Universidad de las Ciencias Informáticas

FACULTAD 10

MA-GMPR-UR2

Metodología ágil para proyectos de software libre.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autora

Gladys Marsi Peñalver Romero

Tutor

Ing. Abel Meneses Abad

Ciudad de La Habana

Junio de 2008

“El software que funciona es más importante que la documentación exhaustiva”.

Manifiesto Ágil.

DECLARACIÓN DE AUTORÍA

Declaro ser autora de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter no exclusivo.

Para que así conste firmo la presente a los ___ días del mes de _____ del año _____.

_____ _____

Gladys Marsi Peñalver Romero **Ing. Abel Meneses Abad**

DATOS DE CONTACTO

Nombre y Apellidos del **Tutor**: Abel Meneses Abad.

Email: abelma@uci.cu

Curriculum: Profesor. Ing. en Telecomunicaciones y Electrónica, 2004, CUJAE. Imparte asignaturas como Sistemas Operativos, Teleinformática II y Practica Profesional; de las asignaturas del 2do perfil de software libre: GNU/Linux Básico, Herramientas para el Trabajo Colaborativo, Programación en Perl, GNU/Linux nivel Medio; ha impartido postgrados de GNU/Linux nivel Básico y Programación Web. Posee categoría docente de Instructor; ha cursado postgrados como: Ciencia, Tecnología y Sociedad, Gnu/Linux Básico, Ideóloga y Política de la Revolución Cubana, Fundamentos de la Docencia Universitaria, Aplicación de las TIC al Proceso de Enseñanza y Aprendizaje, Metodología de la Investigación Científica. Ha presentado ponencias en eventos y forma parte del grupo de investigaciones de Migración a Software Libre de la UCI y del Grupo Técnico Nacional. Es líder del Proyecto Unicornios (Servicios Especializados para la Migración a SWL de la UCI). Miembro organizador del Taller de Software Libre de Informática Habana. Es conferencista de software libre, y ha atendido personalmente la capacitación y la migración en diferentes centros del país. Patrocina y dirige la Revista UXI de Software Libre, y desarrolla investigaciones sobre la historia de esta corriente tecnológica en Cuba. Actualmente se desempeña como Asesor de Investigaciones del Vicedecano de Producción de la Facultad X.

Nombre y Apellidos del **Co-Tutor**: Sergio Jesús García De La Puente.

Email: sgarcia@uci.cu

Curriculum: Profesor Ingeniero en Ciencias Informáticas, 2007, Universidad de las Ciencias Informáticas (UCI). Experiencia docente impartiendo asignaturas tales como Ingeniería de Software y Práctica Profesional: en la asignatura de Macromedia Flash MX 2004. Actualmente está en Adiestramiento; ha cursado postgrados de: Ciencia, Tecnología y Sociedad; Idioma Extranjero: Inglés, nivel básico; Fundamentos de la Docencia Universitaria, Ideología y Política de la Revolución Cubana. Actualmente se encuentra cursando el Diplomado de Software Libre. Es analista de sistema en el proyecto Unicornios, de la Facultad 10.

Nombre y Apellidos de la **Asesora**: Yenisleydi Cariaga Cristo.
Email: yeni@uci.cu.

Curriculum: Graduada de Licenciatura en Sociología en la Universidad de la Habana en el año 2004. Profesora de la Universidad de las Ciencias Informáticas desde su egreso. Ha sido tutora, consultora y co-tutora de tesis durante los últimos 2 cursos de la UCI en temas de software libres. Ha participado en eventos como Forum de Ciencia y Técnica, UCIENCIA, Informática 2007 y Universidad 2008. Opta actualmente por la categoría docente de asistente y es maestrante en Estudios sociales de Ciencia y tecnología. Se ha vinculado al proyecto productivo Unicornio, soporte e inmigración, de la facultad 10, en el rol de Jefe de Capacitación. Ha impartido cursos de postgrado de Metodología de Investigación Científica a profesores de la universidad y conferencias relacionados con el software libre en institutos fuera de la universidad. Fue Sello Forjadores del Futuro a Jóvenes Investigadores de las BTJ en el año 2007. Es Vicedecana de Extensión Universitaria de la Facultad10.

AGRADECIMIENTOS

Agradezco los esfuerzos de todos aquellos que hicieron posible que lograra todo lo que hasta hoy he alcanzado, a esta hermosa y bella Revolución por hacer posible todos mis sueños y a Fidel Castro, fundador de la Universidad de las Ciencias Informáticas.

Agradezco a mi tutor, Abel Meneses Abad, por apoyarme, guiarme y confiar en mí durante mi largo camino de estudiante. Eres una persona maravillosa, no cambies nunca.

Agradezco a mi familia, por todo el apoyo que me han brindado en todos estos años, y en especial a mis padres Maritza Romero Machado y Leonardo Peñalver Téllez.

Agradezco a Marcos Luis por mantener siempre mi pensamiento y mi corazón amado.

A Nancy de los Ángeles, por ser mi ángel de la guarda.

Agradezco a mi segunda familia, mi tío Jorge Luis, papá Luis Raciel y mami Yolanda Zulaima por darme tanto amor, siempre estarán conmigo en mi corazón.

Agradezco a todos mis amigos y compañeros que me han acompañado a lo largo de estos 5 años de mi carrera.

A todos muchas gracias.

DEDICATORIA

A mi padre, por ser el faro que ilumina mi vida.
A mi madre, diamante del cofre de mi corazón.
A mis abuelitos que son mi razón de ser.
A mi tía Maricela, por estar siempre presente.
A mi hermano Leosimar, por darme la confianza en mí misma para siempre seguir adelante.
A mi hermano Yoikel, por dejarme ser parte de su vida, a pesar de la distancia.

RESUMEN

En el 2007 se realizó en la Facultad 7, un estudio sobre las metodologías ágiles, con el objetivo de confeccionar una propuesta adaptable a sus proyectos.

La tendencia hoy en día, es obtener productos de software en el menor tiempo posible y elaborar la documentación necesaria. Por lo que proponer una metodología de procedimientos ágiles para el proceso de producción de software en el grupo UNICORNIOS de la Facultad 10, perteneciente al Polo de Software Libre es el objetivo de este trabajo.

En la investigación se realiza un estudio del estado de las metodologías SCRUM y XP, la concepción del proyecto UNICORNIOS y a cuáles proyectos se les va a aplicar la propuesta realizada. Por lo que surge la necesidad, de realizar un estudio de las características de los proyectos de este grupo y a cuáles de ellos se le aplicará la propuesta.

Este estudio ofrece una estrategia tecnológica, a partir de la inserción de procedimientos ágiles que permitan actualizar los procesos de software para el mejoramiento de la producción, pues fomenta el desarrollo de la mentalidad al trabajo con una metodología totalmente nueva, aumenta el nivel de interés del equipo y el líder del proyecto tiene mejor control del mismo.

Palabras claves: Manifiesto Ágil, Métodos Ágiles, Metodologías Ágiles, Proceso de Software, Proyectos de Desarrollo.

TABLA DE CONTENIDOS

INTRODUCCIÓN

En cada uno de los procesos llevados a cabo para desarrollar un software se debe tener en cuenta que para mejores resultados se tiene que aplicar el uso de una metodología, facilitando una mejor organización y satisfacciones por parte de los desarrolladores y de los clientes que esperan su producto.

Una vez definida la necesidad de una metodología es muy difícil definir cuáles de las existentes para el desarrollo de software se utilizará para el trabajo. Hay que centrarse en las necesidades y solicitudes del cliente para de esta forma analizar los artefactos que se deben producir, las notaciones que se usarán, las herramientas y especialmente el control del proceso en el que se participa.

El factor humano es el aspecto más importante a tener en cuenta, si de desarrollo se trata, darle mayor valor al individuo, a la colaboración con el cliente, al trabajo colaborativo, y a obtener resultados en un período de tiempo más corto y en menos iteraciones es a lo que hacen referencia las metodologías ágiles, que son un tanto desconocidas para todos en la actualidad. Es aún de total desconocimiento para todos si de está temática se trata, pero eso no significa que no se hayan hecho estudios y hasta propuestas.

En la Universidad de las Ciencias Informáticas se desarrollan diversos proyectos productivos que se dedican a la producción de software libre, específicamente en la Facultad 10, y en particular el proyecto UNICORNIOS. En este proyecto se emplean metodologías de desarrollo pesadas, lo que ha provocado el desarrollo de software sin documentación, o con documentación a medias y en general un menor número de desarrollos para el tiempo de ejecución que tiene el grupo de trabajo.

Esto evidencia que no es factible la utilización de este tipo de metodología en el polo de software libre, pues los cambios constantes y no previstos en la modelación del sistema a implementar, la entrega tardía de las versiones de prueba de los sistemas, el exceso de tiempo de desarrollo de las aplicaciones, entre otros motivos traen como consecuencia el fracaso de los proyectos.

Atendiendo a los elementos antes planteados, para darle solución a la **situación problémica** descrita anteriormente, se plantea como **problema** de investigación, ¿Cómo podría la propuesta Metodología ágil Malay Rodríguez Villar revisión 1 (MA-MRV-R1) ser aplicable con resultados positivos en proyectos de software libre, en particular el proyecto

UNICORNIOS? Partiendo de la **idea** de que si se introduce un procedimiento ágil bien definido en el proceso de desarrollo de software se pueden lograr proyectos más eficientes dentro del grupo de proyecto UNICORNIOS. Siendo el **objeto de estudio** el proceso de producción de software del proyecto UNICORNIOS en la Facultad X de la Universidad de las Ciencias Informáticas.

El campo de acción que abarca la investigación es la utilización de procedimientos ágiles en el proceso de producción de software en el proyecto productivo UNICORNIOS. Para la introducción y posterior utilización de dichos procedimientos en el proceso productivo se trazo el siguiente objetivo general: Aplicar y analizar los resultados de la solución de MA-MRV-R1 en el proceso de producción de software en proyectos de software libre, específicamente en el grupo UNICORNIOS de la Facultad X de la Universidad de las Ciencias Informáticas.

Como objetivos específicos se definieron los siguientes:

- Seleccionar los proyectos productivos a los cuales se les va a aplicar la propuesta MA-MRV-R1 para evaluar su viabilidad.
- Analizar las estadísticas para determinar los elementos necesarios para la Metodología ágil Gladys Marsi Peñalver Romero UNICORNIOS revisión 2 (MA-GMPR-UR2) que se quiere implementar.
- Confeccionar la documentación necesaria para la aplicación de la solución MA-GMPR-UR2 en proyectos de Software Libre, según sus características.

Se pretende tener como **posible resultado** la inserción de un procedimiento ágil para el proceso de desarrollo de software de la Facultad X, específicamente en el proyecto UNICORNIOS. Para alcanzar él mismo se deben cumplir los objetivos y se hace necesario desarrollar las siguientes **tareas**:

- Seleccionar proyectos productivos de UNICORNIOS que presenten diferentes características como muestra para poner en práctica la propuesta MA-MRV-R1.
- Aplicar en el proyecto UNICORNIOS la solución MA-MRV-R1.
- Capacitación para la aplicación de la propuesta MA-MRV-R1.
- Revisar la documentación llenada por cada uno de los sistemas donde se aplica MA-MRV-R1 al finalizar cada iteración durante 3 meses.

- Consultar los resultados obtenidos en cada una de las revisiones hechas en cada iteración. Seleccionar los procedimientos adecuados según la clasificación de los proyectos.
- Confeccionar la metodología MA-GMPR-UR2.

La investigación está sustentada en 3 capítulos, estructura que se describe a continuación:

Capítulo 1: Se realiza una pequeña reseña del surgimiento de la propuesta MA-MRV-R1; el estado actual de las metodologías utilizadas (XP y SCRUM); concepción del grupo UNICORNIOS y los tipos de proyectos donde la propuesta se aplica en este grupo.

Capítulo 2: Estudio de los proyectos de igual tipo escogidos en el Capítulo 1 donde se aplican estas metodologías en el mundo y la documentación de todas las revisiones hechas en el proyecto con la puesta en práctica de la propuesta MA-MRV-R1.

Capitulo 3: Metodología MA-GMPR-UR2.

CAPÍTULO 1 FUNDAMENTACIÓN TEÓRICA

La evolución de la disciplina de ingeniería de software ha traído consigo propuestas diferentes para mejorar los resultados del proceso de construcción. Las metodologías tradicionales haciendo énfasis en la plantación, y las metodologías ágiles centrándose en la adaptabilidad del proceso, delinean las principales propuestas presentes en la literatura. De manera paralela, el tema de modelos para el mejoramiento de los procesos de desarrollo ocupa un lugar importante en la búsqueda de la metodología adecuada para producir software de calidad en cualquier contexto de desarrollo.

Grupos de desarrollo han experimentado soluciones que basan su fundamento en la adaptabilidad de los procesos de desarrollo, en lugar de seguir esperando lograr resultados predecibles de un proceso que no evoluciona. Esta comunidad de desarrolladores e investigadores han nombrado su trabajo bajo lo que conocemos como metodologías ágiles. Las metodologías ágiles como puede entenderse mal, no están en contra de administrar procesos de desarrollo. Por el contrario promueve la formalización de procesos adaptables.

El avance de los proyectos de software es algo que siempre ha preocupado a todos los implicados en los mismos. Es algo sobre lo que los clientes centran su interés y algo que, como gestores de proyectos, se necesita comunicar. Tradicionalmente se ha abordado esta necesidad de mostrar el progreso mediante el uso de diferentes documentos o artefactos. Todos los que han tenido que mantener actualizado “El DotProject” del proyecto saben lo difícil que es esto. Es tan difícil, que rara vez se hace con la disciplina que requiere, de tal modo que habitualmente este tipo de aproximación no proporciona los resultados esperados. [1]

El enfoque tradicional de mostrar el avance de los proyectos mediante documentos es algo que no funciona bien. Nuestros clientes han descubierto que los documentos rara vez muestran el avance real de un proyecto. Es muy posible haber trabajado mucho y tener una gran cantidad de documentación sobre un proyecto y estar a años luz de lo que – quien financia el proyecto – pueda obtener valor. ¿Quién no conoce algún proyecto en el que tras muchos meses de desarrollo lo único que había es un montón de documentos? Los documentos por sí mismos no aportan ningún retorno de la inversión. No se puede hacer nada para obtener valor para tu negocio solamente con la documentación relacionada con un proyecto de software. Solo el software que pueden ejecutar y utilizar es susceptible de crear valor para nuestros clientes. Solo el software que funciona debe ser la medida del

progreso de los proyectos de desarrollo. Aceptar esto obliga a asumir que se tiene que entregar software con frecuencia – y no documentación – a nuestros clientes.

Hay que ser interesados con el esfuerzo que se pone en la documentación; si no, se corre el riesgo de ver que todos aquellos requisitos, por poner un ejemplo, que tan detalladamente documentamos sobre el sistema de gestión que nuestro cliente quiere, son papel mojado porque han comprado una nueva unidad de negocios. Y ya se ha hecho un gasto del que difícilmente se obtiene algún retorno. La documentación en los proyectos de software pierde su relevancia y se queda obsoleta rápidamente, haciendo que mantenerla actualizada sea muy costoso. Un ejemplo práctico de lo que se plantea fue el caso de Joctave (Front End para el asistente matemático Octave creado en el Grupo UNICORNIOS), durante el estudio del arte se encontraron proyectos similares que no tenían la fuerza y calidad suficientes para los requerimientos que se necesitaban; se contaba con una ventaja competitiva que duró seis meses tras los cuales uno de los proyectos estudiados alcanzó un grado de madurez tal como para ocasionar la congelación de la idea. El equipo debió variar su trabajo y emplear el esfuerzo en otros proyectos.

A menudo se comete el error de tratar de sustituir la comunicación fluida por documentación, y cuando se hace esto, se introducen costes e inflexibilidades al proceso de desarrollo realizado. Cuando un desarrollador encuentra una línea en la documentación que no es correcta o no está actualizada rápidamente pierde la confianza y vuelve su vista a la única fuente de verdad absoluta: el código fuente. Esto lleva a la situación de que solo la documentación que se genera directamente desde el código fuente de manera automatizada tiene verdadero valor.

El código es la única fuente de verdad absoluta sobre un proyecto de software a nivel de detalle, y el nivel de detalle es el único útil para modificar, extender o mantener un sistema en producción. Mucha documentación puede emanar de una serie de patrones que se repiten a lo largo del proyecto, tanto a nivel arquitectónico como de diseño, y estos patrones donde viven es en el código. También se tiende cada vez más a sustituir la documentación por la refactorización: si una pieza de software es compleja, tenemos dos posibles estrategias a la hora de hacerla entendible. La primera es documentarla; la segunda es, a base de refactorización y de mejorar el diseño, simplificarla para hacerla más clara. Se puede ganar mucho en la legibilidad de nuestro software simplemente eligiendo buenos nombres, evitando las funciones enormes, limpiando las variables no utilizadas.

La ventaja de este enfoque es que mejora objetivamente el software, haciéndolo más

entendible y más claro, sin necesidad de un artefacto externo que puede fácilmente quedar desactualizado [2].

1.1 Reseña del surgimiento de la propuesta MA-MRV-R1.

En el año 2007 se hizo en la Facultad 7, de la Universidad de las Ciencias Informáticas un estudio basado en las principales características de los proyectos productivos, pues no se encontraban ajenos a la situación a la que estaban sometidos, pues se había venido mostrando que la programación en proyectos tendía a presentar problemáticas o falencias que se presentaban en el equipo de desarrollo, por mala planeación, requisitos mal comprendidos entre otros. Razón por la que se le otorgó como tema de tesis a la estudiante Malay Rodríguez Villar la “Introducción de procedimientos ágiles en la producción de software en la Facultad 7 de la Universidad de las Ciencias Informáticas”.

La ingeniera Malay en su trabajo hizo una amplia y profunda investigación de las metodologías ágiles, exponiendo su definición y rasgos distintivos, lo que la llevo a analizar un importante documento que abarca todo sobre estos procedimientos: el Manifiesto Ágil [3], documento que expone los principios, características y valores de este tipo de metodologías. Para darle solución al problema planteado se propone explicar, guiar y orientar cómo sería introducir procedimientos ágiles en el proceso productivo de la Facultad 7. Precisamente su objetivo es darle solución a dicha problemática haciendo uso adecuado de las diferentes metodologías ágiles de desarrollo que existen. Entre sus principales resultados está la realización del estudio de las metodologías ágiles de desarrollo de software, realizó un estudio de las diferentes clasificaciones de proyectos, hizo una selección de los proyectos productivos de la facultad como muestra de la investigación, realizó numerosas entrevistas a sus líderes, clasificó cada uno de sus proyectos seleccionados y seleccionó los procedimientos adecuados según la clasificación hecha. Las metodologías ágiles usadas en la propuesta son las metodologías: SCRUM, para la planificación de los proyectos que usarán métodos ágiles como proceso de desarrollo, para llevar a cabo el proceso de desarrollo del proyecto se tomará en cuenta las mejores prácticas de XP, procurando que el proceso sea efectivo y eficiente.

