Universidad de Cienfuegos “Carlos Rafael Rodríguez”
Facultad de Informática
Carrera de Ingeniería Informática

“SELECCIÓN DE METODOLOGÍAS DE DESARROLLO PARA APLICACIONES WEB EN LA FACULTAD DE INFORMÁTICA DE LA UNIVERSIDAD DE CIENFUEGOS”.

Autor: Karenny Brito Acuña

Cienfuegos, Cuba
2009

Resumen

El presente trabajo tiene como título “Selección de una metodología de desarrollo para Aplicaciones Web en la facultad de Informática de la Universidad de Cienfuegos”, el centro de esta investigación lo constituyen las metodologías de desarrollo de software.

El software es uno de los productos de la ingeniería que más ha evolucionado en muy poco tiempo, y es un hecho que los productos de software deben ser desarrollados con base en la implantación de estándares mundiales, modelos, sistemas métricos, capacitación del recurso humano y otros principios y técnicas de la ingeniería software que garanticen la producción de sistemas de calidad y competitividad a nivel local e internacional.

Actualmente en la Universidad de Cienfuegos se desarrollan sistemas enfocados a la web que por sus características podrían construirse siguiendo otra metodología que proporcione mejores resultados.

Teniendo en cuenta el estudio realizado sobre las metodologías de desarrollo de software existentes en la actualidad; y las características y situación de los proyectos que se llevan a cabo en la Universidad de Cienfuegos, se realiza la selección de las metodologías, que se ha validado en la construcción del Sistema de Gestión de Información del Control Interno Informático (SGICII), lográndose con ello disminuir el tiempo de desarrollo del mismo y el costo de producción.

Índice

Índice de Tablas

Índice de Figuras

Introducción

El software es uno de los productos de la ingeniería que más ha evolucionado en muy poco tiempo, transitando por el software empírico o artesanal hasta llegar al software desarrollado bajo los principios y herramientas de la ingeniería del software.

Sin embargo, dentro de estos cambios, las personas encargadas de la construcción de software se han enfrentado a problemas muy comunes: unos debido a la exigencia cada vez mayor en la capacidad de resultados del software por el permanente cambio de condiciones, lo que aumenta su complejidad y obsolescencia; y otros, debido a la carencia de las herramientas adecuadas y estándares de tipo organizacional encaminados al mejoramiento de los procesos en el desarrollo del software.

Una necesidad sentida en este medio es el hecho de que los productos de software deben ser desarrollados con base en la implantación de estándares mundiales, modelos , sistemas métricos, capacitación del recurso humano y otros principios y técnicas de la ingeniería software que garanticen la producción de software de calidad y competitividad a nivel local e internacional.

Con el acelerado avance tecnológico de la información, la cantidad y la complejidad de los productos de software se están incrementando considerablemente, así como la exigencia en su funcionalidad y confiabilidad; es por esto que la calidad y la productividad se están convirtiendo en las grandes preocupaciones tanto de gestores como para desarrolladores de software.

En los primeros años del software, las actividades de elaboración de programas eran realizadas por una sola persona utilizando lenguajes de bajo nivel y ajustándose a un computador en especial, que generaban programas difíciles de entender, aún hasta para su creador, luego de pasado algún tiempo de haberlo producido. Esto implicaba tener que repetir el proceso de desarrollo del mismo programa para otras máquinas. Por consiguiente, la confiabilidad, facilidad de mantenimiento y cumplimiento no se garantizaban y la productividad era muy baja.

Posteriormente, con la aparición de técnicas estructuradas y con base en las experiencias de los programadores se mejoró la productividad del software. Sin

embargo, los sistemas seguían teniendo fallas, como por ejemplo: documentación inadecuada, dificultad para su correcto funcionamiento, y por supuesto, insatisfacción del cliente.

Conforme se incrementaba la tecnología de los computadores, también crecía la demanda de los productos de software, pero lentamente, tanto que hacia 1990 se decía que las posibilidades de software estaban retrasadas respecto a las del hardware en un mínimo de dos generaciones de procesadores y que la distancia continuaba aumentando.

En la actualidad muchos de estos problemas subsisten en el desarrollo de software, con una dificultad adicional relacionada con la incapacidad para satisfacer totalmente la gran demanda y exigencias por parte de los clientes. En el trabajo titulado “Metodología para evaluar la calidad de la etapa de análisis de proyectos informáticos orientado a objetos (CAOOSI)” de Lourdes García Ávila [1] plantea que las metodologías guían el proceso de desarrollo y la experiencia ha demostrado que la clave del éxito de un proyecto de software es la elección correcta de la metodología, que puede conducir al programador a desarrollar un buen sistema de software. La elección de la metodología adecuada es más importante que utilizar las mejores y más potentes herramientas.

Muchos ingenieros ante la tarea de investigar y estudiar las metodologías se encuentran en la disyuntiva de que por cuál inclinarse y optan, inclusive, por obviar este importante elemento en la elaboración de sus sistemas. Como resultado de esto la documentación que apoya los productos de software se limita a ser solamente la indispensable y cuando se van a realizar procesos de mantenimiento a la aplicación resulta engorroso trabajar con estos documentos debido a que no han seguido una metodología de trabajo y es prácticamente imposible saber detalladamente los pasos que se sucedieron en la realización del proyecto.

En la Facultad de Informática de la Universidad de Cienfuegos se desarrollan habitualmente sistemas, no solo para favorecer el desempeño del centro docente, sino a otras empresas del territorio que necesitan automatizar procesos que en ellas se realizan. Para trabajar en la realización de estos software se crean grupos de investigación que encaminan su trabajo en líneas determinadas y desarrollan

sistemas mayormente orientados a la web, y la mayoría de los trabajos de diploma también se inclinan por esta vía. Sin embargo, en el momento de seleccionar la metodología a aplicar normalmente se inclinan por elegir la que se estudia en la carrera sin tener en cuenta que quizás exista otra metodología que por las características de grupo y del proyecto cumpla mejor con los requerimientos de este.

Es por ello que el **problema de esta investigación** queda definido como la necesidad de estandarizar una metodología ya existente para desarrollar aplicaciones web en la Facultad de Informática de la Universidad de Cienfuegos.

Se considera como el **objeto de estudio** de la presente investigación el proceso de desarrollo de software, de este modo se deriva como **campo de acción** el desarrollo de aplicaciones web en la Facultad de Informática de la Universidad de Cienfuegos

Para resolver el problema que se refleja en la situación anterior se plantea como **objetivo general**:

- Proponer el uso de una metodología ya existente para el desarrollo de aplicaciones web en la Facultad de Informática de la Universidad de Cienfuegos

De este objetivo general se desprenden los siguientes **objetivos específicos**:

- Realizar un estudio de las metodologías de desarrollo de software.
- Seleccionar dentro de las metodologías existentes las que mejor se adaptan al desarrollo de software en la Universidad de Cienfuegos.
- Validar la propuesta en un ejemplo práctico.

Para lograr el cumplimiento exitoso de estos objetivos se llevarán a cabo las siguientes **tareas**:

- Ejecución de una búsqueda de información acerca de las metodologías de desarrollo de software.

- Conceptualización de las metodologías de desarrollo.
- Determinación de las metodologías de desarrollo para aplicaciones web.
- Describir las metodologías de desarrollo de software en general y para aplicaciones web.
- Aplicación de las metodologías propuestas a un problema real vinculado directamente a la producción.

Teniendo en cuenta la información anterior se define la siguiente **idea a defender** la realización de esta investigación permitirá contar con metodologías de desarrollo que se adapten a las características de los sistemas desarrollados en la Universidad de Cienfuegos y a los equipos de trabajo, que permitan la realización exitosa de sistemas de alta calidad y que además disminuirá el tiempo de entrega de los mismos.

Esta investigación **aporta teóricamente** la propuesta de metodologías de desarrollo para aplicaciones web que posibilitará la realización de sistemas de calidad que cumplan con las expectativas del cliente.

El desarrollo de este trabajo está estructurado por los siguientes capítulos:

Capítulo 1: En este capítulo se hace una introducción a la ingeniería de software. Se hace un estudio de los trabajos que anteriormente se han realizado sobre este tema y además se realiza una breve descripción de las metodologías que existen para el desarrollo de software.

Capítulo 2: En este capítulo se tratan los conceptos asociados al tema de la investigación. Asimismo se seleccionan las metodologías para el desarrollo de aplicaciones web y se describen detalladamente las mismas.

Capítulo 3: En este capítulo se valida la elección de las metodologías con la aplicación a un software que se encuentra en producción actualmente. Además se especifican las herramientas que apoyan la aplicación de las metodologías y se describen los resultados obtenidos.

Capítulo I: Marco teórico de la investigación

En el desarrollo de este capítulo se hace un análisis de los trabajos que se han realizado anteriormente relacionados con las metodologías de desarrollo de software y sus características. Se describen además las metodologías existentes en la actualidad para el desarrollo de software.

I.1 Ingeniería de Software

En la construcción y desarrollo de proyectos se aplican métodos y técnicas para resolver los problemas, la informática aporta herramientas y procedimientos sobre los que se apoya la ingeniería de software con el fin de mejorar la calidad de los productos de software, aumentar la productividad y trabajo de los ingenieros del software, facilitar el control del proceso de desarrollo de software y suministrar a los desarrolladores las bases para construir software de alta calidad en una forma eficiente.[2]

El objetivo es convertir el desarrollo de software en un proceso formal, con resultados predecibles, que permitan obtener un producto final de alta calidad y satisfaga las necesidades y expectativas del cliente.

La Ingeniería de Software [3] es un proceso intensivo de conocimiento, que abarca la captura de requerimientos, diseño, desarrollo, prueba, implantación y mantenimiento. Generalmente a partir de un complejo esquema de comunicación en el que interactúan usuarios y desarrolladores, el usuario brinda una concepción de la funcionalidad esperada y el desarrollador especifica esta funcionalidad a partir de esta primera concepción mediante aproximaciones sucesivas. Este ambiente de interacción motiva la búsqueda de estrategias robustas para garantizar que los requisitos del usuario serán descubiertos con precisión y que además serán expresados en una forma correcta y sin ambigüedad, que sea verificable, trazable y modificable.

El término ingeniería del software [4] empezó a usarse a finales de la década de los sesenta, para expresar el área de conocimiento que se estaba desarrollando en torno a las problemáticas que ofrecía el software. En esa época, el crecimiento espectacular de la demanda de sistemas de computación cada vez más y más

complejos, asociado a la inmadurez del propio sector informático (totalmente ligado al electrónico) y a la falta de métodos y recursos, provocó lo que se llamó la crisis del software. Durante esa época muchos proyectos importantes superaban con creces los presupuestos y fechas estimados. La crisis del software finalizó pues se comenzó a progresar en los procesos de diseño y metodologías. Así pues, desde 1985 hasta el presente, han ido apareciendo herramientas, metodologías y tecnologías que se presentaban como la solución definitiva al problema de la planificación, previsión de costes y aseguramiento de la calidad en el desarrollo de software [5]. Cada año surgen nuevas ideas e iniciativas encaminadas a ello. En combinación con las herramientas, también se han hecho esfuerzos por incorporar los métodos formales al desarrollo de software, argumentando que si se probaba formalmente que los desarrolladores hacían lo que se les requería, la industria del software sería tan predecible como lo son otras ramas de la ingeniería. La dificultad propia de los nuevos sistemas, y su impacto en el negocio, han puesto de manifiesto las ventajas, y en muchos casos la necesidad, de aplicar una metodología formal para llevar a cabo los proyectos de este tipo.

I.2. Investigaciones realizadas en el campo de las metodologías de desarrollo de software.

Una parte importante de la ingeniería de software es el desarrollo de metodologías. Es por ello que la investigación acerca de las mismas, y por consiguiente su utilización, está tomando auge en el mundo de la ingeniería de software de hoy. Han sido muchos los esfuerzos que se han encaminado al estudio de los métodos y técnicas para lograr una aplicación más eficiente de las metodologías y tener en correspondencia sistemas de mayor calidad con la documentación necesaria en perfecto orden y en el tiempo requerido.

I.2.1 Las metodologías de desarrollo en el contexto mundial

Internacionalmente se realizan numerosos estudios sobre las metodologías de desarrollo de software que han sido reflejados en artículos científicos, a continuación se hace un análisis de algunos de ellos.

En el artículo “Métodos de desarrollo de software: el desafío pendiente de la estandarización” [6] se plantea que una metodología impone un proceso disciplinado sobre el desarrollo de software con el objetivo de hacerlo más predecible y eficiente, por tanto, define un camino reproducible para obtener resultados confiables. Es considerado por el autor que una metodología define una representación que permite facilitar la manipulación de modelos, y la comunicación e intercambio de información entre todas las partes involucradas en la construcción de un sistema.

En el artículo “Balanceo de Metodologías Orientadas al Plan y Ágiles Herramientas para la Selección y Adaptación” [7] se plantea que la experiencia ha demostrado que los proyectos exitosos son aquellos que son administrados siguiendo una serie de procesos que permiten organizar y luego controlar el proyecto, considerando válido destacar que aquellos procesos que no sigan estos lineamientos corren un alto riesgo de fracasar. Además se coincide con el autor de este artículo cuando expresa que el profesional se ve envuelto en una discusión en la que muchos plantean sus opiniones como verdades indiscutibles, y en la que la decisión de qué metodología utilizar, parece ser un acto de fe antes que una evaluación de alternativas técnicas, con sus costos, beneficios y riesgos asociados. Es necesario destacar que los métodos son importantes, pero no se debe perder de vista que el éxito del proyecto depende más de la comunicación efectiva con los interesados, el manejo de las expectativas, el valor generado para el negocio y las personas que participan en el proyecto.

En el artículo “La incertidumbre y la ingeniería de software” [8] plantea que la mayor crítica que han tenido las metodologías surgidas alrededor de 1960 es que son burocráticas, es decir, se debe invertir una cantidad significativa del tiempo del

proyecto en perseguir objetivos paralelos impuestos, lo cual hace que se reduzca el tiempo efectivo para cumplir con el objetivo real: construir *software*. No obstante, hay quienes afirman que el fracaso del llamado "edificio metodológico de la ingeniería de *software* temprana" se debe a que se apoya en una premisa alejada de la realidad del universo, especialmente del universo *software*, donde se supuso que la incertidumbre era erradicable, siendo inevitable. [9]

En el artículo "Metodologías Ágiles en el Desarrollo de Software" [10] se presenta resumidamente el contexto en el que surgen las metodologías ágiles, según el cual el término ágil comienza a emplearse para caracterizar metodologías luego de una reunión celebrada en Utah-EEUU en febrero de 2001, se dice que fue en esa reunión donde un grupo de 17 expertos esbozaron los valores y principios que deberían permitir a los equipos desarrollar software rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto. Pero según otro artículo publicado por Ailin Orjuela Duarte y Mauricio Rojas titulado "Las Metodologías de Desarrollo Ágil como una Oportunidad para la Ingeniería de Software Educativo" [11] fue realizada otra reunión en marzo de 2001, la cual fue convocada por Kent Beck donde se acuñó el término "Métodos Ágiles" para definir a los métodos que estaban surgiendo como alternativa a las metodologías formales a las que se consideraba "pesada" y rígida por su carácter normativo y fuerte dependencia de planificaciones detalladas previas al desarrollo. Los integrantes de esta reunión resumieron los principios sobre los que se basan los métodos alternativos en cuatro postulados, lo que ha quedado denominado como Manifiesto Ágil. Plantea además este artículo que las metodologías de desarrollo ágil son más orientadas a procesos de pocas semanas y con bajos niveles de formalización en la documentación requerida, es decir, proyectos pequeños.

En el artículo, "Del Manifiesto Ágil sus Valores y Principios" [5] se presenta de manera general la evolución de las metodologías para el desarrollo de software, haciéndose una breve reseña histórica del surgimiento de la computación, expresa además que las metodologías ágiles resuelven los problemas surgidos

posteriormente a la masificación del uso del computador personal dado que las expectativas y necesidades se hicieron más frecuentes; pero la autora del presente trabajo opina que las metodologías ágiles resuelven el problema surgido en la aplicación de las metodologías tradicionales, que traen aparejadas una demora considerable teniendo en cuenta la documentación que genera y lo engorroso de su aplicación, pero son, sin dudas, una buena opción para aquellos proyectos complejos y de gran tamaño, logrando ser efectivas donde por lo general se exige un alto grado de ceremonia en el proceso, garantizando además tener la documentación necesaria para tal magnitud. Para los proyectos pequeños es conveniente aplicar una metodología ágil que agiliza el proceso, genera solo los documentos necesarios y por consiguiente disminuye el tiempo de realización del mismo.

Plantea que con el surgimiento de las metodologías ágiles el concepto de etapa se desvanece dando paso a la idea de actividades. Estas pueden ser organizadas a comodidad del equipo de trabajo, en paquetes pequeños conservando las mismas labores e identidad de las etapas concebidas en las metodologías tradicionales.

