de codificación y prueba al equipo.

Para todos los sectores relevados, los requerimientos cambian dentro de un proyecto. No existió ningún sector de desarrollo de organismo público o empresa privada que considere que los requerimientos no cambian o cambian muy pocas veces. Para el 78% éstos siempre cambian mientras que para los restantes se modifican algunas veces. Esta situación amerita que el equipo reaccione ante estas situaciones y las prácticas ágiles pueden facilitar la tarea.

Para definir los requerimientos sería recomendable trabajar con historias de usuario y que se definan en una reunión de planificación. Pero para que el equipo entienda realmente el objetivo y meta principal de proyecto sería oportuno tener antes una primera reunión de iniciación del proyecto, exploración, donde participe todo el equipo, el cliente y demás stakehoders. La finalidad de esta reunión sería lograr definir y compartir una visión común del proyecto. Por otro lado, debido a que a que los requerimientos varían, se deberían realizar reuniones de ajustes de requerimientos donde el cliente tenga la posibilidad de agregar o quitar funcionalidades o bien cambiar sus prioridades. Otra reunión que podría ser útil es una reunión de contextualización para que en esa reunión se le explique al equipo cada una de las funcionalidades incorporadas a la lista de trabajo, o por lo menos las de mayor prioridad. De esta manera se evitarían malas interpretaciones de las HU.

- La planificación y seguimiento del proyecto

Más de la mitad de las empresas, el 66.67% de las empresas desarrolla de manera iterativa. Solo el 22.22% trabaja con iteraciones fijas, el 45% trabaja con iteraciones cuya duración es fijada por los productos que deben realizar, y son por lo tanto duraciones variables. Pero el trabajar con iteraciones variables, trae sus inconvenientes porque no permite realmente tomar muchas métricas que pueden ser importantes en el proceso de desarrollo y para la toma de decisiones. El 33% ni siquiera trabaja con iteraciones, uno de los grupos de desarrollo que no trabaja con iteraciones aclaró que trabajan entregando al cliente los productos que le son más importantes al cliente a medida que los van desarrollando. En este último caso podría verse como un flujo continuo de

desarrollo, como Kanban, donde no hay iteraciones. Se debe recordar que la duración de la iteración no es lo mismo que el espacio de tiempo entre entregas (entregas frecuentes). Se puede entregar varias veces en una iteración, o iterar varias veces y luego entregar.

La planificación del desarrollo en su amplia mayoría, el 88.89% planifica al inicio del proyecto pero a lo largo del proyecto va modificando su planificación, mientras que el 11.11% restante realiza la planificación al inicio y no la modifican. Esta acción sería impensada en la actualidad, ya que planificar al inicio y pretender mantener lo planificado durante todo el proyecto es algo prácticamente utópico. Por eso, es necesario aclarar que quien eligió esta opción pertenece al grupo de desarrollo donde se realiza una planificación muy detallada al inicio solo de la parte que va a desarrollar primero y deja otras etapas definidas de forma genérica, para detallarlas justo antes reiniciar su desarrollo. Entonces podría asemejarse a definir al inicio y luego ir modificando a medida que se avanza sobre otra iteración. Pero, el entrevistado aclaró que en caso que el cliente requiera un análisis de todo el proyecto de manera detallada, la empresa lo realiza pero cobrando un adicional por este trabajo y especificando con gran nivel de detalle lo que se debe realizar en cada fase/parte o etapa del desarrollo. De esta manera considera posible poder hacer una planificación muy detallada junto con una definición de requerimientos muy detallada y se vuelve entonces a esa situación utópica, pero que en caso de haber modificaciones en requerimientos “no se podría modificar”.

Las herramientas utilizadas por los grupos de desarrollo para el seguimiento de un proyecto ayudan a tener una visión más completa de la forma de trabajo. El 77.78% de los sectores relevados utiliza alguna de estas herramientas. Las herramientas mencionadas son: dot Project, Microsfot Project, Aplicativo para Gestión de tareas, Sistema de tickets y Kanban board. Se mencionó también Google desktop pero esta herramienta tiene otros fines, no el seguimiento del proyecto. Las cinco herramientas permiten realizar la asignación y control del proyecto. De todas estas herramientas Kanban board es una herramienta utilizada en las metodologías ágiles.

Ninguno hizo mención al uso de burn down chart cuando se pidió herramienta de seguimiento del proyecto. Pero, ante la pregunta específica y concreta, el 22.22% sí lo utiliza, por lo que surge la pregunta, si realmente se utiliza o no. Uno de los sectores lo utiliza junto con el sistema de tickets y el otro con kanban board solo para proyectos grandes.

Dentro de la planificación determinar los alcances y los costos son una actividad muy crítica e importante. Algunos manejan distintos momentos en el desarrollo para definir costos y alcances. Se detectaron tres momentos para definir los alcances, al inicio, para cada iteración o hacer una definición inicial y refinar en cada iteración.

Respecto a los costos, se definen al inicio, para cada iteración o bien no se manejan costos. Todos los que no manejan costos pertenecen a organismos públicos. Los que consideraron los costos al inicio, comentaron de diferentes formas que, en caso de variar los alcances, tratan de renegociar los costos pero esto es sumamente difícil con el cliente y deben mantener los costos definidos, si bien pueden modificar algunos plazos de entrega. Con una metodología ágil, esto podría volverse más fácil, ya que el cliente forma parte del equipo y los costos y alcances se van definiendo al iniciar las iteraciones.

Un solo grupo de desarrollo plantea la posibilidad de definir en forma general todo el proyecto y solo en detalle una primera etapa, tanto en alcances como en costos. Esto les permite focalizarse en una parte del producto a desarrollar, definirlo con la mayor granularidad posible y establecer un precio proporcional a ese trabajo. Este grupo brinda la posibilidad al cliente de ofrecerle un análisis detallado del sistema global pero cobrándoles un costo adicional y ningún cliente se los solicitó hasta el momento.

Una consideración especial tiene que todos los grupos de desarrollo que pertenecen a organismos públicos, quienes no definen los costos, es que todos aducen que ya existen sueldos para el personal y esos son los costos. Pero, esto hace pensar que tal vez no se realiza un análisis costo beneficio del proyecto de desarrollo a implementar, la sola justificación que existe un monto del

organismo destinado a pagar sueldos, no justifica tal vez la realización de un proyecto. Se debería iniciar el desarrollo de acuerdo a un análisis costo-beneficio, indicando el tiempo de recupero de la inversión aunque sea del estado. Así también se debería considerar la innovación y cambio continuo del mercado.

Para realizar una planificación que pueda ir modificándose, deberían considerarse mínimamente una reunión de planificación inicial, donde se definirían los requerimientos y se priorizarían (utilizando HU) y reuniones de ajustes donde se pueda modificar las funcionalidades requeridas o sus prioridades. Ambas reuniones deberían contar con la participación del cliente. Por otro lado sería importante la monitorización del trabajo diario del equipo con reuniones de seguimiento para detectar lo antes posible inconvenientes y visualizar el grado de avance.

- Los resultados obtenidos en los proyectos realizados

El porcentaje promedio de proyectos donde se cumplieron los objetivos pactados con el cliente es alto, un 88.11%. Como algunos de los sectores relevados aclararon que entendían cumplir los objetivos en el tiempo estipulado, y que el resto de los proyectos cumplió con los objetivos del cliente pero se demoró un poco más en el desarrollo podría ser más alto todavía. Un 78% de los sectores relevados afirman haber cumplido en más del 90% de los proyectos con los objetivos del cliente. Estos números harían pensar que no es necesario cambiar la forma de trabajo dado que el porcentaje de éxito es muy alto, pero si se analizan otros datos, como situaciones de riesgo por ejemplo, que se analizará más adelante, no es tan cierto.

Expresar en una sola opción el nivel de satisfacción que en general tuvieron los clientes con los proyectos realizados frente al producto desarrollado, no es una pregunta sencilla. Existen seguramente en el haber de cada área de desarrollo de software diferentes experiencias y resultados. Pero se buscaba obtener la impresión que ellos mismos tienen, en términos generales, de la satisfacción del cliente frente al producto desarrollado. En base a las respuestas obtenidas se desprende que en cada proyecto desarrollado cada grupo logró entender al cliente y satisfacer sus necesidades. O por lo menos, la idea en general del nivel de satisfacción del cliente es bastante alta en todos los casos. Si bien el porcentaje de clientes completamente conformes es muy bajo, 5.56%, es entendible dado que es muy difícil poder cumplir con todas las expectativas del cliente. Pero tener un 61,11% de clientes muy conformes es un buen indicativo de que se está entendiendo bien las necesidades del cliente y se está logrando implementarlas. Este valor también genera la inquietud de si es necesario o no modificar el proceso de desarrollo si se alcanzan tan buenos niveles de satisfacción.

Si por dificultades se ve en riesgo cumplir con los alcances pactados, el 77.78% de los grupos de desarrollo, trabajar horas extras. Es decir que, trabajar horas extras es la medida más común para corregir desviaciones en un proyecto, pero esto como se afirma Beck [Beck 2000], desvaloriza y desmotiva al equipo y, a su vez, éste suele producir código de menor calidad. Por lo tanto, trabajar horas extras no es algo conveniente y no debería hacerse tan frecuentemente. Para un 55.56%, también se utiliza extender la iteración o plazo de entrega (si no hay iteraciones), generalmente asume aquí los costos el sector de desarrollo dado que el cliente difícilmente acceda a aumentar los valores pactados, por eso se entiende que sea tan. Un solo caso planteó otra opción que es la de agregar recursos si es que hay disponibles. Uno solo considera la opción de finalizar la iteración en el tiempo pactado, si el cliente insiste en finalizar en esa fecha.

El riesgo al que claramente se le asignó mayor importancia fue la variación de los requerimientos de los clientes. Es muy importante para el éxito del proyecto tener en claro que es lo que el cliente necesita. Cuando estos requerimientos cambian debe ajustarse el proyecto para poder adaptarse a los cambios, y si no se realiza correctamente puede no satisfacerse las expectativas del cliente. Las imprecisiones del cliente le siguen en importancia, que podría asociarse también al riesgo anterior, dado que si los clientes no son precisos a la hora de definir que es lo que necesitan, lo más probable es que las especificaciones de requerimientos varíen. Ambos riesgos pueden minimizarse o reducirse aplicando alguna práctica ágil, y sobre todo considerándo al cliente como parte del equipo de desarrollo, buscando satisfacer más las necesidades del cliente que un

contrato preestablecido, tal como se postula en el Manifiesto ágil. Siguiendo el orden de importancia se encuentra el tener el equipo involucrado en más de un proyecto. A continuación se encuetra el riesgo de tener un cliente con poca participación, que redunda en no téner la certeza de tener los requerimientos bien definidos. El riesgo con menor importancia es tener un equipo desmotivado. Este tuvo una valoración significativamente menor que hace notoria la diferencia respecto de todos los anteriores. Pero, si se considera que un equipo para que funcione correctamente y desarrolle productos de calidad debe estár motivado, pondría en riesgo la calidad de lo desarrollado. En metodologías ágiles se busca mantener un producto de calidad, no solo que satisfaga los requerimiento dado que esto también facilita las posibles modificaiónes posteriores que el sistema pueda sufrir para adaptarse a cambiso de requerimientos. Por lo tanto, sería importante poder tener una reunión con el cliente de reflexión y validación que permita saber al equipo si el cliente está conforme con lo que se le va entregando y tener una reunión de reflexión interna que permita al equipo expresarse libremente y proponer mejoras a la forma de trabajo para crear un ambiente más confortable para el trabajo.

Dentro de otros posibles riesgos que fueron mencionados por los grupos de desarrollo relevados merece la pena desacar la falta de validación interna cuando todo es urgente dado que se deja de probar. Esta suele ser una práctica común cuando se deben cumplir fechas límites, pero llama la atención que no fue mencionado como un recurso más al momento de tomar medidas para cumplir con el alcance predefinido. Esto podría explicarse si se entiende que no es algo aconsejable, y por eso no se mencionó como una medida a tomar. Pero aquí se manifiesta que esta práctica existe, y se reconoce que puede poner en riesgo el éxito del proyecto. Otro riesgo ya mencionado es tomar una mala decisión de qué objetos reutilizar dado que pueden llegar a necesitar demasiado re-trabajo que hubiera sido menos costoso desarrollarlo desde cero. Este riesgo está claramente explicado y no merece mayores comentarios. Por último se comentará el riesgo de tener una planificación imprecisa, dado que en todos los casos se consideró al lider involucrado directamente en la planificación del proyecto, recae en él este riesgo y justamente quien hizo mención a este riesgo realiza la planificación al inicio y solo, si hay desviaciones, realiza modificaciones dentro de la iteración. Debido a esto es que se entiende su mención en este punto.

- La utilización de metodologías o prácticas ágiles

Una sola de las áreas de desarrollo relevadas utiliza una metodología ágil, Scrum y representa el 11.11% de la muestra. Pero existe otro porcentaje que está pensando en utilizarlas. Es decir que poco más de la mitad (55.56%) usa o piensa utiliza una metodología ágil.

Si bien un grupo de desarrollo puede no adoptar una Metodología ágil puede estar utilizando prácticas ágiles, en este caso el 88.89% reconoce utilizar alguna de las prácticas ágiles que se le presentaron. La práctica más indicada fue refactorización con 66.67%, seguida por entregas frecuentes 55.56% y luego con un mismo porcentaje (44.44%) diseño simple y reuniones diarias. Ninguno realiza pruebas automatizadas

Se mencionaron varias herramientas para ayudar a realizar la práctica ágil de reuniones diarias y otras herramientas como radiadores de información. Pero no se mencionan herramientas que faciliten las prácticas ágiles de: Programación de a pares, Refactorización, Iteraciones cortas, Diseño simple y Entregas frecuentes que según los grupos relevados, practican en el desarrollo de software. Por lo menos es interesante, entonces, considerar que el 75% del total que dijo realizar refactorización, no utilizan ninguna herramienta, corriendo el riesgo de propagar errores al realizar esa acción, sumado a que no tiene pruebas automatizadas

Por lo tanto no es incongruente lo que se plantea en este trabajo de tesis que es sugerirles por lo menos una serie de prácticas ágiles. De esta forma el acercamiento a las metodologías ágiles es progresivo y pueden ir incorporándose las prácticas en forma individual y según necesidades y características del equipo.

- Uso de Desarrollo basado en conocimiento mediante Genexus

Usar DBC, tiene su razón de ser. Dentro de las razones de uso de DBC relevadas, se pueden mencionar los siguientes puntos: se puede realizar un desarrollo rápido y dinámico permitiendo evolucionar el prototipo, se puede trabajar con poco personal, se puede realizar reutilización. Todo esto a su vez permite mayor interacción con el cliente, por la rapidez para generar prototipos y mayor facilidad de mantenimiento del sistema, debido a que se realizan las modificaciones sobre la base de datos del conocimiento y se genera nuevamente el código. Si bien alguien mencionó la agilidad como una característica de Genexus, no debe considerarse desde el punto de vista de los que se conoce dentro de las metodologías ágiles, sino como sinónimo de dinámico.

Existen herramientas complementarias de Genexus. Solo un sector de desarrollo utiliza una de estas herramientas y es GXflow. Pero para cada una de las herramientas existe por lo menos un 50% de los encuestados, que manifiesta la posibilidad de utilizarlo a futuro. Si se trabajar con pruebas automatizadas se debería poder considerar la posibilidad de utilizar también GXTest.

4.2.4 Reflexión

Realmente, obtener la información para analizarla y realizar el estudio de campo fue mucho más complicado de lo esperado. Una actividad que se había planeado realizar en un par de semanas debió extenderse en más de tres meses de insistir amablemente para conseguir la entrevista o la encuesta con los datos. A pesar de insistir con los responsables del sector de desarrollo sobre todo de los organismos públicos, no se pudo recabar la información que se pretendía. Si bien existía un compromiso inicial de colaborar con el estudio de campo, es entendible que estos equipos de desarrollo siempre estén sobrecargados de actividades y también deben adecuarse a las necesidades de reuniones de personas del gobierno. Sin embargo se pudo obtener información con la cual realizar un “modesto” estudio de campo.

En el sector privado, el impedimento inicial a la hora de poder concertar la entrevista o el relevamiento fue que consideraban que, detrás de él, se encontraba una auditoría de la empresa que comercializa Genexus. En todos los casos, dándole las explicaciones pertinentes, accedieron a brindar la información.

Habiendo analizado los datos relevados con detenimiento, se tiene un panorama general de la situación actual de los sectores de desarrollo que utilizan DBC, sus características y su forma de trabajo. Si bien los niveles de satisfacción de los clientes son muy altos y los resultados de los proyectos muy satisfactorios, la propuesta que se busca plantear podría tratar de minimizar los riesgos que fueron identificados y que podrían conducir a no lograr los objetivos del proyecto. Se puede tratar de definir roles dentro del equipo para dar mayor libertad al equipo y sacar tantas responsabilidades al líder de proyecto, se pueden proponer diferentes reuniones para lograr la participación de todos los involucrados y se pueden proponer recursos que ordenen la forma de trabajar y realizar el seguimiento.

4.3 DESCRIPCIÓN DE PROPUESTA DE PRÁCTICAS ÁGILES PARA EL DESARROLLO BASADO EN CONOCIMIENTO

Teniendo en cuenta los lineamientos que fueron planteados a medida que se presentaron los datos relevados en la sección anterior, y habiendo además analizado la información recolectada a través de encuestas a personas que trabajan con metodologías ágiles que asistieron al primer Salta Agile Open Space, cuyo análisis y datos recolectados se encuentran en el anexo 2, se presenta a continuación la descripción detallada de la propuesta de prácticas ágiles para el desarrollo basado en conocimiento.

Si se quiere comenzar a pensar en una metodología ágil, hay que tener en mente los cuatro postulados del manifiesto ágil y buscar una aproximación hacia ellos.

> 1) *Valorar al individuo y a las interacciones del equipo de desarrollo por encima del proceso y las herramientas*.

Teniendo en cuenta la cantidad de personas que conforman los equipos internos de desarrollo, es factible aplicar una metodología ágil o en su defecto ciertas prácticas ágiles. Al no exceder las 7 personas, esto permite una comunicación e interacción fluida entre los miembros del equipo. Pero, las personas involucradas en el desarrollo no siempre comparten el horario y lugar de trabajo. De no poder organizarse los equipos internos con personas que compartan el horario, es necesario suplir esta falta, para que se pueda compartir información casi de manera directa entre todos los miembros del equipo. Inclusive horarios solapados podrían ayudar en esta situación. Se puede utilizar radiadores de información, a la vista y disponibles para quienes estén trabajando en el sector y utilizar también reuniones pactadas.

Pero este postulado, exige un cambio de concepción de las personas dentro del equipo, no como simples desarrolladores, sino como el recurso más importante para lograr el éxito del proyecto. Se debe prestar atención a las personas y se debe proveer mecanismos para mejorar las interacciones de las personas, la forma de comunicarse y transmitir su información.

En los sectores de desarrollo de software utilizando Desarrollo Basado en el Conocimiento de Salta Capital, el líder tiene un rol muy importante. Es él quien toma decisiones, define la forma de trabajo, asigna las tareas y generalmente es quien interactúa con el cliente. Si realmente un sector de desarrollo quiere comenzar a realizar un desarrollo ágil, debe cambiar esta idea y dar libertad al equipo de desarrollo. Se debe cambiar la cultura organizacional, y como se vio en el análisis de encuesta de quienes utilizan metodologías ágiles, esta es una barrera muy importante cuando se pretende comenzar a trabajar con metodologías ágiles

2) *Valorar el desarrollo de software que funcione por sobre una documentación exhaustiva*.

No se trabaja en Desarrollo Basado en Conocimiento en Salta Capital siguiendo una metodología tradicional. No se realiza una documentación exhaustiva, los requerimientos solo en algunos casos se realizan con gran nivel de detalle. Por otro lado no se ve un gran aprovechamiento del uso de prototipos de DBC frente al cliente, pudiendo ser ésta una virtud de este tipo de desarrollos permitiendo de esta forma corroborar los requerimientos implementados. La adopción de metodologías ágiles requeriría que esta situación se modifique.

3) *Valorar la colaboración con el cliente por sobre la negociación contractual*.

Si bien todos afirman tener colaboración por parte del cliente, se pudo determinar que el significado de colaboración no es el mismo para todos. No es solo permitirles probar el sistema, o brindarles información del avance del proyecto o poder realizarles consultas cuando sea necesario. En el desarrollo ágil el cliente se debe integrar y debe colaborar con el equipo de trabajo. Es otro cambio cultural importante para que cliente y equipo realicen la toma de decisiones conjuntas. Es muy importante poder mantener una comunicación rápida, y conexiones de interacción entre todos los individuos.

4) *Valorar la respuesta al cambio por sobre el seguimiento de un plan*.

En los equipos de desarrollo basados en conocimiento de Salta Capital, la planificación del desarrollo se define al inicio, pero se va modificando a lo largo del proyecto, en la gran mayoría de los casos. Esto permite ajustar la planificación según los cambios que vayan surgiendo en el proyecto. Pero, los costos se definen al inicio, y difícilmente cambian en los sectores privados, pudiendo extenderse los plazos en algunas circunstancias. Esto juega en contra de responder al cambio ya que si hay modificaciones que impliquen variaciones en el alcance, los costos corren sólo para el sector de desarrollo y no para el cliente, por lo que la reacción natural no es dar la bienvenida a los cambios como enuncian las metodologías ágiles. Aquí también debe realizarse un cambio cultural y lograr que el

cliente forme parte del equipo, de tal modo que los costos y alcances se vayan definiendo al iniciar las iteraciones. Construir un plan es útil y cada metodología ágil contiene actividades de planificación específicas pero también tienen mecanismos para manejar cambios de prioridades.

Por lo tanto, incorporar un desarrollo ágil al desarrollo basado en conocimiento en Salta, implica por sobretodo un gran cambio cultural, que no es posible lograrlo de un día para otro. Requiere un trabajo con el equipo, la organización de la que depende y el cliente.

A continuación, según las formas actuales en que vienen trabajando los equipos relevados se presenta un lineamiento y plantean prácticas para ir incorporando e internalizando, para poder trabajar con metodologías ágiles.

4.3.1 Estructura de un Proyecto DBC siguiendo prácticas ágiles

La intención de este trabajo no es elegir una metodología ágil de entre aquellas que fueron analizadas, sino buscar la manera de lograr que los equipos de trabajo de desarrollo basado en conocimiento relevados puedan acercarse de una manera simple y sin implicarles demasiados cambios drásticos al desarrollo ágil. Posteriormente, podrán ir incorporando mayores conocimientos sobre metodologías ágiles, obtendrán experiencias y estarán en condiciones de adaptarlo de la mejor manera a su equipo de trabajo y proyectos a desarrollar.

La propuesta se divide en tres secciones principales (Ver Figura 4.3.1), cada uno de ellos con propósitos diferentes, como lo plantea la mayoría de las metodologías ágiles.

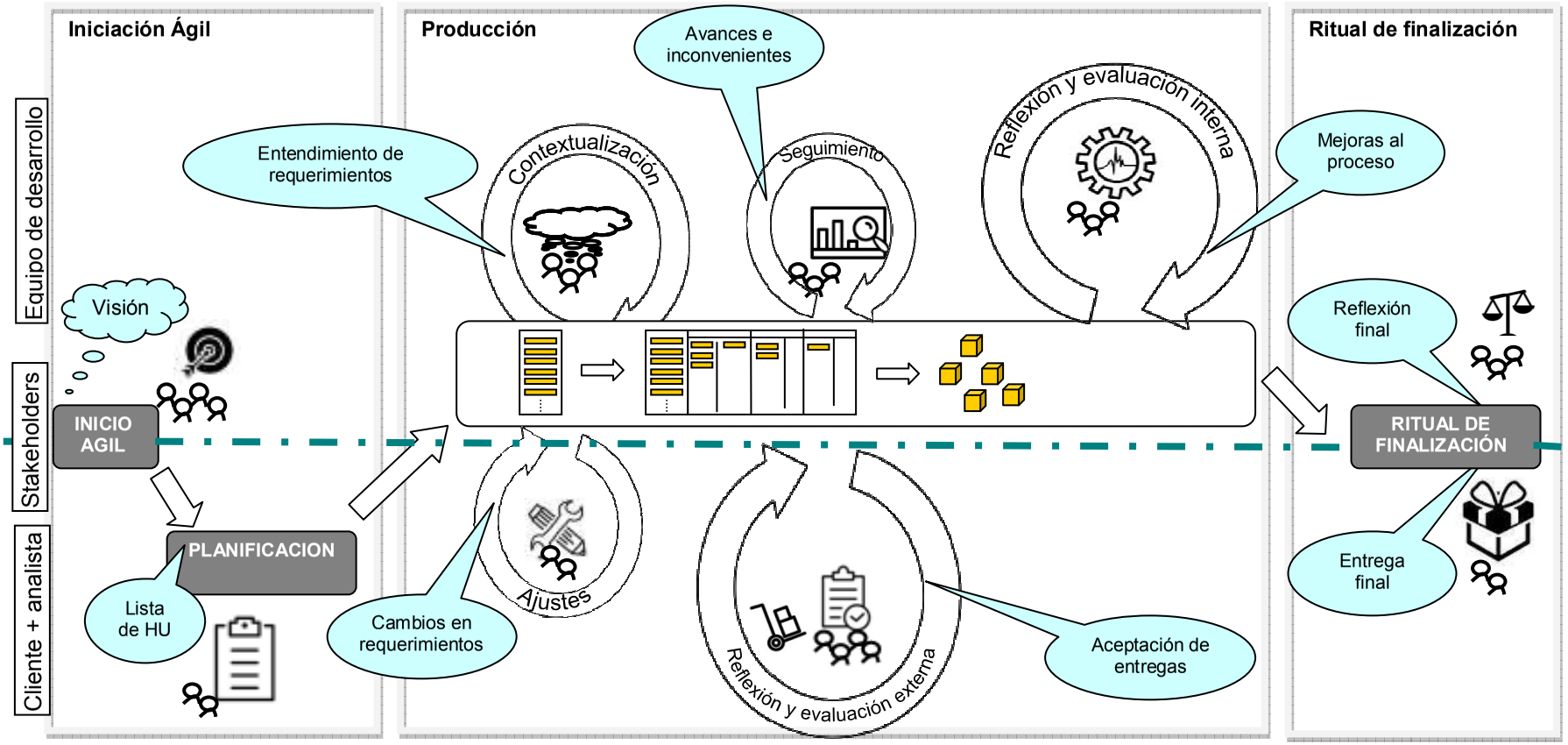


Figura 4.3.1: Prácticas ágiles para el desarrollo basado en conocimiento

La primera parte busca compartir una visión del proyecto a realizar, clarificar los objetivos y razones por las que se realiza el proyecto, la segunda parte se centra en el desarrollo en sí del producto esperado y por último se encuentra el cierre del proyecto, que incluye la entrega final del producto:

1. Iniciación ágil
2. Producción
3. Ritual de finalización del proyecto

A continuación se describe cada una de las etapas que la conforman.

4.3.1.1 Iniciación ágil

Esta etapa busca comprender el alcance del proyecto y sus objetivos y también obtener suficiente información para confirmar que el proyecto debe continuar o convencer que debe cancelarse. El propósito es alcanzar un acuerdo entre todos los stakeholders sobre los objetivos del proyecto. Es similar a la caracterización en Crystal. Puede asemejarse, también, a la fase de inicio de OpenUP o de requerimientos de DSDM.

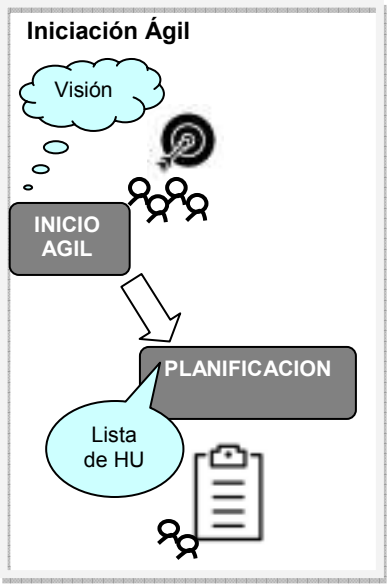


Figura 4.3.2: Etapa de iniciación ágil

El principal objetivo de la iniciación ágil (agile inception) es construir una visión completa sobre el concepto de producto y que además no caiga en sesgos personales, es decir, que esa visión sea compartida y comprendida de idéntica forma por los principales interesados**.** Algunos agilistas consideran que esta conceptualización en el arranque de un proyecto no es necesaria. Ya se sabe que el agilismo es sinónimo de adaptabilidad y que, en cada iteración, se puede contar con actividades de análisis que generarán la retroalimentación necesaria para construir la visión de forma incremental; y que el equipo se adaptará para cumplirla. [Rasmusson]. En metodologías como DSDM, open UP, Crystal existen ciertas tareas relacionadas con la incepción, ya sea que se la llame actividad de caracterización, fase de inicio, definición de requerimientos.

"Los proyectos suelen iniciarse pensando que todos están de acuerdo y piensan lo mismo, asumiendo consenso donde no lo hay. Al comenzar realmente a desarrollar, se detecta que todos tienen algo diferente en mente. Buenos equipos pueden manejar estas situaciones y adaptarse en

la marcha pero es una forma de desperdicio que puede herir la confianza antes de comenzar". [Rasmusson].

En su mayoría, los equipos que trabajan con Desarrollo basado en conocimiento en Salta Capital, delegan en el líder del proyecto la definición de los requerimientos quien trabaja junto con el cliente. Posteriormente presentan el proyecto al equipo encargado de implementarlo. Prestándose esta situación justamente a malas interpretaciones, como las mencionadas por Rasmusson.

Para atacar este problema, Rasmusson propone una herramienta para caracterizar el proyecto de manera liviana. Simplemente plantea responder en una reunión con el equipo y el cliente 10 preguntas antes de comenzar el nuevo proyecto. Estas preguntas sirven para dos objetivos: configurar la alineación y las expectativas. La alineación es sobre asegurarse que todos tengan claro porqué se está tratando de realizar esto y cómo se va a lograr. Las expectativas están relacionadas con la comunicación clara entre el equipo y los stakeholders, que implicará llevar el proyecto al éxito. Se definen aquí las reglas de compromiso, concluyendo si es posible realizar lo solicitado, si es muy complejo y lo que tomará [Rassmusson]. Pero esta reunión puede durar varios días y posiblemente no se pueda contar con el cliente para participar en ella de manera exclusiva. Por esto no se considera en esta propuesta aplicar la herramienta de manera completa.

Para esta etapa se ha simplificado la propuesta de Rassmusson y lo que se propone es realizar dos reuniones: una reunión de inicio ágil y una reunión de planificación inicial. Se busca que al finalizar ambas reuniones se haya construido una visión compartida del proyecto y se tenga un número considerable de historias de usuario priorizadas, listas para comenzar con el proceso de producción. Los detalles de cada reunión se detallan en la siguiente tabla.

	**Una reunión de inicio ágil** donde se realizan dos técnicas propuestas por Ramusson en The Agile Samurai. Todo el equipo debe participar y fundamentalmente los stakeholders. Las dos técnicas son: Definir una visión y Conocer a la comunidad: *Definir una visión común:* si bien parece trivial, todos deben estás de acuerdo en que se quiere construir, porqué. Se debe poder definir los pasos para alcanzarlo e identificar los posibles riesgos que condicionen el éxito del mismo. *Conocer a la comunidad:* también conocida como conocer a los vecinos. Muchas veces cuando el proyecto ya está avanzado aparece un nuevo stakeholder no identificado previamente que debe considerarse y que genera muchas modificaciones en el producto que se está desarrollando. Si se toma el tiempo necesario para poder hacer una lista de todos los posibles involucrados, este riesgo se reducirá. Además se puede delinear la vision para cada uno de los usuarios identificados. El resultado de esta reunión es la determinación de un rumbo común de todos los involucrados en el desarrollo.
	**Una reunión de planificación:** el analista, junto con el cliente definen las historias de usuario, refinando las visiones de los diferentes usuarios previamente definidas. Las historias de usuario son priorizadas y sirven de base para toda la etapa de producción. Se debe lograr tener escritas un número razonable de historias para iniciar, sino, el equipo estaría desarrollando tareas no maduras completamente. [Epstein]. El resultado de esta reunión es una lista priorizada de Historias de Usuario. De ser necesario se puede y debe utilizar la prototipación rápida que permite el DBC para asegurar entender los requerimientos, sobre todo si son complejos utilizándolos ya sea como soluciones SPIKE, o bien siendo la base para el desarrollo posterior refinándolo.

Tabla 4.3.1: Reuniones propuestas para la etapa de Inicio ágil

Lo importante de la reunión de inicio ágil es que se logra entonces introducir en el tema del proyecto que se está iniciando a todos los involucrados, se logra crear una visión compartida del proyecto, inclusive se genera un tiempo de reflexión que permite analizar también posibles riesgos. De esta manera ya no es el líder de proyecto quien transmite la visión al equipo, es el equipo completo quien construye dicha visión. Hay comunicación directa entre todos, se evitan las malas interpretaciones. Es el equipo completo que comienza a avanzar en el desarrollo de este proyecto de manera conjunta.

El núcleo de la reunión de planificación es, no solo, poder definir y priorizar la mayor cantidad de historias de usuario junto con el usuario, sino entender qué es lo que realmente está necesitando el cliente y para ello en muchos casos trabajar con prototipación rápida es de mucha ayuda.

4.3.1.2 Etapa de producción

Antes de presentar esta forma de organizar el desarrollo se tuvo en cuenta la forma de trabajar de los equipos que utilizan Desarrollo basado en conocimiento en Salta Capital. Hay ciertas características, que ya fueron mencionadas y vale la pena volver a resaltar:

- los equipos trabajan en más de un proyecto cada uno: Se podría considerar subdividir al equipo y asignar a cada uno un proyecto, pero algunos equipos son de muy pocas personas y esta acción sería imposible.

- los líderes de proyecto no suelen delegar, ellos deciden que artefactos se reutilizan y fijan pautas del diseño. Sería conveniente que el líder sea uno más del equipo y acompañe desde adentro el proceso de producción.

- existe un sector relevado que si bien divide el personal en equipos de trabajo, estos no son multidisciplinarios. Cada uno realiza una actividad específica para cada proyecto.

- existen especialistas que trabajan en varios equipos al mismo tiempo. Sería importante que ellos puedan tener definidas claramente las tareas a realizar de todos los proyectos donde están involucrados

- no todos trabajan con iteraciones. Dentro de los que trabajan con iteraciones, no todos tienen duraciones establecidas. Los que directamente no trabajan por iteraciones trabajan por módulos o productos a entregar, siendo las duraciones dependientes de la complejidad de cada uno.

Se plantean entonces dos alternativas, definir iteraciones o ver el proceso como un flujo continuo de trabajo. En la encuesta a quienes trabajan en Salta con ágiles, todos trabajan con iteraciones.

Si se quisieran seguir el camino de definir iteraciones, como los equipos trabajan en más de un proyecto a la vez, la situación se vuelve compleja. Sería recomendable subdividir los equipos donde cada sub-equipo tenga asignado un único proyecto y solo sean los especialistas, quienes estén involucrados en varios proyectos, definiendo un Scrum board para cada proyecto y un Kanban board para los especialistas. Pero como se señaló anteriormente, esto puede no ser posible por la cantidad de personal disponible.

La siguiente alternativa, que podría adaptarse más a esta característica, es trabajar en varios proyectos simultáneamente, manteniendo un flujo de trabajo constante, tal como propone Kanban. Es por eso que en esta tesis se propone trabajar de esta manera. Además, dentro de Kanban también es posible trabajar con iteraciones si el equipo desea hacerlo.

Por lo tanto, la etapa de producción de esta propuesta plantea trabajar a través de un flujo de trabajo contante compartido para todos los proyectos que se estén desarrollando en simultáneo. Si surge un nuevo proyecto, se incorporan las historias de usuario o ítems asociados al mismo en el

listado de tareas, según las prioridades que se hayan definido, dentro de los ítems del proyecto y relativos también al resto de proyectos que se vienen desarrollando.

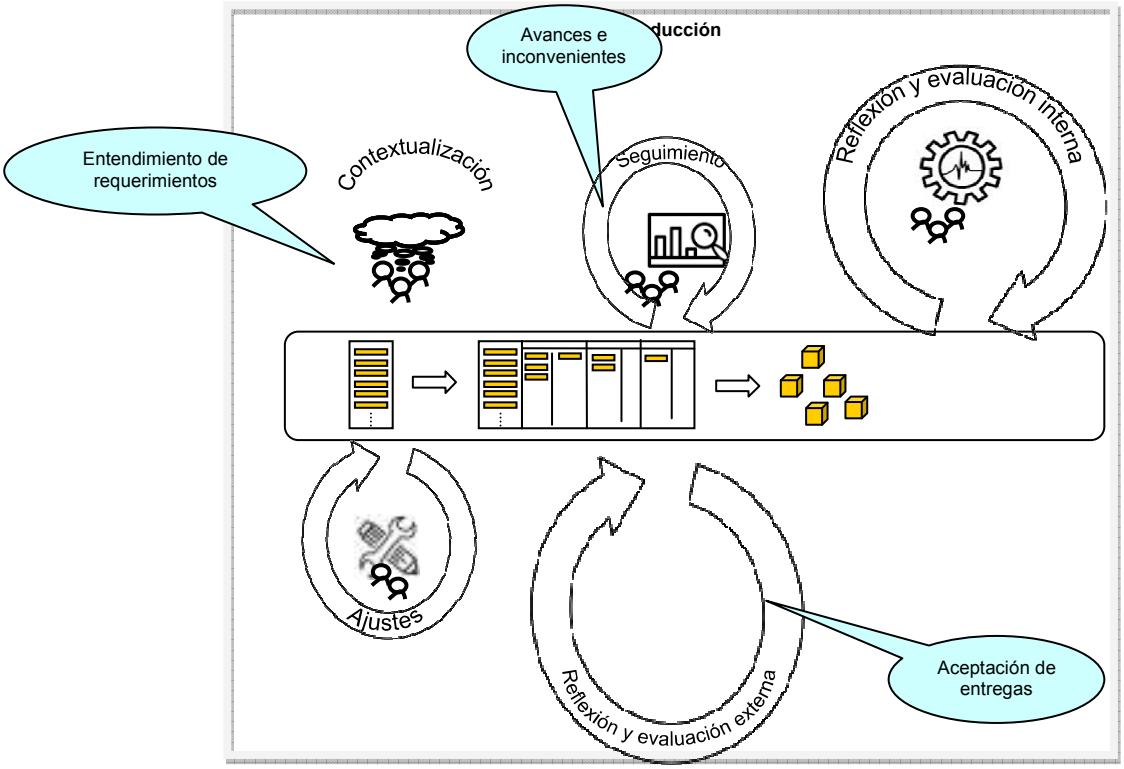


Figura 4.3.3: Etapa de producción

A través de una pizarra Kanban se van incorporando, al listado de tareas, las historias de usuario de diferentes proyectos y se logra tener la visión general de todo lo que se va realizando. Para ello se divide el proceso de desarrollo propiamente dicho en diferentes etapas. Queda a criterio de cada equipo las etapas a considerar, pero lo que se debe establecer claramente es la definición de terminado (*Definition of Done - DoD)* de cada una dentro del proceso, y ésta definición debe ser perfectamente comprendida por cada uno de los involucrados. Esto es muy importante, ya que define la calidad con la que se termina una determinada tarea. Es simple de definir y evita malos entendidos dentro del equipo ya que todos entienden hasta donde se debe avanzar en cada tarea para considerarla terminada. Por ejemplo, el desarrollo debe quedar claro si debe contemplar la realización de todas las pruebas de unidad necesarias o no. También, para evitar cuellos de botella se debería especificar el WIP de cada etapa dentro de la pizarra. La determinación de qué etapas definir dentro de la etapa de producción y el WIP de cada una debe ser definido por el propio grupo y puede ir modificándose y refinando a medida que el equipo adquiera más experiencia.

La lista de Historias de usuario o ítems debe considerar las visiones de los diferentes stakeholders de los proyectos. El analista, junto con el cliente determinará las prioridades de cada ítem. En caso de trabajar en varios proyectos de manera simultánea, luego intercalará los ítems de cada proyecto según sea más conveniente.

Las etapas dentro del proceso de desarrollo pueden ser adaptadas según la forma de trabajar del equipo y la organización. Por ejemplo podría definirse según las actividades del ciclo de desarrollo de Genexus: diseño – prototipación – producción. Otra opción podría plantear etapas como: análisis – desarrollo – prueba – producción. Se pueden ir agregando o eliminando etapas. Se sugiere iniciar con un análisis, dado que en todos los casos relevados del trabajo con DBC, el líder fija pautas de cada funcionalidad a implementar, define si se debe utilizar reutilización y hasta en algunos casos determina algunos detalles específicos a considerar. El que tradicionalmente venía

ejerciendo el rol de líder puede tratar de realizar las tareas en esta etapa. La etapa de desarrollo, es muy general e involucraría toda actividad de diseño adicional, junto con la implementación de la funcionalidad y las pruebas de unidad. Se sugiere dejar explicitada la actividad de pruebas, para que quede evidente la necesidad de probar el software y validarlo con el cliente para cada funcionalidad, independiente de la situación en que se encuentre el proyecto. Las pruebas son muy importantes. Dentro de los sectores relevados esto debería haberse visto como una característica común en todos pero no fue así, por lo tanto se considera necesario recalcar esta tarea, las pruebas son fundamentales para tratar de detectar errores lo antes posible y tratar de validar si se está cumpliendo con los requerimientos especificados. Por último se encuentra la actividad de producción. Cuando una historia de usuario llega a la columna producción, quiere decir que la funcionalidad está lista para ser entregado al cliente, pasó inclusive la etapa de testing que debería incluir pruebas de aceptación en lo posible con participación del cliente. El incremento o prototipo de DBC se encuentra completamente operativo en su ambiente real y puede ser explotado por los usuarios. Es aconsejable tener una cierta cantidad de ítems para ponerlos todos juntos en producción. Cada entrega de producto provee a los stakeholders la oportunidad de entender el valor que ha sido entregado y comprender qué tan bien está marchando el proyecto.

Las correcciones o mantenimiento de sistemas que se encuentren en producción pueden agregarse a la lista de ítems a realizar en cualquier momento y ubicación según su prioridad, después de haber sido planteada en una reunión del analista con el cliente.

Los especialistas, que hubieren dentro de un equipo, pueden tranquilamente trabajar dentro de la misma pizarra con una columna exclusiva y tener una pizarra aparte, donde se duplicará los ítems o historias de usuario. De esta forma cada especialista tiene en una sola pizarra todas sus tareas, priorizadas, de diferentes proyectos y equipos con los que colabora

Si el equipo quisiera trabajar con iteraciones podría definirlas sin ningún inconveniente. De hecho la metodología Kanban puede combinarse con otras metodologías y permite definir iteraciones.

Además de ver el período de producción como un flujo de trabajo, se debe generar un espacio propicio para la comunicación y reflexión. Se plantea mantener cinco reuniones, que se detallan en la siguiente tabla: Una reunión de contextualización, Una reunión de seguimiento, Una reunión de evaluación y reflexión interna, Una reunión de evaluación externa y Una reunión de ajuste. Cada una de estas reuniones surge de una necesidad de comunicación específica.

Cuando aparecen nuevas historias o modificaciones en las historias, el analista debe tener un tiempo para poder explicar al resto del equipo de desarrollo las razones por las que se incorporan o modifican, enriqueciendo las historias con una descripción de lo que conversó con el cliente en la reunión de ajuste. Ya no está presente el cliente, porque difícilmente éste pueda disponer de tiempo para varias reuniones, y por otro lado porque podría haber modificaciones en historias de varios proyectos y pueden discutirse todas en una sola reunión. Como resultado, todo el equipo debe tener claro que es lo que se pretende alcanzar con las historias que se incorpora o modifica. Se ha tomado el nombre de reunión de contextualización porque justamente el objetivo es contextualizar a todos los miembros del equipo de desarrollo en los ajustes que se van a realizar dentro de las historias de usuario a implementar.

El equipo de desarrollo debe acompañarse y conocer qué está realizando cada uno, y que problemas puede estar teniendo. Este tipo de interacción es imprescindible, dentro de las metodologías ágiles y debido a que el objetivo principal es realizar el seguimiento de las tareas realizadas por cada uno del equipo de desarrollo, se tomó de los nombres posibles, para este tipo de reuniones, el nombre de reunión de seguimiento.

En el propio manifiesto ágil se hace mención de la importancia y necesidad de que el propio equipo reflexione sobre su forma de trabajo, los inconvenientes que tuvo, y sobre todo permitiendo un intercambio tranquilo de opiniones que permite hacer crecer al equipo en madurez. De este tipo de reuniones surgen cambios en la forma de trabajar para evitar inconvenientes que se tuvieron anteriormente, o para tratar de mejorar el proceso de desarrollo. Así se va afinando el proceso a

las características propias del equipo de desarrollo. Se ha decidido llamarla reunión de evaluación y reflexión interna porque no participa ninguno de los clientes involucrados, solo el equipo de desarrollo.

También en las metodologías ágiles es importante la retroalimentación del cliente de lo que se le está entregando, y por eso se considera una reunión de evaluación externa. En estas reuniones no participa todo el equipo de desarrollo, solo el analista con un cliente. En estas reuniones se brinda espacio a que el cliente se exprese libremente sobre lo que el equipo de desarrollo viene entregando, manifieste si se están cumpliendo sus expectativas, que tan conforme está con los resultados obtenidos hasta el momento. Toda esta información es muy importante para que el equipo pueda acercarse más a lo que realmente necesita el cliente. El analista, en una de las reuniones de contextualización comentará lo que el cliente comentó en esta reunión.

Una situación muy frecuente es la solicitud de cambios por parte del cliente. Lo ideal es que sean realizados por el cliente en una reunión de ajustes con el analista, para que explique cuáles son las modificaciones ya sea de prioridades, funcionalidades que solicita. De esta manera se tiene la lista de historias de usuario actualizada. Estos datos también son transmitidos al resto del equipo de desarrollo por parte del analista mediante una reunión de contextualización.

En la siguiente tabla se describen todas las reuniones planteadas en esta etapa.

	**Una reunión de contextualización**, cada cierto tiempo el analista comenta al resto del equipo de desarrollo las nuevas historias de usuario incorporadas a la lista de ítems del Kanban board. El resultado de esta reunión es lograr un entendimiento común de lo que el cliente está necesitando y que se encuentra descrito en cada Historia de Usuario.
	**Una reunión de seguimiento**, donde se permite al equipo de desarrollo monitorizar el estado del proyecto, enfocarse en el trabajo a realizar y detectar problemas e inconvenientes. Es decir permite el seguimiento del proyecto Es simplemente una stand-up meeting que podría realizarse por lo menos dos veces a la semana. El resultado de esta reunión es mantener a todo el equipo informado sobre el avance del proyecto y sus inconvenientes.
	**Una reunión de evaluación y reflexión interna,** cada cierto tiempo el equipo de desarrollo, por lo menos cada 2 meses, para evaluar el proceso, reflexionar y proponer mejoras. Ya sea incorporando prácticas, definiendo iteraciones dentro del flujo de trabajo, mejorando actividades dentro del proceso, variando el WIP de cada tarea, etc. El resultado de esta reunión es definir mejoras al proceso de desarrollo.
 	**Una reunión de evaluación externa:** el analista junto con el cliente analizan lo entregado hasta el momento, para obtener retroalimentación y de esta forma brindar al equipo de desarrollo la oportunidad de realizar cambios al proyecto para optimizar la salida. Como consecuencia de esta reunión el cliente puede manifestar la decisión de incorporar nuevas funcionalidades al sistema en desarrollo que se realizará mediante una reunión de ajustes de planificación, que puede realizarse a continuación de esta reunión. Debería realizarse también en períodos no mayores a 2 meses para tener retroalimentación periódica del cliente. El resultado de esta reunión es conocer el grado de aceptación del cliente de lo que se va entregando y brindar oportunidad para realizar nuevos ajustes a los requerimientos por parte del cliente.

	**Una reunión de ajustes:** el analista, junto con el cliente definen nuevas funcionalidades al sistema en desarrollo, eliminan funcionalidades que todavía están en la lista Historias de usuario si ya no son requeridas y se cambian prioridades. En general se realizará cada vez que el cliente solicite modificaciones a los requerimientos solicitados previamente y frecuentemente se realizará a continuación de la reunión de evaluación externa. El resultado de esta reunión es mantener la lista de Historias de Usuario actualizada y priorizada según las necesidades del cliente.

Tabla 4.3.2: Reuniones propuestas para la etapa de Producción

Dentro de las actividades diarias del período de desarrollo es oportuno aclarar que, como se considera el desarrollo basado en conocimiento, se incluye la prototipación rápida. Pero se recuerda que el prototipo de DBC es un producto potencialmente entregable, que con mínimas tareas puede ser instalado en el entorno real de cliente para su explotación. La generación de prototipos de DBC es una característica muy destacada en Desarrollo Basado en Conocimiento y muy útil para realizar validaciones rápidas con el cliente, pero por lo que se ha analiza no es muy utilizada en los equipos para lograr que el cliente participe a lo largo del desarrollo.

Si se trabaja en varios proyectos simultáneos, es aconsejable manejar un color diferente por proyecto, para identificar los ítems de cada uno. No se volvería muy complejo su manejo debido a que los sectores relevados no suelen trabajar en más de dos proyectos a la vez por equipos. El analista debe mantener priorizados los ítems de todos los proyectos asignados a un equipo en un único listado y mantener el equilibrio. Una gran desventaja de que el equipo trabaje en más de un proyecto es que no se focaliza en un solo cliente y un solo proyecto y debe realizar un cambio y contextualizarse, cada vez que deja un proyecto para dedicarse al otro, que pueden tener características completamente diferentes. Sería más saludable, en caso de ser posible, dividir el equipo y asignarle a cada nuevo equipo un proyecto; de esta forma se puede generar la habitabilidad tan anhelada por las metodologías ágiles, y permitir una buena interacción con el cliente. Pero si esta no es la situación, teniendo ciertos recaudos, se puede trabajar con un único flujo de trabajo como se plantea en esta propuesta.

4.3.1.3 Ritual de finalización del proyecto

El período de desarrollo concluye, debido a que se implementaron todas las funcionalidades requeridas por el cliente y no hay más solicitudes de modificaciones o debido a que el cliente decide concluir con lo que ya tiene desarrollado hasta el momento, por ser suficiente para su negocio. En ese momento se empaqueta el software y se realiza la entrega final.

Figura 4.3.4: Etapa de Ritual de finalización

Las reuniones que aquí se realizan son dos, una de entrega final al cliente, entre el cliente y el analista y otra de reflexión final de todo el equipo. Ambas se describen en la siguiente tabla (Ver Tabla 4.3.3). Esta última etapa del proyecto es muy importante para el aprendizaje del equipo de desarrollo. Si bien la reunión de entrega final es importante dado que se cierra el proyecto formalmente con el cliente y se concluye con la relación con el cliente entregándole el sistema empaquetado, la reunión de reflexión final es más importante en cuanto al crecimiento del equipo de desarrollo. El equipo participa de esta reunión con los stakeholders y escuchan la opinión de cada uno de ellos respecto a cuán útil o valioso es para su actividad lo que se desarrolló. Luego el equipo de desarrollo extrae lo positivo y negativo del proyecto y con lo aprendido seguramente propondrá mejoras al proceso de desarrollo.

	Una reunión de entrega final, donde el analista se reúne con el cliente para dar por concluido el proyecto habiendo realizado la entrega del sistema empaquetado desarrollado por el equipo
	Una reunión de reflexión final, el equipo junto con los stakeholders realizan una valoración de lo que se logró construir. Se reflexiona en lo positivo y negativo del proyecto que está concluyendo para aprender la experiencia vivida.

Tabla 4.3.3: Reuniones propuestas para la etapa de Ritual de finalización

4.3.2 Roles

Cuando se comienza a utilizar una metodología ágil la pregunta es qué rol desempeñará el antiguo líder de proyecto, el analista, el tester. La intención fue tratar de definir los roles de tal manera que las personas casi intuitivamente sepan que rol deberían ejercer. Definiendo más roles, se buscó

que los líderes de proyecto actuales deleguen ciertas responsabilidades a determinados miembros del equipo. O bien que participen desde otra perspectiva dentro del equipo al ejercer el rol con el objeto de paulatinamente ceder espacio al propio equipo de desarrollo.

Se tomaron como base las definiciones de roles principalmente de open UP y Crystal, pero se realizaron a algunas de ellas sutiles modificaciones. Los términos utilizados dejan en claro su función principal. Si bien tres tienen una correlación casi directa con los roles principales de Scrum, se plantean otros roles que al iniciarse en metodologías ágiles pueden ser muy útiles y luego pueden ir desdibujando para darle mayor autonomía y poder de auto-organización al grupo.

- **Analista**. Representa las preocupaciones de los clientes y usuarios finales mediante la recopilación de aportes de los stakeholders para entender el problema a resolver y mediante la captura y establecimiento de prioridades para los requisitos. No debe confundirse su función con ser el responsable de realizar las tareas de análisis dentro del Kanban board. Podría considerarse análogo al Product Owner de Scrum.

- **Coordinador**: debe, como mínimo, tomar nota en la planificación del proyecto y las reuniones, así rastrear la información para enviar o presentar. Debe proteger al equipo de distracciones y de otros elementos externos y lo mantiene enfocado. Elimina obstáculos que alejen al grupo de la consecución de objetivos de la iteración. Dirige las reuniones diarias. Realiza el seguimiento del avance. Este rol sería casi el de Scrum Master en Scrum. No es sinónimo de ser el líder del grupo. Puede desempeñar otros roles simultáneamente

- **Diseñador-Programador**: es responsable de desarrollar una parte del sistema, incluyendo diseñarlo de tal forma que se ajuste a la arquitectura, y luego implementar pruebas de unidad, e integrar los componentes que forman parte de la solución. Este rol está tomado de Crystal Clear, donde al definir este rol, Alistair Cockburn combina ambas palabras “diseñador” y “programador” para resaltar que cada persona programa y diseña. Ni diseñador ni programador se mantienen como nombre de un rol separado.

- **Tester**. Este rol es el responsable de acompañar, como describe XP, al cliente en las pruebas funcionales o de aceptación, tomar registro de los resultados. Difunde los resultados en el equipo. Tanto Open UP como Crystal Clear y XP cuentan con un rol de Tester. Pueden ser asignaciones temporales dentro de los miembros del equipo dado que en ninguno de los sectores que trabajan en Salta con Desarrollo basado en conocimiento se tiene una persona específica para diseñar y ejecutar las pruebas, y además debido a que se espera que cada diseñador-programador realice sus pruebas.

- **Cliente**: representa a grupos de stakeholders cuyas necesidades deben ser satisfechas por el proyecto, tal como lo describe OpenUP. Es un papel que puede jugar cualquier stakeholder y puede variar según la experticia necesaria para definir las funcionalidades.

- **Stakeholder**: cualquier persona que es (o será potencialmente) afectada por el resultado del proyecto. Es aquel que se verán afectado directamente con el resultado del proyecto.

Como roles opcional pueden considerarse al diseñador-líder y a los especialistas

- **Diseñador líder**: Es la persona técnica líder, la persona que se supone tiene experiencia con el desarrollo de software, capaz de realizar el diseño mayor del sistema, que avisa cuando el equipo de proyecto está o no en cronograma, y si no lo está como volver al cronograma. Es el encargado de buscar y definir los objetos a reutilizar. El líder de diseño suele tener mayor influencia en el taller para darle forma a la metodología. Se tomó este nombre de Crystal Clear dado que Alistair Cockburn usa la palabra "diseñador" para este rol para cortar el nombre "líder diseñador-programador." El diseñador líder debe diseñar y programar como los otros diseñadores-programadores. Cuando el equipo tiene suficiente experiencia se puede prescindir de este rol y delegar al equipo sus funciones.

- **Especialista:** persona con conocimientos específicos, experto en una temática en particular, por ejemplo Bases de datos, conexiones a bajo nivel con otros dispositivos, etc. Puede trabajar en simultáneo dentro de varios equipos de desarrollo.

El **equipo de desarrollo** por lo tanto está conformado por varios roles: analista, coordinador, diseñadores-programadores, testers, diseñador-líder (si se encuentra definido) y especialistas (si hubieran). A su vez el cliente participa constantemente con el equipo de desarrollo, colabora constantemente y conforma junto con él el **equipo del proyecto**.

Se trató de desdibujar el rol tradicional del líder del proyecto tradicional, incorporando varios roles propuestos por algunas metodologías ágiles. Ser ágil no es sencillo. Implica cambiar preconceptos mentales en la forma de trabajo [VersionOne web].

4.3.3 Elementos propuestos

4.3.3.1 Artefactos

Las historias de usuario (HU) permiten especificar los requisitos del software. Los datos que mínimamente deberían contener son:

- Proyecto al que corresponde: sería conveniente trabajar con diferentes colores por proyecto para que sea fácil identificar las tareas de cada proyecto
- Identificación de la HU.
- Prioridad
- Datos de la funcionalidad descripta: nombre, descripción, criterios de aceptación, etc.
- Fechas: Fecha de ingreso, Fecha comprometida (si es aplicable). También se pueden anotar las fechas de entrada en cada cuadrante que ayuda a obtener información del rendimiento del equipo de trabajo
- Persona que tiene asignado el ítem.

Eventualmente se pueden crear tarjetas desglosando tareas de la HU, especificando las identificaciones de las subtareas asociadas. En la siguiente Figura se muestra un ejemplo de ficha para HU.

<table>
<tr><td>ID_Proy</td><td colspan="2">Nombre de HU</td><td colspan="3">Prioridad</td></tr>
<tr><td>ID_HU</td><td>Fecha Ingreso</td><td>Fecha comprometida</td><td>Cuadrante</td><td>Fecha</td><td>Responsable</td></tr>
<tr><td colspan="3" rowspan="5">Descripción</td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td colspan="3">Criterios de aceptación</td><td colspan="3">Subtareas asociadas:</td></tr>
</table>

Figura 4.3.5: Ejemplo de ficha para HU

Lista priorizada de trabajo: Es una lista priorizada de funcionalidades técnicas y de negocio expresadas a través de historias de usuario o ítems. Estas funcionalidades son requisitos a muy alto nivel de lo que debe hacer la aplicación, donde se listan características, funciones, tecnología, mejoras, etc. que serán aplicadas al producto. Cada ítem es una historia de usuario.

Lista de tareas de especialista: Enumera las historias de usuario que requieren la atención específica de un especialista o sector especial.

Prototipo DBC: Parte de un sistema desarrollado que funciona en el entorno de desarrollo y puede ser potencialmente entregado al cliente.

Podrían considerarse otros artefactos, pero la intención es hacer una propuesta simple, y que ayude a lograr un desarrollo ágil. Adhiriendo a lo que Cockburn menciona, no se busca que sea eficiente sino que sea un conjunto de prácticas suficiente. Después el equipo puede ir adaptando y completando lo que considere más adecuado a su situación particular. Por ejemplo, si fuera necesario realizar un estudio de factibilidad, el documento resultante sería un artefacto a tener en cuenta

Actualmente solo uno de los sector tiene registros muy formales del proceso de desarrollo, que pueden ofrecer cierta resistencia a trabajar solo con los artefactos propuestos, pero cada equipo puede incorporar los artefactos que considere necesarios y en reuniones de reflexión determinará posteriormente si son útiles o no. La intención no es romper con lo que vienen realizando los equipos sino ir sugiriendo alternativas para orientarlos hacia una metodología ágil.

4.3.3.2 Instrumentos

Kanban Board: Tablero de Trabajo con las Historias ordenadas para un equipo, pudiendo corresponder a más de un proyecto. Es recomendable utilizar algún marcador distintivo para cada proyecto, por ejemplo diferentes colores. Cada tarjeta en esta pizarra tiene: Fecha de ingreso, Fecha comprometida (si es aplicable), Descripción y Prioridad. [Modezki 2011]. Se propone que tenga en su forma más básica 5 columnas: ítems o HU, análisis, desarrollo, aceptación (testing), producción. Salvo la primera y última columna, cada columna se divide en 2 sub columnas (haciendo y terminado). También el tablero debe reflejar cual es la definición de terminado (DoD) de cada etapa del proceso de desarrollo y el WIP fijado para cada uno. Si hubieren tareas para especialistas, se debe considerar una columna más por cada especialidad. Cuando una tarjeta sea arrastrada hacia allí, se crea una copia en el Kanban board del especialista. Y, cuando el especialista termina la tarea desplaza la tarjeta a la columna terminado del Kanban board. Pero esta es solo una propuesta (Ver Figura 4.3.6) y puede modificarse, simplemente se agregan o eliminan columnas según sea necesario. Tranquilamente se puede utilizar una pizarra web de las cuales existen muchas propuestas en el mercado, como por ejemplo KanbanFlow, Kanbanize, Jira. La mayoría permite una visión simple y una más detallada de cada item definido, permite el uso de diferentes colores, y tiene una versión para utilizar desde dispositivos móviles.

Items (HU)	*Análisis: (WIP)*		*Desarrollo (WIP)*		*Aceptación (WIP)*		*Producción (Entrega)*
	Haciendo	*Terminado*	*Haciendo*	*Terminado*	*Haciendo*	*Terminado*	
HU1 *HU2* *HU3* *…..*							
	DoD Análisis		*DoD desarrollo*		*DoD Aceptación*		

Figura 4.3.6: Ejemplo de Tablero Kanban para utilizar

Kanban Board para especialista: Los especialista que trabajan en varios proyectos deberán tener su Kanban board donde se les muestre todas las tareas a realizar de todos los proyectos.

El Burndown chart: Lo importante es que se muestre al equipo progreso hacia su objetivo, no en términos de cuanto hicieron en él, sino en término de cuanto trabajo queda en el futuro –lo que separa al equipo de su objetivo. Lo realiza el coordinador o puede generarse con la herramienta si se trabaja con una pizarra web.

4.3.3.3 Reuniones

Los equipos que trabajan con Desarrollo basado en conocimiento en Salta Capital, realizan reuniones pactadas dentro del equipo. Algunos realizan reuniones con los clientes para mostrar prototipos, otros para realizar ciertas pruebas. Lo importante de las reuniones es que permiten obtener información muy relevante para el proyecto, sobre: qué es lo que hay que hacer, si lo que se hizo es correcto, si la forma de trabajo fue correcta o puede mejorarse. La mayoría de las metodologías plantean diferentes tipos de reuniones para obtener esta información. En esta propuesta se plantean 9 tipos de reuniones, enumeradas en la siguiente tabla.

Etapas	Reuniones
Iniciación ágil	Una reunión de inicio ágil Una reunión de planificación inicial
Producción	Una reunión de contextualización Una reunión de seguimiento, Una reunión de evaluación y reflexión interna Una reunión de evaluación externa Una reunión de ajustes
Ritual de finalización	Una reunión de entrega final Una reunión de reflexión final

Tabla 4.3.6: Reuniones propuestas por etapa

Cada organización según su experiencia podrá ir adaptando estas reuniones a las características del equipo. Pero hay algunas consideraciones con tres reuniones claves, que vale la pena mencionar:

Las reuniones de reflexión son una práctica que muchos equipos saltean, y esto es una oportunidad que se pierde del equipo de discutir lo que está funcionando y lo que no y consensuar los cambios a probar-

Las reuniones de seguimiento solo deben poner al día al equipo, no resolver problemas. De ser necesario el analista se reunirá con quien sea necesario para resolver las situaciones que merezcan atención. Dado que aquí se plantea no realizar reuniones diarias sino hacerlas dos veces a la semana, se puede mantener al equipo informado vía correo electrónico, y utilizando alguna herramienta ágil. [Epstein]

Las reuniones de planificación no deben ser muy largas y no deben confundirse con reuniones de diseño.

4.3.4 Prácticas recomendadas

Si se analizan las prácticas propuestas por las diferentes metodologías, y especialmente por Xp, todas ellas brindan grandes ventajas al aplicarlas en el desarrollo. Según Cockburn, a medida que reflexione el grupo irá añadiendo algunas. Las prácticas ágiles son importantes, pero que el equipo internalice los principios que las sustentan y que están declarados en el manifiesto ágil son los que realmente modifican la manera de trabajar.

De las prácticas de XP todas podrían aplicarse al desarrollo basado en conocimiento, si bien programación de a pares y cliente en el lugar son dos prácticas más complejas de llevar a la práctica tal cual las enuncia XP. La primera podría considerarse solo para situaciones muy complejas y la segunda considerar la utilización de un representante del cliente. Una tercera práctica compleja para implementar es la propiedad colectiva del código; si bien es una práctica muy recomendada, en algunos equipos puede no funcionar, especialmente si hay personas con conocimientos específicos, expertos o especialistas, donde ellos claramente son los responsables

de una porción específica del software. En el análisis del uso de metodologías ágiles realizado, sobresalió justamente esta característica de no haber propiedad colectiva del código.

Por otro lado es obvio que se logrará un software de mayor calidad si se practican las prácticas propuestas por XP, pero para comenzar con los primeros pasos en un desarrollo ágil, puede prescindirse de ellas y dejarlas para ir incorporándolas paulatinamente, según el propio grupo vea la necesidad y las ventajas de realizarlas. Mientras menos sea impuesto desde afuera al grupo de desarrollo más fácil el grupo adoptará la forma de trabajo y la irá adecuando a sus características específicas. Jim Highsmith da el consejo de comenzar con menos de lo que se piensa necesitar y probablemente eso sea todo lo necesario; es más fácil agregar algo después que quitar algo

Los equipos ágiles maduros incorporan algunas prácticas técnicas claves dentro del procedimiento estándar de operación para mantener una paz sostenible con un desarrollo de software rápido. Dos de estas prácticas son integración continua y pruebas automatizadas. Producir construcciones limpias, integradas varias veces cada día y correr esos desarrollos a través de suites de pruebas automatizadas, aumenta la velocidad de desarrollo. También ayuda a mantener el código estable, con el que puede interactuar el stakeholder alimentando ciclos más cortos de retroalimentación. Estas suelen ser prácticas difíciles de incorporar debido a restricciones de capital, pero son importantes para la sustentabilidad a largo plazo. [Kindon 2007]. Estas dos prácticas van de la mano con la refactorización.

