Universidad de las Ciencias Informáticas

FACULTAD 10

Propuesta de un expediente, para los proyectos productivos del Polo de Software Libre, de la Facultad 10.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autoras:

Raycel Fernández Céspedes

Susel Pino García

Tutores:

MSc. Graciela Gonzáles Pérez

Ing. Abel Meneses Abad

Ciudad de La Habana, Cuba, 2008

"Año 50 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los____ días del mes de_____ del año _____.

_____ _____

Raycel Fernández Céspedes Susel Pino García

_____ _____

Graciela González Pérez Abel Meneses Abad

"Cuando se es joven, se crea. Cuando se es inteligente, se produce. No se adapta, se innova: la medianía copia: la originalidad se atreve"

José Martí.

Agradecimientos.

Raycel Fernández Céspedes.

Agradezco en primer lugar a mi abuela Hilda, por ser tan especial en mi vida y darme los mejores consejos de este mundo. A mis padres, por estar siempre a mi lado, en los buenos y en los malos momentos, ustedes son mi inspiración para ser mejor cada día. A mí querido novio Aram, por brindarme siempre su apoyo incondicional, y ser mi ángel de la guarda en estos tres años. A mis tíos Yarina, Raúl y Sergio por estar siempre que los necesito y quererme como a su propia hija. A mi hermano, para el que trato de ser un ejemplo a seguir. A mi familia en general, que siempre han sido especiales conmigo. A mis amigas del alma, que han pasado estos 5 años a mi lado, Yanet, Dayana, Yurima, Lisandra, Daymara, y Yubi. A mi nueva familia que me acogió con mucho cariño, y me han hecho sentir como en casa. A mis tutores Graciela González Pérez y Abel Meneses Abad que han sido incondicionales conmigo. A mis compañeros de grupos y amistades de la universidad por formar parte de mi vida. A mi compañera de tesis Susel, por tener paciencia para soportarme.

A todos muchas gracias.

Susel Pino García.

Agradezco a mi papá por ser el mejor de todos, por mi educación, por estar siempre a mi lado, por todo el amor y el apoyo que me ha brindado para el cumplimiento de mi sueño. A mi hermana Susane por ser como una madre para mí, guiándome y aconsejándome en los buenos y malos momentos. A mí cuñado Samuel por su apoyo incondicional en todos estos años. A mi hermana Yaisel y familia por su preocupación. A Raycel por ser la mejor compañera de trabajo que he tenido y por todas las cosas que hemos pasado juntas. A Yenisel, Zenia y Gretter por ser como unas hermanas para mi en esta universidad y por todo el apoyo brindado. A Maria Elena, Mabel y Maydi por toda la confianza que depositaron en mi. A todos mis amigos, Rafael, Omel, David Ernesto, José Adrián y Elieser por preocuparse y estar ahí cuando los necesitaba. A mis compañeros de grupo y amistades de la universidad, por todo los momentos que pasamos juntos. A mis tutores Graciela González Pérez y Abel Meneses Abad por su entrega y dedicación, y haber puesto a mi disposición todos sus conocimientos. Al resto de mi familia y a todos aquellos que se acercaron a indagar por el estado de la tesis.

Muchas gracias.

Dedicatoria

En especial a mi padre, quien más ha influido en mi formación profesional y que siempre se ha enorgullecido de mí. A mi hermana Susane, que ha sido y será mi fiel amiga.

Susel Pino García.

A las personas que más amo en este mundo: mi familia. En especial a mi abuela Hilda y a mis padres, que han sido mi razón de ser. A mis tíos queridos, a mi hermanito, y al amor de mi vida. A ustedes va dedicada esta tesis.

Raycel Fernández Céspedes.

RESUMEN

El proceso de documentación de software ha alcanzado un auge considerable en los últimos tiempos, lo que garantiza un software con mayor calidad. La Universidad de las Ciencias Informáticas (UCI), no se ha mantenido al margen de este desarrollo, creando el expediente de proyecto. Pero debido a que muchos de los grupos de trabajo, fundamentalmente lo pertenecientes al Polo de Software Libre de la Facultad 10, tienen como política la adaptación de su documentación a las metodologías ágiles, la aplicación de este expediente no ha cubierto sus necesidades. Por lo que proponer un expediente de proyecto, que guíe el proceso de documentación, en estos proyectos, es el objetivo de la investigación.

En la investigación se sistematizan los elementos más significativos a tener en cuenta al documentar, centrando el estudio inicialmente en definir, basados en criterios de especialistas, qué es un expediente de proyecto. Además se analizan aspectos claves para la creación del mismo, tales como, el modelo de calidad que dirige el proceso de aseguramiento de la calidad, así como la metodología en la que está centrado el proceso de desarrollo de software en los proyectos. Por lo que surge la necesidad, de realizar un análisis de las principales características de los proyectos de la Facultad 10, del Polo de Software Libre, y posteriormente valorar las ventajas y desventajas del expediente que está aplicándose en ellos.

Esta tesis ofrece una solución a algunos de los problemas al documentar, existentes en los proyectos productivos, a partir de la adaptabilidad del expediente a las metodologías ágiles. Lo que garantiza un proceso de documentación, con mayor calidad y menor tiempo de realización.

Palabras claves: Metodologías ágiles, proceso de documentación, expediente de proyecto, proyectos productivos.

ÍNDICE

Índice de Figuras y Anexos

INTRODUCCIÓN

En la actualidad la industria del software se abre paso, cada vez con mayor fuerza en instituciones, empresas y administraciones públicas a nivel mundial, muchos son los países que se han trazado como propósito, el logro de un desarrollo exhaustivo de la producción del software y una respetada posición en el mercado.

Cuba, a pesar de ser un país bloqueado y del tercer mundo, no se ha mantenido al margen de este desarrollo, y se ha propuesto garantizar que el sustento de la economía esté basado en las producciones intelectuales, por lo que en estos momentos el campo de la informática, que está dando grandes pasos de avance, puede aportar mucho a los ingresos del país por conceptos de producción de software.

Por ello se han creado todos los mecanismos e instituciones que logren que la producción de software se eleve, una de estas instituciones es la UCI, surgida en el año 2002 en medio de la Batalla de Ideas, con el colosal propósito de convertirse en un pilar fundamental dentro del desarrollo de software en el país. *"Los objetivos de esta institución son altamente estratégicos. Al enemigo le va a preocupar diez veces más que cuando era un centro de exploración radioelectrónica, porque este es un centro estratégico del futuro y para el desarrollo del país"* (Ruz, 2002)

La UCI, para lograr un mejor desarrollo productivo, y cumplir con el objetivo fundamental por el que fue creada, se ha dividido en diez facultades, cada una de estas con un perfil de trabajo diferente, pero orientados todos al desarrollo informático.

Estas facultades han sido el espacio donde la producción de software ha alcanzado un auge considerable y se ha convertido en la actividad esencial, siendo una necesidad el perfeccionamiento de los proyectos productivos que en ella se desarrollan, así como de los procesos que se llevan a cabo para garantizar que el ciclo de vida del software se recoja de forma eficiente en los expedientes de proyecto.

Por el valor que poseen estos expedientes, a la hora de medir la calidad de los productos desarrollados por los proyectos productivos, la Dirección de Calidad de la UCI creó un expediente de proyecto, el cual se utiliza en cada uno de estos grupos de trabajo, pero debido a su estandarización y a la diversidad de perfiles que tiene la universidad, han surgido algunos problemas, tales como:

- El expediente posee plantillas que no son necesarias en algunos proyectos.
- Los desarrolladores emplean demasiado tiempo en la documentación.
- Muchas plantillas de las que conforman el expediente reiteran la misma información.

Todas estas dificultades conducen al planteamiento de que en los proyectos productivos, del Polo de Software Libre de la Facultad 10, debido a que tienen como política la adaptación de su documentación a la metodología utilizada, en este caso las metodologías ágiles, el expediente de proyecto confeccionado por la Dirección de Calidad de Software de la universidad, no satisface de manera global sus necesidades y especificidades.

Además, teniendo en cuenta la característica fundamental de las metodologías ágiles, de no necesitar documentaciones extensas, solo los documentos que solicite el cliente y los que sean imprescindibles para la reutilización del software, muchas de las plantillas que conforman a este expediente no son necesarias para proyectos del Polo de Software Libre, mientras que otros necesitan algunas plantillas que no están incluidas en el mismo. Esta estandarización afecta la calidad de la información recogida en los expedientes de proyecto y demoran el trabajo de los desarrolladores de software.

No cabe duda de que el tema es de imperante actualidad. La necesidad de desarrollar una nueva forma de recolectar información relativa a la producción de software se hace indispensable, pues sería una alternativa para aquellos proyectos del Polo de Software Libre, y una manera de reflejar sus particularidades en los expedientes de proyecto.

Para tratar de convertir esta necesidad en una realidad y perfeccionar el desarrollo de la producción, se ha planteado el siguiente **problema científico**, el expediente de proyecto que existe actualmente no puede aplicarse en su totalidad a los proyectos productivos del Polo de Software Libre, de la Facultad 10.

Para lograr obtener una solución fructífera, que erradique total o parcialmente el problema planteado, se necesita realizar un estudio exhaustivo del tema a tratar, por lo que el **objeto de estudio** de la investigación lo constituye el proceso de desarrollo de software. Pero para garantizar que el análisis sea más focalizado se trabaja en el siguiente **campo de acción**, el proceso de documentación en las metodologías ágiles XP y Scrum.

Una vez definido lo anterior, nos planteamos el siguiente **objetivo general,** elaborar una propuesta de expediente de proyecto que satisfaga las necesidades de los proyectos productivos del Polo de Software Libre, de la Facultad 10.

Para alcanzar el objetivo propuesto y teniendo como base el problema a resolver, se definen las siguientes **preguntas científicas**:

- ¿Cuáles son los antecedentes y estado actual del proceso de documentación del software en el mundo Contemporáneo?
- ¿Qué elementos deben tenerse en cuenta para la realización de un expediente de proyecto que satisfaga las necesidades de los proyectos productivos, del Polo de Software Libre?
- ¿Cuáles son los componentes y relaciones que se deben tener en cuenta para modelar un expediente de proyecto?

Para cumplimentar el objetivo trazado y ratificar las preguntas científicas establecidas, se proponen las siguientes **tareas de la investigación:**

- Sistematización de lo relativo al proceso de documentación de software en el mundo Contemporáneo.
  - Estudio de los elementos significativos, del proceso de documentación de software.
  - Análisis de las metodologías ágiles utilizadas en los proyectos productivos del Polo de Software Libre, de la Facultad 10.
  - Estudio de los modelos de calidad más utilizados para el trabajo con metodologías ágiles.
- Identificación de los elementos que deben tenerse en cuenta, para la realización de un expediente que responda a las necesidades de los proyectos productivos del Polo de Software Libre.
  - Estudio de los modelos de calidad más utilizados para el trabajo con metodologías ágiles.
  - Caracterización de los proyectos productivos del Polo de Software Libre, de la Facultad 10.
  - Valoración del expediente de proyecto vigente en la universidad.

- Elaboración de un expediente para los proyectos productivos del Polo de Software Libre, de la Facultad 10.

Para darle cumplimiento a las tareas propuestas con anterioridad, se utilizarán los métodos científicos siguientes:

Métodos teóricos:

Analítico – sintético: este método posibilitó distinguir los elementos relacionados con los expedientes de proyecto y proceder a realizar una revisión ordenada de cada uno de ellos, para posteriormente procesar toda la información y poder sintetizar, así como diferenciar cada una de las características de los expedientes de proyecto. Además permitió extraer los elementos más significativos teniendo en cuenta el objeto de estudio. Estas operaciones no son independientes, pues el análisis del objeto se realiza a partir de la relación que existe entre los elementos que lo conforman y a su vez, el proceso de síntesis se produce sobre la base de la integración de los resultados previos del análisis.

Histórico – lógico: este método permitió conocer los antecedentes y tendencias más actuales de los expedientes de proyecto en el mundo, en Cuba y más específicamente en la UCI, al estar vinculado al conocimiento de las distintas etapas de los objetos en su sucesión cronológica. Mediante este método, se analizó la trayectoria concreta de la teoría, su condicionamiento en los diferentes períodos de la historia y al basarse en el desarrollo histórico, pone de manifiesto la lógica interna de su desarrollo, su teoría y su esencia.

Modelación: este método brindó la posibilidad de modelar un expediente para los proyectos productivos, del Polo de Software Libre, de la Facultad 10, es decir, de modelar un procedimiento de trabajo.

Métodos empíricos:

Observación: este método permitió constatar los problemas que existen en el proceso de recopilación de información por parte de los expedientes de proyecto, por falta de una adaptación de los mismos a las características específicas de los proyectos productivos, precisamente tras estos problemas es que surge la idea de realizar la investigación.

Entrevista: esta técnica propició la obtención de información por parte de los líderes de proyecto, así como algunos de los roles más importantes dentro de los proyectos, tales como diseñadores, analistas, etc., con el fin de conocer en qué estado se encuentra el tema

de la recopilación de la información en los expedientes de proyecto, así como sus criterios sobre la implantación de un nuevo expediente, que satisfaga sus necesidades.

Esta investigación brinda como **aporte teórico,** una variante que constituye una alternativa para desarrollar el proceso de recopilación de información, que debe conformar los expedientes de proyecto, respondiendo a las necesidades específicas de los proyectos productivos del Polo de Software Libre, en la Facultad 10 de la UCI.

La **significación práctica** está dada por la disponibilidad de un expediente, para los proyectos productivos del Polo de Software Libre, lo que posibilita la adaptabilidad de la documentación de estos grupos de trabajo, a las metodologías ágiles, garantizando acortar los plazos de desarrollo de software, así como sus costos.

El presente trabajo está estructurado en una introducción, tres capítulos, las conclusiones y las recomendaciones.

En el **Capítulo 1**, se abordan los temas relacionados con el marco teórico de la investigación, es decir, los expedientes de proyecto, conceptos, antecedentes, elementos del proceso de documentación y reflexiones que permiten el esclarecer la importancia de su adecuada aplicación en los proyectos productivos, así como algunos aspectos relacionados con la calidad de los mismos.

En el **Capitulo 2** se realiza una caracterización de los proyectos productivos de la Facultad 10, del Polo de Software Libre, para conocer su estado actual, informando los resultados de la utilización del expediente de proyecto en cada uno de ellos y de esta forma analizar como marcha el proceso de recopilación de información. Además se realiza una valoración del expediente de proyecto que está vigente, y que fue creado por la Dirección de Calidad de la UCI, para de esta forma definir el expediente de proyecto propuesto, estableciendo una estrategia para su uso en la Facultad 10.

En el **Capitulo 3** se presentan el nuevo expediente, el cual será factible para los proyectos productivos de esta facultad.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA.

Introducción

El software en la actualidad es una parte crítica de todo negocio, debido a la calidad insuficiente o variable que se percibe en su desarrollo, específicamente en el proceso de recopilación de información, desde que se inicia su creación. Para resolver este problema se ha optado por mejorar el proceso de gestión de la documentación, o sea la forma en que deben recogerse las características del producto software desde su fase inicial de desarrollo, hasta el momento que está en manos de los usuarios finales.

A lo largo de los años han existido diversos criterios para llevar a cabo este proceso de documentación, sin embargo no se ha podido alcanzar aún un equilibrio, pues no existe un patrón que proporcione igualdad en este aspecto, es decir, cada empresa o institución desarrolladora de software garantiza que su producto este bien documentado pero sin seguir ningún estándar, incluso aparece muy poca información de este tema, sólo tratan de llevar a la par del desarrollo del software el llenado de aquellas plantillas que respondan al modelo de calidad y a la metodología escogida para regir su trabajo. El documento que recoge todas estas plantillas es conocido hoy en día en nuestro país, como “expediente de proyecto.”

1.1 Definición de un expediente de proyecto.

Luego de una sistematización sobre el tema de los expedientes de proyecto, se obtuvo como resultado, que no existe una fecha exacta en la cual enmarcar el surgimiento de la documentación de software. Pero es con el nacimiento de la primera computadora comercial en 1951, que surgió la necesidad de desarrollar una industria de software, que dirigiera la producción del mismo, aunque su origen se situó en 1955, año en el que nació la primera empresa productora de software independiente, conocida como la Computer Usage Company (CUC). Por lo tanto, se toma la década de los cincuenta, como el período de nacimiento de la documentación de software, teniendo como base que un software, siempre está acompañado de algún documento que facilite su uso o comprensión.

Con el transcurso de los años, se ha ido perfeccionando el uso de esta documentación, y es ya en la actualidad una realidad, que todas las empresas productoras de software, mantienen un registro de la documentación de sus productos de una forma organizada y clara, lo que si no aparece en ninguna de las bibliografías que tratan el tema de la

documentación de proyectos es la manera en que lo hacen, y mucho menos el nombre que lleva este registro. En la UCI por acuerdo de su Departamento de Calidad, al documento que recoge detalladamente la documentación de cada uno de los productos software que se desarrollan en los proyectos productivos, se les llamó expediente de proyecto.

El ingeniero *Yanko Hernández Valdés*, asesor de la Dirección de Calidad, en la Facultad 10, define a un expediente de proyecto como la forma de registro de todos los sucesos, elementos usados y cualquier otro tipo de acontecimiento que sea necesario documentar a lo largo del ciclo de vida del software. Su objetivo es registrar todo el ciclo de forma tal que no se pierdan elementos generados para garantizar la no duplicación de esfuerzos y acceder fácilmente a cualquier necesidad de información.

El ingeniero *Ramsés Delgado*, especialista de la Dirección de Calidad, define al expediente de proyecto como la herramienta que agrupa y organiza todos los artefactos que se generan durante el desarrollo de software.

Las definiciones anteriores reflejan de manera general el objetivo fundamental de un expediente de proyecto, que no es más que organizar y registrar el ciclo de vida del software, pero las autoras de esta investigación teniendo en cuenta lo antes planteado, así como las experiencias personales adquiridas en entrevistas realizadas a líderes de proyecto, definen al expediente de proyecto, como la forma más inmediata de registrar el ciclo de vida de un software, garantizando el almacenamiento de la información más relevante y necesaria del producto, para una posible reutilización de la misma. El expediente debe poseer como característica fundamental, una estructura que proporcione organización y que permita el control de cada uno de los artefactos generados, sin permitir la repetición innecesaria de información, así como el exceso de dedicación y esfuerzo de los desarrolladores en el proceso de documentación.

