Definición de un marco de trabajo para la implementación inicial de un Modelo de Proceso Software. Aplicación de un caso de Estudio.

Alicia Mon, Marcelo Estayno y Patricia Scalzone
{aliciamon, mestayno}@fibertel.com.ar
patricias@vemn.com.ar
G.I.S. – UNLaM[1]
Argentina

Abstract

The present article exposes the definition and practical application of a framework in the system area of an organization. Due to the total lack of the process, this framework has permitted to generate the need of the implementation of a definite Process Model for the software development. The described framework has been applied in a study case, just as it was done in other opportunities by different organizations with similar characteristics. This work provides the possibility of applying it slowly against an improvised and indefinite process, in which a methodology, a process model, a collection of techniques and tools converge. These elements are advisable for large and medium-size software development companies, and would facilitate their way to implement an integral process model which would continuously improve.

Keywords: Process Model, Team Model, Software Development Process.

Resumen

El presente artículo expone la definición y la aplicación práctica de un marco de trabajo, en el área de sistemas de una Organización que, a partir de la carencia total de proceso, ha permitido generar la necesidad de la implementación de un Modelo de Proceso definido, para el desarrollo de software.
El marco de trabajo que se describe, ha sido aplicado en un caso de estudio, así como en otras oportunidades en diferentes organizaciones de características similares. El mismo, provee la posibilidad de la puesta en práctica paulatina de un marco de trabajo frente a un proceso indefinido e improvisado, en el que confluyen una metodología, un modelo de proceso, una colección de técnicas y herramientas recomendables para las pequeñas y medianas empresas o áreas de desarrollo de software, que facilitaría el camino para la implementación de un Modelo de Proceso integral, de mejora continua.

Palabras claves: Modelo de Proceso, Modelo de Equipo, Proceso de Desarrollo de Software.

[1] Grupo de Ingeniería de Software. Secretaría de Posgrado. Departamento de Ingeniería e Investigaciones Tecnológicas. Universidad Nacional de La Matanza.

1. INTRODUCCIÓN

El presente caso de estudio describe la experiencia de la aplicación de un marco de trabajo que, a partir de la carencia total de proceso, ha permitido generar la necesidad de la aplicación de un Modelo de Proceso definido en el área de desarrollo de software de un Estudio Contable, cuyo negocio es el asesoramiento y consultoría de la administración e impacto de resultados económicos en empresas de diferentes rubros.

2. CARACTERIZACIÓN DE LA COMPAÑÍA

La principal actividad de la organización es la administración y valuación de activos empresarios, especialmente el análisis de activos fijos, activos financieros, almacenes, abastecimientos, compras, valuaciones confiables y contactos corporativos. Posee un área de desarrollo de software propia que le provee todos los sistemas para la gestión interna de la misma, así como el tratamiento de la información para sus clientes. Dada la experiencia de la empresa en el área, comenzó a vender sus productos software desarrollados internamente, como productos de gestión a otras compañías, con servicios de asesoramiento y tutoría ofrecidos sobre los productos software.
La compañía cuenta con un total de 50 empleados, con un equipo de 8 profesionales en el área de sistemas, encargados de organizar los recursos de sistemas, desarrollar software y brindar soporte técnico a toda la compañía.

3. DESCRIPCIÓN DE LA EXPERIENCIA

La empresa había obtenido la certificación de ISO 9001:2000 [ISO00], pero el área de desarrollo de software se encontraba en la siguiente situación: la gerencia de sistemas estaba encabezada por un programador que había desarrollado el primer sistema de información para la empresa, que si bien contaba con una vasta experiencia en el dominio del problema, tenía pocos conocimientos en la administración de proyectos. La empresa se encontraba en un período de crecimiento de clientes y servicios, y contaba con una escasa parametrización de los sistemas que se desarrollaban y sin la aplicación de ningún proceso definido, ni metodología de trabajo, a pesar de estar certificada, por lo tanto, cada vez eran mas las versiones de los sistemas que se sucedían y se debían mantener, generando que el estado del desarrollo resultara cada vez mas caótico.
Teniendo en cuenta la amplia experiencia acumulada por la organización en su negocio, sus desarrollos software comenzaron a venderse en el mercado, pero la demanda exigía una aplicación con interfaces gráficas, Web, con bases de datos distribuidas, con todos los avances de las nuevas tecnologías, para lo cual era sumamente necesario una migración, y reingeniería de los productos existentes. Estaban desarrollados en Visual Basic como lenguaje de programación, sin componentes, una arquitectura pobre, con pocas posibilidades de escalar y se tenían que pasar a un entorno Web. Sin dejar de lado que dichas aplicaciones, en sus múltiples versiones, debían seguir prestando servicio y respondiendo a los cambios solicitados por los clientes, hasta que se produjera el reemplazo.
El principal objetivo para iniciar un proceso de mejora residía en contar con la información necesaria para determinar que procesos faltaban, definir los objetivos de mejora, definir los procesos críticos y la metodología a utilizar, preparar un plan de mejora aplicable a la organización, determinar los recursos necesarios y comprobar la eficiencia de la aplicación.