También establecer estándares de codificación ayuda a poder aumentar el porcentaje de reutilización en futuros proyecto.

Por lo tanto solo se analizará una forma de lograr implementar estas cuatro prácticas:

1. **Pruebas:** al ser un desarrollo evolutivo es necesario incorporar el uso de pruebas de unidad y de aceptación. La automatización de pruebas unitarias es crucial para asegurar que no se propagaran errores al realizar modificaciones en el sistema. En el relevamiento a personas que utilizan metodologías ágiles se obtuvo un bajo porcentaje de aquellas que utilizan pruebas automatizadas. La tendencia actual es integrar TDD en cualquier metodología independientemente ya sea ágil [Schmidkonz 2007] o tradicional [Letelier 2003] y aprovechar los beneficios de practicar una metodología que siempre permite deshacer los errores, asegurar una calidad del producto y protegerse de errores tanto malintencionados como humanos. Pero esto se puede ir realizando a medida que el equipo logra cierta madurez. Para pensar en incorporar TDD, se debe contar con ciertas herramientas que sin ellas sería imposible realizar las tareas que la metodología propone, son.
    - **Herramientas para Pruebas unitarias**: actualmente existe GXUnit para realizar este tipo de pruebas con Genexus. Es una herramienta libre y gratuita que permite la ejecución de pruebas unitarias de forma automática.
    - **Repositorios de código**: que facilita el control de versiones, acceso e integración de código. Genexus permite el control de versiones.
    - **Software de integración continúa**: GXServer es útil para esta actividad.
    - **Herramientas de construcción automáticas**: GxTest ayuda en este sentido.

2. **Refactorización**: refactorizar ahorra mucho tiempo y supone un incremento de calidad. En tiempos de cambios rápidos y constantes se necesita prestar más atención en refactorización. Refactorizar está muy relacionado con lo que hoy se refiere al uso de patrones de diseño. Los patrones de diseño proveen formas de mejorar la calidad de diseños iniciales ofreciendo modelos que han probado ser efectivos en el pasado.

3. **Integración continua**: El código se debe integrar como mínimo una vez al día, y realizar las pruebas sobre la totalidad del sistema. La integración continua a menudo reduce la

fragmentación de los esfuerzos de los desarrolladores por falta de comunicación sobre lo que puede ser reutilizado o compartido. En Genexus, sin GXServer no se puede trabajar en un mismo objeto simultáneamente, si bien actualmente se puede dividir en módulos. En esos casos se puede realizar programación con Dummies donde cada programador crea los métodos de su Objeto, pero no con su detalle sino solo la cáscara, son métodos huecos. De manera que el otro objeto pueda interactuar con él aunque no esté completamente implementado.

4. **Estándares de programación**: para facilitar los cambios, mantener el código legible y para maximizar la reutilización posterior de lo desarrollado. El seguir patterns de Genexus en el desarrollo, ayuda mucho en este sentido. Exige, en un principio, control por parte del líder de proyecto para supervisar que realmente se estén cumpliendo los estándares, aplicando patrones, pudiendo hacerlo mediante la toma de muestras aleatorias de lo desarrollado. Una vez que ya se encuentran internalizados los estándares en el equipo, los controles pueden disminuirse. Esta es una práctica muy útil y simple, si bien parece fácil de incorporar, requiere convencer al equipo de las bondades de esta práctica.

El diseñador - líder del proyecto, o en caso de no existir, el coordinador, debe ir capacitando al equipo, y formándolo dentro de la cultura ágil. Debe ir explicando diferentes prácticas. Entre todo el equipo es útil poder construir una lista priorizada de prácticas ágiles que quisieran ir incorporando al proceso de desarrollo. En las reuniones de retrospectiva, se puede dedicar un cierto tiempo a determinar cuál de ellas podrían implementar a continuación y analizar en la siguiente reunión de este tipo los resultados obtenidos, incluyendo los problemas, los inconvenientes y sus posibles causas, y los beneficios obtenidos. Según esta información el equipo podrá decidir realizar ajustes a esta práctica, mantenerla tal cual por lo menos por un tiempo más o bien dejar de utilizarla.

4.3.5 Reflexión

En esta sección se planteó una propuesta para realizar el Desarrollo Basado en Conocimiento tratando de alcanzar, con ciertas prácticas, los postulados el manifiesto ágil, tomando como base un estudio de campo en organizaciones de Salta Capital que trabajan con desarrollo basado en conocimiento. Se planteó una estructura general para realizar los proyectos, se definieron roles, algunos elementos (artefactos, instrumentos y reuniones) y se realizó una recomendación de determinadas prácticas ágiles que sería conveniente aplicar. Nada de lo propuesto pretende ser una regla, más bien un lineamiento que oriente a las organizaciones que quieren iniciar el cambio hacia las metodologías ágiles. Cada organización adoptará lo que considere necesario y reemplazará aquellos que crea que no se adapta a su situación específica.

Para finalizar, como se mencionó reiteradamente, para poder incorporar un desarrollo ágil al desarrollo basado en conocimiento en contextos como los relevados en Salta Capital, es necesario un gran cambio cultural, que debe realizarse paso a paso. A medida que se vayan conquistando pequeños logros, todos obtendrán mayor confianza en esta nueva forma de trabajo, equipo, cliente y organización.

4.4 EXPERIENCIA REAL

Intentar realizar experiencias en ambientes reales con todo el trasfondo analizado anteriormente es muy difícil debido a que es imposible pensar que alguno de los sectores relevados hubiera permitido realizar experiencias empíricas según esta propuesta de prácticas con sus propios equipos. Esto es lógico por la información sensible que se maneja y los valiosos tiempos de trabajo de cada persona involucrada. Se envió la propuesta a todas los sectores relevados para que la evalúen, y se les envió junto con ella una breve encuesta que permitiría relevar su opinión. Solo una de las empresas privadas respondió hasta el momento, sus comentarios se describen en una sección posterior. Es de esperar que se reciban otras opiniones y puedan enriquecer con su aporte a la propuesta más adelante, pero por limitaciones de plazos para realizar este informe solo se mostrará los datos recibidos por esta única empresa.

4.4.1 Contextualización de la experiencia

Para poder realizar una experiencia real de desarrollo de software con la metodología propuesta y poder validar su viabilidad, se solicitó la colaboración de una cátedra del último año de la Licenciatura en Análisis de Sistemas de la Universidad Nacional de Salta llamada Desarrollo Basado en Conocimiento 1. La cátedra ha contado este año con 10 alumnos y fueron divididos en dos equipos (Grupo 1 y Grupo2) de cinco integrantes cada uno para que cada equipo trabaje de manera independiente sobre los mismos proyectos.

Se creó en la plataforma educativa Moodle un espacio para poder compartir información con los alumnos y donde cada grupo pudiera compartir entre ellos información. La información vertida en la plataforma ayudaría también al seguimiento de la experiencia.

Los alumnos involucrados en esta experiencia básicamente se formaron en el desarrollo tradicional, Proceso Unificado. Si bien, en una cátedra de Análisis y diseño se les describió en qué consisten brevemente y de manera superficial este tipo de desarrollos, su única experiencia en el ámbito académico era con metodologías tradicionales. En el ámbito universitario, ninguno realizó previamente alguna actividad de desarrollo siguiendo una metodología ágil.

Lo primero que se realizó a través de la plataforma fue una breve encuesta a los diez alumnos para determinar: la experiencia previa de cada uno con Genexus, cuántos trabajaban, cantidad de horas que trabajan y tipo de trabajo realizado. A su vez, si trabajaban en el desarrollo de software, se sondeó si seguían metodología ágil.

Respecto a la experiencia previa con Genexus antes de iniciar la asignatura, un 10 por ciento de ellos consideraba que tenía experiencia moderada y venía trabajando con Genexus hace un año (Ver Figura 4.4.1). Otro 20 por ciento considera que tenía conocimientos mínimos siendo la experiencia de uno de ellos de un año de trabajo y del otro menos de un mes. Pero la cátedra consideraba que todos tenían los conocimientos mínimos para realizar esta experiencia.

Figura 4.4.1: Experiencia con Genexus dentro del grupo de alumnos para iniciar experiencia

El 50 por ciento de los alumnos de la experiencia se encontraba trabajando, y algunos de ellos acababan de ser contratados (Ver Figura 4.4.2). En Figura 4.4.3 se muestra la cantidad de horas dedicada por cada alumno al trabajo.

Dentro del porcentaje que trabaja, el 60 por ciento realiza desarrollo de software para una misma empresa privada. Curiosamente esta empresa, es una empresa que desarrolla con Desarrollo basado en conocimiento utilizando Genexus, y dicha empresa fue relevada al hacer el estudio de campo presentado previamente. Esta empresa estaba comenzando a trabajar adaptando su proceso de desarrollo a las metodologías ágiles, por lo que alguna noción adicional tenían estos alumnos y solo uno de ellos venía trabajando en esa empresa por aproximadamente un año. Entonces, en términos generales, solo uno de los diez alumnos, un 10 por ciento, tenía cierta experiencia en desarrollo de software tratando de seguir prácticas ágiles.

Figura 4.4.2: Alumnos que trabajan dentro del grupo para iniciar experiencia

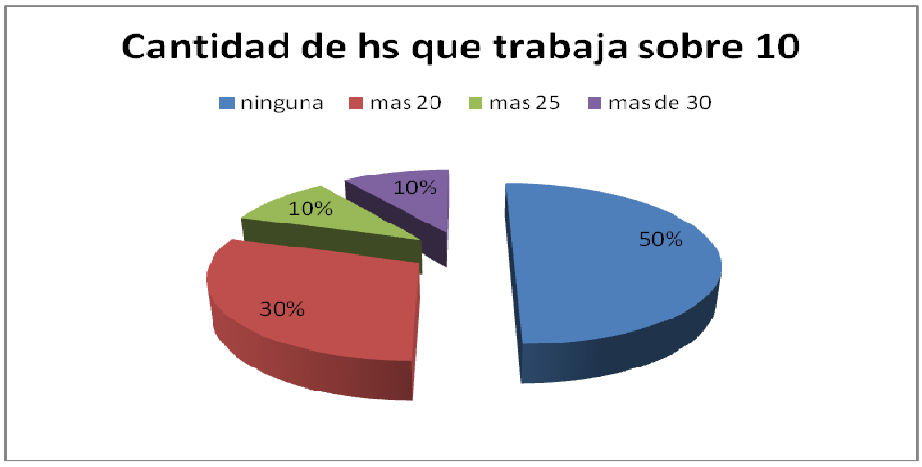


Figura 4.4.3: Cantidad de horas dedicadas al trabajo dentro del grupo de alumnos para iniciar experiencia

Todos los alumnos aun trabajando en desarrollo de software siguiendo metodologías ágiles, según ellos, reconocieron que existía un líder de proyecto. También reconocieron que frecuentemente trabajan en más de un proyecto a la vez pero comparte horario y lugar de trabajo. Este último punto también se debe destacar, dado que en la empresa donde trabajan antes no compartían el horario de trabajo, a veces se solapaba. Por lo que se detecta un cambio que anteriormente los dueños de la empresa consideraban que no podrían realizar.

Por todo lo antes mencionado, se consideró que los equipos reflejarían la situación de los equipos relevados en el estudio de campo de Salta Capital, donde casi no había experiencia con metodologías ágiles, y donde existe la figura presente de un líder de proyecto. De esta manera los

equipos conformados permitirían reflejar el comportamiento de las personas frente a una nueva forma de trabajo, buscando entre otros objetivos desdibujar el rol de líder tradicional siguiendo estas propuestas ágiles

Para determinar los miembros de cada grupo, se buscó que en cada uno participara uno de los alumnos con mayor experiencia en el Desarrollo basado en conocimiento con Genexus, quienes llevaban un año trabajando con este tipo de desarrollo. Dicha división fue planteada desde la cátedra, una vez que los principales conocimientos sobre el Desarrollo basado en conocimiento estaban afianzados en los alumnos.

4.4.1.1 Capacitación y Armando de los equipos

No se debe perder de vista ciertas restricciones en la experiencia realizada:

- los alumnos participantes de la experiencia no fueron alumnos que estuvieren muy familiarizados con las metodologías ágiles, como se mencionó anteriormente, sino más bien estaban formados en metodologías tradicionales.
- los alumnos fueron introducidos al desarrollo basado en conocimiento recién al estar cursando esta asignatura, si bien un 30 por ciento ya se encontraba trabajando fuera del ámbito universitario con Desarrollo basado en Conocimiento, solo un 20 por ciento tenía un año de experiencia. Por lo tanto durante la experiencia surgieron algunas cuestiones relacionadas con la forma de trabajo en Desarrollo basado en Conocimiento, que en equipos con mayor experiencia no hubieran aparecido, o se hubieran solucionado de manera más rápida
- los alumnos tenían otras obligaciones, otras materias, trabajos por lo que el tiempo de dedicación al desarrollo fue durante las horas de la materia (10 hs semanales), y otras horas que pudieran dedicar (mayormente durante los fines de semana). Se solicitó al iniciar la experiencia que por lo menos dediquen otras 10 horas.

Mediante una charla se planteó a ambos grupos la propuesta de trabajo a seguir. Se expuso claramente las tres etapas de la propuesta de desarrollo, dejando bien en claro los objetivos de cada una. Se explicaron todos los tipos de reuniones involucrados en cada etapa. Se detallaron los diferentes roles y se mencionaron los artefactos a utilizar. Se entregó un documento con todos los detalles de la propuesta y se les realizó un coloquio la siguiente clase para asegurar que todos comprendieran la forma de trabajo antes de iniciar. Todos los alumnos parecieron estar entusiasmados por esta propuesta práctica de combinar lo aprendido de Desarrollo Basado en Conocimiento y las Metodologías Ágiles. Esto se vio reflejado en que ninguno desaprobó el coloquio, significando así que dedicaron tiempo de calidad a entender cómo se pretendía trabajar.

A algunos alumnos les costó entender que los requerimientos podrían variar a lo largo del desarrollo, creían imposible poder llegar a un desarrollo aceptable bajo estas condiciones. Otros, vieron con mejor agrado no tener que documentar formalmente los requerimientos y diseños del software a desarrollar. No entendían la necesidad de todas las reuniones detalladas y algunos pensaban que podían llegar a ser opcionales. Pero, luego de las aclaraciones necesarias, los miembros de los dos grupos estuvieron en condiciones de iniciar la experiencia práctica.

Antes de iniciar con los dos proyectos se dio un tiempo para que cada grupo se auto-organice con total libertar. La única consigna fue que expliquen en la Plataforma las decisiones tomadas. Se creó en la plataforma educativa Moodle un espacio para que cada grupo además de compartir entre ellos información, tengan un lugar donde puedan resumir las experiencias y resultados de cada reunión. La información vertida en la plataforma ayudaría también al seguimiento de la experiencia.

Cada grupo pudo distribuir los roles según su propio criterio sin intervención externa. Cada grupo definió los estándares a seguir, si bien estaban basados en patrones. Cada grupo definió las

actividades a considerar en el tablero de trabajo incluyendo la Definición de Terminado (DoD) de cada una.

No se planteó como obligatorio el uso de ninguna herramienta específica, más bien se les dio libertad a cada grupo de buscar y decidir cuáles utilizar. Aunque se les exigió que utilicen por ejemplo alguna pizarra, sea física u on-line.

Solo los clientes fueron personas externas que representaban las necesidades de cada negocio a implementar. Estos clientes no eran alumnos de la cátedra, sino personas con necesidades reales en las áreas planteadas para el desarrollo.

Durante todo el tiempo que duró la experiencia se brindó asesoramiento sobre la propuesta metodológica a seguir y sobre el trabajo con Genexus para realizar el Desarrollo basado en Conocimiento. Pero, se dejó que cada equipo trabaje libremente.

4.4.1.2 Sistemas Objetos

Tal como se mencionó en el estudio de campo, todas las áreas de desarrollo consultadas trabajan en más de un proyecto a la vez. A su vez, todos los equipos dentro del área de desarrollos trabajan en más de un proyecto al mismo tiempo. Debido a esto, se consideró esta cualidad como muy relevante y para poder generar en la experiencia algo similar, se planteó un primer proyecto y apenas iniciado se planteó un segundo proyecto.

- El primer proyecto consistió en el desarrollo de un sitio web para una librería, que buscaba principalmente obtener mayor cantidad de clientes y brindar beneficios a determinados clientes.
- El segundo proyecto era el desarrollo para un gimnasio, cuyo objetivo principal era llevar transparencia al control del mismo.

Para cada proyecto se contó con un cliente que participó de todas y cada una de las reuniones enunciadas en la propuesta donde debía participar. No se mencionan los nombres de los clientes porque solicitaron que no se utilizara su nombre real en ambos casos. Para el caso de la librería se la denominó Mundo Libro, para el gimnasio el cliente dejó que cada grupo cree un nombre de fantasía.

La reacción de los alumnos ante el desarrollo de dos proyectos simultáneos fue de sorpresa, dado que en la carrera nunca se les había pedido algo similar. Consideraban en su gran mayoría, el 88.89 por ciento según encuesta realizada, que iba a ser imposible realizar ambos proyectos, sobre todo por el poco tiempo disponible de cada uno, en algunos casos también influyó el poco conocimiento de Genexus.

4.4.2 Descripción de la experiencia

En esta sección se procederá a realizar la descripción de cada una de las etapas en el marco de la experiencia realizada, y al final de la sección se analizará la forma de trabajo de cada grupo. Para realizar este informe se tuvieron en cuenta los informes redactados en la plataforma, la información en el tablero de tareas, la encuesta realizada al final y otros recursos de distribución utilizados dentro de la plataforma como por ejemplo wikis y foros. Pero, antes de describir como se fue trabajando con las tres etapas en dos proyectos simultáneos; se describirá la forma de auto-organización de cada grupo.

4.4.2.1 Auto-organización

Grupo 1

Como primera medida mediante una reunión definieron los roles de cada miembro, y los horarios disponibles de cada uno. También acordaron trabajar con la herramienta kanbanize y probar el uso

de GXserver en su versión gratuita. Definieron la pizarra, sus tareas y DoD. Y planearon, según lo informado en la plataforma, trabajar con iteraciones de 1 semanas.

Definieron los siguientes roles

- Coordinador / Analista / Diseñador líder: una persona ocupando los tres roles
- Diseñador Programador: cuatro personas
- Tester: según disponibilidad para trabajar con el analista o cliente
- Cliente: cliente real de librería y cliente real de gimnasio
- Stakeholder: vendedor de librería y secretario de gimnasio.
- No hay especialista definido.

En la pizarra de tareas definieron las siguientes actividades y para cada una mantuvieron su columna Realizando y Terminada, salvo para la de Producción:

- Análisis y diseño: el resultado es completar con información relevante que ayude a definir los objetos y especificar si se deben reutilizar otros objetos
- Desarrollo: definición de la estructura de datos y ABM de los objetos y creación de reportes asociados a los objetos involucrados
- Testing: el resultado es obtener objetos que estén probados en diferentes situaciones, en cada tarjeta se escribe los casos de prueba importantes, y se integran en la KB general
- Producción: se procede a Instalar en el servidor provisto por cliente para ir trabajando con las funcionalidades implementadas, se envía mail al cliente informando lo incorporado.

Toda la información fue volcada a una Wiki creada para el grupo, dentro de la plataforma.

Grupo 2

En una reunión presencial de 25 minutos definieron los roles de cada miembro, horarios compartidos de trabajo y horarios en que cada uno trabajaría por separado desde su casa. También acordaron trabajar con la herramienta Kanbanize para la pizarra de tareas, Genexus Evolution 3, Microsoft Excel 2013 y GXServer en su versión gratuita. Definieron la pizarra, sus tareas y DoD. La información más relevante como DoD, así como los horarios de trabajo disponibles y roles adoptados por cada uno fueron expresados en una Wiki dentro de la plataforma

Definieron los siguientes roles

- Coordinador / Analista: una persona ocupando los dos roles.
- Diseñadores-Programadores: dos personas.
- Diseñador líder: una persona
- Tester: una persona.
- Cliente: cliente real de librería y cliente real de gimnasio
- Stakeholder: vendedor de librería y secretario de gimnasio.
- Especialista: sin especificar

En la pizarra de tareas definieron las siguientes actividades:

- Análisis definición clara del nivel de desarrollo y especificación de reutilización si corresponde
- Desarrollo: objetos desarrollados incluyendo interfaz

- Prueba: desarrollado funciones correctamente y aprobado por el cliente
- Producción: Sistema funcionando en el Server determinado por el cliente

Y para cada una se mantuvo las dos sub-columnas propuestas por la herramienta indicando si se encontraba en progreso o bien finalizadas o terminadas. En la Wiki del grupo dentro de la plataforma se registró toda esta información.

4.4.2.2 Iniciación ágil del proyecto LIBRERÍA

Por una cuestión de tiempos del cliente, se trabajó en la iniciación ágil con ambos grupos participando de la misma reunión.

Reunión de inicio ágil

Fecha: 4 de mayo **Duración: 45 minutos**

En la reunión participaron el cliente, los dos equipos y un stakeholder de la librería y se obtuvo con claridad la *visión común del sistema* a desarrollar para la librería: aumentar los ingresos, mediante la captación de mayor cantidad de clientes. Se trabajó con preguntas directas al cliente. En esta reunión también se buscó *conocer a la comunidad*. Aparecieron diferentes tipos de stakeholders: dueño, vendedor, despachante, cliente virtual, cliente real, cliente preferencial, editorial..

Se pudo observar que gracias a la actividad de conocer a la comunidad aparecieron tres tipos de stakeholders que, en un principio, el cliente, dueño de la librería, no había hecho mención: despachante, cliente virtual y cliente preferencial.

Al tener como meta de esta actividad definir una visión, se clarificó el propósito fundamental que algunos habían malinterpretado durante ciertos momentos de la reunión con otros objetivos que para el dueño de la librería eran importantes pero no eran fundamentales. Si no hubiera existido una referencia explícita a enunciar formalmente esta visión, habría quedado en cada integrante de los equipos diferentes visiones: registrar las ventas, obtener reportes estadísticos para controlar al personal, realizar seguimiento de envíos, entre otros.

Reunión de planificación

Fecha: 4 de mayo **Duración: 1 hora aprox**

Luego de la reunión de iniciación ágil, se quedaron en el lugar de reunión el analista de cada grupo y el cliente, dueño de la librería. Durante la misma el cliente enumeró muchísimas funcionalidades que pretendía que tenga el sistema. Uno de los analistas se resistía a registrar todas las funcionalidades aduciendo que no debían ser tan importantes y que podrían dejarse para otro momento. Ante esa actitud, se interrumpió la reunión, y se aclaró que el cliente tenía derecho a expresar todas las funcionalidades que quisiera en el sistema y que después serían priorizadas. Hecha esta salvedad continuó la reunión sin inconvenientes. El haber tenido definido los diferentes stakeholders permitió a los analistas hacerle preguntas al cliente, dueño de la librería, sobre las funciones de cada uno, qué tareas podría realizar interactuando con el sistema. Sobre todo se vio la utilidad en la definición de las actividades relacionadas con el despacho de pedidos de libros, que no habían sido consideradas en profundidad. Ante la pregunta puntual de uno de los analistas quedó completamente claro este circuito y las funcionalidades requeridas. Lo mismo surgió con el trato preferencial a ciertos clientes de la librería. Durante la reunión los analistas fueron los que escribieron las historias de usuarios con las funcionalidades que iban surgiendo. Iban preguntando información complementaria para comenzar también a definir los criterios de aceptación. No fue necesario utilizar un prototipo para las funcionalidades expresadas. Posteriormente se fue revisando cada historia de usuario con el cliente (cada grupo redactó las historias de manera separada), y finalmente el cliente priorizó las 25 historias de usuarios que allí surgieron.

Como se mencionó, el haber tenido identificado a todos los stakeholders facilitó la conducción de la reunión, centrándose en un stakeholder por vez y determinando todas las funcionalidades asociadas al mismo.

Al cliente le resultó un poco complicado poder decidirse en las prioridades de algunas historias. Aquí ayudó haber definido la visión común del sistema, uno de los analistas recordó al cliente cuál era y teniendo esto en claro se determinaron con mayor facilidad las prioridades y algunas otras fueron modificadas. El cliente tomó más tiempo para priorizar las historias más importantes y después de la historia 25 ya no tenía demasiado claro que prefería tener implementado antes. Si bien les dio un orden dejo en claro que para una reunión posterior tal vez podrían variar estas prioridades.

Una vez retirado el cliente, se recordó a los analistas que ya estaban por comenzar la etapa de producción y que el inicio la marcaba la primera reunión de contextualización de la Librería. Se hizo énfasis en la necesidad de asegurarse que el resto del equipo se enterara de lo conversado en esta reunión y se hizo hincapié que en debían aclarar bien a los miembros del equipo cada historia de usuario redactada.

También se aclaró verbalmente y luego por la plataforma dos puntos muy importantes:

1. No se debía confundir Historia de usuario con Casos de uso ya que no se centra en cómo realizarla ni tampoco pretende ser una definición exhaustiva

2. No se debía dar una historia por hecha cuando estuviera prácticamente hecha, sino tener en cuenta que realmente cumpla el DoD.

A continuación, en la siguiente tabla, se listan las historias de usuario y las prioridades definidas.

Título	Descripción	Prioridad
Ver catálogo	Como vendedor quiero poder visualizar un catálogo de libros disponibles para responder rápido la consulta de los clientes.	Media
Registrar ventas	Como vendedor quiero poder registrar ventas para completar el círculo administrativo.	Baja
Mantener listado de libros	Como vendedor deseo mantener actualizada la lista de libros para poder agregar, eliminar y modificar los títulos.	Alta
Clasificar libros por rubro	Como vendedor quiero clasificar libros por rubro para encontrarlos más fácilmente.	Media
Consultar precios y stock	Como cliente quiero consultar precios y stock para comprar más rápidamente.	Media
Reservar libros	Como cliente quiero reservar libros para asegurar la compra.	Baja
Consultar promociones	Como cliente quiero consultar promociones para aprovecharlas y gastar menos dinero.	Media
Publicar datos de la empresa	Como dueño quiero publicar datos básicos de la empresa en la web para posicionar a la empresa.	Alta
Promocionar nuevos libros	Como dueño de la empresa deseo promocionar nuevos libros para fomentar las ventas.	Baja

Comprar por internet	Como cliente quiero comprar por internet para evitarme ir a la librería.	Baja
Recibir opiniones	Como dueño deseo recibir opiniones para adaptarme al mercado.	Baja
Emitir opiniones	Como cliente deseo poder emitir opiniones para compartir mis experiencias y recomendar libros.	Media
Ranking de libros más vendidos (vendedor)	Como vendedor quiero solicitar el ranking de los más vendidos para recomendar a los clientes.	Baja
Ranking de libros más vendidos (cliente)	Como cliente quiero solicitar el ranking de los más vendidos para conocer las tendencias.	Baja
Ranking de libros más vendidos (dueño)	Como dueño quiero solicitar el ranking de los más vendidos para recomendar a los clientes.	Baja
Información de los envíos	Como despachante quiero información de los envíos para cubrir en el menor tiempo posible.	Baja
Información actualizada de los envíos (desp)	Como despachante quiero información actualizada del envío para mantener seguridad de las transacciones.	Baja
Información actualizada de los envíos (cliente)	Como cliente quiero información actualizada de los envíos, para mantener seguridad de la transacción.	Baja
Comparar libros	Como cliente deseo comparar libros de una temática para saber cual comprar.	Baja
Pagar contra reembolso	Como vendedor quiero registrar un pago contra reembolso para mantener circuito completo.	Baja
Definir tipos de usuario	Como dueño deseo definir tipos de usuario para que no todos accedan a la misma información.	Alta
Acceder a descuentos	Como cliente VIP quiero acceder a descuentos para obtener beneficios.	Baja
Definir Promociones	Como Vendedor deseo definir promociones para aumentar y fomentar las ventas.	Media
Pagar con tarjeta	Como cliente quiero pagar con tarjeta de crédito.	Baja
Información de los pagos	Como cliente quiero información de los pagos para cumplir en el tiempo adecuado.	Baja

Tabla 4.4.1: Historias de usuarios iniciales del proyecto de la librería

Ambos grupos decidieron trabajar con números correlativos como identificadores de las tareas.

4.4.2.3 Inicio de Producción LIBRERÍA

Reunión de contextualización Librería - Grupo1

Fecha: 6 de mayo **Duración: 2 horas.**

En una reunión presencial con el equipo, el analista expuso las historias de usuario obtenidas y explicó claramente las 15 primeras. Como plantearon trabajar con iteraciones de una semana, trataron de determinar siguiendo estimaciones por comparación cuantas podrían realizar en la primera iteración, y en principio definieron como meta realizar las tres con prioridad más alta y avanzar si podían con las siguientes en el listado. Además debatieron como podría ser la estructura de datos. El analista ya tenía cargadas las historias de usuario en la pizarra de tareas al momento de iniciar la reunión y había solicitado previamente que cada uno lea las tarjetas definidas.

Reunión de contextualización Librería - Grupo2

Fecha: 6 de mayo **Duración: 1 hora aprox.**

El analista comunicó en reunión presencial al equipo de trabajo lo dialogado en la reunión de planificación inicial. Se confeccionaron las tarjetas de las historias de usuarios definidas y se preparó el Kanban Board. Se debatió sobre la dificultad de implementación de cada una de las tareas solicitadas, a fin de determinar cómo se daría el avance del proyecto.

Reunión de seguimiento - Grupo1

Fecha: 10 de mayo **Duración: 15 minutos**

Reunión virtual vía whatsapp donde cada uno comunicó sobre las actividades realizadas. Se sorprendieron lo rápido que pudieron avanzar, sobre lo habían considerado en la reunión de contextualización.

Reunión de Seguimiento - Grupo2

Fecha: 8 de mayo **Duración: 15 minutos**

Reunión presencial donde cada uno comunicó las actividades realizadas. Se planteó problema sobre la creación de usuario y se buscó solución a la creación de usuarios y roles. Se insistió en aumentar la velocidad de desarrollo.

Ante este reporte se aclaró en la plataforma al equipo cuál era el objetivo de la reunión de seguimiento y que la resolución de problemas no formaba parte de la propia reunión.

Reunión de seguimiento - Grupo1

Fecha: 12 de mayo **Duración: 5 minutos**

Realizada por whatsapp, se explicó lo realizado por cada uno, y lo que cada miembro pensaba realizar a continuación.

Reunión de Seguimiento - Grupo2

Fecha: 11 de mayo **Duración: 5 minutos**

Se debatió sobre las HU desarrolladas y a desarrollar y el desempeño de cada uno en las tareas según el tablero de tareas.

Reunión de reflexión y evaluación interna – Grupo 2

Fecha: 11 de mayo **Duración: 30 minutos**

Se comentó el retraso surgido en el proyecto. Se intercambiaron opiniones sobre los problemas surgidos hasta el momento. El mayor fue el uso de GXServer, en su versión gratuita, que impedía centrarse netamente en el proyecto y, en su lugar, dar prioridad a resolución de cuestiones ajenas. Por tales cuestiones se decidió modificar la manera de trabajar, dejando completamente de lado dicha herramienta. En su lugar se procedió a la exportación e importación de avances mediante archivos .xpz; enviándose oportunamente a través de correos electrónicos.

Se reiteró el compromiso de trabajar responsablemente cada miembro del equipo.

4.4.2.4 Iniciación ágil del proyecto Gimnasio

Similar al caso de la librería, por una cuestión de tiempos del cliente, se trabajó en la iniciación ágil con ambos grupos participando de la misma reunión.

Reunión de inicio ágil

Fecha: 13 de mayo **Duración: 35 minutos.**

En la reunión participaron el cliente, los dos equipos y el secretario del gimnasio y se obtuvo con claridad la *visión común del sistema* a desarrollar para el gimnasio: mayor control de las actividades. Se trabajó con preguntas directas al cliente. En esta reunión también se buscó *conocer a la comunidad*. Aparecieron diferentes tipos de stakeholders: dueño, secretario, cliente, profesor.

La reunión fue más concreta, debido a que los equipos ya tenían experiencia después de haber realizado una reunión similar con el cliente de la librería. Surgieron dos stakeholders cliente y alumno que se mencionaron durante la reunión, pero posteriormente mientras se avanzaba en la actividad de conocer a los vecinos quedo definido que cliente y alumno eran lo mismo y lo llamarían cliente. En otro momento de la reunión alguien del equipo mencionó a un Gerente y quedó aclarado que solo se consideraría a un rol administrativo aparte del dueño que se llamaría secretario.

Reunión de planificación

Fecha: 13 de mayo **Duración: 30 minutos aprox**

Luego de la reunión de iniciación ágil, se quedaron en el lugar de reunión el analista de cada grupo y el cliente, dueño del gimnasio. El cliente manifestó al inicio de esta reunión que no disponía de mucho tiempo.

El cliente enumeró varias funcionalidades que pretendía que tenga el sistema. Los analistas registraron las historias de usuario. Los analistas iban preguntando información complementaria para comenzar también a definir los criterios de aceptación. Se pretendió también cubrir todos los stakeholders para contemplar todas las funcionalidades pero debido a la falta de tiempo del dueño del gimnasio, la reunión se centro básicamente en las funcionalidades relacionadas con el dueño y el secretario. Solo se plantearon unas pocas funcionalidades referidas al cliente y ninguna referida a los profesores. Posteriormente se fue revisando cada historia de usuario con el cliente (cada grupo trabajó las historias de manera separada), y finalmente el cliente priorizó las 7 historias de usuarios que allí surgieron.

Como se mencionó, el haber tenido identificado a todos los stakeholders facilitó la conducción de la reunión, centrándose en un stakeholder por vez y determinando todas las funcionalidades asociadas al mismo. Quedó claro para los analistas que muchas funcionalidades quedaron sin describir por el poco tiempo disponible del cliente.

Al cliente le resultó muy simple poder decidir las prioridades de las historias. El cliente no tomó mucho tiempo, quizás por el apuro de retirarse de la reunión lo más rápido posible.

Una vez retirado el cliente, se recordó a los analistas que ya estaban por comenzar la etapa de producción y que el inicio la marcaba la primera reunión de contextualización del Gimnasio. Se reiteró la necesidad de asegurarse que el resto del equipo se enterara de lo conversado en esta reunión y se hizo hincapié que en debían aclarar bien a los miembros del equipo cada historia de usuario redactada. Además se les hizo notar que desde ese momento comenzaban a trabajar en los dos proyectos de manera simultánea.

A continuación, en la siguiente tabla, se listan las historias de usuario y las prioridades definidas para el Gimnasio.

Título	Descripción	Prioridad
Definir tipos de usuario	Como dueño deseo definir tipos de usuario para que no todos accedan a la misma información.	Alta
Publicar datos de la empresa	Como dueño quiero publicar, en la web, datos básicos de la empresa para posicionar a la empresa.	Alta
Ver clases	Como secretario quiero poder visualizar un listado de clases disponibles para responder rápido la consulta de los clientes	Media
Mantener listado de clases	Como secretario deseo mantener actualizada la lista de clases para poder agregar, eliminar y modificar las características.	Alta
Clasificar clases por tipo de actividad	Como secretario quiero clasificar clases por actividad para encontrarlas más fácilmente.	Media
Consultar precios y cupos	Como cliente quiero consultar precios y cupos para inscribirme más rápidamente.	Media
Realizar inscripción	Como cliente quiero inscribirme a las clases para asegurarme un cupo.	Media

Tabla 4.4.2: Listado de historias de usuario iniciales del proyecto del gimnasio

Ambos grupos decidieron en principio seguir con la identificación numérica de las historias de usuario y las numeraron a partir del número 26, dado que se habían definido 25 historias de usuario de la Librería.

Luego, al crear en el tablero de tareas las historias de usuario, uno de los grupos decidió utilizar colores diferentes para identificar las historias de usuario del Gimnasio respecto de los de la Librería y el otro grupo decidió redefinir los ID de las historias anteponiéndole al número de identificación la inicial L o G dependiendo si se refería a la Librería o al Gimnasio.

4.4.2.5 Inicio de Producción Gimnasio y continuación de Librería

Reunión de evaluación externa y ajuste Librería - Grupo1

Fecha: 13 de mayo **Duración: 40 minutos**

Se realizó la reunión del cliente de librería con el coordinador de grupo. El cliente manifestó estar muy satisfecho con lo desarrollado. No se agregó ninguna historia de usuario con nuevas funcionalidades pero se estableció el cambio de prioridades en alguna de las historias de usuario pendientes.

Reunión de evaluación externa y ajuste Librería - Grupo2

Fecha: 13 de mayo **Duración: 45 minutos**

Reunión con representante de la librería. Se mostró las funcionalidades ya desarrolladas hasta el momento, el cliente se mostró conforme con el producto desarrollado. No se agregó ninguna historia de usuario con nuevas funcionalidades pero sí se solicitó modificaciones a una de las historias implementadas, generándose una nueva historia para este caso. El cliente estableció cambio de prioridades entre algunas historias de usuario pendientes

Reunión de contextualización gimnasio y librería - Grupo1

Fecha: 15 de mayo **Duración: 1 horas**

En una reunión presencial con el equipo, el analista expuso y explicó las historias de usuario obtenidas con el dueño del gimnasio. Se decidió agregar al ID de las historias de usuario una letra G o L para que ayude a identificar las historias de usuario del gimnasio y las de la librería. Se actualizó el Kanban Board con las nuevas historias. Se determinó la prioridad relativa de las nuevas historias respecto de las historias de la Librería. Decidieron o seguir trabajando con iteraciones porque se les complicaba trabajar con los dos proyectos en simultáneo, pero incorporaron dos historias de usuario del Gimnasio con prioridad alta al inicio de la lista de tareas para poder mostrarle al cliente del gimnasio un avance en la reunión de ajuste. Además debatieron como podría ser la estructura de datos.

Además como se había realizado también la reunión de ajustes con el cliente de la librería se ajustaron las prioridades en el tablero y se comentó al equipo la justificación dada por el cliente para dichos cambios. El coordinador comentó además la satisfacción del cliente con lo que ya se había desarrollado y esto motivó al resto del equipo.

Se decidió dar mayor énfasis a las funcionalidades del gimnasio que a los de librería. El analista organizó según esta decisión las tarjetas de ambos sistemas.

Reunión de contextualización gimnasio y librería - Grupo2

Fecha: 14 de mayo **Duración: 45 minutos.**

El analista comunicó en reunión presencial al equipo de trabajo lo dialogado en la reunión de planificación inicial del gimnasio. Se actualizó el Kanban Board con las nuevas historias. Se confeccionaron las tarjetas de las historias de usuarios definidas, utilizando un color diferente para cada proyecto. Se mantuvo una numeración correlativa en los ID de las Historias de usuario. Se debatió sobre la dificultad de implementación de cada una de las tareas solicitadas, a fin de determinar cómo se daría el avance del proyecto.

Además, igual que el grupo 1, como ya se había realizado también la reunión de ajustes con el cliente de la librería se comentó las modificaciones que solicitaba el cliente a una de las historias de usuario ya implementadas por la cual surgió una nueva historia de usuario. Se ajustaron las prioridades en el tablero según lo solicitado por el cliente comentando lo que había aducido el cliente en la reunión de ajuste.

Reunión de seguimiento - Grupo1

Fecha: 18 de mayo **Duración: 5 minutos**

Se planteó lo realizado por cada miembro y taras por realizar. Se plantearon los problemas con GxServer y el abandono de un miembro del equipo y poco involucramiento de otro miembro. Se retomó la idea de programación con Dummies.

Reunión de Seguimiento - Grupo2

Fecha: 18 de mayo **Duración: 5 minutos**

Se comentan las historias implementadas y las próximas a implementar

Reunión de reflexión y evaluación interna - Grupo 1

Fecha: 18 de mayo **Duración: 15-20 minutos**

A continuación de la reunión de seguimiento se realizó esta reunión. Se decidió no utilizar GXServer en su versión gratuita por la pérdida de tiempo que ocasionaba debido a diferentes configuraciones en las máquinas de cada miembro del equipo y problemas en la nube donde se alojaba la KB, se decidió fomentar el uso de programación con Dummies. Se planteó la situación de poco cumplimiento de uno de los integrantes de manera respetuosa. Esta persona explicó lo que le había sucedido, pidió disculpas y se comprometió a cumplir con sus horas de dedicación a la experiencia. Se planteó también el tema del abandono de uno de los integrantes del equipo y cómo afrontarían lo que restaba del proyecto. Decidieron comprometerse más horas para lograr el objetivo. Comentaron que no resultaba tan simple trabajar en dos proyectos simultáneos, pero veían que se lograba avanzar bastante rápido con la implementación. Las pruebas al ser realizadas por otra persona permitían detectar errores y decidieron esforzarse más en la realización de las pruebas. Se decidió también buscar mejoras visuales en los sistemas que guíen mejor al usuario. Se decidió seguir poniendo más énfasis en el gimnasio.

Reunión de seguimiento - Grupo1

Fecha: 20 de mayo **Duración: 5 minutos**

El equipo parece más motivado, hay mayor progreso.

Reunión de Seguimiento - Grupo2

Fecha: 20 de mayo **Duración: 5 minutos**

Reunión de evaluación externa y ajuste Librería - Grupo1

Fecha: 20 de mayo **Duración 45 minutos**

Decide no implementar el pago por tarjeta de crédito y da su aprobación a lo desarrollado. Solicita ajustes finales para cerrar el proyecto, se fija como fecha de entrega final el 1 de junio.

Reunión de evaluación externa y ajuste librería - Grupo2

Fecha: 20 de mayo **Duración: 45 minutos**

El cliente da su aprobación a lo realizado, decide dejar sin implementar el resto de las funcionalidades como el pago por tarjeta de crédito y solicita ajustes finales para cerrar el proyecto, se fija como fecha de entrega final el 1 de junio.

Primera Reunión de evaluación externa y ajuste Gimnasio - Grupo1

Fecha: 20 de mayo **Duración: 20 minutos**

Se realizó la reunión del cliente del gimnasio con el coordinador de grupo. El cliente manifestó estar muy satisfecho con lo desarrollado sobre todo con la interfaz. El cliente manifestó que prefería ahora llamar al cliente, socio o alumno. Se agregaron varias historias de usuario nuevas, no hubo cambio de prioridades. En la siguiente tabla se muestran dichas historias de usuario.

Título	Descripción	Prioridad
Renombrar cliente por alumno o socio	Como dueño quiero llamar a los clientes socios en el sistema, para que no haya confusiones.	Alta
Registrar asistencia (profesor)	Como profesor quiero registrar la asistencia a las clases de clientes para controlar capacidades reales de las clases.	Alta
Registrar asistencia (secretario)	Como secretario quiero registrar la asistencia de los profesores para poder llevar mejor control	Media
Mantener listado de salones	Como secretario quiero administrar las clases por salones para maximizar su uso	Media
Registrar ventas de extras	Como secretario deseo registrar las ventas de productos complementarios para poder tener un control	Media
Manejo de caja chica	Como secretario deseo poder realizar un manejo de caja chica para tener mejor control del dinero a rendir al final del día	Bajo
Obtener información por profesor	Como dueño deseo poder ver datos relacionados a los profesores, sus clases, cantidad de alumnos, tipo de clases ofrecidas para conocer quienes aportan más al gimnasio.	Bajo
Ofrecer promociones	Como dueño deseo poder administrar promociones para captar mayor cantidad de clientes, en épocas críticas	Medio
Administrar un sitio de novedades	Como dueño deseo difundir promociones, clases gratuitas o eventos a los clientes para lograr captar nuevos clientes	Bajo

Tabla 4.4.3: Listado de Historias de usuario del proyecto del gimnasio nuevas

Primera Reunión de evaluación externa y ajuste Gimnasio- Grupo2

Fecha: 20 de mayo **Duración: 25 minutos**

Se realizó la reunión del cliente del gimnasio con el coordinador de grupo. El cliente solicitó cambios en la interfaz que se tradujo en una nueva historia de usuario. Se agregaron varias historias de usuario nuevas que son las mismas detalladas para el Grupo 1.

Reunión de contextualización gimnasio y librería - Grupo1

Fecha: 21 de mayo **Duración: 45 minutos**

Respecto a la librería: Comunica los ajustes finales a realizar al proyecto y comunica la fecha de entrega final: el 1 de junio.

Respecto al Gimnasio: Se comunicó al resto del equipo los comentarios positivos del cliente del gimnasio sobre todo de la interfaz. Se comentaron las nuevas historias de usuario agregada a la pizarra de tareas y se aclararon las dudas surgidas, sobre todo con la historia de usuario de *manejo de caja chica*.

Reunión de contextualización gimnasio y librería - Grupo2

Fecha: 22 de mayo **Duración: 45 minutos.**

Respecto a la librería: Comunica los ajustes finales a realizar al proyecto y comunica la fecha de entrega final: el 1 de junio.

Respecto al Gimnasio Se informó los comentarios del cliente. Se aclaró lo que el cliente esperaba de la interfaz del gimnasio y se expuso las historias a agregar al listado de tareas de la pizarra, tanto las que corresponden a nuevas funcionalidades como la que corresponde a las modificaciones de la interfaz.

Reunión de seguimiento - Grupo1

Fecha: 27 de mayo **Duración: 5 minutos**

Ninguno de los miembros del equipo trabajó en los proyectos por dedicarse a estudiar para el turno de examen extraordinario del día 26 de mayo.

Reunión de Seguimiento - Grupo2

Fecha: 26 de mayo **Duración: 10 minutos**

Cada uno expuso el trabajo realizado. Muchos no dedicaron el tiempo estipulado por prepararse para rendir en el turno de examen extraordinario de Mayo.

Reunión de reflexión y evaluación interna - Grupo 1

Fecha: 27 de mayo **Duración: 20 minutos**

Se analizó la falta de tiempo dedicada al proyecto y esta se debió al tiempo dedicado a la preparación de los exámenes del turno extraordinario. Todos se comprometieron a avanzar lo máximo posible para mostrar al cliente del gimnasio y poder cerrar en el menor tiempo posible el sistema de la librería.

Reunión de seguimiento - Grupo1

Fecha: 29 de mayo **Duración: 5 minutos**

Cada miembro del grupo de desarrollo comunica lo que hizo y está por realizar. No hay inconvenientes

Reunión de Seguimiento - Grupo2

Fecha: 29 de mayo **Duración: 5 minutos**

Se reúnen los miembros del equipo e informan lo realizado y lo que realizarán. No hay problemas detectados

Reunión de reflexión y evaluación interna - Grupo 2

Fecha: 29 de mayo **Duración: 30 minutos**

Se comentó con satisfacción los logros alcanzados, aún trabajando con menos personas que el otro equipo. Dentro del grupo algunos resaltaron el compromiso de cada miembro del equipo. Se alentó a seguir trabajando de igual manera.

Reunión de evaluación externa y ajuste Gimnasio - Grupo1

Fecha: 1 de junio **Duración: 15 minutos**

El cliente se reúne con el analista. El cliente realiza observaciones menores a una de las historias implementadas generando una nueva historia y decide dejar fuera del proyecto el manejo de caja chica.

Reunión de ajuste y evaluación externa y ajuste Gimnasio - Grupo2

Fecha: 1 junio **Duración: 45 minutos**

El cliente se reúne con el analista. El cliente realiza observaciones menores a dos de las historias implementadas y decide dejar fuera del proyecto el manejo de caja chica.

4.4.2.6 Ritual de finalización de la librería

Reunión de entrega final de la librería - Grupo 1

Fecha: 1 de junio **Duración: 30 minutos**

El analista se reunió con el cliente de la librería, mostraron la versión final que se encontraba ya instalada en la web. El analista fue mostrando rápidamente las diferentes vistas del sitio según el usuario y de esta forma el cliente vio un pantallazo general de todo lo desarrollado y estuvo de acuerdo en que cumplía con lo solicitado.

Reunión de reflexión final de la librería - Grupo1

Fecha: 1 de junio **Duración: 35 minutos**

Reunión del equipo junto con el cliente. No pudo asistir ningún stakeholder. Juntos valoraron lo obtenido. Al cliente, debido a que trabajó con dos equipos de desarrollo le resultó demasiado demandante de tiempo las reuniones con ambos equipos. Los productos obtenidos satisficieron sus necesidades. Remarcó que se sorprendió de lo rápido del avance del desarrollo en pocos días, y el resultado final en casi un mes de trabajo. Destacó de este equipo la rapidez con que realizaron las primeras implementaciones, y la facilidad con que entendieron sus requerimientos. En este equipo inclusive se llegaron a implementar todas las funcionalidades con prioridad baja que el grupo 2 no consiguió hacerlo en el tiempo dado.

Reunión de entrega final de la librería - Grupo 2

Fecha: 1 de junio **Duración: 25 minutos**

El analista se reunió con el cliente de la librería, presentó la versión final ya instalada en la web y permitió que el cliente dé un vistazo general a todo lo desarrollado. Ayudó a ingresar a las distintas vistas de usuario. El cliente estuvo de acuerdo en que cumplía con lo solicitado.

Reunión de reflexión final de la librería - Grupo 2

Fecha: 1 de junio **Duración: 30 minutos**

Reunión del equipo junto con el cliente. No pudo asistir ningún stakeholder. Juntos valoraron lo obtenido. Al cliente, debido a que trabajó con dos equipos de desarrollo le resultó demasiado demandante de tiempo las reuniones con ambos equipos. Los productos obtenidos satisficieron sus necesidades. Destacó que teniendo conocimiento del abandono de uno de los miembros del equipo, el equipo logró producir un producto con todo lo que él había indicado que era importante en casi un mes de trabajo. El cliente eligió el producto desarrollado por el grupo 2 para utilizar en su empresa pero quisiera agregarle primero las otras funcionalidades menores que logró implementar el grupo 1. Todavía no lo tiene en uso.

En esta etapa vale la pena realizar la siguiente aclaración. Se cerró el trabajo con el proyecto de la librería y los dos grupos implementaron diferente cantidad de historias, quedando algunas sin implementar en el grupo 2, pero estas eran las menos importantes para el cliente. El grupo 1 cerró el proyecto habiendo cumplido todas las historias salvo la que el mismo cliente decidió dejar de lado en reunión previa y el grupo 2, quedándole algunas pendientes. En el Anexo 4 se adjuntan algunas capturas de pantallas del sistema de la librería desarrollado por cada grupo. Esto muestra que si se fija en determinado tiempo del proyecto el cierre del mismo, siempre se logra entregarle al cliente un producto que satisface sus necesidades prioritarias. Inclusive se vio en esta reunión un conocimiento total por parte del cliente de los dos alumnos que dejaron el grupo 2. Esto también permite ver que se confirma un vínculo entre cliente y equipo de tantas reuniones realizadas, si bien no es cuestión menor la manifestación del cliente de la gran cantidad de tiempo dedicada a las reuniones.

4.4.2.7 Continuación de Producción Gimnasio

Reunión de contextualización gimnasio - Grupo1

Fecha: 3 junio **Duración: 45 minutos**

El analista explica las modificaciones a realizar y saca de la pizarra la historia relacionada con Caja chica

Reunión de contextualización gimnasio - Grupo2

Fecha: 3 junio **Duración: 45 minutos.**

El analista explica las modificaciones a realizar y saca de la pizarra la historia relacionada con Caja chica

Reunión de seguimiento - Grupo1

Fecha: 5 junio **Duración: 5 minutos**

Cada miembro explica las tareas realizadas, todas las historias fueron terminadas

Reunión de Seguimiento - Grupo2

Fecha: 6 junio **Duración: 5 minutos**

Cada miembro explica las tareas realizadas, no hay ningún inconveniente. Solo resta completar la prueba de la última historia y su puesta en producción. Los integrantes están contentos por estar llegado a buenos resultados.

Reunión de Seguimiento - Grupo2

Fecha: 8 junio **Duración: 5 minutos**

Todas las historias fueron implementadas

4.4.2.8 Ritual de finalización del gimnasio

Reunión de entrega final del gimnasio - Grupo 1

Fecha: 10 de junio **Duración: 20 minutos**

El analista se reunió con el cliente del gimnasio, manifestó ya haber ingresado al sistema desde su casa y que su secretario también había estado revisando todo el sitio. El cliente aceptó que todo estaba implementado como él había solicitado

Reunión de reflexión final del gimnasio - Grupo 1

Fecha: 10 de junio **Duración: 25 minutos**

Reunión de todo el equipo junto con el cliente. No pudo asistir ningún stakeholder. El cliente informó que realmente el secretario se lamentaba que se hubiera sacado del listado el manejo de caja chica, ya que era algo muy importante para él.

El cliente también apreció la forma rápida con que se iba desarrollando el sistema, la posibilidad de no definir todo lo que necesitaba en una sola reunión y la buena predisposición para responder a los pequeños cambios y nuevos requerimientos.

El equipo valoró la buena predisposición del cliente en el proyecto. El equipo valoró la buena predisposición del cliente en el proyecto y lamentó no contar con la opinión directa de las otras personas que usarían el sistema. Pero se agradeció el haber compartido la opinión del secretario.

El equipo posteriormente, sin la presencia del cliente, realizó su reflexión final, lamentando los problemas ocasionados por haber trabajado con una versión trial y el abandono de un miembro del grupo pero valorando los resultados obtenidos a pesar de las dificultades y rescatando lo importante de lograr tener participación del cliente.

Reunión de entrega final del gimnasio - Grupo 2

Fecha: 10 junio **Duración: 20 minutos**

El analista se reunió con el cliente del gimnasio. También manifestó ya haber ingresado al sistema desde su casa y que su secretario también había estado revisando todo el sitio. Aceptó todas las funcionalidades implementadas y elogió la interfaz.

Reunión de reflexión final del gimnasio - Grupo 2

Fecha: 10 de junio **Duración: 20 minutos**

Reunión de todo el equipo junto con el cliente. No pudo asistir ningún stakeholder. El cliente informó aquí también que realmente el secretario se lamentaba que se hubiera sacado del listado el manejo de caja chica, ya que era algo muy importante para él. El cliente también apreció la forma rápida con que se iba desarrollando el sistema, la posibilidad de no definir todo lo que necesitaba en una sola reunión y la buena predisposición para responder a los pequeños cambios y nuevos requerimientos, igual que en el caso del otro grupo.

El equipo valoró la buena predisposición del cliente en el proyecto. Cuando éste se retiró reflexionaron sobre la experiencia y estuvieron conformes en términos generales con lo desarrollado, y con el conocimiento adquirido.

En el Anexo 4 se adjuntan algunas capturas de pantallas del sistema del gimnasio desarrollado por cada grupo.

4.4.3 Comentarios a cerca de la experiencia

A lo largo de la experiencia se buscó respetar todas y cada una de las etapas y reuniones propuestas en esta tesis y también cubrir por lo menos los roles esenciales. Al principio los miembros del equipo estaban temerosos de participar, por ser una forma diferente a como realizaron todas sus actividades de desarrollo a lo largo de la carrera. Poco a poco las reuniones se hicieron más fluidas y cada uno de los integrantes del grupo participaba libremente expresando sus opiniones y estas eran respetadas por el resto.

En toda la experiencia se buscó tener un rol muy pasivo, como espectador, para simplemente recabar información y aclarar dudas o aconsejar en situaciones especiales.

A continuación se analizará la experiencia teniendo en cuenta los siguientes puntos:

- Auto-organización del grupo
- Reuniones
- Etapas: Inicio ágil, Producción, Ritual de finalización
- Herramientas
- Comentarios generales de los participantes

Posteriormente se expondrán las conclusiones finales teniendo en cuenta los puntos mencionados anteriormente y se comentará las opiniones recibidas de una de las empresas consultadas.

4.4.3.1 Autoorganización del grupo

Como se mencionó anteriormente, antes de iniciar específicamente la experiencia, se solicitó a los grupos organizarse entre sí y definir las forma en que iban a trabajar, los roles que desempeñarían cada uno, los lineamientos y estándares a seguir, las herramientas a usar y la forma de estructurar la pizarra de tareas definiendo las actividades asociadas a cada uno y posteriormente debían definir el DoD de cada actividad. Todo esto se describió en la plataforma y posteriormente en la encuesta final se volvió a preguntar a cada uno de los participantes sobre estos puntos.

4.4.3.2 Los estándares y lineamientos del grupo

Más que estándares los alumnos reflejaron los lineamientos definidos para trabajar, sobre todo el comprometerse y trabajar dando el máximo posible en los tiempos dedicados a la experiencia y siendo solidarios con el resto del grupo.

Dentro del desarrollo, quedó claro que cada uno debía probar completamente lo desarrollado antes de que pueda pasar a Testing, donde era realizado por otro integrante diferente.

4.4.3.3 Definición de actividades dentro de la producción

Ambos equipos decidieron trabajar con Kanbanize para implementar la pizarra Kanban y, en términos generales, ambos equipos trabajaron con las siguientes etapas: Análisis, Desarrollo, Prueba y Producción. Sin embargo, uno de los grupos detalló un poco más las actividades dentro del desarrollo especificando: Estructuras de datos, diseño, altas bajas y modificaciones de transacciones, reportes.

Según la encuesta, la complejidad de especificar la definición de terminado (*Definition of Done* - DoD) de cada etapa o actividad de producción fue simple para un 67% y tuvo una complejidad media para el 33% restante (Ver Figura 4.4.4). Responde a lo mencionado en la propuesta, respecto a que no es una actividad compleja de realizar.

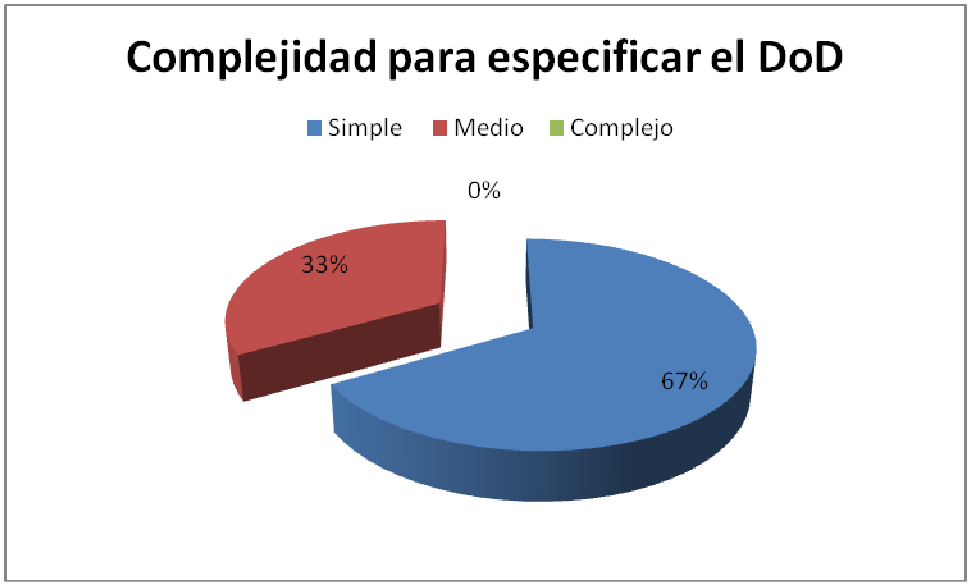


Figura 4.4.4: Complejidad para especificar DoD

Respecto a la importancia de tener una buena especificación del DoD, solo un 33% consideró que era realmente importante (Ver Figura 4.4.5). Esto hace cuestionar, no la importancia en sí del DoD, ya que es indiscutible que es realmente prioritario tenerlo perfectamente especificado para cada etapa dentro de la sección de producción, sino que tal vez tenían el conocimiento tácito de lo que involucraba y por eso lo consideraron como algo trivial. Por otro lado como se les exigió la definición del DoD para cada etapa planteada en la pizarra Kanban al inicio de la experiencia, no surgieron malos entendidos a los largo del proyecto respecto hasta donde involucraba cada etapa y cuando considerarla terminada. Si se hubiera sugerido simplemente su uso, posiblemente hubieran aparecido situaciones que les habría permitido valorar su importancia al cien por ciento de los participantes.

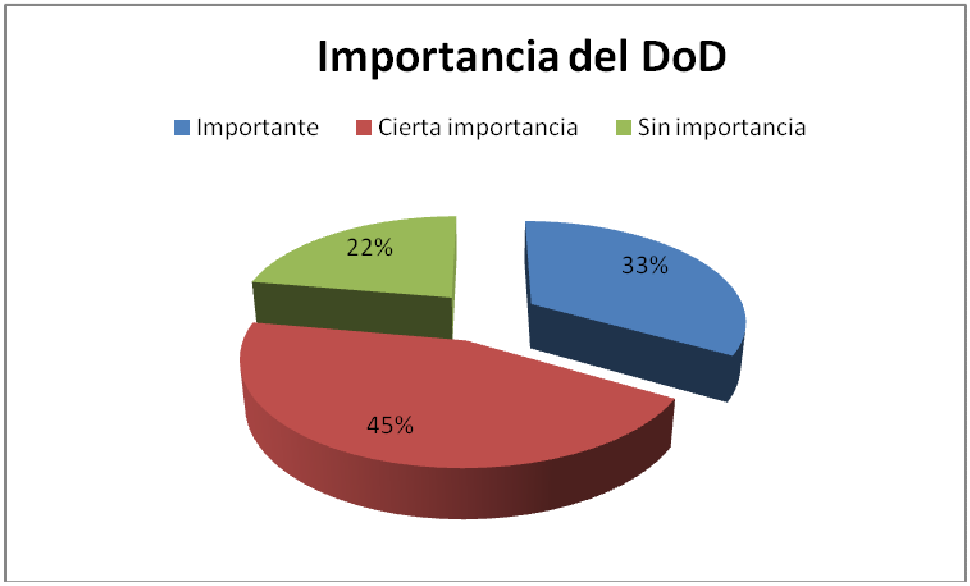


Figura 4.4.5: Importancia del DoD

4.4.3.4 Respecto a la organización y adaptación de la forma de trabajo

Ambos equipos realizaron adaptaciones en la forma de trabajo a lo largo de la experiencia. Todos admitieron haber realizado modificaciones a la manera que habían determinado trabajar, ya sea por cambiar herramientas (dejar de usar GXServer y fomentar programación con Dummies), modificar lineamientos (como establecer ciertos horarios para que todos estén conectados para trabajar en conjunto).

Ambos equipos reconocieron que es importante poder tener un equipo autoorganizado.

4.4.3.5 Roles propuestos

La definición de roles realizada en esta propuesta permitió que todos entendieran perfectamente cada uno de los roles que podían desempeñar. Los roles Analista, Coordinador, Diseñador-Programador, Diseñador Líder, Tester, Cliente y Stakeholder tuvieron lugar en ambos equipos. Ninguno utilizó el rol de Especialista.

En ambos casos el analista tomó el rol de coordinador. Aquellas que trabajaron como coordinador también consideraron que las tareas que este debía realizar estaban bien especificadas y era simple de entender pero no tan simple de ejecutar.

Por otro lado en uno de los equipos el analista y coordinador también ejerció el rol de diseñador líder mientras que en el otro equipo fue desempeñado por una persona diferente al analista / coordinador.

4.4.3.6 Las reuniones

Se realizaron todas las reuniones en ambos equipos, si bien algunos admitieron haber cambiado horario o día de algunas reuniones por situaciones específicas. No hubo uniformidad a la hora de comentar cuál consideraron que fue la reunión más importante. A continuación se muestra el gráfico que muestra los porcentajes de las consideradas más importantes (Ver Figura 4.4.6).

Figura 4.4.6: Reunión más importante

La variedad de respuestas puede interpretarse que todas están casi al mismo nivel de importancia y es difícil elegir una de ellas. Relacionado a esta idea es que ninguno de los alumnos consideró que había alguna reunión que podía obviarse, salvo un alumno que consideró que las reuniones de

seguimiento no eran necesarias hacerlas diariamente – tal cual se propone en este trabajo. Un análisis más detallado de cada tipo de reunión planteada en la porpuesta se realizará en el siguiente punto al analizar las Etapas planteadas en la propuesta. Pero, vale la pena volver a recalcar, que ninguno consideró que alguno de los tipos de reuniones planteadas fuera inecesaria.

- **Participación de los clientes en las reuniones**

Las reuniones con los clientes se realizaban para cada grupo el mismo día, una después de la otra. Lo que se buscó desde la cátedra fue alternar el grupo que tenía la reunión en primer lugar.

4.4.3.7 Los clientes y sus tiempos

A los clientes les resultó pesado trabajar con dos equipos de desarrollo que solicitaban sus reuniones. Si bien se coordinaban desde la cátedra para que sean el mismo día y una después de la otra, los clientes no suelen tener tiempo disponible y menos, disposición de tiempo para ir hasta el lugar de reunión que en este caso era el campus universitario. Debido a que ninguno de los clientes pagó por el desarrollo realizado, es que se tomaron el trabajo de cumplir con las reuniones. Actualmente los clientes deben optar por uno de los dos sistemas desarrollados para utilizar.

En las encuestas realizadas el cien por ciento consideró que la participación tanto del cliente del gimnasio como el de la librería fue algo activa. Todos reconocieron que el cliente del gimnasio colaboró mucho con el desarrollo no solo en las reuniones, sino contestando mails, consultas personales, participación por la plataforma en foros, y realizaba sugerencias y recomendaciones. Si bien para el cliente de la librería el 88.89 por ciento consideró que colaboró de la misma forma que el anterior, un 11.11 por ciento consideró que su participación se redujo a estar en las reuniones y para él eso no mostraba colaboración con el equipo. Pero tanto para el cliente del gimnasio como para el de la librería hay unanimidad para considerar que no fue necesaria mayor participación de ninguno de ellos.