1.1.1 Aspectos para documentar correctamente un sistema de software.

Un software pobremente documentado carece de valor aunque haya funcionado bien en alguna ocasión. En el caso de programas pequeños y poco importantes que sólo se utilizan durante un corto período de tiempo, algunos comentarios en el código podrían ser suficientes. No obstante, la mayoría de los programas cuya única documentación es el código, quedan obsoletos rápidamente y es imposible mantenerlos, por lo que se hace necesario desde un inicio, recopilar toda la información relevante sobre el ciclo de vida del software.

A menos que se viva en un mundo en el que nada cambia, siempre se tendrá que volver a consultar el código que ya está escrito. Poniéndose en duda decisiones que se tomaron durante el desarrollo del mismo. Por lo que, si no se documentan cada una de estas decisiones, se cometerán siempre los mismos errores, y se perderá tiempo tratando de comprender lo que se pudo haber descrito fácilmente en una ocasión. La falta de documentación no sólo genera trabajo adicional, sino que también tiende a dañar la calidad del código. Si no se posee una nítida caracterización del problema, es imposible que se desarrolle una solución clara.

Aprender a documentar software es una tarea complicada y exige un criterio de ingeniería maduro. Además hacerlo de manera demasiado concisa es un error habitual, pero el otro extremo puede resultar igual de perjudicial: si se escriben documentaciones muy extensas, éstas desconcentrarán la atención del lector y constituirán una carga a la hora de conservarlas. Es esencial documentar sólo los asuntos correctos y necesarios. La documentación no sirve de ayuda para nadie si su extensión desanima a las personas a la hora de leerla y hacerla.

Generalmente los desarrolladores de software tienden a mostrarse resistentes ante los problemas de documentación. Demuestra poca visión del futuro, si se conoce que existen problemas en el software tanto en el diseño, como en el código fuente y no se plasma en la documentación hasta que se solucione, esto hará que el lector ahorre tiempo dándole vueltas a algo que es erróneo, y sabrá a dónde tiene que remitirse si encuentra problemas, además tendrá una documentación más útil y honesta, de lo contrario no se desarrolla el proceso de documentación hasta que no se rectifiquen los errores.

Otro asunto es en qué momento documentar. Aunque algunas veces es conveniente posponer la tarea de la documentación mientras se realizan experimentos, los programadores con experiencia y que disponen de tiempo suelen documentar de forma metódica incluso el código provisional, los análisis de un problema inicial y los borradores de un diseño. Se cree que esto hace que la experimentación sea más productiva. Además, debido al hábito de documentar, les resulta normal escribir a medida que van avanzando.

En la actualidad el proceso de recopilación de información se vuelve bastante tedioso para los desarrolladores de software, pues debido a la dinámica de la producción, en muchas ocasiones se dedica más tiempo al proceso de documentar, que al desarrollo del producto. Además existen otras razones que provocan el rechazo a documentar el software, según lo planteado por Nuñez Zuleta en el 2008, tales como:

- **La documentación se vuelve obsoleta desde el momento que la escribes:** es decir lo que se escribe hoy, quizás se tenga que cambiar mañana, pues el software cambia, salen errores, hay más o menos funcionalidades, y todo esto incluye transformaciones en la documentación.
- **No hay tiempo para la documentación:** al menos que tengas un escritor técnico (tecnical writer) seguramente un desarrollador es quien se encarga de esta tarea. Por supuesto no es la única actividad que tendrá que hacer y al menos que sea la documentación del usuario esta fase tendrá una baja prioridad.
- **No es bien vista:** en la mayoría de los casos, para los desarrolladores es un estorbo la documentación, por muchas razones: hay actividades más importantes que realizar, como por ejemplo: código, pruebas, soporte. Además en varias ocasiones la documentación se lleva más tiempo que el desarrollo del software. (Nuñez, 2006)

Debido a los aspectos antes mencionados, es que se necesita que en la actualidad se utilicen metodologías de desarrollo de software que aporten nuevas formas para documentar, teniendo en cuenta que la mayoría de los proyectos son de pequeño formato, fundamentalmente los que trabajan con software libre y aplican metodologías ágiles, para de esta forma lograr productos que posean una documentación útil, pero a la vez no haya sido un obstáculo para los desarrolladores.

1.2 Metodologías de desarrollo de software.

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos software. Es como un libro de recetas de cocina, en el que se van indicando paso a paso todas las actividades a realizar para lograr el producto informático deseado, indicando además qué personas deben participar en el desarrollo de las actividades y qué papel deben de tener. Además detallan la información que se debe producir como resultado de una actividad y la información necesaria para comenzarla (Menéndez, 2005).

Las técnicas indican cómo debe ser realizada una actividad determinada e identificada en la metodología. Combinan el empleo de modelos o representaciones gráficas junto con el empleo de procedimientos detallados. Se debe tener en consideración que una técnica determinada puede ser utilizada en una o más actividades de la metodología de desarrollo de software. Además se debe tener mucho cuidado cuando se quiere cambiar una técnica por otra.

1.2.1 Metodologías ágiles.

Las Metodologías Ágiles o "Ligeras" constituyen un nuevo enfoque en el desarrollo de software, mejor aceptado por los desarrolladores de proyectos que las metodologías convencionales, debido a la simplicidad de sus reglas y prácticas, su orientación a equipos de desarrollo de pequeño tamaño, su flexibilidad ante los cambios y su ideología de colaboración (Gallo, 2006).

De manera general, las metodologías ágiles pueden explicarse a través de los siguientes 4 principios fundamentales:

1. Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas. Dado que el proceso de desarrollo es creativo, no es posible pensar que las personas trabajen respondiendo a órdenes o procesos rígidos.
2. Desarrollar software que funcione es más importante que conseguir una buena documentación. Puesto que si el software no funciona la documentación no vale de nada. A nivel interno puede haber documentación, pero solo la necesaria y a nivel externo lo que el cliente requiera.
3. La colaboración con el cliente más que la negociación de un contrato. Supone que la satisfacción del cliente con el producto será mayor, mientras exista una conversación y realimentación continua entre este y la empresa.
4. Responder a los cambios más que seguir estrictamente un plan. Puesto que si un proyecto de software no es capaz de adaptarse a los cambios fracasará, especialmente en productos de gran envergadura. La estrategia de planificación se basa en: planes detallados para las próximas semanas, planes aproximados para los próximos meses y muy generales para plazos mayores.

A nivel mundial existen varias metodologías para el desarrollo de software que pueden ser catalogadas de ágiles. Cada una con características específicas, por lo que es importante realizar un análisis, antes de aplicar cualquiera de ellas un proyecto. Dentro de las más conocidas podemos citar:

- XP (Extreme Programming – Kent Beck).
- SCRUM (Jeff Sutherland, Ken Schwaber).
- Crystal (Alistair Cockburn).
- DSDM (DSDM Consortium).

- Feature Driven Development (Jeff deLuca, Peter Coad, Together Soft).
- Adaptive Software Development (Jim Highsmith).
- Lean Development (Mary y Tom Poppendieck, Robert Charette).

De todas las metodologías ágiles citadas, se analizarán sólo dos de ellas, XP y Scrum, las cuales tienen un gran peso en el mundo digital. Este binomio fue escogido por la ingeniera Malay Rodríguez Villar en el año 2007, llamándolo MA-MRV-UR1 (Metodologías Ágiles - Malay Rodríguez Villar- Unicornios Revisión 1), para ser utilizado en proyectos productivos de la universidad, siendo aplicado este año en los pertenecientes al Polo de Software Libre, de la Facultad 10, y tomado como referencia para el desarrollo de esta investigación. A continuación se caracterizarán cada una de estas dos metodologías, teniendo en cuenta los siguientes parámetros:

- Características principales.
- Roles.
- Ciclo de desarrollo o proceso.
- Prácticas.

1.2.1.1 Extreme Programming / Programación Extrema (XP).

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 propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios (Beck, 2000)

La definición planteada anteriormente por Kent Beck, recoge varios aspectos significativos de la metodología XP, tales como la proliferación del trabajo en equipo, así como la relación de retroalimentación entre los clientes y los desarrolladores, pero las autoras de esta investigación consideran que la definición de esta metodología puede ser mucho más amplia, y la definen de la siguiente forma: XP es una metodología de desarrollo ágil, basada en una serie de valores y de prácticas que persiguen el objetivo de aumentar la productividad a la hora de desarrollar programas. Tiene como filosofía fundamental la satisfacción de las necesidades del cliente, por lo que se integra como una parte más del equipo de desarrollo. Además permite la simplicidad en el trabajo del equipo, la

comunicación entre clientes y desarrolladores, así como la retroalimentación o reutilización del código implementado. Esta metodología está diseñada para aplicaciones que requieren un grupo de programadores pequeños y que poseen requisitos cambiantes, debido a su rápida adaptabilidad a los cambios.

Características. (Gutiérrez, 2007)

- *Desarrollo iterativo e incremental:* pequeñas mejoras, unas tras otras.
- *Pruebas unitarias continuas*, frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión. Se aconseja escribir el código de la prueba antes de la codificación.
- *Programación en parejas:* se recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto. Se supone que la mayor calidad del código escrito de esta manera- el código es revisado y discutido mientras se escribe- es más importante que la posible pérdida de productividad inmediata.
- Frecuente *interacción del equipo de programación con el cliente o usuario*. Se recomienda que un representante del cliente trabaje junto al equipo de desarrollo.
- *Corrección de todos los errores* antes de añadir nueva funcionalidad.
- *Refactorización del código,* es decir, reescribir ciertas partes del código para aumentar su legibilidad y mantenibilidad pero sin modificar su comportamiento. Las pruebas han de garantizar que en la refactorización no se ha introducido ningún fallo.
- *Propiedad del código compartida:* en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve el que todo el personal pueda corregir y extender cualquier parte del proyecto. Las frecuentes pruebas de regresión garantizan que los posibles errores sean detectados.
- *Simplicidad en el código*: es la mejor manera de que las cosas funcionen. Cuando todo funcione se podrá añadir otra funcionalidad si es necesario. La programación extrema apuesta que es más sencillo hacer algo simple y tener un poco de trabajo extra para cambiarlo si se requiere, que realizar algo complicado y quizás nunca utilizarlo.

Roles (Gutiérrez, 2007)

- *Programador*. El programador escribe las pruebas unitarias y produce el código del sistema.
- *Cliente*. Escribe las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio.
- *Encargado de pruebas (Tester).* Ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.
- *Encargado de seguimiento (Tracker*). Proporciona retroalimentación al equipo. Verifica el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, para mejorar futuras estimaciones. Realiza el seguimiento del progreso de cada iteración.
- *Entrenador (Coach).* Es responsable del proceso global. Debe proveer guías al equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente.
- *Consultor*. Es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto, en el que puedan surgir problemas.
- *Gestor (Big boss)*. Es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje.

Ciclo de vida (Letelier, 2004)

El ciclo de vida ideal de XP incluye seis fases:

1. *Exploración.*

   Los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La exploración toma de pocas semanas a

pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.

2. *Planeamiento de la Entrega (Release)*.

   El cliente establece la prioridad de cada historia de usuarios, y luego los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses. Esta fase dura unos pocos días, y en ella se deben incluir varias iteraciones para lograr una entrega. El cronograma fijado en la etapa de planeamiento se realiza a un número de iteraciones, cada una toma de una a cuatro semanas en ejecución. La primera iteración crea un sistema con la arquitectura del sistema completo. Esto es alcanzado seleccionando las historias que harán cumplir la construcción de la estructura para el sistema completo. El cliente decide las historias que se seleccionarán para cada iteración. Las pruebas funcionales creadas por el cliente se ejecutan al final de cada iteración. Al final de la última iteración el sistema esta listo para la producción.

3. *Iteraciones.*

   Incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fuercen la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema estará listo para entrar en producción. Los elementos que deben tomarse en cuenta durante la elaboración del Plan de la Iteración son: historias de usuario no abordadas, velocidad del proyecto, pruebas de aceptación no superadas en la iteración anterior y tareas no terminadas en la iteración anterior. Todo el trabajo de la iteración es expresado en tareas de programación, cada una de ellas es asignada a un programador como responsable, pero llevadas a cabo por parejas de programadores.

4. *Producción.*

   Requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase. Es posible que se rebaje el tiempo que toma cada iteración, de tres a una semana. Las ideas que han sido propuestas y las sugerencias son documentadas para su posterior implementación (por ejemplo, durante la fase de mantenimiento).

5. *Mantenimiento.*

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

6. *Muerte del Proyecto.*

   Es cuando el cliente no tiene más historias para ser incluidas en el sistema. Esto requiere que se satisfagan las necesidades del cliente en otros aspectos como rendimiento y confiabilidad del sistema. Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura. La muerte del proyecto también ocurre cuando el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo.

En la figura tomada de (Villegas, 1997) se pueden observar de forma gráfica las fases en las que se subdivide el ciclo de vida de XP.

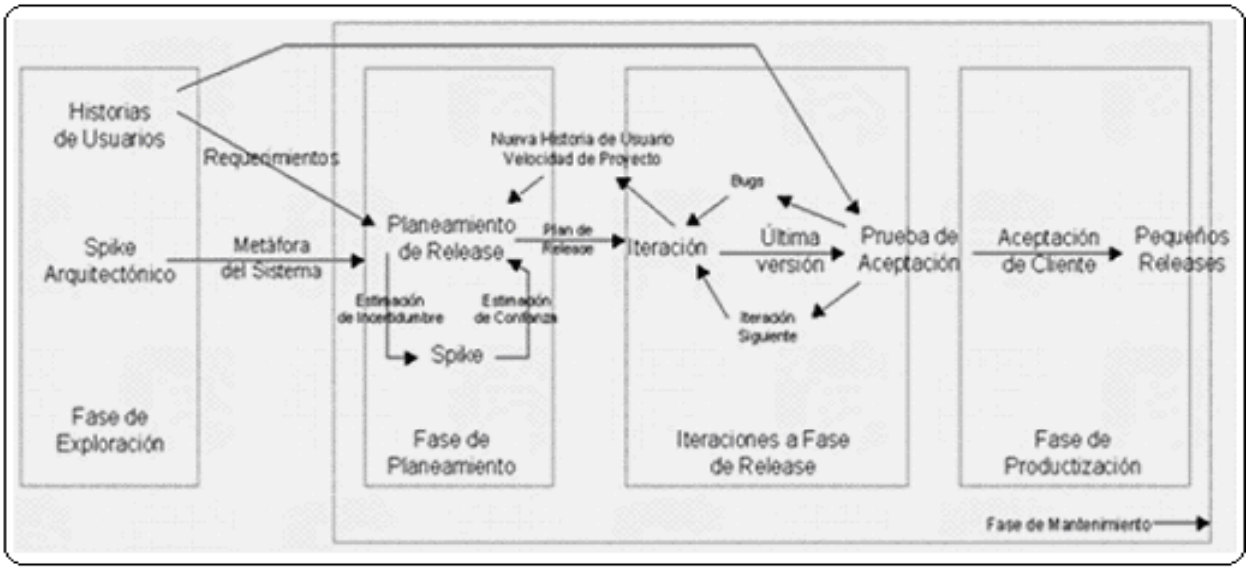


Figura 1. Ciclo de vida de eXtreme Programming.

Proceso XP (Letelier, 2004)

El ciclo de desarrollo consiste a grandes rasgos, en los siguientes pasos:

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

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

Prácticas XP (Letelier, 2004)

La principal suposición que se realiza en XP es la posibilidad de disminuir la mítica curva exponencial del costo del cambio a lo largo del proyecto, lo suficiente para que el diseño evolutivo funcione. Esto se consigue gracias a las tecnologías disponibles para ayudar en el desarrollo de software y a la aplicación disciplinada de las siguientes prácticas.

- *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. Este juego se realiza durante la planificación de la entrega, en la planificación de cada iteración y cuando sea necesario reconducir el proyecto.
- *Entregas pequeñas:* Consiste en producir rápidamente versiones del sistema que sean operativas, aunque obviamente no cuenten con toda la funcionalidad pretendida para el sistema pero si que constituyan un resultado de valor para el negocio. Una entrega no debería tardar más 3 meses.
- *Metáfora:* En XP no se enfatiza la definición temprana de una arquitectura estable para el sistema. Dicha arquitectura se asume evolutivamente y los posibles inconvenientes que se generarían por no contar con ella explícitamente en el comienzo del proyecto se solucionan 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.
- *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. Según Kent Beck en cualquier momento el diseño adecuado para el software es aquel que: supera con éxito todas las pruebas, no tiene lógica duplicada, refleja claramente la intención de implementación de los programadores y tiene el menor número posible de clases y métodos.
- *Pruebas:* 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.
- *Refactorización (Refactoring):* 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.

- *Programación en parejas*: Toda la producción de código debe realizarse con trabajo en parejas de programadores. Las principales ventajas de introducir este estilo de programación son: muchos errores son detectados conforme son introducidos en el código, por consiguiente la tasa de errores del producto final es más baja, los diseños son mejores y el tamaño del código es menor, 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 3 o 4 meses de practicar la programación en parejas. (Cockbun, 2000)
- *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.
- *Cliente in-situ:* El cliente tiene que estar presente y disponible todo el tiempo para el equipo. Gran parte del éxito del proyecto XP se debe a que es el cliente quien conduce constantemente el trabajo hacia lo que aportará mayor valor de negocio y los programadores pueden resolver de manera inmediata cualquier duda asociada. La comunicación oral es más efectiva que la escrita, ya que esta última toma mucho tiempo en generarse y puede tener más riesgo de ser mal interpretada.
- *Estándares de programación:* XP enfatiza la comunicación de los programadores a través del código, con lo cual es indispensable que se sigan ciertos estándares de programación (del equipo, de la organización u otros estándares reconocidos para los lenguajes de programación utilizados). Los estándares de programación mantienen el código legible para los miembros del equipo, facilitando los cambios.

Características de los proyectos (Rodríguez, 2007)

Los proyectos que utilizan XP, generalmente poseen las siguientes características:

- Requisitos imprecisos y muy cambiantes.
- Existencia de un alto riesgo técnico.