4. RESULTADOS OBTENIDOS

La iniciativa de mejora provino del equipo de desarrollo, al darse cuenta que, de la forma en que se encontraban realizando sus desarrollos, les resultaba cada vez mas inmanejable. Tenían en claro que de alguna forma debían ordenarse y crecer.

La actividad de mejora comenzó por inducir al equipo de trabajo a ciertos cambios, con el objetivo de crear una "cultura" de proceso, adaptando herramientas que facilitaran el desarrollo de software reconocido en el mercado, con la dureza de los formalismos, pero tomando la versión más ágil posible. En consecuencia, se comenzó con una capacitación en los temas referentes a documentación, basándose en el Lenguaje de Modelado Unificado (UML) [BOO03]. Luego surgió la necesidad de un proceso de capacitación en forma teórica, facilitando la toma de conciencia de las necesidades de su aplicación.

Se fue guiando al grupo de desarrollo en la detección de sus propias falencias y necesidades. Se fijaron pautas sencillas y realizables que produjeron los siguientes cambios: reasignación de roles, definición de plantillas de documentación con herramientas de Office conocidas y manejadas por ellos mismos y se decidió comenzar por la etapa de requerimientos, observándose rápidamente unos primeros resultados. A consecuencia de ello se decidió avanzar un paso más en el ciclo de capacitación y por las características de la organización y del equipo de sistemas, se eligió adoptar la metodología del Proceso Unificado de Desarrollo Software, en una versión ágil [LAR04], que les permitió capacitarse rápidamente y comenzar a implementar un proceso simple, relativamente informal, para convertirlo paulatinamente en un proceso mas riguroso, capaz de soportar certificaciones mas específicas.

La característica principal del proceso seleccionado, es que está basado en un ciclo de vida de proyectos iterativo e incremental. Esto es ideal para mejorar la productividad del área, sobre todo cuando se cuenta con recursos humanos escasos para el volumen de trabajo. Este ciclo de vida permite aprovechar al máximo los recursos existentes, y no tener roles ociosos, esperando que se complete la actividad precedente. Lo dificultoso es lograr verdaderamente las iteraciones, sin recurrir a la utilización de una metodología en Cascada.

El éxito de este emprendimiento consistió en avanzar paulatinamente, logrando que cada uno de los participantes asumiera un rol de forma natural, comprendiera las limitaciones existentes y visualizara un cambio favorable a futuro al incluir un procedimiento definido y establecido. De esta manera, el equipo avanzaría en forma conjunta y no aisladamente.

Los pasos que se realizaron como marco de trabajo inicial fueron los siguientes:

4.1. Se definieron los roles.

4.2. Se propusieron templates/plantillas de documentos, de reportes, etc.

4.3. Se seleccionó una herramienta Case para que colabore con el equipo de trabajo.

4.4. Se confeccionó una lista de Requerimientos.

4.5. Se confeccionó una lista de riesgos, y se los priorizó, dado que el proceso se orientaba a riesgos.

4.6. Se elaboró el documento de visión del proyecto, en un ámbito de discusión y participación de todo el equipo de trabajo en las definiciones.

4.7. Se realizó la especificación de los requerimientos, utilizando como forma de documentación el modelo de Casos de Uso, lo que posibilitó la definición inicial del testing funcional.