Entre las recomendaciones hechas sugiere dar continuidad a su investigación para cumplir con la implantación de los procedimientos ágiles propuestos, además de aconsejar diseñar un curso optativo mediante el cual sean preparados los estudiantes para la utilización de estos procedimientos. Concluyendo la investigación con un amplio dominio de las

metodologías ágiles más conocidas a nivel mundial, un estudio que recoge la caracterización de los proyectos productivos de la facultad 7, teniendo en cuenta una serie de parámetros establecidos y la determinación de los procedimientos ágiles más factibles a utilizar en la producción de software de la facultad 7, de acuerdo con las características de los proyectos que allí se desarrollan, quedando como resultado una propuesta metodológica de los procedimientos ágiles a seguir.

A continuación aparece el guión de la propuesta [4]:

Planificación ↔ Definición

- Entrevista con el cliente (concepción inicial).
- Juego de la planificación.
- Captura de requisitos:
    - Creación de la Lista de Reserva del Producto (LRP).
    - Priorización de la LRP.
    - Definir las historias de usuario.
    - Asignar las tareas de las historias de usuario.
- Valoración del esfuerzo.
- Diseño con las metáforas.
- Creación de las tarjetas CRC.
- Refactorización (eliminar complejidad, diseñar lo más simple que se pueda).
- Reunión de la revisión del diseño.

Desarrollo

- Junta de planificación.
    - Definir las historias de usuarios a implementar.
    - Tareas para lograr dicha implementación.
- Fase de programación:
    - Programación en pareja.
    - Propiedad colectiva.

- Integración continúa.
- Estándares de programación.
- Cliente en sitio.

➢ Junta de seguimiento.

➢ Junta de revisión.

➢ Pruebas:

- Unitarias.
- Autorizadas.

Entrega

➢ Entrega de la documentación.

➢ Entrenamiento.

➢ Marketing.

Mantenimiento

➢ Soporte.

Observación: Desde el diseño en la fase de Planificación ↔ Definición y toda la fase de Desarrollo es un ciclo.

1.2 Estado actual de las metodologías utilizadas (SCRUM y XP).

SCRUM, más que una metodología de desarrollo de software, es una forma de auto-gestión de los equipos de programadores. Un grupo de programadores deciden cómo hacer sus tareas y cuánto van a tardar en ello. SCRUM ayuda a que trabajen todos juntos, en la misma dirección, con un objetivo claro. Permite además seguir de forma clara el avance de las tareas a realizar, de forma que los jefes pueden ver día a día cómo progresa el trabajo. Es una de las más conocidas metodologías ágiles, y se basa en un enfoque iterativo, donde cada iteración se denomina Sprint. El principio básico es que es muy difícil contar desde el principio con un catálogo completo de funcionalidades, ya que los requisitos van surgiendo conforme el propietario de la aplicación y los usuarios de la misma van haciendo sucesivas aportaciones. Así pues, SCRUM plantea el desarrollo de sucesivas versiones ampliadas, todas ellas plenamente usables y evaluables por el usuario. SCRUM es, además, una

metodología especialmente indicada para pequeños equipos de desarrollo y se orienta a una entrega rápida de resultados y una alta flexibilidad.

Al final implantar SCRUM es un proceso de cambio, y uno además bastante serio. De hecho, hay un 40% de empresas que no lo consiguen en absoluto y un 20% que solo consiguen un modelo híbrido de SCRUM, que les permite mejorar en su trabajo pero queda lejos del auténtico poder de SCRUM desencadenado, ya que ayuda a que trabajen todos juntos, en la misma dirección, con un objetivo claro.

Entre las cosas que nos ofrece esta metodología tenemos que [5]:

- Permite a las organizaciones eliminar los impedimentos clásicos en el desarrollo de los proyectos, aumentando la satisfacción de los clientes mediante la realización de entregas frecuentes de resultados tangibles e integrándolos activamente en el ciclo de desarrollo, lo cuál proporciona además una mayor adaptación y adecuación a sus necesidades.

- Potencia la formación de equipos de trabajo autosuficiente y multidisciplinarios, reduciendo la carga de gestión y proporcionando a los miembros del equipo un entorno amigable y productivo para desarrollar sus habilidades al máximo. Este entorno proporciona además mayor calidad de vida a los trabajadores y mejora drásticamente la moral en las organizaciones.

- Se centra en el producto y las personas, y hace especial hincapié en la eliminación preactiva de todas las trabas e impedimentos que surjan durante el desarrollo. Así pues, permite a muchas organizaciones alcanzar el llamado “Efecto Toyota”: cuatro veces la productividad media del sector, con doce veces la calidad.

- Es simple, aunque duro. Es sencillo combinar SCRUM con otras metodologías y marcos de gestión de proyectos. Si pensamos que sí, y fallamos, sentiremos la tentación de abandonar SCRUM. Por eso es importante recordar constantemente que SCRUM no es una bala de plata.

- SCRUM es iterativo e incremental. Eso quiere decir que liberamos código (o producto) frecuentemente, y cada liberación representa un incremento sobre la anterior.

- El producto que funciona es la única medida del avance del proyecto. No cuántas horas hemos echado. No cuántos recursos llevamos consumidos. No cuántas tareas llevamos hechas. Si no hay producto que funciona, no avanzamos.

- Todo en SCRUM tiene un límite de tiempo. Nada de “alargamos la reunión un poco más”. Nada de “El lunes, tal vez el martes”. La reunión acaba a las cuatro, y trabajaremos con lo que tengamos al final de la misma. La entrega es el lunes, y entregaremos lo que tengamos el lunes. SCRUM cambia el enfoque tradicional de “recursos” + “funcionalidades deseadas” = “fecha de entrega” y lo convierte en “recursos” + “fecha de entrega” = “funcionalidades que podremos entregar”. Si se nos cae una funcionalidad de la lista, vaya por Dios. Pero entregamos.

- Las entregas son productos potencialmente utilizables y/o comercializables. Eso significa que nada de presentación y nada de planos o documentos describiendo lo que hará la aplicación: queremos tocar, ver y usar algo, aunque sea un prototipo de la función de login.

- Mantener Sprints de duración fija. SCRUM es disciplina: quien piense que las metodologías ágiles son una especie de hacking institucionalizado, se equivoca.

- El SCRUM Master no es un jefe. Es un “siervo – líder”. No le dice a la gente lo que debe hacer. No divide las tareas entre los miembros del equipo. No acepta o rechaza las funcionalidades. El equipo es quien decide, y el dueño de producto el que evalúa. El SCRUM Master sólo debe preocuparse de que el proceso SCRUM siga en marcha y de eliminar los impedimentos en el camino del equipo [6].

En cuanto a calidad el 46% dice que los resultados usando SCRUM son los mismos o peores que con otros métodos, mientras que el 54% dice que son mejores. Sin embargo sobre la productividad la opinión es bastante más favorable: el 74% dice que con SCRUM han mejorado la productividad; y en lo que también saca buena nota es en la "simpatía" y buena disposición hacia el modelo: Le cae bien al 77%, y hace que el equipo tenga más claros sus objetivos y lo que se espera de él (80%) [7].

Figura 1. Encuesta realizada en Yahoo de SCRUM en el año 2008.

También tenemos la Programación Extrema (XP); metodología de desarrollo de software más exitosa en la actualidad, utilizada para proyectos de corto plazo y equipos pequeños; pues ha generado gran interés por su creciente número de casos de éxito en la industria. Metodología ágil que requiere gran disciplina. Un proceso ligero de bajo riesgo, flexible, predecible, científico y divertido de desarrollar software. Su utilidad se mide en valores como la simplicidad, comunicación, realimentación y coraje. Es sin duda alguna el método ágil que primero viene a la mente cuando se habla de modelos heterodoxos y el más trangesor entre ellos, entre las metodologías ágiles la más popular con un 38% del mercado ágil, contra su competidor mas cercano FDD.

La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar el éxito del proyecto. Las entregas son frecuentes, y existe una refactorización continua, lo que nos permite mejorar el diseño cada vez que se le añada una nueva funcionalidad. Sustenta la propiedad colectiva, la programación en parejas y pequeños encuentros diarios donde cada uno de los integrantes del equipo cuenta que ha hecho, que problemas tiene, y que hará más adelante [8].

Esta metodología soporta 12 disciplinas:

- Juego de planificación.
- Pequeñas liberaciones.
- Metáfora.

- Diseño simple.
- Pruebas.
- Refactorización.
- Propiedad colectiva.
- Programación por pares.
- 40 horas semanales.
- Integración continúa.
- Cliente en el lugar.
- Estándares de codificación.

¿Qué es lo que propone XP?

- Empieza en pequeño y añade funcionalidad con retroalimentación continua.
- El manejo del cambio se convierte en parte sustantiva del proceso.
- El costo del cambio no depende de la fase o la etapa.
- No introduce funcionalidades antes que sean necesarias.
- El cliente o el usuario se convierte en miembro del equipo.

XP 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 proporcionando un buen clima de trabajo. Se basa en la retroalimentación continúa 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. Está formado por valores, principios y prácticas. Las prácticas son actividades concretas que un equipo puede realizar día a día, mientras que los valores representan el conocimiento fundamental que sustenta dichas prácticas. Tanto los valores como las prácticas son necesarios, pero hay un gran espacio entre ambos.

El crecimiento de estos métodos ágiles y su penetración ocurre a un ritmo pocas veces visto en la industria, según las estadísticas de estos últimos 3 ó 4 años. El 50% de las empresas definen ágiles más de la mitad de los métodos empleados en sus proyectos. [9]

La adopción de estas metodologías ágiles esta creciendo tanto en medianas como en grandes empresas. El 46% de las compañías encuestadas con un número de empleados entre 100 y 1.000 ya se están utilizando prácticas ágiles ampliamente. En compañías más grandes el porcentaje se reduce al 12% aunque el 44% las están utilizando en algún proyecto. La habilidad para responder a los cambios impulsa el uso de las metodologías ágiles. Si bien el interés en estas metodologías tiene motivos distintos entre los profesionales de tecnologías de la información y el resto. Los primeros buscan técnicas que les ayuden a gestionar mejor el ámbito de los proyectos así como a responder mejor a los cambios, mientras que los segundos buscan alternativas que les permitan reaccionar más rápidamente a los frecuentes cambios en las prioridades de negocio. [10]

VersionOne, en asociación con la Red Líder de Proyectos Ágiles presentó los resultados de la segunda encuesta anual "Estado del Desarrollo Ágil". La encuesta se ha convertido en una de las más grandes en el mercado de Desarrollo Ágil; recibiendo mil setecientas respuestas con un alcance verdaderamente global de los encuestados de setenta y un países. [11]

La encuesta en algunos aspectos claves del Desarrollo Ágil, incluyendo:

- Los equipos en la práctica de Desarrollo Ágil son cada vez mayores y más distribuidos.
    - El 31% de los encuestados son de grupos de desarrollo, con más de 250 personas.
    - El 74% de los encuestados son de grupos de desarrollo, con más de 20 personas.
- Con el desarrollo ágil es significativo y mensurable la entrega de los resultados del negocio. Los encuestados informan mejoras especificas superior al 10% incluyendo:
    - Aumento de la productividad – 90% de los encuestados.
    - La reducción de los defectos del software – el 85% de los encuestados.
    - Acelerado Time-to-Market – 83% de los encuestados.
    - La reducción de gastos – el 66% de los encuestados.

La encuesta de este año muestra claramente que los equipos de Desarrollos Ágiles son cada vez mayores y más distribuidos. Adicionalmente, los resultados indican que el desarrollo ágil está rindiendo frutos para las empresas por entregar valor de negocios medibles.

Otros puntos claves que muestran las encuestas incluyen:

- Las dos razones más importantes para la adopción de metodologías ágiles fueron los siguientes:
    - Mejorar capacidad de gestionar los cambios en las necesidades (30%).
    - Acelera el tiempo de salida al mercado (24%).
- Los dos principales obstáculos para una mayor adopción de metodologías ágiles fueron:
    - Resistencia general al cambio (36%) [#2 fue en el 2006].
    - Personal con experiencia en metodologías ágiles (34%) [fue # 1 en el año 2006].

En esta encuesta también se valoro como iba avanzando el uso de las metodologías SCRUM y XP; pues muchas empresas han adoptado la unión de estos métodos ágiles para su desarrollo de software. Y como resultado se muestran las siguientes estadísticas para la aceptación de SCRUM + XP:

- Scrum: 37%
- Scrum + XP: 23%
- XP: 12%

Según los estudios realizados consideramos que muchas de las empresas “solo SCRUM” están de hecho usando muchas de las prácticas de XP. Por lo que para el próximo año, haciendo una pequeña apreciación después de los estudios realizados se considera que quedará SCRUM 30%, SCRUM + XP 40% y XP 25%. [12]

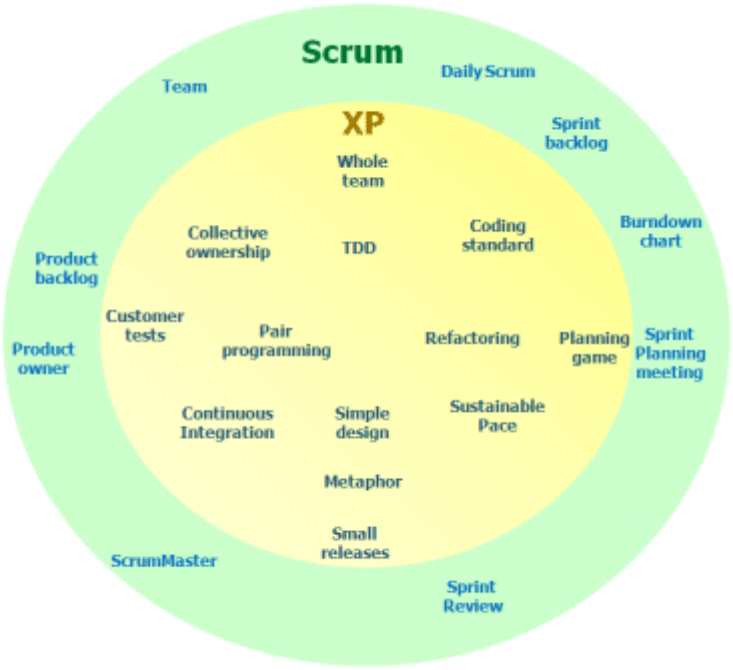

Figura 2. Unión de SCRUM + XP.

¿Cómo acoplan ambas metodologías? SCRUM puede ser visto como una interfaz entre el equipo y los clientes. SCRUM deja un agujero en el medio, ejemplo: como el equipo debe hacer su trabajo diario, en el cual XP encaja muy bien. Una vez que SCRUM es puesto en práctica el equipo es auto administrado y puede escoger como hacer el trabajo. El desarrollo de software ágil es todo sobre círculos de retroalimentación. SCRUM y XP se complementan el uno al otro de manera muy positiva. Como se muestra en la figura anterior, (Amarillo –XP, Verde –SCRUM). Las dos metodologías dan un círculo de retroalimentación de unos pocos segundos, donde los defectos son detectados y corregidos en segundos.

Más del 60% de lo encuestados declaran que las metodologías ágiles funcionan bien para ellos y su equipo y el mismo porcentaje declara que las metodologías ágiles han mejorado como colaboran desarrolladores y probadores [13].

1.3 Concepción del grupo UNICORNIOS.

En el año 2007-2008 la propuesta hecha por la ingeniera Malay Rodríguez Villar fue puesta en práctica en uno de los proyectos productivos de la Facultad 10, de la Universidad de las Ciencias informáticas; el proyecto UNICORNIOS (Servicios Especializados para la Migración a Software Libre); es el proyecto responsable de la migración de dicha facultad,

asesora en materia de migración e investigación de tecnologías libres al resto de las facultades, direcciones y proyectos de la universidad en general.

Fue creado en Septiembre del 2005, con el objetivo de crear un grupo de soporte para la imagen de GNU/Linux de la docencia de la Facultad 10 e inicialmente fue nombrado “Grupo de Soporte Técnico“. Este grupo pertenece al Polo de Software Libre y se caracteriza por su apoyo a la Comunidad de Software Libre de manera incondicional, y las necesidades de la migración de la UCI (Universidad de las Ciencias Informáticas) a plataformas libres.

Asume varios proyectos dentro de ellos, el servicio de soporte técnico de los laboratorios docentes, fomenta el estudio acelerado de las herramientas alternativas, aplicaciones de tipo desktop como N-Internos, Tocororo Desktop, el sistema de clonación de imágenes (SistClon); aplicaciones Web como el portal de ajedrez, Infodrez, el Portal de Software Libre II, el Observatorio de Tecnologías Libres, Service Desk, numerosos servicios como la revista de software libre UXI, servicio de documentación libre, repositorio de Debian, entre otros. Además de brindar capacitación con la realización de numerosos cursos optativos publicados en el sitio de documentación del portal de software libre.

Entre algunos de los cursos optativos que asesora el proyecto UNICORNIOS tenemos:

- Curso Básico de GNU/Linux.
- Curso de Nivel Medio en GNU/Linux.
- Herramientas para el Trabajo Colaborativo.
- Programación en Ncurse.
- Programación en Lenguaje Bash.
- Programación en Lenguaje PHP Avanzado.
- CMS Zope/Plone.
- CMS Zope/Plone Nivel Medio.
- Servicios telemáticos en SWL.
- Metodologías Ágiles.
- Servicios Telemáticos en SWL.

