Convivencia de metodologías: Scrum y RUP en un proyecto de gran escala

Sebastian Cadelli Juan Manuel Fernández

Orientador: Claudia Pons (Facultad de Informática UNLP, CIC y UAI)

Facultad de Informática, Universidad Nacional de La Plata

Buenos Aires, Argentina

jmfernandezinfo@gmail.com
sebaelp18@gmail.com

Resumen. Hoy en día existen diversas alternativas a la hora de elegir una metodología para desarrollar software. Años atrás imperaban las metodologías rígidas, también conocidas como las metodologías tradicionales, en las cuales abunda la documentación, los modelados, las actividades, actores y/o roles. A partir del 2001, surge un punto de inflexión, con el surgimiento de otro tipo de metodologías conocidas como las metodologías ágiles para el desarrollo de software. Estas fueron creadas para ser aplicadas en proyectos pequeños/medianos y que tienen requerimientos volátiles o que cambian con frecuencia; mientras que las metodologías tradicionales, suelen ser más apropiadas para grandes proyectos y donde sus requerimientos son más resistentes a los cambios. A partir de estos conceptos, queremos explicar en esta tesis [1], que si bien no existen recetas magistrales para aplicar metodologías a un determinado proyecto, es posible detectar buenas prácticas y otros patrones que hacen posible decidir acerca de los métodos a utilizar. En particular, a partir de nuestra experiencia de varios años en un proyecto de gran escala vamos a explicar cómo pueden convivir dos tipos de metodologías (Scrum y RUP). Esto es, tomando lo que se sea necesario de cada metodología y adaptándola al proyecto en cuestión.

Keywords. Metodologías de desarrollo de software, -Metodologías tradicionales, Metodologías ágiles, Scrum, RUP, Scrum de Scrum.

1 Planteo del problema

1.1 La crisis del software: necesidad de una metodología

En los inicios de la Informática, el proceso de desarrollo de software era prácticamente artesanal. Luego, debido a la fuerte necesidad de mejorarlo, tuvieron que incorpo-

rarse conceptos y fundamentos de metodologías existentes en otras disciplinas ingenieriles. Así surgieron las metodologías pesadas o tradicionales, las cuales enfatizan la documentación, la planificación y el seguimiento riguroso de las actividades. De esta forma permiten que el desarrollo de software sea más predecible y ordenado. Aunque los modelos generalmente son realistas y están pensados para avanzar de una manera iterativa e incremental, algunas de las características negativas dentro de estos enfoques son los altos costos de implementar cambios y el no ofrecer una buena solución para proyectos que operan en un entorno incierto.

Entre las metodologías tradicionales más populares se encuentran el Rational Unified Process (RUP) [2] o también Microsoft Solutions Framework (MSF), que centran su atención en mantener una documentación exhaustiva del proyecto y cumplir con el plan previsto y definido con precisión en la fase inicial del proyecto.

A partir del año 2001, tras una reunión realizada en los EE.UU., surgió el concepto de ágil. En dicha reunión, (de la cual formaron parte varios expertos de la industria del software) se comenzó a proyectar una alternativa a las metodologías tradicionales de software. El primer avance significativo fue la creación de una organización sin fines de lucro, dedicada a promover conceptos relacionados con el desarrollo ágil, denominada “The Agil Alliance”, la cual creó un documento en donde se sientan las bases, o la “filosofía” que persigue este tipo de metodologías alternativas, es decir, las metodologías ágiles. En contraposición a las tradicionales, las metodologías ágiles se centran en el equipo de desarrollo y sus interacciones, y en la implementación del software por sobre la documentación del mismo.

Entre las metodologías ágiles más aplicadas se encuentran Scrum y eXtreme Programming [3].

1.2 El dilema actual: ¿formalidad o agilidad?

A partir de aquí, surge entonces el debate: ¿bajo qué condiciones es más adecuado utilizar un tipo de metodología u otro? Es decir, en qué tipo de proyecto vamos a obtener resultados más eficientes utilizando una metodología tradicional y en cual va a funcionar de mejor manera una metodología ágil.

Ambos tipos de metodologías atacan el problema de mantener el conocimiento dentro de la organización, pero la más pesada pone el énfasis en la documentación y en los procesos, y la más ágil, en el software funcionando y en las personas.