La metodología XP, debido a las buenas prácticas que brinda a los desarrolladores, es utilizada en los proyectos, para el desarrollo de software, garantizando un ciclo de vida estable, y un producto final con una excelente calidad. Además disminuye el tiempo de desarrollo del software, y minimiza los costos de producción.

1.2.1.2 SCRUM.

El término “Scrum” viene de un estudio realizado en 1986, por los japoneses Takeuchi y Nonaka. En dicho estudio se documentaban una serie de proyectos muy exitosos los cuales tenían en común el uso de equipos chicos y multidisciplinarios. El estudio comparaba a esos equipos hiper-productivos con la formación Scrum de Rugby. Más adelante, en 1993, Jeff Sutherland creó el proceso Scrum para desarrollo de software, usando este estudio como base y adoptó la analogía con los equipos de Rugby. Posteriormente, en 1995, Ken Schwaber formalizó el proceso y lo abrió a toda la industria del software.

Según Joaquín Gutiérrez Gil, Scrum define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos 10 años. Sus principales características se pueden resumir en dos. El desarrollo de software se realiza mediante iteraciones, denominadas sprints, con una duración de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo proyecto, entre ellas destaca la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración. (Gutiérrez, 2007)

Teniendo en cuenta las características mencionadas anteriormente de Scrum, y la sistematización realizada sobre esta metodología, las autoras de esta investigación definen a Scrum como una metodología que enfatiza valores y prácticas de gestión sin pronunciarse sobre requerimientos, implementación y demás cuestiones técnicas. Es totalmente un proceso de administración y control que implementa técnicas de control de procesos. En fin esta metodología define todo un marco para la gestión de proyectos, con un rápido cambio de requisitos. Tiene como características fundamentales, el desarrollo de software mediante iteraciones, denominadas sprints, con una duración de 30 días y las reuniones a lo largo proyecto, entre ellas se destaca la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración.

Características (Rodríguez, 2007)

- Equipos auto dirigidos.
- Utiliza reglas para crear un entorno ágil de administración de proyectos.

- No prescribe prácticas específicas de ingeniería.
- Los requerimientos se capturan como ítems de la lista reserva del producto.
- El producto se construye en una serie de sprints de un mes de duración.
- Usado para proyectos complejos con requerimientos cambiantes.
- Basado en un control de proceso empírico.

Roles (Gutiérrez, 2007)

- *Jefe del Equipo (Scrum Máster):* Es un rol de administración que debe asegurar que el proyecto se está llevando a cabo de acuerdo con las prácticas, valores y reglas de Scrum y que todo funciona según lo planeado. Su principal trabajo es remover impedimentos y reducir riesgos del producto. Este rol suele ser desempeñado por un Gerente de Proyecto o Líder de equipo.
- *Equipo de Scrum (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 de la carrera, revisar la lista de reserva del producto y sugerir obstáculos que deban ser removidos para cumplir con los artículos 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 a tiempo completo, con algunas excepciones. La membresía solo puede cambiar entre sprints, no durante.
- *Dueño del producto (Product Owner)*: Es el responsable del proyecto, administra, controla y comunica la lista de reserva. Es el responsable de encontrar la visión del producto y reflejarla en la lista de reserva.

Además los roles se dividen en dos categorías:

Implicados	Comprometidos
- Usuarios finales - Marketing - Áreas comerciales - Áreas contables - Etc.	- Jefe del equipo - Dueño del producto - Equipo

Scrum refleja esta diferencia entre estos dos grupos para garantizar que las personas que tienen la responsabilidad poseen además la autoridad necesaria para poder lograr el éxito del proceso, y que quienes no la posean no produzcan interferencias innecesarias.

Ciclo de vida (Gutiérrez, 2007)

El ciclo de vida de Scrum cuenta con tres partes, las que se exponen a continuación:

1. *Pre-Juego:*
    - *Planeamiento (Planning):* El propósito es establecer la visión, definir expectativas y asegurar la financiación. Las actividades son la escritura de la visión, el presupuesto, el registro de acumulación o retraso del producto inicial y los ítems estimados, así como la arquitectura de alto nivel, el diseño exploratorio y los prototipos. El registro de acumulación es de alto nivel de abstracción.
    - *Montaje (Staging): El propósito es identificar más requerimientos y priorizar las tareas para la primera iteración. Las actividades son planificación, diseño exploratorio y prototipos.*

2. J*uego o Desarrollo.*
    - El propósito es implementar un sistema listo para entrega en una serie de iteraciones de treinta días llamadas “sprints”.

3. *Pos-Juego*
    - *Liberación:* El propósito es el despliegue operacional. Las actividades, documentación, entrenamiento, mercadeo y venta.

Algunos textos sobre Scrum establecen una arquitectura global en la fase de pre-juego; otros dicen que no hay una arquitectura global en Scrum, sino que la arquitectura y el diseño emanan de múltiples sprints. No hay una ingeniería del software prescripta para Scrum; cada quien puede escoger entonces las prácticas de automatización, inspección de código, prueba unitaria, análisis o programación en pares que le resulten adecuadas. (Gutiérrez, 2007)

En la siguiente figurase muestra una breve descripción del funcionamiento del ciclo de vida de Scrum.

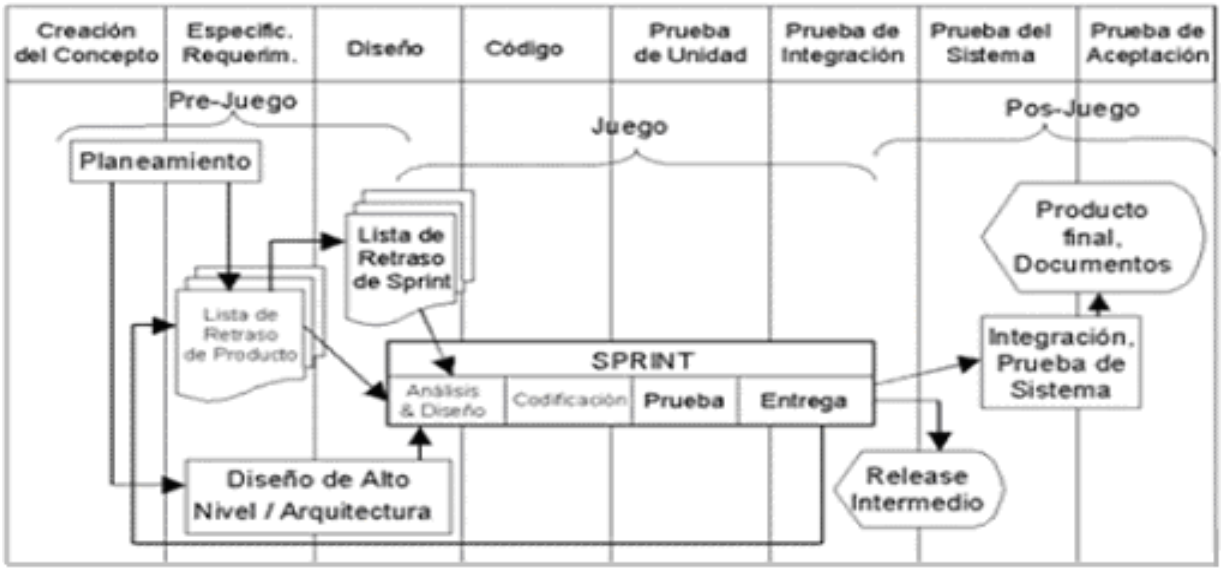


Figura 2. Ciclo de vida de Scrum.

Prácticas (Rodríguez, 2007)

- Iteración (Sprint).
- Reunión de planificación de las iteraciones.
- Reuniones diarias.
- Reuniones de revisión de las iteraciones.
- Reuniones de revisión del diseño.
- Estabilización de las iteraciones.
- Meta Scrum.

Características de los proyectos (Rodríguez, 2007)

Los proyectos que utilizan Scrum, generalmente poseen las siguientes características:

- Rápido cambio de requisitos.
- No mayor de 10 ingenieros (ideal, con más se forman equipos).

Esta metodología es utilizada fundamentalmente para la gestión de proyectos, pues brinda un grupo de funcionalidades y prácticas, que permiten realizar este proceso de manera cómoda, y garantizan una planificación y organización de excelencia.

Luego de analizar cada una de las características antes expuestas, relativas a las metodologías utilizadas en el desarrollo y la gestión de un software, es indudable que son las metodologías ágiles mejores que las pesadas y que la elección realizada por los proyectos productivos del Polo de Software Libre, no fue un error. Lo que se demuestra al identificar las diferencias más significativas que existen entre las metodologías ágiles y las ligeras, así como los beneficios que aportan.

La diferencia fundamental entre las metodologías clásicas o ligeras y las metodologías ágiles es que, para las clásicas, el paradigma se basa en la “predecibilidad”, es decir se predice todo lo que se realizará en cada una de las fases de desarrollo del software, mientras que en las ágiles, se basa en la “adaptabilidad”, es decir se adapta a cada uno de los cambios que puedan ir surgiendo en los requisitos del producto. El mejor método a elegir, para un desarrollo dado, depende del tipo de proyecto, de las características y cultura de la organización, del tamaño del proyecto y la relación con el cliente.

Una cosa si es segura: el cambio que introducen los métodos ágiles de desarrollo, y más recientemente, de análisis y diseño, significan un considerable reto para las organizaciones dedicadas a la construcción de software eficaz, útil y exitoso. Dado que generalmente, implican una carga de gestión más distribuida entre los equipos de trabajo, también constituye una oportunidad para empresas que no opten por una administración pesada y demasiado compleja.

Finalmente, los métodos ágiles reconocen un factor evidente, pero frecuentemente ignorado: el software es concebido, diseñado y desarrollado por personas y no por máquinas. Ello tiene implicaciones sobre aspectos de comunicación, motivación, aprendizaje, comportamiento social etc., que las metodologías clásicas tienden a minimizar o ignorar. Todas las metodologías ágiles dan una primordial importancia a la interacción de los equipos de desarrollo con sus clientes (personas que interactúan con el sistema), los usuarios finales (personas que reciben el producto final), la administración de su organización y otros factores vitales.

Los métodos ágiles de desarrollo de software no son, ni remotamente, el remedio para la solución de los problemas del desarrollo de programas y sistemas para computadoras, pero sí son una importante contribución hacia la obtención de un software útil, eficaz y oportuno, y producido en ambientes entusiastas. Para empresas comprometidas en el desarrollo de software, los métodos ágiles pueden traducirse en un replanteamiento de las técnicas de desarrollo, gestión de proyectos, aseguramiento de la calidad, arquitectura de software,

diseño, métodos de pruebas, administración de configuración y aprovechamiento del personal involucrado en cada proyecto. Lo que tributa a resultados satisfactorios en el mercado.

1.3 Estándares y modelos de calidad.

El software juega un papel muy importante para el desarrollo de las organizaciones. Día tras día son liberados para su uso distintos tipos de programas para diferentes clases de clientes, los hay para cada necesidad, de tal manera que resulta difícil imaginar alguna situación en la que el software no estuviera presente, dado que es uno de los componentes básicos de la tecnología que se involucra en las empresas, no sólo como soporte a los procesos de negocio, productivos y administrativos, sino como parte integral de las estrategias corporativas para la generación de ventajas competitivas.

Es una gran oportunidad y un reto para la industria del software desarrollar las estrategias que le permitan un posicionamiento y un reconocimiento internacional con productos competitivos de exportación, lo que requerirá entre otras cosas, de la elección e implantación del Modelo o Estándar de Calidad indicado, que permitirá organizar y seguir una estrategia de trabajo.

Los Modelos de Calidad no son más que un conjunto de buenas prácticas para el ciclo de vida del software, enfocado en los procesos de gestión y el desarrollo de proyectos (Ronquillo, 2008). La aplicación de estos modelos en empresas o instituciones es muy importante, pues proponen temas de administración en los que cada organización debe hacer énfasis, además integran diferentes prácticas dirigidas a los procesos clave lo que posibilita medir los avances en calidad.

Los Estándares de Calidad son los que reúnen los requisitos mínimos en busca de la excelencia dentro de una organización institucional (Pulido, 2004). Estos son muy importantes en las empresas, pues permiten definir un conjunto de criterios de desarrollo que guían la forma en que se aplica la Ingeniería de Software. Los estándares suministran los medios para que todos los procesos se realicen de la misma forma y son una guía para lograr la productividad y la calidad.

Los Modelos y Estándares permiten que los desarrolladores de software realicen sus tareas y funciones teniendo en cuenta la calidad, la cual desempeña un rol determinante para la competitividad que adquieren los productos en el mercado. Cuando una empresa está funcionando y decide implantar un Modelo o Estándar de Calidad del Software, es señal de

que la empresa tiene el propósito de permanecer y crecer en el mercado, ser competitiva, proteger los intereses de los clientes, cuidar la fuente de trabajo y mejorar la calidad de vida de su personal.

1.3.1 Estándar ISO 9000.

Según, Andrés Senlle y Rosa Torres, "La ISO 9000, apuesta por el futuro y crea normas para facilitar el comercio, así como para que todas las organizaciones sean prósperas, competitivas y rentables, es la única forma para el desarrollo de los países"

Definición General de ISO 9000

La Organización Internacional de Estandarización, son diseños de normas cuya finalidad es el aseguramiento de la calidad de las empresas de productos y servicios que son desarrolladas en las llamadas serie 9000.

La serie ISO 9000 es un conjunto de normas orientadas a ordenar la gestión de la empresa que han ganado reconocimiento y aceptación internacional debido al mayor poder que tienen los consumidores y a la alta competencia internacional acentuada por los procesos integracionistas. Algunas de estas normas especifican requisitos para sistemas de calidad (ISO 9001, 9002, 9003) y otras dan una guía para ayudar en la interpretación e implementación del sistema de calidad (ISO 9000-2, ISO 9004-1). (López, 2001)

Con ISO 9000, se mejora la eficacia y rentabilidad de la organización, ya que al implantar un sistema de calidad se reduce los costos y se aprovecha más racionalmente los recursos, los tiempos o la capacidad de las personas. También permite potenciar la comunicación interna y mejorar las relaciones que se dan entre los diferentes integrantes de la organización, hace que todo el personal sea conciente y trabaje con un fin común.

La ISO 9000, posibilita que los clientes sean fieles, y que se distingan positivamente por la competencia y los deseos de mantener la supervivencia de la empresa. Su finalidad es que, una empresa proporcione productos o servicios según las expectativas del cliente, de tal forma que se pueda prevenir fallas durante el proceso y aplicar las medidas para eliminarlos, siendo así sea una empresa rentable y competitiva.

La ISO 9000, no es un estándar de producto. No contiene ningún requerimiento con el cual un producto o servicio tenga que cumplir. Sin embargo, puede comprobar si un producto en

concreto, tiene un cierto registro, una identidad trazable en los planos correspondientes, así como condiciones de inspección.

Objetivos de las Normas ISO 9000. (López, 2001)

ISO 9000, tiene cuatro objetivos primordiales.

1. Proporcionar elementos a una organización para que pueda lograr la calidad de su producto o servicio, mantenerla a tiempo, así poder satisfacer las necesidades del cliente permanentemente.
2. Establecer directrices, mediante las cuales la organización puede seleccionar y utilizar las normas.
3. Proporcionar a la dirección de la empresa la seguridad de que se obtiene la calidad deseada.
4. Proporcionar a los clientes o usuarios, la seguridad de que el producto o servicios tengan la calidad deseada, concertada, pactada o contratada.

A pesar de que esta norma brinda la posibilidad de registrar cada uno de los productos que se desarrollan en una empresa u organización, no proporciona funcionalidades para medir la calidad del proceso de desarrollo del software, pues no define requerimientos con los que este deba cumplir para estar en un nivel u otro. Por lo que las autoras de la investigación consideran, que no es favorable su utilización para el aseguramiento de la calidad a niveles de organización.

1.3.2 Modelo de calidad CMMI (Capability Maturity Model Integration)

El desarrollo del CMMI (Capability Maturity Model Integration) fue emprendido en el Instituto de la Tecnología de Dotación Lógica de Carnegie Mellons (SEI) que comenzaba en 1986 bajo patrocinio del departamento de defensa de Estados Unidos. El objetivo fundamental de la aplicación de este modelo en las empresas, es contribuir a mejorar su proceso de desarrollo de software.

El CMMI es un modelo orientado a la mejora de procesos relacionados con el desarrollo de software, para lo cual considera un conjunto de mejores prácticas de ingeniería y gestión. Originalmente, el modelo considera cinco niveles para clasificar a las organizaciones en su nivel de madurez, atendiendo a sus prácticas, metodologías y gestión en los procesos de desarrollo y mantenimiento de software, estos niveles según Joaquín García en su artículo

CMM-CMMI del 2003, son los siguientes:

Nivel 1 o Inicial: en este nivel están todas las empresas que no tienen procesos, es decir que no poseen una correcta planificación, organización y control del trabajo, por lo que los presupuestos se disparan, no es posible entregar los proyectos en fecha, y se tienen que redoblar los esfuerzos para garantizar la terminación del proyecto. Todo esto es consecuencia de la falta de control sobre el estado de los proyectos, siendo desconocido como marcha el desarrollo de los mismos.

Nivel 2 o Repetible: en este nivel el éxito de los resultados obtenidos en otro proyecto se puede repetir. La principal diferencia entre este nivel y el anterior es que el proyecto es gestionado y controlado durante el desarrollo del mismo. El desarrollo no es opaco y se puede saber el estado del proyecto en todo momento. Los procesos que hay que implantar para alcanzar este nivel son:

- Gestión de requisitos.
- Planificación de proyectos.
- Seguimiento y control de proyectos.
- Gestión de proveedores.
- Aseguramiento de la calidad.
- Gestión de la configuración.

Nivel 3 o Definido: cuando una empresa alcanza este nivel significa que la forma de desarrollar proyectos (gestión e ingeniería) esta definida, es decir, que esta establecida, documentada y que existen métricas (obtención de datos objetivos) para alcanzar objetivos concretos. Los procesos que hay que implantar para alcanzar este nivel son:

- Desarrollo de requisitos.
- Solución Técnica.
- Integración del producto.
- Verificación.
- Validación.
- Desarrollo y mejora de los procesos de la organización.
- Definición de los procesos de la organización.

- Planificación de la formación.
- Gestión de riesgos.
- Análisis y resolución de toma de decisiones.