1.4 Tipos de proyectos donde se aplica la propuesta en este grupo.

No es tan sencillo enmarcar un proyecto en un tipo de clasificación, pues existen varias, incluso se pueden encontrar clasificaciones ortogonales, es decir con más de un criterio clasificatorio. Es muy importante tener clasificados los proyectos productivos que se desarrollen en una organización, pues esto facilita que el trabajo con ellos sea más viable y ligero. Pero a pesar de lo contradictorio que puede ser este tema de la clasificación de los proyectos informáticos, según Sigfrido González Fulle las tipologías más relevantes con las que podemos clasificarlos son:

- **Proyectos de Desarrollo de aplicaciones:** en estos proyectos se realiza la elaboración y puesta en marcha de programas o sistemas computacionales.
- **Proyectos de Equipamiento:** en estos proyectos se promueve la adquisición por primera vez de equipos, incluyendo tanto hardware como software básico utilitario.
- **Proyectos de Mejoramiento, ampliación o reposición:** en estos proyectos se promueve el aumento de capacidad y calidad de servicios de hardware y/o mejoramiento de software. [14]

Además en el libro Ingeniería de Software un enfoque práctico de Roger Presuman se hace una caracterización genenérica de los proyectos teniendo en cuenta el tipo de software que desarrollan, siendo estas clasificaciones las siguientes:

Software de Sistema: Es un conjunto de programas que han sido escritos para servir a otros programas. Algunos programas de sistema (por ejemplo: compiladores, editores y utilidades de gestión de archivos.) procesan estructuras de información complejas pero determinadas. Otras aplicaciones de sistemas (por ejemplo: ciertos componentes del sistema operativo, utilidades de manejo de periféricos, procesadores de telecomunicaciones) procesan datos en gran medida indeterminados. En cualquier caso, el área del software de sistema se caracteriza por una fuerte interacción con el hardware de la computadora; una operación concurrente que requiere una planificación, una compartición de recursos y una sofisticada gestión de procesos; una estructura de datos complejas y múltiples interfaces externas.

Software de tiempo real: El software que coordina/analiza/controla sucesos del mundo real conforme ocurre, se denomina de tiempo real. Entre los elementos del software de tiempo real se incluye: un componente de adquisición de datos que recolecta y da formato a la información recibida del entorno externo, un componente de análisis que transforma la

información según lo requiera la aplicación, un componente de control/salida que responda al entorno externo, y un componente de monitorización que coordina todos los demás componentes, de forma que pueda mantenerse la respuesta en tiempo real (típicamente en el rango de un milisegundo a un segundo).

Software de gestión: El proceso de la información comercial constituye la mayor de las áreas de desarrollo de aplicación del software. Los sistemas discretos (por ejemplo: nóminas cuentas de haberes-débitos, inventarios, etc.) han evolucionado hacia el software de sistema de información de gestión (SIG) que accede a una o más bases de datos que contienen información comercial. Las aplicaciones en esta área reestructuran los datos existentes para facilitar las operaciones comerciales o gestionar la toma de decisiones. Además de las tareas convencionales de procesamiento de datos, las aplicaciones da software de gestión también realizan cálculo iterativo (por ejemplo: el procesamiento de transacciones en puntos de ventas.)

Software de ingeniería y científico: Esta caracterizado por los algoritmos de manejo de números: las aplicaciones van desde la astronomía hasta la vulcanología, desde el análisis de la presión de los automotores y dinámica orbital de las lanzaderas espaciales y desde la biología molecular a la fabricación de automática: Sin embargo, las nuevas aplicaciones del área de ingeniería/ciencia se han alejado de los algoritmos convencionales numéricos. El diseño asistido por computadoras, la simulación de sistemas y otras aplicaciones interactivas, han comenzado a coger características del software de tiempo real e incluso de software de sistema.

Software empotrado: Reside en memoria de solo lectura y se utiliza para controlar productos y sistemas de los mercados industriales y de consumo. Puede ejecutar funciones muy limitadas y curiosas (el control de las teclas de un horno de microondas) o suministrar una función significativa y con capacidad de control (por ejemplo funciones digitales en un automóvil, tales como de la gasolina, indicadores en el salpicadero, sistemas de frenado. etc.).

Software de computadoras personales: El procesamiento de textos, las hojas de cálculo, los gráficos por computadoras, multimedia, entretenimientos, gestión de base de datos, aplicaciones financieras, de negocios y personales y redes o acceso a bases de datos externas son algunas de los cientos de aplicaciones.

Software basado en Web: Las páginas Web buscadas por un explorador son software que incorpora instrucciones ejecutables (por ejemplo, CGI, HTML, Perl, o Java), y datos (por ejemplo, hipertexto y una variedad de formatos de audio y visuales). En la esencia, la red viene a ser una gran computadora que proporciona un recurso software casi ilimitado que puede ser accedido por cualquiera con un modem.

Software de inteligencia artificial (IA): Hace uso de algoritmos no numéricos para resolver problemas complejos para los que no son adecuados el cálculo o el análisis directo. Los sistemas expertos, también llamados sistemas basados en conocimientos, reconocimiento de patrones (imágenes y voz), redes neuronales artificiales, prueba de teoremas, y los juegos son representativos de las aplicaciones de esta categoría.

De una manera muy significativa las clasificaciones anteriores tienen aspectos que hacen que sean tipologías muy superiores, debido a que se exponen muchas de las actividades que se llevan a cabo en los proyectos informáticos. Después del estudio realizado en los sistemas que se desarrollan en el grupo UNICORNIOS la autora de esta investigación considera, que para el trabajo con estos proyectos productivos, basados en Software Libre, esta clasificación esta incompleta, por lo que se definen nuevas tipologías, teniendo en cuenta las planteadas anteriormente y las características específicas de cada uno de los proyectos.

Las nuevas tipologías que permitirán conocer la clasificación de los proyectos productivos, basados en Software libre, cuando se realice su caracterización son:

- **Proyectos de Desarrollo de aplicaciones:** Son aquellos proyectos en los que se desarrollan fundamentalmente aplicaciones de escritorio, así como programas y sistemas computacionales.
- **Proyectos de Desarrollo Web:** Son aquellos proyectos que se dedican totalmente a la fabricación de sitios Web o portales.
- **Proyectos de servicios:** Son aquellos proyectos, que solamente se dedican a prestar servicios a clientes.
- **Proyectos de personalización:** Son aquellos proyectos, que se encargan de personalizar versiones de sistemas operativos libres, para las áreas o instituciones que lo necesiten. La nueva versión del sistema queda adecuado a las necesidades y requerimientos del cliente.

- **Proyectos de investigación:** Son aquellos proyectos, que su producto final no es precisamente un software, sino un plan, realizado luego de una investigación bastante profunda sobre el tema en el que estén trabajando, y que orienta las acciones para alcanzar un objetivo final.

1.4.1 Características de los proyectos seleccionados en el grupo UNICORNIOS.

Es necesario mantener una clasificación de las diferentes topologías de los sistemas que existen en los proyectos productivos. Los proyectos de desarrollo de software pueden ser caracterizados teniendo en cuenta numerosos puntos que sirven de guía para su clasificación.

En el grupo UNICORNIOS, de la Facultad 10, de la Universidad de las Ciencias Informáticas la organización de los proyectos está dada por polos. En nuestro caso el grupo UNICORNIOS pertenece al Polo de Software Libre. Grupo que consta de sistemas que se encuentran en estado de desarrollo, otros ya terminados y que a la vez constituyen servicios, además de los que se encuentran de forma experimental, ya sea en desarrollo o en investigación.

Para la realización de esta investigación se hizo necesario que se efectuara un estudio de todos los sistemas que se desarrollan según las características. Debido a que en este proyecto tenemos sistemas de escritorio, Web y servicios.

A continuación aparecen los sistemas en desarrollo del grupo UNICORNIOS.

Grupo de trabajo	**Proyectos que se desarrollan.**
UNICORNIOS	Gestión de investigación y evento.
	Home compartido.
	MINPPAL.
	N-Internos.
	Observatorio de tecnologías libres.
	Octave Front-End.
	Portal de ajedrez.
	Portal de SWL 2.
	ScoreX 2.
	ServiceDesk.
	Sistema de clonación de imágenes (SistClon).
	Software para ciegos.
	Tocororo desktop.
	UPLOAD Cliente.
	Revista de SWL: UXI.
	Repositorio.

Entre los proyectos del grupo UNICORNIOS se hizo una selección entre todos sus sistemas para aplicar la propuesta MA-MRV-R1. Entre los que se escogieron 6 sistemas; dos sistemas Web, Portal de SWL (Software Libre) y Service Desk; dos sistemas de escritorio, Tocororo Desktop y Sistema de Clonación; y dos servicios, la revista UXI y el MINPPAL.

Estos sistemas son muy diferentes debido a la tipología desigual que presentan. Esto hace que unos sean más complejos que otros, y las aplicaciones y funcionalidades no sean semejantes.

La recogida de los datos (teniendo en cuenta importantes parámetros que son la base de una posterior clasificación y agrupación de los proyectos, posibilitando un mejor estudio y análisis de los mismos) que caracterizan cada uno de los proyectos fue posible a través de un encuentro realizado con el líder del proyecto UNICORNIOS, Ing. Abel Meneses Abad. Las características de estos sistemas aparecen a continuación:

Características del Portal SWL II:

- Tipo de proyecto: Proyecto de desarrollo Web.
- Cantidad de integrantes: 5 personas.
- Roles: Gerente, Programador, Tester.

- Entorno de desarrollo: Python para la Web, CMS - Zope/Plone.
- Envergadura: Media.
- Tiempo de duración estimado: 12 meses.
- Complejidad: Media.
- Metodología utilizada: Propuesta MA-MRV-R1.
- Problemas en el proceso de desarrollo:
    - Complejidad de las tecnologías.
    - Inestabilidad de la paquetería a utilizar.

Características de ServiceDesk:

- Tipo de proyecto: Proyecto de desarrollo Web.
- Cantidad de integrantes: 8 personas.
- Roles: Gerente, Cliente, Analista, Arquitecto, Programador, Diseño.
- Entorno de desarrollo: LAMP.
- Envergadura: Alta.
- Tiempo de duración estimado: 24 meses.
- Complejidad: Media.
- Metodología utilizada: Propuesta MA-MRV-R1.
- Problemas en el proceso de desarrollo:
    - Complejidad de la tarea por desconocimiento de las tecnologías.
    - Organización del equipo.
    - Surgimiento de nuevas propuestas en el tiempo gracias al estudio del arte.

Características de Tocororo Desktop:

- Tipo de proyecto: Proyecto de desarrollo de aplicación.
- Cantidad de integrantes: 9.
- Roles: Gerente, Cliente, Analista, Arquitecto, Programador, Tester, Diseño.
- Entorno de desarrollo: C++ (Gtk), blender y Gimp.

- Envergadura: Media.
- Tiempo de duración estimado: 28 meses.
- Complejidad: Alta.
- Metodología utilizada: Propuesta MA-MRV-R1.
- Problemas en el proceso de desarrollo:
    - Alta complejidad de diseño conceptual.
    - Alta complejidad de las interfaces.
    - Poca preparación del equipo de desarrollo.

Características del Sistema de Clonación (SistClon):

- Tipo de proyecto: Proyecto de desarrollo de aplicación.
- Cantidad de integrantes: 7
- Roles: Gerente, Cliente, Programador, Analista, Diseñador, Tester, Arquitecto.
- Entorno de desarrollo: PHP, MySQL, Apache, Linux, Smarty, Ajax.
- Envergadura: medio.
- Tiempo de duración estimado:
- Complejidad: alta.
- Metodología utilizada: Propuesta MA-MRV-R1.
- Problemas en el proceso de desarrollo:
    - Muy alta complejidad del lenguaje, librerías.
    - Desconocimiento de las tecnologías o sistemas similares.
    - Problemas de organización y capacitación del equipo.

Características de MINPPAL:

- Tipo de proyecto: Proyecto de Servicio.
- Cantidad de integrantes: 9
- Roles: Gerente, Especialista en migración, Cliente, Especialista en base de dato.

- Entorno de desarrollo: PHP, MySQL, Apache, Linux, Smarty, Ajax.
- Envergadura: medio.
- Tiempo de duración estimado:
- Complejidad: alta.
- Metodología utilizada: Propuesta MA-MRV-R1.
- Problemas en el proceso de desarrollo:
    - Falta de referencias y experiencias sobre la migración del MINPAL, pues es el primer intento realizado.
    - Falta de preparación de los estudiantes sobre el tema de mecanismos de migración.

Características de la Revista UXI:

- Tipo de proyecto: Proyecto de Servicio.
- Cantidad de integrantes: 8 personas.
- Roles: Gerente, Arquitecto, Analista, Tester, Diseñador, Programador y Cliente.
- Entorno de desarrollo: Open Office, Gimp, Blender.
- Envergadura: Alta.
- Tiempo de duración estimado: 1 meses.
- Complejidad: Media.
- Metodología utilizada: Propuesta MA-MRV-R1.
- Problemas en el proceso de desarrollo:
    - Dificultad para la conformación del equipo.
    - Maduración del equipo.

Al analizar cada uno de los sistemas seleccionados en el grupo UNICORNIOS se puede decir que: se identificaron tres tipos de software (aplicación, Web y servicios); la cantidad de personas que involucran los mismos se incluyen en cuatro grupos, todos con menos de 10; de 1 a 5, de 1 a 7, de 1 a 8 y de 1 a 9, con mayor incidencia los de 1 a 8 y de 1 a 9 miembros; las tecnologías utilizadas son muy variadas debido a las diferencias que existen

en cada uno de los sistemas.

Todos los proyectos usan la propuesta MA-MRV-R1 como metodología de desarrollo; todos los proyectos involucran los roles principales (analista, diseñador, programador, gerente, tester, cliente, arquitecto); generalmente son proyectos de complejidad media y alta; teniendo en cuenta que los problemas son común en casi la totalidad de los proyectos, solo que no se centraron en las mismas ideas.

Conclusiones

Como fue visto, en este capítulo se hace una reseña de la autora de la propuesta MA-MRV-R1, se realiza una amplia investigación sobre el estado en el que se encuentran SCRUM y XP, las metodologías utilizadas en la propuesta; aparece un pequeño resumen de la concepción del grupo UNICORNIOS y se realizó un estudio de los aspectos más significativos de los proyectos seleccionados en el proyecto antes mencionado para aplicar dicha propuesta, que permitió la caracterización y clasificación de los mismos. Siendo el punto de partida de la metodología solución que contribuirá a la mejora en el proceso de desarrollo de los proyectos y por ende del proceso productivo.

CAPÍTULO 2 ESTUDIO DEL PROCESO PRODUCTIVO

Son muchas las personas y empresas dedicadas al desarrollo de software que se enfrentan hoy por hoy con el dilema de ser o no ser ágiles. Bien sea porque sus actuales metodologías de desarrollo no dan los resultados esperados o bien porque desean incursionar en prácticas que están de moda a lo largo y ancho de las comunidades de desarrollo en todo el mundo, lo cierto es que el auge de las metodologías ágiles es un tema que apasiona y hace reflexionar a todo aquel que se encuentre inmerso dentro del proceso de desarrollo de software.

La habilidad para responder a los cambios impulsa el uso de las metodologías ágiles. Si bien el interés en estas metodologías tiene motivos distintos entre los profesionales de tecnologías de la información y el resto. Los primeros buscan técnicas que les ayuden a gestionar mejor el ámbito de los proyectos así como a responder mejor a los cambios, mientras que los segundos buscan alternativas que les permitan reaccionar más rápidamente a los frecuentes cambios en las prioridades de negocio.

2.1 Proyectos internacionales que usan las metodologías XP y SCRUM.

Sin duda alguna ser ágil es lo de hoy. Con esto nos referimos a la capacidad para proveer respuestas rápidas y ser adaptables al cambio. Ambas cualidades siempre habían sido deseables, pero en el entorno de negocios actual son indispensables. Este requerimiento de agilidad en las empresas, provoca que el software; que a fin de cuentas es el motor del mundo actual, también deba ser desarrollado de manera ágil. [15]

Las necesidades de un cliente pueden sufrir cambios importantes del momento de contratación de un proyecto de software, al momento de su entrega; y es mucho más importante satisfacer estas últimas que las primeras. Esto requiere procesos de software diferentes, que en lugar de rechazar los cambios, sean capaces de incorporarlos. Dichos procesos también deben:

- Producir versiones ejecutables del software en pocas semanas, con el fin de quedar bien con el cliente y obtener retroalimentación en cuanto al producto.
- Inventar soluciones sencillas, de tal forma que el impacto de los cambios se minimice.

- Mejorar la calidad del diseño de manera continua, haciendo que la siguiente iteración requiera de menos esfuerzo que la actual.
- Probar desde temprano y de manera continua, para encontrar defectos antes de que tengan un alto impacto. [16]

SCRUM no es una metodología de desarrollo tal cual la conocemos. Más bien es una forma de gestionar un equipo de forma que trabaje de forma eficiente y de tener siempre medidos los progresos, de forma que sepamos por dónde andamos. SCRUM no nos dice qué pasos debemos seguir para hacer un software correcto, de hecho, SCRUM podría aplicarse a tareas que no son software, como jardinería. Básicamente, SCRUM consiste en fijar unos objetivos a corto plazo, reunirse todos los días para ver cómo va el tema y solucionar los problemas que salgan. La idea es que todo el equipo, día a día, no pierda de vista el objetivo a corto plazo y trabajen todos en la misma dirección. De hecho, una de las ideas fundamentales de SCRUM es que al salir de la reunión diaria, todos tengan claro qué van a hacer ese día. No es muy acogido por los grupos de trabajo, pero esto es una de sus tareas que se sugiere como fundamental. Normalmente siempre todos tenemos catorce o quince cosas que hacer, se pierde mucho tiempo bailando entre ellas, mirando una y otra, hasta que finalmente nos decidimos por una de ellas. [17]

Sin embargo, la programación extrema, va más allá, en el sentido que dice cosas como que hay que trabajar por parejas, hacer test unitarios antes que el código.

Y por lo visto, SCRUM y XP se llevan muy bien y son complementarios. Por lo que es buena idea utilizar ambos simultáneamente. XP nos dice cómo tenemos que hacer el software y SCRUM nos dice día a día si vamos bien.

Este tipo de metodologías se utiliza principalmente en pequeños proyectos o proyectos internos que resuelvan problemas concretos, lo que esta reñido con su aplicación en el desarrollo de grandes sistemas, ya que una correcta modularización de los mismos es fundamental para su exitosa implantación. Para el desarrollo de proyectos se utilizan equipos reducidos y fuertemente cohesionados, con no más de 20 personas, los grupos de trabajo suelen ser pequeños. Integrados por personas altamente calificadas y motivadas, trabajando en un entorno adecuado y recibiendo todo el apoyo necesario.

SCRUM esta especialmente indicada para proyectos con un rápido cambio de requisitos, mientras que XP se define como especialmente adecuada para proyectos con requisitos imprecisos, muy cambiantes y donde existe un alto riesgo técnico. [18]

Existen numerosos equipos que han desarrollado su software utilizando como premisa el uso de una metodología ágil; debido a que la práctica de la unión de las metodologías SCRUM y XP ha desatando grandes éxitos y resultados. SCRUM ha sido empleado de manera efectiva fisionándolo con las prácticas de ingeniería con XP (Extreme Programming). El primero provee los mecanismos de administración ágiles y el segundo la integración de las prácticas de ingeniería.