- A los casos de uso encontrados se los amplió con un diagrama de transición-estados y un diagrama de actividad.

4.8. Se definió la arquitectura de la aplicación, resaltando los puntos importantes que debe contener y quién debe realizar cada componente.

- Se elaboró un diagrama de componentes de la arquitectura.
- Se realizó un diagrama de secuencia, para discutir e informar como se realizaría la interacción entre componentes, o más en detalle, entre objetos.

4.9. Se establecieron reglas de calidad para escribir código.

4.10. Se planificaron las iteraciones [3], en función de la lógica, de la cantidad de integrantes del equipo involucrados, tratando que sean lo más cortas posibles. Para cada iteración:

- Se especificó detalladamente cada Caso de Uso involucrado.
- Se definieron los objetos necesarios para conformar el modelo de datos y el modelo lógico de clases, que va creciendo en cada iteración.
- Se realizaron algunos diagramas de secuencia adicionales, para observar la interacción entre objetos.
- Se codificaron los casos de uso seleccionados.
- Se realizaron pruebas unitarias de código.
- Se realizaron las pruebas funcionales, utilizando las especificaciones realizadas.
- Se realizaron pruebas de integración.

4.11. Se realizó la migración de datos, con una prueba en paralelo.

4.12. Se realizó la puesta en marcha.

La figura 1 presenta el diagrama de actividades realizado como marco de trabajo.

4.1. Definición de roles

El área de desarrollo contaba con 8 desarrolladores, todos cumplían la misma función, sin importar su experiencia. El desarrollo de nuevas aplicaciones estaba mezclado con el mantenimiento de las ya existentes. El gerente de sistemas también realizaba tareas como desarrollador, dedicándose principalmente a lo referido a la administración de la base de datos, lo que generaba poca atención y dedicación a la gestión de las diferentes actividades.

Ante la decisión de definir roles, se tomó el modelo de “equipos de pares” propuesto por el Microsoft Solution Framework (MSF) [HAY02] en su versión para desarrollo de aplicaciones ágiles, cuya principal característica es la comunicación, y el trabajo por consenso, dejando de lado el modelo jerárquico tradicional. Esto se pudo adaptar fácilmente, dado que el grupo era pequeño y dispuesto al cambio.

La primer tarea de ordenamiento de roles consistió en separar las actividades de desarrollo, de las tareas de adquisición de conocimientos y análisis, incorporando una persona mas al equipo de trabajo, que recibiera capacitación para sumarse al área. Se diferenció el líder de proyectos, y se comenzó a hablar de arquitectura de la aplicación, en una primera instancia, para luego identificar este el rol entre los pares.