En el articulo “Metodologías tradicionales vs. Metodologías ágiles” [12] se detallan los dos grandes enfoques, las primeras están pensadas para el uso exhaustivo de documentación durante todo el ciclo del proyecto mientras que las segundas ponen vital importancia en la capacidad de respuesta a los cambios, la confianza en las habilidades del equipo y al mantener una buena relación con el cliente. Plantea además que las metodologías ágiles se deberían aplicar en proyectos donde exista mucha incertidumbre, el entorno es volátil y los requisitos no se conocen con exactitud, mientras que las metodologías tradicionales obligan al cliente a tomar las decisiones al inicio del proyecto. Vale la pena acotar que estas metodologías están orientadas especialmente a proyectos pequeños, para los que constituyen una solución a la medida, aportando una elevada simplificación, que a pesar de ello no renuncia a las prácticas esenciales para asegurar la calidad del producto.

Este artículo ofrece una guía dejando libertad de elección para el lector el poder juzgar y elegir la mejor metodología que se adapte a su equipo de desarrollo.

En el artículo “Metodologías ágiles: la ventaja competitiva de estar preparado para tomar decisiones los más tarde posible y cambiarlas en cualquier momento” [13] se plantea que la adopción de metodologías ágiles supone una ventaja competitiva, tanto para el cliente como para la empresa de desarrollo, en proyectos donde exista mucha incertidumbre. Para las empresas de servicios profesionales las metodologías ágiles ofrecen una manera razonable de gestionar la incertidumbre en la estimación de los proyectos permitiendo brindar el mejor servicio posible al cliente. Además las metodologías tradicionales no están preparadas para el cambio por lo que no ofrecen una buena solución cuando el proyecto se realiza en un entorno volátil como la mayoría de los entornos en la actualidad. Además obligan al cliente a tomar decisiones al principio que en futuro pueden variar, lo que sería costoso cuando aplicamos estas metodologías pues mientras más tarde sea el cambio mayor parte del proceso se deberá realizar nuevamente para responder a la necesidad del cliente de cambiar un decisión antes tomada.

En el artículo “Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP)” [14] se plantea que las características de los proyectos para los cuales las metodologías ágiles han sido especialmente pensadas se ajustan a un amplio rango de proyectos industriales de desarrollo de software; aquellos en los cuales los equipos de desarrollo son pequeños, con plazos reducidos, requisitos volátiles, y/o basados en nuevas tecnologías. Se considera por tanto que son metodologías recomendables para proyectos que se desarrollan en entornos muy cambiantes donde los requerimientos del cliente pueden variar desde el comienzo del mismo hasta casi el final del software.

En el artículo “Procesos Ágiles en el Desarrollo de Aplicaciones Web” [15] se considera que el modelo de proceso más adecuado para el desarrollo de software es un proceso iterativo e incremental, puesto que a diferencia de otros modelos de

proceso, como por ejemplo el modelo en cascada, permite la obtención de diversas versiones del producto software antes de la entrega final del mismo y la depuración y validación progresiva del mismo, lo que sin duda redundará en un software más satisfactorio para usuarios y cliente. Además y según indica Conallen [16], con este tipo de proceso es posible añadir o modificar requisitos que no han sido detectados con anterioridad.

Se han realizado investigaciones de las metodologías de desarrollo de software también con el objetivo de contribuir en el perfeccionamiento los sistemas de enseñanza en el mundo, llevándose a cabo estudios relacionados con las metodologías a aplicar en un proyecto determinado, que tributan en muchos casos a la asignatura de Ingeniería de software. Un ejemplo es el artículo “Modelo para la exposición de la materia de ingeniería de software I” [17] donde se expone que debido al crecimiento exponencial de la demanda de software son necesarias técnicas y tecnología eficientes de Ingeniería de Software para resolver los múltiples problemas que se derivan de las aplicaciones en donde se desarrollan sistemas. Es válido destacar que la Ingeniería de Software tiene como principal objetivo servir como base para la producción de software de calidad, lo cual se logra definiendo el proceso del software, que comprende las actividades involucradas en la producción del software. Expresa también el trabajo que cuando utilizan las metodologías lo que se obtiene son clientes satisfechos con el resultado, y desarrolladores aún más satisfechos. Se considera sin embargo, que muchas veces no se toma en cuenta el utilizar una metodología adecuada, sobre todo cuando se trata de proyectos pequeños de dos o tres meses de duración.

I.2.2 Las metodologías de desarrollo en el contexto nacional

Cuba es un país donde el desarrollo de software es aún incipiente. Sin embargo la investigación acerca de herramientas que automaticen y agilicen el proceso de creación de sistemas es un tema al cual los ingenieros y desarrolladores en general están prestando más atención. Las metodologías de desarrollo de software como parte importante de la construcción de un sistema han sido objeto

de numerosos estudios que ayudan a conocer sus potencialidades. Los resultados de estas investigaciones son expuestos en artículos científicos, algunos de estos trabajos se detallan a continuación

El artículo “Metodologías de desarrollo de software. ¿Cuál es el camino?” [18] plantea que las metodologías imponen un proceso disciplinado sobre el desarrollo de software con el fin de hacerlo más predecible y eficiente. Lo hacen desarrollando un proceso detallado con un fuerte énfasis en planificar, inspirado por otras disciplinas de la ingeniería. Se considera que los principales problemas detectados en las metodologías ingenieriles son los retrasos en la planificación, o sea, llegada la fecha de entregar el software, éste no esta disponible; los sistemas deteriorados, el software se ha creado, pero después de un par de años el coste de su mantenimiento es tan complicado que definitivamente se abandona su producción; el software se pone en producción, pero los defectos son tantos que nadie lo usa; el software no resuelve los requisitos planificados inicialmente; el problema que resolvía el software ha cambiado y este no se ha adaptado; el software hace muchas cosas técnicamente muy interesantes y divertidas, pero no resuelven el problema del cliente; después de unos años de trabajo los programadores comienzan a odiar el proyecto y lo abandonan.

En el artículo “Experiencias de la aplicación de la ingeniería de software en sistemas de gestión” [1] se plantea que las metodologías aparecen por la necesidad de poner orden al proceso de construcción del software y que resulta importante que el desarrollo de sistemas informáticos sea tratado bajo una disciplina ingenieril, con el fin de desarrollar e implantar sistemas realmente eficaces y eficientes, pero los realizadores de software no hacen uso de metodologías, herramientas CASE y criterios de calidad. Según la opinión de la autora de este trabajo muchas son las personas que cometen el error de no aplicar metodologías en el desarrollo de sus sistemas, apoyados en el criterio de que no es necesario, lo que conlleva a que la documentación del software no queda detallada, o en muchas ocasiones, se realiza en último momento trayendo consigo una demora en la entrega del proyecto.

I.2.3 Las metodologías de desarrollo en el contexto local

En la universidad de Cienfuegos se llevan a cabo igualmente investigaciones en muchos campos de las ciencias informáticas, lo que sin dudas incluye el estudio de las metodologías de desarrollo de software, que han tomado auge en los últimos tiempos. A continuación se muestran algunos de los artículos resultantes de estas investigaciones.

En el artículo Metodologías de Desarrollo de Software [19] se hace un análisis de los modelos de desarrollo de software, se plantea que las metodologías basadas en el proceso en espiral progresan en el desarrollo de software mediante capas; cada capa o espiral representa una fase en el proceso. En este modelo no existen fases prefijadas (captura de requisitos, análisis, diseño, etc.): en un proyecto habrá tantas capas como se necesiten.

El proceso en espiral se introdujo para solucionar los problemas del proceso en cascada, y es la variante de éste más usada en la actualidad. La secuencia de pasos es aún lineal, pero admite retroalimentación. El modelo en espiral considera que los pasos hacia atrás (las iteraciones posteriores a la primera) son errores. El modelo de desarrollo iterativo asume, en cambio, que siempre se van a cometer errores y que, por consiguiente, siempre habrá que efectuar varias iteraciones. Un proyecto basado en este último modelo se construye mediante iteraciones.

I.3. Descripción de las metodologías existentes para el desarrollo de software

Tener metodologías diferentes para aplicar de acuerdo con el proyecto que se desarrolle resulta imprescindible teniendo en cuenta las necesidades cambiantes que tiene el entorno de desarrollo actual y el acelerado progreso de la informática a nivel mundial resulta una idea interesante. Estas metodologías pueden involucrar prácticas tanto de metodologías ágiles como de metodologías tradicionales. A continuación se describen las características de algunas de ellas.

Proceso Unificado de Desarrollo (RUP)

El Proceso Unificado de Desarrollo [20] fue creado por el mismo grupo de expertos que crearon *UML*, Ivar Jacobson, Grady Booch y James Rumbaugh en el año 1998. El objetivo que se perseguía con esta metodología era producir software de alta calidad, es decir, que cumpla con los requerimientos de los usuarios dentro de una planificación y presupuesto establecidos. Como se expresaba anteriormente, esta metodología concibió desde sus inicios el uso de *UML* como lenguaje de modelado.

Es un proceso dirigido por casos de uso, este avanza a través de una serie de flujos de trabajo (requisitos, análisis, diseño, implementación, prueba) que parten de los casos de uso; está centrado en la arquitectura y es iterativo e incremental. Además cubre el ciclo de vida de desarrollo de un proyecto y toma en cuenta las mejores prácticas a utilizar en el modelo de desarrollo de software.

A continuación se muestran estas prácticas.

- Desarrollo de software en forma iterativa.
- Manejo de requerimientos.
- Utiliza arquitectura basada en componentes.
- Modela el software visualmente.
- Verifica la calidad del software.
- Controla los cambios.

Para apoyar el trabajo con esta metodología ha sido desarrollada por la Compañía norteamericana Rational Corporation la herramienta CASE (Computer Assisted Software Engineering) Rational Rose en el año 2000. Esta herramienta integra todos los elementos que propone la metodología para cubrir el ciclo de vida de un proyecto.

Microsoft Solution Framework (MSF)

MSF es una metodología desarrollada por Microsoft Consulting Services en conjunto con varios grupos de negocios de Microsoft y otras fuentes de la industria. MSF provee los principios, modelos y disciplinas para un correcto desarrollo de proyectos en cualquier plataforma (Linux, Citrix, Microsoft, Unix).

Esta es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la

gestión de proyectos tecnológicos. MSF se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas.

MSF tiene las siguientes características [21]:

• Adaptable: es parecido a un compás, usado en cualquier parte como un mapa, del cual su uso es limitado a un específico lugar.

• Escalable: puede organizar equipos tan pequeños entre 3 o 4 personas, así como también, proyectos que requieren 50 personas a más.

• Flexible: es utilizada en el ambiente de desarrollo de cualquier cliente.

• Tecnología Agnóstica: porque puede ser usada para desarrollar soluciones basadas sobre cualquier tecnología.

MSF se compone de varios modelos encargados de planificar las diferentes partes implicadas en el desarrollo de un proyecto: Modelo de Arquitectura del Proyecto, Modelo de Equipo, Modelo de Proceso, Modelo de Gestión del Riesgo, Modelo de Diseño de Proceso y finalmente el modelo de Aplicación.

Aplicando esta metodología todo proyecto es separado en cinco principales fases [22]:

- Visión y Alcances.
- Planificación.
- Desarrollo.
- Estabilización.
- Implantación.

Visión y Alcances: trata uno de los requisitos fundamentales para el éxito del proyecto, la unificación del equipo detrás de una visión común. El equipo debe tener una visión clara de lo que quisiera lograr para el cliente y ser capaz de indicarlo en términos que motivarán a todo el equipo y al cliente. Se definen los líderes y responsables del proyecto, adicionalmente se identifican las metas y objetivos a alcanzar; estas últimas se deben respetar durante la ejecución del proyecto en su totalidad, y se realiza la evaluación inicial de riesgos del proyecto.

Planificación: Es en esta fase es cuando la mayor parte de la planeación para el proyecto es terminada. El equipo prepara las especificaciones funcionales, realiza

el proceso de diseño de la solución, y prepara los planes de trabajo, estimaciones de costos y cronogramas de los diferentes entregables del proyecto.

Desarrollo: Durante esta fase el equipo realiza la mayor parte de la construcción de los componentes (tanto documentación como código), sin embargo, se puede realizar algún trabajo de desarrollo durante la etapa de estabilización en respuesta a los resultados de las pruebas. La infraestructura también es desarrollada durante esta fase.

Scrum

Scrum [23] es una metodología de desarrollo muy simple, que requiere trabajo duro, porque no se basa en el seguimiento de un plan, sino en la adaptación continua a las circunstancias de la evolución del proyecto.

Toma forma de las prácticas de trabajo que a partir de los 80 comienzan a adoptar algunas empresas tecnológicas y que Nonaka y Takeuchi acuñaron como "Campos de Scrum". El modelo Scrum, aplicado al desarrollo de software, emplea el principio de desarrollo ágil: "desarrollo iterativo e incremental", denominando sprint a cada iteración de desarrollo. Las prácticas empleadas por Scrum para mantener un control ágil en el proyecto son:

- Revisión de las iteraciones
- Desarrollo incremental
- Desarrollo evolutivo
- Auto-organización del equipo
- Colaboración

Define un marco para la gestión de proyecto. Está especialmente indicado a proyectos con un rápido cambio de requisitos. En Scrum inicialmente planean el contexto y un estimado amplio de la entrega. Luego desarrollan el estimado de la entrega basado en el desarrollo del ambiente del proyecto. El proceso del ciclo de vida de Scrum reconoce que el proceso de desarrollo fundamental está completamente indefinido y usa mecanismos de control para perfeccionar la flexibilidad, manipular lo imprescindible y el control del riesgo. [11]

Los valores que hacen posible a las prácticas de Scrum crear "campos de Scrum" son:

- Autonomía del equipo -**
- Respeto en el equipo
- Responsabilidad y auto-disciplina
- Foco en la tarea
- Información transparencia y visibilidad

Programación Extrema (Extreme Programming, XP)

XP [24] 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. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

Los principios y prácticas son de sentido común pero llevadas al extremo, de ahí proviene su nombre. Kent Beck, el padre de XP, describe la filosofía de XP en [10] sin cubrir los detalles técnicos y de implantación de las prácticas.

XP utiliza un técnica denominada Historias de Usuario es utilizada para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales.

Roles XP

Los roles de acuerdo con la propuesta original de Beck son: [24]

- Programador.
- Cliente.
- Encargado de pruebas (Tester).
- Encargado de seguimiento (Tracker).
- Entrenador (Coach).

- Consultor.
- Gestor (Big boss).

Proceso XP

El ciclo de desarrollo consiste (a grandes rasgos) en los siguientes pasos: [14]

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.

El ciclo de vida ideal de XP consiste de seis fases [10]: Exploración, Planificación de la Entrega (Release), Iteraciones, Producción, Mantenimiento y Muerte del Proyecto.

ICONIX

El proceso ICONIX [25] se define como un proceso de desarrollo de software práctico. Está entre la complejidad de RUP y la simplicidad y pragmatismo de XP, sin eliminar las tareas de análisis y diseño que XP no contempla.

Es un proceso simplificado en comparación con otros procesos más tradicionales, que unifica un conjunto de métodos de orientación a objetos con el objetivo de abarcar todo el ciclo de vida de un proyecto. ICONIX presenta claramente las actividades de cada fase y exhibe una secuencia de pasos que deben ser seguidos. Además, está adaptado a patrones y ofrece el soporte UML, dirigido por Casos de Uso y es un proceso iterativo e incremental.

Las tres características fundamentales de ICONIX son:

- Iterativo e incremental: varias interacciones ocurren entre el modelo del dominio y la identificación de los casos de uso. El modelo estático es incrementalmente refinado por los modelos dinámicos.

- Trazabilidad: cada paso está referenciado por algún requisito. Se define la trazabilidad como la capacidad de seguir una relación entre los diferentes artefactos producidos
- Dinámica del UML: la metodología ofrece un uso dinámico del UML como los diagramas del caso de uso, diagramas de secuencia y de colaboración.