Un artículo escrito por Ken Schwaber y Kane Mar en el año 2007 expresa las razones que hace que la inclusión de estos dos métodos ágiles aporte grandes beneficios para quienes las adopten en sus sistemas. Aportes que a continuación aparecen:

1- La administración ágil y el control de mecanismos de SCRUM son aplicables para cualquier tipo de proyecto. Incluyendo iniciativas de negocios que consisten en simultáneos y múltiples desarrollos de software, desarrollo de negocio, re-ingeniería, marketing, soporte e implementación de proyectos. Los proyectos con XP y SCRUM encajan con los frameworks de desarrollo de estas iniciativas.

2- Los proyectos con XP y SCRUM tienen en cuenta todos los beneficios de la organización, los equipos están orientados al objetivo principal que persigue la organización.

3- Cuando XP es integrado con SCRUM, ellos se convierten escalables y pueden ser puestos en marcha de manera simultanea por equipos no localizados en un mismo lugar.

4- SCRUM implementa en un día, XP puede ser implementado gradualmente dentro del frameworks de SCRUM.

5- Los proyectos con XP y SCRUM se benefician con los negocios ADM´s, los valores métricos del proceso para medir y gestionar las iniciativas ROI.

Beneficios con los que estamos totalmente de acuerdo según las prácticas que utilizan dichas metodologías. Esta estrategia es seguida por muchas empresas, debido a que es muy práctico implantar SCRUM como metodología de gestión, y luego ir introduciendo las prácticas de ingeniería de XP, adecuándolas a su entorno, logrando mayor productividad, mayores resultados y satisfacciones por parte del cliente y del equipo de trabajo.

Dentro de los grandes proyectos que solo utilizan SCRUM tenemos a Google, Yahoo!, Oracle, Sun, Microsoft, la industria japonesa, y dentro de ella: Toyota, HONDA, Fuji-Xerox, Borland Quattro Pro, Canon, SUN Microsystems, EPSON, 3M, hp, NOKIA connecting

people, MC Software Factory y High Moon Studios que es la pionera en la implantación de esta metodología para el desarrollo [19].

Mientras que Google, ORACLE, Yahoo!, SUN Microsystems y Microsoft utilizan la mezcla de las dos metodologías, SCRUM y XP. En este mismo resumen se menciona a otras empresas como Siemens, State Farm, IBM, I2E y Federal Reserve Bank.

2.1.1 Experiencias de proyectos donde se aplica la metodología "SCRUM + XP".

Microsoft

Microsoft en boca de David Treadwell ha explicado que están promoviendo el uso interno de SCRUM y de ciertas prácticas de XP como la programación por parejas. Lo cierto es que otras grandes empresas están introduciendo métodos ágiles entre sus prácticas de desarrollo. Google incluye conocimiento de las mismas como deseable para los que soliciten trabajo en ella. Por otro lado en un reciente congreso de SCRUM (SCRUM Gathering 2005) Pete Deemer explicaba la introducción de SCRUM y XP en Yahoo.

Esta tendencia puede animar a empresas más pequeñas a introducir SCRUM y XP entre sus prácticas de desarrollo, pero esperemos que no lo hagan como una moda. Es necesario conocer bien estas metodologías para decidir si son las más apropiadas para cada caso concreto y para encontrar la mejor manera de introducirlas si se opta por ellas. [20]

Microsoft ha combinado los modelos de trabajo ágiles SCRUM y XP para finalizar el lanzamiento de las nuevas versiones: SQL Server 2005, Visual Studio 2005 tool suite y Biztalk server 2006 integration server. Esta es la forma elegida por varios de sus equipos de desarrollo para lograr *entregar software de calidad a tiempo y en sintonía con los requisitos de los usuarios*.

Es significativo que la mayor empresa de desarrollo de software del planeta haya terminado apostando por prácticas ágiles para poner fin a las demoras que iba acumulando el lanzamiento de estos productos.

SCRUM dista mucho de los modelos ortodoxos de gestión de proyectos tipo PMI, y hace del gestor un "SCRUM Master". Un rol de facilitador más que de gestor. Encargado de asegurar el cumplimiento de las reglas SCRUM. Figura que asombra a los gestores tradicionales,

preocupados porque con SCRUM las responsabilidades de la gestión no son ya tanto de su propiedad, sino de un equipo auto-gestionado. [21]

Google

Google ha trabajado siempre con principios de desarrollo ágil: pequeños equipos (tres personas), autogestionados y con elevado nivel de autonomía y capacidad de decisión. Sin embargo el crecimiento de la empresa le plantea retos importantes, porque cada vez son más y más grandes los proyectos que tiene en marcha, cuenta con cinco centros de desarrollo repartidos en sedes diferentes, las funcionalidades tienen que implantarse traducidas a cada idioma, hay que mantener el material de marketing, etc. Describe cómo para dar continuidad al mismo espíritu ágil de Google, ha adaptado e implantado principios de SCRUM en la organización y XP para el desarrollo.

Lo curioso es que lo que comenta como ejemplo de una buena metodología ágil es lo que nos cuenta de cómo trabajan en Google: [22]

- Existen "managers", pero la mayoría de ellos escriben código al menos la mitad de su tiempo, haciéndoles más como "gurús-tecnológicos".
- Los desarrolladores pueden cambiar de equipo o proyectos en cualquier momento que quieran, sin preguntas.
- Google tiene la filosofía de no decir nunca a los desarrolladores en qué deben trabajar, y se lo toman muy en serio.
- Los desarrolladores son fuertemente animados a gastar el 20% de su tiempo (de horario laboral) en trabajar en lo que ellos quieran, con tal que no sea en su proyecto principal.
- No hay muchas reuniones. Quizás de media tres por semana, incluyendo una charla con su líder.
- Hay tranquilidad. Los ingenieros están silenciosamente centrados en su trabajo, individualmente o en pequeños grupos de 2 a 5 personas.
- No existen diagramas de Gantt u hojas con tareas-personas-fechas ni otros artefactos de gestión de proyectos visibles, no que se sepa.

- Incluso los relativamente raros periodos de estrés, la gente todavía va a comer y cenar, las cuales son gratuitas, y no se trabaja en horas intempestivas a menos que se quiera.

Es impresionante, no parece una empresa típica-media de desarrollo de software. Hay tres tipos de empresas que funcionan más o menos así: las "start-ups", las universitarias y Google. El mérito de Google es haber mantenido ese tipo de trabajo con su escala actual, con el tamaño que tiene, sin haberse burocratizado. Aunque hay que destacar que Google no es representativa de una empresa de software porque Google “NO” es una empresa de software. Lo que vende son anuncios. Es una empresa que ha desarrollado muchos productos de software. No sólo eso, sino que se trata de software realmente innovador. [23]

- El buscador de google es software.
- Gmail es software.
- Google Docs es software.
- Google Desktop es software.
- Google Maps es software.

A parte de los comentados antes también tienen Picassa, Google search appliance, Google Talk. Sin olvidar todo el Open Source que contribuyen que es de lo más innovador que se libera últimamente. Amazon es una empresa dedicada a la venta de productos. Google es una empresa de servicios, de búsqueda y gestión de información. Ellos venden servicios, no software, más allá de algunas aplicaciones puntuales. Lo nuevo de estos métodos no son las prácticas que utilizan, sino su reconocimiento de la gente como principal factor de éxito de los proyectos. No son más que el uso de reglas ligeras pero suficientes y la aplicación de técnicas orientadas a las personas y su comunicación.

Los principales valores que hacen que estas empresas utilicen estos métodos se expresan en el “Manifiesto Ágil” [24]:

- Individuos e interacciones por encima de procesos y herramientas.
- Software funcional por encima de documentación abundante.
- Colaboración con los clientes por encima de negociación de contratos.
- Respuesta al cambio por encima de seguimiento a planes.

High Moon Studios

Clinton Keith, director técnico de la empresa desarrolladora de videojuegos High Moon Studios afirma que en el desarrollo de un videojuego no es posible conocer cómo va ser éste hasta el final del proyecto, y por tanto los modelos ágiles resultan más apropiados que los formales que necesitan requisitos y planificaciones cerradas desde el principio. En un primer momento introdujo Scrum para el desarrollo del último producto: Darklwatch, porque "*era algo que podíamos adoptar rápidamente*".

Más tarde incorporó también Extreme Programming para superar problemas de documentación:
"*En el pasado empleábamos largas documentaciones técnicas intentando crear el código final a la primera. Desafortunadamente los documentos no eran demasiado correctos, y los cambios en el código podían crear bastantes problemas al equipo si no empleábamos prácticas XP*". Siendo esto propicio al cambio de cultura de gestión que implica la adopción de Scrum. Asumiendo de esta manera la unión de XP + SCRUM.

Este es uno de los principales problemas en su implantación: desde el modelo tradicional de gestión tipo PMI, no se comprende la cultura subyacente de mayor autonomía y "empowerment" del equipo, y a menudo se ve como una "perdida de poder" que alimenta la resistencia al cambio de muchos gestores que sólo trabajan con modelos clásicos. [25]

I2E

En I2E hemos comenzado a utilizar SCRUM para la gestión de los proyectos de desarrollo. Después de muchas horas de lectura y de probar otras metodologías, nos hemos decido por Scrum.

Scrum nos proporciona una metodología para realizar los proyectos, también nos proporciona herramientas como el burn-down y el burn-up para realizar el seguimiento del desarrollo. Hace hincapié en el feedback continuo entre el equipo de desarrollo con las reuniones diarias y con las retrospectivas al finalizar el Sprint, y también en el feedback continuo con el cliente al entregar un incremento completo al final de cada Sprint.

Y luego decidimos complementar SCRUM con otras prácticas ágiles por ejemplo de XP, otras metodologías conocidas. Lo importante es que si utilizamos SCRUM vamos a estar más preparados ante los cambios, y en el desarrollo de software la única cosa segura es

que durante el desarrollo las cosas van a cambiar. Además Google, Yahoo y otras grandes compañías utilizan SCRUM + XP, lo cual quiere decir que ya esta sobradamente probado y se ha comprobado que funciona. [26]

2.2 Revisiones de práctica de la propuesta MA-MRV-R1.

Con la puesta en práctica de la propuesta hecha por la Ing. Malay Rodríguez Villar en el grupo UNICORNIOS específicamente en los 6 sistemas escogidos para realizar el estudio de prueba, se decidió que en cada una de las iteraciones (cada iteración equivale a un Sprint) realizadas se desarrollará al concluir una revisión para observar el estado en que se encontraba cada sistema a medida que se iba trabajando.

Cada iteración se decidió que tendría que una duración de 2 meses, al cabo de ese tiempo cada uno de los sistemas es revisado, primero sin la presencia de los analistas, y luego en su presencia y la del Gerente para discutir los avances, las causas de los atrasos y las dudas.

Hasta el momento se realizaron 3 iteraciones. En cada una de ellas se mostraron numerosos avances, pero esto también aparejado a muchas deficiencias y dificultades. Todo esto nos sirvió para poder limar las asperezas que había quedo de la propuesta hecha. Nos dimos cuenta que hacían falta más artefactos para poder trabajar y tener la documentación completa de los productos.

A continuación se hace el resumen de cada una de las iteraciones realizadas:

2.2.1 Primera Iteración.

La idea de la utilización de una propuesta con metodologías ágiles no se maduró desde el comienzo de la iteración 1. Para muchos este tipo de gestión no existía, y se notaban casados con la metodología estudiada en la asignatura de Ingeniería de Software que se imparte en el tercer año de la carrera de Ingeniería en Ciencias Informáticas**.**

Surge entonces la primera dificultad encontrada: la falta de entusiasmo para utilizar las metodologías ágiles. Es difícil comprender a cada una de las personas que deben tomar esta responsabilidad, pues hacer el estudio de estas metodologías no es tarea fácil, resultando aún más complejo el impacto en cada uno de los implicados. Por esta razón se impartió un breve adiestramiento para que se pudiera comenzar a trabajar, y así empezar el

estudio del uso de esta propuesta en los diferentes tipos de proyectos pertenecientes al Polo de Software Libre. Se realizó la explicación de los artefactos que se utilizarían y lo que se haría en cada fase de trabajo según la propuesta; en poco tiempo los grupos de desarrollo se mostraron entusiasmados.

Pueden explicarse como causas de esta "alegría colectiva", la poca documentación a llenar y las ventajas sicológicas que infunde el método en estudiantes ávidos de programar, labor esta que podía ser realizada con mayor prioridad. Resulta interesante para la mayoría de los miembros además de la práctica de la programación en parejas, la entrega gradual de partes del producto, en conjunto a la integración del cliente al equipo de trabajo. Con esto se logra llegar a la entrega con satisfactorios resultados para los clientes, un problema que consume a otros equipos de desarrollo de la UCI.

Se revisaron los artefactos que se debían llenar en la fase de Planificación – Definición, no existía uno que recogiera la descripción del producto a desarrollar, donde aparecería la organización del equipo de trabajo, las herramientas utilizadas, los antecedentes, alcance y visión del mismo. Y de ahí surge la necesidad de la Plantilla de Concepción del sistema.

Cada uno de los sistemas presenta una carpeta en el SVN (Subversión); sistema de control de versiones del proyecto. En esta carpeta se pone todo lo que se realiza en cuanto a desarrollo, investigación y resultados obtenidos. Cada usuario, es decir miembro del equipo tiene diferenciados los permisos, ya que este servicio hace que también se pueda gestionar la seguridad de los documentos y código que son subidos.

La diversidad de nombres con que cada sistema evaluado coloca un mismo artefacto, hace difícil la revisión de la información. Como segundo paso se buscó una forma de organizar la Arquitectura de la Información en este sistema de control de versiones.

Arquitectura de la Información de los sistemas en el grupo UNICORNIOS

- Documento Concepción del sistema.
- Documento Estado del Arte.
- Carpeta Información de la tecnología.
- Carpeta Expediente de Proyecto.
- Carpeta Desarrollo de Sistema.

- Carpeta Resultados (Solo para los sistemas que ya tienen productos de los que son derivados.)

En cada uno de los sistemas en esta iteración se detecto que:

1- Revista UXI:

- Desconocimiento del trabajo con la propuesta MA-MRV-R1.
- Necesidad de un reajuste en cuanto a las iteraciones, pues el servicio de la revista hace lanzamiento de sus productos mensualmente.
- Necesidad de guardar los productos resultantes en una carpeta para ser puesto en el SVN.
- Desorganización a la hora de poner la información en el SVN.
- Falta del documento de Concepción del sistema.
- No se ha llenado las historias de usuario.

2- Portal de SWL 2:

- Desconocimiento del trabajo con la propuesta MA-MRV-R1.
- Falta del documento de Concepción del sistema.
- Necesidad de guardar los productos resultantes en una carpeta para ser puesto en el SVN.
- Falta colocar la información para colocar en la carpeta Información de la tecnología.
- La analista no ha llenado las historias de usuario para subirlas al SVN.

3- Tocororo Desktop:

- Desconocimiento del trabajo con la propuesta MA-MRV-R1.
- Falta las Historias de Usuario.
- Desorganización a la hora de poner la información en el SVN.
- Necesidad de guardar los productos resultantes en una carpeta para ser puesto en el SVN.
- Despreocupación de las analistas para comenzar a llenar la carpeta Expediente de proyecto.

4- ServiceDesk:

- Desconocimiento del trabajo con la propuesta MA-MRV-R1.
- Desorganización a la hora de poner la información en el SVN.
- Necesidad de guardar los productos resultantes en una carpeta para ser puesto en el SVN.
- Falta el documento de Concepción de sistema.
- Las historias de usuario presentan errores.

5- Sistema de Clonación (SistClon):

- Desconocimiento del trabajo con la propuesta MA-MRV-R1.
- Desorganización a la hora de poner la información en el SVN.
- Falta del documento de Concepción del sistema.
- Necesidad de guardar los productos resultantes en una carpeta para ser puesto en el SVN.
- La analista no ha subido las historias de usuario al SVN.

6- MINPPAL:

- Desconocimiento del trabajo con la propuesta MA-MRV-R1.
- Falta del documento de Concepción del sistema.
- Desorganización a la hora de poner la información en el SVN.
- No se ha realizado el levantamiento de los requisitos.

2.2.2 Segunda Iteración.

En esta iteración con un poco más de experiencia en cuanto al trabajo y estudio de la propuesta, por parte de cada uno de los miembros del equipo de desarrollo, se avanzó en los artefactos a llenar. Aún no quedan muy claros los conceptos de ¿Qué es una Historia de Usuario? ¿Cómo deben ser confeccionadas? Se logra dar un buen paso en cuanto a la conciencia del uso de las metodologías ágiles para trabajar en el desarrollo de software. Algunos equipos de desarrollo tienen su documento de concepción del sistema y todas sus historias de usuarios del sistema a gestionar, de manera regular. Además, cada una de las carpetas del SVN se han actualizado y cada analista organiza su documentación según la Arquitectura de Información acordada.

La organización ya no es un problema, pues cuando se quiere ver en qué estado está cada sistema solo se debe buscar el documento o la que carpeta deseada, toda la información está organizada con un mismo nombre y esto hace fácil la búsqueda y localización de la misma.

Después del estudio de cada uno de los sistemas por sus diferentes características se observa que para los sistemas que pertenecen al grupo de Servicios, es necesaria una plantilla donde se pueda gestionar los cambios realizados una vez el producto sea terminado. En esta iteración el producto del Portal de SWL 2 pasa a ser un servicio y no tiene donde registrar los cambios que se le hacen a su producto mientras se encuentra en mantenimiento. Y surge el artefacto: Plantilla de Gestión de Cambios.

1- Revista UXI:

- Finaliza con todas las historias de usuarios.
- La arquitectura de Información cumple con lo planteado.
- Presenta el documento de Concepción del sistema.
- Debe actualizar sus actividades, para saber como se desarrolla su cronograma de trabajo.

2- Portal de SWL 2:

- Presenta el documento de Concepción del sistema.
- Se integra todo el equipo de desarrollo a la utilización de la propuesta MA-MRV-R1.
- Terminación y colocación en el SVN de toda la información pedida.
- Creación de la carpeta Resultados.

3- Tocororo Desktop:

- Aún no han colocado las historias de usuario que su sistema presenta.
- Falta de comunicación entre el jefe de sistema y sus analistas.
- No se ha colocado la documentación de la carpeta Expediente de proyecto.
- Se centran solo en el diseño y los desarrolladores no están trabajando.