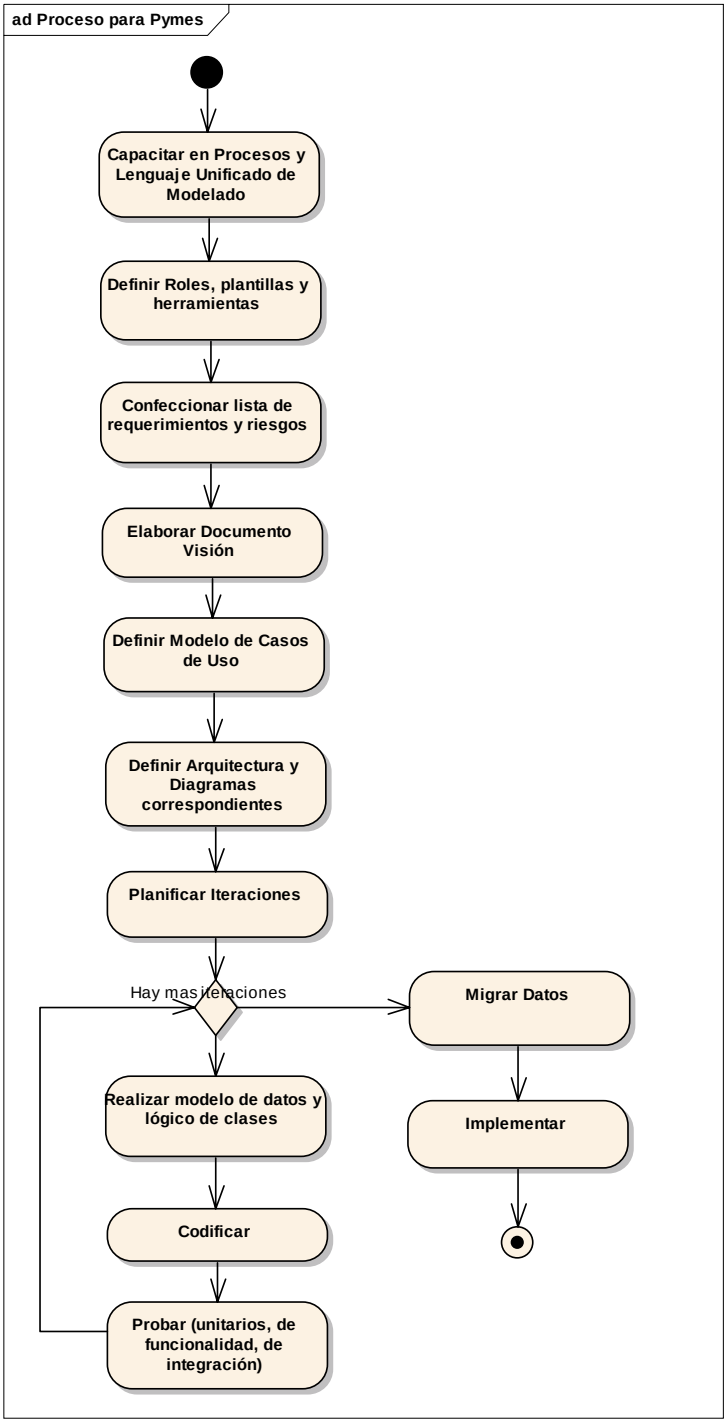

Figura 1: Diagrama de Actividades

Una vez realizada la diferenciación de funciones y la definición de roles, el trabajo se centró en separar las actividades de corrección y mantenimiento de los sistemas ya existentes, de las nuevas aplicaciones que necesitaban ser desarrolladas. Esto generó algunas diferencias entre los pares, porque todos querían trabajar bajo la nueva metodología que se estaba implantando, hasta que se logró un acuerdo y se comprendió que todos los roles tenían objetivos igualmente valorados, y que cada uno de estos objetivos contribuía a cumplir el objetivo global del área, de forma independiente al proyecto asignado.

Esto comenzó a ordenar las actividades, el orden en que debían realizarse y el trabajo generado, a partir de la división de roles, funciones y objetivos claramente definidos para cada uno de los integrantes del equipo.

Proceso Generado. *El modelo de equipos de trabajo utilizado, se adapta bien en las pequeñas y medianas empresas de desarrollo de software, especialmente en aquellas que no tienen incorporado ningún proceso definido. Esta manera de conformar el equipo, y trabajar por consenso, genera descompresión y mejor dinámica de trabajo en grupos pequeños, donde surge de forma habitual, una permanente necesidad de cambio.*

4.2. Definición de Plantillas de documentos, de reportes, etc.

La necesidad de ordenamiento en las actividades generó la necesidad de documentar. Resultaba imprescindible ordenar las solicitudes de cambios y nuevos requerimientos a las aplicaciones existentes. Para comenzar con esta documentación, se diseñó un formato de memo simple, que debía ser completado por los usuarios y enviado por mail. Cada uno de los memos se registraban en una planilla simple, en la cual se registraba la prioridad de la solicitud, se designaba el responsable de efectuar el cambio y la fecha de entrega estimada.

Luego se fueron incorporando otros documentos, clasificando los que serían obligatorios, y los que podrían usarse bajo condiciones de excepción. Se tomaron las plantillas modelos del Proceso Unificado [JAC03] y del Microsoft Solution Framework [GET02], adaptándolas y simplificándolas a las particularidades de la organización. A partir de allí, surgió una plantilla para el registro de las minutas de reunión, para documentar cada entrevista de adquisición de conocimiento con los usuarios.