El uso de metodologías tradicionales ha sido fundamental durante muchos años en la historia del desarrollo de software. Todavía hoy siguen teniendo sentido para sistemas grandes y complejos, especialmente en organizaciones con un alto número de personas involucradas en estos proyectos. Sin embargo hay que tener muy presente que en contextos como el actual, donde hay mucha variabilidad de plataformas, modelos de negocio e incertidumbre en general, es importante ser flexibles y ágiles para llegar adecuadamente con nuestros productos al cada vez más complejo mercado del software.

En base al estudio del estado del arte y en base a nuestra propia experiencia como desarrolladores de software, consideramos que no existe una "receta" magistral que nos indique específicamente cuando se debe utilizar una u otra metodología. Es más, en un mismo proyecto podemos utilizar una combinación de ambas, aprovechando las diferentes ventajas de cada una. Es decir, no son excluyentes. He aquí el motivo de nuestra tesis, de cómo se pueden aplicar conjunta y coherentemente dos metodologías en un mismo proyecto.

2 Objetivos de nuestra tesis

A partir de nuestra participación y experiencia en un proyecto de gran escala, denominado eSidif, nuestra tesis tuvo como objetivo explicar la convivencia de dos tipos de metodologías que tienen distinto enfoque pero que pueden convivir dentro de un proyecto.

Inicialmente el proyecto se regía bajo los principios de la metodología RUP, con el advenimiento del nuevo concepto ágil en el desarrollo de software comenzamos a analizar los beneficios de la inclusión de dicho concepto como parte del proceso. Aquí nos surgió la inquietud de poder consumir las características más beneficiosas de ambas metodologías y adaptarlas en un nuevo proceso de desarrollo donde convivan dichas metodologías con enfoques dispares. Este proceso de fusionar ambas metodologías que parecen opuestas, es lo que se va a explicar en el desarrollo de este documento.

Como consecuencia de la evolución y el crecimiento de nuestro proyecto, la cantidad de recursos fue aumentando paulatinamente dificultando la organización del mismo. Para disminuir esta dificultad decidimos implementar la alternativa "Scrum de Scrum", la cual vamos a explicar de forma detallada, indicando su estructura, organización y comunicación adaptadas a las necesidades de nuestro proyecto.

A través de nuestra tesis realizamos una documentación formal de la adaptación y convivencia de las metodologías identificando las ventajas y/o desventajas de cada una, como así también la ventaja de la utilización de "scrum de scrum" en proyectos de gran tamaño.

Este trabajo formaliza y enriquece a la metodología utilizada en nuestro proyecto laboral, ya que, si bien esta metodología está siendo utilizada de manera eficiente y exitosa, nuestra tesis se encargó de:

- Analizar sus dos visiones complementarias. Es decir, la visión tradicional (como es el caso de RUP), y la visión ágil (como es el caso de Scrum y/o Scrum de Scrum);
- Describir detalladamente cada una de sus componentes, interacciones y comunicaciones;
- Analizar sus ventajas y desventajas;
- Plantear mejoras y trabajos futuros;

- Documentarla de manera formal, fundamentándola en base a la teoría de RUP y Scrum.

3 Nuestro proyecto de referencia: eSidif

El proyecto en cuestión es denominado "eSidif" [4] y su objetivo es la formulación del presupuesto nacional y registro de la ejecución presupuestaria. Este proyecto se realiza en colaboración entre la Universidad Nacional de La Plata (UNLP) y el Estado Nacional. En este contexto, el Laboratorio de Investigación y Formación de Informática Avanzada (LIFIA) se encuentra brindando soporte a la Secretaría de Hacienda del Ministerio de Economía de La Nación.

El 30 de septiembre del año 1992 se sancionó la Ley N°24.156 de Administración Financiera y de los Sistemas de Control del Sector Público Nacional. A partir de allí, se inició el camino en la implementación del proyecto (eSidif) desde sus inicios y variando a través de sus diversas transformaciones, hasta lo que es en la actualidad.

En su versión inicial, el S.I.D.I.F (Sistema Integrado de Información Financiera) fue adoptado como un sistema integrado, que estaba compuesto por varios subsistemas y/o módulos dentro de una visión funcional, que contemplaba la distribución de la base de datos lógica, en una base de datos central y tantas bases institucionales como Servicios de Administración Financiera (SAF) existieran. Esta estructura física era soportada por arquitecturas abiertas y redes locales trabajando en la modalidad "Cliente/Servidor" (C/S).