4- ServiceDesk:

- No aparecen las actividades que deben planificarse para saber el estado del cronograma de trabajo.
- La analista debe organizar la información colocada en Expediente de proyecto.
- Queda cierto grado de desconocimiento sobre el proceso de software que emplean (MA-MRV-R1).

5- Sistema de Clonación (SistClon):

- Analista aún se encuentra haciendo el análisis para definir las historias de usuario.
- Queda cierto grado de desconocimiento sobre el proceso de software que emplean (MA-MRV-R1).
- Desorganización de la información ubicada en el SVN.
- Creación de la carpeta Resultados en el SVN.

6- MINPPAL:

- Presentan la carpeta Expediente de proyecto con toda la información pedida.
- Deben colocar la información de las demás carpetas que le faltan.
- Deben arreglar los errores que se le detectaron al documento de Concepción del sistema.

2.2.3 Tercera Iteración.

En esta iteración el uso de la propuesta MA-MRV-R1 se convierte en una realidad. La organización y conocimiento de los analistas y demás miembros del equipo, hicieron que los resultados en cada sistema fueran satisfactorios. En cada una de las carpetas que se encuentran en el SVN existe la información que debe tener cada uno de los proyectos. Se logró uniformar cada tarea en el cronograma de trabajo, además de la actualización diaria del Dotproject; es bueno resaltar que queda un pequeño número de miembros en cada equipo que deben poner al día sus horas trabajadas, pero cada gerente tiene que ser capaz de encargarse de la verificación de las tareas que realiza cada miembro del equipo.

Es notable el avance obtenido hasta el momento en que se ejecuta esta iteración, y aún así quedan asperezas que se deben llevar de cerca. De forma concreta como principales barreras para la adopción de prácticas ágiles se identifica el conocimiento y la experiencia en estas prácticas teniendo gran importancia el cambio organizacional que implican. El uso de la propuesta, siendo una nueva metodología para el conocimiento de los que la utilizan, hace que se sienta el desconocimiento a la hora de la interacción con los procedimientos que dicta.

1- Revista UXI:

- Organización en la información ubicada en el SVN.
- Mantiene actualizados los cambios realizados en la fase de mantenimiento en la plantilla de Gestión de Cambios.

2- Portal de SWL 2:

- La información colocada en el SVN tienen la organización pedida.
- Presentan todas las plantillas pedidas en su carpeta Expediente de proyecto además de su documento de Concepción del sistema bien redactado.

3- Tocororo Desktop:

- Presentan problemas con la organización de la información en el SVN.
- No han colocado las plantillas de historias de usuario.
- Desarrollares desvinculados de la programación de las plantillas de historias de usuario.
- Gerente indiferente a las tareas por realizar.
- Analistas dedicadas a otras tareas, sin realizar las tareas del proyecto.
- No tienen los cronogramas de trabajo.

4- ServiceDesk:

- Las historias de usuarios sacadas no coinciden con las puestas en cada una de las carpetas por iteraciones.
- Presentan una excelente organización de las carpetas en el SVN.
- Deben ser cuidadosos a la hora de llenar las plantillas de historias de usuario.

5- Sistema de Clonación (SistClon):

- La plantilla Concepción del sistema está bien elaborada.
- Excelente organización de la información guardada en la carpeta de expediente de proyecto.
- Uniformidad en la Arquitectura de la información definida.
- Las plantillas de historias de usuario están bien elaboradas.

6- MINPPAL:

- Servicio que concluyó de manera satisfactoria el proceso.
- Los miembros del equipo de desarrollo muestran la satisfacción sentida con la interacción con la propuesta realizada con SCRUM y XP.
- En la carpeta Expediente de proyecto tienen todos los artefactos que respaldan la propuesta, y fueron pedidos.
- La organización de la Arquitectura de la información es excepcional.

Como resultados de las iteraciones realizadas podemos decir que nos ayudó a ver las debilidades y ventajas que tendría la propuesta MA-MRV-R1 en su puesta en marcha en sistemas para el desarrollo de software. A pesar de que cada uno de los proyectos seleccionados presentaba diferentes características cada uno de ellos supo ajustarse a lo planteado y finalmente concluimos exitosamente. Por tanto, podemos concluir que:

- Se logró la integración de cada uno de los miembros del equipo de desarrollo.
- La motivación por el trabajo con las metodologías ágiles hizo que la mentalidad de todos los que la utilizaban se sintieran estimulados.
- Los productos terminados presentan muy buena calidad.
- Los desarrolladores se sienten a gusto desarrollando sin preocupación alguna para tener al día su documentación.
- Más desarrollo y menos tiempo empleado para la documentación es el lema del equipo.
- La integración del cliente al equipo de trabajo posibilitó mayor satisfacción por parte de este a la hora de las pequeñas y finales entregas del producto.
- Consolidamos y fortificamos la propuesta MA-MRV-R1.

- Se logró reunir todos los artefactos para trabajar la documentación que era necesaria tener.
- Se mantuvo la disciplina ante el reto dado, de cambiar a esta metodología de total desconocimiento para todos.
- En cada una de las reuniones planificadas, los desarrolladores mostraron todas las funcionalidades implementadas, lo que posibilitó que el usuario probara lo que se había obtenido.
- La entrega de pequeñas muestras del producto en cada una de las revisiones proporciona mayor agilidad y rapidez.
- Fomentó el trabajo de la propiedad colectiva para los implicados.

Destacar que una vez concluidas las revisiones de la puesta en práctica de la propuesta realizada por Malay solo se comprobó su viabilidad de manera efectiva en dos de sus fases. Las fases Planificación – Definición y Desarrollo fueron totalmente verificadas con cada uno de sus artefactos correspondiente, quedando las fases de Entrega y Mantenimiento.

Conclusiones

En este capítulo se realizó un estudio de los proyectos de igual tipo escogidos en el Capítulo 1, donde se aplican estas metodologías ágiles en el mundo, abordando algunas de las características principales de las mismas, describiendo brevemente sus procesos y haciendo mención de cada una de las empresas que las utilizan, con ejemplos que nos cuentan por sí solos las experiencias que se han tenido estos grupos en la adopción de esta unión, además de que se abordó sobre la documentación de todas las revisiones hechas en el proyecto con la puesta en práctica de la propuesta MA-MRV-R1.

CAPÍTULO 3 METODOLOGÍA ÁGIL MA-GMPR-UR2

Durante el transcurso de esta investigación se ha abordado temas relacionados con el estado actual de las metodologías ágiles de desarrollo como han sido sus características fundamentales, estadísticas en el año 2007 y los diferentes proyectos que están implicados en este contexto, en la utilización de la unión de XP y SCRUM, además se ha hecho un estudio del proceso productivo de 6 de los sistemas que se encuentran en el grupo UNICORNIOS, de la Facultad 10 basado en las principales características de los proyectos productivos.

La producción de software incluye aspectos importantes y determinantes, pero sin dudas y es a lo que menos se le presta atención es a una selección adecuada de la metodología a seguir para llevar adelante dicho proceso. En el año 2007 la Ing. Malay Rodríguez Villar atendiendo a esta problemática realizó una investigación realizada en la facultad 7 que arrojó como resultado la propuesta MA-MRV-R1. Propuesta que se puso en práctica en 6 sistemas escogidos del grupo UNICORNIOS de la facultad 10, de la Universidad de las Ciencias informáticas.

Precisamente es el objetivo de dicho capítulo dar solución a dicha investigación haciendo uso de la propuesta que fue realizada para llegar a la metodología MA-GMPR-UR2.

Ahora bien, para darle solución a lo antes planteado se propone explicar, guiar y orientar como quedaría el procedimiento ágil para ser usado en el proceso productivo.

Antes de comenzar con la metodología se deben esclarecer algunos términos que han sido empleados a lo largo de toda la investigación y que son un eslabón importante cuando se habla de hacer software. Uno de estos elementos es el proceso de software ¿Qué es? ¿Por qué es importante? ¿Cuáles son los pasos? ¿Cuál es el producto obtenido? Estas y otras preguntas surgen y es necesario darle respuesta.

Proceso de software

El proceso de software no es más que una serie de pasos predecibles a seguir – un mapa de carreteras que te ayude a obtener el resultado oportuno de calidad. Pero ¿Qué es exactamente desde un punto de vista técnico? Pues bien, es un marco de trabajo de las tareas que se requieren para construir software de alta calidad; define el enfoque que se toma cuando el software es tratado por la ingeniería. Su importancia está dada por el hecho de que proporciona estabilidad, control y a una actividad que puede, si no se controla, volverse caótica. Realmente a un nivel detallado el proceso que adoptemos depende del software que se esté construyendo. Ahora, desde el punto de vista de un ingeniero de software los productos obtenidos son programas, documentos y datos que se producen como consecuencia de las actividades de ingeniería de software definidas por el proceso. [27]

Gestión de proyectos

Por otra parte, la Gestión de Proyectos implica la planificación, supervisión y control no solo del proceso de software, sino también del personal y los eventos que ocurren mientras evoluciona el software desde la fase preliminar a la implementación operacional. Es importante debido a que la construcción del software es una empresa compleja, particularmente si participa mucha gente, trabajando durante un período de tiempo relativamente largo, lo que hace que necesiten ser gestionados. Los pasos principales se basan en la comprensión de las cuatro P (persona, producto, proceso y proyecto). El personal debe estar organizado para desarrollar el trabajo del software con efectividad, la comunicación con el cliente debe ocurrir para que se comprendan el alcance del producto y los requisitos, debe seleccionarse el proceso adecuado para el personal y el producto. El proyecto debe planificarse estimando el esfuerzo y el tiempo para cumplir las tareas, definiendo los productos del trabajo, estableciendo puntos de control de calidad y estableciendo mecanismos para controlar y supervisar el trabajo definido en la planificación. Lo que se obtiene del mismo es un plan de proyecto que se realiza al comienzo de la actividad de gestión, el plan define el proceso y las tareas a realizar el personal que realizará el trabajo y los mecanismos para evaluar los riesgos, controlar el cambio y evaluar la calidad.

Ingeniería de software

La ingeniería de software es la aplicación de un enfoque sistémico, disciplinado y cuantificable al desarrollo, operación (funcionamiento) y mantenimiento del software; es decir, la aplicación de ingeniería al software. 2. El estudio de los principios y metodologías para desarrollo y mantenimiento de sistemas de software. [28]

El trabajo que se asocia en la ingeniería de software se puede dividir en tres fases genéricas, con independencia del área de aplicación, tamaño o complejidad del proyecto las mismas son:

1. Fase de Definición: Se centra sobre el qué, aquí se intenta que información ha de ser procesada, que función y rendimiento se desea, que comportamiento del sistema, que interfaces va a ser establecidas, que duración se necesita para definir un sistema correcto, en fin, han de identificarse los requisitos claves del sistemas y del software. Cuenta con tres tareas principales: ingeniería de sistemas o de información, planificación del proyecto y análisis de los requisitos.

2. Fase de Desarrollo: Se centra en el cómo, se intenta definir cómo han de diseñarse las estructuras de datos, cómo ha de implementarse la función dentro de una arquitectura de software, cómo han de implementarse los detalles de los procedimientos, cómo han de caracterizarse interfaces, cómo ha de traducirse el diseño en un lenguaje de programación y cómo ha de realizarse la prueba. Cuenta con tres tareas: diseño del software, generación de código y prueba del software.

3. Fase de Mantenimiento: Se centra en el cambio que va asociado a la corrección de errores, a las adaptaciones requeridas a medida que va evolucionando el entorno del software y a cambios debido a las mejoras producidas por los requisitos cambiantes del cliente. Tipos de cambios: Corrección, Adaptación, Mejora, Prevención.

3.1 PROCEDIMIENTOS

Basados en los conceptos anteriores se procederá a plantear y explicar la metodología de

los procedimientos ágiles que sustentaran la gestión de los proyectos y la ingeniería de software a seguir por los mismos.

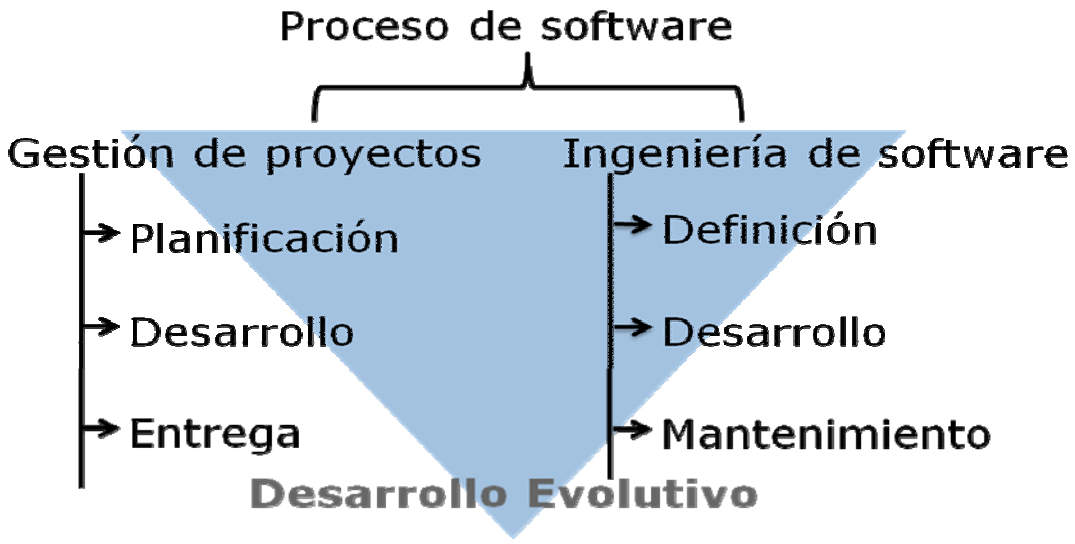

Figura 3. Esquema de la metodología MA-GMPR-UR2.

3.1.1 Gestión del proyecto.

Antes de hacer la presentación Inicial del proyecto es necesario conversar con el usuario/cliente de que se trata el proyecto y se sugiere que se tenga en cuenta los siguientes aspectos:

- Usar la planificación Inicial de Proyecto como elemento mínimo de la planificación del proyecto completo.

- Explicar al cliente/usuario y dejar constancia en la definición técnica que parte del enfoque de solución del proyecto incluye usar métodos ágiles, mencionando de cuales se tratan y que el modelo de ciclo de vida será desarrollo evolutivo.

- El detalle de la planificación se realiza en cada iteración.

En la metodología, se usará SCRUM para la planificación de los proyectos que usarán métodos ágiles como metodología para su proceso de desarrollo pues SCRUM es una forma de gestionar proyectos de software, no es una metodología de análisis, ni de diseño, es una metodología de gestión del trabajo. Aquí no se pretende implantar dicha metodología

en su totalidad, de la misma serán tomadas algunas prácticas para su implantación, describiéndolo a continuación.

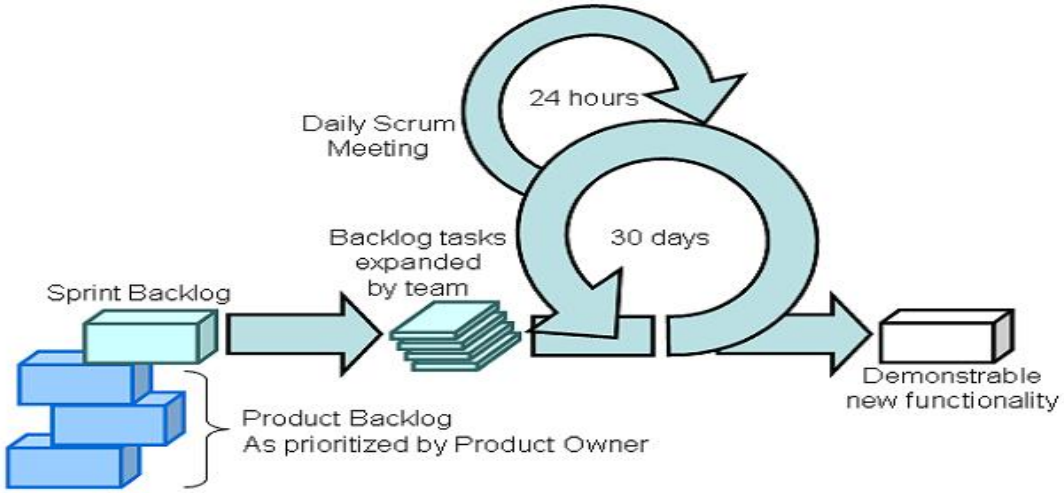

Figura 4. Proceso de desarrollo de SCRUM.

3.1.1.1 Planificación.

Propósito**:** Establecer la visión, fijar expectativas, y asegurar financiamiento.

Entrada: la concepción inicial del producto.

Actividades**:** Escribir la visión (Plantilla de Concepción del Sistema), presupuesto, Reserva del producto con estimaciones iniciales.

- Crear la Lista de Reserva del Producto (LRP) y controlar su consistencia: Posibles elementos de esta lista son requerimientos técnicos y del negocio, funciones, y actualizaciones tecnológicas requeridas.

Es importante controlar la consistencia de la lista. Para esto se agregan, modifican, eliminan, especifican y priorizan sus elementos.

Plantilla Concepción del sistema (Ver anexo 1)

En esta plantilla se refleja la visión general del producto a implementar, recoge los diferentes roles que intervendrán en el desarrollo del software, así como las tecnologías usadas.

Lista de Reserva del Producto (LRP) (Ver anexo 2)

Es una lista priorizada que define el trabajo que se va a realizar en el proyecto. Cuando un proyecto comienza es muy difícil tener claro todos los requerimientos sobre el producto. Sin embargo, suelen surgir los más importantes que casi siempre son más que suficientes para un Sprint (Iteración).

Esta lista puede crecer y modificarse a medida que se obtiene más conocimiento acerca del producto y del cliente. Con la restricción de que solo puede cambiarse entre Sprint. El objetivo es asegurar que el producto definido al terminar la lista es el más correcto, útil y competitivo posible y para esto la lista debe acompañar los cambios en el entorno y el producto.

Roles: Jefe de proyecto, Gerente, Cliente y Equipo de proyecto.

Sprint (Iteración)

Un Sprint es el procedimiento de adaptación de las cambiantes variables del entorno (requerimientos, tiempo, recursos, conocimiento, tecnología). Son ciclos iterativos en los cuales se desarrolla o mejora una funcionalidad para producir nuevos incrementos. Durante un Sprint el producto es diseñado, codificado y probado. Y su arquitectura y diseño evolucionan durante el desarrollo.