Proceso Generado. *La Incorporación de documentos escritos logró cambiar la dinámica de trabajo, la comunicación entre el equipo de desarrollo y los usuarios, y modificó notablemente la imagen de la gerencia de sistemas. A partir de los registros, los propios usuarios comenzaron a sugerir mejoras a las plantillas definidas.*

4.3. Selección de una herramienta Case

Cuando se capacitó al equipo de trabajo en el Lenguaje de Modelado Unificado (UML) [BOO03], se realizaron los diagramas con la herramienta informática de dibujo que poseía la empresa, pero cuando se consideró que era mas prolijo y ordenado almacenar todo junto, se decidió seleccionar una herramienta Case que fuera eficiente bajo ciertos parámetros y de un costo accesible. Se optó por trabajar con Enterprise Architect [SPA06].

En primer término se realizaron todas las pruebas con la versión libre de prueba, y cuando se comprendió que se necesitaba toda la herramienta completa, se adquirió el producto para trabajar de forma compartida, y realizar también las reingenierías necesarias.

Proceso Generado. *La selección de una herramienta única, requiere determinar la estructura a utilizar para organizar todos los proyectos de una misma manera, (por ej. el utilizado en este caso fue el Modelo de 4+1 Vistas, de Kruchten [KRU00]).*

4.4. Confección de una lista de Requerimientos

Como parte de la incorporación de documentos, la lista de requerimientos comenzó con el registro de los mismos en una planilla de cálculos, con la simple enumeración de las solicitudes de los usuarios. Esta lista, en forma paulatina, fue cobrando importancia y se transformó en la base para la definición de la Visión del Proyecto.

Proceso Generado. *La incorporación de una plantilla para registro de los requerimientos permitió mejorar el entendimiento de los mismos, y constituyó la base para la definición de la Visión del Proyecto.*

4.5. Confección de una lista de riesgos

Se elaboró una lista muy sencilla con los principales riesgos detectados para cada proyecto, ordenados por prioridades, según el proceso de Administración de Riesgos [ROB02], indicando el plan de mitigación y contingencia. Los resultados fueron incorporados en los siguientes cronogramas. De esta manera fue mejorando considerablemente la medición y seguimiento de cada desarrollo.

Proceso Generado. *La elaboración de una lista de riesgos, permitió identificarlos, evaluarlos, priorizarlos y definir su seguimiento con la incorporación de los mismos al cronograma y coste del proyecto.*

4.6. Elaboración del documento Visión del proyecto

El documento Visión fue el primer hito significativo luego de la capacitación realizada. Permitió demostrar a la organización que la erogación realizada en capacitación y consultoría en el sector, comenzaba a brindar resultados positivos, considerando, especialmente, que al no existir previamente una cultura de proceso, el "no estar escribiendo líneas de código" no significaba una pérdida de tiempo, ni de inversión.

La realización de este documento implicó poco más de un mes de trabajo, se presentó a la organización, estableciendo como pauta de entregable, que si en un período de 10 días no había respuesta alguna, se daría por aprobado. Antes de que venciera el plazo, uno de los interesados observó que su objetivo no estaba reflejado en el documento, y no por casualidad era del sector que había tenido una participación muy pequeña en el proceso de adquisición de conocimientos. Luego que se dispuso del tiempo necesario, al cabo de 3 semanas la Visión del proyecto estaba finalizada, incluyendo a todos los sectores involucrados.

Por primera vez, en la organización se discutía en base a un documento escrito, y no sobre una aplicación codificada, esto facilitó la discusión y permitió ahorro de tiempo en las definiciones.

El modelo de documento esta basado en una plantilla y analizada a través de una historia de revisión. Este registro permitió controlar toda la interacción que ha ido teniendo el documento Visión durante su confección.

Proceso Generado. *Este documento permitió definir el alcance de los proyectos, incluyendo las características funcionales y no funcionales. La historia de revisión ha permitido controlar la interacción generado durante su confección. A partir de detectar su importancia, el documento Visión ha sido considerado obligatorio en la organización.*

4.7. Especificación de los Requerimientos de la Aplicación

Teniendo definida la lista de requerimientos, se encontraron los actores que participaban, y luego las funciones que llevarían a cabo esos actores. Esto se plasmó en un diagrama de Casos de Uso, un mapa funcional, que permitió ubicarse rápidamente en la aplicación, definir el rol responsable, y las relaciones entre los diferentes actores.

Para cada uno de los casos de uso definidos, se escribió una breve descripción del mismo para especificar su alcance y comprender si se habían considerado todos los requerimientos con sus prioridades.