Bajo este esquema operativo, la Secretaría de Hacienda se comunicaba, a través del SIDIF Central con los sistemas periféricos instalados en los organismos (sistemas locales), con el sistema de gestión para las Unidades Ejecutoras de Préstamos Externos y demás aplicaciones que interactúan con la base de datos central. A través de esta comunicación la base central concentraba el registro de la ejecución presupuestaria. La información de gestión permanecía en las bases locales.

Dentro del S.I.D.I.F. convivían diversos sistemas locales en las distintas entidades con características diferentes. Esta diversidad hizo que se incrementara el costo en mantenimiento y de replicación de las adecuaciones en los sistemas locales, por ende en algunas oportunidades esto recaía en una demora en la disponibilidad.

A partir de los problemas y costos anteriormente mencionados comenzó una etapa de transformación. Es decir, sustituir a los diversos sistemas locales por un único sistema, esto es, a través de la implementación del SLU (S.I.D.I.F Local Unificado).

Este proceso reemplazó los sistemas locales existentes en distintos organismos por la versión unificada con mejoras en cuanto a la provisión de información confiable para la alta gerencia de las entidades, el ajuste de procedimientos de compras, presupuesto, contabilidad y tesorería, y la reducción de aplicativos complementarios requeridos para soportar la gestión.

Luego de la primera transformación, en la cual se introdujeron importantes mejoras en la gestión a través del SLU, se planteó una segunda transformación, en la cual se hizo hincapié en la extensión funcional y en la renovación tecnológica. Esta segunda transformación fue posible con el sistema "eSidif". Este sistema busca mejorar la calidad del S.I.D.I.F a través de la incorporación de innovaciones tecnológicas de las comunicaciones, que se han producido en la última década.

4 Nuestra propuesta de proceso de desarrollo combinado

Como dijimos anteriormente, la metodología que se utiliza en el proyecto es una convivencia de metodología ágil, en este caso Scrum, con una metodología tradicional como RUP.

Por un lado, Scrum se utiliza para gestionar los procesos del proyecto. Esta metodología se utiliza para definir un marco de trabajo para la gestión y el desarrollo de software basándose en un proceso iterativo e incremental. En nuestro caso, la experiencia a partir del uso de esta metodología de desarrollo produjo óptimos resultados, ya que, en proyectos de gran tamaño, los cambios de requerimientos son bastante frecuentes, y de esta forma adoptando esta metodología ágil, adaptarse a los cambios y/o prioridades es relativamente accesible.

Otros aspectos ventajosos de la utilización de Scrum, es la participación de todo el equipo en la toma de las decisiones más significativas del proceso, por ejemplo, en la estimación de las tareas a desarrollar durante el sprint. Otra de las ventajas es la posibilidad que brinda a los desarrolladores para que puedan adquirir un amplio y diverso conocimiento a través de la rotación de tareas.

Para toda la etapa de análisis del proyecto se utiliza una metodología tradicional, más específicamente RUP. El equipo del área de análisis se reúne con el representante de los usuarios del sistema, en encuentros denominados "Talleres". En estos talleres el usuario realiza una presentación de la funcionalidad deseada. Esta funcionalidad habitualmente es un pedido que consiste en la funcionalidad que utilizan en el sistema anterior, más algunas mejoras que puedan surgir. Los requerimientos son plasmados en distintos artefactos generados por los analistas, como por ejemplo: prototipos de interfaz de usuario (PIUs), validaciones, ejemplos de uso, etc. Estos artefactos son presentados al usuario en un taller posterior y en el cual dicho usuario puede realizar algún tipo de modificación que es ajustada por los analistas. Esto permite un buen intercambio de información a la hora de realizar la solicitud y la propuesta de los requerimientos. Una vez que la presentación es aceptada por el usuario, los analistas pueden pasar a la siguiente etapa donde se plasman los requerimientos obtenidos y ajustados, en los artefactos que posteriormente van a consumir desde el área de diseño y desarrollo (DyD).

Con los artefactos nos referimos a documentar en el EA (Enterprise Architecture), los requerimientos obtenidos en los talleres, mediante el Modelo de Casos de Uso (MCU),Diagramas de Transición de Estado (DTE), Modelo de Clases de Análisis (MCA), Modelo de Reglas de Negocio (MRN), Prototipos de Interfaz de Usuario (PIU), etc.