Las tareas que se realizan en la metodología ICONIX son:

- Análisis de requisitos
- Análisis y diseño preliminar
- Diseño
- Implementación

Crystal Methodologies

Se trata de un conjunto de metodologías para el desarrollo de software caracterizadas por estar centradas en las personas que componen el equipo y la reducción al máximo del número de artefactos producidos. El desarrollo de software se considera un juego cooperativo de invención y comunicación, limitado por los recursos a utilizar. El equipo de desarrollo es un factor clave, por lo que se deben invertir esfuerzos en mejorar sus habilidades y destrezas, así como tener políticas de trabajo en equipo definidas. Estas políticas dependerán del tamaño del equipo, estableciéndose una clasificación por colores, por ejemplo Crystal Clear (3 a 8 miembros) y Crystal Orange (25 a 50 miembros). [24]

Crystal Clear

Alistair Cockburn es el propulsor detrás de la serie de metodologías Crystal. Las mismas presentan un enfoque ágil, con gran énfasis en la comunicación, y con cierta tolerancia que la hace ideal en los casos en que sea inaplicable la disciplina requerida por XP. Crystal “Clear” es la encarnación más ágil de la serie y de la que más documentación se dispone. La misma se define con mucho énfasis en la comunicación, y de forma muy liviana en relación a los entregables. Crystal maneja iteraciones cortas con feedback frecuente por parte de los usuarios/clientes, minimizando de esta forma la necesidad de productos intermedios. Otra de las cuestiones planteadas es la necesidad de disponer de un

usuario real aunque sea de forma part time para realizar validaciones sobre la Interfase del Usuario y para participar en la definición de los requerimientos funcionales y no funcionales del software.

Una cuestión interesante que surge del análisis de la serie Crystal es el pragmatismo con que se customiza el proceso. Las personas involucradas escogen aquellos principios que les resultan efectivos y mediante la aplicación de la metodología en diversos proyectos agregan o remueven principios en base al consenso grupal del equipo de desarrollo.

Los siete valores o propiedades de Crystal Clear son: [26]

1. Entrega frecuente. Consiste en entregar software a los clientes con frecuencia, no solamente en compilar el código. La frecuencia dependerá del proyecto, pero puede ser diaria, semanal o mensual.
2. Comunicación osmótica. Todos juntos en el mismo cuarto. Una variante especial es disponer en la sala de un diseñador senior; eso se llama Experto al Alcance de la Oreja. Una reunión separada para que los concurrentes se concentren mejor es descrita como El Cono del Silencio.
3. Mejora reflexiva. Tomarse un pequeño tiempo (unas pocas horas por algunas semanas o una vez al mes) para pensar bien qué se está haciendo, cotejar notas, reflexionar, discutir.
4. Seguridad personal. Hablar cuando algo molesta: decirle amigablemente al manager que la agenda no es realista, o a un colega que su código necesita mejorarse, o que sería conveniente que se bañase más seguido.
5. Foco. Saber lo que se está haciendo y tener la tranquilidad y el tiempo para hacerlo. Lo primero debe venir de la comunicación sobre dirección y prioridades, típicamente con el Patrocinador Ejecutivo. Lo segundo, de un ambiente en que la gente no se vea compelida a hacer otras cosas incompatibles.
6. Fácil acceso a usuarios expertos.
7. Ambiente técnico con prueba automatizada, management de configuración e integración frecuente. Muchos equipos ágiles compilan e integran varias veces al día.

FDD

FDD [27] es un proceso diseñado por Peter Coad, Erich Lefebvre y Jeff De Luca y se podría considerar a medio camino entre RUP y XP, aunque al seguir siendo un proceso ligero (en mi opinión, si tenemos que distinguir entre pesado/ligero) es más similar a este último. FDD está pensado para proyectos con tiempo de desarrollo relativamente cortos (menos de un año). Se basa en un proceso iterativo con iteraciones cortas (~2 semanas) que producen un software funcional que el cliente y la dirección de la empresa pueden ver y monitorizar.

Las iteraciones se deciden en base a *features* (de ahí el nombre del proceso) o funcionalidades, que son pequeñas partes del software con significado para el cliente. Así, construir el sistema de ventas es algo que requiere mucho tiempo, y construir el sistema de persistencia no tiene significado para el cliente, pero si lo tiene enviar pedido por e-mail.

Un proyecto que sigue FDD se divide en 5 fases:

1. Desarrollo de un modelo general
2. Construcción de la lista de funcionalidades
3. Plan de releases en base a las funcionalidades a implementar
4. Diseñar en base a las funcionalidades
5. Implementar en base a las funcionalidades

3. Fases de FDD

4. Vista general de FDD

Las primeras tres fases ocupan gran parte del tiempo en las primeras iteraciones, siendo las dos últimas las que absorben la mayor parte del tiempo según va avanzando el proyecto, limitándose las primeras a un proceso de refinamiento.

El trabajo (tanto de modelado como de desarrollo) se realiza en grupo, aunque siempre habrá un responsable último (arquitecto jefe o jefe de programadores en función de la fase en que se encuentre), con mayor experiencia, que tendrá la última palabra en caso de no llegar a un acuerdo. Al hacerlo en grupo se consigue que todos formen parte del proyecto y que los menos inexpertos aprendan de las

discusiones de los más experimentados, y al tener un responsable último, se asignan las responsabilidades que todas las empresas exigen.

Las funcionalidades a implementar en una release se dividen entre los distintos subgrupos del equipo, y se procede a implementarlas. Las clases escritas tienen *propietario* (es decir, solo quién las crea puede cambiarlas), es por ello que en el equipo que implementa una funcionalidad dada deberán estar todos los *dueños* de las clases implicadas, pudiendo encontrarse un programador en varios grupos, implementando distintas funcionalidades. Habrá también un *programador jefe* (normalmente el más experimentado) que hará las funciones de líder del grupo que implementa esa funcionalidad.

En el proceso de implementar la funcionalidad también se contemplan como partes del mismo (en otros métodos se describen como actividades independientes) la preparación y ejecución de pruebas, así como revisiones del código (para distribuir el conocimiento y aumentar la calidad) e integración de las partes que componen el software.

FDD también define métricas para seguir el proceso de desarrollo de la aplicación, útiles para el cliente y la dirección de la empresa, y que pueden ayudar, además de para conocer el estado actual del desarrollo, a realizar mejores estimaciones en proyectos futuros.

Adaptive Software Development (ASD)

Este proceso consiste en un cambio de filosofía en las organizaciones pasando de la transición del modelo Comando-Control al modelo liderazgo-Colaboración. Lleva los conceptos de los Sistemas Adaptativos Complejos al campo de la Ingeniería de Software en particular. Dada la complejidad inherente al software concluye que la aplicación de esta teoría es esencial para el nuevo escenario que plantea la economía global.

El ciclo de vida de ASD propone tres fases esenciales: especulación, colaboración y aprendizaje. El proyecto comienza con una fase de especulación en que se lleva a cabo la planificación tentativa del proyecto en función de las entregas que se irán realizando. En esta etapa se fija un rumbo determinado a ser seguido en el

desarrollo, sabiendo a partir de ese momento que no será el lugar en que finalizará el proyecto. En cada iteración, se aprenderán nuevas funcionalidades, se entenderán viejas cuestiones, y cambiarán los requerimientos. [28]

La siguiente fase del ciclo de vida, Colaborar, es aquella en la que se construye la funcionalidad definida durante la especulación. ASD define un *Componente* como un grupo de funcionalidades o entregables a ser desarrollados durante un ciclo iterativo. Durante cada iteración el equipo colabora intensamente para liberar la funcionalidad planificada. También existe la posibilidad de explorar nuevas alternativas, realizar pruebas de concepto, pudiendo eventualmente alterar el rumbo del proyecto profundamente. ASD no propone técnicas ni prescribe tareas al momento de llevar a cabo la construcción simplemente mencionando que todas las prácticas que sirvan para reforzar la colaboración serán preferidas, siguiendo de esta forma la línea de las metodologías ágiles respecto a la orientación a componentes.

La fase final de ASD, Aprender, consiste en la revisión de calidad que se realiza al final de cada ciclo.

Para evaluar la calidad desde el punto de vista del cliente se sugieren utilizar grupos de enfoque en el cliente, mediante los cuales se explora un modelo de la aplicación y se anotan los requerimientos de cambio del cliente.

Las revisiones al diseño, al código o a las pruebas permitirán aprender sobre la calidad de los mismos. En este caso, el énfasis estará puesto en aprender cuales han sido los errores o desvíos y poder resolverlos, y no en encontrar culpables. Asimismo, esta es la etapa en que se evaluarán las exploraciones que se hayan realizado dando la capacidad de poder modificar la arquitectura del sistema si se ha encontrado algún camino que se ajusta mejor a lo que necesita el usuario o si han cambiado los requerimientos.

Finalmente se puede afirmar que ASD es un marco filosófico basado en la teoría de *Sistemas Adaptativos Complejos* que permite encarar la construcción de software en forma ágil utilizando las prácticas que resulten convenientes en cada caso. En este sentido resulta similar a Scrum.

Agile Unified Process (AUP)

Agile Unified Process [29] es una versión simplificada de Rational Unified Process, desarrollada por Scott Amber.

Divide el ciclo de desarrollo en 4 fases:

Incepción: identificación del alcance y dimensión del proyecto, propuesta de la arquitectura y de presupuesto del cliente.

Elaboración: Confirmación de la idoneidad de la arquitectura.

Construcción: Desarrollo incremental del sistema, siguiendo las prioridades funcionales de los implicados.

Transición: Validación e implantación del sistema.

Dynamic Systems Development Method (DSDM)

DSDM es otra metodología diseñada para responder a los plazos de entrega cortos y una cantidad limitada de recursos. Nace en 1994 con el objetivo de crear una metodología RAD (Desarrollo Rápido de Aplicaciones, en inglés, Rapid Application Development) unificada. Al igual que Cristal, DSDM se esfuerza por acortar las líneas de comunicación entre el cliente, promotor, y las empresas interesadas, a fin de prestar un servicio más eficiente en el proceso de producción de software. Al fijar el plazo de entrega (generalmente 6 meses) y el establecimiento de límites de los recursos, es más fácil establecer un proceso de desarrollo que responda a los usuarios "reales requerimientos del negocio." Según el sitio web del Consorcio DSDM, DSDM en un marco de desarrollo que "se centra en las prioridades del negocio y ofrece los entregados dentro de los plazos y costos del proyecto, en orden de prioridad determinado por las necesidades y los objetivos de la proyecto." [30]

Sus principales características son: es un proceso iterativo e incremental y el equipo de desarrollo y el usuario trabajan juntos.

El ciclo de desarrollo de DSDM está compuesto de 5 fases.

1. . Estudio de viabilidad
2. Estudio de negocio
3. Iteración de modelado funcional

4. Iteración de diseño y desarrollo
5. Implementación

Las tres últimas son iterativas, además de existir realimentación a todas las fases.

Xbreed

XBreed [31] es una mezcla de XP y Scrum ideas desarrolladas por Mike Beedle. SCRUM XBreed utiliza como marco de gestión, mientras que la adaptación de una versión reducida de Extreme Programming para su proceso de desarrollo. Fue diseñado con la intención de desarrollar "software reutilizables en un tiempo récord." Mediante el empleo de patrones de diseño para crear los objetos reutilizables, XBreed crea una biblioteca de componentes que, idealmente, son fácilmente reinsertados en nuevos proyectos de software. El proceso de desarrollo depende en gran medida de la capacidad y los conocimientos de los miembros de su equipo, que requieren fuertes y poca transferencia de conocimientos generales de comunicación para mantener el proyecto funcionando bien y eficientemente, los requisitos son ayudados por el uso de SCRUM.

En la actualidad está evolucionando y cambiando de nombre. Mantiene los principios de gestión de Scrum, y ahora se denomina AE (Agile Enterprise).

Lean Development

Lean [32] fue descrito por Mary Poppendieck, es una de las últimas contribuciones a la comunidad Ágil de Desarrollo de Software. Basado en los principios de producción ajustada japoneses, Lean es otro paradigma de desarrollo de naturaleza similar a la de Cristal y TEA en la medida en que fomenta un cambio en la cultura, así como un cambio en el proceso. Estos principios están diseñados para aumentar la velocidad de entrega, los productos de software de alta calidad y menor coste de producción.

Win-Win Spiral

Win-Win Spiral Model [33] propuesto por Barry Boehm es un nuevo logro en un proceso tradicional de software. Manteniendo al mismo tiempo muchos de los elementos tradicionales la versión Win-Win Spiral Model se esfuerza por comprender a todos los interesados en el proceso de desarrollo. Se trata de un motor de colaboración que establece que "ganar" las condiciones establecidas por los usuarios, clientes, desarrolladores, ingenieros de sistemas y con el fin de evolucionar y priorizar los requisitos durante todo el proceso. Las prácticas tradicionales, como los requisitos de ingeniería, diseño, código, y probar, aún están presentes en cada iteración de la espiral, pero el paso de colaboración durante el proceso de desarrollo hace que sea claramente de adaptación. Esta colaboración ofrece el software más rápido, con mayor calidad, menos costosa y, debido a la inicial de satisfacción de las necesidades de los usuarios y la cantidad reducida de mantenimiento.

Estabilización: En esta fase se conducen pruebas sobre la solución, las pruebas de esta etapa enfatizan el uso y operación bajo condiciones realistas. El equipo se enfoca en priorizar y resolver errores y preparar la solución para el lanzamiento.

Implantación: Durante esta fase el equipo implanta la tecnología base y los componentes relacionados, estabiliza la instalación, traspasa el proyecto al personal soporte y operaciones, y obtiene la aprobación final del cliente.

MIDAS

MIDAS [34] es una metodología genérica y fácilmente adaptable a cualquier tipo de SIW. Es un *framework* metodológico para el desarrollo de Sistemas de Información para Windows (SIW por sus siglas en inglés), en el que se propone un proceso de desarrollo ágil integrado en una *arquitectura dirigida por modelos*, alineándose así con la propuesta MDA del OMG.

MIDAS propone modelar un SIW atendiendo a dos dimensiones ortogonales. Por un lado, y debido a que la metodología se basa en una propuesta MDA, es necesario tener en cuenta el grado de dependencia de la plataforma de los modelos construidos. En primer lugar, será necesario modelar el sistema

construyendo modelos independientes de la computación (CIM), modelos independientes de la plataforma (PIM) y modelos específicos de la plataforma (PSM) y, en segundo lugar, será necesario especificar las reglas que permitan realizar transformaciones entre estos modelos. Por otro lado, la segunda dimensión a considerar está relacionada con tres aspectos básicos: el contenido (es decir, la información presente en el SIW), el hipertexto (relacionado con el modelado de las posibles rutas de navegación que podría seguir un usuario del SIW durante su interacción con el mismo) y el comportamiento propiamente dicho del sistema.

Además, MIDAS sugiere la utilización del Lenguaje de Modelado Unificado (UML) como única notación para modelar tanto los PIMs como los PSMs.

RMM

RMM [35] está basada en los conceptos implantados en el Modelo de diseño de hipertexto, es decir, en las entidades y en los tipos de entidades. Su objetivo es mejorar la navegación a través de un análisis de las entidades del sistema. En teoría, se obtiene una navegación más estructurada y logra que esta sea más intuitiva para el usuario. Los conceptos de slices y m-slices, que consisten en la agrupación de datos de una entidad en diferentes pantallas, es una de las aportaciones más importantes de esta metodología. Fue la primera metodología completa que se publica para software multimedia. Su problema principal es que no permite realizar consultas a partir de dos entidades y como está muy atado al modelo entidad relación cuando se define una relación se obliga a descomponerlas en dos relaciones copiando el modelo E-R. Además no considera las consultas a la base de datos para la creación de páginas Web dinámicas.

UWE

La propuesta de Ingeniería Web basada en UML es una metodología detallada para el proceso de autoría de aplicaciones con una definición exhaustiva del proceso de diseño que debe ser utilizado. Este proceso, iterativo e incremental,

incluye flujos de trabajo y puntos de control, y sus fases coinciden con las propuestas en el Proceso Unificado de Modelado.

UWE está especializada en la especificación de aplicaciones adaptativas, y por tanto hace especial hincapié en características de personalización, como es la definición de un modelo de usuario o una etapa de definición de características adaptativas de la navegación en función de las preferencias, conocimiento o tareas de usuario. [36]

Otras características relevantes del proceso y método de autoría de UWE son el uso del paradigma orientado a objetos, su orientación al usuario, la definición de un meta-modelo (modelo de referencia) que da soporte al método y el grado de formalismo que alcanza debido al soporte que proporciona para la definición de restricciones sobre los modelos.

OOHDM (The Object-Oriented Hypermedia Design Model)

El Modelo de Diseño de Hipermedia Orientado a Objeto [37], el sucesor del modelo de diseño hipertexto HDM, se trata de una metodología que se fundamenta en la orientación a objeto. Propone las siguientes fases: Diseño conceptual o análisis de dominio que utiliza el método de diseño orientado a objeto para obtener esquemas conceptuales de las clases y las relaciones entre las mismas. Utiliza las “técnicas de modelo de objeto” llamada notación OMT para el diseño de la navegación, donde se define la estructura de navegación por medio de modelos, es decir, a través de diferentes vistas del esquema conceptual; la fase de diseño de interfaz abstracta, se apoya en un modelo orientado a objeto para especificar la estructura y el comportamiento de la interfaz del sistema, este modelo se crea a través de tres tipos de diagramas: diagramas abstractos para cada clase, diagramas de configuración para reflejar los eventos externos y diagrama de estado para señalar el comportamiento dinámico; y por último, la fase de implementación, es decir, la construcción de los programas en programación orientada a objeto.