Con el fin de representar los diferentes estados por los que pasaba una entidad o un subsistema, a partir de una condición o un evento dado, se completó el modelo de análisis con un diagrama de Transición de Estados, conocido también como Máquina de Estados.

Finalmente, se elaboró un diagrama de Actividad para visualizar la interacción entre sectores, o actividades que no quedarían informatizadas.

Proceso Generado. *La representación de los requerimientos funcionales a través de un Diagrama de Casos de uso permitió la identificación de los actores, describirlos y analizar sus relaciones.*

4.8. Definición de la Arquitectura de la aplicación

A partir de contar con los requerimientos funcionales y no funcionales bien conocidos, comprendidos y documentados, se realizó una reunión de arquitectura en la que participaron todos los miembros del equipo, tanto técnicos como funcionales. Con la plataforma tecnológica definida, se dio forma a la arquitectura de la aplicación. Las propuestas surgían de los desarrolladores más experimentados, pero la decisión de la arquitectura se tomó por consenso, fortaleciendo el modelo de equipos implementado.

La definición del diagrama de componentes se fue elaborando en forma conjunta, a partir de un bosquejo de los posibles componentes de la aplicación, en una pizarra blanca, dibujando, borrando, hasta lograr un diagrama de Componentes que contara con el acuerdo de todos los integrantes del equipo. Se utilizaron diferentes colores para identificar quienes serían los desarrolladores responsables de la implementación de los componentes. De esa manera, cada uno de los participantes conocía perfectamente su rol, su objetivo, con qué otro componente se comunicaba, de cuál dependía, y cuál era el objetivo global que se proponía alcanzar, sin ambigüedades.

Por otra parte, este trabajo conjunto permitió organizar los espacios de nombres (NameSpaces) necesarios para trabajar en la plataforma definida.

Los diagramas de secuencia también se dibujaron durante la misma reunión, en una pizarra, para discutir y definir cómo sería la interacción entre los componentes detectados anteriormente, y comprobar, en forma teórica, si estaban bien definidos.

Proceso Generado. *La discusión y definición conjunta de las diferentes alternativas de Arquitectura, con la participación de todo el equipo de trabajo, ha permitido el conocimiento necesario para dar comienzo al desarrollo de la aplicación.*

4.9. Definición de Reglas de calidad para escribir código

Para comenzar con el desarrollo de la aplicación, se establecieron normas muy sencillas de escritura de código, sin recurrir demasiado a conceptos teóricos, pero sí a la coherencia. El objetivo era definir reglas simples y que todos los desarrolladores las conocieran.

Se estableció utilizar notación Pascal Casing para nombres de clases y métodos y nombres de archivos, así como Camel Casing para nombres de miembros y parámetros de los métodos. Se definió comentar el código de forma clara, de modo tal que cualquier programador que lo leyese fuera capaz de entenderlo fácilmente. Por otra parte, el código debía estar encabezado con el nombre del programador, fecha de inicio, nombre del Caso de Uso y fecha de finalización. Asimismo, a medida que el desarrollo iría avanzando, se completaría siempre con la información referente a las fechas de cambios, nombre del solicitante y nombre del programador a cargo.

Proceso Generado. *Las reglas sencillas, permitieron que todos los desarrolladores del equipo las conocieran, les resultaran entendibles y fácilmente aplicables, de modo tal de lograr código reusable y entendible por cualquier miembro del equipo.*

4.10. Planificar las iteraciones

La definición y planificación del desarrollo resultó una tarea muy compleja. No se comprendía la manera en que se debía trabajar, ni cual era el beneficio de realizar iteraciones planificadas, si permanentemente requerirían cambios en la base de datos, en las clases, en los componentes y que cada iteración descubriría nuevas funcionalidades. La secuencia de resolución básica llevaba a un esquema de trabajo clásico, de desarrollo en cascada [ROY70].

Se definieron realizar iteraciones en función del equipo de desarrollo dedicado al proyecto, de los requerimientos funcionales y de la tecnología empleada. Por lo tanto, si se contaba con 3 desarrolladores, se agrupaban los requerimientos funcionales de forma tal que, en una iteración, el equipo pudiese terminar todo junto con un entregable de la aplicación, y luego pasarlo al área de prueba. Cada una de estas iteraciones daba origen a una nueva versión de la aplicación, que se registraba y controlaba con una herramienta automatizada.