La mayoría de las empresas que llegan al nivel 3 paran aquí, ya que es un nivel que proporciona muchos beneficios y no ven la necesidad de ir más allá porque tienen cubiertas la mayoría de sus necesidades.

Nivel 4 o Gestionado: en este nivel los proyectos se trazan objetivos medibles para lograr satisfacer las necesidades de los clientes y de la organización. Además se utilizan métricas para gestionar la organización. Los procesos que hay que implantar para alcanzar este nivel son:

- Gestión cuantitativa de proyectos.
- Mejora de los procesos de la organización.

Nivel 5 u Optimizado: en este nivel los procesos de los proyectos y de la organización están orientados a la mejora de las actividades. Mejoras incrementales e innovadoras de los procesos que mediante métricas son identificadas, evaluadas y puestas en práctica. Los procesos que hay que implantar para alcanzar este nivel son:

- Innovación organizacional.
- Análisis y resolución de las causas.

Normalmente las empresas que intentan alcanzar los niveles 4 y 5 lo realizan simultáneamente ya que están muy relacionados. (García, 2003)

En la siguiente figura, tomada de (Menéndez, 2007) se muestra de forma gráfica los niveles del CMMI anteriormente explicados.

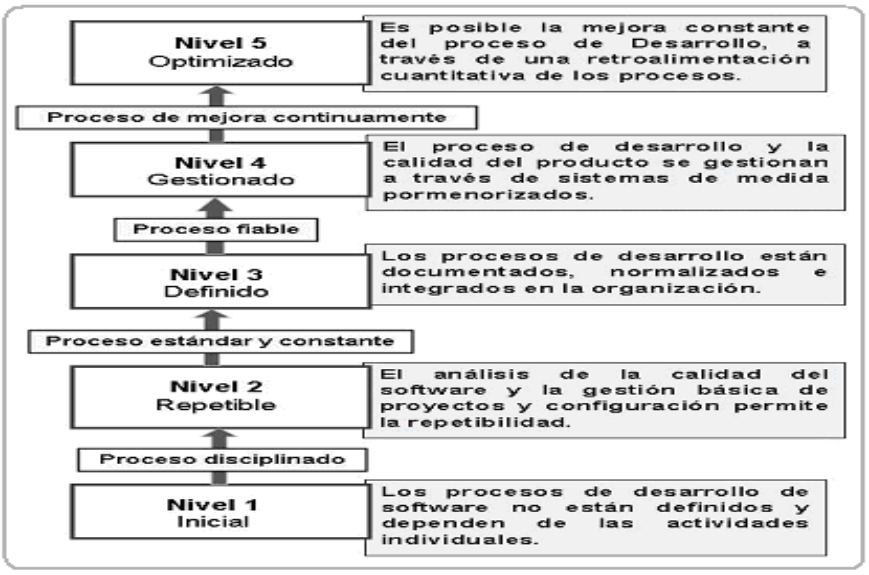


Figura 3. Niveles del CMMI.

Con la aplicación del CMMI, las empresas pueden obtener varios beneficios, que posibilitaran una mejora en su desarrollo de software, tales como:

- Se explica de mejor manera el vínculo entre la gestión y las actividades de ingeniería para los objetivos de negocio.
- Se amplía el alcance y la visibilidad dentro del ciclo de vida de productos y actividades de ingeniería para permitir que el producto o el servicio responda a las expectativas del cliente.
- Se incorpora el aprendizaje de mejores prácticas a otras áreas como manejo del riesgo.
- Se implementan prácticas más robustas y maduras en la organización.

Luego de analizar todo lo planteado del CMMI, se hace evidente, que si lo que se desea es una institución que posea un termómetro de calidad, este es el estándar ideal.

Conclusión

En este capítulo se realizó un estudio de cuales son los aspectos que garantizan mejorar el proceso de documentación de los proyectos productivos, así como un análisis de las principales características de las metodologías ágiles, y algunos de los aspectos significativos que las hacen mejores y diferentes a las pesadas.

Todo esto permitió que se escogiera para el desarrollo de la investigación las metodologías

XP y Scrum, ¿Por qué escoger este binomio?, pues sencillo y es que Scrum se enfoca en las prácticas de organización y gestión, mientras que XP se centra más en las prácticas de programación. Esa es la razón de que funcionen tan bien juntas: tratan de áreas diferentes y se complementan entre ellas. Además de ser las metodologías utilizadas en los proyectos productivos, de la Facultad 10, del Polo de Software Libre, con el nombre de MA-GMPR-UR2.

Para complementar y medir el trabajo que organizan las metodologías se escogió el estándar de calidad CMMI, pues este permite la evaluación por niveles, lo que garantiza que se vaya teniendo una visión de la calidad con la que está trabajando la empresa u organización.

CAPÍTULO 2: ELEMENTOS ESENCIALES PARA LA ELABORACIÓN DE UN EXPEDIENTE DE PROYECTO.

Introducción

Hacer un estudio de las características y estado de los proyectos productivos del Polo de Software Libre, de la Facultad 10, es un eslabón esencial a la hora de revolucionar y optimizar el proceso de recopilación de información de los mismos. Para poder conformar un expediente de proyecto que satisfaga las necesidades de cada uno de estos proyectos, se hace indispensable identificar aquellos problemas que retrasan el desarrollo de los expedientes, y dentro de estos, las dificultades que hacen tedioso el proceso de documentación del software.

2.1 Clasificación de los proyectos.

Es importante tener clasificados los proyectos productivos que se desarrollen en una organización, pues esto posibilita que el trabajo con ellos sea más factible y ágil. Pero no es 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.

Pero a pesar de lo contradictorio que puede ser este tema de la clasificación de los proyectos informáticos, según lo planteado en (González, 2006) 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.

A pesar de que las clasificaciones anteriores tienen aspectos que son relevantes, como es el caso de que se exponen muchas de las actividades que se llevan a cabo en los proyectos informáticos, las autoras de esta investigación consideran, que para el trabajo con los proyectos productivos de la Facultad 10, del Polo de 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.

A continuación se muestran las nuevas tipologías que permitirán conocer la clasificación de los proyectos productivos, del Polo de Software Libre, cuando se realice su caracterización:

- **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.

2.2 Características de los proyectos del Polo de Software Libre de la Facultad 10.

Los proyectos productivos pueden ser caracterizados teniendo en cuenta disímiles puntos, que sirven de guía para conocer su estado y aspectos específicos relevantes.

En la Facultad 10, de la Universidad de las Ciencias Informáticas, la organización de los proyectos productivos está dada por áreas de desarrollo o polos, pero solo se analizarán en esta investigación los proyectos que pertenecen al polo de Software Libre, los que a su vez están agrupados en grupos de trabajo, los que se exponen a continuación.

Grupos de trabajo	Proyectos que desarrollan.
Unicornios	Gestión de investigación y evento.
	MINPPAL
	N-Internos.
	Observatorio de tecnologías libres.
	Octave Front-End.
	Portal de ajedrez.
	Portal de SWL 2.
	Sistema de clonación de imágenes
	ServiceDesk.
	Tocororo desktop.
	Repositorio
	Revista de SWL: UXi.
NOVA	Summon.
	Instalador.
	Ecumenix.
	Nova-Docencia.
	Nova-Producción.
RINDE	Plataformas de Productividad (RINDE 3ra Etapa).

Para llevar a cabo la recogida de los datos, que permitieron realizar una caracterización de cada uno de estos proyectos, teniendo en cuenta importantes parámetros, que son la base de una posterior clasificación y agrupación de los mismos, se aplicó la técnica de la entrevista, la cual se realizó a cada uno de los líderes de proyecto.

Según lo planteado en (Otero, 2008), los tipos de entrevistas son tres:

- *Entrevistas estructuradas*, las cuales consisten en proporcionar cuestionarios estructurados, en los cuales las preguntas están predeterminadas tanto en su secuencia como en su formulación. Es decir, el entrevistador formula en la mayoría de los casos un número fijo de preguntas de forma estándar y en el mismo orden. Las respuestas también están prefijadas de antemano.
- *Entrevistas semiestructuradas*, al igual que las anteriores las preguntas están definidas previamente en un guión de entrevista pero la secuencia, así como su formulación pueden variar en función de cada sujeto entrevistado. Es decir, el (la) investigador(a) realiza una serie de preguntas, generalmente abiertas al principio de la entrevista, que definen el área a investigar, pero tiene libertad para profundizar en alguna idea que pueda ser relevante, realizando nuevas preguntas.
- *Entrevistas en profundidad*, también denominadas entrevistas abiertas. Generalmente suelen cubrir solamente uno o dos temas pero con mayor

profundidad. El resto de las preguntas que el investigador realiza, van emergiendo de las respuestas del entrevistado y se centran fundamentalmente en la aclaración de los detalles con la finalidad de profundizar en el tema objeto de estudio.

Para llevar a cabo la entrevista realizada a los proyectos, definida como estructurada, después de analizar las características de cada uno de los tipos existentes, fue preciso la elaboración de un guión de la entrevista (ver anexo 1) donde se tuvieron en cuenta todos los aspectos importantes que podían ser olvidados, así como para tener y definir un orden en el proceso de recopilación de información. Esta entrevista tuvo como objetivo fundamental la familiarización con las condiciones actuales de los proyectos de la Facultad 10, mediante la identificación de los elementos necesarios.

A continuación queda expuesta la caracterización de los proyectos, teniendo en cuenta algunos parámetros fundamentales.

2.2.1 Proyectos del grupo de trabajo Nova.

Características de Summon.

- *Tipo de proyecto*: Proyecto de desarrollo de aplicaciones.
- *Cantidad de integrantes*: 4 personas.
- *Roles:* Programador, Cliente y Gerente**.**
- *Entorno de desarrollo:* Gnome.
- *Envergadura:* Media.
- *Tiempo de duración estimado:* Indefinido.
- *Complejidad:* Media.
- *Organización:* Equipos de desarrollo.
- *Metodología utilizada:* XP para el desarrollo, Scrum para la gestión.
- *Problemas en el proceso de documentación:*
  - Conflictos entre la metodología en la que esa basada el expediente de proyecto y la metodología utilizada

Características de Instalador.

- *Tipo de proyecto:* Proyecto de desarrollo de aplicaciones.

- *Cantidad de integrantes*: 4 personas.
- *Roles:* Programador, Cliente, Gerente.
- *Entorno de desarrollo:* Gnome.
- *Envergadura:* Media.
- *Tiempo de duración estimado:* Indefinido.
- *Complejidad:* Media.
- *Organización:* Equipos de desarrollo.
- *Metodología utilizada:* XP para el desarrollo, Scrum para la gestión.
- *Problemas en el proceso de documentación*:
  - Conflictos entre la metodología en la que esa basada el expediente de proyecto y la metodología utilizada.

Características de Ecumenix.

- *Tipo de proyecto*: Proyecto de desarrollo de aplicaciones.
- *Cantidad de integrantes*: 4 personas.
- *Roles:* Programador, Cliente, Gerente.
- *Entorno de desarrollo*: Gnome.
- *Envergadura:* Baja.
- *Tiempo de duración estimado:* 4 meses
- *Complejidad:* Baja.
- *Organización:* Equipos de desarrollo
- *Metodología utilizada:* XP para el desarrollo, Scrum para la gestión.
- *Problemas en el proceso de documentación*:
  - Conflictos entre la metodología en la que esa basada el expediente de proyecto y la metodología utilizada.

Características de Nova-Docencia.

- *Tipo de proyecto*: Proyecto de personalización.
- *Cantidad de integrantes*: 22 personas.

- *Roles:* Especialista en sistema base, mantenedor del núcleo, especialista en Live CD, analista del sistema, líder de subproyecto, programador, especialista en sistema, mantenedor de paquetes, líder de grupo de proyecto, arquitecto del grupo de proyecto, cliente, manager, administrador de la infraestructura.
- *Entorno de desarrollo:* Linux.
- *Envergadura:* Alta.
- *Tiempo de duración estimado:* 6 meses
- *Complejidad:* Alta.
- *Organización:* Equipos por roles.
- *Metodología utilizada:* Una metodología creada en el proyecto y que aún no tiene nombre.
- *Problemas en el proceso de desarrollo*:
  - Bajo rendimiento del hardware en la PC de los clientes.

Características de Nova-Producción.

- *Tipo de proyecto:* Proyecto de personalización.
- *Cantidad de integrantes:* 22 personas.
- *Roles:* Especialista en sistema base, mantenedor del núcleo, especialista en Live CD, analista del sistema, líder de subproyecto, programador, especialista en sistema, mantenedor de paquetes, líder de grupo de proyecto, arquitecto del grupo de proyecto, cliente, manager, administrador de la infraestructura.
- *Entorno de desarrollo:* Linux.
- *Envergadura:* Alta.
- *Tiempo de duración estimado:* 6 meses
- *Complejidad:* Alta.
- *Organización:* Equipos por roles.
- *Metodología utilizada:* Una metodología creada en el proyecto y que aún no tiene nombre.
- *Problemas en el proceso de desarrollo:*
  - Bajo rendimiento del hardware en la PC de los clientes.

2.2.2 Proyectos del grupo de trabajo RINDE.

Características de Plataformas de Productividad (RINDE Tercera Etapa).

- *Tipo de proyecto:* Proyecto de desarrollo de aplicaciones.
- *Cantidad de integrantes*: 6.
- *Roles:* Analista, Arquitecto, Jefe de proyecto, Desarrolladores.
- *Entorno de desarrollo*: CMS-Drupal, CodecIgniter y el Zend Estudio.
- Envergadura: Alta.
- Tiempo de duración estimado: Largo plazo.
- *Complejidad:* Alta.
- Organización: equipo de desarrollo.
- Metodología utilizada: RUP.
- Problemas en el proceso de desarrollo:
  - Dificultades con el hardware para el uso de algunas herramientas y para la simulación del entorno final de la arquitectura.
- *Problemas en el proceso de documentación:*
  - Se comenzó a documentar tarde.

2.2.3 Proyectos del grupo de trabajo Unicornios.

Características de MINPPAL.

Es un proyecto totalmente de investigación, donde precisamente su producto es el Plan de migración del MINPPAL, a diferencia de los demás proyectos, que desarrollan software, por lo que hay algunas características, tales como entorno y metodología de desarrollo, que no están definidas.

- *Tipo de proyecto:* Proyecto de investigación.
- *Cantidad de integrantes:* 11 personas.
- *Roles:* Gerente, Especialista de migración, Especialista en base de dato, Cliente.
- *Envergadura:* Media.

- *Tiempo de duración estimado*: Corto plazo.
- *Complejidad:* Media.
- *Organización:* Equipo de investigación.
- *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.
- *Problemas en el proceso de documentación:*
  - Existe redundancia en la mayoría de la documentación que se debe de llenar.
  - Existe información que debe ser almacenada por el proyecto, y no hay plantillas para su recopilación.

Características de Gestión de investigación y evento.

- *Tipo de proyecto*: Proyecto de desarrollo web.
- *Cantidad de integrantes*: 2.
- *Roles:* Analista, Diseñador, Gerente, Programador.
- *Entorno de desarrollo:* LAMP (Linux, Apache, MySQL, PHP).
- *Envergadura:* Media.
- *Tiempo de duración estimado:* 18 meses.
- *Complejidad:* Media.
- *Organización:* Módulo.
- *Metodología utilizada:* XP para el desarrollo, Scrum para la gestión.
- *Problemas en el proceso de desarrollo:*
  - Falta de disponibilidad de personal.
  - Falta de preparación del personal.
  - Inexistencia del proceso que se va a automatizar.
  - El cliente desconoce el proceso.

Características de N-internos.

- *Tipo de proyecto:* Proyecto de desarrollo de aplicaciones e investigación.
- *Cantidad de integrantes:* 9.
- *Roles:* Gerente, Cliente, Analista, Arquitecto, Programador, Tester.
- *Entorno de desarrollo:* LAMP + Javascript + vlc.
- *Envergadura*: Alta.
- *Tiempo de duración estimado:* 12 meses.
- *Complejidad:* Alta.
- *Organización:* Equipo por roles.
- *Metodología utilizada:* RUP.
- *Problemas en el proceso de desarrollo:*
  - Lentitud del proceso de comunicación con los clientes.
  - Exceso de trabajo del analista.
  - Desconocimiento de las tecnologías.
  - No disponibilidad del diseño o identidad, y tampoco de la arquitectura de información.
  - Alta necesidad de investigación de tecnologías que se conocen pero cuya manipulación no es sencilla.
- *Problemas en el proceso de documentación:*
  - Excelente documentación.
  - Exceso de documentación.

Características de Observatorio de tecnologías libres.

- *Tipo de proyecto:* Proyecto de desarrollo de aplicaciones.
- *Cantidad de integrantes*: 2.
- *Roles:* Desarrollador, Revisor técnico.
- *Entorno de desarrollo:* LAMP.
- *Envergadura:* Baja.

- *Tiempo de duración estimado:* 12 meses.
- *Complejidad:* Baja.
- *Organización:* Equipo por roles.
- *Metodología utilizada:* RUP.
- *Problemas en el proceso de desarrollo:*
  - Problemas en la organización del equipo.
  - Estudiantes desmotivados, con problemas docentes.
  - Falta de liderazgo de equipo.
- *Problemas en el proceso de documentación:*
  - Mucha documentación, pocos resultados.

Características de Octave Front-End.

- *Tipo de proyecto:* Proyecto de desarrollo de aplicaciones.
- *Cantidad de integrantes*: 6
- *Roles:* Gerente, Analista, Arquitecto, Tester, Programador.
- *Entorno de desarrollo:* JAVA.
- *Envergadura:* Media.
- *Tiempo de duración estimado:* 12 meses.
- *Complejidad:* Alta.
- *Organización:* Módulos.
- *Metodología utilizada:* XP para el desarrollo, Scrum para la gestión.
- *Problemas en el proceso de desarrollo:*
  - Desconocimiento de las tecnologías.
  - Progreso, por su calidad, del software competidor de igual tipo, es decir del Software Libre.
  - Problemas de organización.
- *Problemas en el proceso de documentación:*
  - Buena recopilación de información.

- El equipo no responde a los intereses del grupo. No documenta. No coloca en el subversion.

Características de Portal de Ajedrez: Infodrez 1.2.