El área de Análisis genera una especificación de requerimientos (como dijimos anteriormente en los encuentros denominados “talleres”) a través de la creación de una serie de artefactos que son tomados por el área de Diseño. Estos artefactos son los siguientes:

- Modelo de Casos de Uso (MCU)
- Modelo de Clases de Análisis (MCA)
- Modelo de Reglas de Negocio (MRN)
- Prototipos de Interfaz de Usuario (PIU)
- Diagrama de Transición de Estados (DTE)

A partir de los artefactos anteriormente mencionados, se analiza e interpreta esa documentación y se toman las decisiones de diseño y de implementación a nivel de equipo.

Aquí es el punto donde se observa claramente la convivencia de ambas metodologías. Por un lado desde el área de análisis, los cuales se reúnen con los clientes en los denominados talleres, con la finalidad de generar la documentación para plasmarla en los artefactos correspondientes (esto es siguiendo los principios de RUP). Los artefactos luego son consumidos por los desarrolladores, utilizados al momento de realizar la planificación, tienen su comunicación (a través de las diversas reuniones que nombramos a lo largo del trabajo) y se organizan a su manera en sus respectivos equipos; esto ocurre bajo la línea de Scrum. Lo mismo sucede en el área de Testing con su propia planificación y comunicación como en el caso de los desarrolladores.