En el proceso de planificación de las iteraciones, una vez que se seleccionaban los casos de uso para cada iteración, se los registraba en una planilla que permitía estimar y controlar cada una de las actividades que se llevarían a cabo para cada caso de uso. Al finalizar la iteración, se documentaban los tiempos de demora que se habían generado, o por el contrario, los tiempos que habían sobrado. Esto permitía obtener información para analizar estadísticamente para futuras mediciones.

Con el objetivo de agrupar las funcionalidades para cada iteración, se priorizaron los Casos de Uso, se especificaron en detalle los seleccionados, y en forma paralela, se iba completando el modelo de datos y el modelo de clases, junto con la interfaz de usuario.

Se desarrollaron prototipos evolutivos de interfaz de usuario, los que se revisaron, modificaron y acordaron con los usuarios. Asimismo, se enriqueció el proceso de documentación, dado que los prototipos se adjuntaron con los documentos correspondientes a cada Caso de Uso.

Para modelar los casos de uso en un segundo nivel de detalle, de modo tal que los programadores tuvieran mayores especificaciones para codificar, se realizaron Diagramas de Secuencia en los que se visualizaba la interacción entre los objetos detectados.

El modelo de Clases se fue definiendo a medida que se confeccionaban los diagramas de secuencia de cada caso de uso analizado, incorporando los nuevos métodos, propiedades, o clases nuevas que surgían.

El proceso de construcción y documentación de la base de datos se realizó de la siguiente manera: el modelo de datos se fue escribiendo directamente en el motor de bases de datos seleccionado y luego se realizaba la importación del modelo a la herramienta Case utilizada para su documentación.

Cada programador debía realizar una prueba de su código antes de habilitar la prueba funcional. Éstas consistirían en las pruebas de unidad, que comenzarían de manera rudimentaria, hasta ser automatizadas con una herramienta.

Para las pruebas funcionales se había definido un rol específico con la conformación del equipo, cuyo responsable debía analizar la especificación funcional de los Casos de Uso y confeccionar los escenarios de prueba.

Se confeccionó un reporte de errores, en la cual se compaginaban en una planilla y se les asignaba responsable de la corrección. De las pruebas funcionales satisfactorias, se pasaba a realizar las pruebas de integración, con la correspondiente generación de lotes de datos y el registro de sus resultados. Para simular datos reales, se generó un importador desde la aplicación anterior, de aquellos datos que podían ser migrados.

Proceso Generado. *Para poder planificar el desarrollo se definían iteraciones pequeñas, acorde al equipo de desarrollo que se encargaría de cada una y se documentaba el esfuerzo y cada una de las tareas propias del desarrollo realizadas. El proceso debía ser claro y definido, para facilitar el control, la documentación y utilizar los recursos eficientemente de un equipo pequeño*.

4.11. Migración de datos

Cuando la aplicación estuvo desarrollada en un 60%, se realizó una migración de los datos del sistema anterior y se lanzo una etapa de prueba, en manos de los usuarios finales. Dado que el desarrollo consistía en reemplazar una aplicación existente y funcionando, se intentó migrar la mayor cantidad de datos posible, para que los usuarios realizaran sus pruebas en un contexto conocido.

La migración se realizó con las herramientas proporcionadas por el motor de Base de datos elegido, y se mantuvo un paralelo con la aplicación anterior para generar la confianza necesaria, dado que este nuevo producto debía venderse en grandes corporaciones, nacionales y del extranjero.

Proceso Generado. *Los usuarios realizaron las pruebas con datos conocidos y tratando de simular situaciones reales, generando mayor confianza y responsabilidad en los mismos.*

4.12. Puesta en marcha

Al efectuarse la puesta en marcha de la aplicación, con la entrega de la primera versión estable al cliente, el equipo completo comenzó a distenderse. Se esperaron dos semanas y se realizó una primera reunión postmortem, en la cual se evaluó el funcionamiento de la aplicación y se extrajeron las lecciones aprendidas de todo el proceso inicial generado, apoyándose en las fortalezas, y discutiendo como mejorar las debilidades.