Fue más difícil todavía contar con la presencia de stakeholders en las reuniones iniciales y finales, si bien es más importante incluirlo en el inicio ágil, es importante su opinión al final de proyecto. Esto debido a sus obligaciones y a la necesidad de trasladarse al campus universitario. Un 55.56% de los alumnos que completaron la encuesta al final, consideraron que hubiera sido más productivo tener a los stakeholders de los sistemas en la revisión final y para el 44.44% restante que no lo hubiera sido. Las razones aducidas por los primeros fueron en términos generales que permite conocer si todos están satisfechos y cada stakeholder aporta una visión distinta del producto final. Para el resto, o bien les daba lo mismo o consideraban que su opinión no influiría en cuestiones del grupo de desarrollo.

4.4.3.8 Etapas: Inicio ágil, producción, ritual de finalización

4.4.3.8.1 INICIACIÓN ÁGIL

En la propuesta, se mencionó como principal objetivo de la iniciación ágil *construir una visión completa sobre el concepto de producto y que además no caiga en sesgos personales, es decir, que esa visión sea compartida y comprendida de idéntica forma por los principales interesados*. El propósito de esta etapa es alcanzar un acuerdo entre todos los stakeholders sobre los objetivos del proyecto.

Para lograr esto se planteó en la propuesta realizar dos reuniones: **una reunión de inicio ágil** y **una reunión de planificación inicial**. Se pretendía como resultado final no solo haber logrado construir una visión compartida del proyecto, sino además tener un número considerable de historias de usuario priorizadas, listas para comenzar con el proceso de producción.

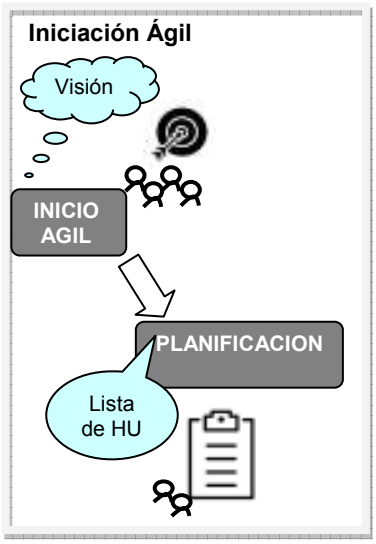


Figura 4.4.7: Etapa de iniciación ágil

Reunión de inicio ágil: según la propuesta plateaba que todo el equipo debía participar y fundamentalmente los stakeholders. Se proponían dos técnicas: Definir una visión y Conocer a la comunidad:

- *Definir una visión común:* cuyo fin era lograr que todos estuvieran de acuerdo en qué se quería construir, y también el porqué.

Por lo que se pudo observar en la reunión de inicio ágil realizada para el proyecto de la librería, los miembros de ambos grupos consideraban este paso como algo muy trivial y cuya respuesta era obvia. Pero, la realizaron, porque debían seguir la consigna que se les había indicado para la reunión.

Mientras avanzaban con el desarrollo de la técnica durante la reunión, quedó en evidencia que las visiones eran bastante divergentes entre los presentes. Para dar un ejemplo, para algunos miembros del equipo de desarrollo la razón de realizar este sistema era para mejorar el manejo de las ventas y el control de stock, para el stakeholder era tener un mejor control de stock y poder registrar y realizar seguimiento de reservas de libros, pero para el cliente la razón por la que más le interesaba el sistema era para aumentar los ingresos, mediante la captación de mayor cantidad de clientes. Esto fue una sorpresa para varios y reamente admitieron verbalmente que sin esta actividad nunca hubieran comprendido la visión del cliente.

Al realizar esta actividad para el proyecto del gimnasio, ya los grupos habían valorado en su gran mayoría está técnica.

Según la encuesta realizada al final de la experiencia, la técnica propuesta para lograr definir la visión fue considerada simple para el 78 por ciento, poco compleja para un 11 por ciento y para el 11 por ciento restante compleja (Ver Figura 4.4.8). Por otro lado, para el 67 por ciento no se hubiera logrado tener una visión común si no se hubiera plateado está técnica en la reunión (Ver Figura 4.4.9). Según estas opiniones y lo que se pudo observar en la experiencia realizada, en líneas generales, podría considerarse aplicable esta técnica con buenos resultados.

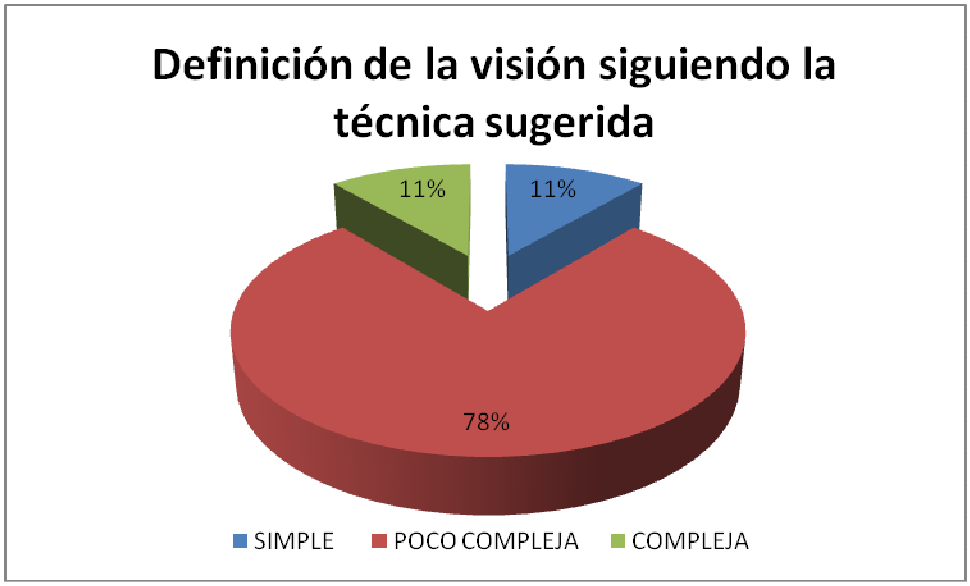


Figura 4.4.8: Complejidad para definir la visión siguiendo la técnica sugerida

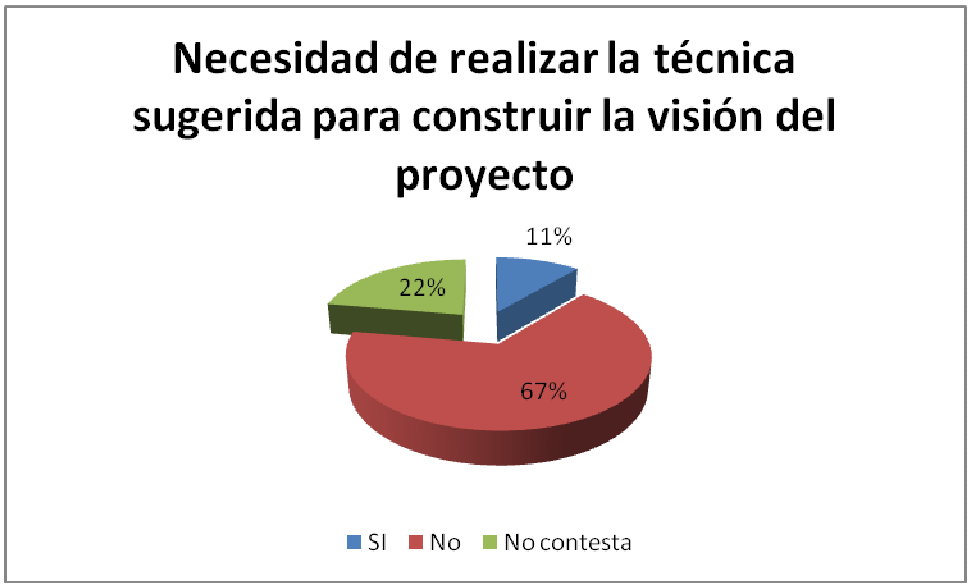


Figura 4.4.9: Necesidad de la técnica de Definir la visión común para construir la visión del proyecto

- *Conocer a la comunidad:* su finalidad era identificar a todos los stakeholders involucraos y la visión de cada uno.

Dentro de la reunión de inicio ágil se realizó para cada proyecto la técnica de identificar a los vecinos. Se detectaron dos situaciones diferentes que hacen válida la propuesta de realizar esta técnica, para comenzar un proyecto sin malas interpretaciones

En el proyecto de la librería, al realizar esta técnica aparecieron diferentes tipos de stakeholders: dueño, vendedor, despachante, cliente virtual, cliente real, cliente preferencial, editorial. Se pudo observar que gracias a la actividad de conocer a la comunidad aparecieron tres tipos de

stakeholders que, en un principio, el dueño de la librería, no había hecho mención: despachante, cliente virtual y cliente preferencial. Luego definiendo la visión de cada uno, se fue comprendiendo el funcionamiento general de la librería y la relación de cada stakeholder con el sistema. De no haberse aplicado la técnica, habrían aparecido más adelante en el proyecto tal vez ocasionando ciertas complicaciones, como por ejemplo que el cliente se olvide en la siguiente reunión de describir historias de usuario relacionadas con el despachante.

Cuando se aplicó la técnica para el proyecto del gimnasio aparecieron también diferentes tipos de stakeholders: dueño, secretario, cliente, profesor, alumno. Se pudo observar que mientras se avanzaba en esta técnica resultó evidente que cliente y alumno eran lo mismo y se estipuló que se llamaría cliente. En otro momento de la reunión alguien del equipo mencionó a un Gerente y, después de un debate, quedó aclarado que solo se consideraría a un rol administrativo aparte del dueño que se llamaría secretario. Nuevamente seguir esta técnica evitó malos entendidos sobre todo con historias de usuario que posteriormente pudieran haber surgido para dos stakeholders en principio diferentes: Clientes y Alumnos.

La técnica propuesta resultó fácil al 78% de los participantes de la experiencia dentro del equipo de desarrollo (Ver Figura 4.4.10). Para un 22 por ciento tuvo ciertas complejidades pero ninguno consideró que la técnica planteada en la propuesta era compleja de aplicar.

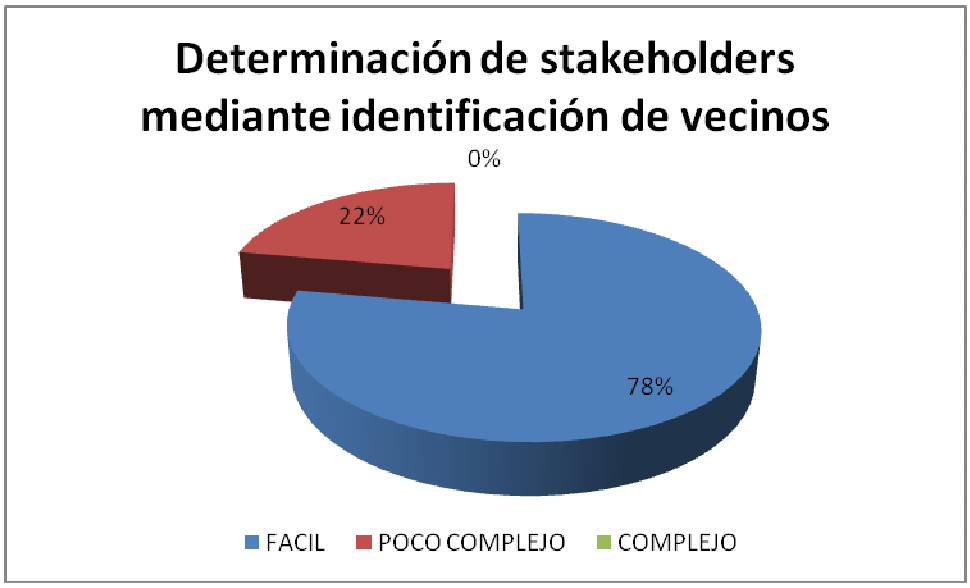


Figura 4.4.10: Necesidad de la técnica de definir la visión común para construir la visión del proyecto

Resumiendo, a través de esta reunión se logró realmente definir un rumbo común de todos los involucrados en el desarrollo, permitiendo clarificar ciertos aspectos que podrían haberse interpretado posteriormente de manera diferente. Teniendo en cuenta los resultados obtenidos, se puede considerar que esta técnica también cumple con los objetivos con los que fue planteada y es simple de aplicar.

La propuesta planteaba las dos técnicas mencionadas anteriormente y debían realizarse con todo el equipo de desarrollo, el cliente y los stakeholders justamente para asegurar contemplar todas las visiones de los posibles involucrados. Según la encuesta final, el 67 por ciento está de acuerdo en la necesidad de participación de todos los involucrados en esta primera reunión. Algunas de las razones citadas son:

- para que todos conozcan el objetivo del proyecto, las funcionalidades requeridas.

- porque ayuda a todo el equipo a conocer, por parte del cliente, el negocio y ayuda al analista a entender mejor lo que se quiere.
- porque todos deben conocer el sistema a desarrollar, aportar a ello y aportar diferentes puntos de vista.
- porque es distinto si solo participa uno solo del grupo porque se puede perder algún detalle de lo que quiere el cliente.
- porque se debe tener claro cuál es el objetivo del proyecto y sus prioridades.

En síntesis, este 67 por ciento logró comprender cuál era el verdadero valor para esta reunión de inicio ágil (Ver Figura 4.4.11). El 11 por ciento lo consideró inútil justificando que es el analista quien luego describe al grupo las historias de usuario y termina ordenándolas por prioridades, este grupo de participantes no valoró la importancia que todo el equipo de desarrollo tenga en claro la visión, tal vez por tener una visión más rígida de un analista que actúa más bien como líder de proyecto que ordena lo que debe realizar cada uno. Esta misma razón fue aducida por una de las personas que consideró que era algo útil, la otra persona consideró que tiene cierto grado de aporte pero no demasiado.

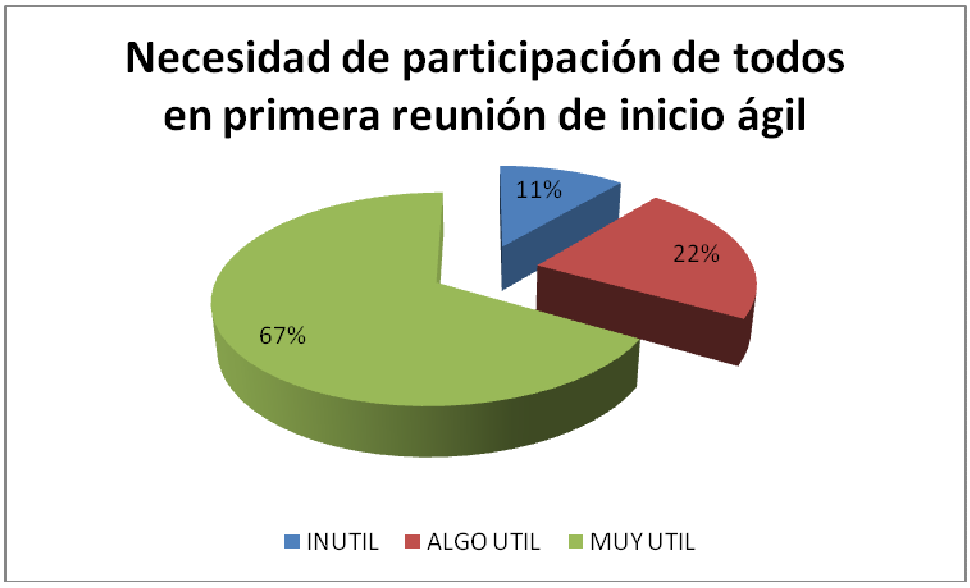


Figura 4.4.11: Necesidad de participación de todos en primera reunión

Vale la pena recordar lo que se mencionó anteriormente, para un 22% esta fue la reunión más importante (Ver Figura 4.4.6).

En base a la experiencia, se considera que la reunión de inicio ágil logró realmente introducir en el tema del nuevo proyecto a todos los involucrados. De esta manera, es el equipo completo quien construye la visión y ya no es el líder de proyecto quien transmite la visión al equipo. Se favorece trabajando de este modo la comunicación directa enter todos, evitando malas interpretaciones. Es el equipo completo que comienza a avanzar en el desarrollo de un nuevo proyecto de manera conjunta.

Reunión de planificación: según la propuesta se planteaba que solo participaba el analista y el cliente. No se especificó una técnica mas bien teniendo en cuenta los diferentes stakeholders y la visión previamente definida, se pretendía que entre ambos fueran construyendo la lista de historias de usuario para luego priorizarla.

Si bien se sugería utilizar prototipación rápida aprovechando las carácteristicas de DBC con Genexus para asegurar entender los requerimientos, ninguno de los grupos la utilizó. Ayudó mucho, por lo que se pudo observar, tener identificados los stakeholders porque los analistas fueron haciendo preguntas sobre stakeholders que no tenian ninguna historia asociada y lograron que el cliente tenga en cuenta.

Durante la reunión los analistas fueron los que escribieron las historias de usuarios con las funcionalidades que iban surgiendo (cada grupo redactó las historias de manera separada). Ellos solicitaban información complementaria para comenzar también a definir los criterios de aceptación. No fue necesario utilizar un prototipo para las funcionalidades expresadas. Posteriormente el cliente junto con cada analista fue revisando cada historia de usuario y finalmente el cliente las priorizó.

El haber tenido definido los diferentes stakeholders permitió a los analistas hacerle preguntas al cliente, dueño de la librería, sobre las funciones de cada uno, qué tareas podría realizar interactuando con el sistema. Sobre todo se vio la utilidad en la definición de las actividades relacionadas con el despacho de pedidos de libros, que no habían sido consideradas en profundidad. Ante la pregunta puntual de uno de los analistas quedó completamente claro este circuito y las funcionalidades requeridas. Lo mismo surgió con el trato preferencial a ciertos clientes de la librería.

La priorización de las historias no fue sencilla para ninguno de los clientes. Pero los analistas recordándole la visión previamente definida, ayudaron a realizar esta tarea, como se pudo observar durante la realización de las mismas.

Para el 11% de los participantes en la experiencia esta fue la reunión más importante (Ver Figura 4.4.6). Analizando los resultados de la experiencia, esta reunión cumplió con los objetivos propuestos y fue completamente viable, obteniendo como resultado un listado priorizado de las historias de usuario para la siguiente etapa.

4.4.3.8.2 ETAPA DE PRODUCCIÓN

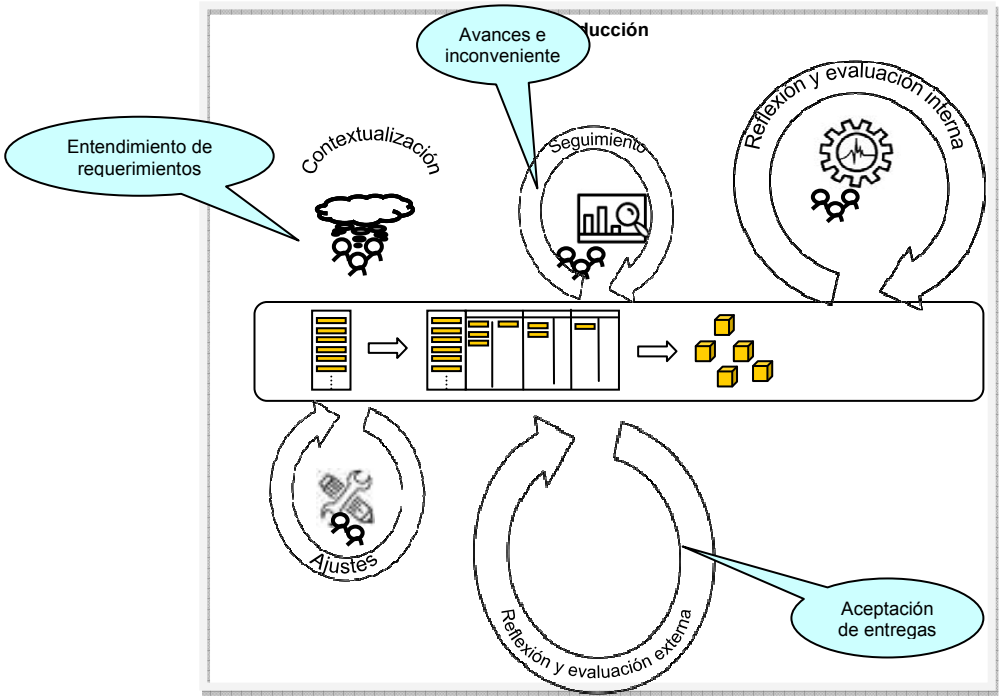


Figura 4.4.12: Etapa de producción

La propuesta planteaba trabajar en la etapa de producción a través de un flujo de trabajo contante compartido para todos los proyectos que se estén desarrollando en simultáneo. También se aclaraba en la misma que, si **el equipo quisiera trabajar con iteraciones, podría definirlas** sin ningún inconveniente. Uno de los equipos hizo esto en cierta medida, definiendo iteraciones de 2 semanas.

Como herramienta fundamental en esta etapa se enunció la necesidad de utilizar una pizarra Kanban, donde se reflejen las etapas que el equipo determinó considerar en el desarrollo propiamente dicho incluyendo su definición de terminado (DoD). En la experiencia, en una reunión de auto-organización previo al inicio de los proyectos se solicitó a los dos grupos que establezcan estas consideraciones consensuando entre todos, tal como se comentó anteriormente.

En esta etapa se plantearon cinco reuniones: Una reunión de contextualización, Una reunión de seguimiento, Una reunión de evaluación y reflexión interna, Una reunión de evaluación externa y Una reunión de ajuste. Cada una de estas reuniones surgía de una necesidad de comunicación específica.

A continuación se tratará de expresar cómo funcionó cada una de las reuniones en la experiencia, y determinar si colaboraron a los objetivos por los que fueron establecidas en la propuesta.

Reunión de contextualización, el objetivo según la propuesta, era lograr un entendimiento común de lo que el cliente está necesitando y que se encuentra descrito en cada Historia de Usuario que se va incorporando al proyecto. Participaba el analista con todo el equipo de desarrollo.

A lo largo de la experiencia cada grupo realizó una reunión de contextualización después de una reunión de planificación inicial para describir las historias que surgieron y explicar un poco la justificación de las prioridades de cada una. Uno de los grupos aprovechó estas reuniones para discutir aspectos de diseño y esbozar un diagrama de clases para generar consenso en la forma de trabajo del equipo.

También se realizaron estas reuniones cada vez que el cliente expresaba nuevas historias de usuario, las modificaba o cambiaba prioridad, es decir después de las reuniones de ajuste. Tal como lo refleja la Figura 4.4.6, para un 11 por ciento está fue la reunión más importante.

Realizando estas reuniones todo el equipo logró tener en claro que es lo que se pretendía alcanzar con las historias que se incorporaban o modificaban.

Reunión de seguimiento: el objetivo buscado a través de esta reunión fue mantener a todo el equipo informado sobre el avance del proyecto y sus inconvenientes. Se proponía realizarla dos veces por semana.

Se pudo observar por los reportes realizados en la plataforma que a veces realizaron algunas reuniones de manera virtual, pero que si bien es una posibilidad todos concuerdan en que el diálogo cara a cara es mejor. En el grupo donde hubo deserción de uno de los miembros, también sirvió de motivación esta reunión. Para un 34% de los participantes de la experienca (Ver Figura 4.4.6),esta fue la reunión más importante. Las razones aducidas principalente fueron que permite aclarar dudas, plantear problemas, saber en qué se está trabajando.

Reunión de evaluación y reflexión interna: se buscaba que el equipo de desarrollo pueda reflexionar cada cierto tiempo para evaluar el proceso, y proponer mejoras.

Ambos equipos realizaron dos reuniones de evaluación y reflexión interna durante la experiencia. Coincidieron siempre en su realización luego de una reunión de seguimiento, pero en diferentes fechas.

Analizando los resúmenes descriptos en la plataforma, en su primera reunión de evaluación y reflexión interna, el grupo 1 planteó varios temas a destacar:

- Se decidió no utilizar GXSERVER por los problemas y demoras ocasionados y adaptaron la forma de trabajo.
- Se plantearon dos situaciones que afectaban la productividad del equipo: el abandono de uno de los integrantes y el poco cumplimiento de otro. La persona cuestionada por su bajo rendimiento explicó las razones, se disculpó y volvió a asumir el compromiso de trabajar las horas pactadas. Para avanzar con el proyecto a pesar del abandono de uno de los integrantes, consensuaron en tratar de comprometerse un par de horas más cada uno.
- Se comentó que no resultaba tan simple trabajar en dos proyectos simultáneos, pero veían que se lograba avanzar bastante rápido con la implementación.
- Se vio como positivo que las pruebas sean realizadas por otra persona ya que permitían detectar errores y decidieron esforzarse más en la realización de las pruebas.
- Se decidió también buscar mejoras visuales en los sistemas que guíen mejor al usuario.
- Se decidió en ese momento poner más énfasis en el gimnasio, ya que la librería estaba más avanzada.

En su primera reunión de este tipo, el Grupo 2 la realizó básicamente porque tenían muchos problemas con GXServer en su versión gratuita y esto les impedía centrarse netamente en el proyecto. En grupo se decidió modificar la manera de trabajar, dejando completamente de lado dicha herramienta y definieron otra alternativa: exportación e importación de avances mediante archivos .xpz; enviándose oportunamente a través de correos electrónicos y realizar programación con Dummies. Además reiteraron el compromiso de trabajar responsablemente cada miembro del equipo.

En la segunda reunión de cada grupo se vió como objetivo alentarse a seguir trabajando para concluir los proyectos. El grupo 1 destacó los logros alcanzados, aún trabajando con menos personas que el otro equipo y también se resaltó el compromiso de cada miembro del equipo. Se alentó a seguir trabajando de igual manera.

En este caso, el grupo 2 analizó la falta de tiempo dedicada al proyecto y concluyeron que esta se debió al tiempo dedicado a la preparación de los exámenes del turno extraordinario. Todos se comprometieron a avanzar lo máximo posible para mostrar al cliente del gimnasio y poder cerrar en el menor tiempo posible el sistema de la librería.

Analizando los resultados de cada una de las reuniones de evaluación y reflexión interna se puede observar que cumplió con la finalidad por la que fueron establecidas: hubo lugar para alentar a los integrantes, lugar para resolver problemas técnicos y proponer mejoras, lugar para buscar mayor compromiso de cada uno. Por todo esto se considera que realmente esta reunión es necesaria en la propuesta.

Reunión de evaluación externa: según la propuesta, la realizaba el analista con el cliente buscando la aceptación del cliente de lo entregado y brindándole la oportunidad para expresar nuevos ajustes a los requerimientos.

En algunas reuniones de este tipo, reportadas en la plataforma y también observadas durante su realización, el cliente simplemente realizó cambios de prioridades y manifestó estar confirme con lo que estaba realizando. En otros casos aclaró que una historia de usuario que había sido mal interpretada por un grupo, otras veces manifestó el deseo de incorporar gran cantidad de funcionalidades, o bien eliminar otras.

Siempre a continuación de esta reunión se continuó con la reunión de ajuste, para detallar los cambios requeridos y posteriormente el analista convocó a una reunión de contextualización para comunicar al grupo dichos cambios solicitados.

Para el 22% de los que completaron la encuesta (Figura 4.4.6), ésta fue la reunión más porque consideraron que permitió comprender mejor lo que el cliente necesitaba, si se estaba entendiendo y satisfaciendo realmente sus necesidades y obteniendo retroalimentación directa respecto del rumbo del desarrollo para ir ajustándolo. También porque servía como puntapié inicial para relevar nuevas inquietudes del cliente.

Reunión de ajustes: El objetivo al proponer esta reunión fue brindar una herramienta para mantener la lista de Historias de Usuario actualizada y priorizada según las necesidades del cliente. Se reúnen únicamente el analista, junto con el cliente

En estas reuniones el cliente tuvo portunidad de detallar nuevas historias de usuario como por ejemplo en el caso del Gimnasio, o realizar modificaciones en prioridades de la Librería. Siempre al final de estas reuniones se actualizó la pizarra Kanban con lo convenido en esta reunión tal como se establecía en la propuesta y se hizo una reunión de contextualización con el resto del equipo para explicar lo acontecido.

Al respecto de los cambios solicitados en esta reunión, se consultó en la encuesta final si el equipo aceptó los cambios de requerimientos realizados por el cliente obteniendo una respuesta afirmativa de un 89% y el 11% restante no contestó (Ver Figura 4.4.13).

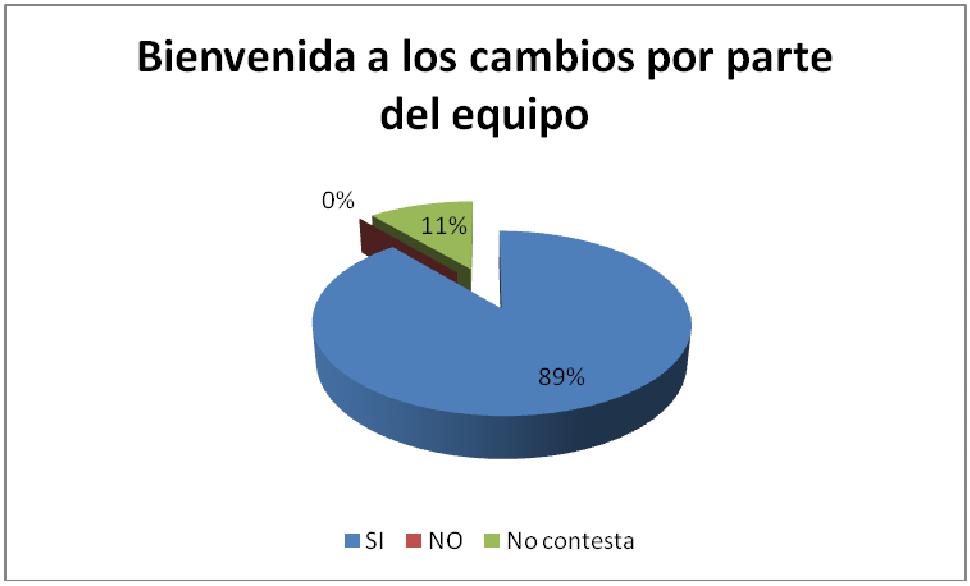


Figura 4.4.13: Bienvenida a los cambios por parte del equipo

Y respecto a la complejidad para realizar los cambios solicitados (Ver Figura 4.4.14):

- el 37% consideró a la complejidad baja debido a que el analista logró entender bien los cambios y expresar de manera concreta al grupo los mismos, también a que Gx se adapta fácilmente a la metodoogía ágil y porque no fueron grandes cambios.
- el 50% consieró una complejidad media, aduciendo algunos por cuestiones de estar trabajando con un trial, otro adujo que para ciertas funcionalidades se debió profundizar ciertos conocimientos de Genxus y otro por ejemplo destacó que los cambios solicitados tenían cierta complejidad de diseño.
- un 13% consieró que fue complejo debido a que no se contaba con todo el conocimiento necesario sobre Genexus

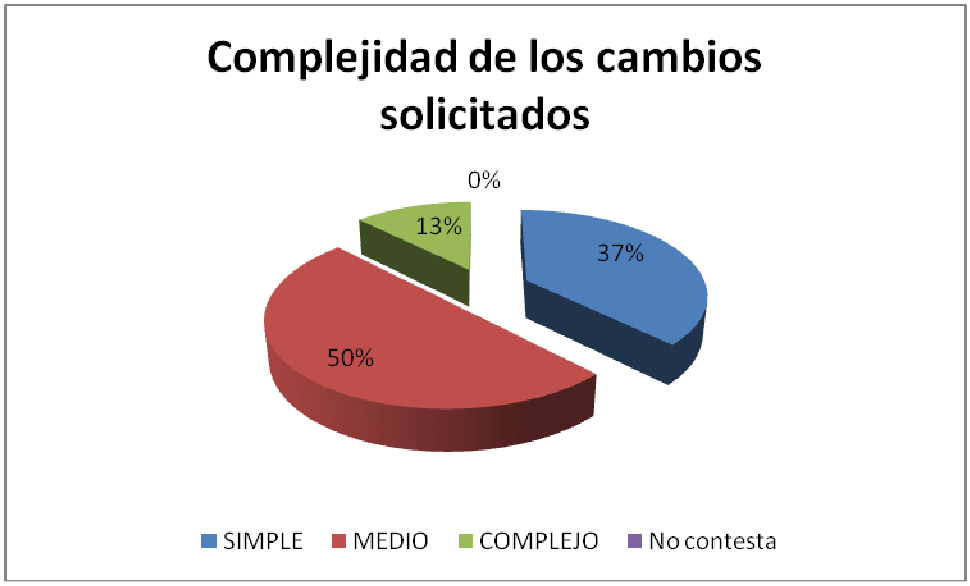


Figura 4.4.14: Complejidad de los cambios solicitados

En la experiencia se realizó siempre esta reunión de ajuste después de la reunión de evaluación externa, pero no por haberse dado de esta forma se debe considerar unificarlas. La diferencia es sutil, se podría tener una reunión de ajustes breve debido simplemente a un cambio en el negocio que implica por lo tanto modificaciones en prioridades o historias de usuario. No es necesario que en la reunión se realice una reflexión del producto entregado hasta el momento, que es el objetivo de la reunión anteriormente detallada. Por lo tanto se concluye que si bien generalmente despues de una reunión dereflexión externa suele realizarse una reunión de ajuste, podría realizarse directamenteuna reunión de ajustes a pedido del cliente por necesitar realizar modificaciones a los requerimientos. Mientras que las reunión de reflexión externa suelen ser reuniones pactadas previamente, sugeridas por el analista hacia el cliente.

Por lo tanto, se considera que el objetivo planteado al proponer esta reunión se logró en ambos grupos durante la experiencia.

Para resumir la opinión generada a partir de la experiencia sobre las reuniones de la etapa de producción, se puede decir que todas y cada una de las reuniones de esta etapa cumplieron con su objetivo para ambos grupos, logrando desarrollar el producto requerido por el cliente, aceptando los cambios que surgían y manteniendo una comunicación fluida dentro del equipo de desarrollo y con el cliente. Tal vez, como al cliente no le agradan demasiadas reuniones, frente a él no convenga hablar de dos tipos de reuniones, como son las de ajuste y de evaluación externa. Pero se debe brindar oportunidad cuando el cliente desee de realizar ajustes a los requerimientos y no solo cuando evalúa el producto que se le ha entregado hasta cierto momento.

Esta etapa de desarrollo concluyó para ambos equipos de manera posterior a una reunión de evaluación externa y ajuste donde el cliente informó que ya no implementaría más funcionalidades y se definió la fecha de entrega.

Otros puntos a considerar en etapa de producción

Pero dentro de esta etapa vale la pena resaltar también otros puntos, que se mencionarán a continuación

- **Herramientas**

Si bien ambos grupos trabajaron con una pizarra Kanban utilizando la herramienta Kanbanize, ambos grupos reconocieron no haber explotado todas las posibilidades que brindaba la

herramienta. Incluso uno de los grupos utilizó planillas Excel para complementar el seguimiento, siendo que la herramienta elegida proporcionaba dicha información.

Dentro de la herramienta Kanbanize definieron las historias de usuario, establecieron fechas tentativas de entrega, dentro de la reunión de contextualización. Establecieron como se mencionó anteriormente las etapas a seguir dentro del desarrollo y para cada una se estableció su definición de terminando (DoD). Un 20 por ciento en la encuesta respondió haber trabajado limitando los WIP de las actividades.

- **Trabajo en varios proyectos simultáneo**

Para trabajar en una misma pizarra dos proyectos, se sugería trabajar con dos colores diferentes para que visualmente ayude a diferenciarlos. Uno de los grupos siguió este lineamiento mientras que el otro trabajó diferenciando el ID de la historia con una inicial según el proyecto, porque prefirió usar los colores para distinguir diferentes tipos de historias de usuario, si eran pequeñas modificaciones, grandes cambios, etc.

Se pudo trabajar con un mismo equipo en ambos proyectos, por lo que es viable la propuesta pero de ser posible sería mejor trabajar cada proyecto con un grupo diferente ya que cada uno tiene un contexto diferente.

- **Aprovechamiento cualidades del DBC**

Si se consideran las reuniones propuestas y la pizarra de tareas, éstas podrían realmente servir para cualquier equipo de desarrollo que trabaje con reutilización y en equipos que quiera comenzar a realizar una transformación a su proceso de desarrollo para incorporar prácticas ágiles. Pero dentro de las actividades diarias del período de producción se había colocado énfasis en la prototipación rápida, ya que ésta era una cualidad muy beneficiosa al consideraba el desarrollo basado en conocimiento.

Esta herramienta para el desarrollo no fue aprovechada. Podría explicarse esto por la poca experiencia de los participantes, quienes tal vez no se sentirían seguros de realizarlo frente al cliente, por miedo a que ocurra algún error. Como los objetivos se lograron de igual manera podría pensarse en que la prototipación no es necesaria, pero en esta propuesta se considera que utilizarla podría haber clarificado más en primeras instancias algunos detalles que posteriormente surgieron en la primera reunión de evaluación interna. Además, como se verá en los comentarios de la empresa que opinó sobre la propuesta, consideran una gran herramienta prototipar frente al cliente sobre todo en la reunión de planificación inicial y de ajuste.

- **Pruebas**

Ninguno de los grupos utilizó Framework de pruebas, realizándolas de manera manual, buscando que el que realice la prueba no sea quien realizó la implementación. La razón de no haberla utilizado en la experiencia, según encuesta, se visualiza en el siguiente gráfico:

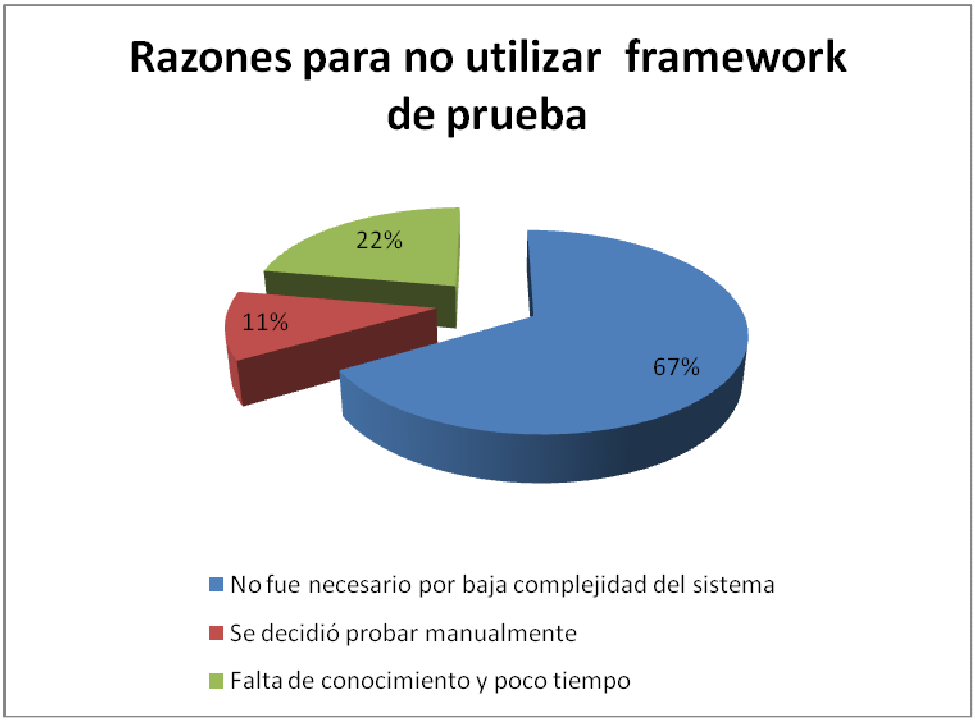


Figura 4.4.15: Razones aducidas para no utilizar Framework de prueba

La forma en que se garantizó que cambios no propagaran errores fue: mediante evaluaciones de reportes de referencia generados por Genexus, realizando múltiples pruebas de todo el sistema y analizando los objetos sobre los que iba a impactar los cambios realizados. En caso de surgir algún error, se contaba con backups para volver a versión anterior siempre.

- **Integración**

Al no compartir todo el tiempo el lugar de trabajo y haber tenido problemas con GxServer libre, se volvió más complejo el desarrollo. Esto llevó a que no lo utilice ninguno de los grupos. Entonces debido a que muchas veces no estaban en el mismo lugar, ambos grupos determinaron un lugar en la web para colocar la KB y esta se actualizaba después de que el encargado de pruebas consideraba que estaba bien (se agregó esta actividad al DoD del testing) e informaba al resto por chat. Ambos grupos trabajaron con programación con Dummies. Si se trabajara en un mismo lugar compartiendo la KB directamente no habría estos problemas, por tanto se debería tratar de que los miembros compartan el lugar de trabajo así se minimizan estos inconvenientes. La mayoría de los encuestados, en el relevamiento para realizar el estudio de campo, manifestaron compartir el lugar de trabajo, por lo tanto esto no sería un inconveniente.

- **Reutilización**

Se hizo mucho hincapié en la reutilización, sobre todo al inicio de la experiencia. Como los alumnos no tenían demasiada experiencia y objetos desarrollados previamente se sugirió que también en un desarrollo era válido reutilizar objetos creados por otros y se sugirió buscar en la nube de GXServer. Esto también causó una reacción de sorpresa, dado que se les estaba permitiendo usar algo no desarrollado por ellos. Pero al final de la experiencia el 88.89 por ciento concluyó que es más rápido aunque en algunos casos no tanto como hubieran imaginado. Para el 11.11% restante es más rápido solo en algunos casos, y consideró más rápido desarrollar desde cero que modificar propiedades y atributos. Caso especial fueron las reutilizaciones de objetos que no necesitaron adaptarse y se reutilizaron sin modificaciones, como por ejemplo aquellos objetos involucrados con la seguridad del sistema.

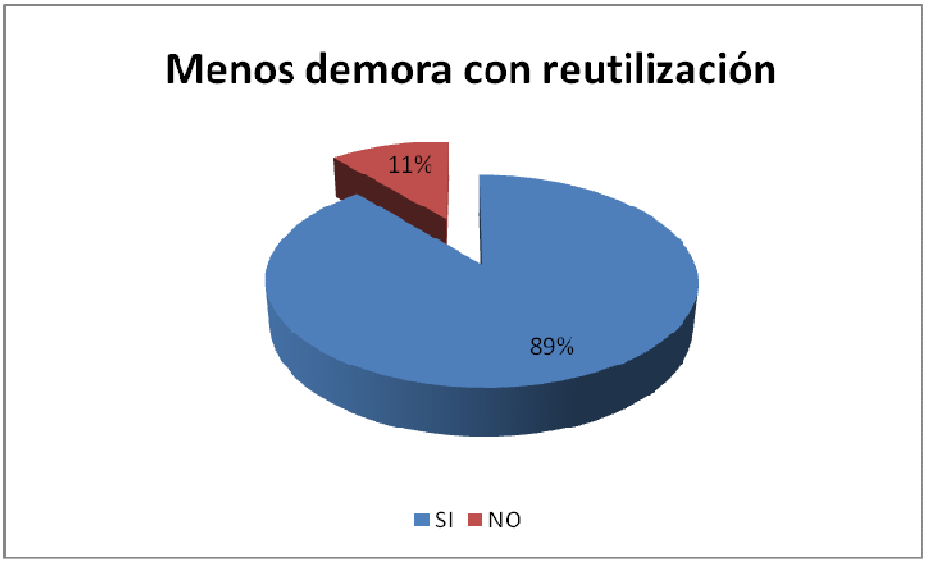


Figura 4.4.16: Reducción de tiempos realizando reutilización

Por todo lo descripto en esta etapa de la experiencia, se considera que es posible realizarla sin inconvenientes. Tal vez como al cliente no le agradan demasiadas reuniones, frente a él no convenga hablar de dos tipos de reuniones, como son las de ajuste y de evaluación externa. Pero se debe brindar oportunidad cuando el cliente desee de realizar ajustes a los requerimientos y no solo cuando evalúa el producto que se le ha entregado hasta cierto momento.

4.4.3.8.3 RITUAL DE FINALIZACIÓN.

Figura 4.4.17: Etapa de Ritual de finalización

En la propuesta, se planteó como objetivo de esta última etapa no solo entregar al cliente el producto final sino también aprender de la experiencia. Para lograr esto se consideró realizar dos reuniones: una de **entrega final al cliente**, y otra de **reflexión final**.

Reunión de entrega final, en la propuesta se planteaba que en esta reunión participaba el analista con el cliente con el objetivo de dar por concluido el proyecto habiendo realizado la entrega del sistema empaquetado.

Tanto en la reunión de entrega final para el proyecto de la librería como para la del gimnasio, se vio al cliente conforme con lo desarrollado. Antes de llevar a cabo esta reunión cada grupo había instalado el sistema en un servidor privado. Se instalaron ambos ya que hasta ese momento no se sabía cuál de los dos sistemas elegiría el cliente en cada caso.

Algo a destacar es el entusiasmo observado al presenciar la reunión con el dueño del gimnasio. En la reunión con cada grupo, comentó haber estado accediendo al sistema desde su trabajo y que su secretario también había navegado por el sitio final.

El 88.89% que respondió la encuesta consideró que realmente es necesaria esta reunión y el 11% restante no contestó la pregunta (Ver Figura 4.4.18).

Figura 4.4.18: Necesidad de la reunión de entrega final

Las razones aducidas pueden generalizarse en las siguientes:

- representa la entrega del producto final
- permite al equipo mostrar su trabajo y aprender nuevas cosas para el futuro
- permite evaluar el nivel final de satisfacción del cliente con el producto desarrollado y brinda experiencia para aprender de los errores cometidos
- permite saber el grado de satisfacción final del cliente, y posibles mejoras futuras
- permite recibir información del cliente sobre el desempeño del equipo

En base a lo analizado, la reunión cumplió con su propósito.

Reunión de reflexión final, antes de terminar de cerrar el proyecto la propuesta considera muy importante aprender de la experiencia vivida. Plantea reunir todo el equipo con los stakeholders para tal fin

En ninguna de estas reuniones se pudo contar con al menos un stakeholder, como se había logrado al inicio de la experiencia para cada proyecto. Se realizó con el mismo cliente que había participado de la reunión anterior.

En el caso de la librería, el cliente aprovecho para comentar que fue muy demandante el tiempo requerido para las reuniones. Cabe aclarar que el cliente debió participar en una reunión por cada grupo, salvo en la inicio ágil. Por lo que no amerita reconsiderar las reuniones planteadas en la propuesta por este comentario.

Otra cuestión a destacar del proyecto de la librería es que el cliente hizo un comentario al grupo que había tenido la deserción de uno de los integrantes, valorando su esfuerzo y resultados obtenidos. Esto permite detectar que las reuniones cara a cara generan un vínculo entre el equipo de desarrollo y el cliente. Posteriormente, cuando ese grupo estuvo sin la presencia del cliente remarcaron ese comentario y si bien el cliente prefirió el software desarrollado por el otro grupo, quedó conforme con lo realizado, y reconocieron haber aprendido muchas cosas

Para el proyecto del gimnasio, el cliente informó que, si bien el secretario no había asistido a la reunión, había lamentado que se hubiera sacado del listado de funcionalidades del sistema el manejo de caja chica, ya que era algo muy importante para él. Destacó además como algo positivo, rapidez con que se fue desarrollando el sistema y, sobre todo, la posibilidad de no definir todo lo que necesitaba en una sola reunión. Valoró la buena predisposición para responder a los pequeños cambios y nuevos requerimientos de ambos grupos.

El equipo valoró la buena predisposición del cliente en el proyecto y lamentó no contar con la opinión directa de las otras personas que usarían el sistema. Pero se agradeció el haber compartido la opinión del secretario.

Después que el cliente se retirar de la reunión, cada grupo buscó aprender y extraer cosas positivas y negativas de lo que había ocurrido. Algunos de los comentarios surgidos en estas reuniones, sin discriminar si fue en el proyecto de la de la librería o del gimnasio, relevados de las encuestas fueron los siguientes:

- el equipo no quedó del todo conforme con algunos detalles de los productos, por las limitaciones de haber trabajado con una versión trial de Genexus
- el equipo estuvo muy conforme con la forma de trabajar en equipo
- el equipo comentó diferentes cosas que “se podrían haber hecho” para considerar a futuro.
- el equipo enfatizó la reutilización de objetos para el proyecto que seguía en desarrollo (opinión surgida en la reunión de reflexión final para el proyecto de la librería)

Analizando estos puntos se puede decir que hubo un cierto aprendizaje basado en las experiencias vividas que enriqueció cada grupo y este era el objetivo de esta reunión. Lograron extraer comentarios positivos y negativos del proyecto y en el último punto descripto, donde se sugería maximizar la reutilización, incluso llegaron a proponer “mejoras” al proceso de desarrollo.

La propuesta requería la presencia de los stakeholders, pero no fue posible. Debido a esto, en la encuesta, se consultó la opinión de cada uno respecto a la importancia de contar con los stakeholders en la reunión. Para el 56% sí es importante contar con su presencia, para un 33% no lo es y un 11% restante no contestó la pregunta (Ver Figura 4.4.19).

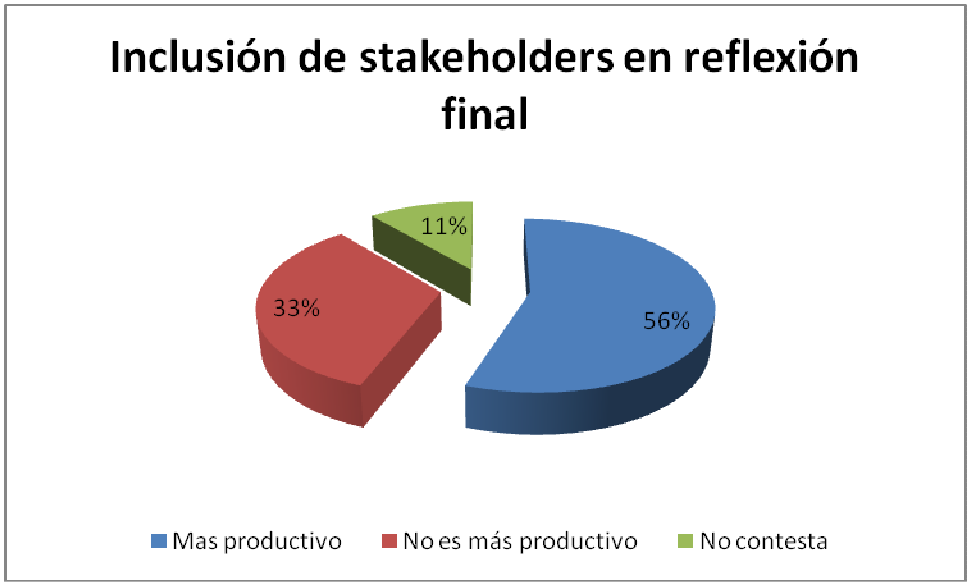


Figura 4.4.19: Inclusión de stakeholders en reflexión final

Quienes respondieron afirmativamente en términos generales la consideraron necesaria porque permite conocer si todos los stakeholders están satisfechos, teniendo en cuenta que cada stakeholder tiene una visión diferente. Quienes respondieron negativamente, simplemente no consideraron que influyera en el grupo para resolver problemas o bien era irrelevante.

El incluir los stakeholders en la reunión, tal vez, debería reconsiderarse, dado que en cualquier proyecto esta situación tiene muchas posibilidades de repetirse. Es casi una utopía imaginarse con la disponibilidad horaria de todos los stakeholders en una reunión, más habiendo realizado ya la entrega del sistema ya que incluso el cliente puede no ver ningún beneficio de ésta. La forma en que el cliente podría acceder es tal vez, organizando una jornada de capacitación y realizando al final de la misma esta reunión. Otra manera de obtener su apoyo y colaboración es mostrarle que es una oportunidad para corroborar que todos ya conocen el sistema y obtener información que a él también pueda servirle. Pero si no se tiene el apoyo del cliente en esto, se puede pensar como alternativa realizar una encuesta breve a los stakeholders. Si bien la retroalimentación que se puede obtener de esta forma es más limitada, se tendría una idea del grado de satisfacción y algunas observaciones, quejas o comentarios de lo transcurrido durante el desarrollo desde el punto de vista de cada stakeholder y son indicadores que pueden aportar algo significativo al aprendizaje del grupo.

Por lo tanto, en base a la experiencia realizada, se considera que en el ritual de finalización ambas reuniones son necesarias. Cada una permitió alcanzar los objetivos por los que fueron incorporadas a la propuesta, entregar el producto y aprender en base a las vivencias durante el proyecto. Lo que se debería reconsiderar es la participación de los stakeholders. Si bien sería conveniente que estuvieran los stakeholders, su ausencia podría suplirse si se les solicita que completen una encuesta simple con su opinión del sistema desarrollado. Esta alternativa debería incorporarse como una opción concreta dentro de la propuesta.

Para el 22% de los que completaron la encuesta (Figura 4.4.6), ésta fue la reunión más porque consideraron que permitió comprender mejor lo que el cliente necesitaba, si se estaba entendiendo y satisfaciendo realmente sus necesidades y obteniendo retroalimentación directa respecto del rumbo del desarrollo para ir ajustándolo. También porque servía como puntapié inicial para relevar nuevas inquietudes del cliente.

4.4.3.9 Herramientas

En la propuesta, se planteó el uso de pizarra Kanban, que como se comentó anteriormente ambos grupos decidieron trabajarla on-line a través de Kanbanize, pero ninguno de los grupos maximizó todo su potencial. En opiniones vertidas en la encuesta, todos reconocen que es una herramienta que ayuda al seguimiento del proyecto. A continuación se mencionan algunos comentarios sobre la herramienta detallados en las encuestas:

- Ayudó a aplicar las pautas ágiles propuestas
- Ayudó a repartir tareas y organizar mejor el trabajo, establece roles claros
- Permitió organizar las tareas de desarrollo del equipo
- Bastante buena, clara y da a primera vista un panorama actual del proyecto
- Permitía visualizar como avanzaba el desarrollo y que estaba realizando cada uno
- Se podía ver que tareas estaba realizando cada individuo en cada instante, el grupo no la implementó del todo
- Bastante completa, y compleja solo se usó una pequeña parte
- Fue muy útil para aplicar la metodología ágil

Respecto a otras herramientas utilizadas, trataron además de utilizar GXServer para la integración y poder compartir fácilmente la KB, pero en su versión gratuita y por problemas lo dejaron de lado ambos grupos. Uno de los participantes comentó también el uso de planilla Excel y Factbook. Planilla de cálculo, para hacer seguimiento fácilmente, esto se explica debido a que no explotaron todas las características de la herramienta adoptada para llevar la pizarra Kanban de manera digital on-line. De haberse profundizado en su uso, no habría sido necesario.

4.4.3.10 Lugar y tiempo de trabajo

En ambos grupos, algunas veces, se compartió el lugar y horario de trabajo. Al no estar en un lugar común gran parte del tiempo, utilizaron correo electrónico, mensajería instantánea, Facebook, y otras herramientas que brinda Internet para poder comunicarse y compartir información, dudas, problemas, etc.

El 88.89 por ciento está de acuerdo en la importancia de compartir el lugar de trabajo, para trabajar de manera más simple, con una sola KB y sobre todo para compartir dudas, ideas, cooperar. Reconocen que hay cuestiones que solo se pueden resolver en persona. El 11.11 por ciento restante no contestó la pregunta por SI o NO pero consideró que “se logra mayor comunicación y entendimiento del proyecto, además de que los problemas se ponen en común y muchas veces se solucionan más rápido”. Debido a esta justificación, podría considerarse que también es un SI, por lo que se puede concluir que todos se dieron cuenta de la importancia de compartir el lugar de trabajo.

El 100 por ciento también considera que es necesario compartir el horario de trabajo, casi por las mismas razones por las que deben compartir el lugar de trabajo. El dialogo cara a cara es más rápido y hay cosas que se resuelven personalmente, fueron las dos razones más nombradas. También que ayuda a mejorar la organización del equipo, plantear ideas, evacuar dudas.

Pero, aún sin compartir completamente el horario de trabajo suplieron la comunicación directa por medios alternativos. Aunque, en base a la experiencia, se vuelve más evidente la necesidad de compartir el lugar de trabajo, de esta forma se puede beneficiar el equipo al tener radiadores de información colgados en las paredes que mantengan a todos informados además de la pizarra Kanban y tener acceso directo al Servidor general sin necesidad de conexiones de Internet. Compartir horarios sería óptimo para que fluya la información directa entre todos, pero puede bastar sin tantos beneficios compartir ciertos horarios para reuniones.

Todos trabajarían nuevamente con Genexus. Estos son los comentarios por los que lo utilizarían:

- Acelera mucho los tiempos para finalizar un proyecto.
- Facilita el desarrollo de software,
- Ahorra tiempo de desarrollo
- No es muy compleja en aprender a utilizar por lo menos lo básico
- No hace falta conocimiento de muchos lenguajes de programación
- Tiene mucho soporte en línea para consultar

4.4.3.11 Dificultades en la experiencia

Para poder obtener mayor información de la experiencia de cada alumno, se solicitó en la encuesta nombrar las tres dificultades mayores enfrentadas indicando para cada uno el nivel del impacto, pudiendo optar por MUY BAJO, BAJO, MODERADO, ALTO, MUY ALTO.

Se pueden nombrar seis que fueron comentarios repetitivos entre los alumnos.

- El uso de GxServer (77,77%)
- Poco conocimiento de Genexus (66,66%)
- La coordinación de horarios (55,55%)
- Utilización de una versión trial en la nube (33.33%)
- Los tiempos para realizar las entregas (22,22%)
- Posibilidad de compartir lugar de trabajo (22.22%)

Se muestra en la Figura 4.4.20 cada una de las dificultades mencionadas, sus porcentajes de referencias (mencionado) y nivel de impacto definido para cada uno.

Otros tres puntos fueron resaltados por uno sólo pero vale la pena nombrarlos:

- Tiempo para documentar (Muy alto)
- Hacer que todos los integrantes trabajen relativamente iguales (Alto)
- Trabajo simultáneo en diferentes sistemas (Bajo)

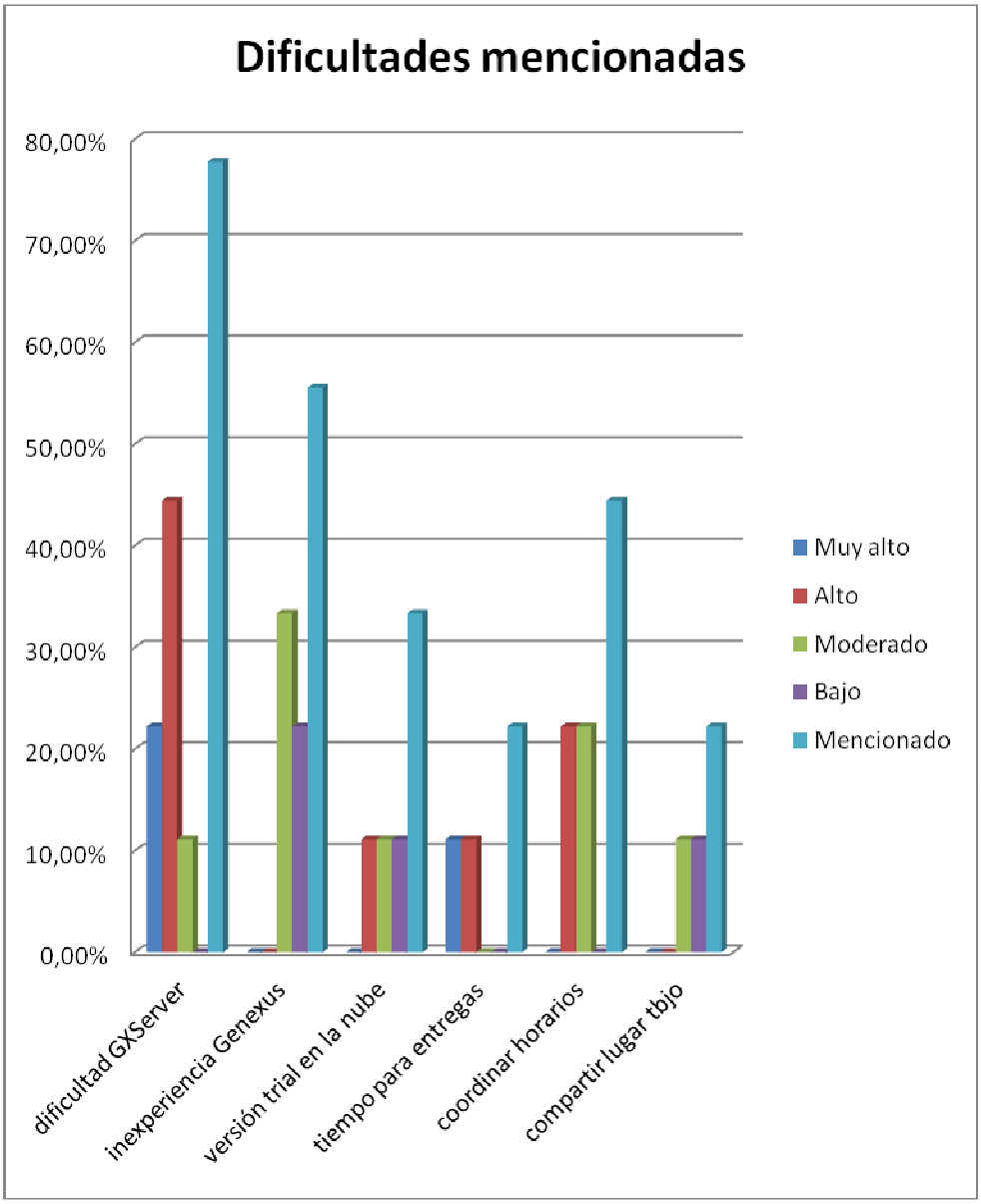


Figura 4.4.20: Dificultades mencionadas

En un proyecto real, trabajando con la versión oficial de Genexus muchas de estas dificultades desaparecerían. También la falta de experiencia no estaría considerada como dificultad porque se sobreentiende que ya los equipos que viene trabajando con DBC con Genexus manejan este software y forma de desarrollo.

Por otro lado, el poder trabajar en un lugar común, como se mencionó anteriormente reduciría muchas complicaciones a la hora de compartir la KB, aún sin utilizar GXServer. El coordinar horarios comunes de trabajo es más probable cuando existe una relación laboral y no en una experiencia universitaria, donde queda a responsabilidad de cada uno debe buscar cumplir con los horarios que demanda el grupo.

Siempre existirá una presión para tratar de entregar lo máximo posible en la menor cantidad de tiempo, independientemente del proyecto y metodología a utilizar. Esto es común en cualquier proyecto de desarrollo de software, no solamente en esta experiencia. Pero, al poder ir entregando

software funcionando al cliente, se puede ir cumpliendo con sus necesidades más urgentes para su negocio y esto en la mayoría de las situaciones siempre será valorado.

El tiempo que debe tomarse el coordinador para registrar la información relacionada con el proyecto, debe considerarse tiempo valioso que realmente aporta al grupo.

Y por último, en todo equipo siempre habrá personas más comprometidas que otras, pero a través de las reuniones de seguimiento se puede persuadir a aumentar el compromiso. De esta manera todos trabajarán dando lo mejor de cada uno. Por lo tanto si bien se planteó la dificultad de hacer que todos los integrantes trabajen relativamente iguales puede aparecer en cualquier metodología, brindando esta propuesta, basándose en prácticas ágiles, las herramientas para tratar de minimizar la falta de compromiso.

4.4.4 Conclusión experiencia

En términos generales, según la experiencia se podría decir que la propuesta cumplió su objetivo de introducir a quienes están desarrollando con DBC al desarrollo ágil. El problema de trabajar con una versión gratuita desvió un poco la atención de aplicar la propuesta a lograr implementar con Genexus ciertas historias de usuario.

Costó al principio hacer que los alumnos participen en las reuniones pero luego los resultados fueron muy buenos en cada tipo de reunión diferente. Se pudo constatar que no es siempre posible contar con los stakeholders en todas las reuniones, se debería proponer alternativas dentro de la propuesta en estos casos. Se plantea entonces, utilizar encuestas para obtener cierta retroalimentación o realizar visitas al lugar de trabajo.

La etapa de inicio ágil es completamente aplicable, haciendo la salvedad que tal vez no sea posible que todos los stakeholders estén participando, pero las dos técnicas plateadas resultaron efectivas.

- La reunión de inicio ágil logró realmente introducir en el tema del nuevo proyecto a todos los involucrados. De esta manera, es el equipo completo quien construye la visión y ya no es el líder de proyecto quien transmite la visión al equipo. Se favorece trabajando de este modo la comunicación directa entre todos, evitando malas interpretaciones. Es el equipo completo que comienza a avanzar en el desarrollo de un nuevo proyecto de manera conjunta.
- La reunión de planificación inicial cumplió con obtener un listado priorizado de las historias de usuario

En la etapa de producción, se pueden nombrar varios puntos:

- Reuniones planteadas, se puede decir que todas y cada una de las reuniones de esta etapa cumplieron con su objetivo para ambos grupos, logrando desarrollar el producto requerido por el cliente, aceptando los cambios que surgían y manteniendo una comunicación fluida dentro del equipo de desarrollo y con el cliente. Tal vez, como al cliente no le agradan demasiadas reuniones, frente a él no convenga hablar de dos tipos de reuniones, como son las de ajuste y de evaluación externa. Pero se debe brindar oportunidad cuando el cliente desee de realizar ajustes a los requerimientos y no solo cuando evalúa el producto que se le ha entregado hasta cierto momento
- La programación con dummies, técnica sugerida en la propuesta, ayudó a trabajar en objetos relacionados al mismo tiempo.
- El trabajo de dos proyectos en paralelo es posible, siendo siempre mejor si se pudieran trabajar independientemente.
- La pizarra Kanban permite tener la información visible de todo el proyecto, permitiendo organizarse, realizar el seguimiento, trabajar con dos proyectos simultáneos, tener definidos el DoD de cada etapa establecida en la pizarra.

- Genexus permite prototipación rápida, si bien la experiencia sugiere su aplicación en algunas situaciones, se podría haber hecho mayor hincapié en ella.