- *Tipo de proyecto:* Proyecto de desarrollo web y servicios.
- *Cantidad de integrantes*: 2.
- *Roles*: Analista, Diseñador, Gerente, Programador.
- *Entorno de desarrollo:* LAMP.
- *Envergadura:* Media.
- *Tiempo de duración estimado:* 12 meses.
- *Complejidad:* Media.
- *Organización:* Equipo por roles.
- *Metodología utilizada:* XP para el desarrollo, Scrum para la gestión.

Características de Portal de SWL 2.

- *Tipo de proyecto:* Proyecto de desarrollo web y servicios.
- *Cantidad de integrantes*: 5.
- *Roles:* Gerente, Programador, Tester.
- *Entorno de desarrollo:* Python para la Web., CMS - Zope/Plone.
- *Envergadura:* Media.
- *Tiempo de duración estimado:* 12 meses.
- *Complejidad:* Media.
- *Organización:* Equipo por roles.
- *Metodología utilizada:* XP para el desarrollo, Scrum para la gestión.
- *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 de aplicaciones y servicios.
- *Cantidad de integrantes*: 8.
- *Roles:* Gerente, Cliente, Analista, Arquitecto, Programador, Diseño.
- *Entorno de desarrollo:* LAMP.
- *Envergadura:* Alta.
- *Tiempo de duración estimado:* 24 meses.
- *Complejidad:* Media.
- *Organización:* Equipo por roles.
- *Metodología utilizada:* XP para el desarrollo, Scrum para la gestión.
- *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.
  - Lentitud del equipo.
- *Problemas en el proceso de documentación:*
  - No hay problemas porque se comenzó con una tesis que realizaba el estudio del arte.

Sistema de Clonación de imágenes.

- *Tipo de proyecto:* Proyecto de desarrollo de aplicaciones e investigación.
- *Cantidad de integrantes:* 10.
- *Roles:* Gerente, Cliente, Analista, Arquitecto, Programador, Tester, Diseño.
- *Entorno de desarrollo:* C++ (Ncurse).
- *Envergadura:* Alta.
- *Tiempo de duración estimado:* 36 meses.
- *Complejidad:* Alta.

- *Organización:* Equipo por roles.
- *Metodología utilizada:* Comenzó con RUP, pasó luego a XP para el desarrollo, Scrum para la gestión.
- *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.
- *Problemas en el proceso de documentación:*
  - Se ha atrasado la documentación.
  - Se estimó el tiempo de manera incorrecta.

Características de Tocororo Desktop.

- *Tipo de proyecto*: Proyecto de desarrollo de aplicaciones e investigación.
- *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.
- *Organización:* Equipo por roles.
- *Metodología utilizada:* XP para el desarrollo, Scrum para la gestión.
- *Problemas en el proceso de desarrollo:*
  - Alta complejidad de diseño conceptual y de las interfaces.
  - Poca preparación del equipo de desarrollo.

Características de Revista de SWL: UXi.

- *Tipo de proyecto:* Proyecto de servicios.
- *Cantidad de integrantes*: 10.

- *Roles:* Gerente, Programador.
- *Entorno de desarrollo:* Openoffice, Gimp, Tecnologías de Internet.
- *Envergadura:* Media.
- *Tiempo de duración estimado:* largo plazo o indefinido.
- *Complejidad:* Alta.
- Organización: Equipo por roles.
- *Metodología utilizada:* XP para el desarrollo, Scrum para la gestión.
- *Problemas en el proceso de desarrollo:*
  - Dificultad para la conformación del equipo.
  - Maduración del equipo.
- *Problemas en el proceso de documentación:*
  - Excelente documentación.

Características de Repositorio.

- *Tipo de proyecto:* Proyecto de servicios.
- *Cantidad de integrantes:* 2.
- *Roles:* Gerente, Programador.
- *Entorno de desarrollo:* Linux, Apache.
- *Envergadura:* Media.
- *Tiempo de duración estimado:* Largo plazo o indefinido.
- *Complejidad:* Baja.
- *Organización:* Equipo por roles.
- *Metodología utilizada*: RUP
- *Problemas en el proceso de desarrollo*:
  - Personas aisladas, no había equipo.
  - Sencilla la tecnología.
- *Problemas en el proceso de documentación*
  - No existe documentación.

2.2.4 Singularidades de las caracterizaciones.

Al analizar las características antes expuestas, obtenidas de las entrevistas realizadas a los líderes de proyectos de la Facultad 10, del Polo de Software Libre, se puede afirmar que se identificaron cinco tipos de proyectos: el de desarrollo de aplicaciones, desarrollo Web, investigación, personalización y servicios; aunque existe un solapamiento de tipos, es decir existen algunos proyectos que por sus características específicas – y el desarrollo del ciclo de vida del software que producen – pueden clasificarse en uno o varios tipos de proyectos anteriormente definidos.

Un proyecto en su etapa inicial puede ser de desarrollo de aplicaciones e investigación, pues no solo es importante la producción, sino el conocimiento total de todo lo que esté relacionado con el tipo de software que se está desarrollando, ya sean herramientas, tecnologías u otros software, y esto solo puede conseguirse con un estudio investigativo del tema que se está tratando.

Luego puede suceder que un software que fue desarrollado por un proyecto, se dedique sólo a prestar servicios a usuarios o clientes, entonces ya este proyecto tendría otra clasificación, y puede incluso suceder que se necesite continuar investigando, para realizar mejoras en el producto.

Todas estas situaciones se aprecian en el proceso de desarrollo de software, de los proyectos productivos de la Facultad 10, del Polo de Software Libre, y más que una dificultad es una forma de perfeccionar el proceso productivo, formando grupos de trabajo capaces de realizar tareas diversas, y explotar al máximo las potencialidades de un software, teniendo como único elemento indispensable las clasificaciones consideradas por los proyectos, así como las actividades a realizar por los mismos en cada una de ellas. De manera general el tipo de proyecto que más prevalece es el de desarrollo de aplicaciones.

La cantidad de personas que involucran cada uno de los proyectos se incluyen en tres grupos, menos de 10, de 10 a 20 y de 20 a 25, con mayor incidencia en los de menos de 10 miembros. Las tecnologías más utilizadas son las LAMP y las tecnologías libres, teniendo las LAMP mayor por ciento de usabilidad, aunque no muy significativo.

La mayoría de los proyectos del Polo de Software Libre, de la Facultad 10, utilizan metodologías ágiles, entre ellas XP para el desarrollo y Scrum para la gestión, los restantes utilizan RUP, o alguna metodología específica del proyecto, excepto los investigativos, para los cuales no existe aún una metodología adaptable.

Todos los proyectos involucran los roles principales de las metodologías ágiles utilizadas (cliente, programador, gerente, etc.). Generalmente son proyectos de complejidad alta, en solo tres es baja y los problemas en cuanto a documentación y desarrollo son comunes en casi la totalidad de los proyectos.

De lo antes expuesto se puede decir además que los proyectos de la Facultad 10, del Polo de Software Libre, en su gran mayoría son proyectos de desarrollo de aplicaciones, con una alta complejidad, caracterizados por ciclos relativamente cortos de desarrollo y un número reducido de integrantes y en los casos de ser amplio dicho número, la organización es por equipos pequeños de desarrollo.

En la Facultad 10 casi la totalidad de los proyectos del Polo de Software Libre, utilizan metodologías ágiles para guiar el proceso de desarrollo y gestión de los mismos, por ser la más factible para su desarrollo productivo, de ahí que esto sea un problema seriamente influyente en el proceso de recopilación de evidencias, pues el expediente que debe registrar la información está basado en RUP.

Para comprender mejor el análisis realizado en los proyectos productivos del Polo de Software Libre, de la Facultad 10, se realizaron gráficas de pastel (ver anexo 2), donde se muestran los resultados gráficamente.

2.3 Análisis del expediente de proyecto vigente en la Universidad de las Ciencias Informáticas.

Es muy importante para comprender y optimizar los problemas existentes actualmente en el proceso de documentación de los proyectos productivos de la Facultad 10, del Polo de Software Libre, conocer cuales son las fortalezas y debilidades del expediente de proyecto que esta en vigencia en la UCI, y que fue elaborado por la Dirección de Calidad de Software de la Universidad.

Es imprescindible resaltar que debido a la falta de referencias en las diferentes bibliografías consultadas, de expedientes de proyecto en Cuba y en el mundo, el análisis de este expediente se realizará teniendo en cuenta sus características específicas, y los problemas detectados en los proyectos productivos.

2.3.1 Características del expediente de proyecto.

Es importante tener presente que el proceso de documentación de un software, es un aspecto significativo en su desarrollo, por lo que la documentación asociada a los proyectos de software y sistemas debe cumplir con algunos requisitos imprescindibles, que según el ingeniero Ramsés Delgado son:

- Servir como medio de comunicación entre los miembros del equipo.
- Servir de repositorio de información que pueda ser utilizado por los ingenieros de sistemas.
- Proveer información para el control de los planes, cronogramas e hitos en el proceso de desarrollo de software.
- Definir quién hace y cómo hace las actividades específicas del desarrollo.

Satisfacer estos requerimientos implica la realización de un grupo de artefactos que no son solamente las documentaciones técnicas, asociadas al software. En varias ocasiones se hace imposible organizar la documentación de los proyectos si no se define un esquema genérico que permita su control y que norme como usarla y evaluarla según los tipos de proyectos. Es por eso que la Dirección de Calidad de Software de la UCI, para lograr una mejor organización en los proyectos de desarrollo de software definió una estructura para el expediente de proyecto actual, así como un grupo de plantillas. (Ver anexo 3)

Esta estructura garantiza que el expediente tenga una excelente organización, pero a pesar que la documentación que se recoge en él, cumple con cada uno de los requisitos que se mencionaron anteriormente, las autoras de esta investigación consideran, luego de haber tenido varias entrevistas con los líderes de proyectos, donde se analizaron las deficiencias del proceso de documentación en estos momentos, que se obviaron aspectos significativos, que son indispensables si se quiere obtener una documentación con calidad, tales como:

- Evitar la duplicación de información de una plantilla a otra.
- Garantizar que se registre solamente la información que es imprescindible y obligatoria para comprender el ciclo de vida del software, es decir evitar documentaciones extensas.
- Garantizar la adaptabilidad del expediente para cualquier metodología de desarrollo utilizada por los proyectos productivos.

La ausencia de este último elemento en la estructura del expediente, radica fundamentalmente en que el expediente de proyecto actual, esta basado totalmente en la metodología de desarrollo RUP, por ser la metodología que hasta este momento es aplicada por prácticamente todas las empresas productoras de software del país, además es la que se enseña en el plan de estudio de las universidades y demás centros de estudios informáticos.

Esto como es lógico afecta a los proyectos productivos que aplican otro tipo de metodología de desarrollo, como es el caso de los proyectos productivos de la Facultad 10, del Polo de Software Libre, que en su mayoría trabajan con metodologías ágiles, entre ellas XP para el desarrollo y Scrum para la gestión, de ahí la necesidad de conformar un nuevo expediente que tenga en cuenta las características especificas de estas metodologías.

En cuanto a las normas de calidad, para la confección del expediente actual fueron analizados varios modelos y estándares que norman la documentación de software, como el Modelo de Madurez y Capacidades (CMMI) en su versión 1.2, del cual se tomó el modelo escalonado definido por cada uno de sus niveles de madurez, iniciando por el nivel 2 e incluyendo algunas exigencias del nivel 3. También se tuvieron en cuenta las normas ISO y los estándares de la IEEE. Pero el más importante de los analizados, y que tuvo una mayor influencia en la estructura del expediente fue el CMMI, pues este divide sus Áreas de Procesos (AP) en categorías, de las cuales se tuvieron en cuenta las siguientes:

- Ingeniería.
- Soporte.
- Gestión de proyecto.

Y precisamente estas mismas clasificaciones fueron las utilizadas para agrupar las plantillas en el expediente de proyecto, agregando una cuarta categoría para la documentación legal que, dada su naturaleza y lo delicado de la misma no se consideró conveniente mezclar con documentación técnica. Esta división por áreas hace muy cómodo el trabajo con el expediente, sólo que las plantillas que se incluyen en cada una de ellas, responden totalmente a los artefactos generados por RUP en sus fases, por lo que para los proyectos que trabajan con metodologías ágiles es necesario adaptar estos aspectos definidos por CMMI, a las características de las mismas.

2.3.2 Análisis de las áreas que conforman la estructura del expediente de proyecto.

Como se había planteado anteriormente, el expediente de proyecto actual se divide en tres áreas fundamentales, Ingeniería, Soporte y Gestión de proyecto, dentro de las cuales se recogen las plantillas que almacenan la información referente al área.

En el *grupo de Ingeniería* se encuentran los documentos que están directamente relacionados con los procesos de la Ingeniería de Software. Según especialistas de la Dirección de Calidad de Software de la UCI su construcción no está basada en ninguna metodología en específico, aunque si se tienen en cuenta los procesos básicos para la construcción de software. Esto provoca que el expediente sea una herramienta genérica, por lo que muchas metodologías, tales como las ágiles, entre ellas XP y Scrum, no van a quedar satisfechas con la documentación que en él se recoge.

En el *grupo de Gestión de proyecto*, se recogen todos aquellos documentos que son imprescindibles para alcanzar una administración efectiva de un proyecto software, es decir los planes para la ejecución de cada uno de los procesos presentes en el desarrollo de software, así como aquellos aspectos generales del proyecto, tales como cronogramas, recursos, presupuestos, reuniones, etc. Hay que tener presente en este aspecto que pueden variar todas estas medidas de planificación de una metodología a otra.

En el *grupo de Soporte* se encuentran los documentos que garantizan el soporte al software en construcción, y según especialistas de la Dirección de Calidad de Software se encuentra formado por dos subgrupos: Gestión de Configuración de Software y Aseguramiento de la Calidad.

Con la aplicación del expediente, anteriormente caracterizado, en los proyectos productivos, se propició la aparición de las siguientes ventajas:

- Existe una herramienta que guía el proceso de documentación.
- Se logra una mejor organización de la documentación en los proyectos productivos.

Pero como todo lo que se aplica en una organización está sujeto a ser aceptado, o no por los que lo utilizan, también surgieron desventajas, tales como:

- El expediente no es adaptable a cualquier metodología.
- Propicia que el proceso de documentación sea extenso y tedioso para los desarrolladores.

- Los roles, con sus respectivas tareas, están según RUP.

Conclusión

En este capitulo se realizó un estudio de cuales son las clasificaciones más conocidas de los proyectos informáticos en la actualidad, lo que permitió definir cada uno de los tipos de proyectos que existen en la Facultad 10, del Polo de Software Libre. Estas clasificaciones posibilitaron ubicar a cada uno de esos proyectos en una línea de trabajo, conociendo así el tipo de software que desarrollan.

También se realizó una caracterización de cada uno de estos grupos de desarrollo, lo que posibilitó la identificación de cual es su estrategia de trabajo, así como los problemas que presentan en su proceso de desarrollo de software. Otro de los aspectos significativos tratados en este capítulo fue la caracterización y valoración del expediente de proyecto vigente en la UCI y que fue elaborado por la Dirección de Calidad de la universidad, lo que permitió conocer cuales son sus fortalezas y debilidades, para de esta manera contribuir a mejorar este expediente en la nueva propuesta.

CAPÍTULO 3: DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA.

Introducción

Durante el transcurso de esta investigación se han abordado temas relacionados con el proceso de documentación en el ciclo de vida de un software, tales como los elementos más importantes a tener en cuenta para la confección de un expediente de proyecto, entre ellos las metodologías y el modelo de calidad que organice y controle el proceso de desarrollo, así como la importancia que tiene para la reutilización de un software, haber elaborado una documentación con calidad. Además se ha hecho un estudio del proceso productivo de la Facultad 10, en el área de Software Libre, basado en las principales características de los proyectos productivos.

La producción de software incluye aspectos importantes y determinantes, tales como el diseño y la implementación. Pero sin dudas existe un elemento significativo en su desarrollo, y al que en varias ocasiones no se le presta la atención que requiere, se trata de la documentación. No se puede ser esquemático cuando se va a documentar un software, hay que valorar las condiciones y características específicas del proyecto, así como buscar vías alternativas que permitan el logro de una documentación con calidad y que no duplique los esfuerzos de los desarrolladores.

La UCI y principalmente la Facultad 10, no está ajena a la situación antes planteada, pues es lógico que el esquematismo al documentar trae como consecuencias, proyectos con posibilidades de fracasar, otros con un desarrollo seriamente retardado y afectado por el tiempo, y en la mayoría de los casos la imposibilidad de reutilizar el software producido, por falta de información y de referencias sobre el tema.

Precisamente es el objetivo de este capítulo darle solución a dicha problemática, haciendo un uso adecuado de la metodología MA-GMPR-UR2 (Metodologías Ágiles - Gladys Marsi Peñalver Romero -Unicornios Revisión 2), metodología probada en los proyectos productivos del Polo de Software Libre de la Facultad 10 y que conjuga exitosamente el binomio XP – Scrum, así como del modelo de calidad CMMI, en la confección de una propuesta de expediente de proyecto, que satisfaga las necesidades de cada uno de los proyectos productivos de la Facultad 10, del Polo de Software Libre.

Para darle solución al problema planteado se propone elaborar y explicar, una nueva

estructura para el expediente de proyecto, que responda a las características de la metodología utilizada actualmente en los proyectos productivos, del Polo de Software Libre de la Facultad 10 y mencionada anteriormente. La cual está dividida en cuatro fases, que son precisamente la base de la estructura del nuevo expediente de proyecto, estas son:

- Planificación-Definición
- Desarrollo.
- Entrega.
- Mantenimiento.

Cada una de estas fases está compuesta por una serie de actividades que son las que generan los artefactos que quedan incluidos en el nuevo expediente de proyecto, estas actividades están recogidas en el guión de la metodología (ver anexo 4).

Para la definición de los artefactos que se generan en cada una de las fases se tiene en cuenta como elemento fundamental, las características de las metodologías ágiles, las cuales tienen como premisa la no duplicación de esfuerzos, así como la integración del cliente en el equipo de desarrollo, esto garantiza que no haya necesidad de documentaciones extensas, solo se documenta lo necesario para una futura reutilización.

3.1 Procedimientos.

Basados en los elementos abordados anteriormente se procederá a plantear y explicar la propuesta de la estructura del nuevo expediente, para los proyectos del Polo de Software Libre, que trabajan con metodologías ágiles.