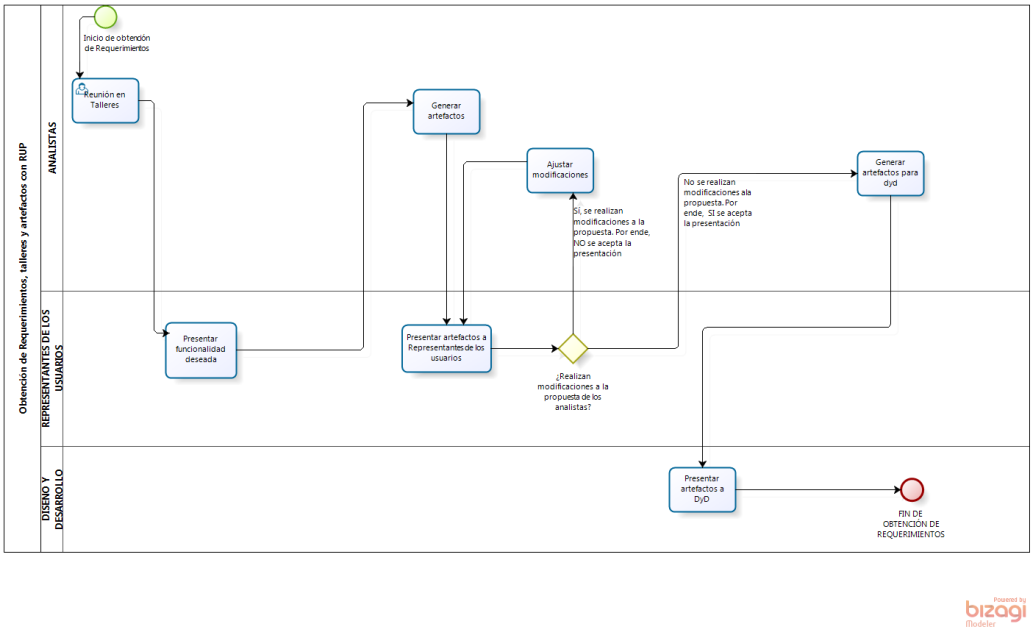

Figura 1 – Proceso de obtención de requerimientos, RUP

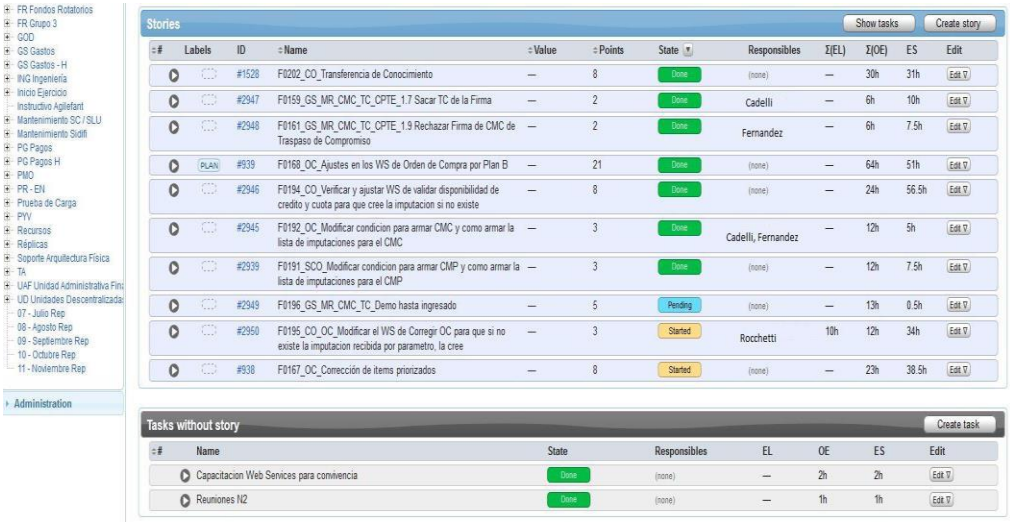

Figura 2 – Ejemplo de product backlog en Agilefant, Scrum

4.1 Scrum de Scrum

Se ha dicho que los métodos de desarrollo de software ágil como Scrum, son ideales para proyectos desarrollados por equipos multidisciplinarios de pocas personas y localizados en la misma oficina. Sin embargo, cada vez se observan más ejemplos de grandes compañías utilizando Scrum en proyectos de gran escala, con equipos de proyectos integrados por decenas y hasta cientos de desarrolladores. La técnica de las reuniones Scrum de Scrum es la que permite escalar el enfoque Scrum para grandes equipos de proyecto a escala corporativa. Esta técnica consiste en dividir un equipo de muchas personas en diferentes equipos Scrum, para luego utilizar reuniones Scrum de Scrum para coordinarlos. A cada reunión de Scrum de Scrum asiste uno o dos integrantes de cada equipo.

“Scrum de Scrum” aplicado al proyecto:

Como ya se explicó anteriormente, nuestro proyecto de referencia tiene una magnitud muy importante, tanto en desarrollo como en cantidad de participantes. La alternativa de metodología que se pensó para llevar a cabo el sistema, fue la adaptación de “Scrum de Scrum”, que permite mantener el concepto de desarrollo ágil sin perder de vista la envergadura del proyecto.

En dicho proyecto trabajan más de 10 equipos, cada uno de estos formados por las respectivas áreas de Análisis (ANA), Diseño/Desarrollo (DyD) y Testing (TST).

La primera gran planificación que se realiza es a nivel gerencial, donde se acuerdan todos los objetivos esperados al momento de realizar el despliegue a producción. Habitualmente esta planificación se realiza a principio del año calendario.

Desde una visión global se realiza una segunda planificación a nivel de los objetivos que posee cada equipo para una determinada fecha de entrega (habitualmente

cada 3 semanas), teniendo en cuenta la cantidad de recursos que posee el equipo, las horas laborales, la complejidad de las tareas, etc. Este es un Scrum a gran escala del proyecto.

Por otro lado, cada equipo tiene sus "áreas" (DyD, ANA, TST), y cada una realiza la planificación de un Sprint por separado. Es decir, el área de DyD, planifica un sprint con los objetivos pensados en la "planificación superior", en forma de tareas a través del product backlog y donde las mismas son estimadas en horas y luego son tomadas por los recursos del equipo los cuales registran sus horas de desarrollo en cada una. Lo mismo realiza el área de TST. Esto representa el Scrum interno de cada área.

En la ***figura 3*** podemos visualizar la estructura y organización de los equipos utilizando Scrum de scrum.

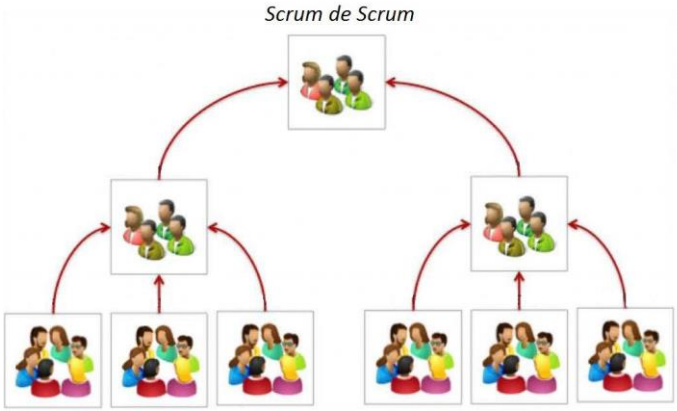

Figura 3 – Organización y estructura de Scrum de Scrum

4.2 Reuniones de coordinación

La coordinación del proyecto se lleva a cabo mediante distintos niveles de reuniones, cada una de las cuales tiene distintos objetivos, alcances y están conformadas por diferentes participantes. A continuación se detallan dichas reuniones:

Reuniones N1 (nivel 1, de priorización):

Objetivos
- Las reuniones conocidas como N1 se utilizan para realizar la priorización de defectos.
- En forma periódica (semanalmente por defecto) se llevan adelante las reuniones N1, en las cuales se presentan los defectos no priorizados. Si como parte de la priorización se detectan defectos de alta criticidad se podría tener que actualizar el sprint backlog para poder desarrollar la solución a dicho defecto de la forma más inmediata posible.