Pero para no reiterar lo dicho antes al analizar cada punto de la experiencia según las etapas propuestas, lo más relevante es que, se considera que es posible realizarla sin inconvenientes. Tal vez como al cliente no le agradan demasiadas reuniones, frente a él no convenga hablar de dos tipos de reuniones, como son las de ajuste y de evaluación externa. Pero se debe brindar oportunidad cuando el cliente desee de realizar ajustes a los requerimientos y no solo cuando evalúa el producto que se le ha entregado hasta cierto momento.

En ritual de finalización, también se concluyó que, en base a la experiencia realizada, ambas reuniones planteadas son necesarias. Cada una permitió alcanzar los objetivos por los que fueron incorporadas a la propuesta, entregar el producto y aprender en base a las vivencias durante el proyecto. Lo que se podría reconsiderar es la participación de los stakehlders. Sería conveniente que estuvieran los stakeholders pero su ausencia podría suplirse si se les solicita que completen una encuesta simple con su opinión del sistema desarrollado. Esta alternativa debería incorporarse en la propuesta.

Respecto al lugar de trabajo, es conveniente que se comparta, si bien los horarios pueden no ser los mismos y de ser posible se debería buscar momentos en que los horarios del equipo se solapen, para fomentar la comunicación directa. En base a la experiencia, se volvió más evidente la necesidad de compartir el lugar de trabajo, de esta forma se puede beneficiar el equipo al tener radiadores de información colgados en las paredes que mantengan a todos informados además de la pizarra Kanban y tener acceso directo al Servidor general sin necesidad de conexiones de Internet. Compartir horarios sería óptimo para que fluya la información directa entre todos, pero puede bastar sin tantos beneficios compartir ciertos horarios para reuniones.

La experiencia realizada para aplicar la propuesta será realizada nuevamente el año próximo a los alumnos que estén cursando la asignatura. Esto se debe a que, como resultado de la misma, los alumnos no solo entendieron más sobre Desarrollo basado en conocimiento utilizando Genexus, sino que ellos mismos reconocieron haber logrado percibir cómo es la filosofía detrás de las metodologías ágiles. Algunos de los alumnos que participaron, consideraron que la experiencia ameritaba ser una asignatura completa, para poder realizar una experiencia más profunda y debatir más sobre los conceptos ágiles utilizados. El responsable de la materia también manifestó su conformidad a los resultados, sobre todo por haber logrado un gran compromiso y participación por parte de los alumnos. Los mismos alumnos comentaron a otros sobre la experiencia y hay muchas expectativas para el próximo año entre ellos.

4.5 OPINION DE EMPRESA DEL MEDIO

Como se mencionó al inicio del capítulo, se reenvió una encuesta a todos los sectores que participaron del relevamiento sobre la forma de trabajo con Desarrollo Basado en conocimiento en Salta Capital. La encuesta fue diseñada de manera breve para que puedan contestarla en poco tiempo.

Hasta el momento respondió a la encuesta sólo uno de ellos y sus comentarios se describen a continuación. Es de esperar que se reciban otras opiniones y puedan enriquecer con su aporte a la propuesta más adelante, pero por limitaciones de plazos para realizar este informe solo se mostrará los datos recibidos por esta única empresa. En Anexo 5, se encuentra dicha encuesta.

A continuación se reflejan los datos obtenidos para cada uno de los 15 puntos planteados en la encuesta.

1. Consideró que podría aplicarse a su situación la forma de dividir el proyecto en Inicio Ágil, Producción y Ritual De Finalización

2. Dentro del Inicio Ágil, consideró que proveería un aporte considerable. Su justificación fue la siguiente “Tener una idea común de todo el equipo es valioso, la propuesta propone algo simple. Se puede llegar a considerar incorporar en la empresa el inicio ágil, por las ventajas enunciadas en la propuesta y por no consumir demasiado tiempo.”

3. En la etapa de Producción. No consideró viable realizar todas las reuniones ya que en la propuesta se solicita que las reuniones de ajuste sean presenciales, para que ahí solicite los cambios. Aclara además que eso es poco probable, por manejarse con sistema de tickets, salvo cambios necesarios complejos o profundos. Pero el resto de las reuniones consideró que sí son viables de realizar.

4. Dentro de las reuniones propuestas en la etapa de Producción las reuniones que figuran en la propuesta y que tiene semejanza a reuniones realizadas actualmente son: SEGUIMIENTO se realizan diariamente, REFLEXION INTERNA aproximadamente cada dos meses. REFLEXION EXTERNA, es una reunión similar, tiempo variable.

5. En la Etapa del Ritual de Finalización consideró algo probable realizar ambas reuniones propuestas debido a la dificultad de contar con disponibilidad horaria del cliente y los stakeholders, pero se reconoce los beneficios de estas reuniones. Proponer buscar alternativas.

6. En la reunión específica reflexión final dentro de la etapa de Ritual de Finalización, considera poco contar con participación de stakeholders en las reuniones. Si se lograra la participación de los stakeholders como se plantea en la propuesta, se vitarían muchos malos entendidos, pero esto es poco probable. Hay que buscar alternativas

7. Se consideró algo probable utilizar la pizarra Kanban debido a que están trabajando actualmente con una pizarra Scrum, si bien se consideró que sería factible su uso.

8. En la propuesta se consideró que compartir el lugar de trabajo es importante, en la encuesta se consideró muy probable lograr esta situación. Actualmente todos comparten el lugar de trabajo, nadie trabaja desde sus casas u otro lugar, si bien a veces el horario no es el mismo.

9. Se consideró probable realizar el desarrollo con reutilización, como plantea la propuesta. Se utiliza actualmente muchísima reutilización, por el momento es el líder de proyecto quien determina que reutilizar. Podría probarse de la forma que se sugiere en la propuesta, el tema es que el líder es quien más conoce del proyecto y sus objetos.

10. Se consideró poco probable incorporar los roles propuestos. Actualmente el cambio llevaría a malos entendidos, y el proyecto en curso es muy grande y crítico para arriesgar. Se debería realizar con mucha cautela, además todos están acostumbrados a rendir cuentas al líder del proyecto. Implicaría un cambio cultural grande que no serpia conveniente en estos momentos, tal vez a futuro.

11. Se consideró muy probable incorporar la prototipación en las reuniones de ajuste o planificación inicial. Se reconoció que en la reunión inicial es muy útil y ayuda a evitar malos entendidos. Actualmente se está utilizando prototipación. En reuniones de ajustes o modificaciones no se podría porque generalmente los cambios se solicitan por sistema de tickets. Se comentó que a veces se realiza prototipación en el lugar de trabajo del cliente o se llevan prototipos y se muestran a él o a quién él designe.

12. Se consideró muy probable poder utilizar algún Framework para automatizar las pruebas, ya que actualmente utilizan GxTest. Se informó además que tienen varias personas avocadas a esa tarea exclusivamente de probar. Además se comentó que están mejorando mucho en la producción, trabajando de esta manera

13. Se consideró probable trabajar con GXServer para realizar la integración. Si bien es una herramienta costosa, se está analizando utilizarla. El proyecto actual es muy grande.

14. Se solicitó una opinión general sobre la propuesta, considera si ayuda en entornos de desarrollo con desarrollo basado en conocimiento para acercarlo al desarrollo ágil. A lo que se respondió que ayuda bastante. Su justificación fue la siguiente: "Nosotros venimos realizando el cambio hacia ágiles, haber contado con estos lineamientos antes, nos habría facilitado varias cuestiones al inicio porque nos costó encontrar la forma de iniciar el cambio. Por ejemplo podría haber sido útil definir otros roles para separar la idea del líder, podría haberse replanteado si hace falta iteraciones o no. El trabajar por flujo tal vez podría volver más solidario a los miembros del equipo. Otras cuestiones como lugar de trabajo, herramientas propuestas, reutilización, algunas reuniones de reflexión ya venimos trabajando y nos damos cuenta de su necesidad".

15. Aprovechando la posibilidad de realizar otros comentarios, aclararon que trabajan por iteración, pero se podría trabajar también por flujo de trabajo como se plantea, definiendo para cada historia de usuario la fecha probable de entrega. Pero por el momento la opción es trabajar por sprint. Aclararon además que el sistema en desarrollo actual tiene alrededor de cinco mil objetos, donde la reutilización es muy importante y el testeo siempre lo realizan más de 2 personas. Se integra y se va realizando backup de versiones anteriores y ante un error se vuelve inmediatamente a la versiona anterior.

Por todo esto podría decirse que esta propuesta podría aplicarse en la empresa relevada, con algunas salvedades.

- No actualmente porque están en medio de un gran proyecto y consideran riesgoso realizarlo ahora.

- De haberla conocido anteriormente podría haberlos ayudado en el cambio de la forma de trabajo más tradicional con la que venían trabajando hasta hace poco tiempo a un trabajo más ágil.

- No es muy sencillo conseguir disponibilidad de clientes y stakeholders, se deberían buscar alternativas.

4.6 CONCLUSIÓN

Se considera que la propuesta, en términos generales, es aplicable para quienes utilizan el desarrollo basado en conocimiento con Genexus y buscan acercarse a las metodologías ágiles. Es un lineamiento a seguir que cada empresa o sector determinará como adaptarlo y posteriormente mejorarlo con las reflexiones que realice el equipo.

Permite trabajar en más de un proyecto a la vez, siendo siempre más conveniente tener un equipo avocado a un único proyecto.

Cada una de las etapas logró cumplir con su objetivo específico planteados en la propuesta: inició ágil, producción y ritual de finalización.

Los roles planteados permiten ir eliminando la idea del líder tradicional del desarrollo de software, favoreciendo que el grupo se auto-organice y defina la forma de trabajo. Un detalle que se podría modificar es el nombre del rol analista que al principio confundió en la experiencia a los participantes debido a que dentro de las etapas propuestas dentro de la pizarra Kanban existe una etapa análisis. Otra alternativa sería, modificar el nombre de la etapa dentro de la pizarra. De esta manera se evitaría la asignación casi instintiva e intuitiva de asignar la tarea de este análisis al analista.

Es bueno considerar dentro de uno de los roles sugeridos, el análisis de objetos a reutilizar para que quede en evidencia esta actividad y pueda aprovecharse lo que el equipo ya ha desarrollado y probado anteriormente.

Se debe tratar de maximizar la ventaja de generar código rápidamente que no se expresó con profundidad en la propuesta. Se podría considerar prototipación en la mayor cantidad de reuniones con el cliente, permitiendo rápido entendimiento.

Las herramientas a utilizar como en toda metodología ágil, ayudan a reducir tiempos de trabajo, para agilizar entregas. Realmente utilizar un Framework de prueba es muy conveniente, porque brinda confianza, y por otro lado tener una etapa dentro del proceso de producción de prueba propiamente dicha ayuda a no menospreciarla y tener a través de su DoD a mano disponible todas las tareas necesarias para completar las pruebas.

En cuanto al DoD, es muy importante que se encuentre en un lugar accesible a todos y describirlo dentro de la pizarra Kanban se considera una muy buena alternativa. La pizarra Kanban es muy útil para visualizar el estado del proyecto y es muy simple de adaptar a las necesidades del equipo.

La propuesta no plantea desarrollo con iteraciones pero puede utilizarse dentro de iteraciones de tiempo determinadas sin problemas. Para ello, dentro de la iteración se trabajará con un subgrupo de tareas a implementar en un período de tiempo acotado y no con toda la lista.

Las fichas de historias de usuario permiten especificar fecha probable de entrega que puede ayudar a realizar planificaciones con sectores que necesitan proyecciones a largo plazo. Estas fechas deben establecerse dentro de las reuniones de contextualización y ser establecidas por todo el equipo, teniendo una idea de las fechas en las que el cliente ha manifestado su necesidad de obtenerlas en la reunión de planificación inicial o reunión de ajuste.

Las reuniones planteadas en cada etapa del proyecto cumplen con su objetivo específico. Hay una situación que no es menor y es la poca disponibilidad horaria del cliente y/o stakeholders para asistir a las reuniones que la propuesta estipula que participen. La propuesta debe ampliarse brindando alternativas en caso de no ser viable su participación, para cada situación. A continuación se detallan alternativas que se incorporan a la misma:

- Permitir a los clientes informar cambios de manera no presencial, por algún medio: correo, sistema de tickets reservando la presencia de los mismos para situaciones muy complejas. Para evitar malos entendidos en estas situaciones, se podría buscar generar un prototipo rápido que permita clarificar lo solicitado y llevarlo al lugar de trabajo, o subirlo a algún servidor de prueba para recibir opinión al respecto.
- Visitar a los stakeholders en su ámbito laboral o bien solicitar a los stakeholders completar una encuesta, reconociendo la perdida de cierta información que solo se logra en la comunicación cara a cara, sobre todo al definir la visión del proyecto:
    - que permita determinar la visión de cada stakeholder si es imposible que asistan a la reunión inicial
    - que permita determinar la opinión de cada stakeholder sobre el producto desarrollado si no pueden asistir a la reunión de reflexión final.

Si bien sería conveniente que el equipo comparta el lugar de trabajo, por los beneficios de la transmisión de información visual, mediante radiadores de información distribuidos en las paredes, o el acceso directo al Servidor general sin necesidad de conexiones de Internet, no es una condición indispensable para esta propuesta. Lo mismo sucede con los horarios de los participantes en el proyecto, sería óptimo para que fluya la información directa entre todos, pero puede bastar sin tantos beneficios compartir ciertos horarios para reuniones y fomentar, en ciertos momentos, la comunicación directa.

En resumen la propuesta es viable de aplicar con las salvedades mencionadas anteriormente. Es un puntapié inicial en el camino hacia el desarrollo ágil, permitiendo de una manera sencilla ir realizando el cambio cultural que es tan difícil e importante de lograr. Paulatinamente el equipo en las reuniones de reflexión interna irá refinando su forma de trabajo. Tal como manifiesta Cockburn, el equipo a medida que adquiera más experiencia adoptará nuevas prácticas.

5 CONCLUSIONES FINALES

Ágil toma un día para aprenderlo y una vida para perfeccionarlo [Epstein]

5.1 LA AGILIDAD REQUIERE UNA TRANSFORMACIÓN [KINDON 2007]

Después de un análisis de campo y estudio de propuestas de diferentes metodologías se pudo proponer un marco de trabajo donde se sugieren prácticas ágiles para comenzar a cambiar la forma de desarrollar utilizando desarrollo basado en conocimiento. Pero el paso hacia un desarrollo ágil debe ser tomado desde dentro de la organización y estar el equipo convencido del rumbo que está por tomar y los cambios que se deben realizar.

Como recalca Kindon, "*La transformación ágil efectiva es frecuentemente un cambio cultural total.*" Las personas deben cambiar su forma de interactuar, pensar y trabajar. Es un cambio cultural que involucra la comunicación abierta y la colaboración entre las personas ya sean parte del equipo de desarrollo propiamente dicho o bien estén más relacionados con los stakeholders. Con una buena comunicación se puede lograr confianza, confianza para exponer las dificultades, confianza para proponer alternativas de mejora, confianza que cada uno realizará de la mejor manera posible su tarea. Y no solo es comunicación y relación entre individuos, es tener un tiempo de reflexión para poder adaptar el proceso de desarrollo al equipo y al proyecto. Todo esto brinda transparencia y genera responsabilidad de cada uno de los involucrados.

Si bien las prácticas ágiles son importantes, los principios que las sustentan y que están declarados en el manifiesto ágil son los que realmente modifican la manera de trabajar. Si el equipo no tiene en claro los conceptos esenciales del manifiesto ágil, difícilmente entenderán las razones por las que es aconsejable seguir alguna práctica, y a la larga pueden dejar de utilizarla o utilizarla a medias. Si el equipo entiende los fundamentos básicos de estas metodologías ágiles, va a ser mucho más fácil la adopción de nuevas propuestas de prácticas ágiles ya que entenderán las razones por las que se deben seguir y perdurará su utilización en el tiempo.

Si no se entiende el valor, por ejemplo, las reuniones de retrospectiva, cuando el equipo esté con poco tiempo puede no realizarlas. Pero la reflexión y adaptación son una piedra fundamental dentro de las metodologías ágiles. A intervalos de tiempo el equipo debe reflexionar sobre la forma de trabajo. La mejora continua solo puede alcanzarse cuando se hace una pausa para reflexionar sobre lo que funciona bien y lo que no, y tomar una decisión consciente de ajustar las prácticas. Pequeños ajustes pueden hacer la diferencia entre el éxito y el fracaso de un equipo.

En esta propuesta particular, se buscó poder ayudar a los líderes actuales de proyecto a brindarles ciertas ideas de cómo encuadrar sus proyectos de desarrollo basado en conocimiento dentro de las metodologías ágiles.

1. Se planteó un ciclo de vida bastante general para el proyecto en tres etapas.
2. Se planteó trabajar con flujo de trabajo continuo, dando la posibilidad de definir iteraciones pero sin imponerlas.
3. A lo largo de la etapa de producción se van construyendo los prototipos de DBC y se realizan entregas al cliente.
4. Se definieron los posibles roles que podrían distribuirse dentro del equipo y considerando también al cliente, siempre teniendo en cuenta la situación relevada.
5. Se definieron artefactos para trabajar, que la mayoría de las metodologías plantean:
    a. Historias de usuario para especificar funcionalidades, o modificaciones a realizar

b. Una lista priorizada de trabajo, que contiene las funcionalidades ítems que se desean implementar.

c. Una lista de tareas por especialistas, cuando éste trabaje en varios proyectos simultáneamente.

d. Entrega de prototipos DBC que se instalan luego en entorno real

6. Se propuso el uso de una pizarra Kanban para la gestión de actividades del equipo, pudiendo especificarse en ella tareas de diferentes proyectos. Los especialistas, al trabajar en varios proyectos de varios equipos, necesitan otra pizarra Kanban para administrar sus tareas
7. Se propuso el uso de herramientas para realizar pruebas automatizadas.
8. Se propuso mantener las reuniones que mayoría de las metodologías ágiles vienen sosteniendo: reuniones de planificación, de seguimiento y de finalización, junto con otras reuniones.
9. Se impulsó el uso de los prototipos de DBC que pueden generarse rápidamente para obtener retroalimentación del cliente, que por el estudio de campo realizado casi no era utilizado.
10. Se consideró importante poder definir dentro del equipo que se entiende por *terminado* (DoD) en cada etapa de proceso
11. Se decidió sugerir solo un pequeño número de prácticas ágiles para intentar implementar, sugiriendo que en las reuniones de reflexión se puede analizar ir incorporando el resto.
12. Se planteó alternativas para suplir las ausencias de los stakeholders en las reuniones

Se buscó realizar la aplicación de esta propuesta trabajando con dos equipos de alumnos del último año de la Licenciatura en Análisis de Sistemas a dos proyectos específicos con desarrollo basado en conocimiento utilizando Genexus. Se obtuvieron buenos resultados y surgieron algunos detalles a considerar, como por ejemplo encontrar alternativas a la presencia de los stakeholders en las reuniones para conocer su visión o la opinión sobre el producto o la presencia del cliente en reuniones de ajuste. La propuesta debe ampliarse brindando alternativas en caso de no ser viable su participación, para cada situación. A continuación se detallan alternativas que se incorporan a la misma que permiten suplir estas ausencias:

- Permitir a los clientes informar cambios de manera no presencial, por algún medio: correo, sistema de tickets reservando la presencia de los mismos para situaciones muy complejas. Para evitar malos entendidos en estas situaciones, se podría buscar generar un prototipo rápido que permita clarificar lo solicitado y llevarlo al lugar de trabajo, o subirlo a algún servidor de prueba para recibir opinión al respecto.
- Visitar a los stakeholders en su ámbito laboral o bien solicitar a los stakeholders completar una encuesta, reconociendo la perdida de cierta información que solo se logra en la comunicación cara a cara, sobre todo al definir la visión del proyecto:
    - que permita determinar la visión de cada stakeholder si es imposible que asistan a la reunión inicial
    - que permita determinar la opinión de cada stakeholder sobre el producto desarrollado si no pueden asistir a la reunión de reflexión final.

También se planteó la propuesta obtenida a algunos sectores de desarrollos relevados, que manifestaron interés en los resultados que se pudieran arribar para analizarlos. Solo respondió uno de ellos, quien consideró que sería viable su aplicación, pero también hizo notar la imposibilidad de poder contar con la presencia de stakeholders o clientes en las reuniones.

En resumen, se considera que la propuesta cumple con la intención de acercar a equipos de desarrollo basado en conocimiento hacia las prácticas ágiles. Si bien existen detalles a mejorar

permite que los equipos comiencen a trabajar de una manera más participativa, incluso con el cliente. Es una propuesta de prácticas que es viable de aplicar, con las salvedades mencionadas anteriormente. Es, simplemente, un puntapié inicial en el camino hacia el desarrollo ágil, permitiendo de una manera sencilla ir realizando el cambio cultural que es tan difícil e importante de lograr. Paulatinamente el equipo en las reuniones de reflexión interna irá refinando su forma de trabajo. Tal como manifiesta Cockburn, el equipo a medida que adquiera más experiencia adoptará nuevas prácticas.

> *Agile es más sobre personas, interacciones y cultura que sobre procesos, prácticas y herramientas. Mientras las prácticas ágiles son importantes y proveen lineamientos prácticos que ayuden a aumentar el éxito del proyecto, los principios son lo que hace a las metodologías ágiles sustentables a largo plazo y maximiza los grandes beneficios que tiene para ofrecer. [*Kindon 2007]

Esp. Loraine Gimson

Mg. Gustavo Daniel Gil

Dr. Gustavo Rossi

6 BIBLIOGRAFÍA

6.1 METODOLOGÍAS ÁGILES

[Abrahamsson 2002] Abrahamsson Pekka, Salo Outi, Ronkainen Jussi & Juhani Warsta. *Agile software development methods- Review and analysis.* ESPOO 2002 - VTT Publications 478 – Universidad de Oulu. ISBN 951-38-6009-4. http://www.vtt.fi/inf/pdf/publications/2002/P478.pdf

[AgilAlliance web] Agile Alliance, http://www.agilealliance.org/. Accedido: 10/10/2011.

[Agile web] *Principios del Manifiesto Ágil*. http://agilemanifesto.org/iso/es/principles.html. Acceso:10/10/2011.

[AgileData web] Agile Data. Sitio de Amber Scott. Sitio: http://www.agiledata.org/. Accedido: 10/02/2012.

[AgileJournal web] Revista electrónica sobre metodologías de desarrollo de software. Agile Journal, nº1, Marzo-2006. Http:// www.agilejournal.com. Accedido 05/12/2011.

[AgileSpain web] Agil Spain. Sitio: http://www.agilspain.com. Accedido: 15/03/2012.

[Anderson web] Anderson David. Sitio Oficial. Sitio: http://agilemanagement.net/index.php. Accedido: 10/12/2011.

[Balduino 2007] Balduino Ricardo. Año 2007. *Introduction to OpenUP (Open Unified Process)*. Sitio:http://www.eclipse.org/epf/general/OpenUP.pdf Accedido: 08/02/2012.

[Beedle web] Beedle Mike. Sitio oficial de Mike Beedle. Sitio: http://www.Xbreed.org. Accedido: 12/09/2011

[Beck 2000] Beck Kent. *Extreme Programming Explained: Embrace Change* Boston Addison Wesley. 2000.

[Bonilla 2010] Bonilla David. *Historias de Usuario vs. Casos de Uso*. Fecha: Febrero 2010. Sitio: http://sixservix.com/blog/david/2010/02/08/historias-de-usuario-casos-de-uso/. Accedido: 14/11/2011

[Bonilla 2010b] Bonilla David. *Estructura de una Historia de Usuario*. Fecha: Febrero 2010. Sitio: http://sixservix.com/blog/david/2010/02/10/estructura-historia-usuario/. Accedido: 14/11/2011

[Burrows 2010] Burrows, M. *Learning together: Kanban and the Twelve Principles of Agile Software*. Junio de 2010. Sitio: http://positiveincline.com/index.php/2010/06/learning-together-Kanban-and-the-twelve-principles-of-agile-software/. Accedido 11/06/2011.

[Caine 2012] Caine. Matthew *DSDM Atern –The biggest Agile Secret?* Bern – Swiss – ICS Scrum breakfast, 25 de enero 2012. Sitio: http://www.mcpa.biz/2011/11/dsdm-atern-%E2%80%93-the-biggest-agile-secret-%E2%80%93-bern-swissict-scrumbreakfast-january-25th-2012/. Accedido 08/02/2012.

[Caine 2011] Mathew Caine.*DSDM Atern.* Sitio Web de M.C. Partners & Associates – Crafting sustainable agility. Agosto/Octubre 2011. Sitio: http://www.mcpa.biz/2011/10/dsdm-atern-project-structure-overview/; http://www.mcpa.biz/2011/10/dsdm-atern-principals-overview/; http://www.mcpa.biz/2011/10/dsdm-atern-roles-and-responsibilities-an-overview/; http://www.mcpa.biz/2011/10/dsdm-atern-products-overview/; http://www.mcpa.biz/2011/10/an-overview-of-dsdm-atern/; http://www.mcpa.biz/2011/08/what-is-dsdm-atern/; http://www.mcpa.biz/2011/09/agile-project-management/. Accedido: 20/02/2012.

[Calero 2003] Solís Manuel Calero. *Una explicación de la programación extrema (XP)*. V Encuentro usuarios xBase. Madrid. 2003. www.willydev.net/descargas/prev/Explicaxp.pdf

[Canós 2003] Canós José H., Letelier Patricio y Penadés Mª Carmen. *Metodologías Ágiles en el desarrollo de software*. Universidad Politécnica de Valencia. Sitio: http://www.willydev.net/descargas/prev/TodoAgil.Pdf. - 2003.

[CaraballoMaestre 2009] Caraballo Maestre, Alejandro. Scrum para Dummies. Desarrollo Software. Sitio: http://caraballomaestre.blogspot.com/2009/05/scrum-para-dummies.html. Accedido: 05/05/2011

[Carvajal 2008] Carvajal Riola, José Carlos. Metodologías ágiles: herramientas y modelo de desarrollo para aplicaciones java ee como metodología empresarial. Tesis Final de Máster. Septiembre 2008 - Sitio: http://upcommons.upc.edu/pfc/bitstream/2099.1/5608/1/50015.pdf

[CnetNews] Barnes Cecily - *More programmers going "Extreme".* CNet News. Abril 2001. Sitio: http://news.cnet.com/2100-1040-255167.html. Accedido: 02/01/2012.

[Cockburn 2002] Cockburn, Alistair - *Agile Software Development.* Editorial: Pearson Edication Inc. Stoughton, MA. Estados Unidos. Año 2002 .ISBN:0-201-69969-9

[Cockburn 2004] Cockburn, Alistair. *Crystal Clear A Human-Powered Methodology for Small Teams*. Editorial: Addison Wesley Professional. Estados Unidos. Año 2004. ISBN: 0-201-69947-8

[Cockburn 2007] Cockburn, A. Agile *Software Development The cooperative Game*. Second Edition. Alistair Cockburn. Addison Wesley. ISBN:0-321-48275-1 Año 2007 Boston. USA. (Fourth printing, Agust 2009)

[Cockburn web] Cockburn, Alistair - Sitio oficial de Alistair Cockburn. Sitio: http://alistair.cockburn.us/ Accedido: 12/09/2011.

[Crisp web] Tjänster - Sitio: http://www.crisp.se/Kanban. Último acceso: 03/02/2012.

[Cunningham 1986] Cunningham, W. and Beck, K.: *A Diagram for Object-Oriented Programs* in Proceedings of OOPSLA-86. Octubre 1986.

[Deemer 2010] Deemer Peter, Benefield Gabrielle, Larman Craig, Vodde Bas. *Scrum Primer.* Versión 1.2. Año 2010. Sitio: http://www.goodagile.com/scrumprimer/scrumprimer.pdf. Accedido: 10/11/2011

[Deemer 2009] Deemer Peter, Benefield Gabrielle, Larman Craig, Vodde Bas. Traducción al castellano de Leo Antoli. *Básica de Scrum* (*Scrum Primer).* Scrum Training Institute. Versión 1.1. Año 2009. Sitio: www.scrumalliance.org/resource_download/2481. Acceso: 10/11/2011

[DeSeta web] De Seta, L. *Los 7 principios del desarrollo Lean*. Sitio Web Dos Ideas. Sitio: http://www.dosideas.com/metodologias/410-los-7-principios-deldesarrollo-lean.html. Accedido: 10/09/2011.

[DSDM web] DSDM Consortium. Sitio oficial. Sitio: http://www.dsdm.org/. Accedido: 08/02/2012

[DSDM Ebook] - *DSDM Atern Enables More Than Just Agility* - Infonic AG, Zurich, Switzerland -- M.C. Partners and Associates, Zurich, Switzerland Abril 2011. Sitio http://www.dsdm.org/wp-content/uploads/2011/12/DSDM-Enables-More-Than-Just-Agility.pdf. Accedido: 08/02/2012.

[Eclipse web] *Dynamic Systems Development Method (DSDM) plugin for OpenUP*. DSDM Consortium. Año 2006. Sitio: http://www.eclipse.org/epf/downloads/configurations/pubconfig_downloads.php. Accedido: 10/02/2012.

[Ferreiras 2010] Ferreiras, M. - Abril de 2010. Sitio: http://es.scribd.com/doc/30651405/Curso-de-Conceptualizacion-Manufactura-Lean-Ud-1. Accedido: 02/06/2011.

[Ferrer 2003] Ferrer Jorge. *Metodologías Ágiles*. Publicado en diciembre de 2003. Sitio http://libresoft.dat.escet.urjc.es/html/downloads/ferrer-20030312. Accedido: 02/06/2011.

[Fowler 2003a] Fowler, M. *The new methodology.* Trabajo. © Copyright Martin Fowler. Actualización Año 2003. Sitio: http://martinfowler.com/articles/newMethodology.html. Actualización Año 2003. Accedido 02/10/2011

[Fowler 2003b] Fowler, M. *La nueva metodología.* Trabajo. © Copyright Martin Fowler. Traducción Alejandro Sierra. Sitio: http://www.programacionextrema.org/articulos/newMethodology.es.html. Actualización 2003. Accedido 10/10/2011

[Fowler 2005] Fowler, M. *Ponga su proceso a dieta*. Trabajo. 2005. ThougthWorks, Inc. Sitio: http://www.thoughtworks.com/

[Fowler 2006a] Fowler, M. Sitio personal de Martin Fowler. Artículo: *Is Design Dead?* Sitio: www.martinfowler.com/ Accedido 10/10/2011

[Fowler 2006b] Fowler, M., Foemmel M. Sitio personal de Martin Fowler. Artículo: *Continuous Integration*. Sitio:www.martinfowler.com/ Accedido 10/10/2011

[Fowler 2006c] Martin Fowler, Sitio personal de Martin Fowler. Artículo: *Continous Integration*. Mayo 2006. Sitio: http://martinfowler.com/articles/continuousIntegration.html. Accedido: 10/02/2012

[Fowler 2006d] Martin Fowler. Sitio personal de Martin Fowler. Artículo: *Test-Driven Development*. Mayo 2006. Sitio: http://martinfowler.com/bliki/TestDrivenDevelopment.html. Accedido: 10/02/2012

[Frankel 2004] Frankel David. Abril 2004. sitio: http://www.lcc.uma.es/~av/MDD-MDA/publicaciones/p_2704-04%20COL%20MDSD%20Frankel%20-%20Bettin%20-%20Cook.pdf. Accedido: 08/02/2012.

[Giro 2002] Giro Rodolfo, Leiva Fernando, Mesa Oscar y Pinciroli Fernando. *Consideraciones acerca de XP*. UNLP. Año 2002

[Gurley 2008] Gurley Greg y Oster Nate de ATSC. *Applying Agile OpenUp in a New Team: A Case Study.* Trabajo presentado en junio de 2008 en Rational Software Conference Presentations. Sitio: http://www.atsc.com/documents/briefings/20080605-atsc-agileopenup-presentation.pdf. Accedido: 08/02/2012.

[Hartmann 2008] Interview with Hakan Erdogmus by Deborah Hartmann. *Hakan Erdogmus on TDD Misunderstandings and Adoption Issues*. Julio 2008. http://www.infoq.com/interviews/TDD-Hakan-Erdogmus. Accedido: 12/12/2012.

[Higsmith 2002] Highsmith Jim. *Agile Software Development Ecosystems*. AddisonWesley. 2002

[IBM web] Koll, Ben. OpenUP In a Nutshell. Septiembre 2007. Sitio: http://www.ibm.com/developerworks/rational/library/sep07/kroll/. Accedido: 10/02/2012.

[Jeffries 2007] Jeffries Ron , Melkin Grigori *TDD: The Art of Fearless Programming.* Mayo/Junio 2007 (Vol. 24, No. 3) pp. 24-30 © 2007 IEEE. Published by the IEEE Computer Society. Sitio: http://www.computer.org/csdl/mags/so/2007/03/s3024.html Accedido: 10/02/2012.

[Jeffries web] Sitio de Ronald E. Jeffries: *An Agile Software Development Resource*. http://www.xprogramming.com/, Accedido: 10/12/2011.

[Joyce 2009] Joyce, D. *Pulling Value: Lean and Kanban*. 27 de Junio de 2009. Recuperado el 9 de Junio de 2011, de Systems Thinking, Lean and Kanban: Sitio:http://leanandKanban.files.wordpress.com/2009/06/pulling-value-lean-and-Kanban.pdf.

[Kindon 2007] Kindon, S., Pain, R., Kesby, M.: Participatory Action Research Approaches and Methods. Abingdon, Routledge. Año 2007

[Kniberg 2010] Kniberg, H., & Skarin, M. *Kanban vs Scrum: obteniendo lo mejor de ambos*. 2010. Madrid, España: Proyectalis.

[Letelier 2003] Leteiler Patricio et. al., *An Experiment Working with RUP and XP, Extreme Programming and Agile Processes in Software Engineering*, 4th International Conference, 2003.

[Letelier 2006] Letelier Patricio, Penadés Mª Carmen - *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* Depto. de Sist. Informáticos y Computación.Universidad Politécnica de Valencia. 2006. Sitio: http://www.willydev.net/descargas/masyxp.pdf o bien Sitio: http://www.cyta.com.ar/ta0502/v5n2a1.htm Accedido: 13/10/2011

[LimitedWipSociete web] Limited WIP Society. The Home of Kanban Systems for Software Engineering. Sitio: http://www.limitedwipsociety.org/ Accedido: 03/01/2012.

[Linden 2011] Linden Reed Janice. Quotable Kanban. *Sabiduría recolectada de los participantes de Kanban Leadership Retreat*. Reykjavik, Iceland, 2011. Sitio: http://agilemanagement.net/index.php/site/quotable/. Accedido: 11/12/2011.

[Lizeth 2008] Lizeth Torres Flores, Carmina & Harvey Alférez Salinas, Germán - *Establecimiento de una Metodología de Desarrollo de Software para la Universidad de Navojoa Usando OpenUP* - Departamento de Sistemas, Universidad de Navojoa, A.C.& Facultad de Ingeniería y Tecnologia, Universidad de Montemorelos. Reporte Técnico año 2008

[ManifiestoAgil web] Beck, K., et.al, *Manifesto for Agile Software Development*, Sitio:http://agilemanifesto.org/.

[MendesCalo 2008] Mendes Calo, K., Estevez, E. Fillottrani, P., *Un Framework para Evaluación de Metodologías Ágiles* CACIC 2008 Sitio:http://www.egov.iist.unu.edu/cegov/content/download/ /1878/47500/version/8/file/Un+Framework+para+Evaluacion+de+Metodologias+Agiles.pdf.

[MendesCalo 2010] Mendes Calo, K., Estevez, E. Fillottrani, P., *Evaluación de Metodologías Ágiles para Desarrollo de Software*, WICC 2010 - XII Workshop de Investigadores en Ciencias de la Computación

[Middleton 2012] Peter Middleton David Joyce. *Lean Software Management BBC Worldwide - Case Study.* IEEE Transactions on Engineering Management. Febrero 2012. Sitio: http://leanandKanban.wordpress.com/. Accedido: 03/02/2012.

[Modezki 2011] Modezki, M. *Kanban para Gestión de Proyectos y otras Aplicaciones*. 30 de Marzo de 2011. Sitio: http://www.cafeproyectos.com/gestion-de-proyectos/Kanban-para-gestion-de-proyectos-y-otras-aplicaciones/. Accedido 01/06/2011.

[Navegapolis web] Palacios Bañeres Juan. *¿Qué es DSDM?* Navegapolis.net. Sitio: http://www.navegapolis.net/content/view/361/59/. Accedido: 08/02/2012.

[OPENUP web] OpenUP. Sitio Oficial. http://epf.eclipse.org/wikis/openup/index.htm Acceso: 10/02/2012.

[PalacioBañeres 2008] Palacio Bañeres, Juan. *Navegapolis 2008*. Edición Navegápolis® Febrero 2008. http://www.navegapolis.net. Accedido: 05/04/2011

[PalacioBañeres 2010] Palacios Bañeres Juan - Navegapolis 2010. Edición Navegapolis. Febrero 2010. e-book. Sitio:http://www.navegapolis.net/files/navegapolis_2010.pdf. Descargado: 08/02/2012.

[Parasuraman 2011] Parasuraman Narayan - *Understanding and Managing Agile Transitions.* Julio 2011. Sitio: http://www.modernanalyst.com/Resources/Articles/tabid/115/articleType/ArticleView/ articleId/ 1926/Understanding-and-Managing-Agile-Transitions.aspx. Accedido: 15/12/2011.

[Poole 2009] Damon B. Poole, *Do It Yourself Agile*, Septiembre, 2009 Sitio:http://www.accurev.com/whitepaper/pdf/Do-It-Yourself-Agile.pdf. Accedido: 08/08/2011.

[Poole 2009b] Damon B. Poole, *Do It Yourself Agile - Agile Development Material for the DIY'er* Sitio:http://damonpoole.blogspot.com/. Accedido: 08/08/2011

[Prince2 web] *"PRINCE2 in Spanish"*. Sitio:http://jlfr-prince2.blogspot.com/. Accedido: 04/04/2012.

[Programación-extrema] sitio web miembro de la Agile Aliance. *Programación extrema*. Sitio: Http:// www.programacionextrema.org. Accedido 05/12/2011.

[ProyectosAgiles web] Proyectos ágiles. Sitio http://www.proyectosagiles.org/historia-de-scrum. Accedido: 11/11/11.

[Rassmunsson 2009] Rasmusson, J., The Agile Samurai: How Agile Masters Deliver Great Software. PragmaticBookshelf. Año 2009

[Rubio 2009] Rubio Jimenez, M.A. *Kanban: el método Toyota aplicado al software*. Junio 2009. Sitio: http://bosqueviejo.net/2009/06/22/Kanban-el-metodo-toyota-aplicadoal-software/. Accedido: 10/09/2011.

[Sanders web] Sanders Aaron. *Constantly changing*. Sitio: http://aaron.sanders.name/agile-fashion/naked-planning-explained-Kanban-in-the-small. Accedido: 10/12/2011.

[ScrumAlliance web] Sitio oficial del Grupo de profesionales para compartir trabajo con Scrum. Sitio: www.Scrumalliance.org. Accedido: 10/09/2011. Directores: Chairman Mike Cohn, Steve Fram, Treasurer Dan Hintz, Michele Sliger, Harvey Wheaton, Scott Duncan, and Mitch Lacey.

[Schmidkonz 2007] Schmidkonz Christian, CSM & Jürgen Staader, SAP AG, Germany. Noviebre 2007. "*Piloting of Test-driven Development in Combination with Scrum*" http://www.scrumalliance.org/resources/267. Accedido: 10/12/2012.

[Schwaber web] Schwaber Ken. Sitio oficial de Ken Schwaber. Sitio:www.controlchaos.com/. Accedido: 10/09/2011.

[Shore web] Shore, James. *Kanban-systems.* Sitio: http://jamesshore.com/Blog/Kanban-Systems.html. Actualizado: 15 de octubre de 2008. Accedido: 10/12/2011.

[Stapleton 1997] Stapleton, Jennifer. *DSDM Dynamic Systems Development Method – The method in practice.* DSDM Consortium. Edison Wesley Professional. 1997

[Sutherland web] Sutherland Jeff. Sitio oficial de Jeff Sutherland. Sitio: http://www.jeffSutherland.org. Accedido: 12/09/2011

[Sutherland 2011] Sutherland Jeff & Schwaber Ken. *The Scrum Papers: Nut, Bolts and Origins of an Agile Framework.* Enero 2011. París.

[ThoughtWorks web] Barnes, Cecily *More Programmers Going "Extreme"* CNET News. 3 abril 2001. Sitio:http://news.cnet.com/2100-1040-255167.html. Accedido: 09/10/ 2011.

[Turley 2010] Turley Frank. *El Modelo de Procesos PRINCE2®. Una magnífica introducción a PRINCE2*. Traducido por José Luis Fernández Ramírez. Año 2010.

[Vega 2010] Vega Frank X. *SCRUM, XP, and beyond: One Development Team's Experience Adding Kanban to the Mix.* Proceedings of lean software & systems conference. Atlanta. EEUU. 2010. Descargado del Sitio http://agilemanagement.net. Accedido: 12/12/2011.

[VersionOne web] VersionOne. *Agile Made Easier*, *8th Annual State of Agile Development Survey* – Results 2013 - VersionOne Inc . 2014. Descargado del Sitio: http://www.versionone.com/. Accedido: 01/10/2014.

[XP 2011] *Agile Processes in Software Engineering and Extreme Programming*. 12th International Conference, XP 2011- Madrid, Spain, Mayo de 2011. Proceedings. ISBN 978-3-642-20676-4.

[XP web] Sitio Extreme Programming. http://www.extremeprogramming.org/ Accedido: 08/07/2011.

6.1.1 Referencias Bibliográficas

[Anderson 2010] Anderson D.J. *Kanban: Successful Evolutionary Change for Your Technology Business.* Blue Hole Press (2010)

[Ambler web] Ambler, Scott W.. *When Is a Model Agile?.* Fecha: sin especificar. Sitio: http://www.agilemodeling.com/essays/whenIsAModelAgile.htm

[Astels 2003] David Astels. *Test-Driven Development. A practical guide*. Prentice Hall PTR 2003.

[Beck 1995] K. Beck, *Request for SCRUM Information"* J. Sutherland, Ed. Boston: Compuserve, 1995.

[Beck 1999] K. Beck, *Extreme Programming Explained: Embrace Change*. Boston: Addison-Wesley, 1999.

[Beck 2003] K. Beck, *Test Driven Development: By Example. Año 2002.* Addison-Wesley Professional. ISBN-10: 0321146530 | ISBN-13: 978-0321146533

[Bohem 1976] Boehm, B. *Software Engineering.* IEEE Transactions on Computers, 1226-1241, 1976.

[Bohem 2002] Boehm, B. (2002) *Get ready for the agile Methods, with Care*. Computer 35(1):64-69

[Cockburn 1999] Cockburn, A., *Characterizing People as Non-Linear, First-Order Components in Software Development.* Octubre 1999.

[Cockburn 2001] Cockbun, A. *Agile Software Development*. Addison-Wesley. 2001.

[Ehn 1990] Ehn P. - *Work-Oriented Design of Software Artifacts* – Editorial L. Erlbaum Associates Inc. Hillsdale, NJ, USA 1990 Año 1990. ISBN:0805807810

[Fowler 2005] Fowler, M., Beck, K., Brant, J. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley. 1999

[eWorkshop 2002] eWorkshop. Resumen del segundo workshop en metodologías ágiles. Sitio: http://fc-md.umd.edu/projects/agile/secondeWorkshop/summary2ndeworksh.htm. Año 2002.

[Glass 2000] Glass, R.L. (2001) *Agile Versus traditional: Make Love, not War!* Cutter IT Journal 14(12):12-18

[Glass 2002] Glass, R. L., *Facts and Fallacies of Software Engineering*. 2002.

[Hawrysh 2002] Hawrush, S. & Ruprecht, J. (2000). *Light Methodologies: It's like Déjà Vu All over again*. Cutter IT Journal. 13:4-12.

[Highsmith 2001] Highsmith, J & Cockburn A. (2001). *Agile Software Development: The Business of Innovation*. Computer 34(5):120-122.

[IABG web] IABG, *The V-Model*, Sitio:http://www.vmodell.iabg.de/.

[Jacobson 1999] Jacobson, I., et al, *The Unified Software Development Process*. Addison-Wesley. 1999

[Jeffires 2001] Jeffries, R., Anderson, A., Hendrickson, C. *Extreme Programming Installed*. Addison-Wesley. 2001

[Kroll 2003] Kroll, P. and Kruchten, P. *The Rational Unified Process Made Easy.* Addison Wesley, 2003.

[Kroll 2006] Per Kroll, Bruce Macisaac. *Agility and Discipline Made Easy: Practices from OpenUP and RUP*. 2006. Addison-Wesley.

[Larman 2004] Craig Larman. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*. Tercera Edición. 2004. Prentice Hall.

[Martin 2003] Martin, Robert C. *Agile Software Development, Principles, Patterns, and Practices*, Año 2002. Prentice Hall., ISBN-10: 0135974445. ISBN-13: 978-0135974445.

[McConnell 2003] Mc Connell, S., *Professional Software Development: Shorter Schedules, Higher Quality Products, More Successful Projects, Enhanced Careers.* Addison Wesley, 2003.

[McCourley 2001] McCourley, R. (2001) *Agile Development Methods Poised to Upset Status Quo*. SIGCSE Bulletin 33(4):14-15

[Miller 2001] Miller, D & Lee, J. (2001) *The people make the process: commitment to employees, decision making, and performance*. Journal of Managment 27:163-189

[Pfeffer 1998] Pfeffer, J. (1988) *The human equation: building profits by putting people first*. Boston. MA. Harvard bussiness School Press.

[Ohno 1988]. Ohno T.: *Just-In-Time for Today and Tomorrow*. Productivity Press (1988)

[Poppendieck 2003]. Poppendieck, M., Poppendieck, T.: *Lean software development: An agile toolkit.* Addison Wesley, Boston (2003)

[Ridderstrale 2000] Ridderstrale, J., et al, *Business:Talent Makes Capital Dance*. Prentice Hall, EEUU, 2000.

[Schwaber 1997] K. Schwaber, *Scrum Development Process* in *OOPSLA Business Object Design and Implementation Workshop*, J. Sutherland*, et al.*, Eds., London: Springer, 1997.

[Schwaber 2001] Schwaber K., Beedle M., Martin R.C. *Agile Software Development with SCRUM*. Prentice Hall. 2001.

[Sutherland 2004] J. Sutherland. *Agile Development: Lessons Learned from the First Scrum.* Cutter Agile Project Management Advisory Service: Executive Update, vol. 5, pp. 1-4, 2004.

[Thomas 2005] Thomas, Dave y Heinemeier Hansson, David. 2005. *Agile Web Development with Rails (2nd edition).* s.l. : The Pragmatic Programmers , 2005.

[Womack 1991] Womack, J.P., Jones, D.T., Roos, D. *The Machine That Changed the World: The Story of Lean Production*. Harper Business, New York (1991)

6.2 BASE DE DATOS DEL CONOCIMIENTO

[Almeida 2008] Almeida Enrique, Araújo Alejandro, Larre Borges Uruguay. *Proyecto GxUnit - Construcción de una herramienta para automatización de pruebas unitarias en GeneXus.* Congreso

Argentino de Ciencias de la Computación (CACIC2008) Universidad Nacional de Chilecito. La Rioja. Argentina.

[Alonso 2004] Alonso Betanzos Amparo. *Ingeniería del conocimiento: Aspectos metodológicos*, Pearson Education. ISBN: 9788420541921 Año 2004.

[Alvarado 2010] Artículo *Gobierno garantizó la migración de las bases de datos del Estado.* Hormiga analítica – Marcaibo-Venzuela –Julio 2010- Año 2 Nro 56 Editor Heberto Alvarado Vallejo CNP 12270

[ANSI-SPARC] ANSI-SPARC: Final report of the ANSI/X3/SPARC DBS-SG relational database task group (portal ACM, citation id=984555.1108830)

[Artech 2008] *Visión General de Genexus.* Artech Consultores S.R.L. . Uruguay. Octubre 2008. Sitio:http://www.genexus.es/wp-content/uploads/2010/06/vision_general_gx_esp2009.pdf. Accedido:05/08/2013.

[Balrer 1983] R Balrer, et all. *Software Technology in the 1990's using a New Paradigm*" *Computer,* Noviembre 1983, pp 19-45

[Borlawsky 2009] Borlawsky Tara B. et al. *Development of an Agile Knowledge Engineering Framework in Support of Multi-Disciplinary Translational Research*. Año 2009. Sitio web NCBI: National Center for Biotechnology Information de: PMC3041563. Sitio: http://www.ncbi.nlm.nih.gov/pmc/articles/ PMC3041563/#b6-amia-s2009-14. Fecha publicación: 1/3/2009. Fecha acceso: 04/04/2015

[Bracci 2008] Bracci Luigino. *YVKE mundial radio.* Gobierno Bolivariano de Venezuela / Ministerio de Poder Popular para la comunicación y la información Últimas Noticias (Heberto Alvarado), YVKE Mundial. Publicado 24/02/2008. Sitio: http://www.radiomundial.com.ve/yvke/noticia.php?3323. Accedido: 05/08/2013.

[CNTI web] *Centro nacional de Tecnología de información*. Gobierno Bolivariano de Venezuela / Ministerio de Poder Popular para la comunicación y la información. Sitio oficial: http://www.cnti.gob.ve/index.php?option=com_content&view=article&id=191&Itemid=63. Accedido: 09/09/2011.

[Genexus web] Sitio oficial de Genexus. Sitio:http://www.genexus.com/. Accedido: 05/07/2014.

[GenexusLibre web] *GENEXUS LIBRE. Proyecto Altagracia CNTI* - Langecor Proyecto Administrador de proyectos. Embajada de la republica Bolivariana de Venezuela en la Republica Oriental del Uruguay. Ministerio de Poder Popular para Relaciones exteriores. Sitio: http://173.83.84.126/buscar.aspx // http://173.83.84.126/ficha.aspx?id=68. Accedido: 16/09/2011.

[Giro 2002] Giro Rodolfo, Leiva Fernando, Mesa Oscar y Pinciroli Fernando. *Consideraciones acerca de XP.* Administración de Proyectos. UNLP. 2002

[Gonda 1995] Gonda B.,Jodal N. *Proyecto GeneXus*. Academia Nacional de Ingeniería, Uruguay, 1995 Sitio:http://www.aiu.org.uy/gxpsites/agxppdwn?2,1,4,O,S,0,79%3BS%3B1%3B21. Accedido: 16/09/2011.

[Gonda 2003] Gonda Breogán. *¿Desarrollo orientado a procesos u orientado a datos? Algunas reflexiones en el 40° aniversario de los Sistemas de Gerencia de Bases de Datos*. Artech. 2003

[Gonda 2007] Gonda Breogán y Jodal Nicolás. *Desarrollo Basado En Conocimiento - Filosofía Y Fundamentos Teóricos De Genexus.* Artech. Mayo de 2007. Sitio:http://www.genexus.es/wp-content/uploads/2010/06/desarrollo_basado_en_el_conocimiento.pdf. Accedido: 16/09/2011.

[Gonda 2010] Gonda, Breogán & Jodal, Nicolás. *GeneXus: Filosofía*. Artech Consultores S. R. L. Última actualización: 2010.

[GxUnit web] GxUnit: *GxUnit1 y GxUnit2*, Sitio: http://www.gxopen.com/gxopenrocha/servlet/hproject?721, 2008. Accedido: 15/07/2014.

[IEEE web] Andrew J. Symonds, IBM Data System Division. *Creating a Software Engineering Knowledge Base*. IEEExplore. Digital Library. Año 1988. Sitio:http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=2011. Accedido: 16/09/2011.

[Knublauch 2002] Holger Knublauch. *An Agile Development Methodology for Knowledge-Based Systems Including a Java Framework for Knowledge Modeling and Appropriate Tool Support.* PhD Thesis, Universität Ulm, Alemania. 2002

[Latorres 2003] Latorres E.,Salvetto P., Larre Borges U., Nogueira, J. *Una herramienta de apoyo a la gestión del proceso de desarrollo de software.* IX Congreso Argentino de Ciencias de la Computación (CACIC 2003). La Plata, Argentina.

[MárquezLisboa 2007] Márquez Lisboa Daniel, Fernández Cecilia. *Genexus Rocha – Episodio Uno.* Editorial Grupo Magro. Montevideo, Uruguay. 2007

[Salvetto 2006] Salvetto P. *Modelos Automatizables de Estimación muy Temprana del Tiempo y Esfuerzo de Desarrollo de Sistemas de Información.* Tesis de Doctorado. Univ. Politécnica de Madrid. Doctorado conjunto en ingeniería informática UPM-ORT. Año: 2006. Sitio: http://oa.upm.es/367/01/PEDRO_SALVETTO_LEON.pdf. Accedido: 16/09/2011.

6.2.1 Referencias bibliográficas

[Almeida 2003] Almeida E. *En el proceso de crear un framework de testing*. Año: 2003. Sitio: http://ealmeida.blogspot.com/2003/10/estoy-en-el-proceso-de-crear-un.html.

[Almeida 2004] Almeida E. *Software Testing: Tres enfoques para un mismo problema.* Encuentro Internacional de Usuarios GeneXus del año 2004.

[Almeida 2006a] Almeida E.,Araújo A.,Larre Borges U. *Proyecto colaborativo GxUnit. Año:* 2006. Sitio: http://www.gxopen.com/commwiki/servlet/hwiki?GxUnit,

[Almeida 2006b] Almeida E. ,Araújo A.,Larre Borges U. Collaborative Projects. Año: 2006 http://www.concepto.com.uy/archivosvinculados/EncuentroGX2006CollaborativeProjects.ppt

[Araujo 2008] Araujo Perez, Alejandro. *Especificación de un marco de pruebas asociados a Genexus con adaptación de funcionalidades FIT.* Tesis de Maestría en Ingeniería de Computación. Instituto de Compución. Facultad de Ingeniería, Universidad de la República, Montevideo, Uruguay. 2008

[Cunningham 1986] Cunningham W. *Framework for Integrated Tests. Año:* 2007. Sitio: http://fit.c2.com/

[GxUnit web] Proyecto de Ingeniería de Software 2007. *GxUnit.* Año 2007. Docente: Triñanes J. Facultad de Ingeniería, Universidad de la República, Uruguay.

[JUnit web] JUnit, sitio: http://junit.org, 2008.

[Mugridge 2005], Mugridge R., Cunningham W. *Fit for Developing Software: Framework for Integrated Tests.* Año: 2005. ISBN 0-321-26934-9. Prentice Hall PTR.

[WikiGenexus web] GeneXus Community Wiki, *GeneXus Rocha/GXextensions.* Año: 2007. Sitio: http://wiki.gxtechnical.com/commwiki/servlet/hwiki?category%3AGeneXus+Extensions

7 ANEXO1: ENCUESTAS SOBRE DESARROLLO BASADO EN CONOCIMIENTO CON GENEXUS

7.1 ESTRUCTURA DE ENCUESTA

7.1.1 Encuesta en blanco

La siguiente encuesta tiene el único propósito de sondear la experiencia en Salta Capital con Desarrollo Basado en Conocimiento con Genexus y su forma de trabajo. La información relevada se utilizará de manera anónima en un trabajo de tesis de maestría de Ingeniería de Software de la UNLP y en un proyecto de investigación del CIDIA dependiente de la Facultad de ciencias Exactas de la UNSa.

Fecha de realización de la encuesta:

DATOS GENERALES

1. Tipo de empresa: **PÚBLICA / PRIVADA**
2. Tiempo en el mercado:
3. Tiempo de uso de Genexus:
4. Cantidad de proyectos desarrollados con Genexus:
5. Proporción de Proyectos para terceros: (100 – 0%)
6. Cantidad de personas involucradas:
7. La empresa cuenta con más de un equipo de desarrollo: **SI – NO**

   Cantidad de personas de cada equipo
8. La empresa trabaja en más de un proyecto a la vez: **SI – NO**
9. Los equipos trabajan en más de un proyecto a la vez: **SI – NO**
10. Las personas involucradas comparten el horario de trabajo: **SIEMPRE –SE SOLAPAN – NUNCA**. Explique: ……..
    ……
11. Las personas involucradas comparten el lugar de trabajo: **SI – NO**. Especifique cómo se suple la falta de lugar común: **VIDEO CONFERENCIA – CHAT – TELEFONO – CORREO – MEMO – REUNIONES PACTADAS – OTRAS……..…**
12. Existe un líder de proyecto: **SI – NO**
13. El líder de proyecto es responsable de más de un proyecto a la vez: **SI – NO**

    En caso afirmativo, indique cantidad de proyectos que suele manejar al mismo tiempo: ……..
    ……

FORMA DE TRABAJO CON GENEXUS

1. Tiene un procedimiento fijo a seguir para cada desarrollo **SI – NO**
2. Si es afirmativo, puede describirlo. ……
   ……
   ……
   ……
   ……

……
……

3. Cada equipo puede adaptar el procedimiento según el proyecto **SI – NO – DEPENDE (especificar)……**
4. El procedimiento a seguir en cada proyecto de desarrollo es definido: **POR LA DIRECCIÓN / POR EL LÍDER DE PROYECTO /POR EL EQUIPO**
5. El cliente está involucrado: **SOLO AL INICIO / DURANTE TODO EL PROCESO / AL FINAL**

   Especifique. ……
   ……
6. Se realizan entregas frecuentes (a lo sumo cada 2 meses) de producto desarrollado al cliente. **SI – NO.**

   Especifique el tiempo: ……
7. Se tiene colaboración permanente con el cliente. **SI – NO.**

   Explique cómo: ……
   ……
8. Los requerimientos se definen al inicio del proyecto: **SI – NO**

   Especifique COMO define los requerimientos ……
   ……
9. Los requerimientos cambian a lo largo del proyecto**: SI SIEMPRE – ALGUNAS VECES – CASI NUNCA – No CAMBIAN**
10. Se realiza un desarrollo iterativo**: SI – NO**. Indique duración de la iteración…….
11. La planificación del desarrollo se realiza

    AL INICIO
    AL INICIO PERO SE VA MODIFICANDO ENCADA ITERACION –
    OTRA OPCION (especificar)……..

    Especifique Como se planifica……...
    …… ……
12. Los alcances y costos se pactan con el cliente:

    AL INICIO
    PARA CADA ITERACION
    OTRA OPCION (especificar)

    Especifique COMO se definen los alcances……..……
    …… ……
13. Se realiza un diseño utilizando diagramas UML: **SI – NO.**

    Especifique cuales utiliza ……...
    ……..
    ……..... ……..……
14. Se sigue una metodología ágil: **SI – NO**

    En caso afirmativo, especifique cuál ……..……..

    En caso negativo, tiene pensado incorporar alguna**: SI – NO**.

    Especifique cuál incorporaría o la razón por la que no lo considera necesario o apropiado

…… ……

…… ……

15. Especifique cuáles de las siguientes prácticas ágiles se siguen en sus proyectos

- Programación de a pares
- Refactorización
- Diseño simple
- Reuniones diarias
- Pruebas automatizadas
- Iteraciones cortas
- Entregas frecuentes
- -----

16. Especifique herramientas utilizadas para realizar algunas de las prácticas ágiles seleccionadas. ……
……

17. Especifique herramientas utilizadas para el seguimiento del proyecto:

- -
- -
- -

18. Utiliza un diagrama Burn Down para seguimiento del proyecto. **SI – NO.**

19. Se desarrolla con reutilización. **SI – NO**. Especifique el porcentaje de reutilización de proyectos anteriores. ¿Cómo se seleccionan los artefactos a reutilizar?
……
……

20. Especifique el porcentaje de proyectos donde se cumplieron con los objetivos pactados con el cliente. ……

21. Cuando surgen dificultades, y no se logran los objetivos de la iteración pactados,

SE TRABAJAN HORAS EXTRAS –
FINALIZA IGUALMENE LA ITERACION EN FECHA PACTADA
SE EXTIENDEN LA ITERACION
OTRAS ACCIONES (especificar)……..

22. Especifique el nivel de satisfacción del cliente frente al producto desarrollado.

COMPLETAMENTE SATISFECHO
MUY SATISFECHO,
CONFORME,
CON LEVES DISCONFORMIDADES,
TOTALMENTE DISCONFORME

23. Identifique las situaciones de riesgo más comunes detectadas en los proyectos. Numerarlas según su importancia, siendo 1 el riesgo más importante:

- Imprecisiones del cliente
- Variaciones de los requerimientos del cliente
- Poca participación del cliente
- Equipo de desarrollo desmotivado
- Equipo de desarrollo involucrado en varios proyectos
- Otras (especificar)……..

24. ¿Por qué trabaja con Genexus?

……
……

……
……

25. Indique si Utiliza o piensa utilizar a futuro

- GXServer: No Utilizo – Utilizo – Pienso utilizar
- GXFlow: No Utilizo – Utilizo – Pienso utilizar
- GXQuery: No Utilizo – Utilizo – Pienso utilizar

26. Otros comentarios:

……
……

……
……

Muchas gracias por su colaboración.

7.2 ENCUESTAS COMPLETAS

7.2.1 Encuesta 1

Fecha de realización de la encuesta: Marzo 2014

DATOS GENERALES

1. Tipo de empresa: Pública / **Privada**
2. Tiempo en el mercado: **4 años**
3. Tiempo de uso de Genexus: **10 años**
4. Cantidad de proyectos desarrollados con Genexus: **>100**
5. Proporción de Proyectos para terceros: (100 – 0) **100%**
6. Cantidad de personas involucradas: **2. A veces se incorpora un programador extra (o asesor de diseño gráfico)**
7. La empresa cuenta con más de un equipo de desarrollo: **NO**
8. La empresa trabaja en más de un proyecto a la vez: **SI**
9. Los equipos trabajan en más de un proyecto a la vez: **SI**
10. Las personas involucradas comparten el horario de trabajo: **NUNCA**. Explique: **por diferentes actividades uno por mañana otro por tarde**
11. Las personas involucradas comparten el lugar de trabajo: SI – NO. Especifique cómo se suple la falta de lugar común: **VIDEO CONFERENCIA – CHAT – TELEFONO – CORREO – MEMO – REUNIONES PACTADAS**
12. Existe un líder de proyecto: **SI**
13. El líder de proyecto es responsable de más de un proyecto a la vez: **SI**. En caso afirmativo, indique cantidad de proyectos que suele manejar al mismo tiempo 3 o 4

FORMA DE TRABAJO CON GENEXUS

1. Tiene un procedimiento fijo a seguir para cada desarrollo **SI** . Si es afirmativo, puede describirlo.

Se realizan entrevistas iniciales y se realiza un mapa conceptual (una persona con cierta capacidad).
Se realiza un análisis de negocio (discusión en conjunto).
Si es desde cero
Discusión de diseño
Se cuantifica el tiempo según cantidad de objetos y acciones (basado en artefactos hechos)
Se modulariza y se asigna
No se personaliza el 99%
El jefe es el que reparte los módulos y toma el control en contexto general
Si hay reuso
Se retoca lo utilizado
Nota: tiene una tabla ordenada de objetos

2. Cada equipo puede adaptar el procedimiento según el proyecto **SI**

3. El procedimiento a seguir en cada proyecto de desarrollo es definido: **por el líder de proyecto**
4. El cliente está involucrado: **durante todo el proceso** Especifique. **Diálogo desde la primera entrega**
5. Se realizan entregas frecuentes (a lo sumo cada 2 meses) de producto desarrollado al cliente. **SI** – NO. Especifique el tiempo: **40% sin cliente solo programan, después todos los días y se estabiliza**
6. Se tiene colaboración permanente con el cliente. **SI**. Explique como: **el cliente al ver el potencial del desarrollo, se le ocurren otras cosas nuevas**
7. Los requerimientos se definen al inicio del proyecto: **SI** – NO. Especifique COMO define los requerimientos: **Los requerimientos generales se definen al principio a través de un documento de requerimiento representado con mapas conceptuales usando Free mind(no buscan definir los alcances totales pero sí conceptualizar). Se comparte con el cliente y el cliente coloca burbujas en el modelo sin inconvenientes**
8. Los requerimientos cambian a lo largo del proyecto: **SI SIEMPRE**
9. Se realiza un desarrollo iterativo: **SI** . Indique duración de la iteración. **No son fijos, 5 meses tal vez para el prototipo del 40% y después un mes**
10. La planificación del desarrollo se realiza: **AL INICIO PERO SE VA MODIFICANDO ENCADA ITERACION** – Especifique Como se planifica **Hay una cota no rígida de plazos (para buscar más ayuda)**
11. Los alcances y costos se pactan con el cliente: **AL INICIO y PARA CADA ITERACION.** Especifique COMO se definen los alcances. **Quedan abiertos los alcances. Hay un valor pactado de costo. También hay un período de garantía de 6 meses donde se pueden pedir reformas o pequeños listados**
12. Se realiza un diseño utilizando diagramas UML: **NO**. Especifique cuales utiliza **Solo mapas conceptuales que tienen diseño de tablas OO**
13. Se sigue una metodología ágil: **NO.** En caso negativo, tiene pensado incorporar alguna: **NO**. Especifique cuál incorporaría o la razón por la que no lo considera necesario o apropiado `**por desconocimiento de casos de aplicación con éxito**
14. Especifique cuáles de las siguientes prácticas ágiles se siguen en sus proyectos
    - **Programación de a pares (para probar cosas nuevas)**
    - **Refactorización**
    - **Reuniones diarias (reuniones semanales)**
    - **Entregas frecuentes**

No hay pruebas automatizadas, pero hay un beta tester, después del 40% y el cliente via web los verifica. GXtest simplifica el trabajo

15. Especifique herramientas utilizadas para realizar algunas de las prácticas ágiles seleccionadas: **Free mind, Team viewr (video conferencias), Ticket (módulo desarrollado tomado de herramientas open source php)**
16. Especifique herramientas utilizadas para el seguimiento del proyecto: **tickets**
17. Utiliza un diagrama Burn Down para seguimiento del proyecto. **NO**.
18. Se desarrolla con reutilización. **SI**. Especifique el porcentaje de reutilización de proyectos anteriores. ¿Cómo se seleccionan los artefactos a reutilizar? **50% (a futuro de modulo reutilización). Lo decide el líder. Se trata de serializar y optimizar y minimizar los sistemas a medida. No se pierde el control y diseño en Genexus**
19. Especifique el porcentaje de proyectos donde se cumplieron con los objetivos pactados con el cliente. **80%**

20. Cuando surgen dificultades, y no se logran los objetivos de la iteración pactados: **SE EXTIENDEN LA ITERACION**

21. Especifique el nivel de satisfacción del cliente frente al producto desarrollado: **MUY SATISFECHO,**

22. Identifique las situaciones de riesgo más comunes detectadas en los proyectos. Numerarlas según su importancia, siendo 1 el riesgo más importante:

- **Variaciones de los requerimientos del cliente (2)**
- **Equipo de desarrollo involucrado en varios proyectos (1)**

23. ¿Por qué trabaja con Genexus?