El objetivo de un Sprint debe ser expresado en pocas palabras para que sea fácil de recordar y esté siempre presente en el equipo. Es posible definir una serie de restricciones que el equipo deba aplicar durante un Sprint, por ejemplo: un Sprint tiene una duración planificada de entre una semana y un mes. No es posible introducir cambios durante el Sprint, por lo tanto para planificar su duración hay que pensar en cuanto tiempo se puede comprometer a mantener los cambios fuera del Sprint. Dependiendo del tamaño del sistema, la construcción de un release puede llevar entre 3 y 8 Sprint. Por otra parte podrían formarse equipos para desarrollar en forma paralela distintos grupos de funcionalidad.

- Priorizar la Lista de reserva del producto: Esta actividad se basa en considerar que elementos tienen más o menos influencia en el éxito del proyecto en un momento dado; considerando que los elementos con mayor prioridad se realizan primero.

Roles: Cliente.

- Valoración del esfuerzo: Es un proceso iterativo que reúne toda la información que haya acerca de un elemento para tener un mayor nivel de precisión en la estimación. Siempre se mide el esfuerzo que falta para cumplir con el o los objetivos tanto a nivel de la lista de reserva del producto como para la Reserva del Sprint.

Se actualiza en la plantilla Historia de Usuario los campos de los puntos estimados y reales. Donde se recoge el tiempo en semanas que se le asigna a la historia de usuario, además del tiempo real dedicado a la misma en semanas.

- Valoración de riesgos: En esta actividad se debe definir desde un inicio todos los riesgos que pueda atravesar un proyecto. Estos riesgos son las posibles causas al fracaso a tener en cuenta por el líder de proyecto y el Gerente mientras dure el proceso de desarrollo. Quedando de este modo todos alertas.

Plantilla Lista de riesgos (Ver Anexo 3).

En ella son definidos los posibles riesgos que actuarán sobre el proceso de desarrollo de software, así como la estrategia trazada para mitigarlos.

Nota: Esta valoración no puede exceder a un mes.

Roles: Gerente.

- Reunión de Revisión del diseño: En esta instancia se comunica el diseño a los interesados para revisar el cumplimiento de los ítems especificados en la Reserva del producto.

Salida: Lista de Reserva del Producto (LRP), Arquitectura.

3.1.1.2 Desarrollo.

Propósito**:** Implementar un sistema listo para entrega en una serie de iteraciones de 60 días. (El tiempo puede decrementarse a la medida que se está refinando el producto).

Entrada**:** Lista de Reserva del Producto (LRP).

Actividades**:** Reunión de planificación de la iteración. Definir la reserva de la iteración. Reuniones de coordinación, revisión de la iteración y la realización de las pruebas de aceptación del sistema.

- Junta de planificación del Sprint: Es una reunión organizada por el Líder del Proyecto, que se realiza en dos fases.

  ✓ La primera fase tiene como objetivo establecer que ítems de la Lista de Reserva del Producto van a ser realizados durante el Sprint. Esto se realiza a partir de lo que el Equipo considera que puede construir durante el Sprint.

  ✓ En la segunda fase se decide como se van a alcanzar los objetivos del Sprint. En esta fase se crea la Reserva del Sprint, indicando qué tareas debe desempeñar el equipo para cumplir con dichos objetivos.

Generando la plantilla Tareas de ingeniería, además de la plantilla Cronograma de producción.

Plantilla de Tareas de Ingeniería (Ver Anexo 4).

En esta plantilla se recogen las tareas por historias de usuario a realizar.

Plantilla Cronograma de producción (Ver Anexo 5).

El cronograma de producción no es más que la plantilla que recoge las actividades realizadas en el equipo de desarrollo durante la iteración. Se realiza un cronograma para cada iteración planificada durante el proceso.

- Junta de seguimiento: Encuentro para hacer los chequeos de lo que se está implementando. Las reuniones se realizan en el mismo lugar y a la misma hora una vez por semana, idealmente a principios de semana para definir el trabajo para el día. Tienen una duración de 30 minutos, no son para resolver problemas, en ellas se realizan tres preguntas:

  · ¿Qué hiciste? (Destacar que el trabajo que se hizo para la consecución del objetivo es válido).

  · ¿Qué harás? (Al salir de la reunión todo el mundo sabe lo que tiene que hacer y todos están alineados. Mucha gente no sabe organizarse y pierde el tiempo en cosas menos importantes por lo que esto le sirve para organizar su trabajo diario).

  · ¿Qué obstáculos ves en tu camino? (Es el momento para que salga a la luz todos los problemas que tienen las personas para la realización de su trabajo).

Sugerencia: Estas reuniones no pueden ser sustituidas por reportes vía e-mail por dos motivos:

- El equipo entero ve todo el paisaje cada día.
- Es un elemento de presión para que el individuo haga lo que dijo que va a hacer.

Roles: Miembros del Proyecto.

- Taller técnico: Se evacuan las dudas que tienen cada uno de los participantes del equipo de desarrollo, y se toman decisiones técnicas. Es una reunión informal que no puede tomar más de 3 horas.

- Junta de revisión: Es una reunión informal que tiene como regla que su preparación no puede tomar más de 2 horas. En ella el equipo presenta lo que ha logrado durante el Sprint. Generalmente toma la forma de una demo de las nuevas características o la arquitectura.

Roles: Cliente, Gerente, Cliente, otros interesados.

Salida: Incremento del producto.

3.1.1.3 Entrega.

Propósito: Puesta en Operación

Actividades**:** Documentación, Entrenamiento, Marketing.

Contiene el cierre del release. Para ingresar a esta fase se debe llegar a un acuerdo respecto a las variables del entorno por ejemplo que los requerimientos fueron completados. El sistema está listo para ser liberado y es en esta etapa en la que se realiza integración, la entrega de las pruebas del sistema y la documentación en general.

Salida: Producto.

- Entrega de la documentación: Es una reunión organizada por el Líder del Proyecto, el Cliente y otros interesados donde se hace la entrega de la documentación del sistema implementado.

- Entrenamiento: No es más que la actividad realizada para darle capacitación al Cliente y otros interesados que utilizarán el producto.

Se generan 3 artefactos:

1- Plantilla Manual de usuario.

En esta plantilla el desarrollador realiza una guía de los pasos necesarios y de conceptos de aspectos importantes con los que cada uno de los usuarios podrá consultar en caso de dudas para interactuar con el sistema.

2- Plantilla Manual de Identidad.

El diseñador debe hacer una caracterización general de los aspectos que se tomaron en cuanta para la realización del diseño del sistema.

3- Plantilla Manual de desarrollo.

Es una plantilla realizada por los desarrolladores, donde se recoge todo lo relacionado con las pautas realizadas para la programación del sistema.

Nota: Esta plantilla se genera automáticamente.

• Instalación: Montaje y adiestramiento a los encargados en la entidad cliente que interactuarán como administradores del sistema desarrollado.

• Marketing: Técnicas empleadas para lograr la inserción del producto a través de estudios de mercado, donde se intenta lograr el máximo beneficio en la negociación del producto, y saber a qué tipo de público le interesa su producto. Esto se logra con la divulgación y publicación de datos relacionados con el producto que se desarrollo.

Roles: Programadores, Diseñadores, Gerente, Líder del Proyecto, Cliente

3.1.1.4 Definición de Roles.

• Líder del Proyecto (Scrum Master)

Es un rol de administración que debe asegurar que el proyecto se está llevando a cabo de acuerdo con las prácticas y que todo funciona según lo planeado. Su principal trabajo es remover impedimentos y reducir riesgos del producto.

– Coordinar y facilitar las reuniones.

– Asegurar que se consigue los objetivos de la reunión de planificación de la iteración.

– Determina cuándo es necesario realizar algún cambio para lograr los objetivos de cada iteración.

• Gerente (Management)

Es el responsable de tomar las decisiones finales, acerca de estándares y convenciones a seguir durante el proyecto. Participa en la selección de objetivos y requerimientos y en la selección del Usuario Interno. Tiene la responsabilidad de controlar el progreso y trabaja junto con el Jefe de Proyecto en la reducción de la Lista de Reserva del Producto.

– Realiza el seguimiento del progreso de cada iteración.

– Verifica el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, comunicando los resultados para mejorar futuras estimaciones.

– Evalúa si los objetivos son alcanzables con las restricciones de tiempo y recursos presentes.

– Puede ocupar el rol de programador master (líder de desarrollo) durante la etapa de desarrollo.

• Especialistas

Es el responsable del proceso global. Es necesario que conozca a fondo el proceso, ya sea de la metodología utilizada o cualquier otro proceso o elementos de gran importancia para el desarrollo de software. Particularmente es una especialización que está activa, el miembro del grupo de trabajo que la desempeña siempre está ejecutándola y alcanzando un grado mayor de conocimientos en el tema. Ejemplos: Ingenieros de Software, Especialistas de Servicios (migración a software libre, jefe del consejo editorial), Especialistas en diseño gráfico, etc.

• Consultor

Es un miembro externo del equipo con un conocimiento especifico en algún tema necesario para el proyecto, en el que puedan sugerir problemas, además aportan ideas y experiencias para el beneficio del sistema en desarrollo. Esta es una especialización menos activa, quien

la ejecuta funciona en este rol por un corto período de tiempo. Está más orientada a roles de desarrollo, y sus ejecutores pueden trabajar en otros roles (en otro equipo) durante el desarrollo del software. Ejemplos: Consultor de desarrollo Web, consultor de desarrollo de escritorio, diseñador de base de datos, etc.

• Cliente (Customer)

El cliente participa en las tareas que involucran la lista de reserva del producto.

- Presentar la Reserva del producto al equipo, enfatizando el valor y prioridades del mismo.
- Definir la meta de la iteración.
- Aprobar las modificaciones en la Reserva del producto y en el alcance de la iteración.

• Miembros del Proyecto (Scrum Team)

Es el equipo del proyecto que tiene la autoridad para decidir cómo organizarse para cumplir con los objetivos de un Sprint. Sus tareas son: Estimar esfuerzo, crear la reserva del Sprint, revisar la Lista de Reserva del Producto y sugerir obstáculos que deban ser removidos para cumplir con los ítems que aparecen.

Típicamente es un equipo de entre 5 y 10 personas cada una especializada en algún elemento que conforma los objetivos a cumplir, por ejemplo: Programadores, Diseñadores de Interfaz de usuario, etc. La dedicación de los miembros del equipo debería ser full-time con algunas excepciones. La membrecía solo puede cambiar entre sprints (no durante).

• Programadores (Programmers)

Es el encargado de producir el código y escribir las pruebas unitarias. Debe existir una comunicación y coordinación adecuada entre los programadores y otros miembros del equipo.

• Analista (Analyst)

Es el encargado de escribir 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 centrándose en aportar mayor valor al negocio, todo esto lo

realiza junto con el cliente.

• Diseñadores (Designers)

Encargados del diseño del sistema; así como el de los prototipos de interfaces, máximos responsables de la realización del diseño de las metáforas y supervisan el proceso de construcción.

• Encargado de Pruebas (Tester)

Es el encargado de ayudar 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.

• Arquitecto (Architect)

Se vincula directamente con el analista y el diseñador debido a que su trabajo tiene que ver con la estructura y el diseño en grande del sistema. Ayuda en el diseño de las metáforas.

3.1.1.5 Planificación de una Iteración.

Propósito de la Iteración

• Transformar un subconjunto de la Reserva del producto en un incremento en la funcionalidad del producto que sea potencialmente entregable a los usuarios. Para ello, Usuario Interno, el Líder del Proyecto y los Miembros del Proyecto se juntan, previamente a cada iteración, para determinar en qué funcionalidad del producto trabajará el equipo durante la próxima iteración; esta reunión se denomina “Reunión de Planificación de la Iteración” y cuenta con dos partes:

Parte 1 de Planificación de la Iteración

• Objetivo: El equipo define una meta para la iteración, así como también selecciona las características de la Reserva del producto que van a ser planificadas para conseguir la meta propuesta.

• Duración: 1 a 4 horas dependiendo de qué tan obvio es el trabajo que hay que desarrollar durante la próxima iteración.

• Entradas:

– El producto resultante de la iteración previa.

– La reserva del producto priorizado.

– El estado actual de la tecnología y las condiciones del negocio.

Parte 2 de Planificación de la Iteración

• Objetivo: Definir el trabajo que es necesario desarrollar durante la iteración. Esto da origen la Reserva de la Iteración.

• Duración: 4 horas.

• Salidas:

– Reserva de la Iteración: Lista de tareas de no más de 16 horas cada una.

– Arquitecturas y Diseños.

– Asignación de recursos a cada tarea.

– Fechas para la reunión de revisión de los resultados de la iteración, y para la próxima reunión de planificación.

Esto se recoge en la plantilla de Plan de Releases.

Plantilla de Plan de Releases (Ver Anexo 6)

En esta plantilla se recogen las iteraciones a realizar con sus características, además del orden de las historias de usuario con su planificación estimada para ser implementadas.

3.1.2 Ingeniería de software.

Para llevar a cabo el proceso de desarrollo del proyecto se tomará en cuenta las mejores prácticas de la metodología XP, procurando que el proceso sea efectivo y eficiente.

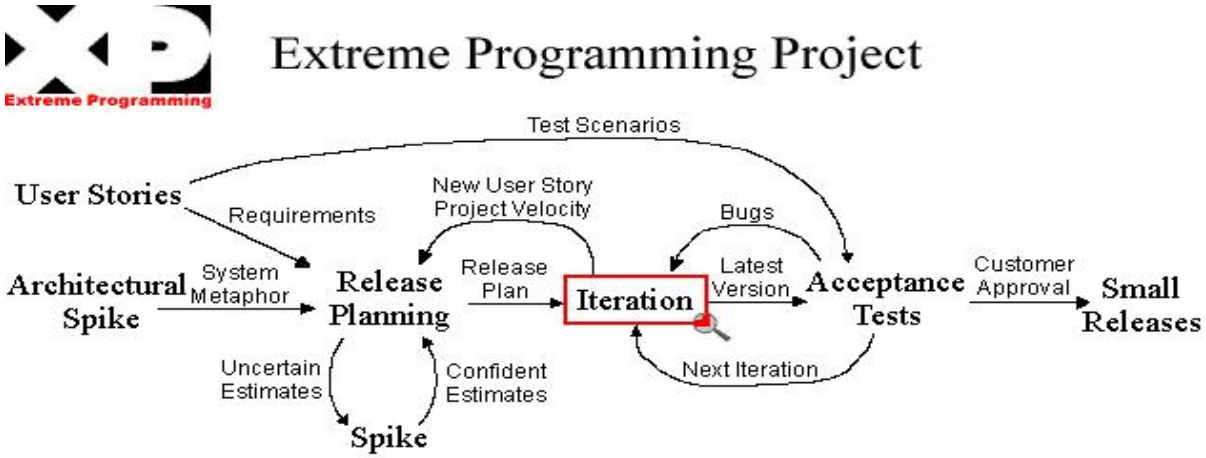

Figura 5. Modelo XP (Programación Extrema).

3.1.2.1 Definición

La planificación

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.

1. Cada entrega es lo más corta posible:
    - Contenga requisitos más valiosos del sistema (básicos).
    - Reducen el riesgo: mayor retroalimentación desde el cliente, y más frecuente.
2. Minimizar el número de historias de usuarios que componen una entrega: No realizar historias de usuarios a medias.
3. A cada programador se le asigna una tarea de la historia de usuario.

Historias de usuario (User Stories) (Ver anexo 7)

Las historias de usuario son la técnica utilizada en XP para especificar los requisitos del software, lo que equivaldría a los casos de uso en el proceso unificado. Las mismas son escritas por los clientes como las tareas que el sistema debe hacer y su construcción depende principalmente de la habilidad que tenga el cliente para definirlas. Son utilizadas

como el único documento de requisitos que se genera en XP. Son escritas en lenguaje natural, sin un formato predeterminado, no excediendo su tamaño de unas pocas líneas de texto.

Las historias de usuario guían la construcción de los tests de aceptación, elemento clave en XP (deben generarse uno o más tests para verificar que la *story* ha sido correctamente implementada) y son utilizadas para estimar tiempos de desarrollo. En este sentido, sólo proveen detalle suficiente para hacer una estimación razonable del tiempo que llevara implementarla. En el momento de implementar se deben detallar a través de la comunicación con el cliente. Son la base para las pruebas funcionales.

El juego de la planificación.

Es un espacio frecuente de comunicación entre el cliente y los programadores. 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. Esta práctica se puede ilustrar como un juego, donde existen dos tipos de jugadores: Cliente y Programador. El cliente establece la prioridad de cada historia de usuario, de acuerdo con el valor que aporta para el negocio. Los programadores estiman el esfuerzo asociado a cada historia de usuario. Se ordenan las historias de usuario según prioridad y esfuerzo, y se define el contenido de la entrega y/o iteración, apostando por enfrentar lo de más valor y riesgo cuanto antes. Este juego se realiza durante la planificación de la entrega, en la planificación de cada iteración y cuando sea necesario reconducir el proyecto.

1. Decisiones de negocio (cliente):
    - **Alcance:** ¿Cuándo debe estar listo el producto para que sea valioso en producción?
    - **Prioridad:** Prioriza la incorporación de las historias de usuario.
    - **Composición de entregas:** ¿Qué se necesita para que el negocio sea mejor antes de tener el software?
    - **Fechas de entrega:** Fechas cuando el software funcionando causaría una gran diferencia.
2. Decisiones técnicas (programadores y otros):
    - **Estimaciones**: ¿Cuánto tiempo tardará en implementarse una historia de

usuario?

- **Consecuencias:** Tener en cuenta las consecuencias técnicas de determinadas decisiones de negocio.
- **Proceso:** Organización del proceso y el equipo.
- **Planificación detallada:** Dentro de una entrega, qué historias de usuario se realizan primero. Intentar trasladar los segmentos de desarrollo más arriesgados al principio, intentando respetar las prioridades del negocio.

3. Reunión diaria "Stand-up Meeting"

- Todo el equipo.
  - ✓ Problemas.
  - ✓ Soluciones.
- De pie en un círculo.
  - ✓ Evitar discusiones largas.
  - ✓ Sin conversaciones separadas.

Generándose la plantilla Modelo de Historia de Usuario del negocio.

Plantilla Modelo de Historia de Usuario del negocio (Ver Anexo 8).

En esta plantilla se modela las características específicas del negocio, así como la forma en que interactúa el sistema con los clientes y viceversa.

Roles: Analista y Arquitecto.