I.4 Conclusiones del capítulo

En este capítulo se realizó una introducción a la ingeniería de software y su historia. Además se llevó a cabo un estudio de las investigaciones que se han realizado en el campo de las metodologías de desarrollo no solo en Cuba sino también en el mundo. Se llevó a cabo igualmente un análisis de estas metodologías de desarrollo de software que son utilizadas en la actualidad por los desarrolladores de sistemas para lograr productos de alta calidad.

Capitulo II: Fundamentación de las metodologías de desarrollo de software seleccionadas

En este capítulo se exponen los conceptos principales que están relacionados con el tema de la investigación. Asimismo se proponen dos metodologías de desarrollo efectivas en la realización de aplicaciones web y además se describen detalladamente las características de las metodologías seleccionadas.

II.1- Principales conceptos asociados al dominio del problema

La experiencia acumulada a lo largo de los años de ejecutar proyectos indica que los proyecto exitosos son aquellos que son administrados siguiendo una serie de procesos que permiten organizar y luego controlar al proyecto.

Es sin dudas un objetivo fundamental de los diseñadores de software alcanzar y mantener un nivel técnico en los sistemas, acorde con el desarrollo actual en la automatización de la información para la gestión y la dirección, y es conocida la importancia de la ingeniería de software para elevar la productividad y la calidad en diseños de sistemas automatizados a partir de un examen detallado de las metodologías existentes.

La rama de la metodología, dentro de la ingeniería de software, se encarga de elaborar estrategias de desarrollo de software que promuevan prácticas adaptativas en vez de predictivas; centradas en las personas o los equipos, orientadas hacia la funcionalidad y la entrega, de comunicación intensiva y que requieren implicación directa del cliente.

Muchos investigadores en sus estudios acerca del tema de las metodologías de desarrollo de software se han detenido a definir el concepto de metodología, algunas de estas definiciones se muestran a continuación.

En el trabajo “Una Metodología para el desarrollo de Base de Datos” se dice que una metodología es un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de un producto de software.

Enrique Barreiro Alonso plantea que una metodología es una colección de métodos para la resolución de una clase de problemas (OMT, metodología de Booch, Catalysis, Proceso Unificado de Desarrollo,...).[38]

José Joaquín Cañadas define una metodología como un enfoque estructurado para el desarrollo de software cuyo propósito es facilitar la producción de software de alta calidad. Incluyen modelos de sistemas, notaciones, reglas sugerencias de diseño y guías de procesos. [39]

Una metodología define: [40]

- Estados, etapas o fases de un desarrollo, junto con los criterios de transición entre ellos.
- Tareas, actividades, etc.
- Roles, con sus skills necesarios y las interacciones entre ellos.
- Artefactos o entregables.
- Herramientas de control, seguimiento, medición y perfeccionamiento.
- Principios, criterios para tomar decisiones, estrategias para manejar distintos tipos de situaciones, herramientas de manejo de riesgos, etc.

La dificultad propia del desarrollo de software, y su impacto en el negocio, han puesto de manifiesto las ventajas, y en muchos casos la necesidad, de aplicar una metodología formal para llevar a cabo los proyectos de este tipo.
El objetivo es convertir el desarrollo de software en un proceso formal, con resultados predecibles, que permitan obtener un producto de alta calidad, que satisfaga las necesidades y expectativas del cliente. Atrás queda el modo de trabajar artesanal, que a menudo requiere grandes esfuerzos para obtener el resultado final, con los consecuentes desfases de fechas y coste, y es más que probable el desgaste personal del equipo de proyecto. Por lo que utilizar las metodologías implica mejoras en los procesos de desarrollo, en el producto y en la satisfacción del cliente.

Mejoras de los procesos de desarrollo

- Todos los integrantes del equipo del proyecto trabajan bajo un marco común.
- Estandarización de conceptos, actividades y nomenclatura.
- Actividades de desarrollo apoyadas por procedimientos y guías.
- Resultados de desarrollo predecibles.
- Uso de herramientas de ingeniería de software.
- Planificación de las actividades en base a un conjunto de tareas definidas y a la experiencia en otros proyectos.
- Recopilación de mejores prácticas para proyectos futuros.

Mejoras de los productos

- Se asegura que los productos cumplen con los objetivos de calidad propuestos.
- Detención temprana de errores.
- Se garantiza la trazabilidad de los productos a lo largo del proceso de desarrollo.

Mejoras en las relaciones con el cliente

- El cliente percibe el orden en los procesos.
- Facilita al cliente el seguimiento de evolución del proyecto.
- Se establecen mecanismos para asegurar que los productos desarrollados cumplan con las expectativas del cliente.

II.2- Tipos de metodologías

La comparación y clasificación de metodologías no es una tarea sencilla debido a la diversidad de propuestas y diferencias en el grado de detalle, la información disponible y alcance de cada una de ellas.

Si se toma como criterio las notaciones utilizadas para especificar artefactos producidos en actividades de análisis y diseño, se pueden clasificar las metodologías en dos grupos: Metodologías Estructuradas y Metodologías Orientadas a Objetos.

Por otra parte si se considera su filosofía de desarrollo se pueden diferenciar en metodologías tradicionales y ágiles.

II.2.1 Metodologías Estructuradas y Metodologías Orientadas a Objetos

Las metodologías estructuradas se basan en la estructuración y descomposición funcional de problemas en unidades más pequeñas interrelacionadas entre sí. Representan los procesos, flujos y estructuras de datos, de una manera jerárquica y ven el sistema como entradas-proceso-salidas.[41]

Existe una gran cantidad de proyectos implementados utilizando estas metodologías, generalmente orientados a la manipulación de datos (persistentes en ficheros o bases de datos) y gestión. Estas metodologías funcionan muy bien con los lenguajes de programación estructurados, como por ejemplo el COBOL.

Las metodologías estructuradas hacen fuerte separación entre los datos y los procesos. Producen una gran cantidad de modelos y documentación y se basan en ciclos de vida en cascada. [41]

El enfoque estructurado tiene un alto grado de especialización en sus perfiles y las relaciones entre ellos tienen fuertes raíces en los principios de la descomposición funcional. Su principio fundamental es examinar el sistema desde las funciones y tareas, mientras que en las metodologías orientadas a objetos modelan el sistema examinando el dominio del problema como un conjunto de objetos que interactúan entre sí.

Con la aparición del paradigma de la Orientación a Objetos surgieron métodos, procesos y metodologías específicas como OMT (Object Modeling Technique), Objectory, RUP o Métrica 3 (en su enfoque OO), entre otras.

Ante la necesidad de normalizar el proceso desarrollo se propusieron las metodologías en las que priman las fases, actividades y tareas antes que a las personas: lo más importante es el rol que juega la persona dentro del proceso de desarrollo. Esto se ha justificado históricamente argumentando que el desarrollo de software es una actividad compleja que debe estar dirigida por un proceso y debe llevarse a cabo en grupos.

Para cubrir todas las fases se necesitan distintos y diversos perfiles, que no se encuentran en una sola persona. El rol se asigna dependiendo de la experiencia, formación y capacidades de cada individuo.

Estos roles suelen ser muy similares en todos los procesos: administrador de proyecto, analista, diseñador, programador, probadores, asegurador de calidad, documentalista, ingeniero de manutención, ingeniero de validación y verificación, administrador de la configuración. Para cada uno de estos roles, se definen sus objetivos, actividades, interacción con otros roles, herramientas a utilizar, perfil de las personas en ese rol y un plan de trabajo.

Una de las ventajas de estos procesos es que el intercambio de un recurso es relativamente fácil: cuanto más identificadas se tengan las tareas que realiza un rol, más fácil es formar a una persona en esas tareas específicas. [41]

II.2.2 Metodologías tradicionales y metodologías ágiles

Teniendo en cuenta la filosofía de desarrollo de las metodologías, aquellas con mayor énfasis en la planificación y control del proyecto, en especificación precisa de requisitos y modelado, reciben el apelativo de Metodologías Tradicionales o Pesadas. [27]

Estas metodologías tradicionales imponen una disciplina de trabajo sobre el proceso de desarrollo del software, con el fin de conseguir un software más eficiente. Para ello, se hace énfasis en la planificación total de todo el trabajo a realizar y una vez que está todo detallado, comienza el ciclo de desarrollo del producto software. Se centran especialmente en el control del proceso, mediante una rigurosa definición de roles, actividades, artefactos, herramientas y notaciones para el modelado y documentación detallada [42]. Además, las metodologías tradicionales no se adaptan adecuadamente a los cambios, por lo que no son métodos adecuados cuando se trabaja en un entorno, donde los requisitos no pueden predecirse o bien pueden variar.

Entre las metodologías tradicionales o pesadas podemos citar:

- RUP (Rational Unified Procces)
- MSF (Microsoft Solution Framework)

- Win-Win Spiral Model
- Iconix

Pero sin dudas adaptarse a la agitada sociedad actual implica ser “ágil”, es decir, tener la capacidad de proveer respuestas rápidas y ser adaptables al cambio. Ambas cualidades siempre han sido deseables, pero en el entorno de negocio actual resultan indispensables. Este requerimiento de agilidad en las empresas, gobiernos y cualquier otra organización provoca que el software también deba ser desarrollado de manera ágil.

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

Los procesos ágiles son una buena elección cuando se trabaja con requisitos desconocidos o variables. Si no existen requisitos estables, no existe una gran posibilidad de tener un diseño estable y de seguir un proceso totalmente planificado, que no vaya a variar ni en tiempo ni en dinero. En estas situaciones, un proceso adaptativo será mucho más efectivo que un proceso predictivo. Por otra parte, los procesos de desarrollo adaptativos también facilitan la generación rápida de prototipos y de versiones previos a la entrega final, lo cual agradará al cliente.

Las metodologías ágiles proporcionan una serie de pautas y principios junto a técnicas pragmáticas que puede que no curen todos los males pero harán la entrega del proyecto menos complicada y más satisfactoria tanto para los clientes como para los equipos de entrega. En la figura 1 se muestran los principios que rigen el desarrollo ágil.

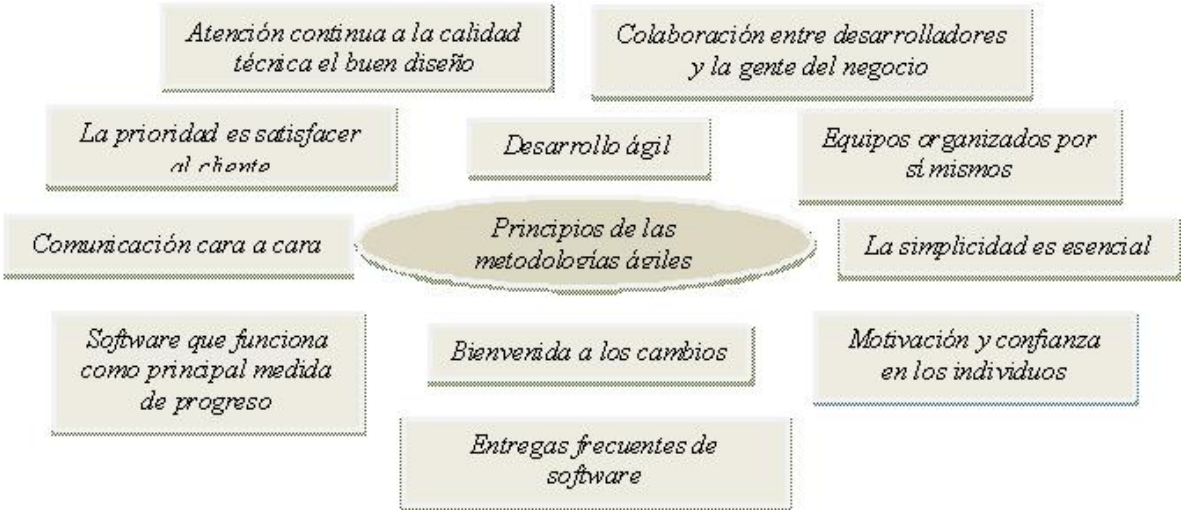


Figura 1: Principios del Manifiesto Ágil

Entre las metodologías ágiles más destacadas hasta el momento se pueden nombrar:

• XP (Extreme Programming)
• Scrum
• Crystal Clear
• DSDM (Dynamic Systems Developmemt Method)
• FDD (Feature Driven Development)
• ASD (Adaptive Software Development)
• XBreed
• Extreme Modeling

En el Anexo 1 se muestra una comparación entre las metodologías tradicionales y las ágiles.

II.3 Uso de las metodologías en el desarrollo de aplicaciones web

El almacenamiento y análisis de la información ha sido uno de los grandes problemas a que se ha enfrentado el hombre desde que inventó la escritura. No es sino hasta la segunda mitad del siglo XX que el hombre ha podido resolver, parcialmente ese problema gracias a la invención de la computadora y al uso que se le ha dado a la misma.

La industria de ordenadores ha mostrado un progreso espectacular en muy corto tiempo. El viejo modelo de tener un solo ordenador para satisfacer todas las necesidades de cálculo de una organización se está reemplazando con rapidez por otro que considera un número grande de ordenadores separados, pero interconectados, que efectúan el mismo trabajo.

En la sociedad actual el acceso a la información se ha convertido en algo de vital importancia y la buena marcha de las empresas dependerá, cada vez más, de la calidad de sus telecomunicaciones.

Por este motivo el número de ordenadores está aumentando progresivamente en las empresas; pero, este fenómeno ocasiona otra serie de problemas que la empresa necesita solucionar como: enviar la información correcta a la persona indicada, actualizar los programas que se están usando, controlar el acceso a los datos, etc. Cuando se emplea una tecnología que cambia tan rápidamente sin disponer de los controles adecuados, el riesgo más inminente es que el sistema quede obsoleto. Es por ello que cada día se construyen nuevos sistemas capaces de satisfacer las exigencias de las empresas en la actualidad, la mayoría de los cuales se realizan en entornos web que facilitan la accesibilidad de la información desde cualquier puesto de trabajo.

El desarrollo de aplicaciones web no solo con fines educativos sino también productivos, tiene un importante papel en la Universidad de Cienfuegos donde cerca del 75% de los sistemas que se llevan a cabo son para la web.

Desde el punto de vista de la ingeniería del software es importante dotar de los mecanismos adecuados, para que la realización de este tipo de aplicaciones satisfaga las necesidades tanto de los usuarios como de los clientes que contratan el desarrollo de estas aplicaciones. Pero actualmente no existe una metodología universalmente aceptada, que guíe en el proceso de desarrollo de aplicaciones Web.

En cualquier caso, existen criterios universalmente aceptados acerca del desarrollo software. Por ejemplo, y según afirma Jacobson [43] el modelo de proceso más adecuado para el desarrollo de software es un proceso iterativo e

incremental, puesto que a diferencia de otros modelos de proceso, como por ejemplo el modelo en cascada, permite la obtención de diversas versiones del producto software antes de la entrega final del mismo y la depuración y validación progresiva del mismo, lo que sin duda redundará en un software más satisfactorio para usuarios y cliente. Además y según indica [16], con este tipo de proceso es posible añadir o modificar requisitos que no han sido detectados con anterioridad.

Aún no existe ninguna propuesta universalmente aceptada para el desarrollo Web, pero Fraternali [34] indica que una posible solución al desarrollo adecuado de aplicaciones Web, sería combinar los ciclos de vida tradicionales con las propuestas de diseño para el desarrollo de aplicaciones hipermedia. De hecho, algunos de los trabajos existentes, relacionados con la tecnología hipermedia y Web, combinan el tratamiento de esas características especiales, con el uso de un modelo de proceso iterativo e incremental. En cualquier caso los métodos clásicos no son adecuados para el desarrollo de aplicaciones Web, puesto que no contemplen determinadas características específicas de este tipo de aplicaciones, [44]. Por otra parte, las metodologías tradicionales generalmente imponen un proceso de desarrollo demasiado pesado y burocrático según afirma Fowler [45], lo que impide un desarrollo ágil y rápido para este tipo de aplicaciones.

Sin embargo existen proyectos que por su envergadura y la complejidad de la información que en el se maneja resulta necesario aplicar una metodología pesada pues han demostrado ser efectivas y necesarias en proyectos de gran tamaño (respecto a tiempo y recursos), donde por lo general se exige un alto grado de ceremonia en el proceso.

Las metodologías ágiles por su parte están especialmente orientadas para proyectos pequeños y constituyen una solución a medida para ese entorno, aportando una elevada simplificación que a pesar de ello no renuncia a las prácticas esenciales para asegurar la calidad del producto.

En la universidad de Cienfuegos, en particular, se desarrollan diferentes proyectos, a la mayoría de los cuales se le aplican metodologías poco recomendadas teniendo en cuenta las características de los mismos.