- Otros de los objetivos de este nivel es plantear las dudas funcionales que pudieron surgir en el Sprint.

Precondiciones
- Contar con la lista de defectos a priorizar.
- Contar con un detalle de todos los incidentes e impedimentos que ponen en riesgo el sprint.

Participantes
- Referentes por área, es decir, Diseño y Desarrollo, Testing, Análisis.
- Réplicas, que hacen de intermediario representando al usuario.

Reuniones N2 (nivel 2):

Objetivos
- El objetivo de las reuniones N2 es reportar el avance del negocio al PMO a cargo.
- También se realiza la sincronización necesaria entre las diferentes áreas.
- Tiene una frecuencia semanal.
- Se tratan también temas sobre recursos, es decir, vacaciones de los mismos, ingresos, egresos.

Precondiciones
Para esta reunión se deben completar y actualizar una serie de artefactos que comprenden:
- Informe de Avance.
- Burndown Chart del Sprint en curso.
    - Es un gráfico que da una visión "general" del progreso del proyecto. Permite al equipo visualizar la velocidad de avance del proyecto. Este gráfico se va actualizando a medida que se van cumpliendo y realizando las tareas correspondientes al product backlog.

- Product Burndown Chart.
    - Idem al Burndown Chart pero con la vision del producto completo.
- Defectos.
- Presentación de estado de defectos a cargo del equipo de Testing.
- Cambios de alcance o redefinición de requerimientos.
- Ítems pendientes de Arquitectura.
- Alertas N1.
- Incidentes.
- Riesgos.
- Novedades de Recursos.

Participantes
- PMO encargado del seguimiento externo.
- Analistas.

- Testing reporta datos estadísticos, por ejemplo cantidad de ítems reingresados, cantidad de ítems corregidos, etc.

Reuniones N1 de PMO (Project Management Office):

Objetivos
El objetivo de las reuniones N1 de PMO es poner al tanto el estado de avance de los distintos equipos.

Precondiciones
- Se sincronizan los objetivos comunes.
- Dependencia funcional con otro equipo.

Participantes
- PMO de cada equipo

Cabe aclarar que se realizan reuniones de N3 pero en este caso los PMO participan de forma no presencial a través de un avance que envían en un formulario estándar donde se visualiza el avance del proyecto

5 Trabajos relacionados

A continuación se describen documentos que están relacionados con esta tesis:

– RUP versus Scrum: una comparación empírica en un ámbito académico [5]. Este documento realiza una comparación empírica entre RUP y Scrum. Esta se basa en un experimento formal que se desarrolló en el contexto de un taller de diseño de 4to año, que forma parte de la carrera de Ingeniería en Informática correspondiente a la Facultad de Ingeniería de la Universidad Austral. En este experimento, se divide al curso de alumnos en dos grupos: un grupo el cual trabajará con Scrum y un grupo que trabajará con RUP. Ambos grupos trabajarán en forma independiente en el desarrollo de un juego de estrategia por turnos partiendo de una misma definición de requerimientos. A partir de esta división ambos grupos trabajan en su objetivo y se realiza el experimento en base a los resultados que obtienen ambos grupos, realizando así la comparación entre RUP y Scrum.

– Tesis de grado PDSM: Proceso de Desarrollo de Software Mixto, combinando RUP y Scrum [6]. En esta tesis se realiza una combinación de metodologías para generar un nuevo proceso de desarrollo de software que a diferencia de nuestra propuesta de desarrollo combinado en proyectos a gran escala en este caso se adapta de mejor forma a proyectos pequeños como medianos y proyectos nuevos y de mejoras y mantenimientos.