- ***Vocación orientada al desarrollo***
- ***Herramienta potente que sigue al desarrollador (usuario), es intuitivo. La sinergia (retroalimentación)***
- ***Los usuarios piden cosas a Genexus (un requerimiento) y los desarrolladores de Genexus lo incorporan en un nuevo upgrade***
- ***En castellano***
- ***Analizaron otras y no es lo mismo (maguie - powerbuilder)***

24. Indique si Utiliza o piensa utilizar a futuro

- GXServer: **Pienso utilizar**
- GXFlow: Pienso utilizar **tal vez**
- GXQuery: **No Utilizo**

25. Otros comentarios: *Al trabajar con patterns, permite saber:*

- ***Complejidad y tiempo según una tabla armada después de tanta experiencia***
- ***Dónde realizar modificaciones. Ya que todo se mantiene lo mas estándar posible***
- ***El trabajar con Genxus es más rápido y estándar***
- ***Uso Licencia anual 18000$***

7.2.2 Encuesta 2

Fecha de realización de la encuesta: Abril 2014

DATOS GENERALES

1. Tipo de empresa: **Privada**

2. Tiempo en el mercado**: 7 años**

3. Tiempo de uso de Genexus: **7 años**

4. Cantidad de proyectos desarrollados con Genexus: **10 grandes "algunas replicas"**

5. Proporción de Proyectos para terceros: (100 – 0) **99% (1 interno de calidad de gestión)**

6. Cantidad de personas involucradas: **el núcleo son 4. Hoy hay 9, otras veces de 15 hasta 25. Mucha rotación del personal. Hay 80% Personal part - time**

7. La empresa cuenta con más de un equipo de desarrollo: **SI**. Cantidad de personas de cada equipo**: 3. Dividen equipos por tareas (UNO: análisis, especificación, diseño, pruebas – OTRO: documentación – OTRO: desarrollo y testing). Hay personas en varios equipos a la vez**

8. La empresa trabaja en más de un proyecto a la vez: **SI**

9. Los equipos trabajan en más de un proyecto a la vez: **SI**

10. Todas las personas involucradas comparten el horario de trabajo: **NUNCA**. Explique: **mucha gente no comparte los horarios de trabajo, otros sí.**

11. Las personas involucradas comparten el lugar de trabajo: **SI** . Especifique cómo se suple la falta de lugar común: **REUNIONES PACTADAS --Antes daban posibilidad de trabajar desde la casa pero no cumplían)**

12. Existe un líder de proyecto: **SI**. **También existen especialistas por área (contable, etc)**

13. El líder de proyecto es responsable de más de un proyecto a la vez: **SI**. En caso afirmativo, indique cantidad de proyectos que suele manejar al mismo tiempo: **4 proyectos. La empresa trata de asegurar continuidad de trabajo para sostenerse en el mercado y poder afrontar los costos de existencia, por eso un solo proyecto no es viable**

FORMA DE TRABAJO CON GENEXUS

1. Tiene un procedimiento fijo a seguir para cada desarrollo **SI**. Si es afirmativo, puede describirlo. ***SI***

Se trato en un principio de definir rigurosamente un procedimiento formal siguiendo Iso 9001. La empresa busca seguir sus lineamientos, y CMM buscando procedimientos formales con mejoras, etc. Luego lo fueron adaptando y haciendo más flexible pero mantienen muchos registros.

- ***Realizan Análisis de campo, entrevistas a usuarios y análisis de procesos y normativas***
- ***Se busca que se puede adaptar de lo ya desarrollado (no hay un registro, depende mucho de los jefes de proyecto)***
- ***Según lo que hay que adaptar y desarrollar el líder escribe las HU***
- ***Se presenta a consideración del cliente y se realizan modificaciones sugeridas***
- ***"todo se pide por órdenes de servicio" que escribe el cliente(sacado del PMI) permite aclarar el alcance***
- ***Las estimaciones se realizan en base a la experiencia histórica de la empresa y es bastante acertada***
- ***Se pasa a desarrollo, se prueba y se entrega por módulos***
- ***Se desarrolla primero los módulos más importantes para el cliente***
- ***Se entrega periódicamente (mensualmente) un documento "informe de avance (cuenta a los clientes que pagan mes a mes la evolución del proyecto. Sirve también internamente para saber cómo va el plan de ejecución.***
- ***Existen encargado de las pruebas***

Se puede decir que se hace la propuesta inicial…lo antes mencionado, luego……registra la implementación, la capacitación, informe de avance mensual, orden de servicio y orden de entrega. También hacen seguimiento de incidentes (errores que aparecen una vez entregado al cliente o "errores según cliente que no son errores reales")

Siempre incluyen capacitación, El registro de capacitación: fecha, hora, las personas capacitadas y temario, es fundamental, dado que los usuarios no están motivados para participar, no tienen interés, por falla de las empresas y luego suelen tener reclamos la empresa (mayoría de clientes abogados, se cubren las espaldas con documentación)

2. Cada equipo puede adaptar el procedimiento según el proyecto. **DEPENDE.** (especificar) **Siempre los dueños están involucrados en los equipos y son ellos los que modifican, no hay autonomía de los grupos.**

3. El procedimiento a seguir en cada proyecto de desarrollo es definido**: por la dirección**.

4. El cliente está involucrado: **durante todo el proceso**. Especifique. A veces el mayor servicio es post-implementación. Está involucrado dado que puede realizar pedidos de servicio y recibe informes de avance del proyecto
5. Se realizan entregas frecuentes (a lo sumo cada 2 meses) de producto desarrollado al cliente. **SI**. Especifique el tiempo. Depende del proyecto y el módulo a desarrollar, no trabaja con iteraciones, sino por módulos
6. Se tiene colaboración permanente con el cliente. **SI**. Explique como**: utilizando órdenes de servicio y a través de los informes de avance**
7. Los requerimientos se definen al inicio del proyecto: **SI**. Especifique COMO define los requerimientos: **A través de HU y luego órdenes de servicio. En HU se hace énfasis en los puntos de verificación y a veces se detallan objetos /tablas/Estructuras de datos. Un analista luego toma los puntos de verificación para hacer las pruebas**
8. Los requerimientos cambian a lo largo del proyecto: **SI SIEMPRE**. Usan las órdenes de servicio para solicitar cambios los clientes
9. Se realiza un desarrollo iterativo: **NO**. Indique duración de la iteración No hay iteraciones, es evolutivo. Se va entregando al cliente los módulos más importantes y se continua desarrollando (en forma cíclica)
10. La planificación del desarrollo se realiza: **OTRA OPCION: AL INICIO PERO SE VA MODIFICANDO en el transcurso (no hay iteraciones).** Especifique Como se planifica: **El líder determina el tiempo de realización de los módulos y determina los responsables de las tareas asociadas, delega a cada desarrollador la tarea a realizar, todos siguen el dot Project.**
11. Los alcances y costos se pactan con el cliente: **AL INICIO.** Especifique COMO se definen los alcances: **Hay cuatro variables en juego: alcance, $, calidad y tiempo. Definen alcance, $ con el cliente (calidad es interna) y para eso los lideres determinan un tiempo con una cierta holgura. De llegar a sobrepasar ese tiempo se trata de renegociar, pero generalmente el cliente no quiere pagar más. Aún agregando nuevas funcionalidades con las órdenes deservicio**
12. Se realiza un diseño utilizando diagramas UML: **SI**. Especifique cuales utiliza: **Diagramas de CU solo para proyectos nuevos, no a reutilizar. Lo que pretenden hacer es definir una wiki describiendo ciertos productos ya desarrollados y su estructura así los nuevos desarrolladores fácilmente pueden aprender la forma de desarrollo y estilo del grupo**
13. Se sigue una metodología ágil: **NO.** En caso negativo, tiene pensado incorporar alguna: **SI**. Especifique cuál incorporaría o la razón por la que no lo considera necesario o apropiado **SCRUM, en parte**
14. Especifique cuáles de las siguientes prácticas ágiles se siguen en sus proyectos
    - **Refactorización**
    - **Diseño simple**
    - Reuniones diarias ***se hacen en determinados proyectos**
    - **Entregas frecuentes**
15. Especifique herramientas utilizadas para realizar algunas de las prácticas ágiles seleccionadas: **ninguna**
16. Especifique herramientas utilizadas para el seguimiento del proyecto: **dot project, sistema de tickets**
17. Utiliza un diagrama Burn Down para seguimiento del proyecto. **NO**. **Planean usar a futuro**
18. Se desarrolla con reutilización. **SI**. Especifique el porcentaje de reutilización de proyectos anteriores. ¿Cómo se seleccionan los artefactos a reutilizar**? 80%. El líder o el dueño de**

la empresa conocen lo desarrollado y determinan de dónde reutilizar. No hay una biblioteca para reutilización.

19. Especifique el porcentaje de proyectos donde se cumplieron con los objetivos pactados con el cliente. **90% en tiempo y forma. El 10% superó el tiempo**
20. Cuando surgen dificultades, y no se logran los objetivos de la iteración pactados, **SE TRABAJAN HORAS EXTRAS – OTRAS ACCIONES: no se prueba como se debería**
21. Especifique el nivel de satisfacción del cliente frente al producto desarrollado. **CONFORME**,
22. Identifique las situaciones de riesgo más comunes detectadas en los proyectos. Numerarlas según su importancia, siendo 1 el riesgo más importante:

- **Imprecisiones del cliente (4)**
- **Otras (especificar)**
  - **Desmotivación del usuario, nunca conforme (1)**
  - **Migración de datos por sus problemas asociados (cliente no entiende la inconsistencia de datos que podría haber tenido de antes latente y que ahora es evidente)(2)**
  - **Falta de validación interna cuando todo es urgente, se deja de probar (3)**
  - **Imprecisiones del cliente (punto 1) (4)**

23. ¿Por qué trabaja con Genexus?

- **Lograron Buenos resultados desde 1997. Comenzaron en esa época con un sistema grande y luego otro grande con éxito**
- **Tiene ciclo de aprendizaje corto**
- **Facilidad de mantenimiento del sistema**
- **Generación de código**
- **Buenos resultados rápidos**
- **Al trabajar con proyectos grandes por ahora es la solución**
- **La evolución con la tecnología (comenzaron desarrollando para AS400, pasaron por C++, ahora móviles, genexus se encarga de la base**

24. Indique si Utiliza o piensa utilizar a futuro

- GXServer: **Pienso utilizar**
- GXFlow: **Pienso utilizar**
- GXQuery: **Pienso utilizar**

25. Otros comentarios:

- **Hay lógica de negocio compleja o detalles de interfaz que se programa con c++**
- **El equipo no conoce todo. Solo parte**

7.2.3 Encuesta 3

Fecha de realización de la encuesta: 20 de agosto 2014

DATOS GENERALES

1. Tipo de empresa: **Privada**
2. Tiempo en el mercado: 13 años
3. Tiempo de uso de Genexus: más de 13 años
4. Cantidad de proyectos desarrollados con Genexus: 80

5. Proporción de Proyectos para terceros: (100 – 0) 98%. Se desarrollaron herramientas como instaladores e integradores (que permite integrar con software de báscula o cortadora de vidrios)
6. Cantidad de personas involucradas: 10 aprox. A veces más
7. La empresa cuenta con más de un equipo de desarrollo: **SI**. Cantidad de personas de cada equipo: **6 (4 programadores + analistas + jefe proyecto). Si el proyecto es pequeño (modificación de sistema – tal vez un programador + analista)**
8. La empresa trabaja en más de un proyecto a la vez: **SI**. **Si son proyectos largos (1 año o año y medio), solo 3 o tal vez 4 a la vez. Si son pequeños hasta 7**
9. Los equipos trabajan en más de un proyecto a la vez: **SI**. **Hay especialistas, por ejemplo en BD o para realizar conexiones a bajo nivel con otros dispositivos como cortadora de vidrios. Sobre todo ellos trabajan en varios. Asi tb los analistas.**
10. Las personas involucradas comparten el horario de trabajo: **SE SOLAPAN**. Explique: **Al principio se compartía horario de trabajo. Se trabaja freelance ahora, pero se fijan reuniones. A veces se hacen por skype. Hay alta rotación de personal**
11. Las personas involucradas comparten el lugar de trabajo: **NO Salvo cuando por diseño hay cosas que no se pueden dividir. Por ejemplo dos persona deben trabajar con los mismos objetos**. Especifique cómo se suple la falta de lugar común: **VIDEO CONFERENCIA** – **REUNIONES PACTADAS**
12. Existe un líder de proyecto: **SI**
13. El líder de proyecto es responsable de más de un proyecto a la vez: **SI**. En caso afirmativo, indique cantidad de proyectos que suele manejar al mismo tiempo **Un máximo de 5 en proyectos pequeños y solo 2 por líder de proyecto si son grandes**

FORMA DE TRABAJO CON GENEXUS

1. Tiene un procedimiento fijo a seguir para cada desarrollo **SI**. Si es afirmativo, puede describirlo.

PROYECTOS CORTOS: agregados de funcionalidad o modificaciones a un sistema que ya está funcionado

- ***Se realizan entrevistas.***
  - ***Si es corto tal vez es una sola entrevista y se resuelve ahí, se bosqueja las pantallas y se deja resuelto a nivel interfaz y reportes necesarios***
- ***Se arma prototipo respetando los diseños previos del proyecto (trabaja un solo programador)***
- ***Se presenta el prototipo al cliente sin detalles de interfaz***
  - ***Se puede ir personalmente***
  - ***Se puede presentar por web***
  - ***Se trabaja en los detalles de interfaz***

Se releva al cliente y se muestra el prototipo con la funcionalidad solicitada. Como es simple, no hay más interacción que la antes mencionada

PROYECTOS LARGOS: nuevos clientes, con varias funcionalidades

- ***Relevamiento: Se realizan entrevistas, comenzando a nivel gerencial. Se va bajando en los niveles del organigrama y se relevan las diferentes áreas involucradas, detectando los usuarios que más conocen las necesidades que no necesariamente son los jefes –identificación de stakeholders). Son varias entrevistas a varios niveles.***
- ***Se arma un DER o Diagrama de clases***

- ***Si es un sistema muy grande se divide en fases y se profundiza solo una de ellas (la que se entregará primero) y se presenta un presupuesto y plazo de entrega para esa fase. Dejando el resto pendiente y estimado a grandes rasgos por comparación con esa primera fase.***
- ***El líder (que además cumple rol de analista diseñador) identifica dentro de las librerías –a través de menú de módulos – aquellos módulos u objetos que pueden reutilizarse en el proyecto (por ejemplo seguridad, permisos, grupos, facturador, etc). Como todo se realiza con nomenclatura compatible, no se repiten los nombres y así se evitan conflictos. Por ejemplo, en un sistema se elige llamar usuarioDeSistema a los usuarios y dejar el nombre usuario para el usuario de tarjetas. Lo mismo sucede con los documentos, documentoDeVentas, documentoDeCompras,etc***
- ***El líder diseñador entonces después de elegir de lo desarrollado por procedimientos (funcionalidades) no por objetos,***
  - ***Define las pantallas –interfaces***
  - ***Define los datos deducidos***
  - ***Define de manera completa y muy detallada del Diccionario de Datos (DD). A tal punto de especificar como aparece cada dato en informes, en forms, etc***
- ***El líder presenta el proyecto al equipo en una reunión, donde explica los objetivos y las funcionalidades a implementar. Se genera muy poca documentación que sea práctica y que eventualmente pueda crecer pero que no permanezca desactualizadas. Se especifican las reglas de negocio mediante fichas asociadas a cada objeto. Esto es útil a la hora de reutilizar, para no ver todo el código genexus y comprender con estoa información la funcionalidad que cumple el objeto. Son los lideres los intermediarios con los clientes (similar a un PO)***
- ***El líder diseñador asigna tickets a cada uno de los programadores según sus capacidades y gustos. Se utilizan especialistas para tares específicas como conexiones a BD y creación de todos los store procedures o especialistas para realizar la conexión con dispositivos (como una cortadoras de vidrio). Generalmente los tickets no tienen un timpo asignado, salvo aquellos que tengan dependencia entre sí (estos tiempos se pactan charlando con los programadores dejando una pequeña holgura). Os tickets no duran mas de 2 semanas y se subdividen en tickets con pasos internos para lograrlo (implementar, prueba,etc)***
- ***Cada programador realiza los forms y webforms necesarios – realiza la implementación generando los prototipos. A veces el programador detecta datos no considerados previamente y se itera de manera cíclica con el líder diseñador). A veces hay req adicionales que solicita el programador a cliente, como webservices, de manera directa sin intermediación del líder diseñador.***
- ***Cada programador debe ir informando a través del sistema de tickets el % que trabajo que queda pendiente para terminar la tarea, así el líder puede hacer un seguimiento de la evolución del proyecto***
- ***Cada cierta cantidad de tickets se realiza una exposición de los programadores, donde cada uno expone al grupo (lo que hizo, los problemas que tuvo y como los solucionó). También sirve para que el líder detecte si semanalmente el programador entendió la funcionalidad a implementar.***
- ***Se realizan pruebas cruzadas, intercambian los módulos. En las fichas de documentación el líder tiene definido también casos de prueba y aceptación con los que se realizan las pruebas manuales. Se usó GXTest en un momento, pero el precio de la licencia se elevó y además al trabajar con sistemas que interactúan con otros sistemas, no sirve este CASE para ese tipo de pruebas. Además GXTest requiere mucha programación adicional a la hora de ir definiendo los objetos***
- ***Para segurar la calidad, el líder diseñador hace pruebas por muestreo del cgo escrito por los programadores, para ver si respeta las pautas preestablecidas, sobre todo con la nomenclatura ya que estos criterios son claves a la hora de hacer reusables los objetos de Base del Conocimiento de Genexus.***

- ***Se presenta el prototipo al cliente en una reunión presencial. En las dos primeras presentaciones se toma un tiempo para que los prototipos tengan una interfaz agradable, pero en posteriores presentaciones se aclara al cliente que para realizar más rápida las entregas no se tomará tiempo en la interfaz. Trabajan de esta forma porque al principio al cliente le importa mucho la apariencia, los colores, los botones, los menúes…al ver que están de la forma que les interesa después se avocan a ver las funcionalidades.***
- ***Se instala al final del plazo de la primera fase el sistema. Se busca que el sistema brinde un servicio que sea valorado por el usuario así se socializa y quiere utilizarlo y contagia al resto su entusiasmo.***

En un momento se pensó en delegar el diseño a otra persona que no sea el líder analista diseñador, pero no dio resultados. Para ellos es crítico que se siga un diseño respetando la nomenclatura y teniendo siempre en claro que todo se reutiliza. Les encantaría poder delegar, pero consideran que es una actividad muy delicada.

2. Cada equipo puede adaptar el procedimiento según el proyecto **NO** (especificar) **Existen dos formas de encararlo para proyectos simples (de clientes viejos) y para proyectos nuevos (grandes)**
3. El procedimiento a seguir en cada proyecto de desarrollo es definido: **por la dirección / por el líder de proyecto. La dirección y líder de proyecto son las mismas personas**
4. El cliente está involucrado: **durante todo el proceso**. Especifique:
   - **En proyectos cortos se releva al inicio y se presenta prototipo sin interfaz y luego con la interfaz. Tres contactos. Pudiendo reducirse solo a dos, presentando ya el prototipo con la interfaz.**
   - **En proyectos largos hay muchos momentos. Y se va al lugar a mostrar el producto "hermoseado"**
5. Se realizan entregas frecuentes (a lo sumo cada 2 meses) de producto desarrollado al cliente. **SI**. Especifique el tiempo. **NO hay un tiempo fijo. Se divide el proyecto en partes o fases, pero se trata de entregar en no más de dos meses al cliente una fase. Generalmente primero se entregan transacciones con algunos pequeños reportes esenciales y en lo posible también conexiones a webservices (porque esto evita que usuario cargue en dos lugares diferentes los mismos datos – por ejemplo AFIP). Es decir busca solucionar problemas y detectar los usuarios que los necesitan. Luego reportes estadísticos gerenciales. Para organizar las entregas, el usuario en las entrevistas suele enfatizar lo que más le pesa en su área y esto lo detecta el líder diseñador. Esto es lo que trata de entregar lo antes posible. De esta forma el usuario está entusiasmado con el sistema y ve que no solo es carga de datos sino que ya le brinda cierta información y le soluciona algunas cuestiones ahorrándole trabajo. Este entusiasmo se contagia a otros futuros usuarios que comienzan a anhelar tener su versión del sistema para dejar de hacer por ejemplo doble carga de datos en algunos sistemas.**
6. Se tiene colaboración permanente con el cliente. **SI**. Explique como: **si hay dudas se los consulta, pero se trata de minimizar, porque al cliente no le gusta ser interrumpido en su trabajo**
7. Los requerimientos se definen al inicio del proyecto: **SI**. Especifique COMO define los requerimientos: **Con entrevistas se detectan las necesidades y se fracciona en fases o partes si es muy grande. Solo se va por los detalles de una etapa, la que el cliente considere más importante y que deba implementarse primero**
8. Los requerimientos cambian a lo largo del proyecto: **SI.** Salvo proyectos cortos
9. Se realiza un desarrollo iterativo: **SI**. Indique duración de la iteración: **La duración es variable no fija**

10. La planificación del desarrollo se realiza: **AL INICIO PERO SE VA MODIFICANDO ENCADA ITERACION.** Especifique Como se planifica **Se planifican al inicio las grandes etapas del proyecto (se divide en partes – etapas – fases). Se planifica con detalle solo la primera fase. Finalizada esta, se planifica la siguiente**

11. Los alcances y costos se pactan con el cliente: **PARA CADA ITERACION.** Especifique COMO se definen los alcances. **Al inicio se hace un estimativo general "aproximado". Ya que solo se conoce en detalle lo relacionado con la primera etapa a implementar y no las siguientes, supongo que serán "similares". Pero esto puede variar. En una reunión se define los costos y alcances de la primera fase (que debe cumplirse) sin poder luego negociar el precio pactado. Si el cliente quiere un consto total de todo el proyecto, se explica que la empresa demorará aproximadamente 2 meses en relevar toda la información, realizar un análisis fino y documentar. Esta actividad tiene como resultado una especificación completa y costos detallados, pero la actividad en sí tiene un costo aparte. Ningún cliente quiso realizarla**

12. Se realiza un diseño utilizando diagramas UML: **SI**. Especifique cuales utiliza **diagrama de clases. Pero también se usa DER y DD**

13. Se sigue una metodología ágil: **NO.** En caso negativo, tiene pensado incorporar alguna: **SI**. Especifique cuál incorporaría o la razón por la que no lo considera necesario o apropiado. **Se realizan prácticas ágiles pero no una metodología. Se realizan reuniones entre líderes o reuniones diarias a través de skype. No tiene definido una MA**

14. Especifique cuáles de las siguientes prácticas ágiles se siguen en sus proyectos

- **Refactorización: tienen herramienta que toma por ejemplo módulo y al cambiarle nombre cambia nombre de todas las variables. Herr desarrollada por ellos**
- **Diseño simple**
- **Reuniones diarias: por skype**
- **Iteraciones cortas: variables**

15. Especifique herramientas utilizadas para realizar algunas de las prácticas ágiles seleccionadas: **Skype, pizarras, tickets**

16. Especifique herramientas utilizadas para el seguimiento del proyecto: **tickets que además generan GANTT y Burndown chart y Google Desktop ayuda a encontrar módulos a reutilizar**

17. Utiliza un diagrama Burn Down para seguimiento del proyecto. **SI**.

18. Se desarrolla con reutilización. **SI**. Especifique el porcentaje de reutilización de proyectos anteriores. **90% Todo, salvo lo muy específico del proyecto** ¿Cómo se seleccionan los artefactos a reutilizar? **Usando google desktop. Se armó una librería o catálogo con módulos de proyectos. Por ejemplo seguridad, facturador, conectividad a entidades gubernamentales (como afip). Se desarrolla siguiendo una nomenclatura preestablecida.**

En un proyecto hay tres niveles: objetos (muy específicos), núcleos del proyecto (que se utiliza menos detallado) y núcleo de proyectos (mas generales). Se reutiliza a nivel 2.

19. Especifique el porcentaje de proyectos donde se cumplieron con los objetivos pactados con el cliente**. 98%**

20. Cuando surgen dificultades, y no se logran los objetivos de la iteración pactados: **SE TRABAJAN HORAS EXTRAS – a veces si por ejemplo la demora es por falta de experiencia del programador (freelance) y SE EXTIENDEN LA ITERACION.** Debido al seguimiento con tickets, se detecta rápidamente las demoras. Sobre todo se hace seguimiento de tickets críticos. En caso de deberse a una mala estimación se charla con el cliente y se explica que de cumplirse con la fecha de entrega, no funcionará de manera completa, caso contrario se debe correr el plazo de entrega, por lo que se extiende la iteración

21. Especifique el nivel de satisfacción del cliente frente al producto desarrollado.

xxxCOMPLETAMENTE SATISFECHO
xxMUY SATISFECHO,
xCONFORME,

22. Identifique las situaciones de riesgo más comunes detectadas en los proyectos. Numerarlas según su importancia, siendo 1 el riesgo más importante:

- **4 Imprecisiones del cliente**
- **1 Variaciones de los requerimientos del cliente**
- **0 Poca participación del cliente (siempre buscan que participe sino no lo toman)**
- **4 Equipo de desarrollo desmotivado (solo en proyectos muy largos)**
- **2 Equipo de desarrollo involucrado en varios proyectos**
- **3 Otras (especificar) Problema del empalme de las partes reutilizadas. A veces la decisión de reutilizar algunos módulos termina necesitando demasiado re-trabajo que hubiera sido menos costoso desarrollarlo desde cero**

23. ¿Por qué trabaja con Genexus? **Requiere mínimo personal y permite reutilización para desarrollo s de sistemas a medida. En poco tiempo evoluciona el prototipo**

24. Indique si Utiliza o piensa utilizar a futuro

- GXServer: **No Utilizo - Utilizó un tiempo***
- GXFlow: **Utilizo**
- GXQuery: **No Utilizo**

***Es muy costoso actualmente. Cuando hay dos programadores que trabajan con objetos muy dependientes, se los hace ir a trabajar a las oficinas compartiendo lugar físico y trabajan con programación con dummies (Objeto A interactúa con Objeto B) Cada programador crea los métodos de su Objeto, pero no con su detalle sino solo la cáscara, son métodos huecos. De manera que el otro objeto pueda interactuar con él aunque no esté completamente implementado.**

25. Otros comentarios:

- **Hablando todo se resuelve, es valido para el equipo y para el cliente.**
- **Se analizaron otros case como Magik y Clarion, pero se optó por Genexus porque permite reutilización de código**
- **Actualmente detectan problemas con la parte móvil. Es mentira que se puede usar directamente en plataforma móvil. Si algo tiene interfaz web requiere redefinir los objetos. No es tan directo. Además tablets y móvil tampoco son iguales. Están pensando adquirir otra herramienta para generar la interfaz**

7.2.4 Encuesta 4

DATOS GENERALES

1. Tipo de empresa: **Pública**

2. Tiempo en el mercado**: No forma parte del mercado por ser un área dependiente del Gobierno Provincial.**

3. Tiempo de uso de Genexus: **2 años**

4. Cantidad de proyectos desarrollados con Genexus**: 5 - Fueron realizados por personas que ya no integran el grupo de desarrollo.**

5. Proporción de Proyectos para terceros: (100 – 0) **0%**

6. Cantidad de personas involucradas: **3**

7. La empresa cuenta con más de un equipo de desarrollo: **NO.** Cantidad de personas de cada equipo: **3**
8. La empresa trabaja en más de un proyecto a la vez: **SI**
9. Los equipos trabajan en más de un proyecto a la vez: **SI**
10. Las personas involucradas comparten el horario de trabajo: **SIEMPRE.** Explique: …
11. Las personas involucradas comparten el lugar de trabajo: **SI** . Especifique cómo se suple la falta de lugar común: -----
12. Existe un líder de proyecto: **SI**
13. El líder de proyecto es responsable de más de un proyecto a la vez: **SI. El líder del proyecto no se dedica solamente a proyectos de desarrollo de software**. En caso afirmativo, indique cantidad de proyectos que suele manejar al mismo tiempo: **2 a 3**

FORMA DE TRABAJO CON GENEXUS

1. Tiene un procedimiento fijo a seguir para cada desarrollo **NO.**
2. Cada equipo puede adaptar el procedimiento según el proyecto **SI.**
3. El procedimiento a seguir en cada proyecto de desarrollo es definido: **por el líder de proyecto**
4. El cliente está involucrado: **durante todo el proceso.** Especifique**. Nuestros clientes son las áreas internas de nuestro organismo, se establece contacto con los representantes y usuarios internos las veces que resulta necesario a lo largo de todo el proceso.**
5. Se realizan entregas frecuentes (a lo sumo cada 2 meses) de producto desarrollado al cliente. **SI**. Especifique el tiempo: **Quincenal - mensual.**
6. Se tiene colaboración permanente con el cliente. **SI.** Explique como**: a través de entrevistas.**
7. Los requerimientos se definen al inicio del proyecto: **SI.** Especifique COMO define los requerimientos**: Se realizan reuniones con el encargado del área destinataria del software para establecer las necesidades. Se definen los requerimientos en base a esas necesidades y realizando un estudio del sistema de información in situ, a ello se agregan los que se desprenden de aspectos legales y de seguridad de la información.**
8. Los requerimientos cambian a lo largo del proyecto: **SI** .
9. Se realiza un desarrollo iterativo: **SI**. Indique duración de la iteración: **7 a 14 días**
10. La planificación del desarrollo se realiza: AL INICIO PERO SE VA MODIFICANDO ENCADA ITERACION – Especifique Como se planifica: …..
11. Los alcances y costos se pactan con el cliente: **No se definen costos. El alcance se define al inicio pero se va delimitando en forma más precisa durante las iteraciones.** Especifique COMO se definen los alcances: **Los alcances se definen teniendo en cuenta las funciones específicas del área para el cual se desarrolla la aplicación, la normativa interna que la regula y el marco legal.**
12. Se realiza un diseño utilizando diagramas UML: **NO**.
13. Se sigue una metodología ágil **NO.** En caso negativo, tiene pensado incorporar alguna: **SI**. Especifique cuál incorporaría o la razón por la que no lo considera necesario o apropiado. **Se piensa incorporar una metodología, se analizará implementar en el futuro SCRUM o DSDM**.
14. Especifique cuáles de las siguientes prácticas ágiles se siguen en sus proyectos

- -----

15. Especifique herramientas utilizadas para realizar algunas de las prácticas ágiles seleccionadas**. No se utilizan aún herramientas específicas.**

16. Especifique herramientas utilizadas para el seguimiento del proyecto:

No se utilizan aún herramientas específicas.

17. Utiliza un diagrama Burn Down para seguimiento del proyecto. **NO.**

18. Se desarrolla con reutilización. **SI** Especifique el porcentaje de reutilización de proyectos anteriores**. 20%** ¿Cómo se seleccionan los artefactos a reutilizar?----

19. Especifique el porcentaje de proyectos donde se cumplieron con los objetivos pactados con el cliente**. 75%**

20. Cuando surgen dificultades, y no se logran los objetivos de la iteración pactados, **SE EXTIENDE LA ITERACION**

21. Especifique el nivel de satisfacción del cliente frente al producto desarrollado. **MUY SATISFECHO,**

22. Identifique las situaciones de riesgo más comunes detectadas en los proyectos. Numerarlas según su importancia, siendo 1 el riesgo más importante:

    1. **Variaciones de los requerimientos del cliente**
    2. **Poca participación del cliente**
    3. **Imprecisiones del cliente**
    4. **Equipo de desarrollo involucrado en varios proyectos**
    5. **Equipo de desarrollo desmotivado**

23. ¿Por qué trabaja con Genexus? **Nuestro organismo realizó la compra de licencia de Genexus. Además es una herramienta muy potente para el desarrollo de aplicaciones que permite realizar las iteraciones con el cliente con mayor frecuencia.**

24. Indique si Utiliza o piensa utilizar a futuro

    - GXServer: **Pienso utilizar**
    - GXFlow: **Pienso utilizar**
    - GXQuery: **No Utilizo**

25. Otros comentarios:

Si bien contamos con la herramienta desde hace unos años, nos resultó difícil involucrar a los cargos superiores para poder conformar un grupo de personas que se dedique solamente a programar, siendo un grupo en formación ya que la mayoría recién está capacitándose en el uso de Genexus. Paulatinamente se aplicarán las metodologías ágiles en la programación.

7.2.5 Encuesta 5

Fecha de realización de la encuesta: julio 2014

DATOS GENERALES

1. Tipo de empresa: **Privada**
2. Tiempo en el mercado: **7 años**
3. Tiempo de uso de Genexus: **6 años**
4. Cantidad de proyectos desarrollados con Genexus: **2**
5. Proporción de Proyectos para terceros: (100 – 0) **70%**

6. Cantidad de personas involucradas:**25**
7. La empresa cuenta con más de un equipo de desarrollo: **SI.** Cantidad de personas de cada equipo **7**
8. La empresa trabaja en más de un proyecto a la vez: **SI**
9. Los equipos trabajan en más de un proyecto a la vez: **SI**
10. Las personas involucradas comparten el horario de trabajo: **Dos Turnos de trabajo**
11. Las personas involucradas comparten el lugar de trabajo: **SI** –
12. Existe un líder de proyecto: **SI**
13. El líder de proyecto es responsable de más de un proyecto a la vez: **SI**

En caso afirmativo, indique cantidad de proyectos que suele manejar al mismo tiempo **5 Proyectos**

FORMA DE TRABAJO CON GENEXUS

1. Tiene un procedimiento fijo a seguir para cada desarrollo **NO**
2. Cada equipo puede adaptar el procedimiento según el proyecto ------
3. El procedimiento a seguir en cada proyecto de desarrollo es definido **por el líder de proyecto**
4. El cliente está involucrado: **durante todo el proceso**. Especifique: --------.
5. Se realizan entregas frecuentes (a lo sumo cada 2 meses) de producto desarrollado al cliente. **NO**.
6. Se tiene colaboración permanente con el cliente. **SI**. Explique como**: informes de avance y órdenes de servicio**
7. Los requerimientos se definen al inicio del proyecto: **SI** Especifique COMO define los requerimientos: ------
8. Los requerimientos cambian a lo largo del proyecto: **SI SIEMPRE**
9. Se realiza un desarrollo iterativo: **NO**. Indique duración de la iteración
10. La planificación del desarrollo se realiza: **OTRA OPCION:Al inicio pero se va modificando (no hay iteración)**
11. Los alcances y costos se pactan con el cliente: **AL INICIO.** Especifique COMO se definen los alcances: -----
12. Se realiza un diseño utilizando diagramas UML: **SI**. Especifique cuales utiliza **Casos de USO**
13. Se sigue una metodología ágil: **NO.** En caso negativo, tiene pensado incorporar alguna: **NO**. Especifique cuál incorporaría o la razón por la que no lo considera necesario o apropiado**. Se debe estudiar más al respecto**
14. Especifique cuáles de las siguientes prácticas ágiles se siguen en sus proyectos

- -----

15. Especifique herramientas utilizadas para realizar algunas de las prácticas ágiles seleccionadas: ----------
16. Especifique herramientas utilizadas para el seguimiento del proyecto: **sistema de tickets, dot Project**
17. Utiliza un diagrama Burn Down para seguimiento del proyecto. **NO**.

18. Se desarrolla con reutilización. **SI**. Especifique el porcentaje de reutilización de proyectos anteriores. ¿Cómo se seleccionan los artefactos a reutilizar? **50%. Especifica el líder por catálogo**
19. Especifique el porcentaje de proyectos donde se cumplieron con los objetivos pactados con el cliente**. 90%**
20. Cuando surgen dificultades, y no se logran los objetivos de la iteración pactados, **SE TRABAJAN HORAS EXTRAS** –
21. Especifique el nivel de satisfacción del cliente frente al producto desarrollado. **CONFORME**,
22. Identifique las situaciones de riesgo más comunes detectadas en los proyectos. Numerarlas según su importancia, siendo 1 el riesgo más importante:
    1. **Variaciones de los requerimientos del cliente**
    2. **Poca participación del cliente**
    3. **Imprecisiones del cliente**
    4. **Equipo de desarrollo desmotivado**
    5. **Equipo de desarrollo involucrado en varios proyectos**
23. ¿Por qué trabaja con Genexus? **Buenos resultados rápidos y generación de código. Además es bueno para trabajar en proyectos grandes**
24. Indique si Utiliza o piensa utilizar a futuro

- GXServer: **No Utilizo**
- GXFlow: **No Utilizo**
- GXQuery: **No Utilizo**

25. Otros comentarios: ----------

7.2.6 Encuesta 6

Fecha de realización de la encuesta: Agosto 2014

DATOS GENERALES

1. Tipo de empresa: **Pública**
2. Tiempo en el mercado: **64 años cumple el octubre**
3. Tiempo de uso de Genexus: **13 años**
4. Cantidad de proyectos desarrollados con Genexus**: 5 cinco**
5. Proporción de Proyectos para terceros: (100 – 0): **ninguno**
6. Cantidad de personas involucradas: **6 seis**
7. La empresa cuenta con más de un equipo de desarrollo: **SI**

Cantidad de personas de cada equipo **SI**

8. La empresa trabaja en más de un proyecto a la vez: **SI**
9. Los equipos trabajan en más de un proyecto a la vez: **SI**
10. Las personas involucradas comparten el horario de trabajo: **SIEMPRE**
11. Las personas involucradas comparten el lugar de trabajo: **si**
12. Existe un líder de proyecto: **SI**
13. El líder de proyecto es responsable de más de un proyecto a la vez: **SI**. En caso afirmativo, indique cantidad de proyectos que suele manejar al mismo tiempo: -----

FORMA DE TRABAJO CON GENEXUS

1. Tiene un procedimiento fijo a seguir para cada desarrollo **SI**. Si es afirmativo, puede describirlo. **Entrevista, recopilación de fuentes de información, prototipacion evolutiva**
2. Cada equipo puede adaptar el procedimiento según el proyecto **SI**
3. El procedimiento a seguir en cada proyecto de desarrollo es definido**: por el líder de proyecto + por el equipo**
4. El cliente está involucrado**: durante todo el proceso**
5. Se realizan entregas frecuentes (a lo sumo cada 2 meses) de producto desarrollado al cliente. **SI.** Especifique el tiempo: **a solicitud del cliente, no se puede especificar un tiempo cronológico exacto**
6. Se tiene colaboración permanente con el cliente. **SI**. Explique como: **Si, solicitando su participación durante el desarrollo y pruebas**
7. Los requerimientos se definen al inicio del proyecto: **SI**. Especifique COMO define los requerimientos: **Si, a través de entrevistas que se van refinando con el avance del proyecto**
8. Los requerimientos cambian a lo largo del proyecto: **SI SIEMPRE**
9. Se realiza un desarrollo iterativo: SI. Indique duración de la iteración. **Semanalmente**
10. La planificación del desarrollo se realiza **AL INICIO PERO SE VA MODIFICANDO ENCADA ITERACION –** Especifique Como se planifica: **Se planifica teniendo en cuenta la complejidad del proyecto, el tiempo de entrega solicitado y qué recursos humanos y tecnológicos se cuentan para el equipo de trabajo**
11. Los alcances y costos se pactan con el cliente: **Al inicio se pactan los alcances y se van actualizando según sea requerido. Al ser una empresa del estado que cuenta con desarrolladores propios no se analizan los costos.** Especifique COMO se definen los alcances: **Se elabora a partir de los entregables principales, supuestos y posibles restricciones al Proyecto que se han documentado en la fase de Iniciación, siendo en la fase de Planificación donde el Alcance del Proyecto se describe y se define de manera más específica, según se va obteniendo más información sobre del Proyecto. Durante este proceso, se analizan los riesgos, los supuestos y las restricciones existentes, actualizando esta información según sea necesario**
12. Se realiza un diseño utilizando diagramas UML: **NO**. Especifique cuales utiliza

No, se utiliza técnicas de las metodologías ágiles

13. Se sigue una metodología ágil: **SI**. En caso afirmativo, especifique cuál: Si, scrum
14. Especifique cuáles de las siguientes prácticas ágiles se siguen en sus proyectos
    - **Programación de a pares**
    - **Refactorización**
    - **Reuniones diarias**
    - **Entregas frecuentes**
15. Especifique herramientas utilizadas para realizar algunas de las prácticas ágiles seleccionadas: **Se esta evaluando Kanban Board**
16. Especifique herramientas utilizadas para el seguimiento del proyecto: **Se esta evaluando Kanban Board**
17. Utiliza un diagrama Burn Down para seguimiento del proyecto. **SI**. **para proyectos grandes**

18. Se desarrolla con reutilización. **SI**. Especifique el porcentaje de reutilización de proyectos anteriores. ¿Cómo se seleccionan los artefactos a reutilizar? **30% y también depende de la similitud de los proyectos**
19. Especifique el porcentaje de proyectos donde se cumplieron con los objetivos pactados con el cliente. **90%**
20. Cuando surgen dificultades, y no se logran los objetivos de la iteración pactados, **SE TRABAJAN HORAS EXTRAS –**
21. Especifique el nivel de satisfacción del cliente frente al producto desarrollado **MUY SATISFECHO**
22. Identifique las situaciones de riesgo más comunes detectadas en los proyectos. Numerarlas según su importancia, siendo 1 el riesgo más importante:
    - **Imprecisiones del cliente 1**
    - **Variaciones de los requerimientos del cliente 3**
    - **Poca participación del cliente 2**
    - **Equipo de desarrollo desmotivado 5**
    - **Equipo de desarrollo involucrado en varios proyectos 4**
23. ¿Por qué trabaja con Genexus? **Metodología de desarrollo productiva y ágil**
24. Indique si Utiliza o piensa utilizar a futuro

- GXServer: **Pienso utilizar**
- GXFlow: **Pienso utilizar**
- GXQuery: **Pienso utilizar**

25. Otros comentarios: ----------

7.2.7 Encuesta 7

Fecha de realización de la encuesta: Marzo 2013

DATOS GENERALES

1. Tipo de empresa: **Pública**
2. Tiempo en el mercado: -
3. Tiempo de uso de Genexus: **10 años**
4. Cantidad de proyectos desarrollados con Genexus: **>5**
5. Proporción de Proyectos para terceros: (100 – 0) **100%**
6. Cantidad de personas involucradas: **1. A veces se incorpora un programador extra (o asesor de diseño gráfico)**
7. La empresa cuenta con más de un equipo de desarrollo: **NO**
8. La empresa trabaja en más de un proyecto a la vez: **SI**
9. Los equipos trabajan en más de un proyecto a la vez: **SI**
10. Las personas involucradas comparten el horario de trabajo: **SIEMPRE** Explique: **es una sola persona**
11. Las personas involucradas comparten el lugar de trabajo: **SI**
12. Existe un líder de proyecto: **SI**
13. El líder de proyecto es responsable de más de un proyecto a la vez: **SI**

En caso afirmativo, indique cantidad de proyectos que suele manejar al mismo tiempo **3 o 4**

FORMA DE TRABAJO CON GENEXUS

1. Tiene un procedimiento fijo a seguir para cada desarrollo **SI**

Si es afirmativo, puede describirlo.

Se realizan entrevistas iniciales y se realiza un mapa conceptual (una persona con cierta capacidad).
Se realiza un análisis de negocio (discusión en conjunto).
Si es desde cero
Discusión de diseño
Se cuantifica el tiempo según cantidad de objetos y acciones (basado en cosas hechas)
Se modulariza y se implementa
No se personaliza el 99%
Si hay reuso
Se retoca lo utilizado
Nota: tiene una tabla ordenada de objetos

2. Cada equipo puede adaptar el procedimiento según el proyecto **SI**

3. El procedimiento a seguir en cada proyecto de desarrollo es definido **por el líder de proyecto**

4. El cliente está involucrado: **durante todo el proceso**. Especifique**. Diálogo desde la primera entrega**

5. Se realizan entregas frecuentes (a lo sumo cada 2 meses) de producto desarrollado al cliente. **SI**. Especifique el tiempo **Se realiza un cierto porcentaje del desarrollo y recién se entrega la primera versión, luego es más fluido**

6. Se tiene colaboración permanente con el cliente. **SI**. Explique como: ----

7. Los requerimientos se definen al inicio del proyecto: **SI**. Especifique COMO define los requerimientos **Los requerimientos generales se definen al principio a través de un documento de requerimiento representado con mapas conceptuales Se comparte con el cliente y el cliente modifica**

8. Los requerimientos cambian a lo largo del proyecto: **SI SIEMPRE** –

9. Se realiza un desarrollo iterativo: **SI**. Indique duración de la iteración. **No son fijos, primero se construye un prototipo del 40% y después un mes se entrega mas frecuentemente**

10. La planificación del desarrollo se realiza. **AL INICIO PERO SE VA MODIFICANDO ENCADA ITERACION** – Especifique Como se planifica **Hay una cota no rígida de plazos (para buscar más ayuda)**

11. Los alcances y costos se pactan con el cliente: **AL INICIO - PARA CADA ITERACION.** Especifique COMO se definen los alcances. **Quedan abiertos los alcances. Costo no se maneja (es personal del organismo).**

12. Se realiza un diseño utilizando diagramas UML: **NO**. Especifique cuales utiliza **Solo mapas conceptuales que tienen diseño de tablas OO**

13. Se sigue una metodología ágil: **NO.** En caso negativo, tiene pensado incorporar alguna: **NO**. Especifique cuál incorporaría o la razón por la que no lo considera necesario o apropiado: **por ser una sola persona**

14. Especifique cuáles de las siguientes prácticas ágiles se siguen en sus proyectos

- **Programación de a pares (para probar cosas nuevas)**
- **Refactorización**

- **Reuniones diarias (reuniones semanales)**
- **Entregas frecuentes**

15. Especifique herramientas utilizadas para realizar algunas de las prácticas ágiles seleccionadas. Free mind, Ticket
16. Especifique herramientas utilizadas para el seguimiento del proyecto: tickets
17. Utiliza un diagrama Burn Down para seguimiento del proyecto. **NO**.
18. Se desarrolla con reutilización. **SI**. Especifique el porcentaje de reutilización de proyectos anteriores. ¿Cómo se seleccionan los artefactos a reutilizar? **50% (a futuro de mod reut). Lo decide el líder. Se trata de serializar y optimizar y minimizar los sistemas a medida. No se pierde el control y diseño en Genexus**
19. Especifique el porcentaje de proyectos donde se cumplieron con los objetivos pactados con el cliente**. 80%**
20. Cuando surgen dificultades, y no se logran los objetivos de la iteración pactados, **SE TRABAJAN HORAS EXTRAS** –
21. Especifique el nivel de satisfacción del cliente frente al producto desarrollado. **MUY SATISFECHO,**
22. Identifique las situaciones de riesgo más comunes detectadas en los proyectos. Numerarlas según su importancia, siendo 1 el riesgo más importante:

- **Variaciones de los requerimientos del cliente (2)**
- **Equipo de desarrollo involucrado en varios proyectos (1)**

23. ¿Por qué trabaja con Genexus?

- ***Herramienta potente que sigue al desarrollador (usuario), es intuitivo. La sinergia (retroalimentación)***
- ***En castellano***
- ***Permite responder a las necesidades del Estado rápidamente y obtener tal vez en tres días una funcionalidad que programando PHP demore varios meses. En Estato, es urgente todo. Por ejemplo la copa de leche***

24. Indique si Utiliza o piensa utilizar a futuro

- GXServer: **Pienso utilizar**
- GXFlow: **Pienso utilizar tal vez**
- GXQuery: **No Utilizo**

25. Otros comentarios:

En gobierno generalmente desarrollan pasantes (temporales). Si se desarrollara con php se dedica mucho tiempo. Hay poca documentación y se termina contratando a alguien por más tiempo. El trabajar con Genxus es más rápido y estándar

7.2.8 Encuesta 8

Fecha de realización de la encuesta: Agosto 2014

DATOS GENERALES

1. Tipo de empresa: **Privada**
2. Tiempo en el mercado: **Más de 30 años**
3. Tiempo de uso de Genexus: **Aprox. 15 años**
4. Cantidad de proyectos desarrollados con Genexus: **Más de 30 proyectos**
5. Proporción de Proyectos para terceros: **0**

6. Cantidad de personas involucradas: **Actualmente 4 personas, más 2 externos**
7. La empresa cuenta con más de un equipo de desarrollo: **SI, para proyectos SAP**

Cantidad de personas de cada equipo **2 personas**

8. La empresa trabaja en más de un proyecto a la vez: **SI**
9. Los equipos trabajan en más de un proyecto a la vez: **SI**
10. Las personas involucradas comparten el horario de trabajo: **SIEMPRE. Se cumple un horario establecido, los externos se acomodan a los horarios actuales.**
11. Las personas involucradas comparten el lugar de trabajo: **SI**
12. Existe un líder de proyecto: **SI**
13. El líder de proyecto es responsable de más de un proyecto a la vez: **SI. Puede manejar hasta 2 proyectos a la vez**

FORMA DE TRABAJO CON GENEXUS

1. Tiene un procedimiento fijo a seguir para cada desarrollo **SI**. Si es afirmativo, puede describirlo.

- ***Relevamiento***
- ***Especificación Funcional, confirmación usuario***
- ***Desarrollo y/o configuración***
- ***Pruebas Unitarias***
- ***Pruebas Integrales con funcionales***
- ***Capacitación, si requiere***
- ***Puesta a Punto***
- ***Salida en Vivo***
- ***Soporte***

2. Cada equipo puede adaptar el procedimiento según el proyecto **Si, DEPENDE del alcance del proyecto, agregando o quitando más pasos de acuerdo al procedimiento anterior, se puede ciclar entre esos pasos, generando mini-proyectos hasta conseguir la solución final**
3. El procedimiento a seguir en cada proyecto de desarrollo es definido **por el líder de proyecto**
4. El cliente está involucrado**: durante todo el proceso**. Especifique. **Como es interno y los proyectos siempre se tiene un líder funcional, además del referente de TI. Él es el encargado de comunicar y revisar las especificaciones y validar la solucionar. Se trabaja en conjunto con el área funcional**
5. Se realizan entregas frecuentes (a lo sumo cada 2 meses) de producto desarrollado al cliente. **SI**. **Son periódicos depende la magnitud de la solución, se presentan avances a medida que se avanza y son más periódicos al llegar a la salida en productivo**
6. Se tiene colaboración permanente con el cliente. **SI**. Explique como: **Durante la definición del alcance, relevamiento del proceso a automatizar, validación de la solución y salida a productivo. Es importante contar con su apoyo para la implementación de proyectos.**
7. Los requerimientos se definen al inicio del proyecto: **SI**. Especifique COMO define los requerimientos: **Mediante relevamiento a través de entrevistas y/o encuestas a usuarios claves, se deja especificado en un documento BluePrint BBP**

8. Los requerimientos cambian a lo largo del proyecto: **ALGUNAS VECES. Generalmente se debe a medida que se avanza cuando se especifique en detalle. Además puede modificar por propuestas de mejoras al proceso de parte de TI**
9. Se realiza un desarrollo iterativo: **SI**. Indique duración de la iteración **Las iteraciones por cada etapa del proyecto son de acuerdo al tamaño del mismo. Es de semanas cuando se acerca a la salida en productivo**
10. La planificación del desarrollo se realiza **AL INICIO. Se puede planificar en algunas iteraciones cuando se encuentre desviaciones del proyecto**. Especifique Como se planifica: **Se genera un diagrama de Gantt con los recursos necesarios y actividades que se deben llevar a cabo, indicando cronograma de fechas de cada etapa.**
11. Los alcances y costos se pactan con el cliente: **AL INICIO. Se evalúan al inicio los recursos necesarios.** Especifique COMO se definen los alcances: **Se definen de acuerdo a un análisis costo-beneficio, indicando el tiempo de recupero de la inversión. Así también teniendo en cuenta la innovación y cambio continuo del mercado.**
12. Se realiza un diseño utilizando diagramas UML: **SI**. Especifique cuales utiliza: **Se utilizan diagrama de clases para validar con el diagrama de entidad relación que se genera con Genexus. Diagrama de Casos de Usos para ver el Flujo de los Procesos**
13. Se sigue una metodología ágil: **NO.** En caso negativo, tiene pensado incorporar alguna**: SI.** Especifique cuál incorporaría o la razón por la que no lo considera necesario o apropiado: **Utilizaría la metodología Scrum para los proyectos que se deben resolver en forma rápida, igualmente realizamos proyectos basadas en listado de tareas idéntica a dicha metodología.**
14. Especifique cuáles de las siguientes prácticas ágiles se siguen en sus proyectos

- **Refactorización**
- **Diseño simple**
- **Iteraciones cortas**
- **Entregas frecuentes**

15. Especifique herramientas utilizadas para realizar algunas de las prácticas ágiles seleccionadas: --------
16. Especifique herramientas utilizadas para el seguimiento del proyecto: Microsoft **Project, Aplicativo de Gestión de Tareas, Correo, Excel, PowerPoint.**
17. Utiliza un diagrama Burn Down para seguimiento del proyecto**. NO.**
18. Se desarrolla con reutilización**. SI.** Especifique el porcentaje de reutilización de proyectos anteriores. **30-50%.** ¿Cómo se seleccionan los artefactos a reutilizar? **De acuerdo a los objetos/funciones similares que se pueden aplicar nuevamente.**
19. Especifique el porcentaje de proyectos donde se cumplieron con los objetivos pactados con el cliente**. La mayoría se cumplen en su totalidad, en algunos nos quedaron cosas por mejorar pero el porcentaje siempre de mayor al 90%**
20. Cuando surgen dificultades, y no se logran los objetivos de la iteración pactados: **SE TRABAJAN HORAS EXTRAS, SE EXTIENDEN LA ITERACION y OTRAS ACCIONES: Si es factible se recurre a más recursos.**
21. Especifique el nivel de satisfacción del cliente frente al producto desarrollado: **MUY SATISFECHO, CONFORME.**
22. Identifique las situaciones de riesgo más comunes detectadas en los proyectos. Numerarlas según su importancia, siendo 1 el riesgo más importante:

1. **Poca participación del cliente**

2. **Imprecisiones del cliente**
3. **Planificación imprecisa**
4. **Equipo de desarrollo involucrado en varios proyectos**
5. **Variaciones de los requerimientos del cliente**
6. **Equipo de desarrollo desmotivado**

23. ¿Por qué trabaja con Genexus? **Porque es una herramienta nos permite realizar desarrollos de aplicaciones en forma rápida y dinámica. Además permite trabajar en diferentes plataformas y base de datos lo cual permite elegir la mejor solución tecnológica para el proyecto a encarar**.

24. Indique si Utiliza o piensa utilizar a futuro

- GXServer: **Pienso utilizar**
- GXFlow: **No Utilizo**
- GXQuery: **Pienso utilizar**

25. Otros comentarios: -----

7.2.9 Encuesta 9

Fecha de realización de la encuesta: agosto 2014

DATOS GENERALES

1. Tipo de empresa: **Privada**
2. Tiempo en el mercado: **10 años**
3. Tiempo de uso de Genexus: **10 años**
4. Cantidad de proyectos desarrollados con Genexus: **más de 30**
5. Proporción de Proyectos para terceros: (100 – 0) **99%**
6. Cantidad de personas involucradas**: 11 personas**
7. La empresa cuenta con más de un equipo de desarrollo: **SI.** Cantidad de personas de cada equipo **2 -3 personas**
8. La empresa trabaja en más de un proyecto a la vez: **SI**
9. Los equipos trabajan en más de un proyecto a la vez: **SI**
10. Las personas involucradas comparten el horario de trabajo: **SE SOLAPAN.** Explique: Algunos comparten entre sí. Otros se solapan.
11. Las personas involucradas comparten el lugar de trabajo: **SI**. Especifique cómo se suple la falta de lugar común: **TELEFONO – CORREO - REUNIONES PACTADAS**
12. Existe un líder de proyecto: **SI**
13. El líder de proyecto es responsable de más de un proyecto a la vez: **SI, cada líder puede manejar de 2 a 3 proyectos**

FORMA DE TRABAJO CON GENEXUS

1. Tiene un procedimiento fijo a seguir para cada desarrollo **SI**. Si es afirmativo, puede describirlo.
    - ***Se realiza un relevamiento inicial, generalmente el líder de proyecto. Realiza el modelado del negocio y elabora las minutas de reunión***

- ***Se conforma el equipo para el proyecto. El líder define los módulos necesarios. Define los recursos necesarios y que se reutilizará***
- ***El líder introduce en el proyecto al equipo y distribuye las tareas (registro de distribución de tareas en Mail o minutas). Se charla con el equipo sobre las necesidades***
- ***Los miembros del equipo pueden interactuar con el cliente, ante dudas o necesidad de información pueden tener acceso directo al cliente de manera presencial, por teléfono o correo.***
- ***Equipo implementa funcionalidades y realiza sus propias pruebas e integra***
- ***Las pruebas de aceptación las realiza el cliente, generalmente en su lugar de trabajo, con presencia de alguien del equipo.***
- ***Existen especialistas, generalmente externos en comunicaciones, redes, bases de datos, contabilidad. Que se incorporan en los equipos según sea requerido***
- ***A medida que se completa un módulo se va entregándolo al cliente***

2. Cada equipo puede adaptar el procedimiento según el proyecto **SI** (especificar) **sobre todo según el tamaño del mismo.**
3. El procedimiento a seguir en cada proyecto de desarrollo es definido: **por el líder de proyecto**
4. El cliente está involucrado: **durante todo el proceso** Especifique. **Mails, visitas, llamadas**
5. Se realizan entregas frecuentes (a lo sumo cada 2 meses) de producto desarrollado al cliente. **SI.** Especifique el tiempo. **No hay tiempos definidos, depende de si es un producto nuevo o no. Si son solo cambios puntuales a sistemas existentes**
6. Se tiene colaboración permanente con el cliente. **SI**. Explique como**: pruebas de aceptación y comunicación (mails, visitas, llamadas)**
7. Los requerimientos se definen al inicio del proyecto: **SI.** Especifique COMO define los requerimientos: **Minutas donde se explia cada tema y las necesidades**
8. Los requerimientos cambian a lo largo del proyecto: **ALGUNAS VECES**
9. Se realiza un desarrollo iterativo: **NO**.
10. La planificación del desarrollo se realiza: **OTRA OPCION (especificar) AL INICIO PERO SE VA MODIFICANDO A LARGO DE LA IMPLEMENTACION DE LOS MODULOS.** Especifique Como se planifica: **OJO DE BUEN CUBERO, por similitud de complejidad de proyectos anteriores, analizando módulo por módulo**
11. Los alcances y costos se pactan con el cliente: **AL INICIO.** pecifique COMO se definen los alcances **Minutas y contrato**
12. Se realiza un diseño utilizando diagramas UML: **NO**.
13. Se sigue una metodología ágil: **NO.** En caso negativo, tiene pensado incorporar alguna: **SI**. Especifique cuál incorporaría o la razón por la que no lo considera necesario o **apropiado SCRUM. Se realizó la capacitación y uno de los equipos está comenzando a incorporarlo**
14. Especifique cuáles de las siguientes prácticas ágiles se siguen en sus proyectos: ----
15. Especifique herramientas utilizadas para realizar algunas de las prácticas ágiles seleccionadas.-----
16. Especifique herramientas utilizadas para el seguimiento del proyecto: **NO**
17. Utiliza un diagrama Burn Down para seguimiento del proyecto. **NO**.

18. Se desarrolla con reutilización. **SI**. Especifique el porcentaje de reutilización de proyectos anteriores. ¿Cómo se seleccionan los artefactos a reutilizar? **Más del 75%, se eligen por similitud de módulos anteriormente desarrollados. Los define el líder del proyecto**

19. Especifique el porcentaje de proyectos donde se cumplieron con los objetivos pactados con el cliente. **100%**

20. Cuando surgen dificultades, y no se logran los objetivos de la iteración pactados, **SE TRABAJAN HORAS EXTRAS – SE EXTIENDEN LA ITERACION**

21. Especifique el nivel de satisfacción del cliente frente al producto desarrollado. **MUY SATISFECHO, muchos y CONFORME, algunos**

22. Identifique las situaciones de riesgo más comunes detectadas en los proyectos. Numerarlas según su importancia, siendo 1 el riesgo más importante:

- **Otras (especificar) el cliente no es organizado y piensa que el sistema lo va a organizar**
- **Imprecisiones del cliente**
- **Variaciones de los requerimientos del cliente**
- **Poca participación del cliente**
- **Equipo de desarrollo involucrado en varios proyectos**
- **Equipo de desarrollo desmotivado**

23. ¿Por qué trabaja con Genexus?

- **Buena herramienta que permite desarrollo más rápido**
- **Muy difundido**
- **Se cuenta con todos los ambientes y dispositivos contemplados en una sola herramienta (al igual que progress) por lo que no es necesario contar con especialistas**
- **Usaban progress que es similar pero tiene un costo mayor**

24. Indique si Utiliza o piensa utilizar a futuro

- GXServer: **No Utilizo**, Por la forma de trabajo no lo consideran apropiado ya que trabajan mucho en el cliente además de ser muy costoso 10.000dls.
- GXFlow: **No Utilizo,** Usaron para algunos proyectos
- GXQuery: Pienso utilizar. Lo usaron para un cliente específico. Ahora para otro están por considerar GXExplorer (Query mejorado-→cubo)

25. Otros comentarios: -----

8 ANEXO 2: ENCUESTA SOBRE DESARROLLO CON METODOLOGÍAS ÁGILES EN SALTA

8.1 ANALISIS DE DATOS RELEVADOS

8.1.1 Contexto

En Salta hace unos pocos años se ha comenzado a hablar sobre metodologías ágiles y comienzan a realizarse actividades al respecto. Recientemente, en el mes de junio de 2014 se realizó el primer Salta Agile Open Space. Fue coordinado y difundido por Juan Gabardini, Juan Ladetto y la autor a de este trabajo. Actualmente la Escuela de Administración de la Provincia (EAP) está incluyendo, dentro de los cursos que conforman la capacitación para obtener el título de Especialista en Software, tres cursos relacionados con metodologías ágiles, de los cuales solo dos fueron dictados hasta el momento. Ciertos organismos provinciales están capacitando a su personal y buscando certificarlos como Certified Scrum Developer y Certified Scrum Product Owner. En la Universidad Nacional de Salta, se logró realizar un primer taller con Juan Gabardini para poder certificar a los docentes interesados y alumnos y graduados avanzados como Certified Scrum Developer. Existe en el ambiente universitario, una necesidad de talleres prácticos para poder tener una experiencia más concreta y poder asimilar los conceptos teóricos que se brindan desde algunas cátedras.

La primera actividad de la comunidad ágil de Salta puede decirse que fue el evento antes mencionado del Salta Agile Open Space. Aprovechando la posibilidad de tener reunidas a personas interesadas y con cierto conocimiento de Metodologías ágiles de la zona, se realizó una encuesta a quienes actualmente están trabajando con estas metodologías. De los participantes solo un 15% trabajaba efectivamente con metodologías ágiles. Fueron éstas personas las que completaron la encuesta ya que podrían brindar información de la forma de trabajo real utilizando las metodologías ágiles.

8.1.2 Personas objeto de relevamiento

En esta sección se muestra la información relevada sobre el uso de las Metodologías Ágiles en algunos sectores de desarrollo de software de organizaciones de Salta. Los encuestados fueron, como se mencionó anteriormente, aquellos asistentes del primer Salta Agile Open Space, que trabajan con desarrollo ágil. Entendiendo que estos datos no son completamente representativos, se considera que pueden revelar algunos aspectos del uso de las metodologías ágiles en la zona.

Se pretendió conocer principalmente cuáles son las metodologías y las técnicas ágiles más utilizadas, como así también la conformación de los grupos de trabajo, y forma de trabajo en los proyectos a los que se aplican. La sección finaliza con las conclusiones a las que se pudo arribar sobre cada uno de los temas mencionados, luego de analizadas las respuestas obtenidas. La encuesta puede consultarse dentro de este anexo en la siguiente sección.

8.1.3 Resultados Obtenidos

8.1.3.1 Datos Generales

Los encuestados fueron elegidos de un grupo de 54 personas que asistieron a un evento abierto y gratuito de la comunidad ágil en Salta. El propósito de dicho evento era poder intercambiar experiencias. Sólo completaron la encuesta aquellos que efectivamente estaban trabajando con metodologías ágiles o prácticas ágiles en su ámbito profesional.

Los encuestados residen todos en Salta Capital. El 50% tiene formación universitaria completa y el otro 50% formación universitaria incompleta. Ninguno ha cursado sus estudios superiores en institutos terciarios. Solo dos personas tienen certificaciones otorgados por la Scrum Alliance, de Scrum Developer y Scrum Product Owner cada uno. Esto es un 25% del total de encuestados.

Respecto al tipo de organización donde tiene dependencia laboral, el 62.50% trabaja en la repartición pública y el 37.50% en el ámbito privado.

Figura 8.1.1: Tipo de organismo donde trabaja

8.1.3.2 Datos específicos de trabajo con metodologías ágiles

- ***Metodologías utilizadas***

Se consultó cuáles eran las metodologías utilizadas. La mayoría de los encuestados mencionó Scrum, con un 87,50% como la metodología utilizada, siendo Kanban la siguiente en ranking de uso. Luego, le sigue Scrumban, que es combinación de las dos metodologías anteriores. El 50% trabaja exclusivamente con Scrum sin combinarla, es decir aplicándola de manera pura. Pero, otros encuestados combinan las metodologías.