II.4 Selección de las metodologías

El acelerado desarrollo de software en la actualidad y la necesidad de que los proyectos sean concluidos exitosamente siendo un producto de gran valor para los clientes, generan grandes cambios en las metodologías adoptadas por los equipos para cumplir sus objetivos, puesto que unas se adaptan mejor que otras al contexto del proyecto brindando mejores ventajas. Debido a ello es de vital importancia la selección de una metodología robusta que ajustada en un equipo cumpla con sus metas, y satisfaga mas allá de las necesidades definidas al inicio del proyecto.

En el momento de seleccionar una metodología para aplicar en la construcción de un sistema es necesario tener en cuenta las características del proyecto y del equipo y ser adaptada al contexto del mismo. Una de las características principales a tener en cuenta es la complejidad del sistema a desarrollar, es decir, es necesario valorar la complejidad del proceso a automatizar, la cantidad de requisitos que deben ser implementados en el sistema y la complejidad y cantidad de información que se maneja en el proceso. La complejidad puede ser alta, media o baja en dependencia de las características antes mencionadas. Por ejemplo un proceso económico de una empresa es más complejo que el proceso de marketing de la misma.

En la Facultad de Informática de la Universidad de Cienfuegos los sistemas que se realizan responden a peticiones no solo del propio centro sino de otras instituciones que necesitan automatizar su gestión.

Esta facultad es una facultad joven que cuenta con 9 de años de experiencia en la Carrera de Ingeniería Informática de la cual ha tenido 4 graduaciones. Como trabajo de diploma los estudiantes desarrollan sistemas para beneficiar al centro docente y a la sociedad en general. Estos software se construyen mayormente orientados a la web, en la Tabla 1 se muestra detalladamente las tesis que se han realizado hasta el momento.

Tabla 1: Clasificación de las tesis realizadas en la FInf de la Ucf

Cursos	*Total de Tesis*	*Aplicaciones Web*	*Complejidad*	
			Media	*Baja*
2004-2005	11	7		
2005-2006	16	12	7	5
2006-2007	24	18	10	8
2007-2008	50	37	23	14
Total	**101**	**74**	**73.23%**	

En el período de tesis los estudiantes deben elegir una metodología para desarrollar exitosamente sus sistemas pero normalmente se inclinan por aplicar la metodología estudiada en la carrera, aunque existen otras que se adaptan mejor al proyecto. Es por ello que en el presente trabajo se propone utilizar las metodologías de desarrollo RUP para proyectos de mediana complejidad donde es preciso que se cuente con documentación acerca del proceso de construcción del software para su posterior mantenimiento y evolución; y Scrum para los proyectos de pequeña complejidad, donde se necesita desarrollo rápido y una respuesta ante los cambios durante el proceso.

Existen otras metodologías que están especialmente enfocadas al desarrollo de aplicaciones Web como es el caso de MIDAS, UWE, RMM y OOHDM pero hacen énfasis en la navegabilidad del usuario y la apariencia visual de la aplicación, por lo cual no es factible usarlas en el caso de los proyectos que se realizan en la Universidad de Cienfuegos que además de tener en cuenta estos aspectos, confieren mayor importancia a la gestión de información.

II.4.1 Por qué utilizar RUP para desarrollar aplicaciones web

No existen dos proyectos de desarrollo de software que sean iguales. Cada uno tiene prioridades, requerimientos, y tecnologías muy diferentes. Sin embargo, en todos los proyectos, se debe minimizar el riesgo, garantizar la predictibilidad de los resultados y entregar software de calidad superior a tiempo. Rational Unified Process, o RUP, es una plataforma flexible de procesos de desarrollo de software

que ayuda brindando guías consistentes y personalizadas de procesos para todo el equipo de proyecto.

RUP describe cómo utilizar de forma efectiva reglas de negocio y procedimientos comerciales probados en el desarrollo de software para equipos de desarrollo de software, conocidos como “mejores prácticas”. Captura varias de las mejores prácticas en el desarrollo moderno de software en una forma que es aplicable para un amplio rango de proyectos y organizaciones. Es una guía de cómo utilizar de manera efectiva UML. Provee a cada miembro del equipo fácil acceso a una base de conocimiento con guías, plantillas y herramientas para todas las actividades críticas de desarrollo. Crea y mantiene modelos, en lugar de enfocarse en la producción de una gran cantidad de papeles de documentación.

Como una plataforma de procesos que abarca todas las prácticas de la industria, RUP permite seleccionar fácilmente el conjunto de componentes de proceso que se ajustan a las necesidades específicas del proyecto. Se pueden alcanzar resultados predecibles unificando el equipo con procesos comunes que optimicen la comunicación y creen un entendimiento común para todas las tareas, responsabilidades y artefactos.

Una de las mejores prácticas centrales de RUP es la noción de desarrollar iterativamente. Rational Unified Process organiza los proyectos en términos de disciplinas y fases, consistiendo cada una en una o más iteraciones. Con esta aproximación iterativa, el énfasis de cada workflow variará a través del ciclo de vida. La aproximación iterativa ayuda a mitigar los riesgos en forma temprana y continua, con un progreso demostrable y frecuentes releases ejecutables. Además provee un entorno de proceso de desarrollo configurable basado en estándares; permite tener claro y accesible el proceso de desarrollo que se sigue y que este sea configurado a las necesidades de la organización y del proyecto.

Otras metodologías trabajan de manera similar a RUP en el tratamiento de los casos de uso como es ICONIX pero no con la profundidad de RUP. Otra de sus debilidades es que no puede ser usado para proyectos grandes y necesita información rápida y puntual de los requisitos, el diseño y las estimaciones.

MSF es otra metodología tradicional pero está basado en desarrollo con tecnología Microsoft lo que limita las opciones del cliente en lo que se refiere a herramientas de desarrollo. Esta metodología hace un análisis de riesgo demasiado exhaustivo que puede frenar el avance del proyecto y además solicita demasiada documentación en todas las fases resultando muy engorroso el trabajo y por consiguiente afecta el tiempo de entrega del producto.

II.4.2 Caracterización detallada de la metodología de desarrollo de software RUP

Los autores de RUP destacan que el proceso de software propuesto por RUP tiene tres características esenciales: está dirigido por los Casos de Uso, está centrado en la arquitectura, y es iterativo e incremental.

Proceso dirigido por Casos de Uso

Según Kruchten [46] los Casos de Uso son una técnica de captura de requisitos que fuerza a pensar en términos de importancia para el usuario y no sólo en términos de funciones que sería bueno contemplar. Se define un Caso de Uso como un fragmento de funcionalidad del sistema que proporciona al usuario un valor añadido. Los Casos de Uso representan los requisitos funcionales del sistema.

En RUP los Casos de Uso no son sólo una herramienta para especificar los requisitos del sistema. También guían su diseño, implementación y prueba. Los Casos de Uso constituyen un elemento integrador y una guía del trabajo.

Los Casos de Uso no sólo inician el proceso de desarrollo sino que proporcionan un hilo conductor, permitiendo establecer trazabilidad entre los artefactos que son generados en las diferentes actividades del proceso de desarrollo.

Como se muestra en la Figura 2, basándose en los Casos de Uso se crean los modelos de análisis y diseño, luego la implementación que los lleva a cabo, y se verifica que efectivamente el producto implemente adecuadamente cada Caso de Uso. Todos los modelos deben estar sincronizados con el modelo de Casos de Uso.

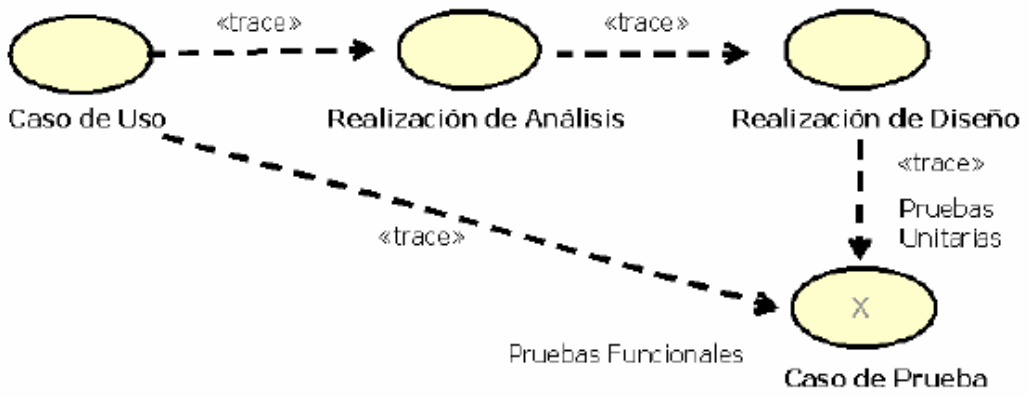


Figura 2: Trazabilidad a partir de los casos de uso

Proceso centrado en la arquitectura

La arquitectura de un sistema es la organización o estructura de sus partes más relevantes, lo que permite tener una visión común entre todos los involucrados (desarrolladores y usuarios) y una perspectiva clara del sistema completo, necesaria para controlar el desarrollo. [46]

La arquitectura involucra los aspectos estáticos y dinámicos más significativos del sistema, está relacionada con la toma de decisiones que indican cómo tiene que ser construido el sistema y ayuda a determinar en qué orden. Además la definición de la arquitectura debe tomar en consideración elementos de calidad del sistema, rendimiento, reutilización y capacidad de evolución por lo que debe ser flexible durante todo el proceso de desarrollo. La arquitectura se ve influenciada por la plataforma software, sistema operativo, gestor de bases de datos, protocolos, consideraciones de desarrollo como sistemas heredados. Muchas de estas restricciones constituyen requisitos no funcionales del sistema.

En el caso de RUP además de utilizar los Casos de Uso para guiar el proceso se presta especial atención al establecimiento temprano de una buena arquitectura

que no se vea fuertemente impactada ante cambios posteriores durante la construcción y el mantenimiento.

Cada producto tiene tanto una función como una forma. La función corresponde a la funcionalidad reflejada en los Casos de Uso y la forma la proporciona la arquitectura. Existe una interacción entre los Casos de Uso y la arquitectura, los Casos de Uso deben encajar en la arquitectura cuando se llevan a cabo y la arquitectura debe permitir el desarrollo de todos los Casos de Uso requeridos, actualmente y en el futuro. Esto provoca que tanto arquitectura como Casos de Uso deban evolucionar en paralelo durante todo el proceso de desarrollo de software.

Proceso iterativo e incremental

Según el libro "El Proceso Unifcado de Desarrollo de Software" [43] el equilibrio correcto entre los Casos de Uso y la arquitectura es algo muy parecido al equilibrio de la forma y la función en el desarrollo del producto, lo cual se consigue con el tiempo. Para esto, la estrategia que se propone en RUP es tener un proceso iterativo e incremental en donde el trabajo se divide en partes más pequeñas o mini proyectos, permitiendo que el equilibrio entre Casos de Uso y arquitectura se vaya logrando durante cada mini proyecto, así durante todo el proceso de desarrollo. Cada mini proyecto se puede ver como una iteración (un recorrido más o menos completo a lo largo de todos los flujos de trabajo fundamentales) del cual se obtiene un incremento que produce un crecimiento en el producto.

Una iteración puede realizarse por medio de una cascada como se muestra en la Figura 3. Se pasa por los flujos fundamentales (Requisitos, Análisis, Diseño, Implementación y Pruebas), también existe una planificación de la iteración, un análisis de la iteración y algunas actividades específicas de la iteración. Al finalizar se realiza una integración de los resultados con lo obtenido de las iteraciones anteriores.

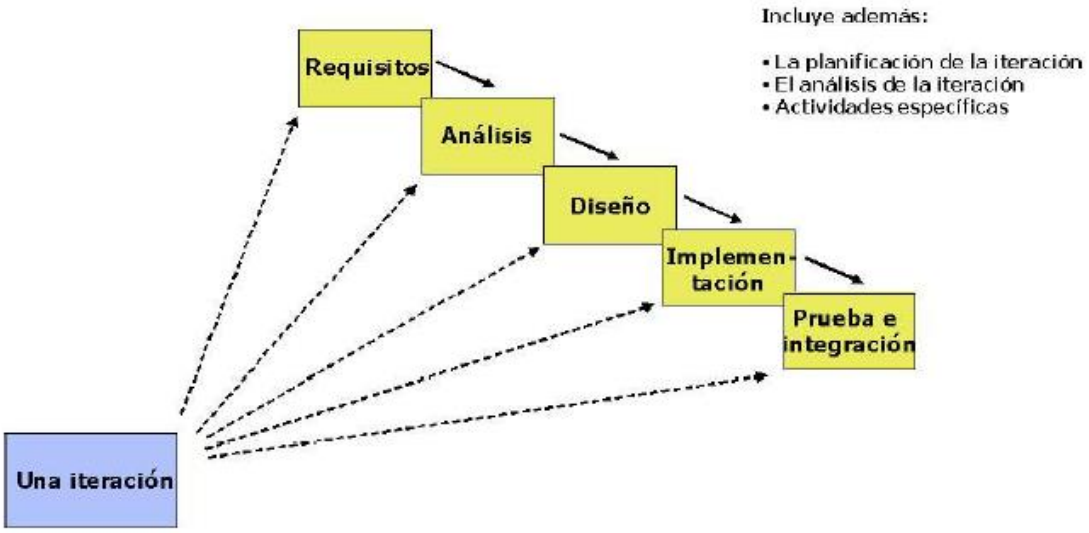


Figura 3: Una iteración en RUP

El proceso iterativo e incremental consta de una secuencia de iteraciones. Cada iteración aborda una parte de la funcionalidad total, pasando por todos los flujos de trabajo relevantes y refinando la arquitectura. Cada iteración se analiza cuando termina. Se puede determinar si han aparecido nuevos requisitos o han cambiado los existentes, afectando a las iteraciones siguientes. Durante la planificación de los detalles de la siguiente iteración, el equipo también examina cómo afectarán los riesgos que aún quedan al trabajo en curso. Toda la retroalimentación de la iteración pasada permite reajustar los objetivos para las siguientes iteraciones. Se continúa con esta dinámica hasta que se haya finalizado por completo con la versión actual del producto.

RUP divide el proceso en cuatro fases, dentro de las cuales se realizan varias iteraciones en número variable según el proyecto y en las que se hace un mayor o menor hincapié en las distintas actividades. En la Figura 3 se muestra cómo varía el esfuerzo asociado a las disciplinas según la fase en la que se encuentre el proyecto RUP.

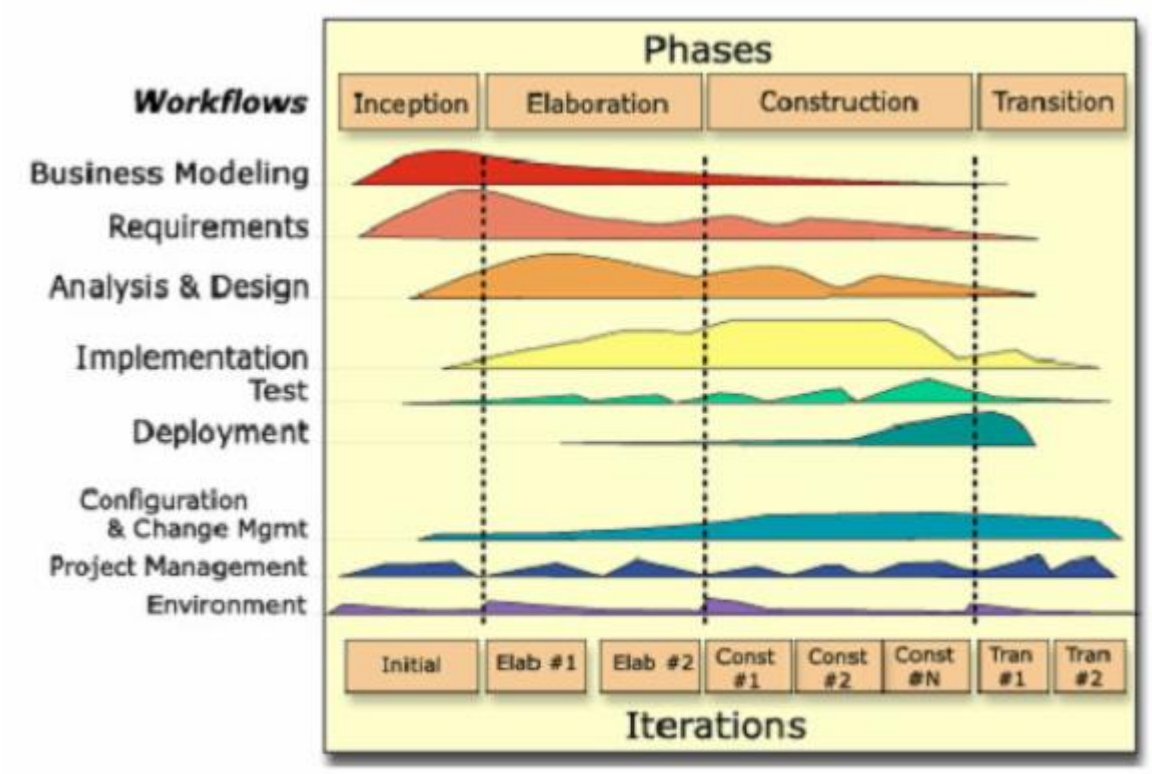


Figura 4: Esfuerzo en actividades según las fases del proyecto

En RUP se han agrupado las actividades en grupos lógicos definiéndose 9 flujos de trabajo principales. Los 6 primeros son conocidos como flujos de ingeniería y los tres últimos como de apoyo.