– Gestión de Proyectos: ¿formal o ágil?[7]. Es un artículo que presenta distintas clasificaciones de la gestión de proyectos así como también sus características. Por otra parte, presenta premisas que indican cuándo y cómo aplicar un estilo de gestión para determinado proyecto, dependiendo de las características del mismo y de la organización subyacente.
– CoMakeIT : White Paper on the Agile process: RUP y Scrum[8]: Es un documento que explica las metodologías RUP y Scrum por separado, como también pueden acoplarse y convivir ambas en un proyecto de pequeña/mediana escala.

Los documentos mencionados y explicados anteriormente están relacionados con nuestra tesis en cuanto a los temas centrales de los mismos que son las metodologías de desarrollo Scrum y RUP. Por otra parte, nuestro artículo realiza una ampliación de estos documentos explicando, además de las metodologías por separado, la posibilidad de convivencia de ambas, centrándonos en la aplicación de dicha convivencia en un proyecto de gran escala y aplicando un concepto importante como es el de scrum de scrum.

6 Conclusiones

Como conclusión general podemos decir que no existen recetas para aplicar metodologías de desarrollo de software a los proyectos. Cada metodología posee sus principios, y por ende sus ventajas y/o desventajas. Por eso, es importante tener en cuenta qué aspectos se van a necesitar de una metodología y de esta forma aplicarla a un proyecto tomando las cosas que sean necesarias para el mismo y adaptándolas.

Por otra parte, es importante resaltar que dos metodologías distintas y con distintos principios pueden convivir, como es el caso del proyecto que presentamos, donde fusionamos las características más convenientes de cada una para sacar el mayor provecho de las mismas y optimizar un nuevo proceso de desarrollo de software combinado.

Con respecto a las metodologías podemos mencionar varias ventajas y utilidades que encontramos con su adaptación y utilización:

- Scrum de Scrum nos facilitó la organización del proyecto teniendo en cuenta la gran cantidad de participantes. Nos permitió una fluidez en la comunicación entre los integrantes del equipo, y con otros equipos, que es muy necesario para mantener un nivel de desarrollo estable.

- Diversidad de aprendizaje a la hora del desarrollo, teniendo en cuenta que la metodología permite que los distintos recursos puedan ir tomando distintas tareas con distintas responsabilidades en un mismo proyecto.

- Encontramos la posibilidad de ir rotando en los roles que cumple un integrante dentro del proyecto. Por ejemplo, un integrante puede cumplir tareas de desarrollador, luego formar parte del equipo de análisis, involucrarse como responsable téc-

nico del equipo, etc. Esto también tiene como finalidad el crecimiento personal en la adaptación a distintos grupos, con diversas personalidades bajo diferentes responsabilidades.

- Se produce un importante intercambio de información entre los participantes a través de las distintas reuniones que fueron explicadas en detalle con anterioridad (retrospectivas, N1, N2, planning, daily). Lo que permite resolver problemas rápidamente, tener distintos puntos de vista o visualizar aspectos que uno no tuvo en cuenta para determinada tarea.

- La planificación a corto, mediano y largo plazo. Esto es un gran aprendizaje, y una experiencia muy necesaria, ya que por la magnitud del sistema es fundamental realizar distintos niveles de planificación, siendo una de las tareas más complicadas a la hora de llevar a cabo un proyecto.

Por otra parte, como desventajas o aspectos a tener en cuenta con la utilización de las metodologías podemos mencionar:

- Sincronización entre los equipos con el fin de llegar a un hito previamente comprometido. Esta es una tarea muy importante a tener en cuenta, ya que la magnitud del proyecto obliga a minimizar las distintas dificultades que pueden tener un impacto directo en el tiempo de entrega de una funcionalidad previamente acordada.

- Sobre-estimación /sub-estimación de tiempo de tareas. Es otra dificultad en la que se debe prestar mayor atención, ya que el recambio de participantes es superior en relación a un proyecto pequeño, por lo tanto la estimación de las tareas debe adaptarse tanto a los recursos con mayor experiencia como a aquellos recursos nuevos, teniendo en cuenta que dicha estimación tiene un impacto directo en la fecha de entrega acordada.