Un gráfico interesante es analizar cada una de las metodologías que surgieron en la encuesta y su porcentaje de aplicación. Con esto, se puede observar que por lejos Scrum es la más utilizada, luego muy por debajo con un 25% sigue Kanban. También, uno de los encuestados manifestó estar analizando la posibilidad de incorporar a futuro TDD y BDD.

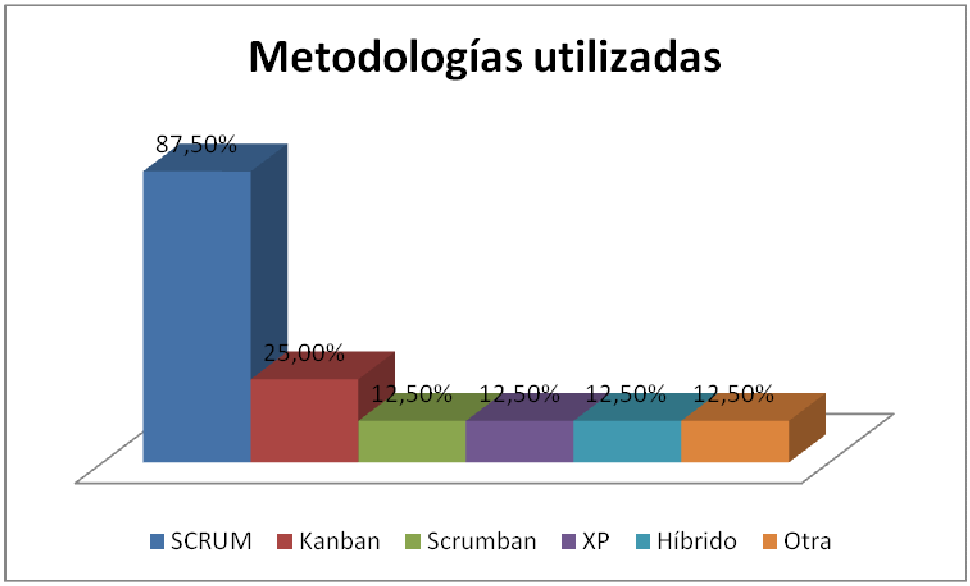


Figura 8.1.2: Metodologías utilizadas

- ***Duración de la Iteración***

En toda metodología ágil es muy importante la duración de la iteración. En la encuesta se permitió que cada uno exprese en la unidad que considere adecuada la duración de la misma. Todos respondieron en semanas, salvo uno que fijó la duración en tres días. Puede observarse claramente que existe mayor cantidad de personas que trabajan con iteraciones de 2 o 4 semanas. La amplia mayoría respeta tener iteraciones de 1 a 4 semanas de duración, siendo atípica la duración mencionada de 3 días. Ninguno mencionó tener duraciones variables en las iteraciones.

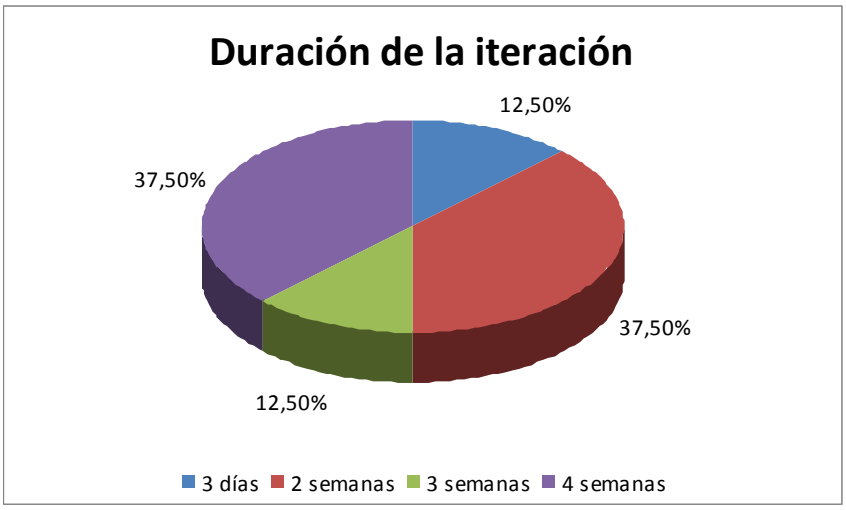


Figura 8.1.3: Duración de la iteración

- ***Planificación de la iteración***

En toda metodología ágil, es importante poder planificar el proyecto y poder modificarlo a medida que se va avanzando. Para corroborar esto, se brindaron tres opciones: al inicio, al inicio modificando en cada iteración, otra opción (especificar). Tal como se esperaba, el 100% contestó que se planifica al inicio, modificando en cada iteración.

- ***Cliente involucrado en el proyecto***

En toda metodología ágil, es importante poder tener al cliente involucrado a lo largo del proyecto. Para corroborar esto, se brindaron tres opciones: al inicio, durante todo el proyecto, al final. Se esperaba una respuesta similar a la anterior, pero se obtuvo una diferente. Si bien la amplia mayoría (87.5%) contestó que el cliente está involucrado a lo largo del proyecto, uno (12.50%) contestó que sólo al inicio y no estaría cumpliendo con el manifiesto ágil.

Figura 8.1.4: Momento en que participa el cliente en el proceso de desarrollo

- ***Entregas Frecuentes***

El primer principio del Manifiesto ágil define que la prioridad es satisfacer al cliente mediante entregas de software tempranas y continuas; solo el 75% respondió afirmativamente respecto a las entregas frecuentes. El resto no realiza estas entregas frecuentes.

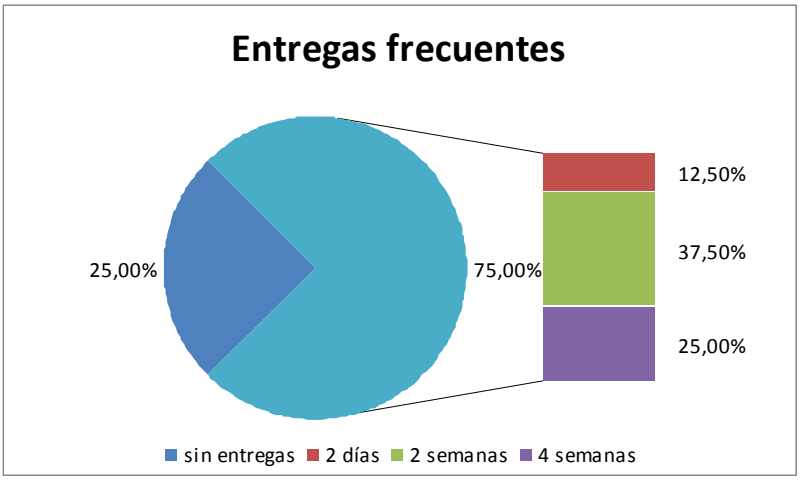


Figura 8.1.5: Realización de Entregas frecuentes

Entre aquellos que sí realizan entregas frecuentes, las entregas cada 2 semanas fueron los períodos más utilizados, seguidos por aquellos que lo realizan cada 4 semanas y por último un pequeño porcentaje que realiza entregas cada 2 días.

- ***Colaboración con el cliente***

El manifiesto ágil menciona que la colaboración del cliente es muy importante y se valora más que a una documentación exhaustiva. La intención con esta pregunta era sondear los diferentes recursos para lograr esta comunicación. Uno de los encuestados no contestó este punto. Los correos electrónicos, las reuniones semanales y las reuniones al final del sprint, fueron las más mencionadas, con un porcentaje de 37.50% cada una. Las videoconferencias fueron una alternativa con menor porcentaje, siendo las charlas y las especificaciones de requerimientos las de menor porcentaje con un 12.50% cada una.

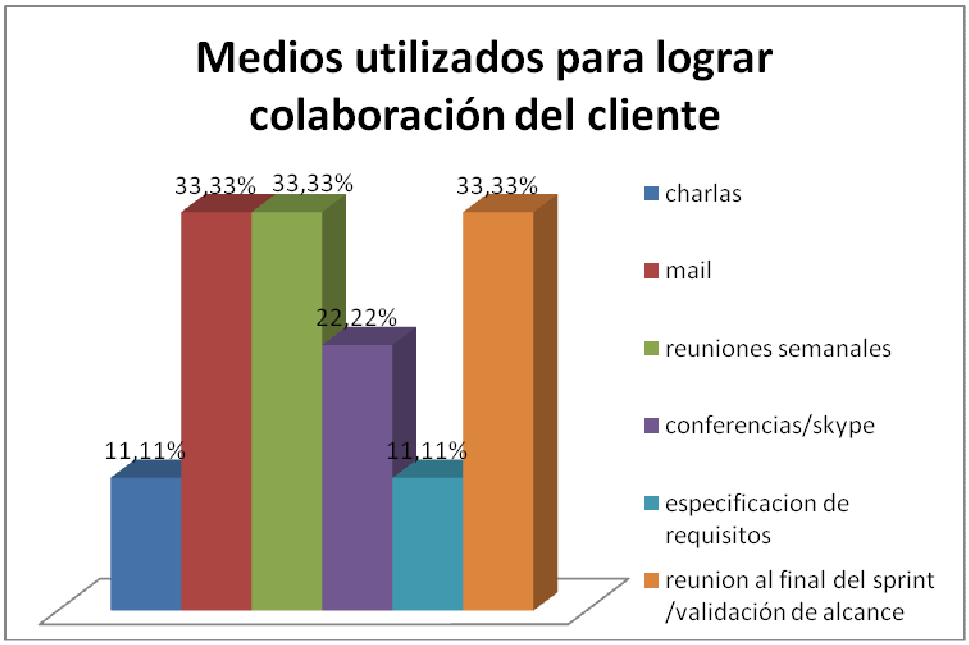


Figura 8.1.6: Medios utilizados para lograr la colaboración del cliente

El diálogo directo cara a cara, sin intermediarios, se puede dar en una charla o a través de reuniones pactadas. Si se analiza la proporción de encuestados que plantean alguna de estas comunicaciones directas, solo se obtiene un 55.56%.

- ***Cambio de requerimientos***

Considerar que existen cambios en los requerimientos es algo inherente a las metodologías ágiles. En las encuestas se puede observar que todos consideran que existen cambios en los requerimientos de los proyectos, solo que se encuentra dividido en dos grupos iguales, entre quienes consideran que los requerimientos siempre cambian y quienes consideran que algunas veces cambian.

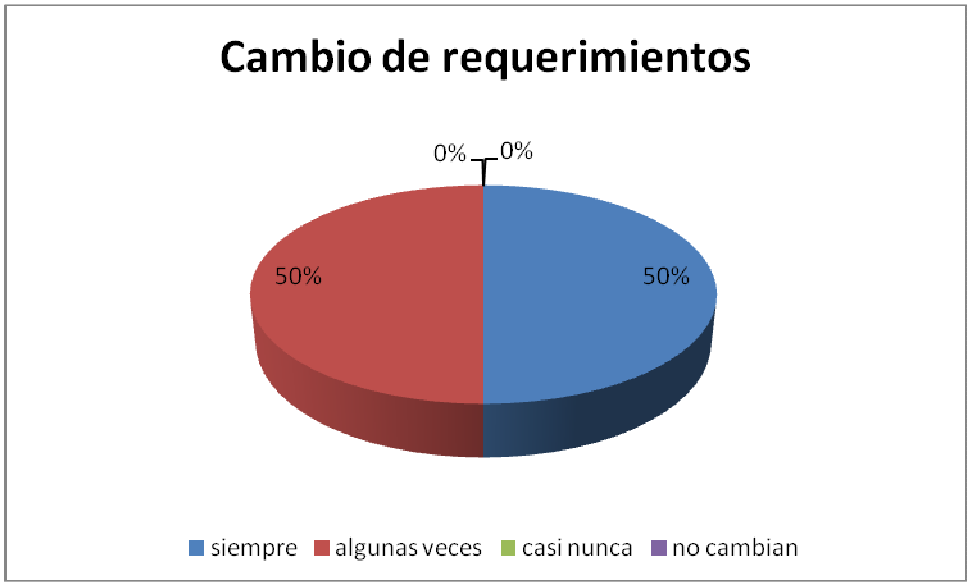


Figura 8.1.7: Cambio de requerimientos

- ***Alcances y costos***

Cuándo se pactan los alcances del proyecto y los costos asociados es otro punto importante en un desarrollo de software. Uno de los encuestados no completó este punto de la encuesta. El 50.00% define alcances por iteración pero existe un 25.00% que lo realiza al inicio y un 12.50% que lo realiza por función (Opción detallada por el encuestado). Aquellos que trabajan en organismos públicos además aclararon que no se manejan costos, debido a que el personal cobra un sueldo.

La proporción de encuestados que definen los alcances y costos al inicio y no por iteración son un poco más del doble de quienes solo tienen colaboración del cliente al inicio. Pero, la planificación en las metodologías ágiles no se define al inicio del proyecto y permanece inalterable, existe en varias metodologías la práctica del juego de la planificación para poder realizar lo que es prioritario para el cliente y le aporta mayor valor a su negocio. El manifiesto ágil habla de valorar la respuesta al cambio sobre un plan contractual.

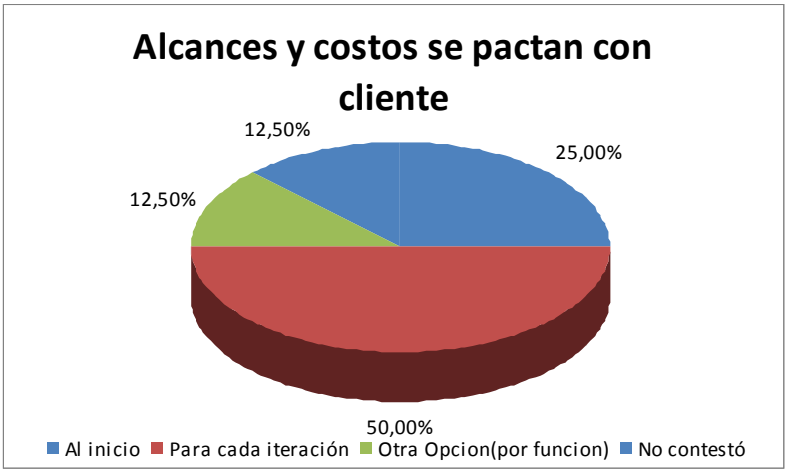


Figura 8.1.8: Definición de alcances

- ***Desarrollo con reutilización***

En este punto se solicitó que indiquen si trabajaban con reutilización y en caso de contestar afirmativamente, el porcentaje de reutilización respecto a proyectos anteriores. Además se solicitaba que especifique la forma de lograrlo.

Figura 8.1.9: Desarrollo con reutilización

Uno de los encuestados, respondió que realiza reutilización, pero no especificó el porcentaje, esto se refleja con un valor NN en el siguiente gráfico. A continuación se muestra en la gráfica los porcentajes de los valores obtenidos.

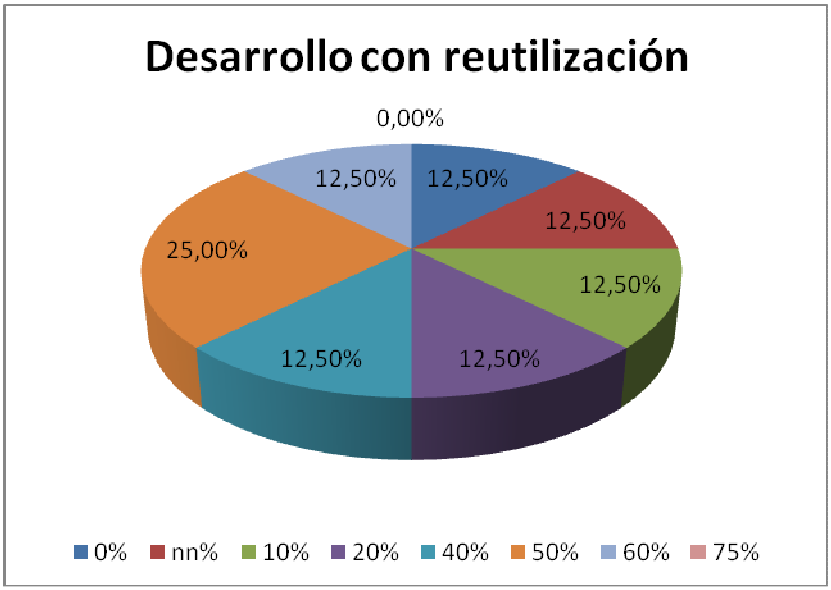


Figura 8.1.10: Porcentaje de reutilización en el desarrollo de software

Si bien ninguno de los encuestados realiza una reutilización de más del 75%, en una de las encuestas se detalló que se podría considerar una reutilización del 75% dentro de sistemas que no son nuevos, es decir reutilización dentro del mismo.

Uno de los encuestados mencionó que cuentan con una biblioteca para facilitar la reutilización, el resto solo se limitó a aclarar que la reutilización se realiza por similitud de funcionalidades. Y uno agregó que también se reutiliza la interfaz o algunos diseños.

- ***Prácticas y herramientas ágiles utilizadas***

Tomando como base la encuesta de Version One sobre la aplicación de prácticas ágiles o metodologías ágiles en empresas en el mundo, se elaboró un listado con 24 prácticas ágiles que pueden utilizarse trabajando con metodologías ágiles. Se mantuvo el término utilizado por la encuestadora, pero algunas de las opciones realmente son herramientas más que prácticas; por ejemplo pizarra digital, burndown chart, etc. Se solicitó indicar cuales utilizan y el resultado puede observarse claramente en la siguiente gráfica.

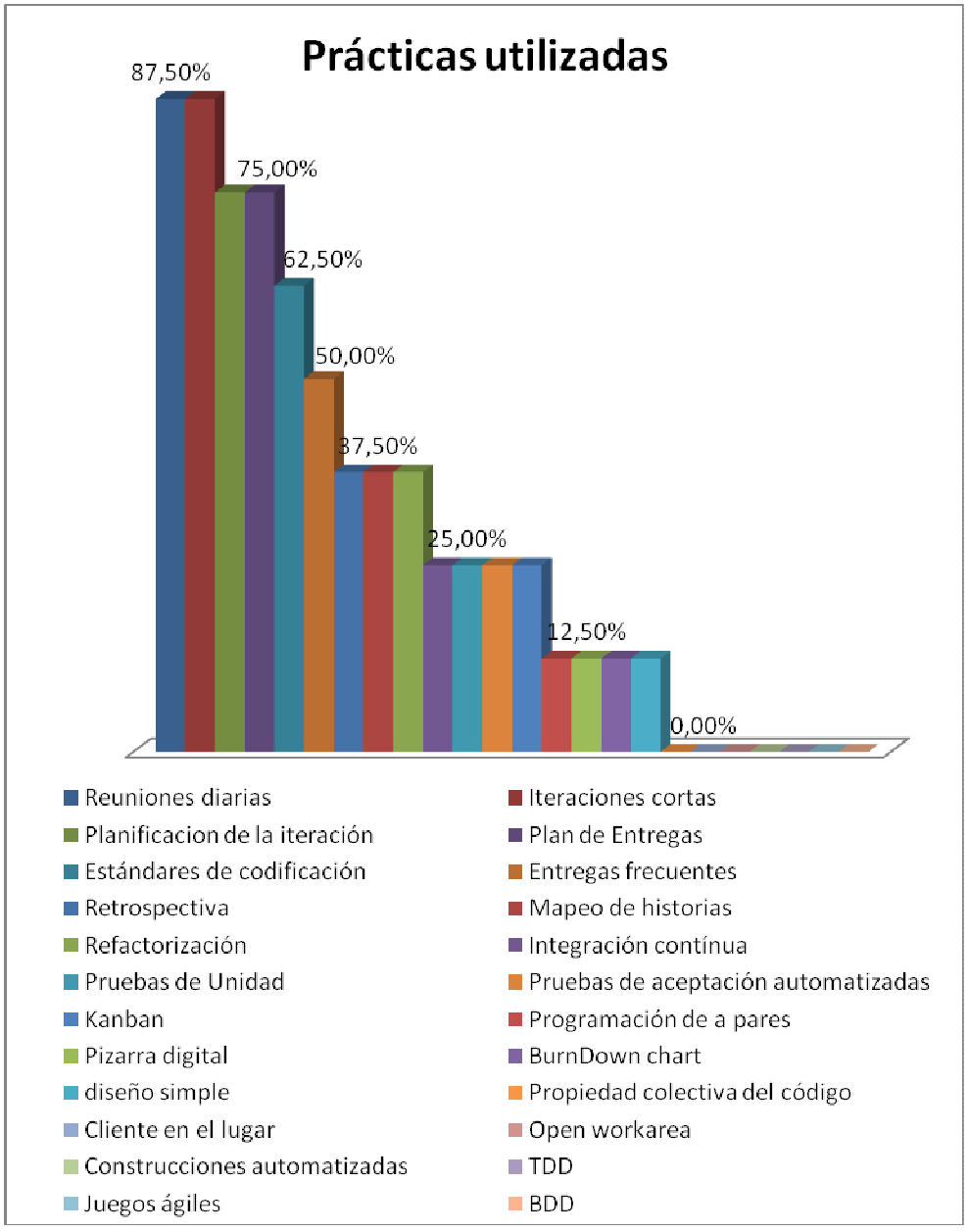


Figura 8.1.11: Prácticas ágiles utilizadas

Las reuniones diarias e iteraciones cortas son las prácticas más utilizadas con el 87.50%. Le siguen la planificación de la iteración y planes de entrega con un 75%. Con un porcentaje un poco menor pero todavía superior al 50% se encuentran los estándares de codificación y con un 50% la reunión de retrospectiva.

Llama la atención el poco porcentaje, 25% de aplicación de pruebas de unidad, integración continua y pruebas de aceptación automatizadas. Es común que el cliente detecte algún error, se soluciones y al cabo de un tiempo vuelva a aparecer, debido a que no hay pruebas automatizadas que corroboren que no se está volviendo a

Las herramientas que no fueron mencionadas por ninguno de los relevados son:

- Propiedad colectiva del código
- Cliente en el lugar
- Open work area
- Construcciones automatizadas
- TDD
- Juegos ágiles
- BDD

Por sobre todo, llama la atención que ninguno realice las prácticas de propiedad colectiva del código, dado que en metodologías ágiles se trata de eliminar cuellos de botella. Y si no se comparte el código, cada persona es responsable de cierta parte y solo esa persona puede modificarla, restando agilidad al proceso de desarrollo. Debido a esto puede entenderse también el bajo porcentaje de quienes realizan integración continua.

- ***Herramientas utilizadas***

Las herramientas ayudan a poder realizar las prácticas ágiles. Esta pregunta buscaba relevar las herramientas utilizadas. Pero, la mayoría no describió herramientas específicas y tres personas no detallaron ninguna, esto es un 37.5%.

La más utilizadas es la pizarra con un 37.50%, seguida con en uso de post-its notes y afiches. Si bien se podría considerar que un afiche podría comportarse como una pizarra utilizando post-its notes, los que eligieron esta segunda alternativa también mencionaron las pizarras, como otra herramienta diferente.

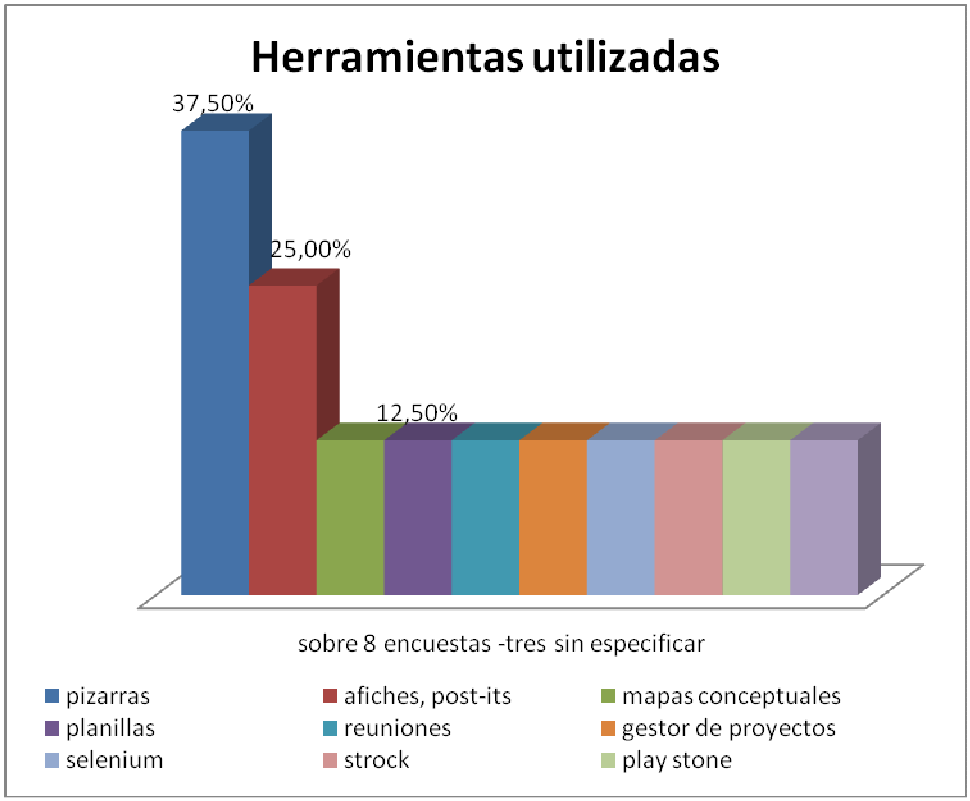


Figura 8.1.12: Herramientas

Selenium es un entorno de pruebas de software para aplicaciones basadas en la web [seleniumhq] y claramente es una herramienta que ayuda a realizar las pruebas. Se mencionan otros que son más herramientas de diseño, como wireframe que representa la estructura visual de un sitio web [Brown]

- ***Riesgos al implementar metodologías ágiles***

Todo proyecto tiene riesgos asociados y un proyecto ágil no está exento a ellos. Se proporcionó un conjunto de 5 riesgos para que fueran priorizados según la importancia que ellos consideren. No hubo una tendencia uniforme, como muestra Figura 8.1.13. Si se calcula el promedio de los valores asignados a cada uno de ellos se obtiene el gráfico de Figura 8.1.14.

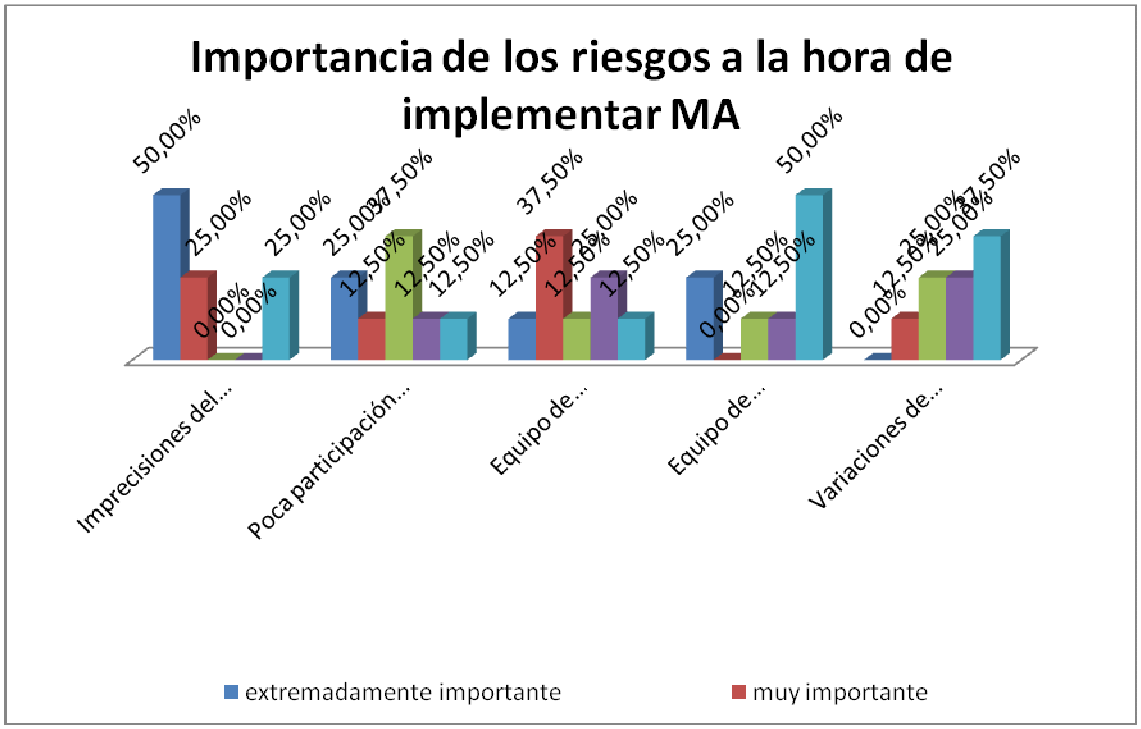


Figura 8.1.13: importancia de los riesgos al trabajar con MA – porcentajes asignados a cada nivel de importancia

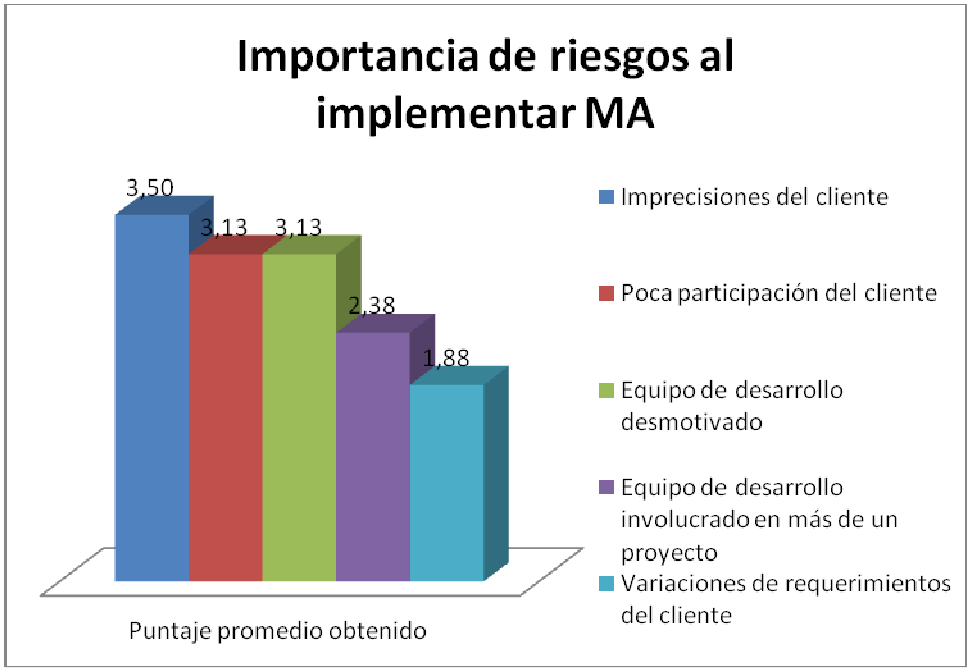


Figura 8.1.14: importancia de los riesgos al trabajar con MA – promedio de los puntajes obtenidos por cada uno

En el gráfico se observa claramente que las imprecisiones del cliente, en promedio son los riesgos más importantes. Superando levemente a la poca participación del cliente y un equipo de

desarrollo desmotivado. Claramente se ve que las variaciones de requerimientos por parte del cliente tienen casi la mitad de importancia que tener un cliente impreciso.

- ***Barreras que impiden implementar metodologías ágiles***

Se brindó siete opciones para que los encuestados señalen aquellos que consideren importantes a la hora de querer implementar una metodología ágil en una organización. En el gráfico siguiente se puede ver que la más importante es tener o no la posibilidad de poder cambiar la cultura organizacional seguida por el tiempo necesario para realizarla transición y la resistencia que pueda haber al cambio. Las restricciones de presupuesto casi no fueron consideradas como barreras para implementar una metodología ágil en la organización.

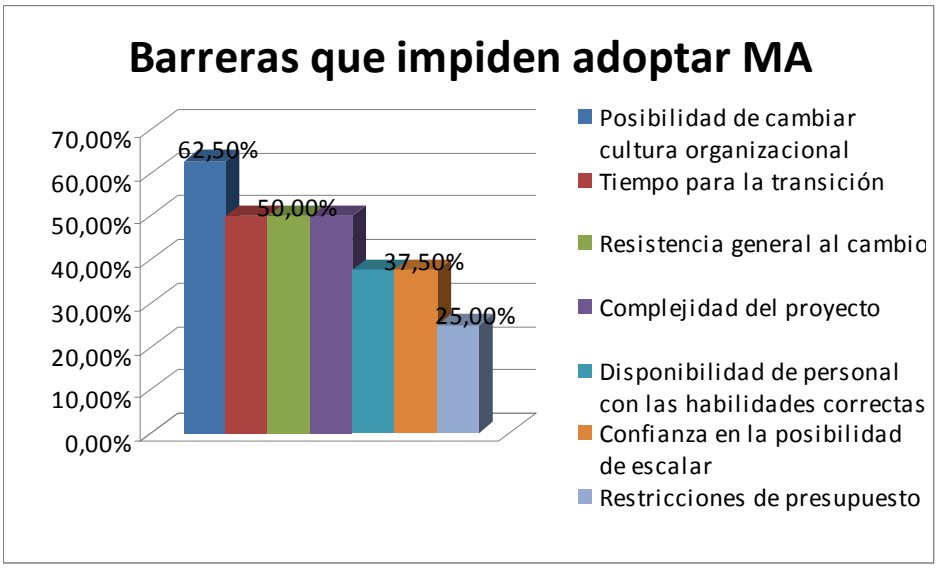


Figura 8.1.15: Barreras que impiden implementar MA

8.1.4 Reflexión o conclusiones generales

En esta encuesta se pudieron observar ciertos puntos que se pueden destacar.

- ***Metodologías utilizadas***

Scrum es utilizado en forma exclusiva por el 50% de los encuestados. Pero hay un grupo grande que trabaja con Scrum y Kanban o Scrumban. Por lo tanto, entre estas tres metodologías se cubre el 87,5%, quedando XP híbrido con el 12.5%. Esto corrobora la idea informal que se tenía de las metodologías que se estaban utilizando en Salta, teniendo una amplia aceptación SCRUM.

- ***Duración de la Iteración***

La amplia mayoría respeta tener iteraciones de 1 a 4 semanas de duración, siendo atípica la duración mencionada de 3 días. Duraciones muy cortas pueden estresar demasiado al equipo y no es posible generalmente llegar a desarrollar algo que realmente le genere valor al cliente. Ninguno mencionó tener duraciones variables en las iteraciones.

- ***Planificación de la iteración***

Tener un 100% de coincidencia en que la planificación se realiza al inicio del proyecto y se va modificando en cada iteración corrobora la filosofía de trabajo que impulsan las metodologías ágiles.

- ***Cliente involucrado en el proyecto***

La amplia mayoría involucra al cliente durante todo el proyecto, pero llama la atención la respuesta de un 12.50% donde el cliente solo participa al inicio y ni siquiera menciona su participación al final del proyecto tampoco. Que exista esta situación, hace pensar que tal vez no se está aplicando una metodología ágil sino que va camino a ello implementando ciertas prácticas. Esto es porque el manifiesto ágil considera importante la participación y colaboración del cliente en todo el desarrollo del proyecto. O bien no tiene en claro la importancia y necesidad de tener al cliente interactuando y participando con el resto del equipo cuando se trabaja con metodologías ágiles.

- ***Entregas Frecuentes***

Solo el 75% respondió afirmativamente respecto a las entregas frecuentes, el 25% restante no realiza estas entregas frecuentes y no cumplen de esta forma el primer principio del Manifiesto ágil donde se define que la prioridad es satisfacer al cliente mediante entregas de software tempranas y continuas.

Dentro de los períodos para realizar las entregas frecuentes, el 50% toma el valor estándar de 2 semanas, habiendo otras respuestas de 4 semanas. Pero nuevamente llama la atención que para un pequeño porcentaje, el período de entregas frecuentes es de 2 días. Nuevamente aparece un período de tiempo demasiado pequeño, inclusive menor que el de la iteración que era de 3 días. Entregar software al cliente funcionando lleva tiempo y realizarlo cada dos días se demoraría demasiado en reuniones y demostraciones.

- ***Colaboración con el cliente***

Los correos electrónicos, las reuniones semanales y las reuniones al final del sprint, fueron las más mencionadas, con un porcentaje de 37.50% cada una. Las charlas o diálogo cara a cara, fueron una de las dos obtuvieron el menor porcentaje. En base a esto, se puede deducir que el diálogo cara a cara no es la forma más común de obtener la colaboración del cliente y por eso se buscan otros medios alternativos, para acercar al cliente con el equipo de desarrollo. Algo que es fundamental al trabajar con metodologías ágiles

- ***Cambio de requerimientos***

Todos consideran que existen cambios en los proyectos, la mitad considera que los requerimientos siempre cambian y la otra mitad considera que algunas veces cambian. Esta diferencia no tiene mayores explicaciones que la experiencia que cada uno de los encuestados haya tenido, dado que los proyectos son diferentes, los clientes son diferentes y los contextos son diferentes. Lo importante, nuevamente es que existen cambios en los requerimientos.

- ***Alcances y costos***

La mitad define alcances por iteración. Un pequeño porcentaje, lo realiza por función, pero existe un 25% que lo realiza al inicio. Todos los que trabajan en organismos públicos además aclararon que no se manejan costos, debido a que el personal cobra un sueldo.

Merece la pena analizar dos datos particulares: el 25% que contestó que realiza al inicio la definición de alcances y costos, y el no considerar costos dentro de organismos públicos.

El primer punto se refiere a definir lo que se quiere realizar y fijar un costo al inicio. Esto no es lo mismo que realizar la planificación, dado que ya se analizó que ésta se realiza en todos los casos al inicio y se va modificando en cada iteración. Si bien no es lo mismo, están relacionados. Se planifica en base a lo que se desea alcanzar. Cuando se ajusta la planificación para cada iteración se debería poder ajustar también el alcance, para darle cierta libertad al cliente. Si se hubiera mantenido el porcentaje del 12.50% para esta opción, tal como estaba en quienes solo tienen la colaboración del cliente al inicio del proyecto sería hasta cierto punto entendible, pero es el doble.

Además todos reconocen que los requerimientos cambian y puede esto hacer variar el alcance. Entonces se puede detectar una contradicción en uno de los encuestados. Tal vez considerando que muchas veces los costos no se renegocian con el cliente, realizó su opción por esta situación.

El segundo punto, no considerar costos dentro de organismos públicos, es algo que hace reflexionar. Todo desarrollo debería tener un cierto análisis de costo – beneficio. No se debería emprender un desarrollo sin saber cuánto recurso consume dentro del organismo, aunque sean sueldo fijo de personal estable. El desarrollo de un sistema podría hacer dedicar más tiempo del personal al nuevo proyecto y luego tener que contratar a otros para realizar otras tareas que se deberían seguir realizando independientemente del proyecto. Podría no calcular directamente costos pero si esfuerzo, que fácilmente se traduce a costos, según los salarios que se perciben en el sector

- ***Desarrollo con reutilización***

El porcentaje de reutilización respecto a proyectos anteriores no es muy alto, ninguno supera el 70%. Y, si bien es una proporción baja, hay quien no reutiliza nada. Si se compara con los valores de reutilización obtenidos del estudio de campo de quienes trabajan con Genexus, realmente se ve que en este grupo es muy bajo.

- ***Prácticas ágiles utilizadas***

Las prácticas que mencionaron más del 50% fueron: las reuniones diarias, las iteraciones cortas, la planificación de la iteración, los planes de entrega, los estándares de codificación. Justo con el 50% se encuentra la reunión de retrospectiva.

Llama la atención el poco porcentaje, 25% de aplicación de pruebas de unidad, integración continua y pruebas de aceptación automatizadas. Pero, por sobre todo, llama la atención que ninguno realice las prácticas de propiedad colectiva del código, dado que en metodologías ágiles se trata de eliminar cuellos de botella. Y si no se comparte el código, cada persona es responsable de cierta parte y solo esa persona puede modificarla, restando agilidad al proceso de desarrollo. Debido a esto puede entenderse también el bajo porcentaje de quienes realizan integración continua.

- ***Herramientas utilizadas***

Hay muy poco uso de herramientas para poder realizar prácticas ágiles. La mayoría no describió herramientas específicas y tres personas no detallaron ninguna. Las pizarras por lejos fueron las más mencionadas, pero no con un alto porcentaje, un 37.50% y le siguen los afiches y post-its. Después no hay coincidencia entre las herramientas utilizadas. Solo uno hizo mención a un entorno de pruebas de software. Otros mencionaron herramientas de diseño más que las que eran requeridas.

En cierta medida, no haber obtenido gran variedad de herramientas para facilitar las prácticas ágiles, se debió a que no se realizan demasiadas prácticas ágiles y muchas de ellas consisten en reuniones donde una pizarra puede colaborar como radiador de información.

- ***Riesgos al implementar metodologías ágiles***

Las imprecisiones del cliente, en promedio son los riesgos más importantes en un desarrollo siguiendo una metodología ágil. Superando levemente a la poca participación del cliente y un equipo de desarrollo desmotivado. Mientras que, las variaciones de requerimientos por parte del cliente tienen casi la mitad de importancia que tener un cliente impreciso.

- ***Barreras que impiden implementar metodologías ágiles***

El tener o no tener la posibilidad de poder cambiar la cultura organizacional donde se quiere implementar una metodología ágil es la barrera más importante, y es entendible. La forma de trabajo es diferente, el control es diferente, el equipo debe poder auto-organizarse, entre otras cosa s y en muchas organizaciones esto no es fácil de conceder.

Las siguientes barreras fueron el tiempo necesario para realizarla transición, la resistencia que pueda haber al cambio y la complejidad del proyecto. Todo cambio requiere un tiempo de aprendizaje, adaptación y entrenamiento y debe poder asignarse ese tiempo. En todo equipo hay personas más resistente al cambio, cuando la gente no está dispuesta a cambiar en su forma de trabajar se está ante otra barrera. Para comenzar a implementar una metodología ágil, se debe tener un proyecto que no sea demasiado complejo, y que permita aprender sobre la marcha.

8.1.5 Reflexión

Si bien la muestra para realizar esta encuesta no es representativa, permite vislumbrar algunos datos de la forma de trabajar con metodologías ágiles. Se pudo ver una tendencia al uso de Scrum, muy amplio donde todos trabajan con iteraciones fijas y ajustan la planificación en cada iteración. La mayoría plantea varios medios para reemplazar el diálogo cara a cara con el cliente.

Si bien todos dicen trabajar con metodologías ágiles, por algunas respuestas detalladas anteriormente algunos de los encuestados no estarían cumpliendo con el manifiesto ágil. Tal vez en estos casos, se está en una etapa inicial de implementación de metodologías ágiles, en un período de aprendizaje.

Las prácticas más utilizadas se relacionan más con la coordinación y gestión del proyecto (reuniones diarias, reuniones de retrospectiva, planificación de la iteración, planes de entregas), aunque, también, más de la mitad define estándares de codificación. Debido a que éstas son las prácticas más utilizadas, no hay un gran uso de herramientas para prácticas ágiles.

Los riesgos considerados más importantes están relacionados con el cliente, sus imprecisiones y su poca participación. También se reconoce la poca motivación del equipo de desarrollo. Y a la hora de implementar una metodología ágil la barrera más importante es tener o no tener la posibilidad de poder cambiar la cultura organizacional donde se quiere implementar.

De todo esto, se toman fundamentalmente dos puntos.

- Se consideran importante las alternativas utilizadas para suplir la falta de un diálogo cara a cara frecuente. En los sectores que trabajan con desarrollo basado en conocimiento, no todos comparten horario de trabajo, ni tienen entrevistas frecuentes con el cliente, por eso las alternativas aquí descriptas son útiles

- Se considera relevante, la identificación de la barrera considerada más importante a la hora de implementar una metodología ágil que es poder cambiar la cultura organizacional. En los sectores de desarrollo basados en conocimiento, cuando decidan trabajar con metodologías ágiles, deberán analizar en particular esta situación, y encontrar una forma de reducir esta barrera.

8.2 ESTRUCTURA DE ENCUESTA

8.2.1 Encuesta en blanco

La siguiente encuesta tiene el único propósito de sondear la experiencia en Salta Capital con Desarrollo Basado en Conocimiento con Genexus y su forma de trabajo. La información relevada se utilizará de manera anónima en un trabajo de tesis de maestría de Ingeniería de Software de la UNLP y en un proyecto de investigación del CIDIA dependiente de la Facultad de ciencias Exactas de la UNSa.

Fecha de realización de la encuesta:

DATOS GENERALES

1. Lugar de Residencia: **Salta Capital / Interior / Otra Provincia**
2. Formación académica: **Secundario / Universitario Incompleto / Universitario completo / Terciario incompleto / Terciario completo**
3. Tipo de organización donde trabaja: **Pública / Privada / No Trabaja**

TRABAJO CON METODOLOGIAS AGILES

1. Se sigue una metodología ágil: **SI – NO**. En caso afirmativo, especifique cuáles utiliza: ……Especifique cuáles incorporaría: …… ……
2. Se realiza un desarrollo iterativo. **SI – NO.** Caso afirmativo, indique duración: ……
3. La planificación del desarrollo se realiza:

   AL INICIO

   AL INICIO, MODIFICANDO EN CADA ITERACION

   OTRA OPCION (especificar)……..

4. El cliente está involucrado:

   AL INICIO

   DURANTE TODO EL PROCESO

   AL FINAL

5. Se realizan entregas frecuentes (a lo sumo cada 2 meses) del producto desarrollado al cliente: **SI – NO.** Caso afirmativo, indique el tiempo: ……..
6. Se tiene colaboración permanente con el cliente: **SI – NO.** Explique cómo: …… …….. ……
7. Los requerimientos cambian a lo largo del proyecto.

   SIEMPRE

   ALGUNAS VECES

CASI NUNCA

NO CAMBIAN.

8. Los alcances y costos se pactan con el cliente:
   AL INICIO

   PARA CADA ITERACION

   OTRA OPCION (especificar)……

9. Se desarrolla con reutilización. **SI – NO.** En caso afirmativo, especifique el porcentaje de reutilización de proyectos anteriores y cómo se seleccionan los artefactos a reutilizar.
   ……
   ……
   ……

10. Especifique cuáles de las siguientes prácticas y herramientas ágiles se siguen en sus proyectos

- Reuniones diarias
- Planificación de la iteración
- Plan de entregas
- Burndown chart
- Retrospectiva
- Integración continua
- Estándares de codificación
- Construcciones automatizadas
- TDD - Test Driven Development
- Pruebas de unidad
- Open workarea
- Pruebas de aceptación automatizadas
- Mapeo de Historias
- Programación de a pares
- Refactorización
- Pizarra digital
- Propiedad colectiva del código
- Kanban
- Cliente en el lugar
- Diseño simple
- Iteraciones cortas
- Entregas frecuentes
- Juegos ágiles
- BDD – Behavior Driven Development

11. Especifique herramientas utilizadas para realizar algunas de las prácticas ágiles seleccionadas……..……
    ……
    ……
    ……

12. Identifique las situaciones de riesgo más comunes detectadas en los proyectos. Numerarlas según su importancia, siendo 1 el riesgo más importante:
    - Imprecisiones del cliente
    - Variaciones de los requerimientos del cliente
    - Poca participación del cliente
    - Equipo de desarrollo desmotivado
    - Equipo de desarrollo involucrado en varios proyectos
    - Otras (especificar) ……

13. Barreras que usted considera impiden adoptar una metodología ágil:
    - Posibilidad de cambiar la cultura organizacional
    - Disponibilidad de personal con las habilidades correctas
    - Resistencia general al cambio
    - Complejidad del proyecto

- Restricciones de presupuesto
- Confianza en la posibilidad de escalar
- Tiempo de transición
- Ninguna

14. Otros comentarios:

……
……
……
……

Muchas gracias por su colaboración.

8.3 ENCUESTAS COMPLETAS

8.3.1 Encuesta 1

Fecha de realización de la encuesta: 31/05/2014

DATOS GENERALES

1. Lugar de Residencia: **Salta Capital**
2. Formación académica: **Universitario completo**
3. Tipo de organización donde trabaja: **Privada**

TRABAJO CON METODOLOGIAS AGILES

1. Se sigue una metodología ágil: **SI**. En caso afirmativo, especifique cuáles utiliza: **XP – Híbrido, Cear Start**.
2. Se realiza un desarrollo iterativo. **SI.** Caso afirmativo, indique duración: **3 días**
3. La planificación del desarrollo se realiza: **AL INICIO, MODIFICANDO EN CADA ITERACION**
4. El cliente está involucrado: **DURANTE TODO EL PROCESO**
5. Se realizan entregas frecuentes (a lo sumo cada 2 meses) del producto desarrollado al cliente: **SI.** Caso afirmativo, indique el tiempo: **2 días.**
6. Se tiene colaboración permanente con el cliente: **SI.**
7. Los requerimientos cambian a lo largo del proyecto. **SIEMPRE**
8. Los alcances y costos se pactan con el cliente**: OTRA OPCION: POR FUNCION**
9. Se desarrolla con reutilización. **SI**. En caso afirmativo, especifique el porcentaje de reutilización de proyectos anteriores y cómo se seleccionan los artefactos a reutilizar. **75% si es similar sino 10% si el producto es nuevo**
10. Especifique cuáles de las siguientes prácticas y herramientas ágiles se siguen en sus proyectos:

- **Programación de a pares**
- **Iteraciones cortas**
- **Diseño simple**

11. Especifique herramientas utilizadas para realizar algunas de las prácticas ágiles seleccionadas: **wireframe, strock, play stone**.
12. Identifique las situaciones de riesgo más comunes detectadas en los proyectos. Numerarlas según su importancia, siendo 1 el riesgo más importante:

- Equipo de desarrollo desmotivado (1)
- Equipo de desarrollo involucrado (1)

13. Barreras que usted considera impiden adoptar una metodología ágil:

- Resistencia general al cambio
- Confianza en la posibilidad de escalar
- Tiempo de transición

14. Otros comentarios: **Sin especificar.**

8.3.2 Encuesta 2

Fecha de realización de la encuesta: 31/05/2014

DATOS GENERALES

1. Lugar de Residencia: **Salta Capital**
2. Formación académica: **Universitario Incompleto**
3. Tipo de organización donde trabaja: **Privada**

TRABAJO CON METODOLOGIAS AGILES

1. Se sigue una metodología ágil: **SI**. En caso afirmativo, especifique cuáles utiliza: **SCRUM, KANBAN.** Especifique cuáles incorporaría: **TDD, BDD.**
2. Se realiza un desarrollo iterativo. **SI.** Caso afirmativo, indique duración: **15 días**
3. La planificación del desarrollo se realiza: **AL INICIO, MODIFICANDO EN CADA ITERACION**
4. El cliente está involucrado: **DURANTE TODO EL PROCESO**
5. Se realizan entregas frecuentes (a lo sumo cada 2 meses) del producto desarrollado al cliente: **SI.** Caso afirmativo, indique el tiempo: **15 días**.
6. Se tiene colaboración permanente con el cliente: **SI.** Explique cómo: **Meeting de presentación al finalizar el sprint.**
7. Los requerimientos cambian a lo largo del proyecto. **SIEMPRE**
8. Los alcances y costos se pactan con el cliente: AL INICIO
9. Se desarrolla con reutilización. **SI.** En caso afirmativo, especifique el porcentaje de reutilización de proyectos anteriores y cómo se seleccionan los artefactos a reutilizar. **50%. Armado de biblioteca de artefactos. Los desarrollos y variables tienen que ser genéricos.**
10. Especifique cuáles de las siguientes prácticas y herramientas ágiles se siguen en sus proyectos

- **Reuniones diarias**
- **Planificación de la iteración**
- **Plan de entregas**
- **Burndown chart**
- **Retrospectiva**
- **Integración continua**
- **Estándares de codificación**
- **TDD - Test Driven Development**
- **Pruebas de unidad**
- **Pruebas de aceptación automatizadas**
- **Mapeo de Historias**
- **Refactorización**
- **Pizarra digital**
- **Kanban**
- **Iteraciones cortas**

11. Especifique herramientas utilizadas para realizar algunas de las prácticas ágiles seleccionadas. **Sin especificar**

12. Identifique las situaciones de riesgo más comunes detectadas en los proyectos. Numerarlas según su importancia, siendo 1 el riesgo más importante:
    - **Imprecisiones del cliente (2)**
    - **Variaciones de los requerimientos del cliente (5)**
    - **Poca participación del cliente (1)**
    - **Equipo de desarrollo desmotivado (3)**

- **Equipo de desarrollo involucrado en varios proyectos (4)**

13. Barreras que usted considera impiden adoptar una metodología ágil:
    - **Posibilidad de cambiar la cultura organizacional**
    - **Disponibilidad de personal con las habilidades correctas**
    - **Resistencia general al cambio**
    - **Complejidad del proyecto**
    - **Restricciones de presupuesto**
    - **Confianza en la posibilidad de escalar**
    - **Tiempo de transición**

14. Otros comentarios: **Sin especificar**

8.3.3 Encuesta 3

Fecha de realización de la encuesta: 31/05/2014

DATOS GENERALES

1. Lugar de Residencia: **Salta Capital**
2. Formación académica: **Universitario Incompleto**
3. Tipo de organización donde trabaja: **Pública**

TRABAJO CON METODOLOGIAS AGILES

1. Se sigue una metodología ágil: **SI**. En caso afirmativo, especifique cuáles utiliza: **SCRUM / KANBAN**
2. Se realiza un desarrollo iterativo. **SI.** Caso afirmativo, indique duración: **3 SEMANAS**
3. La planificación del desarrollo se realiza: **AL INICIO, MODIFICANDO EN CADA ITERACION**
4. El cliente está involucrado: **AL INICIO**
5. Se realizan entregas frecuentes (a lo sumo cada 2 meses) del producto desarrollado al cliente: **NO.**
6. Se tiene colaboración permanente con el cliente: **SI.** Explique cómo: **ESPECIFICACION DE REQUERIMIENTOS**
7. Los requerimientos cambian a lo largo del proyecto: **SIEMPRE.**
8. Los alcances y costos se pactan con el cliente: **PARA CADA ITERACION**
9. Se desarrolla con reutilización. **NO**.
10. Especifique cuáles de las siguientes prácticas y herramientas ágiles se siguen en sus proyectos

- **Reuniones diarias**
- **Planificación de la iteración**
- **Retrospectiva**
- **Estándares de codificación**
- **Kanban**
- **Iteraciones cortas**

11. Especifique herramientas utilizadas para realizar algunas de las prácticas ágiles seleccionadas: **PIZARRAS**

12. Identifique las situaciones de riesgo más comunes detectadas en los proyectos. Numerarlas según su importancia, siendo 1 el riesgo más importante:

- Poca participación del cliente **(3)**
- Equipo de desarrollo desmotivado **(2)**
- Equipo de desarrollo involucrado en varios proyectos **(1)**

13. Barreras que usted considera impiden adoptar una metodología ágil:

- **Posibilidad de cambiar la cultura organizacional**
- **Disponibilidad de personal con las habilidades correctas**
- **Resistencia general al cambio**

14. Otros comentarios: SIN ESPECIFICAR.

8.3.4 Encuesta 4

Fecha de realización de la encuesta: 31/05/2014

DATOS GENERALES

1. Lugar de Residencia: **Salta Capital**
2. Formación académica: **Universitario completo**
3. Tipo de organización donde trabaja: **Pública**

TRABAJO CON METODOLOGIAS AGILES

1. Se sigue una metodología ágil: **SI**. En caso afirmativo, especifique cuáles utiliza: **SCRUM**
2. Se realiza un desarrollo iterativo. **SI.** Caso afirmativo, indique duración: **2 SEMANAS**
3. La planificación del desarrollo se realiza: **AL INICIO, MODIFICANDO EN CADA ITERACION**
4. El cliente está involucrado: **DURANTE TODO EL PROCESO**
5. Se realizan entregas frecuentes (a lo sumo cada 2 meses) del producto desarrollado al cliente: **SI.** Caso afirmativo, indique el tiempo: **CADA 2 SEMANAS, AVANCES DEL PRODUCTO**
6. Se tiene colaboración permanente con el cliente: **SI.** Explique cómo: **REUNIONES, VALIDACIONES DE LOS AVANCES**
7. Los requerimientos cambian a lo largo del proyecto. **SIEMPRE - ALGUNAS VECES**
8. Los alcances y costos se pactan con el cliente: **(ALCANCES) PARA CADA ITERACION – (COSTOS) OTRA OPCION sueldos.**
9. Se desarrolla con reutilización. **SI.** En caso afirmativo, especifique el porcentaje de reutilización de proyectos anteriores y cómo se seleccionan los artefactos a reutilizar. **SE SELECCIONAN POR FUNCIONALIDAD TAMBIÉN SE REUTILIZA LA INTERFAZ, ALGUNOS DISEÑOS. APROXIMADAMENTE 50%**
10. Especifique cuáles de las siguientes prácticas y herramientas ágiles se siguen en sus proyectos

- **Reuniones diarias**
- **Planificación de la iteración**
- **Plan de entregas**
- **Iteraciones cortas**
- **Entregas frecuentes**

11. Especifique herramientas utilizadas para realizar algunas de las prácticas ágiles seleccionadas: **AFICHES, PIZARRAS, PST ITS PARA PLANIFICACIONES, DOCUMENTACIÓN EN PLANILLAS DE PROCESOS, CONCEPTOS, MAPAS CONCEPTUALES, DIAGRAMAS**

12. Identifique las situaciones de riesgo más comunes detectadas en los proyectos. Numerarlas según su importancia, siendo 1 el riesgo más importante:

- Imprecisiones del cliente **(1)**
- Variaciones de los requerimientos del cliente**(4)**
- Poca participación del cliente**(3)**
- Equipo de desarrollo desmotivado**(5)**
- Equipo de desarrollo involucrado en varios proyectos**(5)**

13. Barreras que usted considera impiden adoptar una metodología ágil:

- **Posibilidad de cambiar la cultura organizacional**
- **Complejidad del proyecto**

14. Otros comentarios: **SIN ESPECIFICAR**

8.3.5 Encuesta 5

Fecha de realización de la encuesta: 31/05/2014

DATOS GENERALES

1. Lugar de Residencia: **Salta Capital**
2. Formación académica: **Universitario completo**
3. Tipo de organización donde trabaja: **Pública**

TRABAJO CON METODOLOGIAS AGILES

1. Se sigue una metodología ágil: **SI**. En caso afirmativo, especifique cuáles utiliza: **SCRUM**
2. Se realiza un desarrollo iterativo. **SI.** Caso afirmativo, indique duración: **2 SEMANAS**
3. La planificación del desarrollo se realiza: **AL INICIO, MODIFICANDO EN CADA ITERACION**
4. El cliente está involucrado: **DURANTE TODO EL PROCESO**
5. Se realizan entregas frecuentes (a lo sumo cada 2 meses) del producto desarrollado al cliente: **SI.** Caso afirmativo, indique el tiempo: **CADA 2 SEMANAS ENTREGAS DE AVANCES DEL SISTEMA**.
6. Se tiene colaboración permanente con el cliente: **SI.** Explique cómo: **REUNIONES, VALIDACIONES**
7. Los requerimientos cambian a lo largo del proyecto. **SIEMPRE - ALGUNAS VECES**
8. Los alcances y costos se pactan con el cliente: **PARA CADA ITERACION (ALCANCES) - OTRA OPCION (COSTOS): SUELDOS**
9. Se desarrolla con reutilización. **SI.** En caso afirmativo, especifique el porcentaje de reutilización de proyectos anteriores y cómo se seleccionan los artefactos a reutilizar. **APROXIMADAMENTE 60%. SE SELECCIOAN POR FUNCIONALIDAD**

10. Especifique cuáles de las siguientes prácticas y herramientas ágiles se siguen en sus proyectos

- **Reuniones diarias**
- **Planificación de la iteración**
- **Plan de entregas**
- **Iteraciones cortas**
- **Entregas frecuentes**

11. Especifique herramientas utilizadas para realizar algunas de las prácticas ágiles seleccionadas: **AFICHES, PIZARRAS, POST ITS NOTES.**

12. Identifique las situaciones de riesgo más comunes detectadas en los proyectos. Numerarlas según su importancia, siendo 1 el riesgo más importante:

- Imprecisiones del cliente **(1)**
- Variaciones de los requerimientos del cliente **(4)**
- Poca participación del cliente **(3)**
- Equipo de desarrollo desmotivado **(2)**
- Equipo de desarrollo involucrado en varios proyectos **(1)**

13. Barreras que usted considera impiden adoptar una metodología ágil:

- **Posibilidad de cambiar la cultura organizacional**
- **Complejidad del proyecto**

14. Otros comentarios: **SIN ESPECIFICAR**

8.3.6 Encuesta 6

Fecha de realización de la encuesta: 31/05/2014

DATOS GENERALES

1. Lugar de Residencia: **Salta Capital**
2. Formación académica: **Universitario Incompleto**
3. Tipo de organización donde trabaja: **Privada**

TRABAJO CON METODOLOGIAS AGILES

1. Se sigue una metodología ágil: **SI.** En caso afirmativo, especifique cuáles utiliza: **SCRUM**
2. Se realiza un desarrollo iterativo. **SI.** Caso afirmativo, indique duración: **4 SEMANAS**
3. La planificación del desarrollo se realiza: **AL INICIO, MODIFICANDO EN CADA ITERACION**
4. El cliente está involucrado: **DURANTE TODO EL PROCESO**
5. Se realizan entregas frecuentes (a lo sumo cada 2 meses) del producto desarrollado al cliente: **SI.** Caso afirmativo, indique el tiempo: **1 MES**.
6. Se tiene colaboración permanente con el cliente: **SI.** Explique cómo: **SKYPE / E-MAILS.**
7. Los requerimientos cambian a lo largo del proyecto. **ALGUNAS VECES**
8. Los alcances y costos se pactan con el cliente: **PARA CADA ITERACION**

9. Se desarrolla con reutilización. **SI**. En caso afirmativo, especifique el porcentaje de reutilización de proyectos anteriores y cómo se seleccionan los artefactos a reutilizar. **20% SE REUTILIZA POR FUNCIONALIDAD**
10. Especifique cuáles de las siguientes prácticas y herramientas ágiles se siguen en sus proyectos

- **Reuniones diarias**
- **Planificación de la iteración**
- **Plan de entregas**
- **Integración continua**
- **Estándares de codificación**
- **Pruebas de unidad**
- **Pruebas de aceptación automatizadas**
- **Mapeo de Historias**
- **Refactorización**
- **Iteraciones cortas**
- **Entregas frecuentes**

11. Especifique herramientas utilizadas para realizar algunas de las prácticas ágiles seleccionadas: **MEETINGS, DOCUMENTACION DE ESTÁNDARES A CUMPLIR, SELENIUM, GESTOR DE PROYECTOS**
12. Identifique las situaciones de riesgo más comunes detectadas en los proyectos. Numerarlas según su importancia, siendo 1 el riesgo más importante:

- Imprecisiones del cliente **(2)**
- Variaciones de los requerimientos del cliente **(3)**
- Poca participación del cliente **(1)**
- Equipo de desarrollo desmotivado **(4)**
- Equipo de desarrollo involucrado en varios proyectos **(5)**

13. Barreras que usted considera impiden adoptar una metodología ágil:

- **Posibilidad de cambiar la cultura organizacional**
- **Resistencia general al cambio**

14. Otros comentarios: **SIN ESPECIFICAR**

8.3.7 Encuesta 7

Fecha de realización de la encuesta: 31/05/2014

DATOS GENERALES

1. Lugar de Residencia: **Salta Capital**
2. Formación académica: **Universitario completo**
3. Tipo de organización donde trabaja: **Pública / Privada**

TRABAJO CON METODOLOGIAS AGILES

1. Se sigue una metodología ágil: **SI**. En caso afirmativo, especifique cuáles utiliza: **SCRUM - SCRUMBAN**
2. Se realiza un desarrollo iterativo. **SI.** Caso afirmativo, indique duración: **4 SEMANAS**
3. La planificación del desarrollo se realiza: **AL INICIO, MODIFICANDO EN CADA ITERACION**
4. El cliente está involucrado: **DURANTE TODO EL PROCESO**

5. Se realizan entregas frecuentes (a lo sumo cada 2 meses) del producto desarrollado al cliente: **NO.**
6. Se tiene colaboración permanente con el cliente: **SI.** Explique cómo: **E-AMILS, CONFERENCIAS, REUNIONES SEMANALES.**
7. Los requerimientos cambian a lo largo del proyecto. **ALGUNAS VECES**
8. Los alcances y costos se pactan con el cliente: **AL INICIO**
9. Se desarrolla con reutilización. **SI**. En caso afirmativo, especifique el porcentaje de reutilización de proyectos anteriores y cómo se seleccionan los artefactos a reutilizar. **SIN ESPECIFICAR**
10. Especifique cuáles de las siguientes prácticas y herramientas ágiles se siguen en sus proyectos

- **Reuniones diarias**
- **Planificación de la iteración**
- **Plan de entregas**
- **Retrospectiva**
- **Estándares de codificación**
- **Refactorización**
- **Iteraciones cortas**
- **Entregas frecuentes**

11. Especifique herramientas utilizadas para realizar algunas de las prácticas ágiles seleccionadas. **SIN ESPECIFICAR**
12. Identifique las situaciones de riesgo más comunes detectadas en los proyectos. Numerarlas según su importancia, siendo 1 el riesgo más importante:

- Imprecisiones del cliente **(1)**
- Variaciones de los requerimientos del cliente **(3)**
- Poca participación del cliente **(2)**
- Equipo de desarrollo desmotivado **(4)**
- Equipo de desarrollo involucrado en varios proyectos **(5)**

13. Barreras que usted considera impiden adoptar una metodología ágil:

- **Disponibilidad de personal con las habilidades correctas**
- **Complejidad del proyecto**
- **Tiempo de transición**

14. Otros comentarios: **SIN ESPECIFICAR**

8.3.8 Encuesta 8

Fecha de realización de la encuesta: 31/05/2014

DATOS GENERALES

1. Lugar de Residencia: **Salta Capital**
2. Formación académica: **Universitario Incompleto**
3. Tipo de organización donde trabaja: **Privada**

TRABAJO CON METODOLOGIAS AGILES

1. Se sigue una metodología ágil: **SI**. En caso afirmativo, especifique cuáles utiliza: **SCRUM**

2. Se realiza un desarrollo iterativo. **SI.** Caso afirmativo, indique duración: **4 SEMANAS**
3. La planificación del desarrollo se realiza: **AL INICIO, MODIFICANDO EN CADA ITERACION**
4. El cliente está involucrado: **DURANTE TODO EL PROCESO**
5. Se realizan entregas frecuentes (a lo sumo cada 2 meses) del producto desarrollado al cliente: **SI.** Caso afirmativo, indique el tiempo: **1 MES**
6. Se tiene colaboración permanente con el cliente: **SI.** Explique cómo: **EMAILS O CHARLAS**
7. Los requerimientos cambian a lo largo del proyecto. **ALGUNAS VECES**
8. Los alcances y costos se pactan con el cliente: **SIN ESPECIFICAR**
9. Se desarrolla con reutilización. **SI**. En caso afirmativo, especifique el porcentaje de reutilización de proyectos anteriores y cómo se seleccionan los artefactos a reutilizar: **SE REUTILIZA EN UN 40%. SE SELECCIONA POR LA FUNCIONALIDAD DEL PROYECTO, POR SIMILITUD.**
10. Especifique cuáles de las siguientes prácticas y herramientas ágiles se siguen en sus proyectos
    - **Reuniones diarias**
    - **Plan de entregas**
    - **Estándares de codificación**
    - **Mapeo de Historias**
11. Especifique herramientas utilizadas para realizar algunas de las prácticas ágiles seleccionadas: **SIN ESPECIFICAR**
12. Identifique las situaciones de riesgo más comunes detectadas en los proyectos. Numerarlas según su importancia, siendo 1 el riesgo más importante:

- Imprecisiones del cliente **(1)**
- Variaciones de los requerimientos del cliente **(2)**
- Poca participación del cliente **(4)**
- Equipo de desarrollo desmotivado **(5)**
- Equipo de desarrollo involucrado en varios proyectos **(3)**

13. Barreras que usted considera impiden adoptar una metodología ágil:

- **Confianza en la posibilidad de escalar**
- **Restricciones de presupuesto**
- **Tiempo de transición**

14. Otros comentarios: **SIN ESPECIFICAR**

9 ANEXO3: ENCUESTAS SOBRE EXPERIENCIA SOBRE DESARROLLO BASADO EN CONOCIMIENTO CON GENEXUS SIGUIENDO PRÁCTICAS ÁGILES

9.1 ESTRUCTURA DE ENCUESTA

9.1.1 Encuesta en blanco

La siguiente encuesta tiene el único propósito de sondear la experiencia realizada en la cátedra Desarrollo Basado en Conocimiento I su información será utilizada de manera anónima en un trabajo de tesis de maestría de Ingeniería de Software de la UNLP y en un proyecto de investigación del CIDIA dependiente de la Facultad de Ciencias Exactas de la UNSa.