- Modelamiento del negocio: Describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización.
- Requerimientos: Define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.
- Análisis y diseño: Describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos), por lo que indica con precisión lo que se debe programar.
- Implementación: Define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.
- Prueba (Testeo): Busca los defectos a lo largo del ciclo de vida.

- Instalación: Produce release del producto y realiza actividades (empaque, instalación, asistencia a usuarios, etc.) para entregar el software a los usuarios finales.
- Administración del proyecto: Involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.
- Administración de configuración y cambios: Describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a: utilización/actualización concurrente de elementos, control de versiones, etc.
- Ambiente: Contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización.

Los principales elementos que define esta metodología son:

- Trabajadores: Define el comportamiento y responsabilidades (rol) de un individuo, grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto como un equipo. Ellos realizan las actividades y son propietarios de elementos.
- Actividades: Es una tarea que tiene un propósito claro, es realizada por un trabajador y manipula elementos.
- Artefactos: Productos tangibles del proyecto que son producidos, modificados y usados por las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables.
- Flujo de actividades: Secuencia de actividades realizadas por trabajadores y que produce un resultado de valor observable.

El proceso define una serie de roles que se distribuyen entre los miembros del proyecto y que definen las tareas de cada uno y el resultado (artefactos) que se espera de ellos.

II.4.3 Por qué utilizar Scrum para desarrollar aplicaciones web

En general, las aplicaciones Web cumplen la mayor parte de las características que se mencionan en el epígrafe II.2.2, por lo que la utilización de procesos ágiles podría ser ventajoso para este tipo de desarrollos. La necesidad del cliente que

contrata un desarrollo Web es que su producto esté disponible en la red lo más pronto posible. Si no se tiene en cuenta esta necesidad, la aplicación no resultará un producto provechoso para el cliente. Puesto que los procesos ágiles permiten tener versiones de producto previas a la versión final, si se aplican correctamente estos procesos el cliente podrá disponer de forma rápida de alguna versión intermedia. Además el ciclo de desarrollo de la mayoría de los sitios y aplicaciones Web es extremadamente corto [47]. Por otra parte, los desarrollos Web se perciben como desarrollos sencillos y los desarrolladores son sometidos a una gran presión de trabajo para terminar lo más pronto posible. Esta forma de trabajar implica, sin duda alguna, modificaciones. Luego sería conveniente garantizar un proceso de desarrollo adaptable a los cambios. Otra cuestión fundamental a tener en cuenta es que las aplicaciones Web se desarrollan sin conocer los perfiles de los usuarios finales de las mismas, o lo que es lo mismo sin conocer los requisitos de usuario del sistema. Sin lugar a dudas esto implicará cambios en los requisitos inicialmente detectados, lo que lleva de nuevo a la elección de un proceso adaptativo.

Scrum es un proceso en el que se aplican de manera regular un conjunto de mejores prácticas para trabajar en equipo y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos.

En esta metodología se realizan entregas parciales del resultado final del proyecto, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, Scrum está especialmente indicado para proyectos en entornos complejos**,** donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad y la productividad son fundamentales.

Aunque Scrum surgió como modelo para el desarrollo de productos tecnológicos, también se emplea en entornos que trabajan con requisitos inestables y que requieren rapidez y flexibilidad; situaciones frecuentes en el desarrollo de determinados sistemas de software, y donde al aplicarlo proporciona ventajas como:

- Entrega de un producto funcional al finalizar cada iteración.
- Posibilidad de ajustar la funcionalidad en base a la necesidad de negocio del cliente.
- Visualización del proyecto día a día.
- Alcance acotado y viable.
- Equipos integrados y comprometidos con el proyecto, toda vez que ellos definieron el proyecto.

Entre principales beneficios que reporta utilizar Scrum como metodología de desarrollo de un sistema se encuentran además: [48]

- Entrega mensual (o quincenal) de resultados (los requisitos más prioritarios en ese momento, ya completados) lo cual proporciona las siguientes ventajas:
    - Gestión regular de las expectativas del cliente y basada en resultados tangibles. El cliente establece sus expectativas indicando el valor que le aporta cada requisito del proyecto y cuando espera que esté completado; y comprueba de manera regular si se van cumpliendo sus expectativas, da feedback, ya desde el inicio del proyecto puede tomar decisiones informadas a partir de resultados objetivos y dirige estos resultados del proyecto, iteración a iteración, hacia su meta.
    - Resultados anticipados (*time to market*). El cliente puede empezar a utilizar los resultados más importantes del proyecto antes de que esté finalizado por completo.
    - Flexibilidad y adaptación respecto a las necesidades del cliente, cambios en el mercado, etc. De manera regular el cliente redirige el proyecto en función de sus nuevas prioridades, de los cambios en el mercado, de los requisitos completados que le permiten entender mejor el producto, de la velocidad real de desarrollo, etc.
    - Gestión sistemática del Retorno de Inversión (ROI). De manera regular, el cliente maximiza el ROI del proyecto. Cuando el beneficio pendiente de obtener es menor que el coste de desarrollo, el cliente puede finalizar el proyecto.

  - Mitigación sistemática de los riesgos del proyecto. Desde la primera iteración el equipo tiene que gestionar los problemas que pueden aparecer en una entrega del proyecto. Al hacer patentes estos riesgos, es posible iniciar su mitigación de manera anticipada. La cantidad de riesgo a que se enfrenta el equipo está limitada a los requisitos que se puede desarrollar en una iteración. La complejidad y riesgos del proyecto se dividen de manera natural en iteraciones

- Productividad y calidad. De manera regular el equipo va mejorando y simplificando su forma de trabajar.
- Alineamiento entre el cliente y el equipo de desarrollo. Los resultados y esfuerzos del proyecto se miden en forma de objetivos y requisitos entregados al negocio. Todos los participantes en el proyecto conocen cuál es el objetivo a conseguir. El producto se enriquece con las aportaciones de todos.
- Equipo motivado. Las personas están más motivadas cuando pueden usar su creatividad para resolver problemas y cuando pueden decidir organizar su trabajo.

Scrum también se utiliza para resolver situaciones en que no se está entregando al cliente lo que necesita, cuando las entregas se alargan demasiado**,** los costes se disparan o la calidad no es aceptable, cuando se necesita capacidad de reacción ante la competencia, cuando la moral de los equipos es baja y la rotación alta, cuando es necesario identificar y solucionar ineficiencias sistemáticamente o cuando se quiere trabajar utilizando un proceso especializado en el desarrollo de producto.

Muchos desarrolladores en la actualidad se ven en la disyuntiva de utilizar Extreme Programming o Scrum en sus sistemas pero es válido aclarar que XP es recomendable emplearlo solo en proyectos a corto plazo y reporta altas comisiones en caso de fallar, además puede no ser más fácil que el desarrollo tradicional; los usuarios pueden no querer frecuentes pequeños releases y requiere un rígido ajuste a los principios XP. Además es difícil predecir costo y tiempo de desarrollo ya que no se precisa los elementos a acoplarse en el

proyecto y mantener el producto puede ser difícil, debido a que tiene muy poca documentación.

II.4.4 Caracterización detallada de la metodología de desarrollo de software Scrum

El proceso

En Scrum [48] un proyecto se ejecuta en bloques temporales (iteraciones) de un mes natural (pueden ser de dos semanas, si así se necesita). Cada iteración tiene que proporcionar un resultado completo, un incremento de producto que sea susceptible de ser entregado con el mínimo esfuerzo cuando el cliente lo solicite.

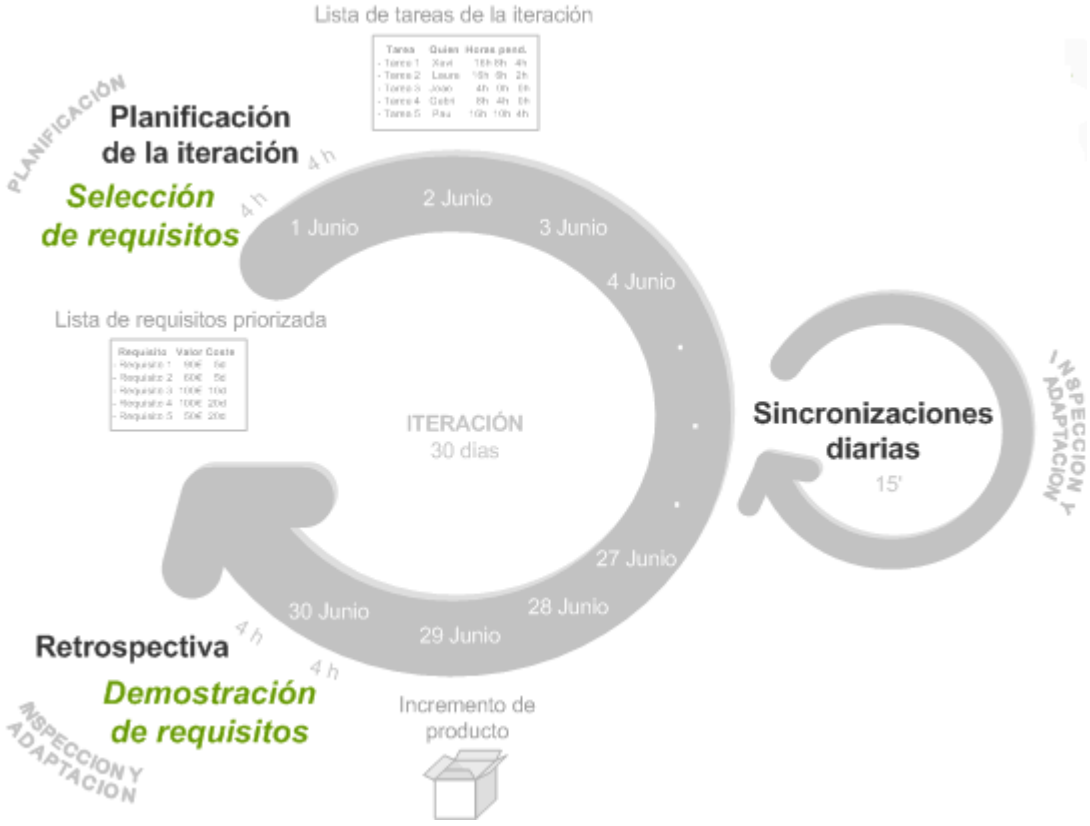


Figura 5: Diagrama del proceso de desarrollo de Scrum

El proceso parte de la lista de requisitos priorizada del producto, que actúa como plan del proyecto. En esta lista el cliente ha priorizado los requisitos balanceando el valor que le aportan respecto a su coste y han sido divididos en iteraciones y entregas.

Las actividades que se llevan se plantea realizar en la metodología Scrum son las siguientes:

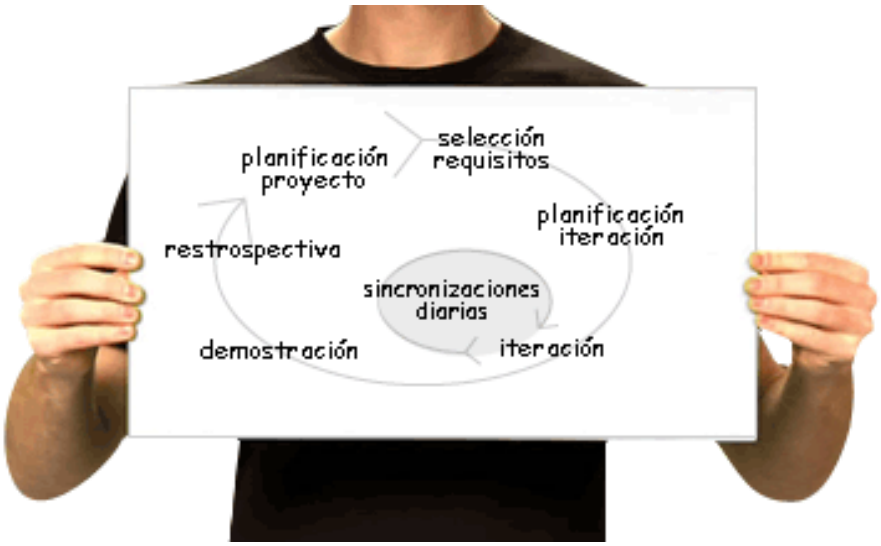


Figura 6: Actividades del proceso de Scrum

Planificación de la iteración

La planificación de las tareas a realizar en la iteración se divide en dos partes:

Primera parte de la reunión. Se realiza en un timebox de cómo máximo 4 horas:

- El cliente presenta al equipo la lista de requisitos priorizada del producto o proyecto, pone nombre a la meta de la iteración (de manera que ayude a tomar decisiones durante su ejecución) y propone los requisitos más prioritarios a desarrollar en ella.
- El equipo examina la lista, pregunta al cliente las dudas que le surgen y selecciona los requisitos más prioritarios que se compromete a completar en la iteración, de manera que puedan ser entregados si el cliente lo solicita.

Segunda parte de la reunión. Se realiza en un timebox de máximo 4 horas. El equipo planifica la iteración, dado que ha adquirido un compromiso, es el responsable de organizar su trabajo y es quien mejor conoce cómo realizarlo.

- Define las tareas necesarias para poder completar cada requisito, creando la lista de tareas de la iteración.

- Realiza una estimación conjunta del esfuerzo necesario para realizar cada tarea.
- Cada miembro del equipo se asigna a las tareas que puede realizar.

Beneficios

Potenciación responsable de organizar su trabajo y es quien mejor conoce cómo realizarlo.

- Define las tareas necesarias para poder completar cada requisito, creando la lista de tareas de la iteración.
- Realiza una estimación conjunta del esfuerzo necesario para realizar cada tarea.

Cada miembro del compromiso del equipo:

- Es el equipo quien asume la responsabilidad de completar en la iteración los requisitos que selecciona.
- Es cada una de las personas la que se responsabiliza de realizar las tareas a las que se asigna.

Una estimación conjunta es más fiable, dado que tiene en cuenta los diferentes conocimientos, experiencia y habilidades de los integrantes del equipo.

Ejecución de la iteración (sprint)

En Scrum un proyecto se ejecuta en iteraciones de un mes natural (pueden ser de dos semanas, si así se necesita). Cada iteración tiene que proporcionar un resultado completo, un incremento de producto que sea susceptible de ser entregado con el mínimo esfuerzo cuando el cliente lo solicite.

Cada día el equipo realiza una reunión de sincronización, donde cada miembro inspecciona el trabajo de los otros para poder hacer las adaptaciones necesarias, así cómo comunicar cuales son los impedimentos con que se encuentra.

- El Facilitador se encarga de que el equipo pueda cumplir con su compromiso y de que no se merme su productividad. Elimina los obstáculos que el equipo no puede resolver por sí mismo. Protege al equipo de interrupciones externas que puedan afectar su compromiso o su productividad.

Recomendaciones

- Para poder completar el máximo de requisitos en la iteración, se debe minimizar el número de requisitos en que el equipo trabaja simultáneamente (*WIP, Work In Progress*), completando primero los que den más valor al cliente. Esta forma de trabajar, que se ve facilitada por la propia estructura de la lista de tareas de la iteración, permite tener más capacidad de reacción frente a cambios o situaciones inesperadas.

Restricciones

- No se puede cambiar los requisitos de la iteración en curso**.**

El hecho de no poder cambiar los requisitos de la iteración una vez iniciada facilita que el cliente cumpla con su responsabilidad de conocer qué es lo más prioritario a desarrollar, antes de iniciar la iteración.

Terminación anormal de la iteración

Sólo en situaciones muy excepcionales el cliente o el equipo pueden solicitar una terminación anormal de la iteración. Esto puede suceder si, por ejemplo, el contexto del proyecto ha cambiado enormemente y no es posible esperar al final de la iteración para aplicar cambios, o si el equipo encuentra que es imposible cumplir con el compromiso adquirido. En ese caso, se dará por finalizada la iteración y se dará inicio a otra mediante una reunión de planificación de la iteración.

Reunión diaria de sincronización del equipo (Scrum daily meeting)

El objetivo de esta reunión es facilitar la transferencia de información y la colaboración entre los miembros del equipo para aumentar su productividad.