Captura de requisitos:

1. Historias del Usuario (*User-Stories*)

- Establecen los requisitos del cliente.
- Trozos de funcionalidad que aportan valor.
- Se les asignan la persona encargada de la programación con un número de horas de desarrollo.
- Las establece el cliente.

Para hacer la asignación de las historias de usuario se debe actualizar en la plantilla el campo “Usuario”.

Roles: Cliente, Jefe de proyecto, Gerente.

Diseño

Diseño simple

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. Kent Beck dice que en cualquier momento el diseño adecuado para el software es aquel que: supera con éxito todas las pruebas, no tiene lógica duplicada, refleja claramente la intención de implementación de los programadores y tiene el menor número posible de clases y métodos.

1. Se diseña “la cosa más simple que pueda funcionar”.
2. Modelo de diseño.
3. Diseño de software correcto, es aquel que:
    - Supera todas las pruebas.
    - No tiene lógica duplicada.
    - Pone de manifiesto las intenciones importantes de los programadores.
    - Tiene el mínimo número de clases y métodos.

En la propuesta el uso de las tarjetas CRC se tiene en cuenta para ayudar a identificar jerarquías de generalización/especificación, o jerarquías de agregación entre las clases. Aspecto que en muchos de los sistemas que se seleccionaron para realizar el estudio con la puesta en práctica de la propuesta elaborada no es analizable debido a que son sistemas que están compuestos por numerosos script, como es el caso de los proyectos que desarrollan sistemas Web. Razón por la cual en la metodología MA-GMPR-UR2 se decidió cambiar esta plantilla por la realización de la plantilla Modelo de diseño, realizando solo el modelando del diagrama de paquetes, respondiendo así a uno de los lineamientos que exige Calidad.

Plantilla Modelo de diseño (Ver Anexo 9).

Se define en esta plantilla, un esbozo inicial del diseño del sistema, sin entrar en especificaciones, ni detalles, solo lo que el diseñador necesita para hacer un primer entregable del sistema.

Roles: Diseñador.

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. Martin Fowler en *Is Design Dead?* [29] explica que 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 al diseño y métodos del sistema.

1. Da un contexto al equipo para entender los elementos básicos y sus relaciones.
2. Proporciona integridad conceptual.

Refactorización

La refactorización es una actividad constante de reestructuración 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 [30]. Robert Martin [31] señala que el diseño del sistema de software es una cosa viviente. No se puede imponer todo en un inicio, pero en el transcurso del tiempo este diseño evoluciona conforme cambia la funcionalidad del sistema. Para mantener un diseño apropiado, es necesario realizar actividades de cuidado continuo durante el ciclo de vida del proyecto. De hecho, este cuidado continuo sobre el diseño es incluso más importante que el diseño inicial. Un concepto pobre al inicio puede ser corregido con esta actividad continua, pero sin ella, un buen diseño inicial se degradará.

1. Refactorización = Mejora del código.

2. Intentar eliminar complejidad.
3. Código duplicado: Refactorización.
4. Se plantea su aplicación después de implementar cada historia de usuario.

Roles: Líder del proyecto, Gerente, Programador, Consultor, Cliente.

3.1.2.2 Desarrollo

Codificación

Programación en pareja

Toda la producción de código debe realizarse con trabajo en parejas de programadores. Según Cockburn y Williams en un estudio realizado para identificar los costos y beneficios de la programación en parejas [32], las principales ventajas de introducir este estilo de programación son: muchos errores son detectados conforme son introducidos en el código (inspecciones de código continuas), por consiguiente la tasa de errores del producto final es más baja, los diseños son mejores y el tamaño del código menor (continua discusión de ideas de los programadores), los problemas de programación se resuelven más rápido, se posibilita la transferencia de conocimientos de programación entre los miembros del equipo, varias personas entienden las diferentes partes del sistema, los programadores conversan mejorando así el flujo de información y la dinámica del equipo, y finalmente, los programadores disfrutan más su trabajo. Dichos beneficios se consiguen después de varios meses de practicar la programación en parejas.

1. Todo el código se escribe en parejas.
    - Se produce código de mayor calidad.
    - Extiende el conocimiento.
2. Se realiza el trabajo de 1 persona en casi la mitad del tiempo y mejor (cuestionable).

Propiedad colectiva

Cualquier programador puede 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.

1. Cualquiera puede modificar el código en cualquier momento. Se evitan cuellos de botella en la codificación.
2. Todos asumen las responsabilidades sobre el conjunto del sistema.
3. Todos conocen algo sobre todas las partes y muy bien aquellas en las que trabajan.

Integración continúa

Cada pieza de código es integrada en el sistema una vez que esté lista. Así, el sistema puede llegar a ser integrado y construido varias veces en un mismo día. Todas las pruebas son ejecutadas y tienen que ser aprobadas 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. Martin Fowler en *Is Design Dead?* [33] afirma que el desarrollo de un proceso disciplinado y automatizado es esencial para un proyecto controlado, 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.

1. El código se integra y se prueba después de pocas horas.
2. Existe un ordenador dedicado para la integración.
3. Cada pareja integra su código en dicho ordenador.

Cliente en sitio

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 *Extreme Programming Installed* [34] Jeffries indica que se debe pagar un precio por perder la oportunidad de un cliente con alta disponibilidad. Algunas recomendaciones propuestas para dicha situación son las siguientes: intentar conseguir un representante que pueda estar siempre disponible y que actúe como interlocutor del cliente, contar con el cliente al menos en las reuniones de planificación, establecer visitas frecuentes de los programadores al cliente para validar el sistema, anticiparse a los problemas asociados estableciendo llamadas telefónicas frecuentes y conferencias, reforzando el compromiso de trabajo en equipo.

1. Cliente real = Aquel que usará el sistema cuando esté en producción
2. El cliente real debe estar con el equipo de trabajo:
    - Responder preguntas.
    - Resolver disputas.
    - Establecer prioridades.
    - Discutir mejoras.

Estándares de programación.

Enfatizar 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.

1. Son fundamentales cuando los programadores cambian de pareja o hacen *refactoring* del código de otros.
2. Se consigue un código con el mismo estilo, homogéneo, legible.

De esta fase se genera la plantilla de Estándares de programación.

Plantilla de Estándares de programación

El documento recoge el estándar utilizado y su explicación. No existe una plantilla definida para recoger esta información, debido a que cada estándar tiene sus características específicas.

Roles: Programador, Cliente.

Prueba

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. Los clientes escriben las pruebas funcionales para cada historia de usuario que deba validarse. En este contexto de desarrollo evolutivo y de énfasis en pruebas constantes, la automatización para apoyar esta actividad es crucial.

1. Las pruebas unitarias se escriben ANTES que el código.
2. Pruebas automatizadas.
3. Permiten el desarrollo de proyectos de forma rápida y segura.
4. Pruebas unitarias: programadores.
5. Pruebas funcionales: cliente.
6. Resultado: Un programa cada vez más seguro.

Roles: Encargado de pruebas, Programador.

Recomendación General: *Espacio de trabajo:* Espacio abierto

Ahora bien, el mayor beneficio de las prácticas se consigue con su aplicación conjunta y equilibrada puesto que se apoyan unas en otras. Esto se ilustra en la Figura 3 (obtenida de [35]), donde una línea entre dos prácticas significa que las dos prácticas se refuerzan entre sí.

La mayoría de las prácticas propuestas no son novedosas sino que en alguna forma ya habían sido propuestas en ingeniería del software e incluso demostrado su valor en la práctica. El mérito de XP es integrarlas de una forma efectiva y complementarlas con otras ideas desde la perspectiva del negocio, los valores humanos y el trabajo en equipo.

Generándose el artefacto plantilla Caso de Prueba de aceptación.

Plantilla Caso de Prueba de aceptación (Ver Anexo 10).

En esta plantilla el desarrollador escribe las pruebas realizadas según la historia de usuario seleccionada para realizar la comprobación y validar las funcionalidades del sistema, y de esta forma saber si esta apto para ser liberado.

Roles: Programador, Cliente, Encargado de pruebas.

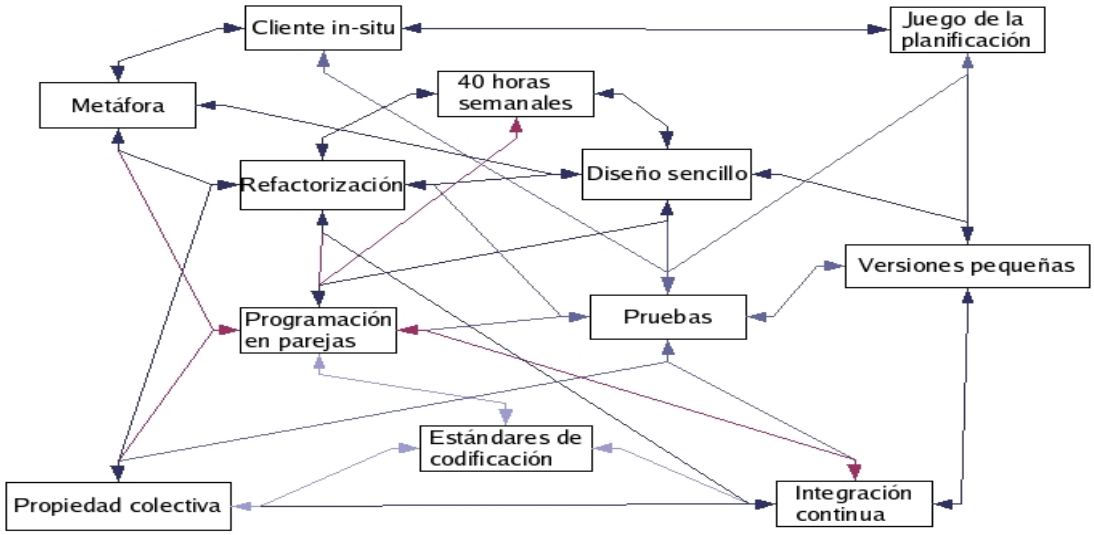

Figura 6. Las prácticas se refuerzan entre sí.

3.1.2.3 Mantenimiento

Mientras la primera versión se encuentra en producción, el proyecto 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.

De ahí surge la necesidad de guardar todos los cambios que van siendo realizados en el sistema desarrollado como constancia de arreglos en el código. Estos cambios se registran en la plantilla Gestión de Cambios.

Plantilla de Gestión de Cambios (Ver anexo 11).

Esta plantilla guarda los cambios realizados en el sistema una vez es terminado el producto y se pasa a la fase donde se mantiene dándole soporte al cliente. Recoge los datos para identificar el nombre de la historia de usuario de donde se deriva, quien realiza el cambio y el lugar donde ocurre la actualización en el código.

Roles: Programador, Cliente, Gerente, Líder de Proyecto.

3.2 Resultados Prácticos.

Esto es solo los primeros pasos de esta gran tarea de revolucionar el proceso productivo y llevarlos al enfoque ágil, solo fueron comprobadas dos de las fases del proceso, la fase de Planificación – Definición y Desarrollo demostrando con resultados positivos la efectividad de la aplicación de cada uno de los artefactos que se generaban según las diferentes tareas que se realizaban; por lo que no se cuenta con los datos concretos del índice de éxito de las restantes fases, Entrega y Mantenimiento.

En cada uno de los sistemas que fueron seleccionados se obtuvieron resultados significativos y otros un tanto preocupantes que permitieron centrar la metodología terminada y así evitar posibles tareas que no respondían a las necesidades según las características de los proyectos pertenecientes al Polo de Software Libre.

- Se logró que la organización de la documentación de cada uno de los sistemas fuera eficiente, además de que cada uno de los miembros del equipo se mostraba interesado y motivado para el desarrollo. El trabajo era más ágil y mantener al cliente dentro del equipo de desarrollo proporcionó mejores resultados y una mayor satisfacción por parte de los interesados finales del producto.
- Con la utilización de SCRUM para la gestión, se logra una planificación y organización inigualable; mientras que XP respalda con sus prácticas todo el proceso de desarrollo, obteniéndose de esta forma un proceso de software completo.
- Mostrar interés en el desarrollo y no en la documentación a llenar priorizó el tiempo para que este fuera empleado por el equipo en otras tareas de importancia, tales así como la atención a las tareas docentes, de estudio independiente entre otras.
- La generación de los artefactos necesarios e imprescindibles respalda la documentación de cada uno de los sistemas, logrando así que no queden sistemas sin ser analizados y documentados.
- Con los valores y principios que respaldan las metodologías ágiles se fomentó en cada uno de los miembros del equipo la unidad, el colectivismo y colaboración.

Estos resultados constituyen el objetivo principal para el desarrollo de esta investigación, en busca de un procedimiento ágil factible para ser puesto en práctica en proyectos productivos pertenecientes al Polo de Software Libre.

3.3 Valoración

Con esta metodología se pretende dar solución a muchos de los problemas que afectan el desarrollo y adecuada ejecución de los proyectos, debido al comportamiento de los estudiantes, la motivación y entrega de los mismos; pues al llevarse un control de que se debe hacer y cuando debe entregarse una determinada tarea, el nivel de preocupación y responsabilidad de los miembros del equipo aumenta. Cuando un estudiante tiene que prepararse para desarrollar una tarea de gran envergadura consume más tiempo en su preparación que si dicha tarea es dividida en sub tareas que viabilicen con mayor facilidad su aprendizaje y le posibiliten avanzar. Además, el líder de proyecto puede llevar un mejor control de las tareas y la planificación de las mismas. Asimismo reconocer la tendencia al compañerismo y solidaridad, no dejando margen al egoísmo e individualidad. Involucrar a los miembros del equipo de desarrollo en las decisiones sobre el proyecto y sus vías de desarrollo, puede ser de gran ayuda a la hora de aumentar la motivación, sin dejar fuera que se logra una mayor interacción con el cliente al ser parte del equipo. Proporcionando una mejor calidad en el producto a entregar.

Siempre se debe tener en cuenta que una metodología ágil puede ser una buena forma de empezar cuando no existe un proceso o existe pero no reacciona bien a los cambios o existe pero el equipo no está contento con él. Además, es fácil de financiar, los programadores se sienten cómodos y al cliente le agrada el valor añadido.

Conclusiones

En este capítulo se confeccionó una metodología de procedimientos ágiles teniendo en cuenta las características de los proyectos que se desarrollan en el grupo UNICORNIOS, partiendo de las metodologías SCRUM y XP todo sobre la base del proceso de software y sus dos vertientes: La gestión de proyectos y la ingeniería de software.

CONCLUSIONES

A partir de los objetivos planteados y el trabajo realizado en esta investigación donde se realizó una metodología de procedimientos ágiles, para los proyectos productivos pertenecientes al Polo de Software Libre, en particular al grupo UNICORNIOS, de la Facultad 10, se arribó a las siguientes conclusiones.

- El estudio del estado en que se encuentran internacionalmente las metodologías ágiles XP y SCRUM, se demostró que son los procedimientos ágiles más utilizados en el proceso de desarrollo de software según sus características.
- La caracterización de los proyectos productivos en el grupo UNICORNIOS del Polo de Software Libre, se identificó el conocimiento de sus peculiaridades, así como las necesidades de los mismos en cuanto a la documentación.
- La revisión de los proyectos seleccionados facilitó reconocer las dificultades y ventajas de la propuesta MA-MRV-R1, además de una mejor visión sobre el trabajo con este tipo de metodología.
- El estudio de los resultados de cada una de las revisiones realizadas en las iteraciones se logró un procedimiento ágil para proyectos de Software Libre.

El procedimiento ágil MA-GMPR-UR2 contiene la organización de los procedimientos a seguir paso a paso, con la generación de cada uno de los artefactos necesarios para lograr una documentación con el éxito y la eficiencia necesaria que requiere un proceso de software.

RECOMENDACIONES

Como resultado de la investigación y elementos a tener en cuenta, se hacen las siguientes recomendaciones, estas pueden ser desarrolladas en trabajos de diplomas en el curso 2008-2009:

- Poner en práctica la Metodología Ágil Gladys Marsi Peñalver Romero UNICORNIOS revisión 2 (MA-GMPR-UR1).
- Impartir un taller en el proyecto antes de comenzar la nueva iteración para Gerentes y Analistas con el objetivo de prepararlos para el proceso.
- Continuar el estudio del proceso de desarrollo XP y SCRUM para una mejor visión de la metodología.
- Realizar estudio de los artefactos que no han sido comprobados.
- Estudiar las técnicas de mitigación de riesgos para adaptarlas a esta forma de concebir el proceso y ajustarlas a los entornos UCI.
- Estudiar las técnicas de estimación, particularmente la técnica del póquer.
- Argumentar y probar un flujo de trabajo que contemple las actividades y artefactos de investigación, utilizan la generación mínima de documentos como principio propuesto para la metodología que se obtendrá al concluir este estudio.

REFERENCIAS BIBLIOGRÁFICAS

[1] RODRIGO, C., 10 de febrero 2008 Disponible en: http://geeks.ms/blogs/rcorral/archive/2008/02/10/documentos-o-ejecutables.aspx/

[2] Idem [1]

[3] ÁGIL, A., 2006 Disponible en: http://www.agilealliance.org

[4] MALAY, R. V., 2007 *Introducción de procedimientos ágiles en la producción de software en la Facultad 7 de la Universidad de las Ciencias Informáticas*. Disponible en: http://bibliodoc.uci.cu/TD/TD_0693_07.pdf

[5] HENRIK, K., 2007 *SCRUM y XP desde las trincheras.* Disponible en: http://infoq.com/minibooks/scrum-xp-fromthetrenches

[6] Idem [5]

[7] *Encuesta en Yahoo sobre Scrum.* 11.02.2008. Disponible en: http://www.navegapolis.net/content/view/739/62/

[8] GREGORIO, R. M., JORGE, F. Z., 2002 *Programación extrema y Software Libre.* Disponible en: http://10.33.20.195/svn/Documentacion%20General/IGSW_Metodologias/libros/robles-ferrer-todo%20sobre%20XP.pdf

[9] Idem [8]

[10] JOSE, A., 2006 Agile 2006 Survey. Disponible en: http://legnita.wordpress.com/2006/08/page/2/

[11] Agosto 2007 Survey Results, *2nd Annual "State of Agile Development"*. Disponible en: http://www.versionone.com/AgileSurvey/

[12] Idem [11]

[13] RODRIGO, C., Publicado el jueves, 17 de abril de 2008 21:17, Gestión de proyectos. Disponible en: http://geeks.ms/blogs/rcorral/archive/2008/04/17/uso-de-metolog-237-as-225-giles-en-microsoft.aspx

[14] SIGFRIDO G. F., *Metodologías de proyectos informáticos.* Citado el: 28 de marzo del 2008, Disponible en: http://bip.mideplan.cl/bip-consultas/SEBI/2006/metodologias/metodologiainformatica.pdf]

[15] ROBERT GUSTAVO F. D., *Métodos ágiles.* 2007, Disponible en: http://www.sg.com.mx/content/view/487

[16] Idem [15]

[17] *Scrum y Programación Extrema*. 2007/05/14, Disponible en: http://blog.chuidiang.com/2007/05/14/scrum-y-programacion-extrema/

[18] JOSÉ H. C., PATRICIO L., MARÍA CARMEN P., *Metodologías Ágiles en el Desarrollo de Software*. 2007, Disponible en: http://www.willydev.net/descargas/prev/TodoAgil.Pdf

[19] JOSE MANUEL N., *Metodologías ágiles para el desarrollo de software SCRUM + XP*. 2007, Disponible en: http://unkasoft.googlepages.com/desarrollo_agil_SCRUM_XP.pdf

[20] JOSE F., *Microsoft promueve internamente el uso de SCRUM.* Enviado el Domingo, 13. 11. 2005 - 18:13, Disponible en: http://www.agilespain.com/agilev2/microsoft_promueve_internamente_el_uso_de_scrum

[21] *Microsoft elogia a Scrum.* 13.11.2006, Disponible en: http://www.navegapolis.net/content/view/148/51/

[22] *La experiencia en Google*. 24.12.2006, Disponible en: http://www.navegapolis.net/content/view/522/62/

[23] Idem [22]

[24] Idem [22]

[25] *Scrum + XP como marco en el desarrollo de videojuegos.* 25.12.2005, Disponible en: http://www.navegapolis.net/content/view/227/62/

[26] EMILIO F., *Comenzando con SCRUM*. 01. 04. 2008, Disponible en: http://www.i2e.com.es/blog/?cat=19

[27] PRESSMAN, R. *Ingeniería de Software un enfoque práctico*. 5ta p.