Fecha de realización de la encuesta:

DATOS LABORALES

14. Trabaja además de estudiar: **SI - NO.**
15. Trabaja en organismo **PUBLICO - PRIVADO**.
16. Cantidad de hs semanales que trabaja: ……..……...
17. Tipo de trabajo que realiza: ……..……....
18. Si realiza desarrollo de sw,
    - Especificar metodología de desarrollo que utiliza en el trabajo:…….. ……
    - Varios proyectos realizados al mismo tiempo: **NUNCA – RARAS EXCEPCIONES - A VECES – FRECUENTEMENTE – SIEMPRE**
    - Comparte horario de trabajo con el resto del equipo de trabajo: **SI – NO – SE SOLAPAN**
    - Las personas involucradas comparten el lugar de trabajo: **SI – NO**.
    - Existe un líder de proyecto: **SI – NO**

DATOS DESARROLLO BASADO EN CONOCIMIENTO

26. Experiencia previa a la asignatura desarrollando con Genexus: **Nula – Mínima - Moderada – Alta - Nivel Experto**
27. Si tiene experiencia previa indique **tiempo** de uso de Genexus: ……..
28. Cantidad de proyectos desarrollados con Genexus: ……..……....

FORMA DE TRABAJO DURANTE LA EXPERIENCIA

1. Se trabajó compartiendo el lugar de trabajo**: SI - NO - A VECES**.
2. Cómo se suplió la falta de lugar de trabajo común: …… ……..……
3. Considera importante compartir el lugar de trabajo: **SI - NO - INDIFERENTE**. Justifique: ……..…… ……..……
4. Se trabajó compartiendo los horarios dedicados a la experiencia: **SI – NO - A VECES**.
5. Considera importante compartir el horario de trabajo: **SI - NO - INDIFERENTE**. Justifique: ……..…… ……..……

6. Cuáles fueron los estándares y lineamientos definidos dentro de su equipo.

   ……
   ……

7. Cuales fueron las actividades a realizar definidas en el tablero de tareas. Explique el DoD de cada una.

   ……
   ……
   ……
   ……

8. La definición de DoD fue **SIMPLE – MEDIO – COMPLEJO**.
9. La definición de DoD considera que fue importante. **SI – ALGO – NO**
10. La forma de trabajo fue adaptándose durante el avance de la experiencia **SI – NO**. En caso de respuesta afirmativa. Ejemplifique: ……
    ……
    ……
11. Considera importante tener un equipo de desarrollo autoorganizado: **SI - NO - INDIFERENTE**
12. Se realizaron todos los tipos de reuniones especificadas. **SI – NO**. En caso de no haber realizado alguna por favor justifique porqué no se realizó. ……...
    ……
    ……
13. Indique la reunión que considera más importante justificando la respuesta. ……..
    ……
    ……
14. Considera que alguna de las reuniones no es necesaria, **SI - NO**. Justifique. ...……
    ……
    ……
15. Se comprendieron los roles involucrados en la experiencia: **SI – NO**. En caso de respuesta negativa, justifique……
    ……
16. Indique el o los ROL/ES que desempeñó durante la experiencia……
    ……
17. Nombre las tres dificultades mayores enfrentadas indicando para cada uno el nivel del impacto: **MUY BAJO, BAJO, MODERADO, ALTO, MUY ALTO.**

    ……
    ……
    ……
    ……
    ……
    ……

OPINION SOBRE EL INICIO AGIL

1. Considera necesaria la participación de todos los participantes en la primera reunión. **INUTIL – ALGO UTIL – MUY UTIL**. Justifique: ……
   ……
2. La definición de la visión siguiendo la práctica sugerida fue **SIMPLE – POCO COMPLEJA – COMPLEJA**
3. En caso de no haberse realizado, el ejercicio de la visión, considera que igualmente todos compartirían la misma visión. **SI – NO**
4. Siguiendo la técnica de identificación de los vecinos, determinar los stakeholders resultó: **FACIL – ALGO COMPLEJO – COMPLEJO**

5. Si usted ejerció el rol de coordinador y participó en la reunión de planificación inicial, considera que:
    - Las tareas propuestas son: **SIMPLE – ALGO COMPLEJO – COMPLEJO**.
    - Como fue la participación del cliente del gimnasio**: PASIVA – ALGO ACTIVA – MUY ACTIVA.**
    - Como fue la participación del cliente de la librería: **PASIVA – ALGO ACTIVA – MUY ACTIVA.**

OPINION SOBRE LA PRODUCCION

1. Los cambios en requerimientos o nuevos requerimientos fueron aceptados por el equipo: **SI – NO**
2. Los objetos a modificar para implementar los cambios en requerimientos fue **SIMPLE – MEDIO –COMPLEJO.** Amplie: ……..……..……
   ……..……..……
   ……..……..
3. Se realizó un desarrollo iterativo: **SI – NO**. Indique duración de la iteración: ……..
   ……..……..……
4. Se realizó un diseño utilizando diagramas UML**: SI – NO**. Especifique cuales utilizó: …
   ……..……..……
   ……..……..……
5. Para realizar las pruebas, se trabajó con algún Framework específico. **SI –NO.**
   Especifique cual utilizó o bien porqué no utilizó. ……..……..……
   ……..……..
6. Explique cómo garantizó que los cambios no propagaran errores en otras secciones que ya funcionaban bien……..……..……
   ……..……..
7. Se fijaron WIP (límites de tareas por actividad) en la pizarra de tareas. **SI – NO**. Justifique.……..……..……
   ……..……..
8. Se realizaron entregas parciales de producto desarrollado al cliente. **SI – NO**.
   Especifique el tiempo ……..……..……..…
   ……..……..…

OPINION SOBRE EL RITUAL DE FINALIZACIÓN

1. En la reunión de entrega final, indique para cada sistema si considera que es necesario, y que beneficios se obtiene de esta reunión
   ……..……..……
   ……..……..…
2. De la reunión de reflexión final de cada sistema, especifique qué opiniones surgieron.
   ……..……..……
   ……..……..…
3. Considera que hubiera sigo más productivo tener a los stakeholders de los sistemas en la reunión de revisión final. **SI / NO.** Explique:
   ……..……..……
   ……..……..…
   ……..……..……
   ……..……..…

OPINION GENERLA SOBRE LA EXPERIENCIA

1. Al inicio del proyecto consideraba posible la realización de la experiencia. **SI - NO.** Justifique su respuesta. ……

2. Según la experiencia realizada, esta: **INSATISFECHO – CONFORME - MUY CONFORME** con los resultados obtenidos. Justifique ……
……. ……
3. Se tuvo colaboración permanente con el cliente del gimnasio. **SI – NO**. Explique como:
……..
……
4. Se tuvo colaboración permanente con el cliente de la librería. **SI – NO**. Explique como:
……..……
……
5. Fue necesaria mayor participación de alguno de los clientes aparte de las reuniones establecidas: **SI – NO**. Si respuesta es afirmativa explique: ……
……
6. Especifique las herramientas utilizadas y los beneficios de cada una:

……
……
7. Especifique herramientas utilizadas para el seguimiento del proyecto y su opinión sobre la misma:……
……..
8. Quién se encargaba de actualizar el tablero: ……
……..……
9. Se desarrolló con reutilización**. SI – NO**.
   - Si es afirmativo ¿Cómo se seleccionan los artefactos a reutilizar?

   ……
   ……
   - ¿Considera que se demora menos tiempo reutilizando? Justifique

   ……
   ……
10. Trabajaría con Genexus**. SI – NO.** Justifique

    ……
    ……
11. Otros comentarios:

    ……
    ……

    ……
    ……

Muchas gracias por su colaboración.

9.2 ENCUESTAS COMPLETAS

9.2.1 Encuesta 1

Fecha de realización de la encuesta: Junio 2015

DATOS LABORALES

1. Trabaja además de estudiar: **NO.**

DATOS DESARROLLO BASADO EN CONOCIMIENTO

1. Experiencia previa a la asignatura desarrollando con Genexus: **Nula**
2. Si tiene experiencia previa indique **tiempo** de uso de Genexus: ……..
3. Cantidad de proyectos desarrollados con Genexus: ……....

FORMA DE TRABAJO DURANTE LA EXPERIENCIA

1. Se trabajó compartiendo el lugar de trabajo**: A VECES**.
2. Cómo se suplió la falta de lugar de trabajo común: **e-mail, mensajería instantánea**
3. Considera importante compartir el lugar de trabajo: **SI**. Justifique**: hay cosas que necesariamente deben resolverse personalmente**
4. Se trabajó compartiendo los horarios dedicados a la experiencia: **A VECES**.
5. Considera importante compartir el horario de trabajo: **SI**. Justifique: **hay cosas que necesariamente deben resolverse personalmente.**
6. Cuáles fueron los estándares y lineamientos definidos dentro de su equipo. **Concentrarse 100% en los momentos en los que el equipo estaba reunido**
7. Cuales fueron las actividades a realizar definidas en el tablero de tareas. Explique el DoD de cada una.

   Análisis: definición de nivel de desarrollo (reutilización o nuevo)

   Desarrollo: Porción de funcionalidad de software

   Prueba: Funciones sin errores y aceptados por el cliente

   Producción: Instalado en el cliente
8. La definición de DoD fue **MEDIO**.
9. La definición de DoD considera que fue importante. **SIN IMPORTANCIA**
10. La forma de trabajo fue adaptándose durante el avance de la experiencia **SI**. En caso de respuesta afirmativa. Ejemplifique**: se planteo usar GxServer, luego se decidió abandonarlo y trabajar mediante importaciones y exportaciones**
11. Considera importante tener un equipo de desarrollo autoorganizado: **SI.**
12. Se realizaron todos los tipos de reuniones especificadas. **NO**. En caso de no haber realizado alguna por favor justifique porqué no se realizó: **reunión de finalización - entrega final por el momento**
13. Indique la reunión que considera más importante justificando la respuesta**: Reunión de Seguimiento, trata aspectos concretos del sistema y su desarrollo.**
14. Considera que alguna de las reuniones no es necesaria, **SI**. Justifique: **hay cuestiones que podrían obviarse porque sobrecargan innecesariamente el trabajo y estructura demasiado**
15. Se comprendieron los roles involucrados en la experiencia: **SI**
16. Indique el o los ROL/ES que desempeñó durante la experiencia: **Analista, Coordinador, desarrollador.**

17. Nombre las tres dificultades mayores enfrentadas indicando para cada uno el nivel del impacto:

- **estructuración del tiempo MUY ALTO**
- **dificultades de la herramienta ALTO**
- **trabajo simultáneo en diferentes sistemas BAJO**

OPINION SOBRE EL INICIO AGIL

1. Considera necesaria la participación de todos los participantes en la primera reunión. **INUTIL**. Justifique: **El analista después describe al grupo las historias de usuario y termina ordenándolas por prioridades**
2. La definición de la visión siguiendo la práctica sugerida fue **POCO COMPLEJA**
3. En caso de no haberse realizado, el ejercicio de la visión, considera que igualmente todos compartirían la misma visión. **NO**
4. Siguiendo la técnica de identificación de los vecinos, determinar los stakeholders resultó: **FACIL**
5. Si usted ejerció el rol de coordinador y participó en la reunión de planificación inicial, considera que:
   - Las tareas propuestas son: **SIMPLE**.
   - Como fue la participación del cliente del gimnasio**: ALGO ACTIVA.**
   - Como fue la participación del cliente de la librería: **ALGO ACTIVA.**

OPINION SOBRE LA PRODUCCION

1. Los cambios en requerimientos o nuevos requerimientos fueron aceptados por el equipo: **SI**

2. Los objetos a modificar para implementar los cambios en requerimientos fue **MEDIO.** Amplíe: **Por cuestiones relativas al trial de genexus**
3. Se realizó un desarrollo iterativo: **SI**. Indique duración de la iteración: **dos iteraciones no bien definidas de 2 semanas aprox**
4. Se realizó un diseño utilizando diagramas UML**: NO**.
5. Para realizar las pruebas, se trabajó con algún Framework específico. **NO.** Especifique cual utilizó o bien porqué no utilizó**: complejidad de aplicación del framework**

6. Explique cómo garantizó que los cambios no propagaran errores en otras secciones que ya funcionaban bien**: mediante evaluaciones de reportes de referencia generados por Genexus**

Se fijaron WIP (límites de tareas por actividad) en la pizarra de tareas. **NO**. Justifique. **No justifica.**

7. Se realizaron entregas parciales de producto desarrollado al cliente. **SI**.

   Especifique el tiempo **No especifica**

OPINION SOBRE EL RITUAL DE FINALIZACIÓN

1. En la reunión de entrega final, indique para cada sistema si considera que es necesario, y que beneficios se obtiene de esta reunión. **SI, muy útil en el equipo para demostrar su trabajo y aprender nuevas cosas para el futuro**
2. De la reunión de reflexión final de cada sistema, especifique qué opiniones surgieron.

   Hubo algunas molestias por el producto final obtenido puesto que no se pudieron concretar las ideas fielmente debido a las limitaciones del trial

3. Considera que hubiera sigo más productivo tener a los stakeholders de los sistemas en la reunión de revisión final. **NO.** Explique: **no explica.**

OPINION GENERLA SOBRE LA EXPERIENCIA

1. Al inicio del proyecto consideraba posible la realización de la experiencia. **NO.** Justifique su respuesta: **constituía una carga adicional a las materias que se cursaban que de por sí eran pesadas. En mi caso además sabía que se iba a complicar por el trabajo**
2. Según la experiencia realizada, esta: **CONFORME** con los resultados obtenidos. Justifique: **podría haber sido mejor**
3. Se tuvo colaboración permanente con el cliente del gimnasio. **SI**. Explique como: **sugerencias y recomendaciones para cumplir con las funcionalidades**
4. Se tuvo colaboración permanente con el cliente de la librería. **SI**. Explique como: **ayuda en la determinación clara de las funcionalidades y prioridades de cada una**
5. Fue necesaria mayor participación de alguno de los clientes aparte de las reuniones establecidas: **NO**.
6. Especifique las herramientas utilizadas y los beneficios de cada una
    - **Genexus para desarrollo**
    - **Kanbanize para seguimiento del equipo**
    - **GXServer posteriormente se descartó**
    - **Excel para cumplimentar el seguimiento y burndown chart**
7. Especifique herramientas utilizadas para el seguimiento del proyecto y su opinión sobre la misma: **Kanbanize, bastante completa, y compleja solo usamos una pequeña parte y Excel, muy útil para hacer seguimiento rápido de evolución mediante gráficos**
8. Quién se encargaba de actualizar el tablero: **todos**
9. Se desarrolló con reutilización**. SI**.
    - Si es afirmativo ¿Cómo se seleccionan los artefactos a reutilizar?

    según similitud de funcionalidad…
    - ¿Considera que se demora menos tiempo reutilizando? Justifique

    Sí, pero no mucho. A veces es más difícil entender algo que hizo otro que hacerlo completamente
10. Trabajaría con Genexus**. SI.** Justifique: **No justifica**
11. Otros comentarios: **Para que la experiencia tenga mayor sentido pienso que debería plantearse al inicio de la materia o bien definir una nueva optativa para profundizar en metodologías ágiles**

Muchas gracias por su colaboración.

9.2.2 Encuesta 2

Fecha de realización de la encuesta: Junio 2015

DATOS LABORALES

1. Trabaja además de estudiar: **NO.**

DATOS DESARROLLO BASADO EN CONOCIMIENTO

4. Experiencia previa a la asignatura desarrollando con Genexus: **Nula**

FORMA DE TRABAJO DURANTE LA EXPERIENCIA

2. Se trabajó compartiendo el lugar de trabajo**: A VECES**.

3. Cómo se suplió la falta de lugar de trabajo común: **conectados por internet**
4. Considera importante compartir el lugar de trabajo: **SI**. Justifique**: aclarar dudas de manera más rápida, compartir conocimientos, y cooperar todos para resolver los problemas que se presenten**
5. Se trabajó compartiendo los horarios dedicados a la experiencia: **SI**.
6. Considera importante compartir el horario de trabajo: **SI**. Justifique: **No justifica.**
7. Cuáles fueron los estándares y lineamientos definidos dentro de su equipo. **Dividir las tareas entre todos los integrantes y cada uno enfocarse en realizar las tareas asignadas de la MEJOR manera y entregarlas en un timepo preestablecido**
8. Cuales fueron las actividades a realizar definidas en el tablero de tareas. Explique el DoD de cada una.

   En progreso /finalizados o terminados
9. La definición de DoD fue **SIMPLE**.
10. La definición de DoD considera que fue importante. **CIERTA IMPORTANCIA**
11. La forma de trabajo fue adaptándose durante el avance de la experiencia **NO contesta**. En caso de respuesta afirmativa. Ejemplifique**: No contesta**
12. Considera importante tener un equipo de desarrollo autoorganizado: **SI.**
13. Se realizaron todos los tipos de reuniones especificadas. **SI**.
14. Indique la reunión que considera más importante justificando la respuesta**: Reunión de Seguimiento, para aclarar dudas y permite después resolver problemas.**
15. Considera que alguna de las reuniones no es necesaria, **NO**. Justifique: **todas son necesarias dado que permiten aumentar la productividad del trabajo.**
16. Se comprendieron los roles involucrados en la experiencia: **SI**
17. Indique el o los ROL/ES que desempeñó durante la experiencia: **todos.**
18. Nombre las tres dificultades mayores enfrentadas indicando para cada uno el nivel del impacto:
    - **Arreglo de horarios: ALTO**
    - **Nivel de conocimiento de Genexus dispar: MODERADO**
    - **Tiempo de entrega: MODERADO**

OPINION SOBRE EL INICIO AGIL

1. Considera necesaria la participación de todos los participantes en la primera reunión. **MUY UTIL**. Justifique: **para que todos conozcan el objetivo del proyecto, las funcionalidades requeridas**
2. La definición de la visión siguiendo la práctica sugerida fue **POCO COMPLEJA**
3. En caso de no haberse realizado, el ejercicio de la visión, considera que igualmente todos compartirían la misma visión. **NO CONTESTA**
4. Siguiendo la técnica de identificación de los vecinos, determinar los stakeholders resultó: **FACIL**
5. Si usted ejerció el rol de coordinador y participó en la reunión de planificación inicial, considera que: **NO EJERCIO.**

OPINION SOBRE LA PRODUCCION

1. Los cambios en requerimientos o nuevos requerimientos fueron aceptados por el equipo: **SI**
2. Los objetos a modificar para implementar los cambios en requerimientos fue **COMPLEJO.** Amplíe: **Algunos cambios no pudieron implementarse por no contar con algunos conocimientos de genexus**

3. Se realizó un desarrollo iterativo: **NO**. Indique duración de la iteración: **no contesta**
4. Se realizó un diseño utilizando diagramas UML**: NO**.
5. Para realizar las pruebas, se trabajó con algún Framework específico. **NO.** Especifique cual utilizó o bien porqué no utilizó**: no contesta**
6. Explique cómo garantizó que los cambios no propagaran errores en otras secciones que ya funcionaban bien**: realizando múltiples pruebas de todo el sistema y analizando los objetos sobre los que iba a impactar los cambios realizados**
7. Se fijaron WIP (límites de tareas por actividad) en la pizarra de tareas. **SI**. Justifique. **para poder cumplir con los plazos de entrega del proyecto.**
8. Se realizaron entregas parciales de producto desarrollado al cliente. **SI**. Especifique el tiempo **2 semanas.**

OPINION SOBRE EL RITUAL DE FINALIZACIÓN

1. En la reunión de entrega final, indique para cada sistema si considera que es necesario, y que beneficios se obtiene de esta reunión. **SI, permite evaluar el nivel de satisfacción del cliente con el producto desarrollado y experiencia para aprender de los errores cometidos**
2. De la reunión de reflexión final de cada sistema, especifique qué opiniones surgieron. **productos bien desarrollado pero el de librería con errores por el trial**
3. Considera que hubiera sigo más productivo tener a los stakeholders de los sistemas en la reunión de revisión final. **NO.** Explique: **no explica**

OPINION GENERLA SOBRE LA EXPERIENCIA

1. Al inicio del proyecto consideraba posible la realización de la experiencia. **NO.** Justifique su respuesta: **tiempo de entrega corto y muchas funcionalidades**
2. Según la experiencia realizada, esta: **MUY CONFORME** con los resultados obtenidos. Justifique: **se pudo culminar con dos proyectos en tiempo y forma y funcionalidades requeridas**
3. Se tuvo colaboración permanente con el cliente del gimnasio. **SI**. Explique como: **explicando los cambios que quería en base a lo que se presentó primero**
4. Se tuvo colaboración permanente con el cliente de la librería. **SI**. Explique como: **seguimiento permanente de lo que estaba realizando, especificando cambios que debían realizarse en base a las presentaciones realizadas**
5. Fue necesaria mayor participación de alguno de los clientes aparte de las reuniones establecidas: **NO**.
6. Especifique las herramientas utilizadas y los beneficios de cada una
    - **Kanbanize mejor organización de las tareas**
    - **GXServer no se pudo utilizar**
7. Especifique herramientas utilizadas para el seguimiento del proyecto y su opinión sobre la misma: **Kanbanize, ídem anterior**
8. Quién se encargaba de actualizar el tablero: **Coordinador**
9. Se desarrolló con reutilización**. SI**.
    - Si es afirmativo ¿Cómo se seleccionan los artefactos a reutilizar? **se evalúa la funcionalidad a realizar y se compara con las realizadas, si es similar se reutiliza**
    - ¿Considera que se demora menos tiempo reutilizando? Justifique. **Sí, solo hay que preocuparse por adaptar los aertefactos al sistema**
10. Trabajaría con Genexus**. SI.** Justifique: **Es muy poderosa y acelera mucho los tiempos para finalizar un proyecto. No hay que preocuparse por codificar mucho**

11. Otros comentarios: **No contesta**

Muchas gracias por su colaboración.

9.2.3 Encuesta 3

Fecha de realización de la encuesta: Junio 2015

DATOS LABORALES

1. Trabaja además de estudiar: **NO.**

DATOS DESARROLLO BASADO EN CONOCIMIENTO

1. Experiencia previa a la asignatura desarrollando con Genexus: **Nula**
2. Si tiene experiencia previa indique **tiempo** de uso de Genexus: ……..
3. Cantidad de proyectos desarrollados con Genexus: ……..……...

FORMA DE TRABAJO DURANTE LA EXPERIENCIA

1. Se trabajó compartiendo el lugar de trabajo**: A VECES**.
2. Cómo se suplió la falta de lugar de trabajo común: **encuentros, reuniones**
3. Considera importante compartir el lugar de trabajo: **SI**. Justifique**: debido a dudas que surgieron en el proyecto, la comunicación persona a persona hace que se entiendan las cosas de un mejor modo**
4. Se trabajó compartiendo los horarios dedicados a la experiencia: **No contesta**.
5. Considera importante compartir el horario de trabajo: **SI**. Justifique: **para poder salvar algunas dudas, errores que surgen al realizar una tareas que es influenciada por otra tarea no desarrollada por uno.**
6. Cuáles fueron los estándares y lineamientos definidos dentro de su equipo. **Las prioridades fueron definidas por el analista**
7. Cuales fueron las actividades a realizar definidas en el tablero de tareas. Explique el DoD de cada una.

   Análisis: el analista inicia el proceso de definir las tareas a realizar

   Desarrollo: el equipo elige tareas a desarrollar

   Prueba: se pone a prueba todas las tareas realizadas

   Producción: tareas realizadas, probadas e instaladas
8. La definición de DoD fue **SIMPLE**.
9. La definición de DoD considera que fue importante. **CIERTA IMPORTANCIA**
10. La forma de trabajo fue adaptándose durante el avance de la experiencia. **No contesta**
11. Considera importante tener un equipo de desarrollo autoorganizado: **No contesta.**
12. Se realizaron todos los tipos de reuniones especificadas. **SI**.
13. Indique la reunión que considera más importante justificando la respuesta**: Reunión de definición de visión, para determina bien que es el negocio y lo que necesita el sistema.**
14. Considera que alguna de las reuniones no es necesaria, **NO**. Justifique: **No contesta**
15. Se comprendieron los roles involucrados en la experiencia: **SI**
16. Indique el o los ROL/ES que desempeñó durante la experiencia: **Tester, desarrollador.**

17. Nombre las tres dificultades mayores enfrentadas indicando para cada uno el nivel del impacto:

- **Genexus usando GXServer: ALTO**
- **No tener lugar de trabajo común:BAJO**
- **No contesta**

OPINION SOBRE EL INICIO AGIL

1. Considera necesaria la participación de todos los participantes en la primera reunión. **MUY UTIL**. Justifique: **ayuda a todo el equipo a conocer por parte del cliente el negocio y ayuda al analista a entender mejor lo que se quiere.**
2. La definición de la visión siguiendo la práctica sugerida fue **POCO COMPLEJA**
3. En caso de no haberse realizado, el ejercicio de la visión, considera que igualmente todos compartirían la misma visión. **NO**
4. Siguiendo la técnica de identificación de los vecinos, determinar los stakeholders resultó: **FACIL**
5. Si usted ejerció el rol de coordinador y participó en la reunión de planificación inicial, considera que: **NO EJERCIO.**

OPINION SOBRE LA PRODUCCION

1. Los cambios en requerimientos o nuevos requerimientos fueron aceptados por el equipo: **SI**

2. Los objetos a modificar para implementar los cambios en requerimientos fue **SIMPLE.** Amplíe: **El analista logra entender bien los cambios y expresa de manera concreta al grupo**
3. Se realizó un desarrollo iterativo: **No contesta.**
4. Se realizó un diseño utilizando diagramas UML**: SI, se utilizó DER o Diagrama de clases**.
5. Para realizar las pruebas, se trabajó con algún Framework específico. **NO.** Especifique cual utilizó o bien porqué no utilizó**: no fue necesario**
6. Explique cómo garantizó que los cambios no propagaran errores en otras secciones que ya funcionaban bien**: pruebas antes de utilizar algunos objetos, realizar backups antes de realizar alguna tarea compleja**
7. Se fijaron WIP (límites de tareas por actividad) en la pizarra de tareas. **NO**. Justifique. **No justifica.**

8. Se realizaron entregas parciales de producto desarrollado al cliente. **SI**. Especifique el tiempo **Una semana y media.**

OPINION SOBRE EL RITUAL DE FINALIZACIÓN

1. En la reunión de entrega final, indique para cada sistema si considera que es necesario, y que beneficios se obtiene de esta reunión. **SI, muy necesario para saber el nivel de satisfacción del cliente, para recibir información de nuestro cliente sobre nuestro desempeño**

2. De la reunión de reflexión final de cada sistema, especifique qué opiniones surgieron. **Algunas decepciones con respecto a Genexus, algunas satisfacciones respecto a la herramienta. Pero muy conformes respecto al trabajo en equipo**

3. Considera que hubiera sigo más productivo tener a los stakeholders de los sistemas en la reunión de revisión final. **SI.** Explique: **para evaluar el nivel de satisfacción del cliente**

OPINION GENERLA SOBRE LA EXPERIENCIA

1. Al inicio del proyecto consideraba posible la realización de la experiencia. **NO.** Justifique su respuesta: **poco tiempo, materias cursando, aplicación de una**

metodología. Debido a ya tener un conocimiento previo de GX y ágiles no resultó ser tan complejo

2. Según la experiencia realizada, esta: **CONFORME** con los resultados obtenidos. Justifique: **al ser ágiles una metodología muy usada considero importante haber tenido esta experiencia, con respecto a la aplicación creo que faltó más tiempo.**
3. Se tuvo colaboración permanente con el cliente del gimnasio. **SI**. Explique como: **mediante consultas y predisposición a responder emails y foro**
4. Se tuvo colaboración permanente con el cliente de la librería. **SI**. Explique como: **buena predisposición a responder dudas mediante diversos medios como por ejemplo email**
5. Fue necesaria mayor participación de alguno de los clientes aparte de las reuniones establecidas: **NO**.
6. Especifique las herramientas utilizadas y los beneficios de cada una
    - **Genexus: hubo complicaciones debido a ser una versión trial pero es una herramienta muy potente para realizar sistemas de forma rápida**
    - **Kanbanize: costó un poco entenderlo pero fue muy útil para aplicarla metodología ágil**
7. Especifique herramientas utilizadas para el seguimiento del proyecto y su opinión sobre la misma: **Kanbanize, costó un poco entenderlo pero fue muy útil para aplicarla metodología ágil**
8. Quién se encargaba de actualizar el tablero: **al principio analista, luego todos.**
9. Se desarrolló con reutilización**. SI**.
    - Si es afirmativo ¿Cómo se seleccionan los artefactos a reutilizar?

    en reuniones se proponía utilizar algunos objetos ya creados

    - ¿Considera que se demora menos tiempo reutilizando? Justifique

    Sí, se ahorra ese tiempo aunque en nuestro caso no fue tanto tiempo
10. Trabajaría con Genexus**. SI.** Justifique: **es muy potente con la sola instalación ya crea o instala todo lo necesario, los ABM ya se definen**
11. Otros comentarios: **respecto a genexus hay muchas limitaciones al usar la versión free**

Muchas gracias por su colaboración.

9.2.4 Encuesta 4

Fecha de realización de la encuesta: Junio 2015

DATOS LABORALES

1. Trabaja además de estudiar: **NO.**

DATOS DESARROLLO BASADO EN CONOCIMIENTO

4. Experiencia previa a la asignatura desarrollando con Genexus: **Nula**
5. Si tiene experiencia previa indique **tiempo** de uso de Genexus: ……..
6. Cantidad de proyectos desarrollados con Genexus:4 ..……....

FORMA DE TRABAJO DURANTE LA EXPERIENCIA

1. Se trabajó compartiendo el lugar de trabajo**: A VECES**.

2. Cómo se suplió la falta de lugar de trabajo común: **mensajería instantánea, conectados por Internet y GXServer.**
3. Considera importante compartir el lugar de trabajo: **SI**. Justifique**: comodidad y capacidad o facilidad de expresar y compartir ideas y conocimiento**
4. Se trabajó compartiendo los horarios dedicados a la experiencia: **A VECES**.
5. Considera importante compartir el horario de trabajo: **SI**. Justifique: **comodidad y capacidad o facilidad de expresar y compartir ideas y conocimiento.**
6. Cuáles fueron los estándares y lineamientos definidos dentro de su equipo. **división de tareas y explicación de lo realizado al respecto**
7. Cuales fueron las actividades a realizar definidas en el tablero de tareas. Explique el DoD de cada una.

   No recuerdo los nombres
8. La definición de DoD fue **SIMPLE**.
9. La definición de DoD considera que fue importante. **SIN IMPORTANCIA**
10. La forma de trabajo fue adaptándose durante el avance de la experiencia **SI**. En caso de respuesta afirmativa. Ejemplifique**: la mayoría podía conectarse a la noche así que también comencé a hacer a la noche para coincidir el horario. Fallas de GXSever nos llevaron a usar facebook para exportar e importar el proyecto**
11. Considera importante tener un equipo de desarrollo autoorganizado: **SI.**
12. Se realizaron todos los tipos de reuniones especificadas. **SI**.
13. Indique la reunión que considera más importante justificando la respuesta**: Reunión de Definición de Visión, la más importante es la reunión con los clientes, y la primer reunión del equipo de desarrollo ya que se definen los roles y tareas de cada integrante y el modelo base del sistema.**
14. Considera que alguna de las reuniones no es necesaria, **NO**. Justifique: **siempre son necesarias para definir el avance y funcionamiento del sistema mover tareas, etc.**
15. Se comprendieron los roles involucrados en la experiencia: **SI**
16. Indique el o los ROL/ES que desempeñó durante la experiencia: **todos.**
17. Nombre las tres dificultades mayores enfrentadas indicando para cada uno el nivel del impacto:
    - **dificultad para compartir los objetos de la KB Alto**
    - **Dificultad para coordinar el grupo y tiempo ALTO**
    - **Dificultad para manejo de Genexus BAJO**

OPINION SOBRE EL INICIO AGIL

1. Considera necesaria la participación de todos los participantes en la primera reunión. **MUY UTIL**. Justifique: **todos deben conocer el sistema a desarrollar, aportara ello y aportar diferentes puntos de vista**
2. La definición de la visión siguiendo la práctica sugerida fue **POCO COMPLEJA**
3. En caso de no haberse realizado, el ejercicio de la visión, considera que igualmente todos compartirían la misma visión. **SI**
4. Siguiendo la técnica de identificación de los vecinos, determinar los stakeholders resultó: **FACIL**
5. Si usted ejerció el rol de coordinador y participó en la reunión de planificación inicial, considera que: **NO CONTESTA.**

OPINION SOBRE LA PRODUCCION

1. Los cambios en requerimientos o nuevos requerimientos fueron aceptados por el equipo: **SI**
2. Los objetos a modificar para implementar los cambios en requerimientos fue **MEDIO.** Amplíe: **para implementar que los clientes creen cuentas de usuario se debió conocer el GAM**
3. Se realizó un desarrollo iterativo: **NO**.
4. Se realizó un diseño utilizando diagramas UML**: NO**.
5. Para realizar las pruebas, se trabajó con algún Framework específico. **NO.** Especifique cual utilizó o bien porqué no utilizó**: se uso el definido por el SO ya que en ningún momento tuvimos problema**
6. Explique cómo garantizó que los cambios no propagaran errores en otras secciones que ya funcionaban bien**: con pruebas y Genexus ayuda tirando mensajes de error**
7. Se fijaron WIP (límites de tareas por actividad) en la pizarra de tareas. **NO**. Justifique. **No justifica.**
8. Se realizaron entregas parciales de producto desarrollado al cliente. **SI**. Especifique el tiempo **una semana**

OPINION SOBRE EL RITUAL DE FINALIZACIÓN

1. En la reunión de entrega final, indique para cada sistema si considera que es necesario, y que beneficios se obtiene de esta reunión. **SI, aprobación o no por parte del cliente, permite conocer el grado de satisfacción del cliente y posibles mejoras**
2. De la reunión de reflexión final de cada sistema, especifique qué opiniones surgieron. **Aprobación por parte del cliente y algunas fallas o arreglos para el sistema librería / excelente para sistema Gimnasio**
3. Considera que hubiera sigo más productivo tener a los stakeholders de los sistemas en la reunión de revisión final. **NO.** Explique: **no creo que influya su opinión en problemas del grupo**

OPINION GENERLA SOBRE LA EXPERIENCIA

1. Al inicio del proyecto consideraba posible la realización de la experiencia. **NO.** Justifique su respuesta**: bajo conocimiento de Genexus y de metodología ágil y tiempo**
2. Según la experiencia realizada, esta: **CONFORME** con los resultados obtenidos. Justifique: **se me complicó con los tiempos y otras materias pero Genexus es una herramienta que me llama la atención y causo mucho gusto**
3. Se tuvo colaboración permanente con el cliente del gimnasio. **SI**. Explique como: **email.**
4. Se tuvo colaboración permanente con el cliente de la librería. **NO**. Explique como: **solo reuniones pactadas.**
5. Fue necesaria mayor participación de alguno de los clientes aparte de las reuniones establecidas: **NO**.
6. Especifique las herramientas utilizadas y los beneficios de cada una
   - **Genexus para desarrollo del sistema**
   - **Kanbanize para desarrollo de la metodología ágil**
   - **GXServer compartir KB**
7. Especifique herramientas utilizadas para el seguimiento del proyecto y su opinión sobre la misma: **no menciona.**
8. Quién se encargaba de actualizar el tablero: **coordinador**
9. Se desarrolló con reutilización**. SI**.

- Si es afirmativo ¿Cómo se seleccionan los artefactos a reutilizar? **exportando e importando objetos**
- ¿Considera que se demora menos tiempo reutilizando? Justifique: **No, es más rápido solo en algunos casos, considero mas rápido desarrollar desde cero que modificar propiedades y atributos.**

10. Trabajaría con Genexus**. SI.** Justifique: **herramienta potente y completa que facilita el desarrollo de sw, por gusto personal y por tener la posibilidad de conocer GX full**
11. Otros comentarios: **un curso muy útil e interesante que nos perite conocer nueva modalidad de desarrollo de sw y nos brinda la posibilidad de conocer todo el entorno de Gx. Brinda los conocimientos para una posible certificación GX**

Muchas gracias por su colaboración.

9.2.5 Encuesta 5

Fecha de realización de la encuesta: Junio 2015

DATOS LABORALES

1. Trabaja además de estudiar: **SI.**
2. Trabaja en organismo **PRIVADO**.
3. Cantidad de hs semanales que trabaja: **20hs**
4. Tipo de trabajo que realiza: **soporte informático**

DATOS DESARROLLO BASADO EN CONOCIMIENTO

1. Experiencia previa a la asignatura desarrollando con Genexus: **Mínima**
2. Si tiene experiencia previa indique **tiempo** de uso de Genexus: **2 semanas**
3. Cantidad de proyectos desarrollados con Genexus: **0**

FORMA DE TRABAJO DURANTE LA EXPERIENCIA

1. Se trabajó compartiendo el lugar de trabajo**: A VECES**.
2. Cómo se suplió la falta de lugar de trabajo común: **no contesta**
3. Considera importante compartir el lugar de trabajo: **SI**. Justifique**: ya que se puede interactuar de manera directa con el grupo**
4. Se trabajó compartiendo los horarios dedicados a la experiencia: **SI**.
5. Considera importante compartir el horario de trabajo: **SI**. Justifique: **así nos organizamos mejor.**
6. Cuáles fueron los estándares y lineamientos definidos dentro de su equipo. **las tareas fueron designadas por el coordinador y cuales miembros del equipo la podía desarrollar**
7. Cuáles fueron las actividades a realizar definidas en el tablero de tareas. Explique el DoD de cada una.

   Análisis: se definen los procesos a realizar

   Desarrollo: tareas en desarrollo

   Prueba: testeo de los módulos que se terminaron de desarrollar

   Terminado: instalado en servidor del cliente
8. La definición de DoD fue **MEDIO**.
9. La definición de DoD considera que fue importante. **IMPORTANTE**

10. La forma de trabajo fue adaptándose durante el avance de la experiencia **NO**
11. Considera importante tener un equipo de desarrollo autoorganizado: **SI.**
12. Se realizaron todos los tipos de reuniones especificadas. **SI**.
13. Indique la reunión que considera más importante justificando la respuesta**: Reunión de evaluación externa, la más importante es con el cliente ya que de esa forma puede ir puliéndose el sistema.**
14. Considera que alguna de las reuniones no es necesaria, **NO**. Justifique: **todas tienen algo en particular que sirven a la retroalimentación del grupo.**
15. Se comprendieron los roles involucrados en la experiencia: **SI**
16. Indique el o los ROL/ES que desempeñó durante la experiencia: **desarrollador.**
17. Nombre las tres dificultades mayores enfrentadas indicando para cada uno el nivel del impacto:
    - **Limitación de Genexus BAJO**
    - **Problemas con GXServer MODERADO**
    - **No contesta**

OPINION SOBRE EL INICIO AGIL

1. Considera necesaria la participación de todos los participantes en la primera reunión. **MUY UTIL**. Justifique: **es distinto si solo participa uno solo del grupo porque se puede perder algún detalle de lo que quiere el cliente**
2. La definición de la visión siguiendo la práctica sugerida fue **POCO COMPLEJA**
3. En caso de no haberse realizado, el ejercicio de la visión, considera que igualmente todos compartirían la misma visión. **NO CONTESTA**
4. Siguiendo la técnica de identificación de los vecinos, determinar los stakeholders resultó: **FACIL**
5. Si usted ejerció el rol de coordinador y participó en la reunión de planificación inicial, considera que: **NO CONTESTA.**

OPINION SOBRE LA PRODUCCION

1. Los cambios en requerimientos o nuevos requerimientos fueron aceptados por el equipo: **SI**
2. Los objetos a modificar para implementar los cambios en requerimientos fue **SIMPLE.** Amplíe: **solo eran detalles.**
3. Se realizó un desarrollo iterativo: **No contesta.**
4. Se realizó un diseño utilizando diagramas UML**: NO**.
5. Para realizar las pruebas, se trabajó con algún Framework específico. **NO.** Especifique cual utilizó o bien por qué no utilizó**: no se creía necesario.**
6. Explique cómo garantizó que los cambios no propagaran errores en otras secciones que ya funcionaban bien**: probando todas las partes del módulo y con los que estaban relacionados**
7. Se fijaron WIP (límites de tareas por actividad) en la pizarra de tareas. **NO**. Justifique. **no hizo falta.**
8. Se realizaron entregas parciales de producto desarrollado al cliente. **SI**. Especifique el tiempo **cada do semanas**

OPINION SOBRE EL RITUAL DE FINALIZACIÓN

1. En la reunión de entrega final, indique para cada sistema si considera que es necesario, y que beneficios se obtiene de esta reunión. **SI, es la entrega o se muestra el producto final.**

2. De la reunión de reflexión final de cada sistema, especifique qué opiniones surgieron. **Cuando terminamos el de la librería, teníamos que seguir con el otro sistema, el punto central fue la reutilización de objetos para la construcción del sistema del Gimnasio**
3. Considera que hubiera sigo más productivo tener a los stakeholders de los sistemas en la reunión de revisión final. **SI.** Explique: **conocer si todos están satisfechos es mejor**

OPINION GENERLA SOBRE LA EXPERIENCIA

1. Al inicio del proyecto consideraba posible la realización de la experiencia. **SI.** Justifique su respuesta: **no se veía que fuera imposible o muy complejo**
2. Según la experiencia realizada, esta: **MUY CONFORME** con los resultados obtenidos. Justifique: **aprendí un uso más comprensivo de la herramienta**
3. Se tuvo colaboración permanente con el cliente del gimnasio. **SI**. Explique como: **ante las dudas que surgían, en las reuniones con el cliente se redujeron**
4. Se tuvo colaboración permanente con el cliente de la librería. **SI**. Explique como: **ídem gimnasio**
5. Fue necesaria mayor participación de alguno de los clientes aparte de las reuniones establecidas: **NO**.
6. Especifique las herramientas utilizadas y los beneficios de cada una
    - **Genexus desarrollo del sistema en tiempo corto**
7. Especifique herramientas utilizadas para el seguimiento del proyecto y su opinión sobre la misma: **Kanbanize, se podía visualizar como avanzaba el desarrollo del sistema**
8. Quién se encargaba de actualizar el tablero: **todos**
9. Se desarrolló con reutilización**. SI**.
    - Si es afirmativo ¿Cómo se seleccionan los artefactos a reutilizar? **se encuentran las cosas en común entre los sistemas a desarrollar**
    - ¿Considera que se demora menos tiempo reutilizando? Justifique. **Sí, dado que no hay que tipear o generar todo de nuevo**
10. Trabajaría con Genexus**. SI.** Justifique: **herramienta muy potente sobre todo se ahorra tiempo y o hace falta conocimiento de muchos lenguajes de programación**
11. Otros comentarios: **versión de gx limitada - gxserver trajo muchos problemas.**

Muchas gracias por su colaboración.

9.2.6Encuesta 6

Fecha de realización de la encuesta: Junio 2015

DATOS LABORALES

1. Trabaja además de estudiar: **NO.**

DATOS DESARROLLO BASADO EN CONOCIMIENTO

1. Experiencia previa a la asignatura desarrollando con Genexus: **Nula**
2. Si tiene experiencia previa indique **tiempo** de uso de Genexus: ……..
3. Cantidad de proyectos desarrollados con Genexus: ……...

FORMA DE TRABAJO DURANTE LA EXPERIENCIA

1. Se trabajó compartiendo el lugar de trabajo**: A VECES**.
2. Cómo se suplió la falta de lugar de trabajo común: **conectados por internet**

3. Considera importante compartir el lugar de trabajo: **SI**. Justifique**: se llega a conclusiones más rápidas y se avanza mejor**
4. Se trabajó compartiendo los horarios dedicados a la experiencia: **A VECES**.
5. Considera importante compartir el horario de trabajo: **SI**. Justifique: **es más rápido despejar dudas.**
6. Cuáles fueron los estándares y lineamientos definidos dentro de su equipo. T**rabajar solidariamente para evitar que el proyecto se retrase. Otro la forma que se realizaban los avances teneido en cuenta que se trabajabala mayor parte del tiempo a distancia**
7. Cuales fueron las actividades a realizar definidas en el tablero de tareas. Explique el DoD de cada una.

   Utilizamos un tablero simple de tres columnas, las cuales eran las tareas por hacer, las que estaban en proceso y las finalizadas (instaladas en servidor del cliente). Luego se iba actualizando
8. La definición de DoD fue **SIMPLE**.
9. La definición de DoD considera que fue importante. **CIERTA IMPORTANCIA**
10. La forma de trabajo fue adaptándose durante el avance de la experiencia **SI**. En caso de respuesta afirmativa. Ejemplifique**: algunos del grupo teníamos horarios distintos por cursar materias lo que hacia más difícil sincronizar horarios y casi todo se hizo a distancia**
11. Considera importante tener un equipo de desarrollo autoorganizado: **SI.**
12. Se realizaron todos los tipos de reuniones especificadas. **SI**. En caso de no haber realizado alguna por favor justifique porqué no se realizó: **situaciones imprevistas que obligaron a hacer las reuniones vía Internet en otro horario.**
13. Indique la reunión que considera más importante justificando la respuesta**: Reunión de planificación inicial, donde se especifica cuáles van a ser las historias de usuario y su importancia dentro del sistema.**
14. Considera que alguna de las reuniones no es necesaria, **NO**. Justifique: **todas son importantes, es necesario ir llevando de cerca el avance del equipo en el proyecto y ver también las dificultades que surgieron, o surgirán.**
15. Se comprendieron los roles involucrados en la experiencia: **SI**
16. Indique el o los ROL/ES que desempeñó durante la experiencia: **desarrollador.**
17. Nombre las tres dificultades mayores enfrentadas indicando para cada uno el nivel del impacto:
    - **dificultad para trabajar en mismo lugar y horario MODERADO**
    - **inexperiencia en Genexus BAJO**
    - **trabajar en una versión TRIAL con limitaciones MODERADO**

OPINION SOBRE EL INICIO AGIL

1. Considera necesaria la participación de todos los participantes en la primera reunión. **MUY UTIL**. Justifique: **tener claro cual es el objetivo del proyecto y sus prioridades**
2. La definición de la visión siguiendo la práctica sugerida fue **SIMPLE**
3. En caso de no haberse realizado, el ejercicio de la visión, considera que igualmente todos compartirían la misma visión. **NO**
4. Siguiendo la técnica de identificación de los vecinos, determinar los stakeholders resultó: **FACIL**
5. Si usted ejerció el rol de coordinador y participó en la reunión de planificación inicial, considera que: **NO CONTESTA**

OPINION SOBRE LA PRODUCCION

1. Los cambios en requerimientos o nuevos requerimientos fueron aceptados por el equipo: **SI**
2. Los objetos a modificar para implementar los cambios en requerimientos fue **MEDIO.** Amplíe: **no variaban mucho**
3. Se realizó un desarrollo iterativo: **SI**. Indique duración de la iteración: **semanas**
4. Se realizó un diseño utilizando diagramas UML**: NO**.
5. Para realizar las pruebas, se trabajó con algún Framework específico. **NO.** Especifique cual utilizó o bien porqué no utilizó**: proyectos no muy complejos en sus funcionalidades**
6. Explique cómo garantizó que los cambios no propagaran errores en otras secciones que ya funcionaban bien**: realizando pruebas locales y luego de asegurarse recién subirlo para que los demás lo actualicen**
7. Se fijaron WIP (límites de tareas por actividad) en la pizarra de tareas. **NO**. Justifique. h**abía un acuerdo interno en cuanto a las cantidades límites.**
8. Se realizaron entregas parciales de producto desarrollado al cliente. **SI**. Especifique el tiempo **cada semana, durante la misma se ostraban avances**

OPINION SOBRE EL RITUAL DE FINALIZACIÓN

1. En la reunión de entrega final, indique para cada sistema si considera que es necesario, y que beneficios se obtiene de esta reunión. **SI, sirve para ver si se llegó a lo que se esperaba y que cosas las cosas de relevancia a la hora del desarrollo como beneficio se obtiene que se puede hacer una retroalimentación hacia el grupo para aclarar cuestiones a futuro**
2. De la reunión de reflexión final de cada sistema, especifique qué opiniones surgieron. **No contesta**
3. Considera que hubiera sigo más productivo tener a los stakeholders de los sistemas en la reunión de revisión final. **NO.** Explique: **da lo mismo.**

OPINION GENERLA SOBRE LA EXPERIENCIA

1. Al inicio del proyecto consideraba posible la realización de la experiencia. **NO.** Justifique su respuesta: **parecía bastante trabajo pero, a medida que se iba desarrollando, no parecía tan complejo y se logró terminar a tiempo**
2. Según la experiencia realizada, esta: **CONFORME** con los resultados obtenidos. Justifique: **se aprendió más respecto a desarrollo ágil y hubo que aprender por cuenta propia algunas cuestiones sobre Genexus**
3. Se tuvo colaboración permanente con el cliente del gimnasio. **SI**. Explique como: **en cada entrega que se hacía salían ajustes para realizar**
4. Se tuvo colaboración permanente con el cliente de la librería. **SI**. Explique como: **en cada entrega se hacían sugerencias para llegar al objetivo de la mejor manera**
5. Fue necesaria mayor participación de alguno de los clientes aparte de las reuniones establecidas: **NO**.
6. Especifique las herramientas utilizadas y los beneficios de cada una
    - **Genexus para desarrollo**
    - **Kanbanize organización**
7. Especifique herramientas utilizadas para el seguimiento del proyecto y su opinión sobre la misma: **Kanbanize, bastante buena porque es clara y primera vista del panorama actual del proyecto**
8. Quién se encargaba de actualizar el tablero: **todos**
9. Se desarrolló con reutilización**. SI**.

- Si es afirmativo ¿Cómo se seleccionan los artefactos a reutilizar? **se puede descargarlos y aplicarlos al proyecto**
- ¿Considera que se demora menos tiempo reutilizando? Justifique: **se ahorran horas de desarrollo simplemente modificando un poco**

10. Trabajaría con Genexus**. SI.** Justifique: **muy potente para el desarrollo, no es muy compleja en aprender a utilizar por lo menos lo básico y tiene mucho soporte en línea para consultar**
11. Otros comentarios: **Gx es muy buena herramienta pero la versión trial es limitada en algunas cuestiones del desarrollo.**

Muchas gracias por su colaboración.

9.2.7 Encuesta 7

Fecha de realización de la encuesta: Junio 2015

DATOS LABORALES

1. Trabaja además de estudiar: **NO.**

DATOS DESARROLLO BASADO EN CONOCIMIENTO

1. Experiencia previa a la asignatura desarrollando con Genexus: **Mínima**
2. Si tiene experiencia previa indique **tiempo** de uso de Genexus: **doce meses**
3. Cantidad de proyectos desarrollados con Genexus: **3** ……...

FORMA DE TRABAJO DURANTE LA EXPERIENCIA

1. Se trabajó compartiendo el lugar de trabajo**: A VECES**.
2. Cómo se suplió la falta de lugar de trabajo común: **e-mail, mensajería instantánea**, **gxserver**.
3. Considera importante compartir el lugar de trabajo: **No contesta**. Justifique**: mayor comunicación y entendimiento del proyecto, además de que los problemas se ponen en común y muchas veces se solucionan más rápido**
4. Se trabajó compartiendo los horarios dedicados a la experiencia: **SI**.
5. Considera importante compartir el horario de trabajo: **SI**. Justifique: **ayuda a la organización general del grupo.**
6. Cuáles fueron los estándares y lineamientos definidos dentro de su equipo. **Concentrarse 100% en los momentos en los que el equipo estaba reunido**
7. Cuales fueron las actividades a realizar definidas en el tablero de tareas. Explique el DoD de cada una.

   Las tareas fueron análisis, desarrollo, testing y producción. El DoD del análisis era tener la historia de usuario definida y especificado el objeto a reutilizar si era el caso (reutilización).
8. La definición de DoD fue **SIMPLE**.
9. La definición de DoD considera que fue importante. **IMPORTANTE**
10. La forma de trabajo fue adaptándose durante el avance de la experiencia **NO**
11. Considera importante tener un equipo de desarrollo autoorganizado: **SI.**
12. Se realizaron todos los tipos de reuniones especificadas. **NO**. En caso de no haber realizado alguna por favor justifique porqué no se realizó: **falta de tiempo y organización de horarios de integrantes**
13. Indique la reunión que considera más importante justificando la respuesta**: Reunión de Seguimiento, ayuda a plantear de forma común los problemas a solucionarlos**

más rápido y saber en qué está trabajando el resto del equipo aunque no sea de manera presencial.

14. Considera que alguna de las reuniones no es necesaria, **NO** Justifique: **todas necesarias.**

15. Se comprendieron los roles involucrados en la experiencia: **SI**

16. Indique el o los ROL/ES que desempeñó durante la experiencia: **Analista, desarrollador.**

17. Nombre las tres dificultades mayores enfrentadas indicando para cada uno el nivel del impacto:

- **documentación de cada reunión y del avance MUY ALTO**
- **dificultad para encontrar información de ciertos objetos Genexus MOPDERADO**
- **Dificultad para hacer funcionar GXServer MUYALTO**

OPINION SOBRE EL INICIO AGIL

1. Considera necesaria la participación de todos los participantes en la primera reunión. **ALGO UTIL**. Justifique: **si los analistas definen luego todas las funciones, lo que se perciba en la primera reunión no aporta mucho**
2. La definición de la visión siguiendo la práctica sugerida fue **COMPLEJA**
3. En caso de no haberse realizado, el ejercicio de la visión, considera que igualmente todos compartirían la misma visión. **NO**
4. Siguiendo la técnica de identificación de los vecinos, determinar los stakeholders resultó: **POCO COMPLEJO**
5. Si usted ejerció el rol de coordinador y participó en la reunión de planificación inicial, considera que:
   - Las tareas propuestas son: **NO CONTESTA**.
   - Como fue la participación del cliente del gimnasio**: MUY ACTIVA.**
   - Como fue la participación del cliente de la librería: **MUY ACTIVA.**

OPINION SOBRE LA PRODUCCION

1. Los cambios en requerimientos o nuevos requerimientos fueron aceptados por el equipo: **SI**

2. Los objetos a modificar para implementar los cambios en requerimientos fue **SIMPLE.** Amplíe: **Gx se adapta fácilmente a la metodología ágil, implementar los cambios resultó fácil**
3. Se realizó un desarrollo iterativo: **SI**. Indique duración de la iteración: **no fue tajante, aprox 1 semana.**
4. Se realizó un diseño utilizando diagramas UML**: NO**.
5. Para realizar las pruebas, se trabajó con algún Framework específico. **NO.** Especifique cual utilizó o bien porqué no utilizó**: para el alcance del sistema no era necesario usar framework**
6. Explique cómo garantizó que los cambios no propagaran errores en otras secciones que ya funcionaban bien**: cuando compilábamos si no arrojaba error continuábamos**
7. Se fijaron WIP (límites de tareas por actividad) en la pizarra de tareas. **NO**. Justifique. **no se consideró necesario.**

8. Se realizaron entregas parciales de producto desarrollado al cliente. **SI**. Especifique el tiempo **2 semanas**

OPINION SOBRE EL RITUAL DE FINALIZACIÓN

1. En la reunión de entrega final, indique para cada sistema si considera que es necesario, y que beneficios se obtiene de esta reunión. **No contesta**
2. De la reunión de reflexión final de cada sistema, especifique qué opiniones surgieron. **No contesta**
3. Considera que hubiera sigo más productivo tener a los stakeholders de los sistemas en la reunión de revisión final. **SI.** Explique: **cada uno tendría una visión distinta del producto final**

OPINION GENERLA SOBRE LA EXPERIENCIA

1. Al inicio del proyecto consideraba posible la realización de la experiencia. **NO.** Justifique su respuesta: **poco tiempo para lo que se pedía.**
2. Según la experiencia realizada, esta: **CONFORME** con los resultados obtenidos. Justifique: **los sistemas funcionan bien y sirvieron para tener cierta práctica real de la metodología**
3. Se tuvo colaboración permanente con el cliente del gimnasio. **SI**. Explique como: **email, consultas personales**
4. Se tuvo colaboración permanente con el cliente de la librería. **SI**. Explique como: **email, consultas personales**
5. Fue necesaria mayor participación de alguno de los clientes aparte de las reuniones establecidas: **NO**.
6. Especifique las herramientas utilizadas y los beneficios de cada una
   - **Genexus desarrollo y adaptable a la metodología**
   - **Kanbanize metodología ágil**
7. Especifique herramientas utilizadas para el seguimiento del proyecto y su opinión sobre la misma: **Kanbanize, sirve bastante para repartir tareas y organizar mejor el trabajo, establece roles claros**
8. Quién se encargaba de actualizar el tablero: **todos y analista**
9. Se desarrolló con reutilización**. SI**.
   - Si es afirmativo ¿Cómo se seleccionan los artefactos a reutilizar? **se observa que objetos son comunes con el sistema y se los importa**
   - ¿Considera que se demora menos tiempo reutilizando? Justifique: **Sí, no se pierde tiempo desarrollando cosas que ya se hicieron**
10. Trabajaría con Genexus**. no contesta**
11. Otros comentarios: **no contesta**

Muchas gracias por su colaboración.

9.2.8Encuesta 8

Fecha de realización de la encuesta: Junio 2015

DATOS LABORALES

1. Trabaja además de estudiar: **SI.**
2. Trabaja en organismo **PRIVADO**.
3. Cantidad de hs semanales que trabaja: **20hs**……...
4. Tipo de trabajo que realiza: **programador genexus**……....
5. Si realiza desarrollo de sw,

a. Especificar metodología de desarrollo que utiliza en el trabajo: **SCRUM, KANBAN**

b. Varios proyectos realizados al mismo tiempo: **FRECUENTEMENTE**

c. Comparte horario de trabajo con el resto del equipo de trabajo: **SI**

d. Las personas involucradas comparten el lugar de trabajo: **SI**.

e. Existe un líder de proyecto: **SI**

DATOS DESARROLLO BASADO EN CONOCIMIENTO

1. Experiencia previa a la asignatura desarrollando con Genexus: **Nula**
2. Si tiene experiencia previa indique **tiempo** de uso de Genexus: ……..
3. Cantidad de proyectos desarrollados con Genexus: ……..……...

FORMA DE TRABAJO DURANTE LA EXPERIENCIA

1. Se trabajó compartiendo el lugar de trabajo**: A VECES**.
2. Cómo se suplió la falta de lugar de trabajo común: **conectados por internet**
3. Considera importante compartir el lugar de trabajo: **SI**. Justifique**: mas fácil despejar dudas con el equipo, plantear nueva ideas**
4. Se trabajó compartiendo los horarios dedicados a la experiencia: **A VECES**.
5. Considera importante compartir el horario de trabajo: **SI**. Justifique: **más fácil despejar dudas con el equipo, plantear nueva ideas.**
6. Cuáles fueron los estándares y lineamientos definidos dentro de su equipo. **No contesta**
7. Cuales fueron las actividades a realizar definidas en el tablero de tareas. Explique el DoD de cada una.

   Análisis, desarrollo, testing, producción
8. La definición de DoD fue **MEDIO**.
9. La definición de DoD considera que fue importante. **IMPORTANTE**
10. La forma de trabajo fue adaptándose durante el avance de la experiencia **SI**. En caso de respuesta afirmativa. Ejemplifique**: tuvimos complicaciones con la herramienta GXServer por lo que debimos adaptarnos a otra estrategia de desarrollo**
11. Considera importante tener un equipo de desarrollo autoorganizado: **SI.**
12. Se realizaron todos los tipos de reuniones especificadas. **SI**.
13. Indique la reunión que considera más importante justificando la respuesta**: Reunión de Evaluación externa, reunión con cliente al final del sprint porque permite conocer qué está bien y mal desarrollado y nuevas inquietudes del cliente.**
14. Considera que alguna de las reuniones no es necesaria, **SI**. Justifique: **reunión diaria todos los días, ya que al desarrollar uno ya va hablando con el equipo de las cosas que pasan**
15. Se comprendieron los roles involucrados en la experiencia: **SI**
16. Indique el o los ROL/ES que desempeñó durante la experiencia: **desarrollador.**
17. Nombre las tres dificultades mayores enfrentadas indicando para cada uno el nivel del impacto:
    - **GXServer Muy ALTO**
    - **Servidor en la nube: ALTO**
    - **Tiempo MODERADO**

OPINION SOBRE EL INICIO AGIL

1. Considera necesaria la participación de todos los participantes en la primera reunión. **ALGO UTIL**. Justifique: **ayuda al equipo a entender un poco el sistema objeto.**
2. La definición de la visión siguiendo la práctica sugerida fue **POCO COMPLEJA**
3. En caso de no haberse realizado, el ejercicio de la visión, considera que igualmente todos compartirían la misma visión. **NO**
4. Siguiendo la técnica de identificación de los vecinos, determinar los stakeholders resultó: **POCO COMPLEJA**
5. Si usted ejerció el rol de coordinador y participó en la reunión de planificación inicial, considera que: **No contesta.**

OPINION SOBRE LA PRODUCCION

1. Los cambios en requerimientos o nuevos requerimientos fueron aceptados por el equipo: **SI**
2. Los objetos a modificar para implementar los cambios en requerimientos fue: **No contesta**
3. Se realizó un desarrollo iterativo: **SI**. Indique duración de la iteración: **sin tiempo fijo**
4. Se realizó un diseño utilizando diagramas UML**: NO**.
5. Para realizar las pruebas, se trabajó con algún Framework específico. **NO.** Especifique cual utilizó o bien porqué no utilizó**: decidimos probar nosotros mismos cargando datos y analizando la situación**
6. Explique cómo garantizó que los cambios no propagaran errores en otras secciones que ya funcionaban bien: **a pesar de trabajar con objetos distintos no se puede garantizar que un cambio en alguno no propagará errores, ya que algunos se relacionan**
7. Se fijaron WIP (límites de tareas por actividad) en la pizarra de tareas. **SI**. Justifique. **ayudaron a repartir las tareas en el equipo.**
8. Se realizaron entregas parciales de producto desarrollado al cliente. **SI**. Especifique el tiempo **2 semanas**

OPINION SOBRE EL RITUAL DE FINALIZACIÓN

1. En la reunión de entrega final, indique para cada sistema si considera que es necesario, y que beneficios se obtiene de esta reunión. **SI, se expone al cliente el producto finalizado sirve de retrospectiva**
2. De la reunión de reflexión final de cada sistema, especifique qué opiniones surgieron. **para la librería hubo para corregir una pantalla y comparación de libros**
3. Considera que hubiera sigo más productivo tener a los stakeholders de los sistemas en la reunión de revisión final. **SI.** Explique: **hubieran surgido dudas que se podrían haber resuelto en menor tiempo.**

OPINION GENERLA SOBRE LA EXPERIENCIA

1. Al inicio del proyecto consideraba posible la realización de la experiencia. **NO.** Justifique su respuesta: **pensé que llevaba más tiempo desarrollar dossistemas en paralelo**
2. Según la experiencia realizada, esta: **CONFORME** con los resultados obtenidos. Justifique: **aprendí muchas cosas de genexus, sin esta experiencia creo que no hubiera sido lo mismo**
3. Se tuvo colaboración permanente con el cliente del gimnasio. **SI**. Explique como: **reuniones**
4. Se tuvo colaboración permanente con el cliente de la librería. **SI**. Explique como: **reuniones**

5. Fue necesaria mayor participación de alguno de los clientes aparte de las reuniones establecidas: **NO**.
6. Especifique las herramientas utilizadas y los beneficios de cada una
    - **Genexus permite obtener porción de producto funcionando en poco tiempo**
    - **Kanbanize permite organizar las tareas de desarrollo del equipo**
7. Especifique herramientas utilizadas para el seguimiento del proyecto y su opinión sobre la misma: **Kanbanize, permite organizar las tareas de desarrollo del equipo**
8. Quién se encargaba de actualizar el tablero: **analista**
9. Se desarrolló con reutilización**. SI**.
    - Si es afirmativo ¿Cómo se seleccionan los artefactos a reutilizar? **dependiendo de sus funcionalidades y compatibilidad con el sistema en desarrollo**
    - ¿Considera que se demora menos tiempo reutilizando? Justifique. **Sí, no hay que pensar la lógica del componente solo adatarlo**
10. Trabajaría con Genexus**. SI.** Justifique: **permite obtener rápida porción de producto funcional. Además no se debe programar tanto como con otros lenguajes**
11. Otros comentarios: **no contesta**

Muchas gracias por su colaboración.