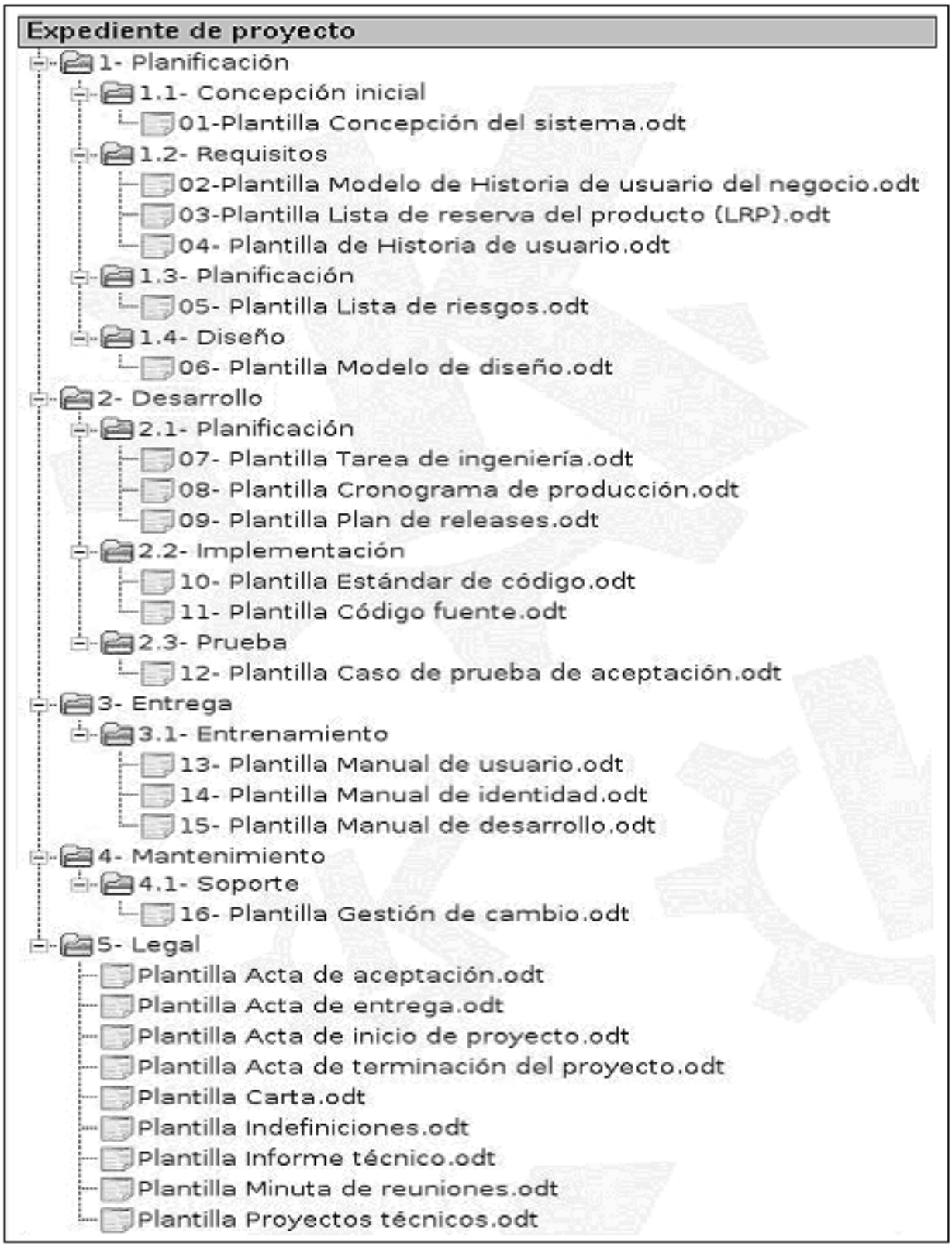


Figura 4. Expediente de proyecto propuesto.

Cada una de las plantillas expuestas en la propuesta son responsabilidad de los diferentes roles que existen dentro del equipo de desarrollo de software, por lo que es importante antes de explicar la estructura del expediente, conocer los roles que define la metodología utilizada.

3.1.1 Definición de roles.

El desarrollo de software con metodologías ágiles exige de la creación de pequeños grupos de trabajo, donde los roles son pocos, pero están bien definidas sus actividades. El principal aspecto antes de comenzar el proceso de documentación es distribuir las tareas por cada uno de los roles existentes, lo que garantiza un trabajo organizado, de ahí la necesidad de tenerlos bien definidos. A continuación se muestran los roles que definen las autoras de la investigación, para los proyectos del Polo de Software Libre, de la Facultad 10, luego de realizar un estudio en capítulos anteriores, sobre las metodologías ágiles XP y Scrum.

- **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 los impedimentos y definir, así como reducir los riesgos del producto. Además:

  - Coordina y facilita las reuniones.
  - Asegura que se consigan los objetivos de la reunión de planificación de la iteración.
  - Determina cuándo es necesario realizar algún cambio para lograr los objetivos de cada iteración.

  Se debe garantizar que la persona que se desempeñe en este rol, posea suficiente experiencia en la gestión de proyectos, además no es conveniente que dirija grupos de trabajos de más de cincuenta personas; este grupo se organiza en pequeños proyectos de no más de diez integrantes.

- **Gerente (Management):**

  Es el responsable de tomar las decisiones finales, acerca de estándares y convenciones a seguir durante el proyecto. Participa en la definición de objetivos y requerimientos, así como en la selección de los miembros del Equipo del Proyecto. Tiene la responsabilidad de controlar el progreso del software y trabaja junto con el

Líder de Proyecto en la reducción de la Lista de reserva del producto, así como en la de riesgos. Además

- 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.

- **Especialista:**

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.

- **Cliente (Customer):**

El cliente contribuye a definir las historias de usuario y los casos de prueba de aceptación, 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 y participa en la concepción inicial del sistema.

El cliente es sólo uno dentro del proyecto pero puede corresponder a un interlocutor que está representando a varias personas que se verán afectadas por el sistema.Contribuye a definir las tareas que involucra la Lista de reserva del producto y permite:

- 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.

- **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.

- **Equipo del Proyecto (Scrum Team):**

  Es el equipo del proyecto quien tiene la autoridad para decidir cómo organizarse para cumplir con los objetivos de una iteración. Sus tareas son: estimar esfuerzo, crear la Reserva de la iteración, revisar la Lista de reserva del producto y sugerir obstáculos que deban ser removidos para cumplir con los items que aparecen.

  Típicamente es un equipo de entre 5 y 10 personas, cada una especializada en algún elemento que conforman los objetivos a cumplir, por ejemplo: programadores, diseñadores de Interfaz de usuario, etc. La dedicación de los miembros del equipo debería ser todo el tiempo, con algunas excepciones. La membresía solo puede cambiar entre iteraciones (no durante).

  El equipo del proyecto estará conformado por otros roles, como son:

  - ***Programadores (Programmers):*** El programador define las tareas de ingeniería, produce el código del sistema y escribe las pruebas unitarias. Además selecciona el estándar de programación a utilizar, controlando incluso la gestión recambios. Debe existir una comunicación y coordinación adecuada entre los programadores y otros miembros del equipo. También dedica parte de su tiempo a la confección de los Manuales de usuario y de desarrollo.
  - ***Analista (Analyst):*** escribe la concepción del sistema y las historias de usuario. Crea el Modelo de historia de usuario del negocio y la LRP. 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):*** son los encargados del diseño del sistema, así como el de 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):*** escribe los casos de prueba de aceptación. 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.

Es importante tener en cuenta, que a pesar de aplicar este tipo de organización por roles en los proyectos productivos, no se debe de crear un esquematismo, ni pensar que un integrante de un grupo de desarrollo, en todo el ciclo de vida de un software solamente se relacionará con su funcionalidad, pues en la mayoría de los casos ocurre un solapamiento de roles, es decir una persona puede desempeñar en diferentes etapas, en dependencia de su responsabilidad inicial, varios roles.

Esto garantiza la creación de un grupo de trabajo fuerte, donde siempre existirán personas preparadas para realizar cualquier actividad que sea necesaria, sin obviar como es lógico la necesidad de formar especialistas en temas específicos dentro de cada proyecto, que podrían convertirse más adelante en consultores.

3.1.2 Planificación - Definición.

La fase de Planificación - Definición, es la primera que define la metodología MA-GMPR-UR2., utilizada por los proyectos productivos de la Facultad 10, del Polo de Software Libre. En esta fase se generan todos los documentos que se encuentran relacionados con la concepción inicial del sistema, así como la definición del mismo. También se incluyen algunos que están vinculados a la primera parte de los procesos de Ingeniería de Software, tales como los relacionados con el negocio, los requisitos y el diseño. Y como también hay una parte de planificación en esta fase, se incluyen aquellos documentos que están relacionados con la estimación inicial de esfuerzos, y la valoración de los riesgos.

Cada una de las plantillas que han sido incluidas en esta fase se generan de una actividad en específico, y tienen su importancia en el proceso de documentación de software, por lo

que son analizadas detalladamente a continuación, describiendo los objetivos que persiguen.

Plantilla de Concepción del sistema.

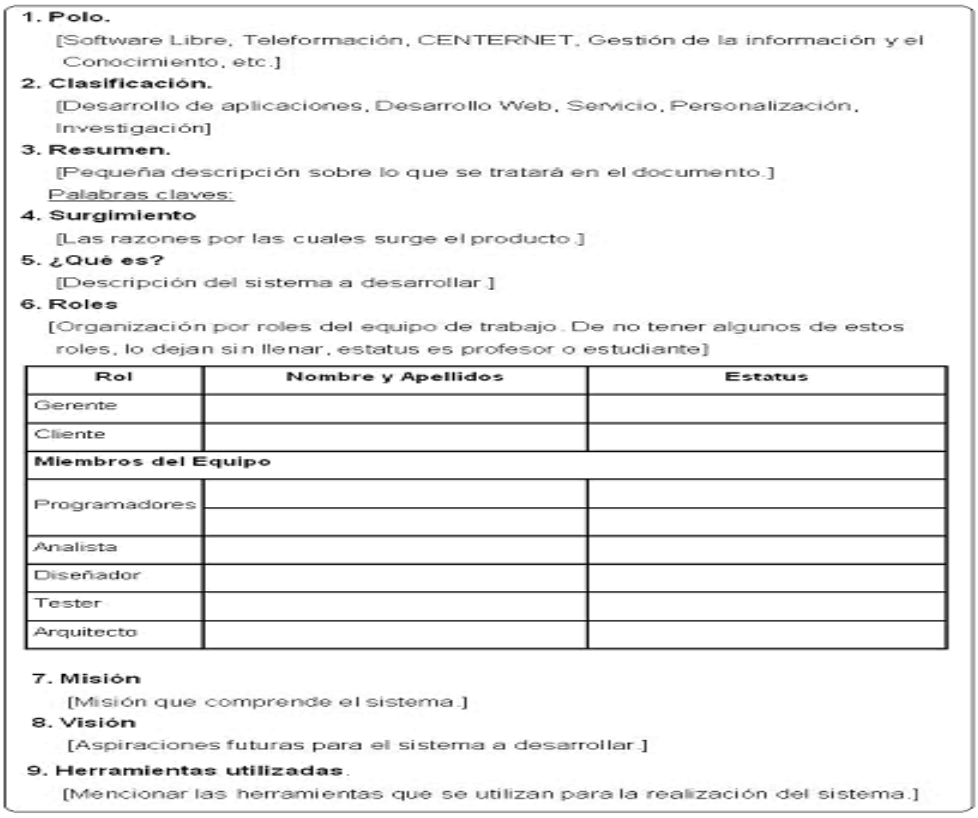
La plantilla de Concepción del sistema, es el primer documento generado en la fase de Planificación-Definición, este queda elaborado luego de realizarse la actividad de Entrevista con el cliente, momento en el que se define la concepción inicial del sistema. Este documento además de reflejar la visión general del producto a implementar, también recoge los diferentes roles que intervendrán en el desarrollo del software, así como las responsabilidades que tendrán en dicho proceso.

Esta plantilla posee gran importancia dentro de la documentación, pues es la base para los demás documentos que se generan durante el ciclo de desarrollo de software. Proporciona ventajas tales como:

- Permite conocer el producto a implementar, pues recoge la concepción inicial del sistema.
- Facilita el proceso de negocio, pues define aspectos relacionados con el mismo.

Roles encargados de la plantilla: Analista.

Para comprender cuales son los datos que se recogerán en esta plantilla, se muestra la misma gráficamente.



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.]

Figura 5. Concepción del sistema

Plantilla del Modelo de Historias de usuario del negocio

La plantilla del Modelo Historias de usuario del negocio, es un artefacto que se genera del Juego de la planificación, luego de estar definida la concepción del sistema, se hace mucho más fácil comprender el negocio.

En esta plantilla se definen las características específicas del negocio, así como la forma en que interactúa el sistema con los clientes y viceversa. El Modelo de negocio cuando se trabaja con metodologías ágiles, es diferente al ya conocido en el proceso unificado, ya que en este caso se trabaja con historias de usuarios, en vez de con casos de uso. Pero independientemente de los cambios técnicos que puedan existir, el negocio se modela igual en cualquier metodología.

Cuando se realiza un buen Modelo del negocio, se obtienen ventajas, tales como:

- Se comprende mejor la interacción del sistema con los usuarios.
- Se facilita la captura de requisitos, pues ya se ha definido el negocio y se tiene una noción de lo que se va a realizar.

Roles encargados de la plantilla: Analista y Arquitecto.

Para comprender cuales son los datos que se recogerán en esta plantilla, se muestra la misma gráficamente.

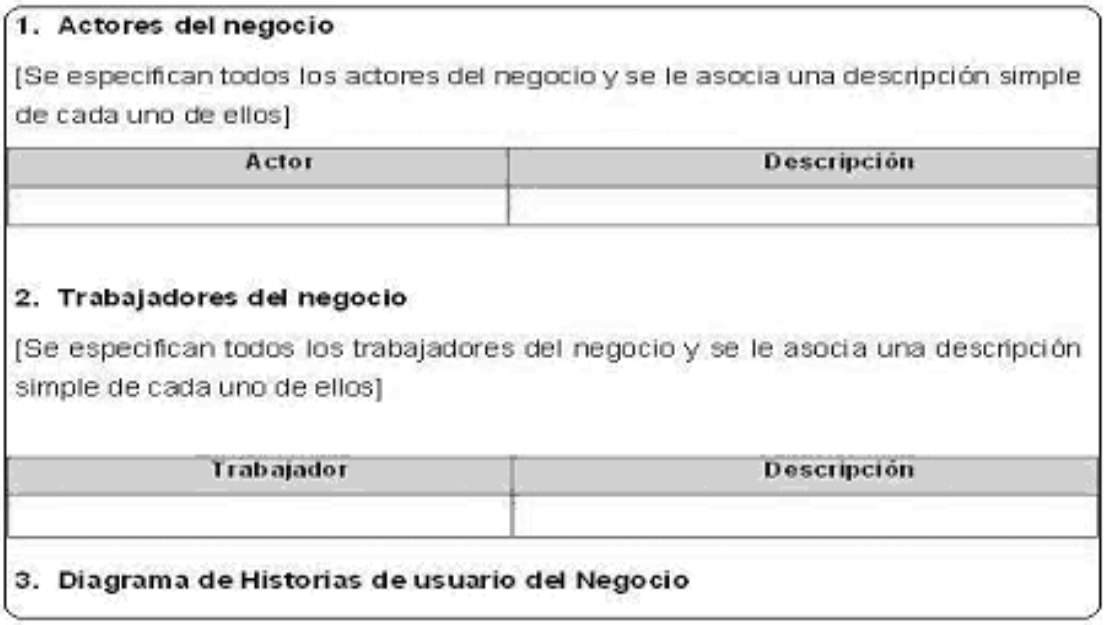


Figura 6. Modelo de Historia de usuario del negocio.

Plantilla de Lista de reserva del producto (LRP).

La plantilla de Lista de reserva del producto, es el primer artefacto generado en la etapa de Captura de requisitos, está conformada por 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 una iteración.

Esta lista puede crecer y modificarse a medida que se obtienen más conocimientos acerca del producto y del cliente. Con la restricción de que sólo puede cambiarse entre iteraciones. 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.

Esta lista puede estar conformada por requerimientos técnicos y del negocio, funciones, errores a reparar, defectos, mejoras y actualizaciones tecnológicas requeridas. Si se realiza el llenado de esta plantilla con calidad, se garantizarán las ventajas siguientes:

- La organización de los requisitos, tanto funcionales como no funcionales, en dependencia de la prioridad que tengan para el desarrollo del sistema.
- Se facilitará el trabajo al confeccionar las historias de usuarios.

Roles encargados de la plantilla: Analista.

Para comprender cuales son los datos que se recogerán en esta plantilla, se muestra la misma gráficamente.

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				

Figura 7. Lista de reserva del producto (LRP).

Plantilla de Historias de usuario.

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 escritas en lenguaje natural, no excediendo su tamaño de unas pocas líneas de texto.

Las historias de usuario guían la construcción de las pruebas de aceptación, elemento clave en XP (deben generarse una o más pruebas para verificar que la historia ha sido correctamente implementada) y son utilizadas para estimar tiempos de desarrollo. En este sentido, sólo proveen detalles suficientes para hacer una estimación razonable del tiempo que llevará implementarlas. En el momento de implementar una historia de usuario, se debe

detallar a través de la comunicación con el cliente. Estas son la base para las pruebas funcionales.

En esta plantilla los campos de puntos estimados y puntos reales se llenan, luego del desarrollo de la actividad de Estimación de esfuerzo, donde se decide que tiempo se le dedicará a cada historia de usuario definida con anterioridad.

Las historias de usuario proporcionan ventajas, tales como:

- Están escritas en lenguaje del cliente, por lo que es muy fácil su compresión.
- Especifican cada uno de los requisitos del sistema, sin necesidad de documentaciones extensas.
- Reflejan todas las características del sistema.
- Si se definen correctamente, guían el proceso de implementación.

Roles encargados de la plantilla: Analista.

Para comprender cuales son los datos que se recogerán en esta plantilla, se muestra la misma gráficamente.

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.]*	

Figura 8. Historia de usuario.

Plantilla de Lista de riesgos.