[28] Idem [27]

[29] FOWLER, M. *Is Design Dead?*, 2001.Disponible en: www.martinfowler.com/articles/designDead.html

[30] FOWLER, M., BECK, K., BRANT, J. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley, 1999. p.

[31] MARTIN, R. *Continuo's Care vs. Initial Design*, 2002. [Disponible en: www.objectmentor.com/resources/articles/Continuous_Care.pdf.

[32] COCKBUN, A., WILLIAMS, L. *The Costs and Benefits of Pair Programming*. Humans and Technology Technical Report, 2000. p.

[33] Idem [29]

[34] JEFFRIES, R., ANDERSON, A., HENDRICKSON, C. *Extreme Programming Installed*. Addison-Wesley. , 2001. p.

[35] BECK, K. *Extreme Programming Explained. Embrace Change*. Pearson Education, 1999. p. Traducido al español como: *"Una explicación de la programación extrema. Aceptar el cambio",* Addison Wesley, 2000.

BIBLIOGRAFÍA

BALAKINAN, S. K., G. *Resolving Gathering of User Requirements – A Lightweight Methodology Approach in the Development of Medium Maturity Level Web Sites,* 2006. Disponible en: http://www.ucti.edu.my/wps/issuel/wp-06-04-paper.pdf

CARLOS R. *Métodos Heterodoxos en Desarrollo de Software,* Universidad de Buenos Aires, 2004. Disponible en: http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/heterodox.mspx

CONSUELO, D. C. C. *Metodologías ágiles, un nuevo enfoque metodológico.* Universidad Católica Santo Toribio de Mogrovejo, Escuela de Ingeniería de Sistemas y Computación.

HARRISON, N. B. *A Study of Extreme Programming in a Large Company,* 2006. Disponible en: http://www.agileshift.cl/Tutorial/DesarrolloAgilParte2.pdf#search=%22%20driven%20development%22%22

HENRYK, K. Prólogos de Jeff Sutherland y Mike Cohn *Scrum y XP desde las trincheras. Cómo hacemos Scrum,* 2007. Disponible en: http://infoq.com/minibooks/scrum-xp-from-the-trenches.pdf

JOSÉ, H. C., PATRICIO, L. Y. MARÍA C. P. *Metodologías Ágiles en el Desarrollo de Software.* DSIC – Universidad Politécnica de Valencia, 2006. Disponible en: http://www.willydev.net/descargas/prev/TodoAgil.Pdf

JORGE, F. Z. *Metodologías Ágiles,* 2007. Disponible en: http://libresoft.es/downloads/ferrer-20030312.pdf

Metodologías ágiles para el desarrollo de software SCRUM + XP Disponible en: http://unkasoft.googlepages.com/desarrollo_agil_UPSA.pdf

JUAN, P. *Flexibilidad con Scrum,* 2007. Disponible en: http://www.navegapolis.net/files/Flexibilidad_con_Scrum.pdf

MATÍAS, R., ESTEBAN, P*. Buenas prácticas de las metodologías ágiles*, 2007. Disponible en: http://www.efn.uncor.edu/escuelas/computacion//files/Buenas%20Pr%C3%A1cticas%20de%20Metodolog%C3%ADas%20%C3%81giles.pdf

MARÍA, C. P., PATRICIO L. *Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* Universidad Politécnica de Valencia, 2006. Disponible en: http://www.journaldatabase.org/articles/112000/Maetodologias_agiles_para.html

MARÍA, A. M. S. *Metodologías de Desarrollo de Software, 07.06.2004.* Disponible en: http://www.informatizate.cualmetodología.pdf

PANADÉS, P. L. Y. M. C. *Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP),* 2006. Disponible en: http://www.willydev.net/descargas/masyxp.pdf

PATRICIO, L. T., EMILIO, A. S. L. *Metodologías Ágiles en el Desarrollo de Software,* Alicante, España, 12.11.2003. Disponible en: http://issi.dsic.upv.es/tallerma/actas.pdf

RAFAEL, M. *Ingeniería del software. Metodologías de desarrollo,* 05.02.2008. Disponible en: http://www.um.es/docencia/barzana/IAGP/IAGP2-Metodologias-de-desarrollo.html

ANEXOS

Anexo 1: Plantilla Concepción del Sistema.

1. Polo.

[Software Libre, Teleformación, CENTERNET, Gestión de la información y el Conocimiento, etc.]

2. Clasificación.

[Desarrollo de aplicaciones, Desarrollo Web, Servicio, Personalización, Investigación]

3. Resumen.

[Pequeña descripción sobre lo que se tratará en el documento.]

Palabras claves:

4. Surgimiento

[Las razones por las cuales surge el producto.]

5. ¿Qué es?

[Descripción del sistema a desarrollar.]

6. Roles

[Organización por roles del equipo de trabajo. De no tener algunos de estos roles, lo dejan sin llenar, estatus es profesor o estudiante]

Rol	Nombre y Apellidos	Estatus
Gerente		
Cliente		
Miembros del Equipo		
Programadores		
Analista		
Diseñador		
Tester		
Arquitecto		

7. Misión

[Misión que comprende el sistema.]

8. Visión

[Aspiraciones futuras para el sistema a desarrollar.]

9. Herramientas utilizadas.

[Mencionar las herramientas que se utilizan para la realización del sistema.]

Anexo 2: Plantilla Lista de Reserva del Producto.

Prioridad	Ítem *	Descripción	Estimación	Estimado por
Muy Alta				
	[Números en secuencia según la cantidad de requerimiento]	*[Nombre de los requerimientos, ordenados según la prioridad de implementación, ubicados en Muy Alta, Alta, Media, Baja (aparecen los requerimientos de menor complejidad además de los requerimientos no funcionales del sistema a desarrollar.)]*	*[Estimación de cada uno de los requerimientos para su implementación por semanas.]*	*[Iniciales del rol que hizo la estimación del requerimiento.]*
Alta				
Media				
Baja				

Anexo 3: Plantilla Lista de Riesgos.

Riesgo	Tipo de Riesgo	Impacto	Descripción	Probabilidad	Efectos	Mitigación del riesgo
[Número del riesgo]	*[Los tipos de riesgos pueden ser: Tecnológico, Personal, Organización, Herramientas, Requerimientos, Estimación.]*	*[Lista de impactos en el proyecto o producto.]*	*[Breve descripción del tipo de riesgo]*	*[La probabilidad puede ser Muy alta, Alta, Media, Baja.]*	*[Los efectos pueden ser: Catastrófico, Serias, Tolerable, Insignificante.]*	*[Describir como se puede evitar el riesgo.]*

Anexo 4: Plantilla Tarea de Ingeniería.

Tarea de Ingeniería	
Número Tarea: *[Los números deben ser consecutivos]*	**Número Historia de Usuario:** *[Número de la historia de usuario a la que pertenece la tarea]*
Nombre Tarea: *[Nombre que identifica a la tarea.]*	
Tipo de Tarea : *[Las tareas pueden ser de: Desarrollo, Corrección, Mejora, Otra(Especificar)]*	**Puntos Estimados:** *[Tiempo en semanas que se le asignará. (Estimado)]*
Fecha Inicio:	**Fecha Fin:**
Programador Responsable:: *[Nombre y Apellidos del programador]*	
Descripción: *[Breve descripción de la tarea.]*	

Anexo 5: Plantilla Cronograma de Producción.

[Nota: En este documento se recoge las actividades planificadas por iteraciones por todo el equipo de desarrollad]

No	Hito	Descripción	Inicio	Fin	% ejec	Ejecutor
[número consecutivo]	*[Actividad planificada para esa iteración.]*	*[Breve descripción de la actividad a desarrollar]*	*[fecha de inicio estimada]*	*[fecha de fin estimada]*	*[% en que se encuentra realizada]*	*[Rol que la realiza]*

Observaciones: [Riesgos, Dependencias o Gestiones que queden fuera del alcance del grupo productivo que pueda atentar contra el cumplimiento en tiempo de la misma]

Nota: no menos de 2 hitos por mes por miembro del grupo. No más de 1 hito por semana por miembro del grupo.

Anexo 6: Plantilla Plan de Releases.

Release	Descripción de la iteración	Orden de la HU a implementar	Duración total
[Número de la iteración que se van a desarrollar para la realización del producto.]	*[Hacer una breve descripción del objetivo de la iteración]*	*[Número de las historia de usuario que se van a implementar en cada una de las iteraciones por orden de prioridad.]*	*[La duración total va a ser el tiempo estimado según las HU propuestas en que demorará su implementación]*

Anexo 7: Plantilla Historias de Usuario.

Historia de Usuario	
Número: *[Número de la historia]*	**Nombre Historia de Usuario:** *[Nombre que identifica la historia.]*
Modificación de Historia de Usuario Número: *[Cantidad de modificaciones que se le ha realizado a la historia de usuario (de no tener modificaciones se pone ninguna, sino la cantidad de veces que ha sido modificada).]*	
Usuario: *[Programador responsable de su implementación]*	**Iteración Asignada:** *[Que iteración se desarrollará. (Según su importancia)]*
Prioridad en Negocio: *[Prioridad puede ser Alta, Media o Baja(Según Cliente)]*	**Puntos Estimados:** *[Tiempo en semanas que se le asignará. (Estimado)]*
Riesgo en Desarrollo: *[Riesgo puede ser Alto, Medio o Bajo (Según Programadores)]*	**Puntos Reales:** *[Tiempo real dedicado a la realización de la HU en semanas.]*
Descripción: *[Breve descripción del proceso que define la historia.]*	
Observaciones: *[Alguna acotación importante de señalar acerca de la historia.]*	
Prototipo de interfase: *[Imagen de cada una de las interfaces relacionadas con la HU.]*	

Anexo 8: Plantilla Modelo de Historias de Usuario del Negocio.

1. Actores del negocio

[Se especifican todos los actores del negocio y se le asocia una descripción simple de cada uno de ellos]

Actor	Descripción

2. Trabajadores del negocio

[Se especifican todos los trabajadores del negocio y se le asocia una descripción simple de cada uno de ellos]

Trabajador	Descripción

3. Diagrama de Historias de usuario del Negocio

Anexo 9: Plantilla Modelo de diseño.

1. Diagrama de Paquetes

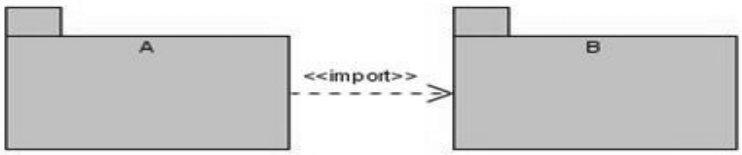

Fig. <No. del módulo>>.<<No. de la imagen>>
Diagrama de Paquetes.

Observaciones

- <<Observación 1>>.
- <<Observación 2>>.

2. Diagrama de Clases

[Realizar el diagrama de clases con sus características (atributos y métodos) y las relaciones de clase.]

Anexo 10: Plantilla Caso de Prueba de Aceptación.

Caso de Prueba de Aceptación	
Código Caso de Prueba: *[Inicial del proyecto-número de la HU a la que pertenece la prueba-número de la prueba.]*	**Nombre Historia de Usuario:** *[Nombre de la HU a realizar prueba.]*
Nombre de la persona que realiza la prueba: *[Nombre y apellidos.]*	
Descripción de la Prueba: *[Descripción de la prueba realizada.]*	
Condiciones de Ejecución: *[Condiciones necesarias para poder realizar la prueba.]*	
Entrada / Pasos de ejecución: *[Serie de pasos necesarios para lograr la realización de la HU, y así realizar la prueba.]*	
Resultado Esperado: *[Que cumpla con las restricciones del producto.]*	
Evaluación de la Prueba: *[Satisfactoria o no satisfactoria.]*	

Anexo 11: Plantilla Gestión de Cambios.

Gestión de Cambios		
Número: *[Número de la plantilla de la HU que le corresponde-número del*	**Producto:** *[Nombre del producto.]*	**Nombre del Proyecto:** *[Nombre del proyecto.]*
Fecha de la última Revisión:	**Fecha de la revisión actual:**	
Nombre del Modificador: *[Nombre y Apellido del modificador]*		
Nombre del Revisor: *[Nombre y Apellido del revisor]*		
Modificación del Servicio:	*[Nombre del cambio que se va a realizar.]*	
Descripción:	*[Breve descripción de la gestión de cambio.]*	
Localización del Cambio:	*[Poner de forma breve el lugar exacto donde se realizó algún cambio o URL de donde se encuentra el cambio.]*	

GLOSARIO DE TÉRMINOS

Agilidad: Nos dice el diccionario que es una calidad de un ágil o persona ágil. Definición de ágil: Ligero, pronto, expedito, dícese de la persona que mueve o utiliza sus miembros con soltura.
También podemos describir a la agilidad como una combinación de flexibilidad, velocidad y elasticidad. En el contexto de la investigación es la habilidad de responder de forma versátil al cambio para maximizar los beneficios.

Artefactos: En tecnología, es un dispositivo concebido y fabricado, sea de modo artesanal o industrial, por una o más personas.

Builds: Estructuras.

Calidad: La palabra calidad tiene múltiples significados. La calidad de un producto o servicio es la percepción que el cliente tiene del mismo. Es una fijación mental del consumidor que asume conformidad con un producto o servicio determinado, que solo permanece hasta el punto de necesitar nuevas especificaciones. La calidad es un conjunto de propiedades inherentes a un objeto que le confieren capacidad para satisfacer necesidades implícitas o explícitas.

Ciclo de vida: Es un proceso por el cual los analistas de sistemas, los ingenieros de software, los programadores y los usuarios finales elaboran sistemas de información y aplicaciones informáticas.

CMM: Por sus siglas en Ingles, en español, Modelo de Capacidad y madurez. Es un modelo de evaluación de los procesos de una organización.

Desarrollo evolutivo: Forma en la que la especificación y el desarrollo están intercalados. Cuenta con tres actividades concurrentes: Especificación, Desarrollo y Validación.

Desarrollo incremental: Forma de reducir la repetición del trabajo en el proceso de desarrollo y dar oportunidad de retrasar la toma de decisiones en los requisitos hasta adquirir experiencia con el sistema. Es una combinación del Modelo de Cascada y Modelo Evolutivo.

Estándares: Es una especificación que regula la realización de ciertos procesos o la fabricación de componentes para garantizar la interoperabilidad.

Gurús: Une la experiencia de una veintena de profesionales de la comunicación, tan conocedores como apasionados del mundo de las nuevas tecnologías de la información e Internet. Son un equipo especializado en generar contenidos multiplataforma que lleva desarrollando proyectos de este tipo desde hace más de siete años.

Gurú [Indio]

1. Maestro con capacidad para orientar en el camino de la liberación, que vive tan solo para transmitir sus conocimientos.
2. En Occidente, persona respetada por dominar una materia.

Herramientas: Son los ambientes de apoyo necesario para automatizar las prácticas de Ingeniería de Software.

Hitos: Aptitudes que pueden ser identificadas y que sirven de guía para el desarrollo normal.

Iteraciones: En el contexto de un proyecto se refieren a la técnica de desarrollar y entregar componentes incrementales de funcionalidades de un negocio. Una iteración resulta en uno o más paquetes atómicos y completos del trabajo del proyecto que pueda realizar alguna función tangible del negocio. Múltiples iteraciones contribuyen a crear un producto completamente integrado.

Métodos: Son las maneras que se efectúan las tareas de Ingeniería de Software o las actividades del ciclo de vida.

Metodología ágil: Nuevo enfoque metodológico orientado a la gente y los resultados

Metodología de desarrollo: Es una versión amplia y detallada de un ciclo de vida COMPLETO de desarrollo de sistemas que incluye: Reglas, procedimientos, métodos, herramientas, funciones individuales y en grupo por cada tarea, productos resultantes, normas de Calidad

Metodologías tradicionales: Metodologías basadas en procesos.

Procedimiento: Son los mecanismos de gestión que soportan a los métodos: El control de los proyectos, el control de la calidad.

Proceso de software: Es un proceso complejo que involucra diversas tareas de gestión y desarrollo. Como resumen de las etapas para la creación de un software, se pueden mencionar: Análisis, Desarrollo, Construcción, Pruebas (unitarias e integradas), Paso a Producción.

Prototipo: Un prototipo es una representación limitada del diseño de un producto que permite a las partes responsables de su creación experimentar su uso, probarlo en situaciones reales y explorar su uso. Un prototipo puede ser cualquier cosa, desde un trozo de papel con sencillos dibujos a un complejo software.

Proyecto de desarrollo: Elemento organizativo a través del cual se gestiona el desarrollo de software. El resultado de un proyecto es una versión de un producto.

Requisitos: Capacidades, condiciones o cualidades que el sistema debe cumplir y tener.

Sprint: Carrera breve a todo correr. Correr velozmente.