Proceso Generado. *Análisis postmortem y revisión de las lecciones aprendidas con todo el equipo de desarrollo. Este proceso constituyó la base de conocimiento futura para la mejora de todo el proceso aplicado.*

5. CONCLUSIONES

El caso de estudio presentado, nos ha permitido describir la elaboración de un marco de trabajo inicial, que permitiría la definición y mejora de los procesos en el desarrollo de software de un área de sistemas pequeño que no contaba con ninguna estructura, ni definición de sus procesos, no documentaba, no registraba control de versiones, no priorizaba los requerimientos ni los riesgos, y cuya actividad de desarrollo comenzaba directamente por la codificación.

Las actividades desarrolladas en este caso, presentan la puesta en práctica paulatina de un marco de trabajo inicial, previo a la aplicación de un Modelo de Procesos o de una Norma de Calidad. Dicho marco inicial, genera un primer ordenamiento, para una organización pequeña que se encontraba en un estado caótico en el desarrollo de sus productos, con ausencia de toda estructura organizacional, un proceso indefinido e improvisado, pero certificada bajo una Norma de Calidad Internacional en lo que a los procesos de Gestión de la compañía respecta.

En el marco de trabajo expuesto, confluyen una metodología, un modelo de proceso y una colección de técnicas y herramientas recomendables para las pequeñas y medianas empresas o áreas de desarrollo de software, que facilitan el camino para la implementación de un Modelo de Proceso integral, de mejora continua.

El objetivo para la organización consistió en cambiar la imagen y organizar el trabajo del departamento de sistemas de la compañía, de modo tal que sea valorado ante los demás departamentos, deje de ser considerado un gasto para la empresa y pudiera transformarse en un centro generador de negocios, alineado con el "core business" de la compañía.

Los resultados obtenidos fueron satisfactorios, con la introducción paulatina de un proceso controlable en la organización, modificando la situación cultural y transformando las falencias en los desafíos, donde cada pequeño avance con el tiempo se convertiría en un logro y una mejora. Este proceso paulatino ha brindado una experiencia exitosa, frente a otras experiencias en las cuales se había intentado inducir al proceso de una única vez, escribir todos los documentos desde el inicio, definir las iteraciones y el modelado, todo junto y de manera forzada, a lo cual el grupo de desarrollo no llegó a asimilarlo.

6. REFERENCIAS

[BOO03] Booch, Jacobson & Rumbaugh. (2003). *El lenguaje Unificado de Modelado*. Addison Wesley.

[JAC03] Booch, Jacobson & Rumbaugh. (2003). *El Proceso Unificado de Desarrollo de Software*. Addison Wesley.

[GET02] Getchell, Hargrave, Haynes, Lubrecht, Kazmi, Oikawa, Paschino, Robin, Short. (2002, June). *MSF Process Model v. 3.1*. Retrieved October 10, 2004, from http://www.microsoft.com/msf

[HAY02] Haynes, Robin, Paschino, Short, Kazmi, Oikawa, Getchell, Hargrave, Lubrecht, Huber, Vukcevic, Leocadio, Rocha, Santello, Schimpl. (2002, June). *MSF Team Model v. 3.1*. Retrieved October 10, 2004, from http://www.microsoft.com/msf

[ISO00] ISO 9001:2000 Sistemas de gestión de la calidad - Requisitos.

[KRU00] Kruchten, Philippe, (2000 July); *The Rational Unified Process An Introduction*, Second Edition, Addison Wesley.

[LAR04] Larman, Craig (2004). *Agile & Iterative Development*, A Manager´s Guide, Addison Wesley.

[ROB02] Robin, Preedy, Campbell, Paschino, Hargrave, Born, Huber, Haynes, Kazmi, Oikawa, Getchell. (2002, June). *MSF Risk Management Discipline v. 1.1*. Retrieved October 10, 2004, from http://www.microsoft.com/msf

[ROY70] Royce, W.W. (1970). *Managing the development of large software systems: Concepts and techniques*, in Proceedings of Wescon, August.

[SPA06] Sparx Systems. (2006). *Enterprise Architect User´s Guide*. Retrieved July 7, 2006, from http://www.sparxsystems.com.au.