La plantilla de Lista de riesgos, es el documento que se genera de la actividad de Valoración de riesgos. En ella quedan definidos los posibles riesgos que actuarán sobre el proceso de desarrollo de software, así como la estrategia trazada para mitigarlos. Posee una gran importancia, pues a pesar que es imposible definir desde un inicio todos los riesgos que pueda atravesar un proyecto, si se tendrán algunos en cuenta, fundamentalmente si se trata de un equipo de desarrollo con experiencia. Esta plantilla propicia algunas ventajas, tales como:

- Se definen los posibles riesgos, así como la forma de mitigarlos, lo que disminuye el efecto de los mismos, si ocurrieran.
- Se lleva un control de todos los problemas que han azotado al proyecto, así como de la manera que fueron enfrentados y el impacto que tuvieron en el proceso de desarrollo.

Roles encargados de la plantilla: Gerente y Scrum Master.

Para comprender cuales son los datos que se recogerán en esta plantilla, se muestra la misma gráficamente.

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.]*

Figura 9. Lista de riesgo.

Plantilla del Modelo de diseño.

La plantilla del Modelo de diseño, es el documento que se genera del Diseño con las metáforas, donde se debe diseñar la solución más simple que pueda funcionar y ser implementada en un momento determinado del proyecto. En XP no se enfatiza la definición temprana de una arquitectura estable para el sistema. Dicha arquitectura se asume de forma 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.

Teniendo en cuenta las características anteriores, 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.

Esta plantilla proporciona ventajas, tales como:

- Permite confeccionar un diseño inicial y sencillo del sistema.
- Es la base para le definición de una futura arquitectura.

Roles encargados de la plantilla: Diseñador, Arquitecto.

Para comprender cuales son los datos que se recogerán en esta plantilla, se muestra la misma gráficamente

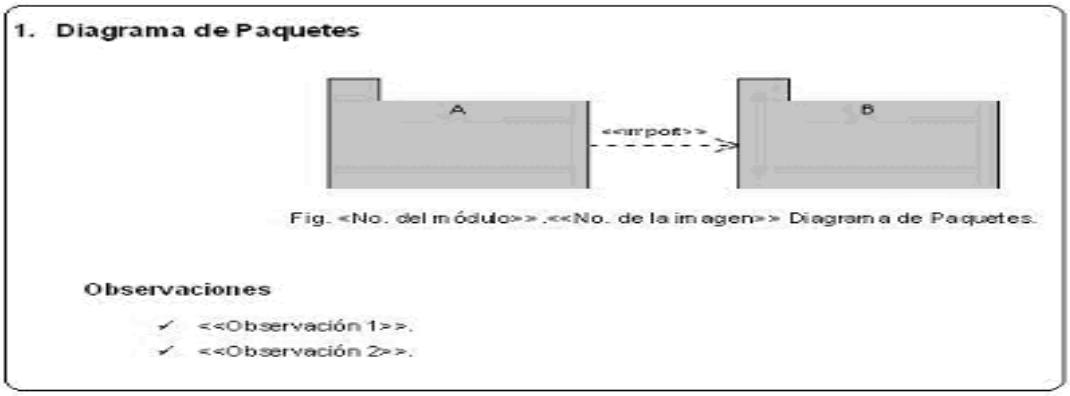


Figura 10. Modelo de diseño.

3.1.3 Desarrollo.

La fase de Desarrollo, es la segunda que define la metodología MA-GMPR-UR2., utilizada por los proyectos productivos de la Facultad 10, del Polo de Software Libre. En la primera parte de esta fase se generan todos los documentos relacionados con la planificación de las iteraciones, así como de las tareas a realizar durante la implementación. Además se genera el código fuente en la etapa de implementación, y luego los documentos relacionados con las pruebas, como última parte de esta etapa.

Plantilla de Tarea de ingeniería.

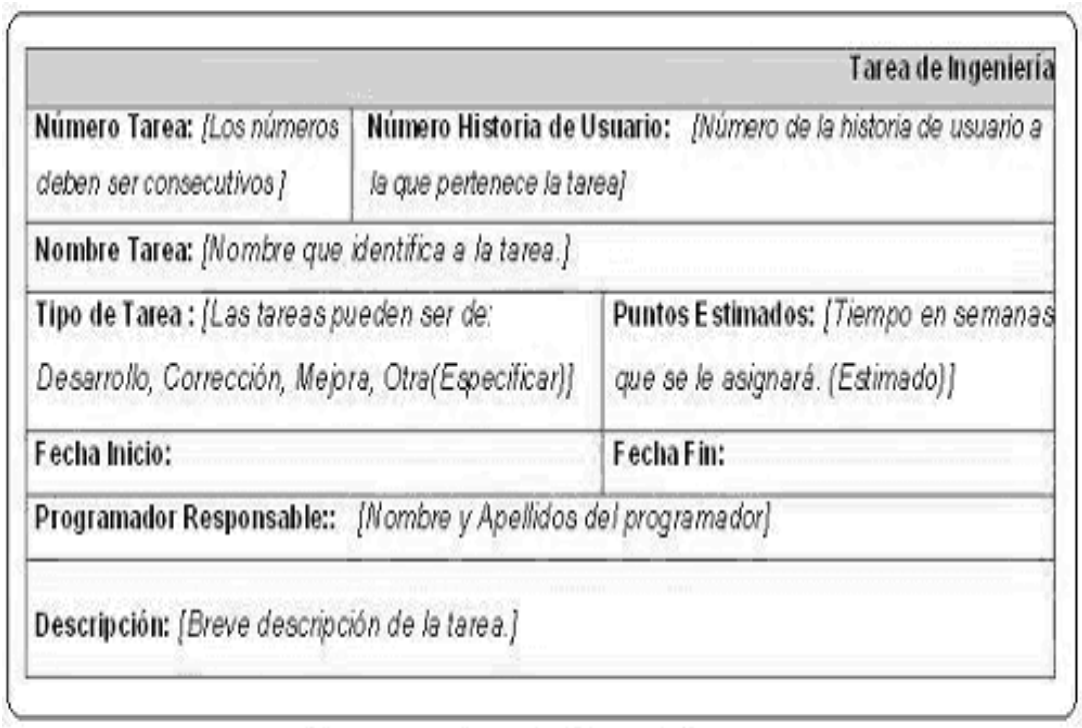
La plantilla de Tarea de ingeniería, es el primer artefacto de la fase de Desarrollo, la cual se genera de la actividad de Junta de planificación. Esta plantilla posee una gran importancia, pues permite definir cada una de las actividades que estarán asociadas a las historias de usuario y que permitirán su implementación. También posibilita conocer que programador está asignado a cada tarea, así como el tiempo que se necesita para su realización, lo que facilita la estimación del tiempo que se llevará cada historia de usuario en implementarse, de acuerdo a su complejidad.

Esta plantilla proporciona ventajas tales como:

- Permite organizar el proceso de implementación, pues las tareas se van implementando de acuerdo a su prioridad.
- Posibilita conocer el grado de complejidad de cada historia de usuario, teniendo en cuenta la cantidad de tareas asociadas.

Roles encargados de la plantilla: Programador.

Para comprender cuales son los datos que se recogerán en esta plantilla, se muestra la misma gráficamente.



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.]*	

Figura 11. Tarea de ingeniería.

Plantilla de Cronograma de producción.

La plantilla de Cronograma de producción, es un documento que se genera de la Junta de planificación. Este cronograma es de suma importancia, pues guía y controla las actividades que se realizan en cada iteración. En este, no sólo se recogen las actividades planificadas, sino el tiempo que durarán y el rol responsable de desarrollarlas.

Con la existencia del Cronograma de producción en la documentación de un software, se garantizan ventajas, tales como:

- Control y organización de cada una de las actividades a desarrollar por el equipo de trabajo.
- Planificación eficaz de las iteraciones, garantizando una buena producción.

Roles encargados de la plantilla: Gerente.

Para comprender cuales son los datos que se recogerán en esta plantilla, se muestra la misma gráficamente.

[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.

Figura 12. Cronograma de producción.

Plantilla de Plan de releases

La plantilla de Plan de releases, es un documento que se genera de la Junta de planificación. En esta plantilla el cliente define el valor que posee el negocio según las características deseadas, es decir las historias de usuario. Los programadores proporcionan estimaciones de 1, 2 ó 3 puntos, y las historias mayores de los 3 puntos deben dividirse en otras más pequeñas.

Es el cliente el que decide qué historias de usuario deben ser incluidas en un lanzamiento. Las historias de mayor riesgo y mayor prioridad se incluyen en las primeras iteraciones. El lanzamiento y las iteraciones tienen fechas fijas para su consecución, así como un alcance variable. Esta plantilla proporciona ventajas, tales como:

- Define cuales son las historias de usuario más significativas, y las ubican en las iteraciones según esta prioridad.
- Divide el proceso de desarrollo de software en iteraciones, planificando el trabajo a realizar en cada una de ellas.

Roles encargados de la plantilla: Gerente.

Para comprender cuales son los datos que se recogerán en esta plantilla, se muestra la misma gráficamente.

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}*

Figura 13. Plan de releases.

Estándares de programación.

El documento Estándares de programación, se genera de la etapa de implementación. En él se 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. La presencia de este documento en la documentación de un producto, proporciona ventajas, tales como:

- Facilita una posible reutilización del código, conociendo el estándar utilizado para su programación.
- Posibilita una mejor descripción de los elementos fundamentales del proceso implementación.

Roles encargados de la plantilla: Programador.

Plantilla de Caso de prueba de aceptación.

La plantilla de Caso de prueba de aceptación, se genera de la etapa de pruebas. El objetivo

de las pruebas de aceptación es validar que un sistema cumple con el funcionamiento esperado y permitir al usuario de dicho sistema que determine su aceptación, desde el punto de vista de su funcionalidad y rendimiento.

Las pruebas de aceptación son definidas por el cliente y preparadas por el equipo de desarrollo, aunque la ejecución y aprobación final corresponden al cliente. La utilización de estas proporcionan ventajas, tales como:

- Son el termómetro de los desarrolladores, fundamentalmente de los programadores a la hora de medir la calidad de su trabajo.
- Garantizan la entrega de un producto con calidad, que responde a las necesidades del cliente.

Roles encargados de la plantilla: Programador y Tester.

Para comprender cuales son los datos que se recogerán en esta plantilla, se muestra la misma gráficamente.

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.]*	

Figura 14. Caso de prueba de aceptación.

3.1.4 Entrega.

La fase de Entrega, es la tercera que define la metodología MA-GMPR-UR2., utilizada por los proyectos productivos de la Facultad 10, del Polo de Software Libre. En esta fase se

realiza la entrega del software y su documentación, generándose aquellos documentos que son imprescindibles para el entrenamiento y entendimiento del producto.

Manual de usuario.

El manual de usuario es un documento, que se genera en la etapa de entrenamiento, sirve de apoyo para la actividad de capacitación. En este se recogen todos los aspectos significativos, que contribuyan a que los usuarios que interactúen con el sistema lo comprendan mejor. Para su confección hay que tener en cuenta a quién va dirigido, es decir el manual puede ser manejado por diferentes usuarios, desde los más simples hasta los más complejos. Por consiguiente, debe redactarse de forma clara y sencilla para que lo entienda cualquier tipo de usuario.

El manual expone los procesos que el usuario puede realizar con el sistema implantado. Para lograr esto, es necesario que se detallen todas y cada una de las características que tienen los programas y la forma de acceder e introducir información. También permite a los usuarios conocer el detalle de qué actividades ellos deberán desarrollar para la consecución de los objetivos del sistema. Además reúne la información, normas y documentación necesaria para que el usuario conozca y utilice adecuadamente la aplicación desarrollada.

De manera general, el manual de usuario proporciona ventajas tales como:

- Posibilita que el usuario conozca como interactuar con el sistema.
- Se utiliza como manual de aprendizaje.
- Sirve como manual de referencia.

Roles encargados de la plantilla: Programador.

Manual de identidad.

El Manual de identidad, es un documento generado en la etapa de entrenamiento, y es definido como una herramienta estratégica para proyecto, en el que se trazan las líneas maestras de la imagen corporativa con las que el público identificará el producto. Se entiende como imagen corporativa la personalidad del software en todo aquello que lo representa. Dicha imagen debe estar presente en los documentos u objetos relacionados con el producto para ayudar al posicionamiento de éste en el mercado.

De esta forma el manual describe los signos gráficos escogidos por los diseñadores del

equipo de trabajo, apoyados por los demás miembros, para identificar el producto, y darle una imagen en el mercado; así como todas sus variantes o estilos según el medio de reproducción final u objetivo.

Su desarrollo es un proceso laborioso en el que se deben tener en cuenta diversos factores como: los signos gráficos que se usan o se han usado en dicho ámbito, los de otros productos de la misma rama de desarrollo, así como los objetivos a conseguir. Los beneficios que conlleva su desarrollo, a grandes rasgos son:

- Aumento de la notoriedad y prestigio del producto.
- Se crea una imagen que permite la identificación del producto en el mercado.
- Si se realiza un buen diseño, puede llamar la atención de muchos clientes.

En el expediente no se propone una plantilla para la realización de este manual, solo se muestra (ver anexo 5), un ejemplo de los aspectos que incluye este documento.

Roles encargados de la plantilla: Consultor especializado en diseño gráfico.

Manual de desarrollo.

El Manual de desarrollo, es un documento generado en la etapa de entrenamiento, el mismo recoge toda la información respecto a la implementación y el código fuente del producto, lo que posibilita una reutilización del mismo, sin que sea necesaria la presencia de sus desarrolladores.

Es importante describir en este manual el proceso de implementación de forma detallada, incluyendo que es lo que se logra con cada línea de código, esto facilitará el trabajo al realizar mejoras futuras al software. Poseer un manual de desarrollo en la documentación de un software, proporciona ventajas, tales como:

- Permite la reutilización del código de manera muy sencilla.
- Describe de forma precisa el proceso de implementación.

Roles encargados de la plantilla: Programador.

3.1.5 Mantenimiento.

La fase de Mantenimiento, es la cuarta y última que define la metodología MA-GMPR-UR2., utilizada por los proyectos productivos de la Facultad 10, del Polo de Software Libre. En esta fase se realizan las actividades relacionadas con el soporte del software y se generan los documentos relacionados con los cambios que puedan ocurrir en el mismo.

Plantilla de Gestión de cambios.

La plantilla de gestión de cambio, es el documento que se genera en la fase de Mantenimiento, en la actividad de soporte. En ella se recogen los cambios que se realicen en esta etapa, que pueden ser de corrección, adaptación o mejora. Además se lleva el control de la persona que realiza el cambio, así como de la que lo revisa.

Esta plantilla es muy importante, pues es la que controla cualquier mejora que se le realice al producto, además proporciona ventajas, tales como:

- Gestiona los cambios realizados al software, permitiendo conocer detalles de los mismos.
- Controla la persona encargada de cada cambio, así como su revisor.

Roles encargados de la plantilla: Programador.

Para comprender cuales son los datos que se recogerán en esta plantilla, se muestra la misma gráficamente.

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.]*	

Figura 15. Gestión de cambios.

3.1.6 Legal.

El expediente de proyecto propuesto incluye un área legal, que no fue definida por las autoras de la investigación, sino por las personas que atienden la parte jurídica en la universidad. En ésta área se recogen todos aquellos documentos que son necesarios para la entrega de los productos, y su comercialización en el mercado.

Es importante realizar una aclaración sobre este tema, pues las autoras de la investigación consideran que no es necesario realizar trámites jurídicos a todos los proyectos que se desarrollen en un grupo de trabajo, pues hay muchos de estos que sólo sirven de apoyo a otros proyectos, a los que se les conoce como proyectos experimentales. Por lo que será responsabilidad del Scrum Master decidir cuales son los equipos de trabajo que llenarán las plantilla, propuestas en esta área, fundamentalmente aquellos que serán comercializados.

3.2 Resultados prácticos.

Es importante resaltar que a pesar de que el nuevo expediente de proyecto propuesto en esta investigación, aún no ha sido aplicado íntegramente en los proyectos productivos de la Facultad 10, del Polo de Software Libre, se han obtenido resultados con la aplicación de muchas de las ideas que se tuvieron en cuenta para su realización.

Con la utilización de la metodología MA-GMPR-UR2 en cada uno de los proyectos, nació la necesidad de buscar una nueva forma de documentar los productos que fueron desarrollándose, de manera que respondieran al proceso de desarrollo de las metodologías ágiles. Es así como se comenzaron a utilizar muchas de las plantillas que han sido incluidas en este expediente, para que el proceso de documentación de software fluyera en los proyectos, tales como:

- Concepción del sistema, que posibilitó resumir los aspectos más relevantes del sistema, así como realizar una concepción inicial.
- Historia de usuario, que posibilitó ratificar que en ella se resumía todo lo referente a las especificaciones de los requerimientos, así como lo relacionado con los casos de uso, conocidos del proceso unificado.
- Prueba de aceptación, que resumió los aspectos significativos para garantizar la validación de un producto.
- Gestión de cambio, que permitió la recopilación de cada una de las mejoras realizadas a los productos.

Con la utilización de estas plantillas se alcanzaron resultados significativos, como son:

- La entrega en tiempo y forma de la documentación mínima y necesaria, de la mayoría de los proyectos que trabajan con la metodología MA-GMPR-UR2.
- Un proceso de documentación menos tedioso, y con resultados relevantes.
- Una gestión más sencilla y eficiente por todos los niveles de dirección de proyectos.

Precisamente estos resultados se convirtieron en el incentivo para el desarrollo de esta investigación, en busca de un enriquecimiento del proceso de recopilación de información en los proyectos, teniendo como premisa que con una mejor organización de la documentación, así como su adaptabilidad a la metodología utilizada, se obtendrán resultados superiores a los antes expuestos.

3.3 Valoración.

Con esta propuesta se pretende dar solución a muchos de los problemas que afectan el proceso de documentación en los proyectos productivos, como son las documentaciones extensas, la duplicación de esfuerzos, así como el tiempo que se dedica a documentar, que en la mayoría de los casos es mayor que el dedicado a la producción.

La aplicación del nuevo expediente de proyecto garantizará que los miembros del equipo no se sientan desanimados al documentar, ni consideren una tarea tan importante en el ciclo de vida del software, como un estorbo. Además con menos esfuerzo se alcanzarán mejores resultados y productos que contengan los documentos necesarios para una posible reutilización, así como para la comprensión de los mismos.