9.2.9Encuesta 9

Fecha de realización de la encuesta: Junio 2015

DATOS LABORALES

1. Trabaja además de estudiar: **SI - NO.**
2. Trabaja en organismo **PRIVADO**.
3. Cantidad de hs semanales que trabaja: **35hs**……..……...
4. Tipo de trabajo que realiza: **programador y gerente proyecto familiar**.
5. Si realiza desarrollo de sw,
    a. Especificar metodología de desarrollo que utiliza en el trabajo: **Scrum, metodologías ágiles.**
    b. Varios proyectos realizados al mismo tiempo: **FRECUENTEMENTE**
    c. Comparte horario de trabajo con el resto del equipo de trabajo: **SI**
    d. Las personas involucradas comparten el lugar de trabajo: **SI**.
    e. Existe un líder de proyecto: **SI**

DATOS DESARROLLO BASADO EN CONOCIMIENTO

1. Experiencia previa a la asignatura desarrollando con Genexus: **Moderada**
2. Si tiene experiencia previa indique **tiempo** de uso de Genexus; **12 meses**
3. Cantidad de proyectos desarrollados con Genexus: **2** ……...

FORMA DE TRABAJO DURANTE LA EXPERIENCIA

1. Se trabajó compartiendo el lugar de trabajo**: A VECES**.
2. Cómo se suplió la falta de lugar de trabajo común: **internet**

3. Considera importante compartir el lugar de trabajo: **INDIFERENTE**. Justifique**: Para algunas cosas es necesario poder reunirse personalmente para poder sacar dudas mucho más rápido**
4. Se trabajó compartiendo los horarios dedicados a la experiencia: **A VECES**.
5. Considera importante compartir el horario de trabajo: **INDIFERENTE**. Justifique: **Existen tareas donde no es necesario que estén los compañeros de trabajo en el mismo horario, mientras que para algunas otras sí.**
6. Cuáles fueron los estándares y lineamientos definidos dentro de su equipo. **Cada uno realizar su trabajo, intentando solucionar cada uno sus problemas. Cumplimiento de sus tareas y tiempos programados**
7. Cuales fueron las actividades a realizar definidas en el tablero de tareas. Explique el DoD de cada una.

   Análisis, Estructuras de datos, diseño, altas bajas y modificaciones de transacciones, reportes, testings, producción.
8. La definición de DoD fue **SIMPLE**.
9. La definición de DoD considera que fue importante. **CIERTA IMPORTANCIA**
10. La forma de trabajo fue adaptándose durante el avance de la experiencia **SI**. En caso de respuesta afirmativa. Ejemplifique**: Cada uno sabia que debía entregar su tarea para que se realice el testing correspondiente, por lo que sabían que tenían un tiempo determinado para hacerlo. Si bien costó primeramente cumplir ese tiempo, se logró concretarlo**
11. Considera importante tener un equipo de desarrollo autoorganizado: **SI.**
12. Se realizaron todos los tipos de reuniones especificadas. **NO**. En caso de no haber realizado alguna por favor justifique porqué no se realizó: **La deserción de un integrante complicó al grupo. Se tuvo que realizar una reorganización, y las tareas a entregar fueron diferentes a las pactadas originalmente. Otra complicación fue la diferencia de tiempo de los desarrolladores**
13. Indique la reunión que considera más importante justificando la respuesta**: Reunión de contextualización, es la reunión de organización, análisis y estructura de datos. Porque al colaborar todos con el análisis, todos opinaban y se desarrollaba una única estructura que conforme a todos los integrantes de desarrollo.**
14. Considera que alguna de las reuniones no es necesaria, **NO**. Justifique: **Todas las reuniones fueron importantes para el grupo, ya que cada uno podía sacarse una inquietud, o consultar algunos errores que por ahí ya le habían pasado a otro integrante.**
15. Se comprendieron los roles involucrados en la experiencia: **SI**
16. Indique el o los ROL/ES que desempeñó durante la experiencia: **Analista, Coordinador, desarrollador.**
17. Nombre las tres dificultades mayores enfrentadas indicando para cada uno el nivel del impacto:
    - **Hacer que todos los desarrolladores trabajen por igual, es decir que nadie "haga más" que otro, ni viceversa. - ALTO**
    - **GX Server - El uso de gxserver generó mucha disconformidad con los desarrolladores, ya que fue más el tiempo empleando en intentar hacer andar que "desarrollando" - ALTO**
    - **Gestión de Usuarios - MODERADO**

OPINION SOBRE EL INICIO AGIL

1. Considera necesaria la participación de todos los participantes en la primera reunión. **MUY UTIL**. Justifique: **En nuestro caso fue muy importante, ya que se necesitaba**

la opinión de todos los integrantes para la definición de como encarar el proyecto

2. La definición de la visión siguiendo la práctica sugerida fue **POCO COMPLEJA**
3. En caso de no haberse realizado, el ejercicio de la visión, considera que igualmente todos compartirían la misma visión. **NO**
4. Siguiendo la técnica de identificación de los vecinos, determinar los stakeholders resultó: **FACIL**
5. Si usted ejerció el rol de coordinador y participó en la reunión de planificación inicial, considera que:
    - Las tareas propuestas son: **SIMPLE.** ACLARACIÓN**:** las tareas son simples, el problema son las limitaciones del trial
    - Como fue la participación del cliente del gimnasio**: ALGO ACTIVA.**
    - Como fue la participación del cliente de la librería: **ALGO ACTIVA.**

OPINION SOBRE LA PRODUCCION

1. Los cambios en requerimientos o nuevos requerimientos fueron aceptados por el equipo: **SI**

2. Los objetos a modificar para implementar los cambios en requerimientos fue **MEDIO.** Amplíe: **Algunos nuevos requerimientos consistían en algo más que una modificación del diseño, lo cual no era tan fácil como el cliente pensaba**
3. Se realizó un desarrollo iterativo: **NO**
4. Se realizó un diseño utilizando diagramas UML**: SI, utilizando Diagramas de clases**.
5. Para realizar las pruebas, se trabajó con algún Framework específico. **NO.** Especifique cual utilizó o bien porqué no utilizó**: Porque no conocemos, ni ninguno del equipo utilizó alguna vez algún Framework, lo cual íbamos a perder tiempo aprendiendo para poder después implementarlo**
6. Explique cómo garantizó que los cambios no propagaran errores en otras secciones que ya funcionaban bien**: Testeando varias veces por diferentes usuarios.**
7. Se fijaron WIP (límites de tareas por actividad) en la pizarra de tareas. **NO**. Justifique. **Al no ser proyectos tan grandes, no se fijaron límites de tareas por actividad.**

8. Se realizaron entregas parciales de producto desarrollado al cliente. **SI**. Especifique el tiempo **una semana.**

OPINION SOBRE EL RITUAL DE FINALIZACIÓN

1. En la reunión de entrega final, indique para cada sistema si considera que es necesario, y que beneficios se obtiene de esta reunión. **SI. En la última reunión se pudo obtener un balance final sobre cómo trabajo el equipo. Los tiempos de desarrollo que empleo cada desarrollador y las habilidades de cada uno**

2. De la reunión de reflexión final de cada sistema, especifique qué opiniones surgieron. **Las opiniones que surgieron fueron sobre como se podría trabajar a futuro. Las diferentes cosas que "se podrían haber hecho".**

3. Considera que hubiera sigo más productivo tener a los stakeholders de los sistemas en la reunión de revisión final. **SI.** Explique: **Porque es necesario la opinión final del cliente. Sobre si realmente era eso lo que esperaban o si bien "confiaban" en lo que habíamos hecho**

OPINION GENERLA SOBRE LA EXPERIENCIA

1. Al inicio del proyecto consideraba posible la realización de la experiencia. **NO.** Justifique su respuesta: **Debido al tiempo y al no haber trabajado nunca juntos, nos parecía algo inalcanzable**

2. Según la experiencia realizada, esta: **CONFORME** con los resultados obtenidos. Justifique: **Se realizó mucho más de lo que se había previsto desde un principio.**
3. Se tuvo colaboración permanente con el cliente del gimnasio. **SI**. Explique como: **Mediante consultas y reuniones en clases, se pudieron aclarar muchas cosas**
4. Se tuvo colaboración permanente con el cliente de la librería. **SI**. Explique como: **Mediante consultas y reuniones en clases, se pudieron aclarar muchas cosas**
5. Fue necesaria mayor participación de alguno de los clientes aparte de las reuniones establecidas: **NO**.
6. Especifique las herramientas utilizadas y los beneficios de cada una
    - **Kanbanize para seguimiento del proyecto**
7. Especifique herramientas utilizadas para el seguimiento del proyecto y su opinión sobre la misma: **Kanbanize, si bien es una herramienta que se puede ver que tareas está realizando cada individuo en cada instante, nosotros como grupo no la implementamos del todo**
8. Quién se encargaba de actualizar el tablero: **analista**
9. Se desarrolló con reutilización**. SI**.
    - Si es afirmativo ¿Cómo se seleccionan los artefactos a reutilizar? **Se seleccionaron mediante observar de las funcionalidades que eran iguales en ambos sistemas. Por ejemplo, el menú, al igual que las promociones, fueron objetos reutilizados para el gimnasio**
    - ¿Considera que se demora menos tiempo reutilizando? Justifique. **Sí, ya que se utilizan cosas ya creadas, por lo que no es necesario "inventar la rueda" cuando "la rueda ya esta creada.**
10. Trabajaría con Genexus**. SI.** Justifique: **Actualmente me encuentro trabajando con Genexus. Al no ser muy amigo de la programación, encontré en genexus la herramienta más amigable, y más "compatible" conmigo mismo para desarrollar software, y realizar análisis de sw.**
11. Otros comentarios: **no contesta**

Muchas gracias por su colaboración.

10 ANEXO4: CAPTURA DE ALGUNAS PANTALLAS DE LOS SISTEMAS DESARROLLADOS EN LA EXPERIENCIA

10.1 DESCRIPCIÓN

En esta sección se presentan algunas capturas de pantallas de los sistemas desarrollados por ambos grupos en la experiencia de aplicación práctica de la propuesta de este trabajo. Se presentará primeramente pantallas del sistema de la librería y luego del sistema del gimnasio para ambos grupos.

10.2 SISTEMA DE LIBRERIA

10.2.1 Grupo 1

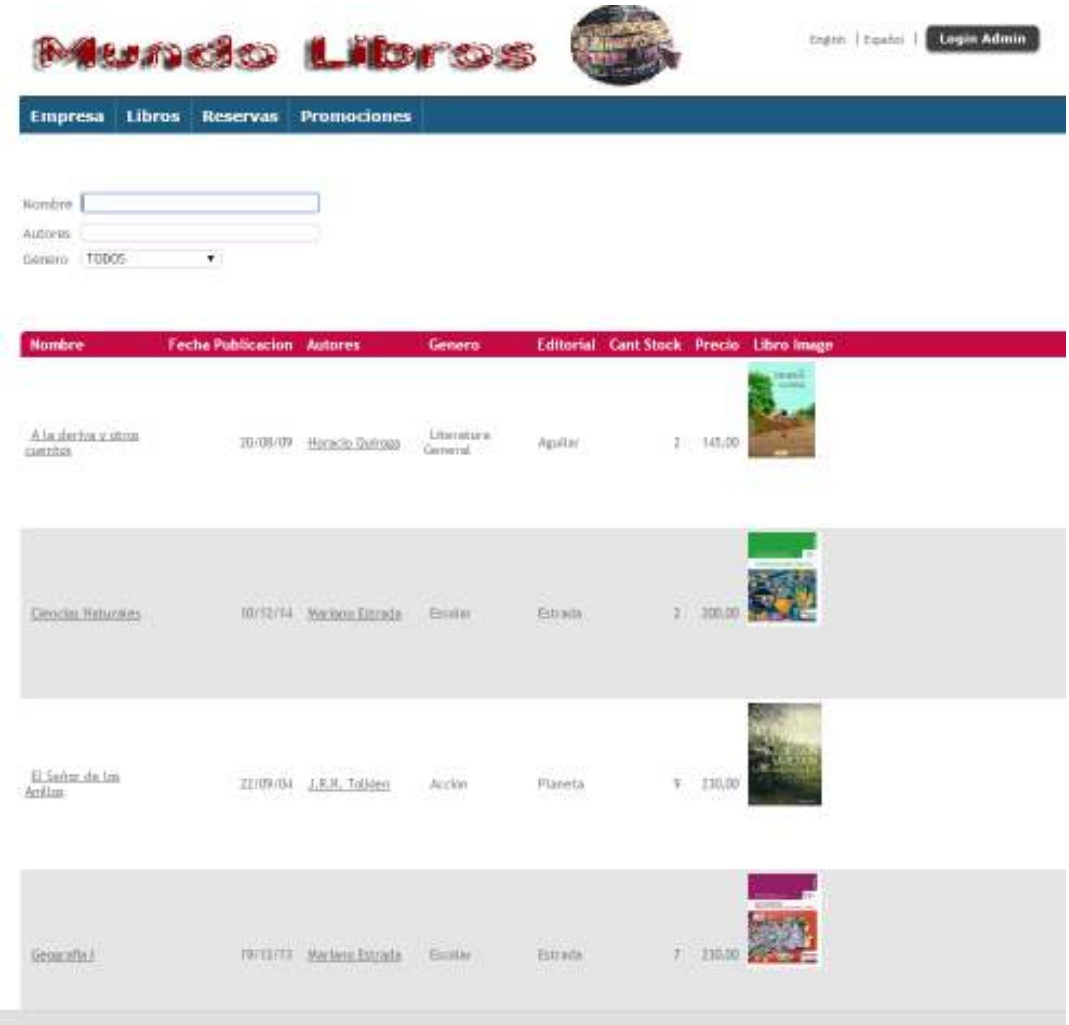


Figura 10.2.1: Home del sitio de la librería – Grupo 1

Figura 10.2.2: Reserva cliente del sitio de la librería – Grupo 1

Figura 10.2.3: Alta de reserva del sitio de la librería – Grupo 1

Figura 10.2.4: Gestión de datos de Proveedores del sitio de la librería – Grupo 1

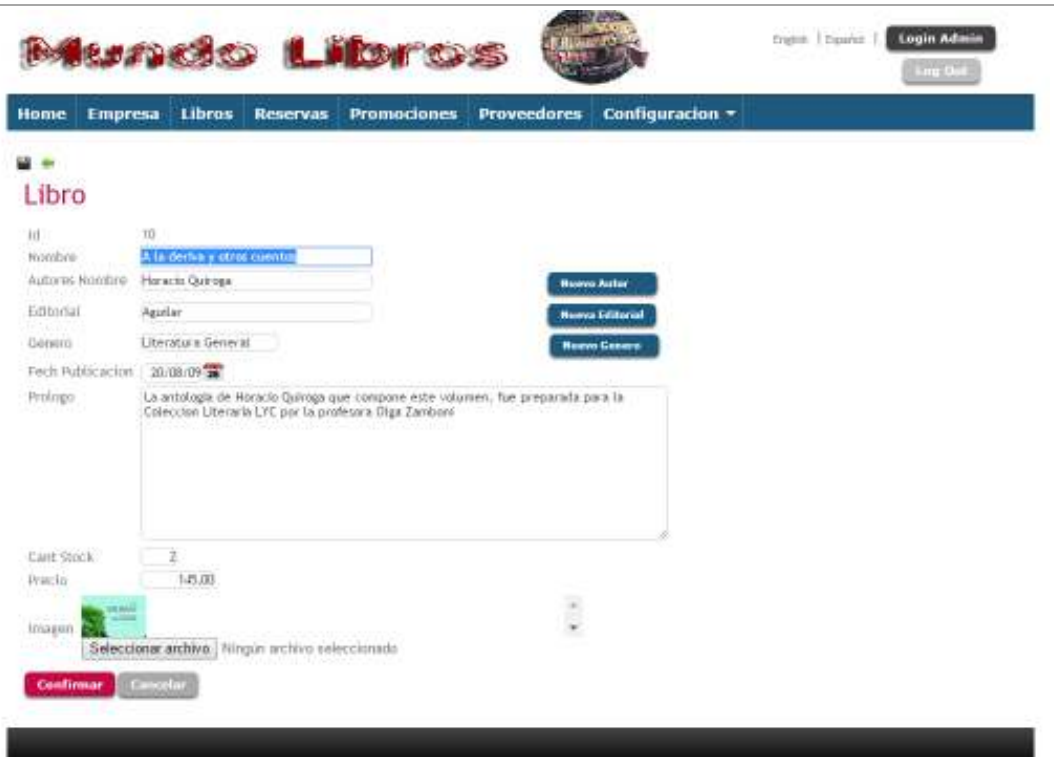


Figura 10.2.5: Alta de libro del sitio de la librería – Grupo 1

Figura 10.2.6: Alta de Promociones del sitio de la librería – Grupo 1

Figura 10.2.7: Configuración del sitio de la librería – Grupo 1

Figura 10.2.8: Configuración del sitio de la librería – Grupo 1

10.2.2 Grupo2

Figura 10.2.9: Home del sitio de la librería – Grupo 2

Figura 10.2.10: Contacto del sitio de la librería – Grupo 2

Figura 10.2.11: Catálogo de libros del sitio de la librería – Grupo 2

Figura 10.2.12: Vista Administrador del sitio de la librería – Grupo 2

Figura 10.2.13: Alta de libro del sitio de la librería – Grupo 2

Figura 10.2.14: Alta de autor del sitio de la librería – Grupo 2

Figura 10.2.15: Opinión del sitio de la librería – Grupo 2

10.3 SISTEMA DEL GIMNASIO

10.3.1 Grupo 1

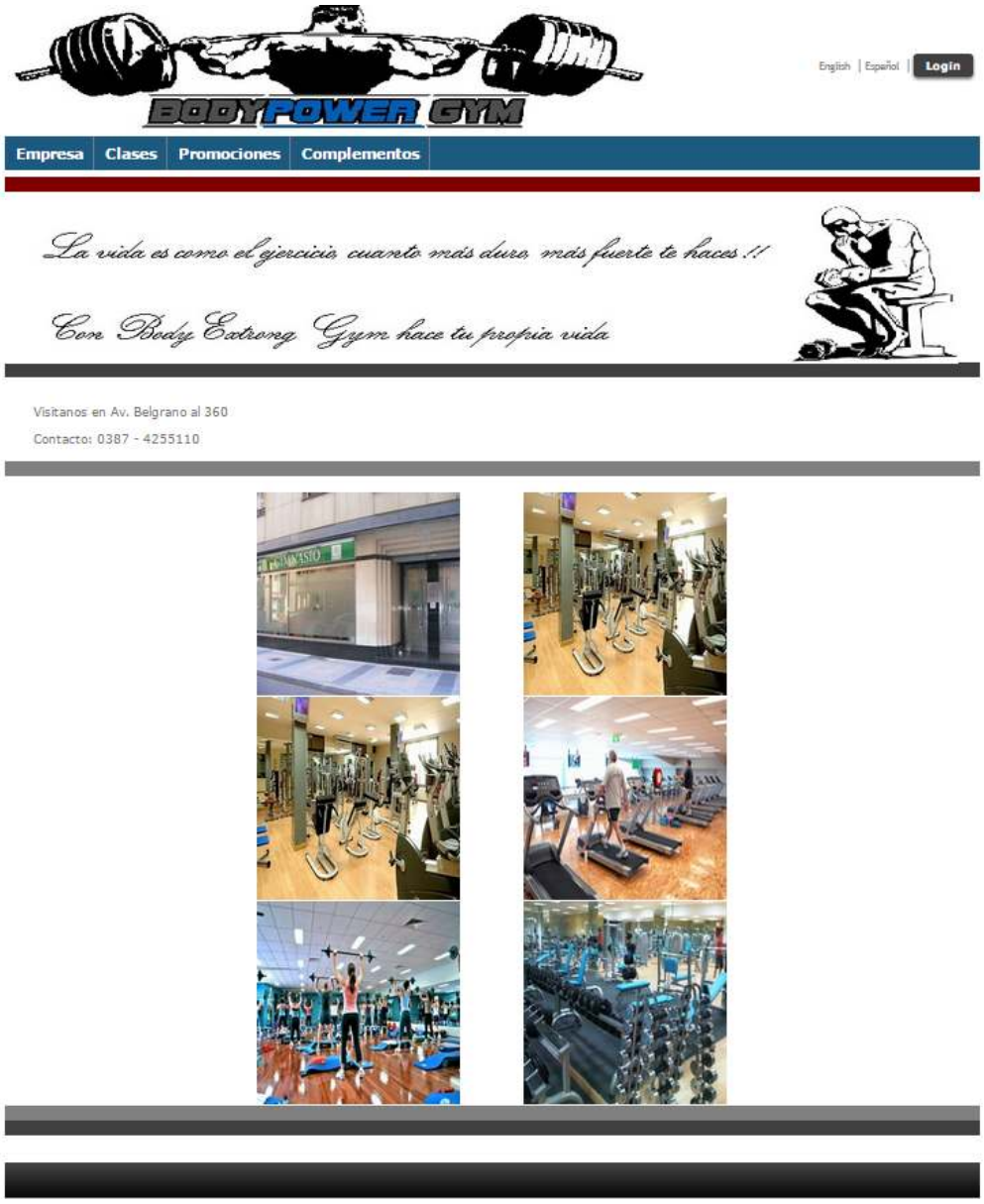


Figura 10.3.1: Home del sitio del gimnasio – Grupo 1

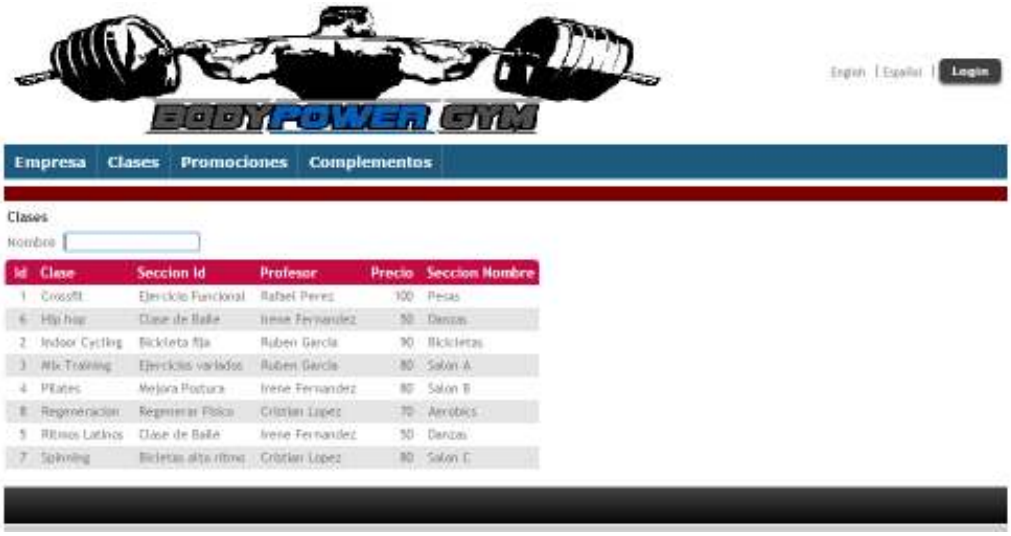


Figura 10.3.2: Listado de clases del sitio del gimnasio – Grupo 1

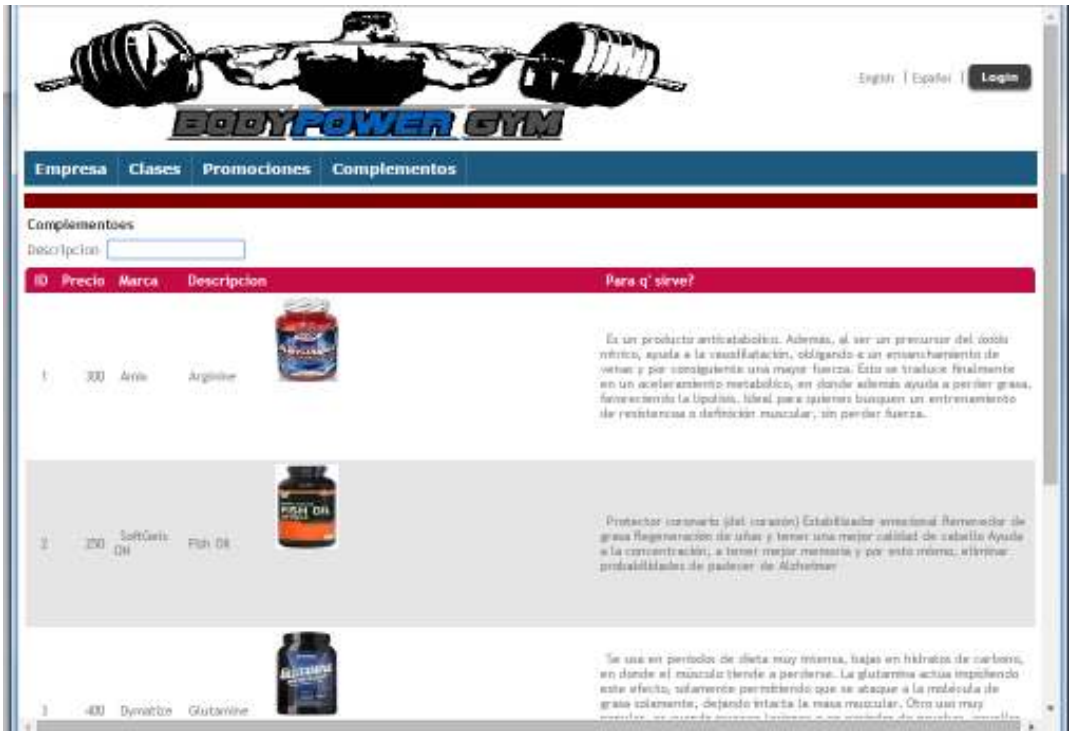


Figura 10.3.3: Listado de complementos del sitio del gimnasio – Grupo 1

Figura 10.3.4: Listado de promociones del sitio del gimnasio – Grupo 1

Figura 10.3.5: Datos socio del sitio del gimnasio – Grupo 1

Figura 10.3.6: Opciones para socios del sitio del gimnasio – Grupo 1

Figura 10.3.7: Opciones de administración - del sitio del gimnasio – Grupo 1

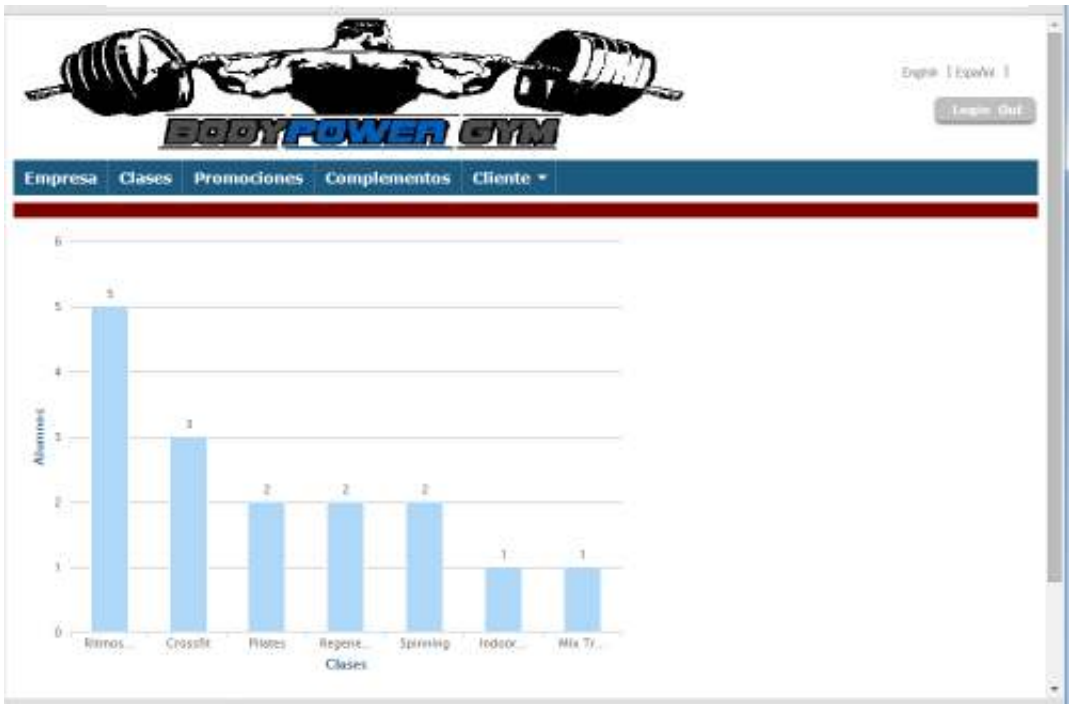


Figura 10.3.8: Datos de alumnos por clase del sitio del gimnasio – Grupo 1

Figura 10.3.9: Calendario de clases del sitio del gimnasio – Grupo 1

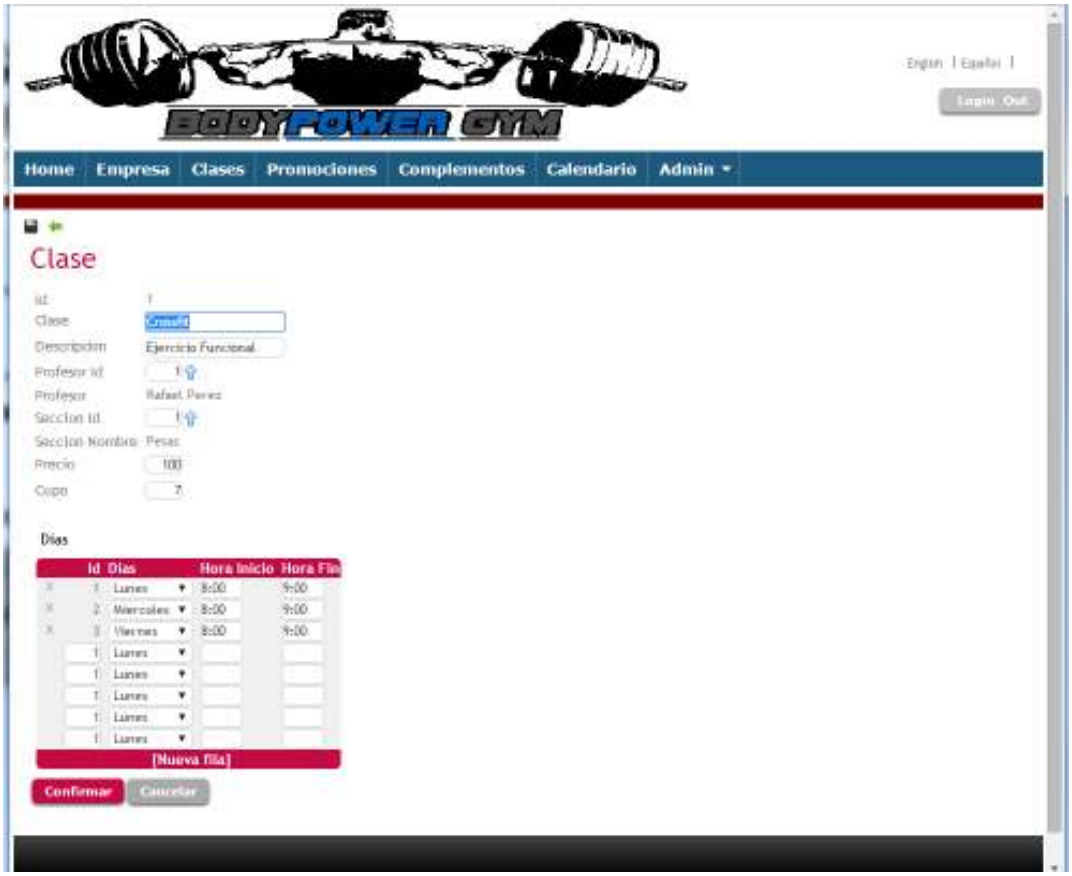


Figura 10.3.10: Gestión de clases del sitio del gimnasio – Grupo1

Figura 10.3.11: Administración de salones del sitio del gimnasio – Grupo 1

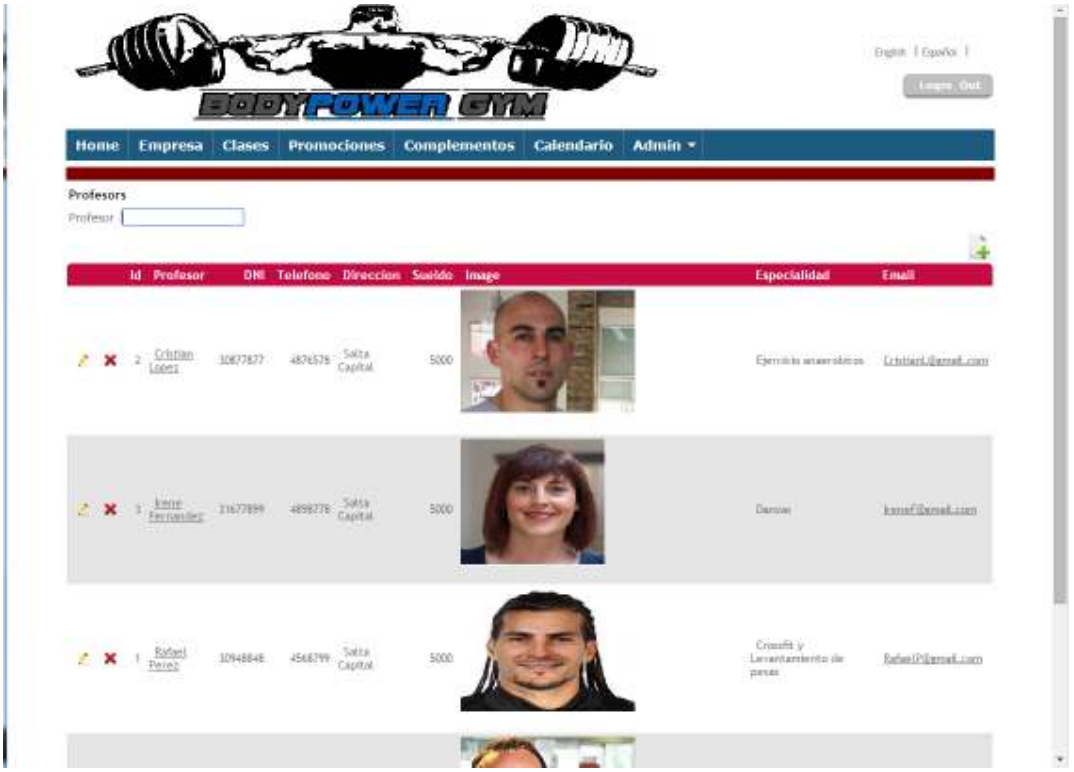


Figura 10.3.12: Profesores del sitio del gimnasio – Grupo 1

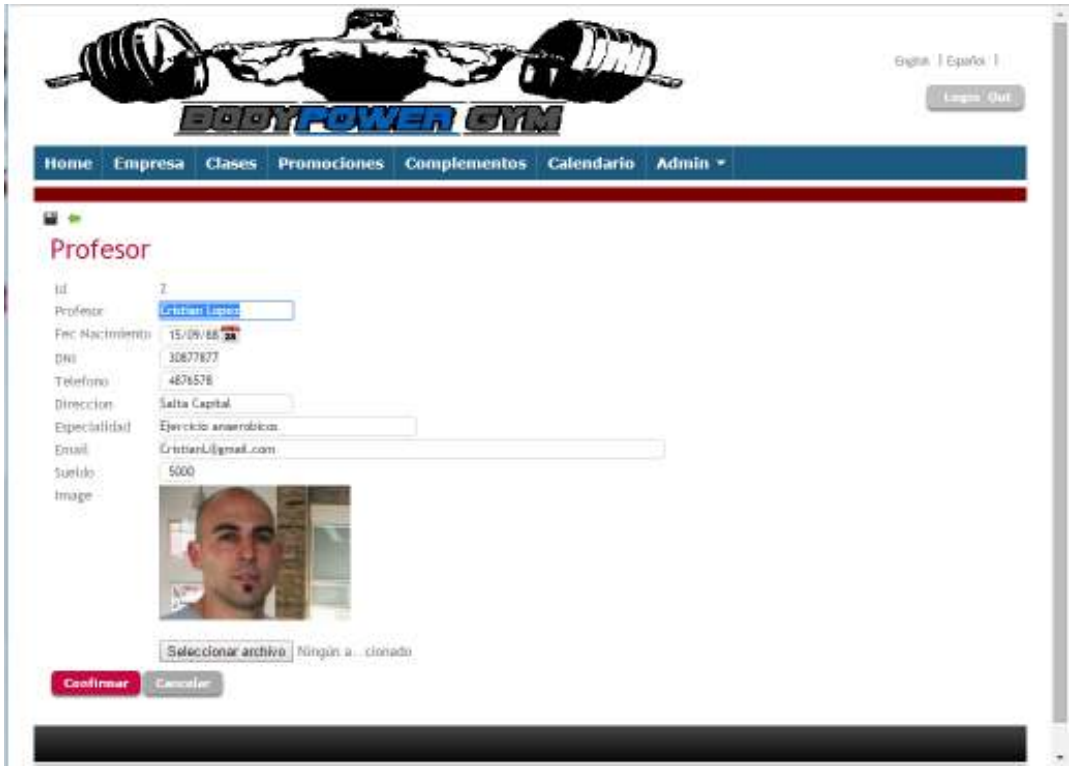


Figura 10.3.13: Administración de Datos de un profesor del sitio del gimnasio – Grupo 1

10.3.2 Grupo2

Figura 10.3.14: Home del sitio del gimnasio – Grupo 1

Figura 10.3.15: Clases del sitio del gimnasio – Grupo 2

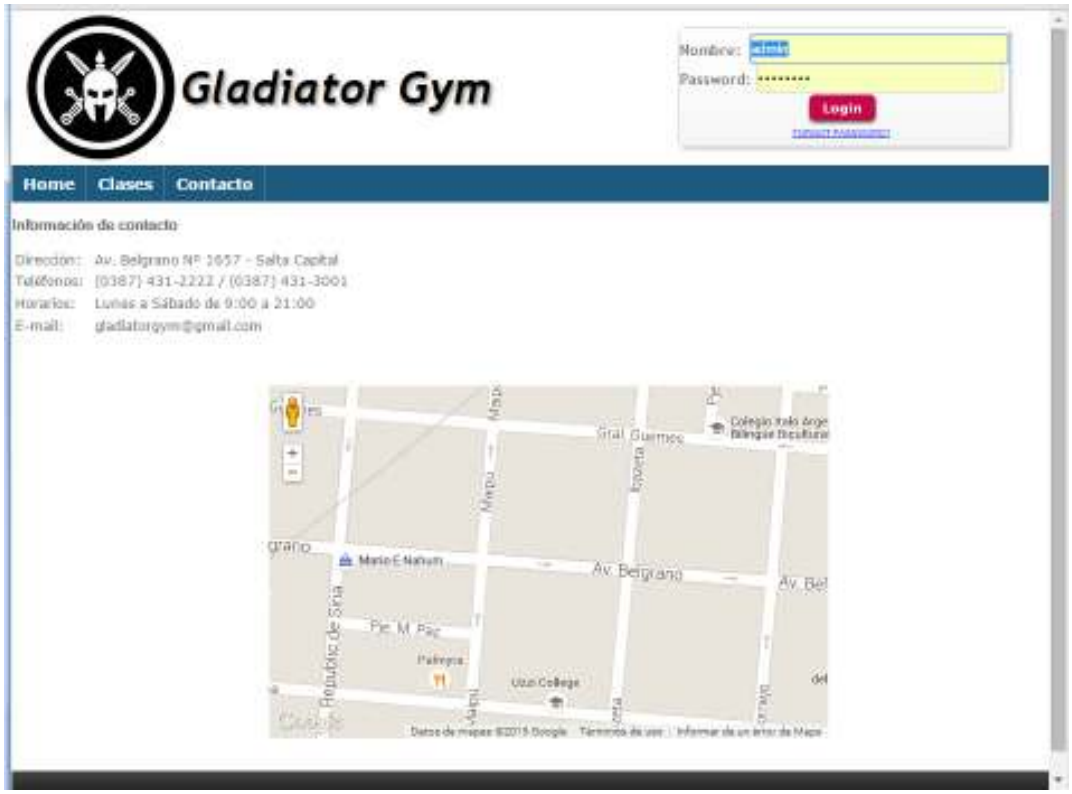


Figura 10.3.16: Contacto del sitio del gimnasio – Grupo 2

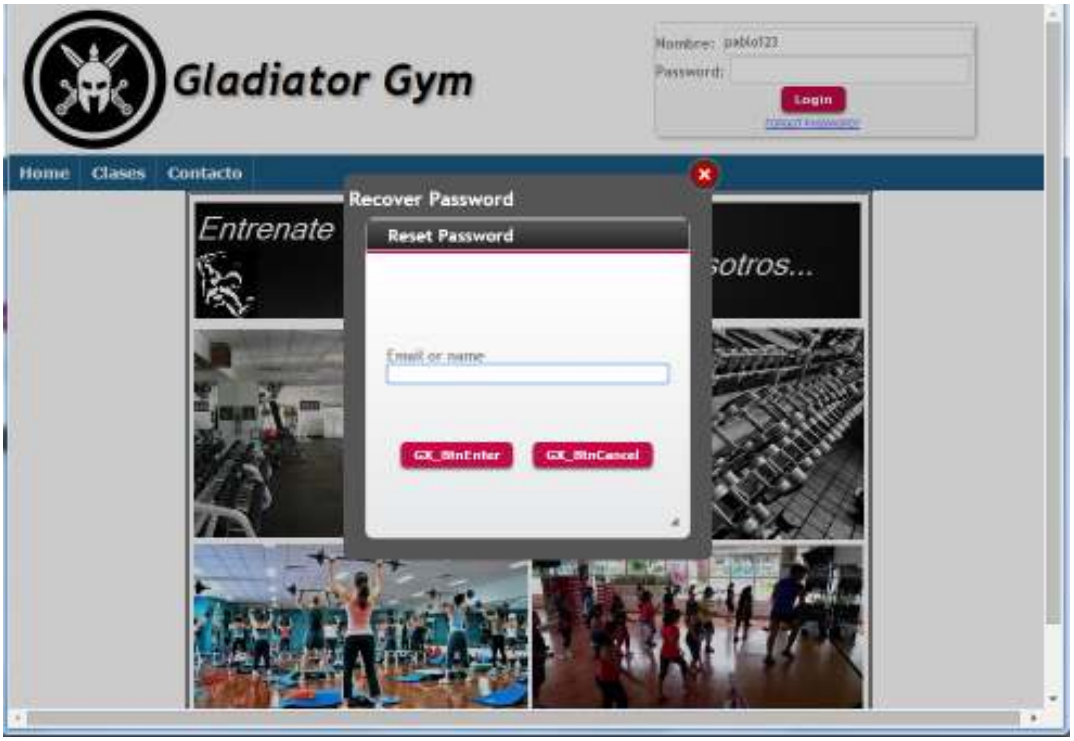


Figura 10.3.17: Login – password olvidada del sitio del gimnasio – Grupo 2

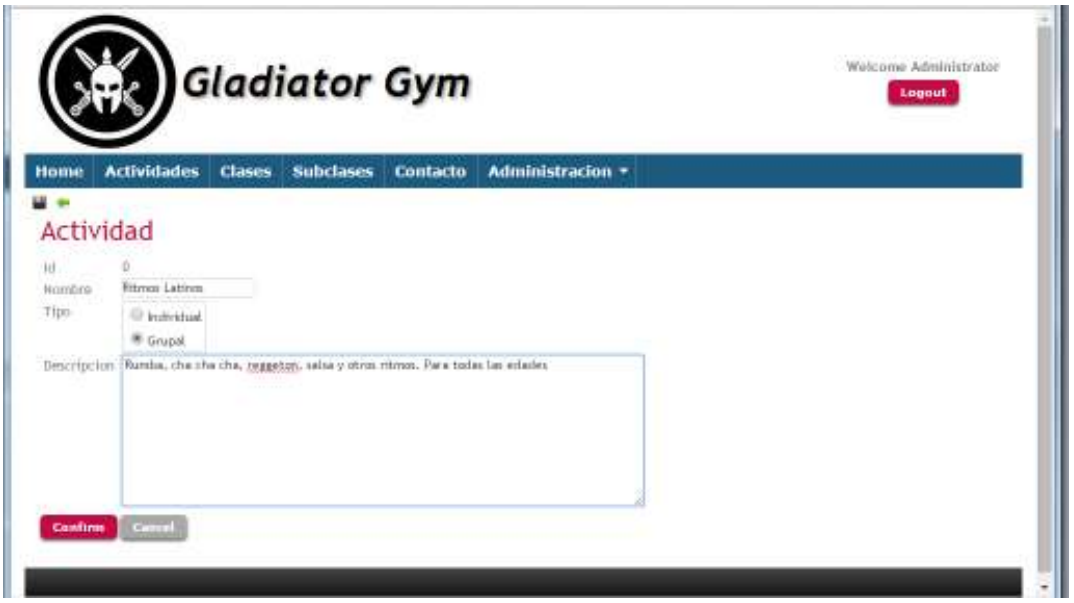


Figura 10.3.18: ABM de actividad del sitio del gimnasio – Grupo 2

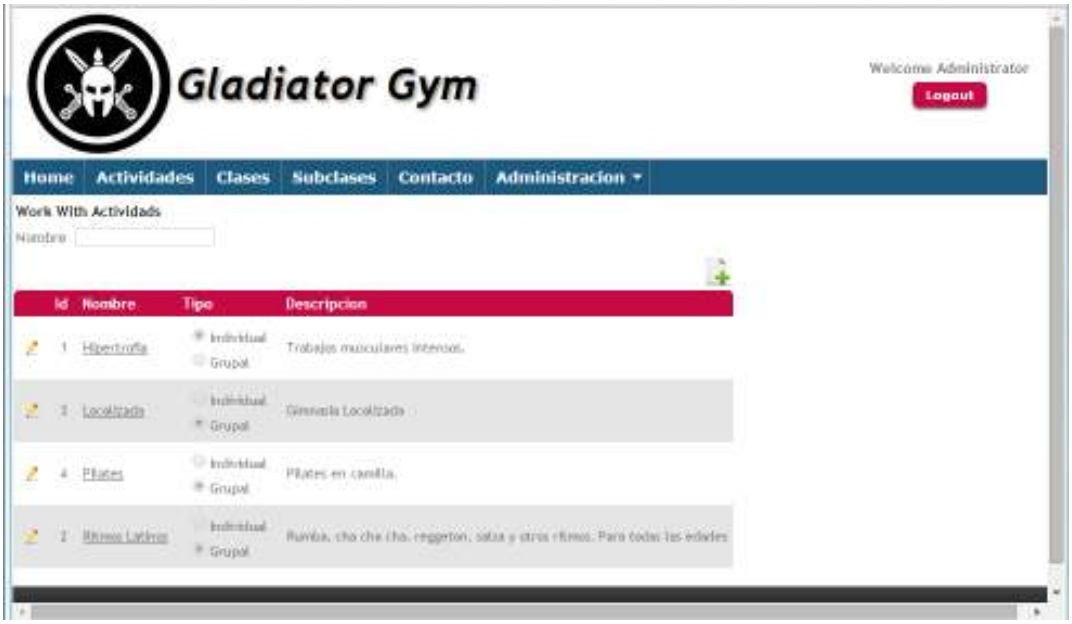


Figura 10.3.19: Listado de actividad del sitio del gimnasio – Grupo 2

Figura 10.3.20: Listado de Salas - del sitio del gimnasio – Grupo 2

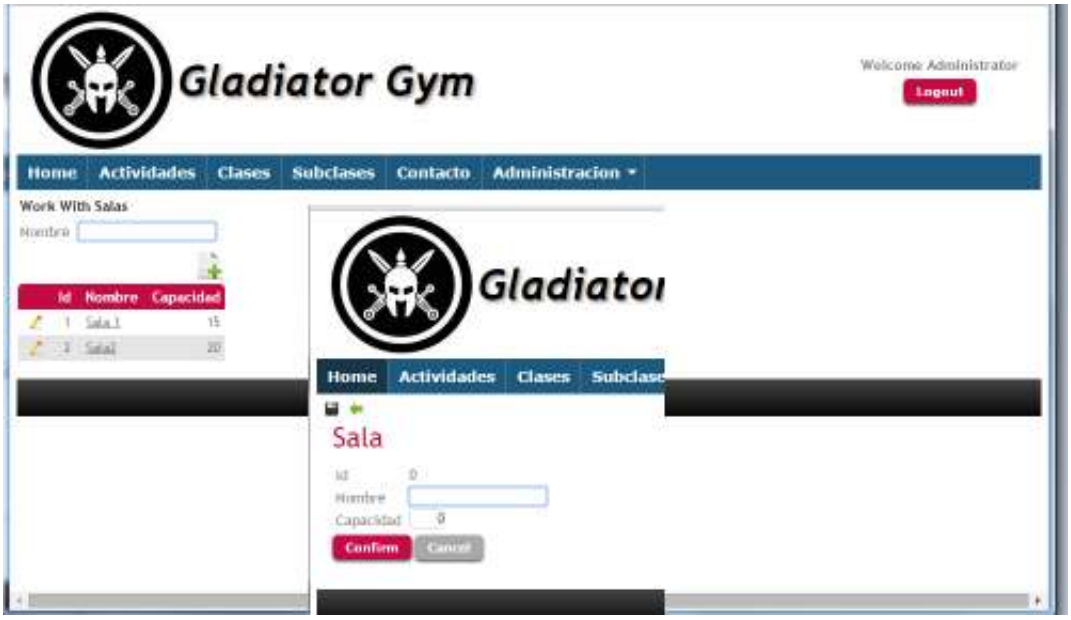


Figura 10.3.21: ABM de salas del sitio del gimnasio – Grupo 2

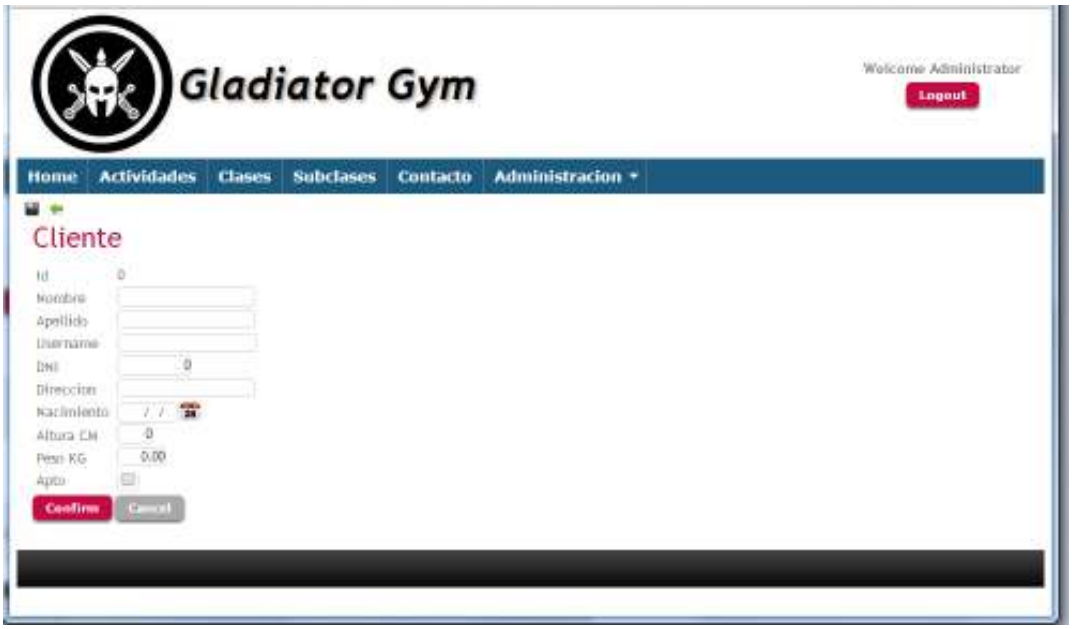


Figura 10.3.22: ABM de cliente del sitio del gimnasio – Grupo 2

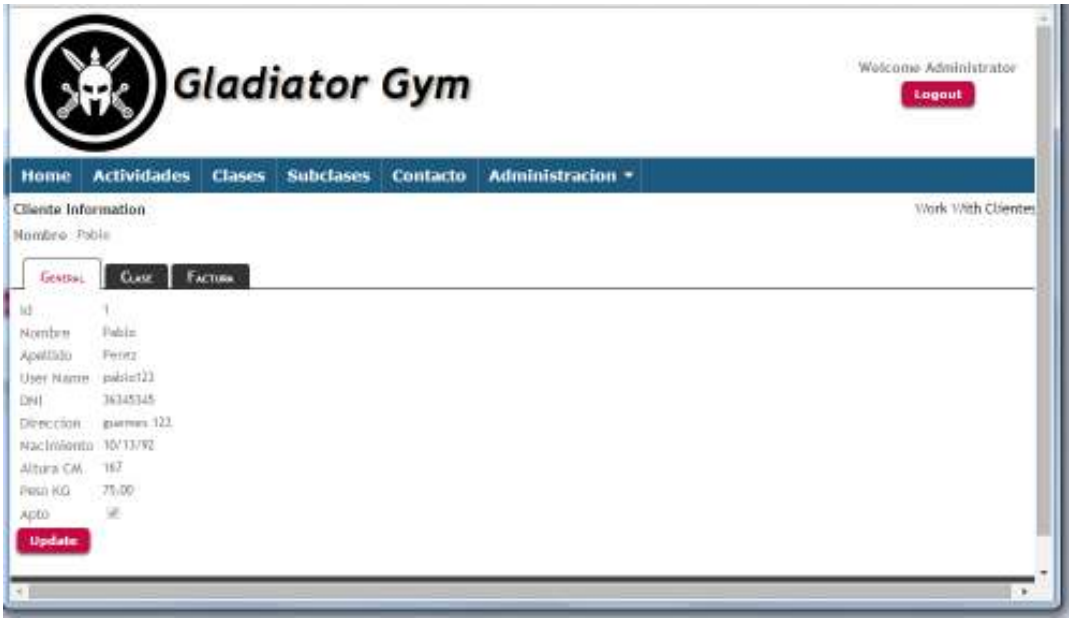


Figura 10.3.23: Datos de Cliente del sitio del gimnasio – Grupo 2

Figura 10.3.24: Listado de Cliente del sitio del gimnasio – Grupo 2

Figura 10.3.25: Menú de administración del sitio del gimnasio – Grupo 2

11 ANEXO5: ENCUESTA SOBRE OPINION DE PROPUESTA DE DESARROLLO BASADO EN CONOCIMIENTO SIGUIENDO PRÁCTICAS ÁGILES REALIZADA A SECTORES RELEVADOS

11.1 ESTRUCTURA DE ENCUESTA

11.1.1 Encuesta en blanco

La siguiente encuesta tiene Esta encuesta tiene la intención de poder sondear las posibilidades de ser aplicada en sector de desarrollo con las características suyas. Los datos recolectados serán utilizados en una tesis de maestría de Ingeniería de software de la UNLP y en un trabajo de investigación del CIDIA

Fecha de realización: Junio de 2015

1. ETAPAS DEL PROYECTO. La propuesta plantea tres etapas: Inicio Ágil, Producción y Ritual De Finalización. Considera que son aplicables a su situación. **INADECUADAS – POCO ADECUADAS – ALGO ACEPTABLES - ADECUADAS - MUY ADECUADAS.**

2. INICIO AGIL. Considera que proveería algún aporte esta etapa. **NO – POCO ALGO – ALGO – APORTE CONSIDERABLE - GRAN APORTE.** Justifique……
……...
……...
……...
……...

3. PRODUCCION. Considera viable realizar todas las reuniones. **SI – NO** – Justifique.
……...
……...
……...
……...

4. PRODUCCION. Liste las reuniones que figuran en la propuesta y que tiene un equivalente en su ámbito de trabajo actualmente indicando su frecuencia:
……...
……...
……...
……...
……...

5. RITUAL DE FINALIZACION: Considera probable realizar ambas reuniones. **MUY PROBABLE - PROBABLE – ALGO PROBABLE – POCO PROBABLE – IMPROBABLE.** Justifique: ……...
……...
……...
……...
……...

6. RITUAL DE FINALIZACION: Considera probable contar con participación de stakeholders en las reuniones de reflexión final. **MUY PROBABLE - PROBABLE – ALGO PROBABLE – POCO PROBABLE – IMPROBABLE**. Justifique:……..
……...
……...
……...

7. PIZARRA KANBAN: considera probable utilizar este tipo de pizarra. **MUY PROBABLE - PROBABLE – ALGO PROBABLE – POCO PROBABLE – IMPROBABLE**.

Justifique: ……...
……...
……
……...
……...

8. LUGAR DE TRABAJO. Considera que se puede compartir el lugar de trabajo entre todos los del equipo. **MUY PROBABLE - PROBABLE – ALGO PROBABLE – POCO PROBABLE – IMPROBABLE.** Justifique. ……...
……...
……
……...
……...

9. HERRAMIENTA DE INTEGRACION. Considera que se puede trabajar con GXServer. **MUY PROBABLE - PROBABLE – ALGO PROBABLE – POCO PROBABLE – IMPROBABLE**. Justifique. ……...
……...
……
……...
……...

10. PRUEBA. Considera probable poder utilizar algún Framework para automatizar las pruebas. **MUY PROBABLE - PROBABLE – ALGO PROBABLE – POCO PROBABLE – IMPROBABLE.** Justifique. ……...
……...
……
……...
……...

11. REUTILIZACIÖN. Considera probable realizar el desarrollo con reutilización en sus proyectos como plantea la propuesta. **MUY PROBABLE - PROBABLE – ALGO PROBABLE – POCO PROBABLE – IMPROBABLE**. Justifique: ……...
……...
……
……...
……...

12. ROLES PLANTEADOS. Considera probable incorporar los roles propuestos. **MUY PROBABLE - PROBABLE – ALGO PROBABLE – POCO PROBABLE – IMPROBABLE.** Justifique. ……...
……...
……
……...
……...

13. PROTOTIPACION. Considera probable incorporar la prototipación en las reuniones de ajuste o planificación inicial. **MUY PROBABLE - PROBABLE – ALGO PROBABLE – POCO PROBABLE – IMPROBABLE**. Justifique. ……...
……...
……
……...
……...

14. OPINION GENERAL DE LA PROPUESTA. Considera que es una propuesta que ayuda en estos entornos de desarrollo con desarrollo basado en conocimiento para acercarlo al desarrollo ágil. **NADA – MUY POCO – ALGO – BASTANTE – MUCHISIMO**. Justifique: ……...
……...
……

……..……...
……..……...

15. OTROS COMENTARIOS: ……
……...
……
……...
……...
……..……..……

Gracias por su colaboración.

11.1.2 Encuesta1

Fecha de realización: Junio de 2015

1. ETAPAS DEL PROYECTO. La propuesta plantea tres etapas: Inicio Ágil, Producción y Ritual De Finalización. Considera que son aplicables a su situación. **ADECUADAS**.

2. INICIO AGIL. Considera que proveería algún aporte esta etapa. **APORTE CONSIDERABLE**. Justifique. **Tener una idea común de todo el equipo es valioso, la propuesta propone algo simple. Se puede llegar a considerar incorporar en la empresa el inicio ágil, por las ventajas enunciadas en la propuesta y por no consumir demasiado tiempo.**

3. PRODUCCION. Considera viable realizar todas las reuniones. **NO.** Justifique. **Se solicita que las reuniones de ajuste sean presenciales, para que ahí solicite los cambios. Esto es poco probable, nos manejamos con sistema de tickets, salvo cambios necesarios complejos o profundos. El resto si son viables de realizar.**

4. PRODUCCION. Liste las reuniones que figuran en la propuesta y que tiene un equivalente en su ámbito de trabajo actualmente indicando su frecuencia: **SEGUIMIENTO** se realizan diariamente, **REFLEXION INTERNA** aproximadamente cada dos meses. **REFLEXION EXTERNA**, es una reunión similar, tiempo variable.

5. RITUAL DE FINALIZACION: Considera probable realizar ambas reuniones. **ALGO PROBABLE**. Justifique: **Se está trabajando para desarrollar un software para un hospital, es difícil contar con el cliente y los stakeholders con tiempo para reuniones, pero se reconoce los beneficios. Se deberían buscar alternativas**

6. RITUAL DE FINALIZACION: Considera probable contar con participación de stakeholders en las reuniones de reflexión final. **POCO PROBABLE**. Justifique: **Si se lograra la participación de los stakeholders como se plantea en la propuesta, se evitarían muchos malos entendidos, pero esto es poco probable. Hay que buscar alternativas**

7. PIZARRA KANBAN: considera probable utilizar este tipo de pizarra. **ALGO PROBABLE**. Justifique: **Trabajamos con una pizarra Scrum pero sería factible trabajar con la pizarra propuesta**

8. LUGAR DE TRABAJO. Considera que se puede compartir el lugar de trabajo entre todos los del equipo. **MUY PROBABLE**. Justifique. **Todos comparten el lugar de trabajo, nadie trabaja desde sus casas u otro lugar. A veces el horario no es el mismo**.

9. HERRAMIENTA DE INTEGRACION. Considera que se puede trabajar con GXServer. **PROBABLE**. Justifique. **Es una herramienta costosa, pero se está analizando utilizarla. El proyecto actual es muy grande**.

10. PRUEBA. Considera probable poder utilizar algún Framework para automatizar las pruebas. **MUY PROBABLE**. Justifique. **Actualmente utilizamos GxTest para ayudar con las pruebas. Se tienen varias personas avocadas a esa tarea exclusivamente. Estamos mejorando mucho en la producción trabajando de esta manera.**

11. REUTILIZACIÖN. Considera probable realizar el desarrollo con reutilización en sus proyectos como plantea la propuesta. **PROBABLE**. Justifique: **Se utiliza muchísima reutilización. Actualmente es el líder de proyecto quien determina que reutilizar, pero podría probarse de la forma que se sugiere en la propuesta, el tema es que el líder es quien más conoce del proyecto y sus objetos.**

12. ROLES PLANTEADOS. Considera probable incorporar los roles propuestos. **POCO PROBABLE**. Justifique. **Actualmente el cambio llevaría a malos entendidos, y el proyecto en curso es muy grande y crítico para arriesgar. Se debería realizar con mucha cautela, además todos están acostumbrados a rendir cuentas al líder del proyecto. Implicaría un cambio cultural grande que no serpia conveniente en estos momentos, tal vez a futuro**

13. PROTOTIPACION. Considera probable incorporar la prototipación en las reuniones de ajuste o planificación inicial. **MUY PROBABLE**. Justifique. **En reunión inicial es muy útil ayuda a evitar malos entendidos. Actualmente se está utilizando. En reuniones de ajustes o modificaciones no se puede porque generalmente los cambios se solicitan por sistema de tickets. A veces se realiza prototipación en el lugar de trabajo del cliente o se llevan prototipos construidos y se muestran a él o a quién él designe**

14. OPINION GENERAL DE LA PROPUESTA. Considera que es una propuesta que ayuda en estos entornos de desarrollo con desarrollo basado en conocimiento para acercarlo al desarrollo ágil. **BASTANTE**. Justifique: **Nosotros venimos realizando el cambio hacia ágiles, haber contado con estos lineamientos antes, nos habría facilitado varias cuestiones al inicio porque nos costó encontrar la forma de iniciar el cambio. Por ejemplo podría haber sido útil definir otros roles para separar la idea del líder, podría haberse replanteado si hace falta iteraciones o no. El trabajar por flujo tal vez podría volver más solidario a los miembros del equipo. Otras cuestiones como lugar de trabajo, herramientas propuestas, reutilización, algunas reuniones de reflexión ya venimos trabajando y nos damos cuenta de su necesidad**.

15. OTROS COMENTARIOS: **Trabajamos por iteración, pero se podría trabajar simplemente por flujo de trabajo como se plantea, definiendo para cada historia de usuario la fecha probable de entrega. Pero realmente hemos implementado trabajar por sprint. En el desarrollo actual de un sistema de aproximadamente cinco mil objetos, la reutilización es muy importante y el testeo siempre lo realizan más de 2 personas. Se integra y se va realizando backup de versiones anteriores y ante un error se vuelve inmediatamente a la versiona anterior**.

Gracias por su colaboración