Por otra parte podemos decir que a pesar de que RUP es una metodología tradicional la cual centra su importancia en la documentación exhaustiva y a veces excesiva, en esta convivencia de metodologías resulta útil poder documentar determinadas actividades. En nuestro caso - en el proyecto que presentamos y explicamos durante la tesis - la metodología documenta artefactos que son necesarios y útiles para distintos roles aplicados en diferentes etapas. Por ejemplo, el MCA resulta útil para que el desarrollador pueda comprender los requerimientos que los analistas plasman en dicho modelo y de esta manera poder llevar a cabo la implementación de lo deseado. Esto resulta importante ya que, por un lado, el desarrollador puede interpretar los requerimientos descriptos por el analista de una forma más abstracta y posteriormente volcarlo, primeramente al diagrama de objetos o de clases, y luego al código. Por otra parte, el desarrollador tiene la libertad y el poder de decisión de ver cómo va a implementar lo propuesto por análisis, es decir, que hay una cierta flexibilidad para cumplir con los requerimientos presentados en el MCA.

Otra lección que aprendimos en este trabajo fue valorar la importancia de la comunicación entre todos los participantes, permitiendo la organización de los equipos de

forma distribuida a través de las distintas reuniones, obteniendo varios puntos de comunicación con diferentes enfoques.

Otro punto destacable es que en algunos casos no es necesario abrumar con tanta documentación, la cual es difícil de mantener actualizada en un proyecto de tal magnitud, sino contar con la documentación más útil para los integrantes del sistema.

7 Trabajos Futuros

A partir de nuestro análisis y comparación de metodologías con distintos principios, como son Scrum y RUP, de la convivencia de las mismas y de la aplicación de Scrum de Scrum en el proyecto, proponemos una tarea similar pero con respecto a otra técnica que nos resultó interesante en la actualidad, y que se postula para ser de gran utilidad. Esta técnica es la denominada "Kanban". "El término kanban aplicado a la gestión ágil de proyectos se refiere a técnicas de representación visual de la información para mejorar la eficiencia en la ejecución de las tareas de un proyecto."[9]

Entonces planteamos como posibles trabajos futuros; análisis y desarrollo de proyectos que utilicen las nuevas alternativas de gestión ágiles como por ejemplo Kanban y su fusión con Scrum denominada "ScrumBan". Definiendo ventajas y desventajas, detallando qué aspectos de Scrum son comunes con Kanban y que otros aspectos particulares agrega Kanban a la gestión.

También puede resultar de interés realizar una comparación detallada de Kanban con las diferentes alternativas de gestión ágiles más utilizadas en la actualidad, como pueden ser, XP [10], Lean; entre otras.

8 Referencias bibliográficas

1. Juan Manuel Fernández, Sebastián Cadelli. Convivencia de metodologías: Scrum y Rup en un proyecto de gran escala. Tesis de Licenciatura en Informática. UNLP. Fecha de presentación. Julio 2014.
2. Philippe Kruchten, *The Rational Unified Process: An Introduction* - Addison Wesley (2004).
3. José H. Canós, Patricio Letelier y Mª Carmen Penadés, *Metodologías Ágiles en el Desarrollo de Software* DSIC. Universidad Politécnica de Valencia, Camino de Vera s/n, 46022 Valencia.
4. Portal Web del Laboratorio de Investigación y Formación en Informática Avanzada. URL:http://lifia.info.unlp.edu.ar/es/hacienda.htm. Fecha de acceso: 15/04/2014.
5. Santiago D'Andre, Miguel Martinez Soler, Gabriela Robiolo, *RUP versus Scrum: una comparación empírica en un ámbito académico*. Universidad Austral (39JAIIO- ASSE 2010).
6. Mariani Maria Florencia, Okabe Evangelina. PDSM: Desarrollo de Software Mixto. - Tesis de grado en Licenciatura en Informática, Facultad de Informática, UNLP (Universidad Nacional de La Plata). Directora: Dra. Claudia Pons. Noviembre 2010.

7. Juan Palacio Gestión de Proyectos: ¿formal o ágil?-. Edición año 2006. URL de acceso: http://www.navegapolis.net/files/s/NST-004_01.pdf. Fecha de acceso: 02/03/2014.
8. White Paper on the Agile Process: RUP and Scrum, CoMakeIT, Publication Date: October 10, 2007. URL de acceso :http://www.comakeit.com/download/agile_process_rup_scrum.pdf. Fecha de acceso: 30/03/2014.
9. Javier Garzás, Juan Enríquez S., Emmanuel Irrazábal. Gestión Ágil de Proyectos Software- – Edición año 2012. URL de acceso: http://www.javiergarzas.com/2011/11/kanban.html. Fecha de acceso: 10/04/2014
10. Henrik Kniberg. Scrum and XP from the Trenches -. Edición año 2007.