Por otra parte al repartir la responsabilidad de llenado de las plantillas que conforman al expediente, teniendo en cuenta los roles definidos por la metodología utilizada en los proyectos, se garantizará que no existan plantillas cuyo rol responsable, no sea miembro del equipo de trabajo, lo que facilita las acciones del líder de proyecto, el cual no necesitará asignar a ninguna otra persona desconocedora de los aspectos que recoge la plantilla, para desarrollarla.

Sin duda alguna si este expediente de proyecto, se aplica en todos los proyectos de la Facultad 10, del Polo de Software Libre, llegando incluso en un futuro a mejorarlo, de acuerdo a las particularidades que surjan, se logrará alcanzar un equilibrio en cuanto a la producción y a la documentación de software. Convirtiéndose el proceso de recopilación de

información en el mejor amigo de los desarrolladores.

Conclusiones.

En este capitulo, teniendo en cuenta la metodología de desarrollo utilizada en los proyectos productivos de la Facultad 10, del Polo de Software Libre, que en este caso utilizan XP para el desarrollo, y Scrum para la gestión, se modeló la propuesta de expediente de proyecto, que responde a las características de cada uno de estos proyectos. Se definieron además los roles propuestos por las metodologías y que serán los encargados de llenar cada una de las plantillas.

La estructura del nuevo expediente, facilitará el proceso de documentación, por parte de los miembros del equipo de desarrollo, pues se generan pocos documentos por cada una de las actividades que se realizan en las fases, todos de formato sencillo y fácil de llenar, evitando crear documentaciones largas y tediosas, tanto para hacerlas, como para leerlas.

CONCLUSIONES

En esta investigación, se modeló una propuesta (gdfgdfg, 1236) de expediente de proyecto, para los proyectos productivos de la Facultad 10, del Polo de Software Libre, dando cumplimiento al objetivo propuesto.

- Con el estudio de los elementos más significativos de los expedientes de proyecto, así como del proceso de documentación, se demostró que no existe un estándar al documentar y que este proceso se desarrolla de forma particular en cada empresa u organización.
- Al utilizar la metodología de desarrollo XP y para la gestión de proyectos Scrum, se facilitó la adaptabilidad del expediente de proyecto a las metodologías ágiles.
- Con la utilización del modelo de calidad CMMI, se posibilitó el aseguramiento de la calidad del expediente.
- Con la caracterización de los proyectos productivos del Polo de Software Libre, se facilitó el conocimiento de sus peculiaridades, así como las necesidades de los mismos en cuanto a la documentación.
- Al valorar el expediente de proyecto utilizado en la UCI, se identificaron sus deficiencias, las que fueron eliminadas en el nuevo expediente.
- Con la puesta en práctica de todos los elementos anteriormente analizados, y la experiencia adquirida en los grupos de trabajo, se modeló una propuesta de expediente de proyecto, acorde con las necesidades de los proyectos productivos, del Polo de Software Libre.

Esta propuesta de expediente de proyecto constituye una base para el desarrollo del proceso de documentación, en los proyectos productivos, garantizando una eficiente recopilación de información.

RECOMENDACIONES

- Validación del expediente de proyecto, a partir de la fase de entrega en cada uno de los proyectos productivos, de la Facultad 10, que trabajen con metodologías ágiles.
- Aplicación de la propuesta en todos los proyectos de la Facultad 10, del Polo de Software Libre.
- Utilización de trabajo de diploma como fuente bibliográfica para investigaciones referentes al tema.

REFERENCIAS BIBLIOGRÁFICAS

Beck, K. 2000. *Extreme Programming Explained.* s.l. : Pearson Education, 2000.

Cockbun, A. 2000. *¿The Costs and Benefits of Pair Programming? Humans and Technology Technical Report?* 2000.

Gallo, E. 2006. Metodologías ágiles. [En línea] 2006. [Citado el: 12 de mayo de 2008.] http://www.esi.es/Berrikuntza.

García, P. 2003. *Calidad en el desarrollo y mantenimiento del software.* Madrid : RA-MA Editorial, 2003.

González, S. 2006. Metodologías de proyectos informáticos. [En línea] 2006. [Citado el: 28 de marzo de 2008.] http://bip.mideplan.cl/bipconsultas/SEBI/2006/metodologias/metodologiainformatica.pdf.

Gutiérrez, J. 2007. *Metodologías ágiles.* s.l. : Universidad Pablo de Olavide, 2007.

Letelier, P. 2004. *Metodologías ágiles para el desarrollo del software: eXtreme Programming (XP).* s.l. : Universidad Politécnica de Valencia, 2004.

López, C. 2001. Las normas ISO 9000. [En línea] 2001. [Citado el: 26 de marzo de 2008.] http://www.gestiopolis.com/canales/gerencial/articulos/27/ISO.htm.

Menéndez, R. 2005. *Metodologías de desarrollo del software.* s.l. : Barzanallana Asensio, 2005.

Nuñez, Joaquín. 2006. Como documentar proyectos. [En línea] abril de 2006. [Citado el: 18 de Marzo de 2008.] http://kodegeek.com/2006/04/como-documentarproyectos-wiki-l.shtml.

Pulido, J. 2004. Estándares de calidad. [En línea] 2004. [Citado el: 10 de marzo de 2008.] www.udenar.edu.co/viceacademica/acre_files/REGISTRO%20CALIFICADO/EST%.

Rodríguez, M. 2007. *Introducción de procedimientos ágiles en la producción de software en la Facultad 7 de la Universidad de las Ciencias Informáticas.* Ciudad de La Habana : s.n., 2007.

Ronquillo, U. 2008. ¿Que es un modelo de calidad? [En línea] 2008. [Citado el: 10 de marzo de 2008.] http://hazloxl.wordpress.com/2008/01/08/que-es-un-modelo-de-calidad/.

Ruz, Fidel Castro. 2002. *Discurso de Fidel Castro.* Ciudad de la Habana : Oficina de Publicaciones del Consejo de Estado, 2002. 959-7030-78-0.

Otero, L. 2008. nº33, s.l. : Nure Investigación, 2008.

Villegas, A. 1997. Ciclo de vida de XP. [En línea] 1997. [Citado el: 17 de marzo de 2008.] http://www.monografias.com/trabajos51/ProgramacionExtrema/programacion-extrema2.shtml#ciclo.

BIBLIOGRAFÍA

Dowson, R. 2007. CMM o ISO 9000: Cuál es correcto para su organización. [En línea] 2007. [Consultada el: 5 de febrero del 2008] http://www.e-articles.info/t/i/1262/l/es/

García, J. 2005. CMM-CMMI [En línea] 2005. [Consultada el: 25 de marzo del 2008] http://www.ingenierosoftware.com/calidad/cmm-cmmi.php

Gutiérrez, J. **2007**. *Metodologías Ágiles*. Universidad Pablo de Olavide. Estados Unidos. C4Media, 2007. ISBN: 978-1-4303-2264-1.

Hernández, R. **2002**. *El paradigma cuantitativo de la investigación científica*. Ciudad de la Habana. EDUNIV. ISBN: 959-16-0343-6.

Kniberg, H. 2007. Scrum y XP desde las trincheras. [En línea] 2007. [Consultada el: 26 de marzo del 2008] /www.proyectalis.com/2008/02/26/scrum-y-xp-desde-las-trincheras]

Mario, E. 2007. Scrum en pocas palabras**.** [En línea] 2007. [Consultada el: 1 de abril del 2008] http://scrumenespanol.blogspot.com/]

Menéndez, R. 2007. Ingeniería del software. Metodologías de desarrollo. [En línea] 2007. [Consultada el: 5 de febrero del 2008] http://www.um.es/docencia/barzana/IAGP/IAGP2-Metodologias-de-desarrollo.html

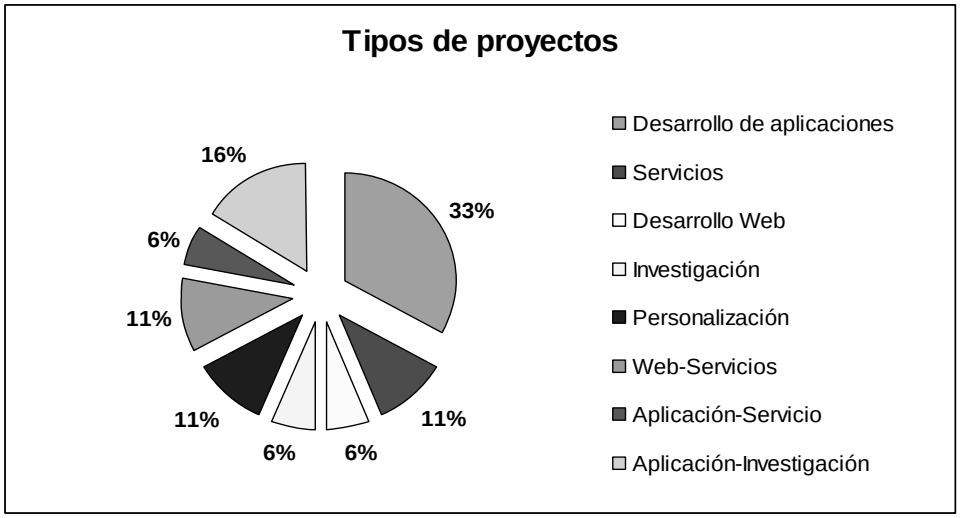
Sánchez, C. 2004. ONess: un proyecto open source para el negocio textil mayorista desarrollado con tecnologías open source innovadoras. [En línea] 2004. [Consultada el: 15 de febrero del 2008] http://oness.sourceforge.net/proyecto/html/index.html

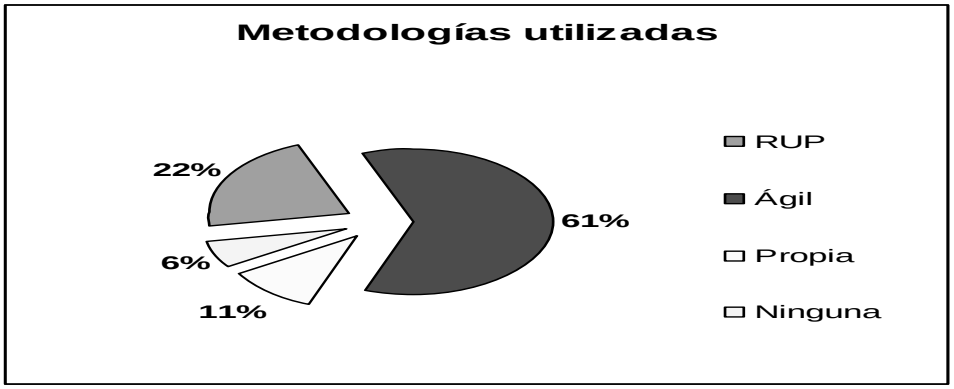
ANEXOS

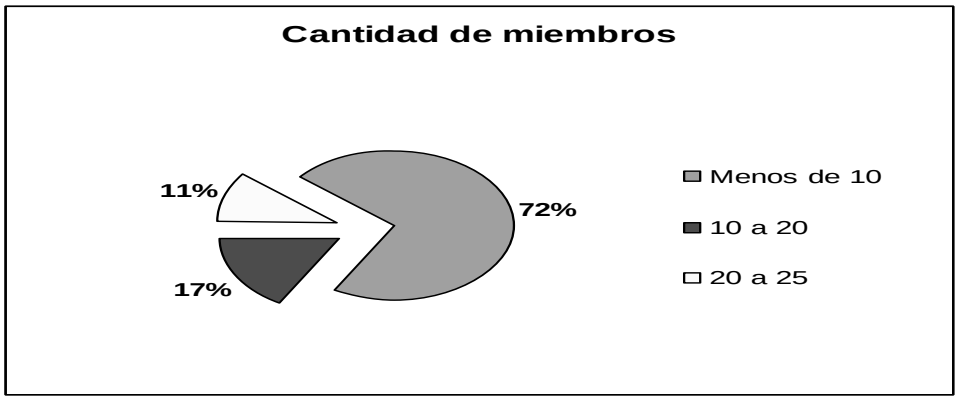
Anexo 1. Guión de la entrevista.

1. Teniendo en cuenta los tipos de proyectos existentes, ¿Dónde ubicaría su proyecto?
2. ¿Cuántas personas integran el equipo de desarrollo?
3. ¿Cuáles son los roles existentes en el proyecto?
4. ¿Con qué entorno de desarrollo trabaja el proyecto?
5. ¿Cómo caracterizaría usted al proyecto en cuanto a envergadura, tiempo de desarrollo y complejidad, teniendo en cuenta una escala de niveles, muy alta, alta, media y baja.?
6. ¿Cómo se organiza el proyecto en cuanto a la planificación, en equipos de desarrollo o módulos?
7. ¿Qué metodología de desarrollo de software utilizan en el proyecto para guiar el proceso?
8. ¿Cuáles son los principales problemas que se han presentado a lo largo del desarrollo del proyecto?
9. ¿Cuáles son los principales problemas que se han presentado a lo largo del proceso de recopilación de información en los expedientes de proyecto?

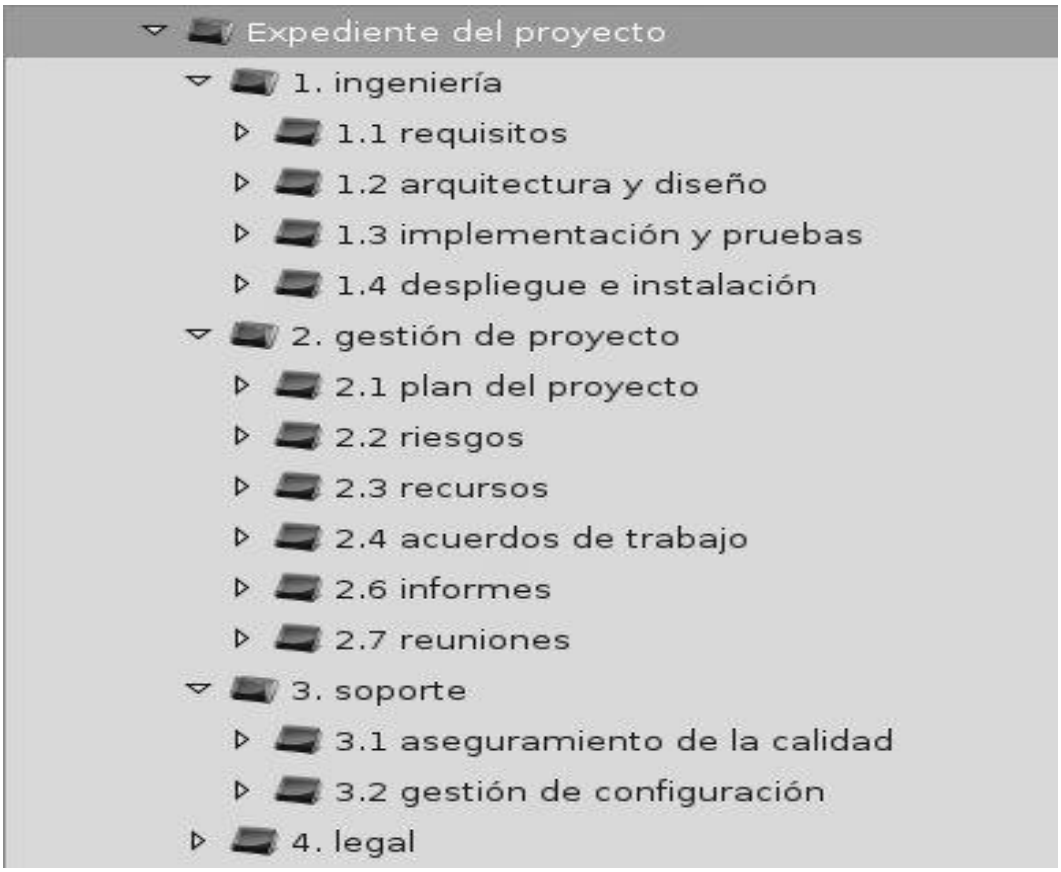
Anexo 2. Gráficos de análisis de los proyectos.







Anexo 3. Expediente de proyecto de la UCI.



Anexo 4. Guión de la metodología.

GUIÓN DE LA METODOLOGÍA

Planificación ↔ Definición

- Entrevista con el cliente (concepción inicial)
- Juego de la planificación.
- Captura de requisitos:
  - Creación de la LRP.
  - Priorización de la LRP.
  - Definir las historias de usuario.
  - Asignar las historias de usuario.
- Valoración del esfuerzo.
- Valoración de riesgos.
- Diseño con las metáforas.
- Refactorización.
- Reunión de revisión del diseño.

Desarrollo

- Junta de planificación.
  - Definir las historias de usuario a implementar.
  - Tareas para lograr dicha implementación.
- Implementación.
- Junta de seguimiento
- Taller técnico
- Junta de revisión.
- Pruebas

Entrega

- Entrega de la documentación.
- Entrenamiento.
  - Capacitación
  - Instalación.
- Marketing

Mantenimiento

- Soporte.

Anexo 5. Guía para Manual de identidad.

Guía para un Manual de identidad

Introducción

- ✔ Objetivo social del software.
- ✔ Perspectivas futuras.
- ✔ Historial anterior al software.

Capítulo 1: *Discurso de identidad.*

- ✔ Atributos y rasgos de estilo.

Capítulo 2: *El identificador Institucional.*

- ✔ Identificador
- ✔ Articulación
- ✔ El logotipo
- ✔ El isotipo.
- ✔ Construcción y distribución del identificador.
- ✔ Variantes.
- ✔ Variantes en blanco y negro y a escala de grises.
- ✔ Reducciones.
- ✔ Restricciones.
- ✔ Tipografía.
- ✔ Colores.

GLOSARIO DE TÉRMINOS

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

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.

Cliente: Persona, organización o grupo de personas que encargan la construcción de un producto software.

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 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.

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

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.

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: secuencia de actividades que tienen un marcado inicio y fin.

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.

Pruebas de aceptación: son las pruebas realizadas por el cliente para validar el software.

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

Sprint: Equivale a una iteración, en la metodología Scrum.

Usuario: Persona encargada de utilizar el sistema, obteniendo algún beneficio.

Validación: no es más que verificar que un producto determinado cumple con los requisitos que fueron pactados con el cliente.