Cada miembro del equipo inspecciona el trabajo que el resto está realizando (dependencias entre tareas, progreso hacia el objetivo de la iteración, obstáculos que pueden impedir este objetivo) para al finalizar la reunión poder hacer las adaptaciones necesarias que permitan cumplir con el compromiso conjunto que el equipo adquirió para la iteración (en la reunión de planificación de la iteración).

Cada miembro del equipo debe responder las siguientes preguntas en un timebox de cómo máximo 15 minutos:

- ¿Qué he hecho desde la última reunión de sincronización? ¿Pude hacer todo lo que tenía planeado? ¿Cuál fue el problema?
- ¿Qué voy a hacer a partir de este momento?
- ¿Qué impedimentos tengo o voy a tener para cumplir mis compromisos en esta iteración y en el proyecto?

Como apoyo a la reunión, el equipo cuenta con la lista de tareas de la iteración, donde se actualiza el estado y el esfuerzo pendiente para cada tarea, así como con el gráfico de horas pendientes en la iteración.

Recomendaciones

- Realizar la reunión diaria de sincronización de pie, para que los miembros del equipo no se relajen ni se extiendan en más detalles de los necesarios.
- Realizar las reuniones de colaboración entre miembros del equipo justo después de la de sincronización.

Demostración de requisitos completados (Sprint Demonstration)

Reunión informal donde el equipo presenta al cliente los requisitos completados en la iteración, en forma de incremento de producto preparado para ser entregado con el mínimo esfuerzo, haciendo un recorrido por ellos lo más real y cercano posible al objetivo que se pretende cubrir.

En función de los resultados mostrados y de los cambios que haya habido en el contexto del proyecto, el cliente realiza las adaptaciones necesarias de manera objetiva, ya desde la primera iteración, replanificando el proyecto.

Se realiza en un timebox de cómo máximo 4 horas.

Beneficios

- El cliente puede ver de manera objetiva cómo han sido desarrollados los requisitos que proporcionó, ver si se cumplen sus expectativas, entender más qué es lo que necesita y tomar mejores decisiones respecto al proyecto.

- El equipo puede ver si realmente entendió cuáles eran los requisitos que solicitó el cliente y ver en qué puntos hay que mejorar la comunicación entre ambos.
- El equipo se siente más satisfecho cuando puede ir mostrando los resultados que va obteniendo. No está meses trabajando sin poder exhibir su obra.

Retrospectiva (Sprint Retrospective)

El equipo analiza cómo ha sido su manera de trabajar durante la iteración, qué cosas han funcionado bien, cuáles hay que mejorar, qué cosas quiere probar hacer en la siguiente iteración, qué se ha aprendido y cuáles son los problemas que podrían impedirle progresar adecuadamente, con el objetivo de mejorar de manera continua su productividad. El Facilitador se encargará de ir eliminando los obstáculos identificados que el propio equipo no pueda resolver por sí mismo.

Se realiza en un timebox de cómo máximo 3 horas.

Beneficios

- Incrementa la productividad y el aprendizaje del equipo de manera sistemática, iteración a iteración, con resultados a corto plazo.

Replanificación del proyecto

Durante el transcurso de una iteración, el cliente va trabajando en la lista de requisitos priorizada del producto o proyecto, añadiendo requisitos, modificándolos, eliminándolos, repriorizándolos, cambiando el contenido de iteraciones y definiendo un calendario de entregas que se ajuste mejor a sus nuevas necesidades.

Los cambios en la lista de requisitos pueden ser debidos a:

- Modificaciones que el cliente solicita tras la demostración que el equipo realiza al final de cada iteración sobre los resultados obtenidos, ahora que el cliente entiende mejor el producto o proyecto.
- Cambios en el contexto del proyecto (sacar al mercado un producto antes que su competidor, hacer frente a urgencias o nuevas peticiones de clientes, etc).

- Nuevos requisitos o tareas como resultado de nuevos riesgos en el proyecto.

Para realizar esta tarea, el cliente colabora con el equipo y obtiene de él la estimación de costes de desarrollo para completar cada requisito. El equipo ajusta el factor de complejidad, el coste para completar los requisitos y su velocidad de desarrollo en función de la experiencia adquirida hasta ese momento en el proyecto.

Hay que notar que el equipo sigue trabajando con los requisitos de la iteración en curso, (que de hecho eran los más prioritarios al iniciar la iteración). No es posible cambiar los requisitos que se desarrollan durante la iteración. En la reunión de planificación de la iteración el cliente presentará la nueva lista de requisitos para que sea desarrollada.

Beneficios

De manera sistemática, iteración a iteración, se obtienen los siguientes beneficios:

- El cliente puede tomar decisiones con tiempo respecto al progreso del proyecto y posibles desviaciones:
  - Replanificar el proyecto para obtener un nuevo calendario de entregas que cumpla con sus necesidades actuales.
  - Incorporar nuevos recursos.
  - Cancelar el proyecto con los requisitos completados hasta el momento plenamente operativos, si el beneficio pendiente de obtener es menor que el coste de desarrollo.
- El plan de proyecto se actualiza con la velocidad de desarrollo del equipo, se evitan sorpresas de última hora.

Cuando se aplica la metodología Scrum se determinan las responsabilidades siguientes:

Cliente (Product Owner)

Las responsabilidades del Cliente (que puede ser interno o externo a la organización) son:

- Ser el representante de todas las personas interesadas en los resultados del proyecto (internas o externas a la organización, promotores del proyecto y usuarios finales) y actuar como interlocutor único ante el equipo, con autoridad para tomar decisiones.
- Definir los objetivos del producto o proyecto.
- Dirigir los resultados del proyecto y maximizar su ROI (*Return Of Investment*).
    - Es el propietario de la planificación del proyecto: crea y mantiene la lista priorizada con los requisitos necesarios para cubrir los objetivos del producto o proyecto, conoce el valor que aportará cada requisito y calcula el ROI a partir del coste de cada requisito que le proporciona el equipo.
    - Divide la lista de requisitos estableciendo un calendario de entregas.
    - Antes de iniciar cada iteración replanifica el proyecto en función de los requisitos que aportan más valor en ese momento, de los requisitos completados en la iteración anterior y del contexto del proyecto en ese momento (demandas del mercado, movimientos de la competencia, etc.).
- Participar en la reunión de planificación de iteración, proponiendo los requisitos más prioritarios a desarrollar, respondiendo a las dudas del equipo y detallando los requisitos que el equipo se compromete a hacer.
- Estar disponible durante el curso de la iteración para responder a las preguntas que puedan aparecer.
- No cambiar los requisitos que se están desarrollando en una iteración, una vez está iniciada.
- Participar en la reunión de demostración de la iteración, revisando los requisitos completados

Facilitador (Scrum Master)

Lidera al equipo llevando a cabo las siguientes responsabilidades:

- Velar que todos los participantes del proyecto sigan las reglas y proceso de Scrum, encajándolas en la cultura de la organización, y guiar

la colaboración intraequipo y con el cliente de manera que las sinergias sean máximas. Esto implica:

- Asegurar que la lista de requisitos priorizada esté preparada antes de la siguiente iteración.
- Facilitar las reuniones de Scrum (planificación de la iteración, reuniones diarias de sincronización del equipo, demostración, retrospectiva), de manera que sean productivas y consigan sus objetivos.

- Quitar los impedimentos que el equipo tiene en su camino para conseguir el objetivo de cada iteración (proporcionar un resultado útil al cliente de la manera más efectiva) y poder finalizar el proyecto con éxito. Estos obstáculos se identifican de manera sistemática en las reuniones diarias de sincronización del equipo y en las reuniones de retrospectiva.
- Proteger y aislar al equipo de interrupciones externas durante la ejecución de la iteración (introducción de nuevos requisitos, "secuestro" no previsto de un miembro del equipo, etc.). De esta manera, el equipo puede mantener su productividad y el compromiso que adquirió sobre los requisitos que completaría en la iteración [notar, sin embargo, que el equipo debe reservar tiempo para colaborar con al cliente en la preparación de la lista de requisitos para la próxima iteración].
- Asegurar que los requisitos se desarrollan con calidad.
- Enseñar al equipo a autogestionarse. No da respuestas, si no que guía al equipo con preguntas para que descubra por sí mismo una solución.

Equipo (Team)

Grupo de personas que de manera conjunta desarrollan el producto del proyecto. Comparten la responsabilidad del trabajo que realizan (así como de su calidad) en cada iteración y en el proyecto. El tamaño del equipo está entre 5 y 9 personas.

Por debajo de 5 personas cualquier imprevisto o interrupción sobre un miembro del equipo compromete seriamente el compromiso que han adquirido y, por tanto, el resultado que se va a entregar al cliente al finalizar la iteración. Por encima de 9 personas, la comunicación y colaboración entre todos los miembros se hace más difícil y se forma subgrupos (ver los requisitos de Scrum).

Es un equipo autogestionado, que realiza de manera conjunta las siguientes actividades:

- Seleccionar los requisitos que se compromete a completar en una iteración, de forma que estén preparados para ser entregados al cliente.
- En la lista de requisitos priorizados del producto, estimar la complejidad de cada uno de ellos.
- En la reunión de planificación de la iteración decide cómo va a realizar su trabajo:
    - Seleccionar los requisitos que pueden completar en cada iteración, realizando al cliente las preguntas necesarias.
    - Identificar todas las tareas necesarias para completar cada requisito.
    - Estimar el esfuerzo necesario para realizar cada tarea.
    - Cada miembro del equipo se asigna a las tareas.
- Durante la iteración, trabajar de manera conjunta para conseguir los objetivos de la iteración. Cada especialista lidera el trabajo en su área y el resto colaboran si es necesario para poder completar un requisito.
- Al finalizar la iteración:
    - Demostrar al cliente los requisitos completados en cada iteración.
    - Hacer una retrospectiva final de cada iteración para mejorar de forma continua su manera de trabajar.

El equipo es multidisciplinar:

- Los miembros del equipo tienen las habilidades necesarias para poder identificar y ejecutar todas las tareas que permiten proporcionar al cliente los requisitos comprometidos en la iteración.

- Tienen que depender lo mínimo de personas externas al equipo, de manera que el compromiso que adquieren en cada iteración no se ponga en peligro.
- Se crea una sinergia que permite que el resultado sea más rico al nutrirse de las diferentes experiencias, conocimientos y habilidades de todos. Colaboración creativa.

Los miembros del equipo dedicarse al proyecto a tiempo completo para evitar dañar su productividad por cambios de tareas en diferentes proyectos, para evitar interrupciones externas y así poder mantener el compromiso que adquieren en cada iteración.

Todos los miembros del equipo trabajan en la misma localización física, para poder maximizar la comunicación entre ellos mediante conversaciones cara a cara, diagramas en pizarras blancas, etc. De esta manera se minimizan otros canales de comunicación menos eficientes, que hacen que las tareas se transformen en un "pasa pelota" o que hacen perder el tiempo en el establecimiento de la comunicación (como cuando se llama repetidas veces por teléfono cuando la persona no está en su puesto).

El equipo debe ser estable durante el proyecto, sus miembros deben cambiar lo mínimo posible, para poder aprovechar el esfuerzo que les ha costado construir sus relaciones interpersonales, engranarse y establecer su organización del trabajo.

Entre las herramientas que son necesarias en la aplicación de Scrum se encuentran:

Lista de requisitos priorizada (Product Backlog)

La lista de requisitos priorizada representa las expectativas del cliente respecto a los objetivos y entregas del producto o proyecto. El cliente es el responsable de crear y gestionar la lista (con la ayuda del Facilitador y del equipo, quien proporciona el coste estimado de completar cada requisito). Al reflejar las expectativas del cliente, esta lista permite involucrarle en la dirección de los resultados del producto o proyecto.

- Contiene los requisitos de alto nivel del producto o proyecto. Para cada

requisito se indica el valor que aporta al cliente y el coste estimado de completarlo. La lista está priorizada balanceando el valor que cada requisito aporta al negocio frente al coste estimado que tiene su desarrollo, es decir, basándose en el Retorno de la Inversión (ROI).

- En la lista se indican las posibles iteraciones y las entregas esperadas por el cliente (los puntos en los cuales desea que se le entreguen los requisitos completados hasta ese momento), en función de la velocidad de desarrollo del (los) equipo(s) que trabajará(n) en el proyecto.
- La lista también tiene que considerar los riesgos del proyecto e incluir los requisitos o tareas necesarios para mitigarlos.

Antes de iniciar la primera iteración, el cliente debe tener definida la meta del producto o proyecto y la lista de requisitos creada. No es necesario que la lista sea completa ni que todos los requisitos estén detallados al mismo nivel. Basta con que estén identificados y con suficiente detalle los requisitos más prioritarios con los que el equipo empezará a trabajar. Los requisitos de iteraciones futuras pueden ser mucho más amplios y generales. La incertidumbre y complejidad propia de un proyecto hacen conveniente no detallar todos los requisitos hasta que su desarrollo esté próximo. De esta manera, el esfuerzo de recoger, detallar y desarrollar el resto de requisitos (menos prioritarios) está repartido en el período de ejecución del proyecto. Esto produce varias ventajas:

- Se evita caer en parálisis de análisis al inicio del proyecto, de manera que se puede iniciar antes el desarrollo y el cliente puede empezar a obtener resultados útiles.
- Se evita analizar en detalle requisitos no prioritarios que podrían cambiar durante el transcurso del proyecto, dado que se conocerá mejor cuál ha de ser el resultado a conseguir, o bien porque podrían ser reemplazados por otros.
- Puede llegar a un punto del proyecto en que no valga la pena analizar ni desarrollar los requisitos restantes, dado el poco retorno de inversión (ROI) que tienen.

En el caso del desarrollo de un producto, la lista va evolucionando durante toda la vida del producto (los requisitos "emergen"). En el caso de un proyecto, conforme éste avance irán apareciendo los requisitos menos prioritarios que falten.

El cliente y el equipo tienen que acordar la definición de “completado” de los requisitos, qué será lo que el equipo habrá realizado para considerar que el producto esté preparado para ser entregado al cliente al finalizar cada iteración, de manera que no haya tareas pendientes que puedan impedir utilizar los resultados del proyecto lo antes posible. De este modo, el cliente podrá tomar decisiones correctas cuando al final de cada iteración el equipo le haga una demostración de los requisitos completados (por ejemplo, solicitar una entrega del producto).

Cuando el cliente solicita una entrega de los requisitos completados hasta ese momento, el equipo puede necesitar añadir una iteración de entrega, más corta que las iteraciones habituales, donde realizar alguna tarea que no ha sido necesaria o posible hasta el momento de la entrega final.

Lista de tareas de la iteración (Sprint Backlog)

Lista de tareas que el equipo elabora como plan para completar los requisitos seleccionados para la iteración y que se compromete a demostrar al cliente al finalizar la iteración, en forma de incremento de producto preparado para ser entregado.

Esta lista permite ver las tareas donde el equipo está teniendo problemas y no avanza, con lo que le permite tomar decisiones al respecto.

La lista contiene las tareas, el esfuerzo pendiente para finalizarlas y la autoasignación que han hecho los miembros del equipo.

El progreso de la iteración y su velocidad con respecto a tareas u horas pendientes se muestra mediante un gráfico de trabajo pendiente (Burndown chart).

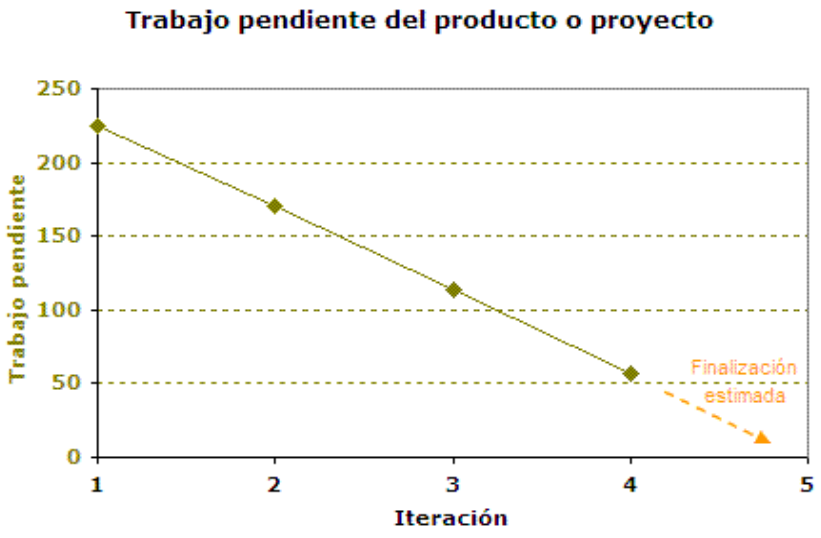
Gráficos de trabajo pendiente (Burndown charts)

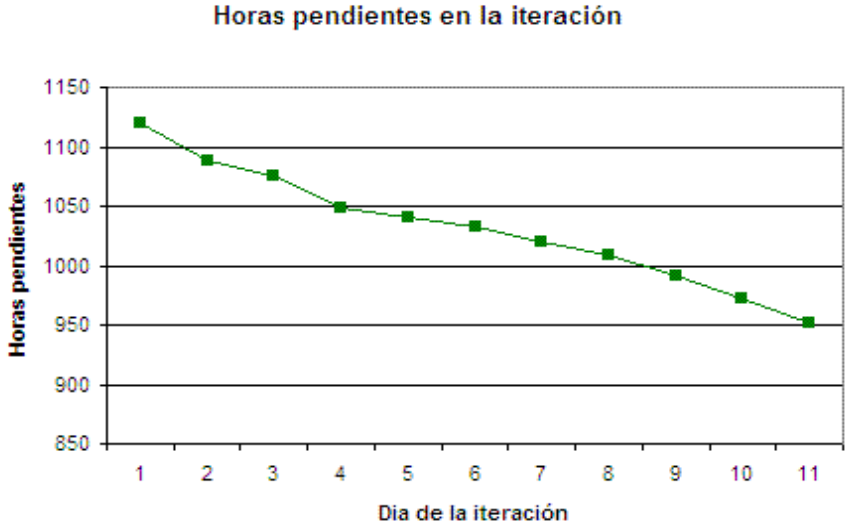
Un gráfico de trabajo pendiente a lo largo del tiempo muestra la velocidad a la que se está completando los requisitos. Permite extrapolar si el Equipo podrá completar el trabajo en el tiempo estimado.

Se pueden utilizar los siguientes gráficos de esfuerzo pendiente:

- Días pendientes para completar los requisitos del producto o proyecto (product burndown chart), realizado a partir de la lista de requisitos priorizada (Product Backlog).
- Horas pendientes para completar las tareas de la iteración (sprint burndown chart), realizado a partir de la lista de tareas de la iteración (Iteration Backlog).

Este tipo de gráfico permite realizar diversas simulaciones: ver cómo se aplazan las fechas de entrega si se le añaden requisitos, ver cómo se avanzan si se le quitan requisitos o se añade otro equipo, etc.





II.5 Conclusiones del capítulo

En este capítulo se aborda con profundidad el tema de las metodologías de desarrollo, se definen los conceptos asociados con el tópico y se especifican además los diferentes tipos de metodologías. Se hace un estudio de la necesidad de desarrollar aplicaciones web y de emplear metodologías en la construcción de las mismas para lograr sistemas de alta calidad y competitividad. Asimismo se proponen dos metodologías a aplicar a proyectos de desarrollo web, exponiéndose la razón de la elección y se describen detalladamente estas metodologías, puntualizándose sus características y los beneficios de su utilización.

Capítulo III: Aplicación de las metodologías de desarrollo de software RUP y Scrum

En este capítulo se aplican, de forma general, cada una de las metodologías de desarrollo de software seleccionadas anteriormente en la construcción del Sistema De Gestión De Información Del Control Interno Informático (SGICII). Se describe detalladamente además el proceso a automatizar que va a ser objeto estudio.

III.1 Descripción del proceso a automatizar

En la mayoría de las empresas e instituciones existe una extensa red de computadoras que tiene como objetivo interconectar los sistemas de cómputo de las áreas y la integración de las aplicaciones en red, de modo que permita el intercambio de información en todas las esferas. Con el propósito de hacer cumplir estos objetivos existen normas y medidas de Seguridad Informática que garantizan el cuidado de la información que a través de ella se maneja.

Para garantizar la seguridad de la información y de las tecnologías informáticas se realizan controles sistemáticos a los equipos de cómputo de la red, además, estos controles pueden ser producto de cualquier eventualidad ocurrida, y realizadas por el Responsable de Seguridad Informática de la propia institución o de organismos facultados para tal desempeño.

Usualmente la información que se deriva de tales controles es recopilada en archivos físicos y digitales, cuya organización, búsqueda y actualización resulta engorrosa; existiendo demora en el manejo de la documentación para el proceso de solicitud de informes de controles.

Con la construcción de un sistema se facilita el rápido acceso a la información referente al proceso en cuestión, evita la demora que produce el tratamiento manual de la información y contribuye a la organización y control de esta, permitiendo obtener una evaluación del control interno basado en criterios propios del proceso expuestos en cuestionarios. Se mejoran además, las condiciones de trabajo de los empleados involucrados en el control interno, logrando una reducción del tiempo y de los costos asociados a la labor.

Siendo así el software queda concebido como SISTEMA DE GESTIÓN DE INFORMACIÓN DEL CONTROL INTERNO INFORMÁTICO (SGICII) que está compuesto por 5 módulos fundamentales:

• Módulo de Codificadores
• Módulo de Configuración del Sistema
• Módulo de Gestión de Usuarios
• Módulo de Registros de controles
• Módulo de Archivos
• Módulo de Ayuda

El módulo **Codificadores** es el encargado de la gestión de los datos persistentes en el sistema, con el objetivo de minimizar la inserción de información por parte del usuario. Solo se pueden insertar y actualizar codificadores.

El módulo **Configuración del Sistema** permite establecer parámetros necesarios para el adecuado funcionamiento de la aplicación contando para ello con dos sub-módulos:

• Sub-módulo de Gestión del Control Interno.
• Sub-módulo de Gestión de Información de la Entidad.

Gestión del Control Interno: es donde se establecen y se crean las plantillas del cuestionario a utilizar, respondiendo a dos tipos de filosofía de calificación, según el método seleccionado para la evaluación se establecen parámetros para obtener el resultado final del control interno.

Gestión de Información de la Entidad: se registra la información de la entidad donde se implementa el sistema, para incluirla posteriormente en documentos de salida e interfaz de presentación.

El módulo **Gestión de Usuarios** es el encargado de la administración de los usuarios que van a trabajar en el sistema, permite añadir y hacer búsquedas de los mismos basado en parámetros, para modificar sus datos, como contraseña, privilegios administrativos e información personal.

El módulo **Registros de Controles** es donde se llevan a cabo los registros de los controles internos establecidos por la entidad y las informaciones concernientes a las computadoras.

En el módulo **Archivos** es donde se obtienen informes de los controles, registros y estadísticas de la información almacenada.

III.2 Descripción de las herramientas a utilizar para aplicar RUP y Scrum

Para apoyar la construcción de sistemas y debido al auge de las metodologías se han creado herramientas que facilitan el trabajo de los desarrolladores y automatizan la mayor parte del proceso. Además mejoran de la calidad de los desarrollos realizados y aumentan la productividad de los equipos de trabajo.

III.2.1 Rational Rose

Hoy día, muchas empresas se han extendido a la adquisición de herramientas CASE (Ingeniería Asistida por Computadora), con el fin de automatizar los aspectos claves de todo el proceso de desarrollo de un sistema, desde el inicio hasta el final.

Rational Rose [49] es una herramienta CASE que da soporte al modelado visual con UML ofreciendo distintas perspectivas del sistema.

Da soporte al Proceso Unificado de Rational (RUP) para el desarrollo de los proyectos de software, desde la etapa de Ingeniería de Requerimientos hasta la etapa de pruebas. Para cada una de estas etapas existe una herramienta que ayuda en la administración de los proyectos, Rose es la herramienta de Rational para la etapa de análisis y diseño de sistemas.

- Modelado de Negocio
- Captura de Requisitos (parcial)
- Análisis y Diseño (Completo)
- Implementación (como ayuda)
- Control de Cambio y gestión de configuración

Rational Rose ofrece un diseño dirigido por modelos que redundan en una mayor productividad de los desarrolladores, admitiendo el lenguaje de modelado UML y técnicas de modelado de objetos (OMT). Asimismo favorece el diseño centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad. Utiliza un lenguaje estándar común a todo el equipo de desarrollo que facilita la

comunicación. Posee capacidades de ingeniería inversa. Con Rational Rose el modelo y código permanece sincronizado en todo el ciclo de desarrollo. Además se encuentra disponible en múltiples plataformas.

III.2.2 Sprintometer

Sprintometer [50] es una aplicación simple y fácil de usar en el desarrollo ágil de proyectos. Es una herramienta para la gestión y el seguimiento de Scrum. Con el fin de simplificar el intercambio de datos con programas externos todas las hojas de cálculo en Sprintometer se pueden exportar a Microsoft Excel, también se puede utilizar el formato XML para los archivos locales.

Algunas de las principales características de este software son:

- Seguimiento de proyectos con Scrum y XP.
- Moderno y fácil de usar con una interfaz estilo Microsoft Office 2007.
- Aumento de información en los gráficos.
- Seguimiento separado de desarrollo y pruebas.
- Seguimiento de iteraciones (Sprint) en equipos de composición variable.
- Predicción de la desviación prevista de la iteración de la fecha de finalización.
- Asignación de recursos a tareas e historias de usuarios.
- Exportación para Microsoft Excel de todos los gráficos y las hojas de cálculo.
- Exportación / Importación de los datos del proyecto en formato XML.

III.3 Aplicación de RUP

Cuando se utiliza la metodología RUP (*Rational Unified Process*) en el desarrollo de un sistema se deben seguir los flujos de trabajo en los que se han agrupado las actividades, que se especifican en el capítulo anterior. En el Anexo 29 se muestra una guía de los pasos a seguir para aplicar eficientemente esta metodología.

III.3.1 Modelado del negocio

El modelado del negocio es una técnica que permite comprender los procesos de negocio de la organización y se desarrolla en dos pasos:[43]

1. Confección de un modelo de casos de uso del negocio que identifique los actores y casos de uso del negocio que utilicen los actores.
2. Desarrollo de un modelo de objetos del negocio compuesto por trabajadores y entidades del negocio que juntos realizan los casos de uso del negocio.

Modelo de casos de uso del negocio

El modelo de Casos de Uso del Negocio (CUN) describe los procesos de una empresa en términos de casos de uso y actores del negocio en correspondencia con los procesos del negocio y los clientes, respectivamente. El modelo de casos de uso presenta un sistema desde la perspectiva de su uso y esquematiza cómo proporciona valor a sus usuarios. Este modelo permite a los modeladores comprender mejor qué valor proporciona el negocio a sus actores. [43]

Este modelo es definido a través de tres elementos: el diagrama de casos de uso del negocio, la descripción de los casos de uso del negocio y el diagrama de actividades.

Un actor del negocio es cualquier individuo, grupo, entidad, organización, máquina o sistema de información externos; con los que el negocio interactúa. Lo que se modela como actor es el rol que se juega cuando se interactúa con el negocio para beneficiarse de sus resultados. [43]

Los actores del negocio se listan a continuación

Tabla 2: Actores del negocio

Actores	*Justificación*
Directivo	Solicita registros de los controles que se realizan en la entidad
Responsable de seguridad informática general (RSIG)	Solicita expedientes de equipos, registros de controles, registros de software instalados entre otros resultados de procesos en cuestión. Es el principal beneficiado con los resultados del negocio.

Diagramas de casos de uso del negocio

Para comprender los procesos de negocio se construye el diagrama de casos de uso del negocio en el que aparece cada proceso del negocio relacionado con su actor.

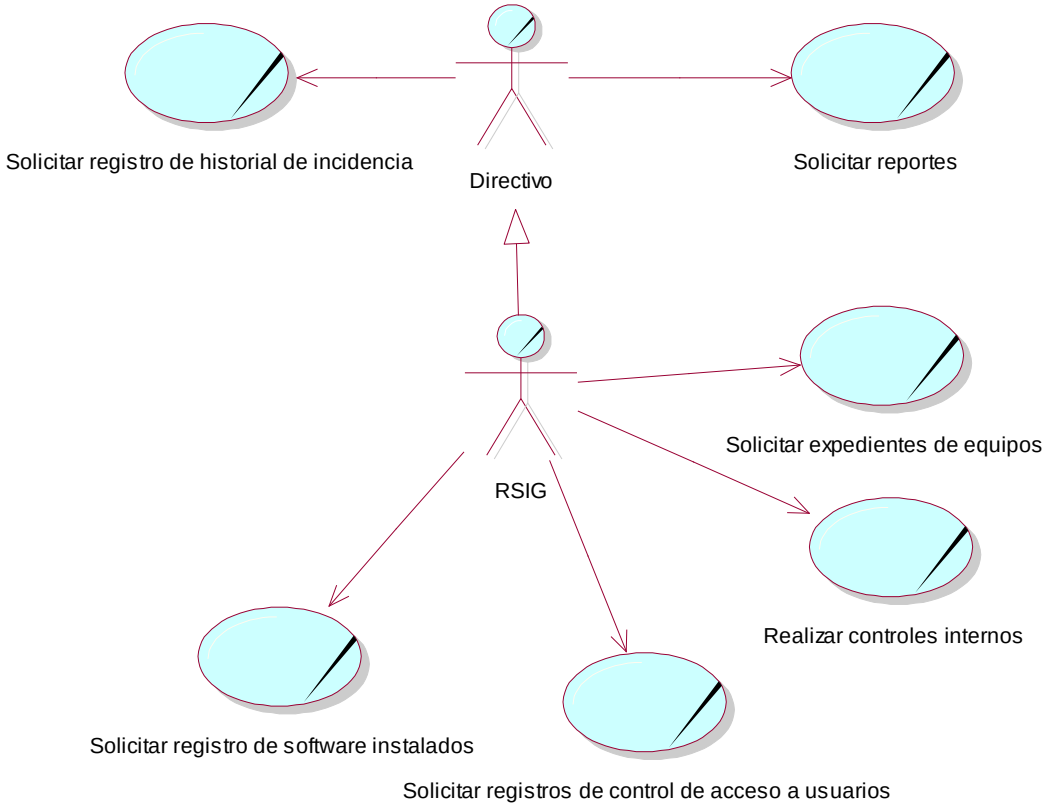


Figura 7: Diagrama de casos de uso del negocio

Trabajadores del negocio

Un trabajador del negocio es una abstracción de una persona (o grupo de personas), una máquina o un sistema automatizado; que actúa en el negocio realizando una o varias actividades, interactuando con otros trabajadores del negocio y manipulando entidades del negocio. Representa un rol. [43]

Los trabajadores del negocio se listan a continuación

Tabla 3: Trabajadores del negocio

Trabajadores	*Justificación*

Responsable de seguridad informática del área (RSIA)	Encargado de realizar los diferentes registros y controles que se llevan a cabo en el proceso del negocio
Informático General	Encargado de actualizar el registro de incidencias
Usuario	Es el responsable de las tecnologías, el responsable de realizar el control interno a la PC.

Realización de los casos de uso del negocio

Tabla 4: Descripción del Caso de Uso Solicitar reportes

Nombre del Caso de Uso		Solicitar reportes
Actores		Directivo (inicia) o RSIG(inicia)
Propósito	Brindar a los directivos de la empresa y al RSIG la posibilidad de obtener los diferentes reportes	
Resumen	El caso de uso inicia cuando un directivo solicita al RSIG el estado de las tecnologías o el estado del control interno realizado en las áreas o en la entidad en general.	
Curso Normal de los eventos		
Acciones del Actor		**Respuesta del proceso de negocio**
1. El directivo solicita el reporte del estado de las tecnologías o el estado del control interno		1.1 El RSIA busca el reporte y lo entrega
2. El directivo recibe el reporte		
Prioridad	Alta	
Mejoras	El proceso se realizará de forma más fácil, pues se podrán obtener los reportes más rápidamente y quedarán almacenados para posteriores revisiones de los mismos	

Tabla 5: Descripción del Caso de Uso Revisar expedientes de equipo

Nombre del Caso de Uso	Revisar expedientes de equipo

Actores		RSIG(inicia)
Propósito	Brindar al RSIG la posibilidad de inspeccionar el estado del equipo	
Resumen	El caso de uso inicia cuando el RSIG solicita al RSIA el expediente de equipo para su revisar el estado de las tecnologías y en caso de encontrar problemas lo refleja en el registro de incidencia y finaliza el caso de uso.	
Curso Normal de los eventos		
Acciones del Actor		**Respuesta del proceso de negocio**
1.El RSIG solicita al RSIA el expediente de equipos		1.1 El RSIA interroga por los equipos objetos de la revisión
2. El RSIG especifica los equipos		2.1 El RSIA busca el expediente (los) y lo entrega
3. El RSIG revisa el expediente		
Curso Alternativo de los eventos		
Acción 3	Si el RSIG encuentra anomalías en el expediente de equipo las refleja en el registro de incidencia	
Prioridad	Alta	
Mejoras	El proceso se realizará de forma más fácil, pues se podrán obtener los expedientes de los equipos deforma automática y en menor tiempo	

Tabla 6: Descripción del Caso de Uso Solicitar registro de control de acceso a usuarios

Nombre del Caso de Uso		Solicitar registro de control de acceso a usuarios
Actores		RSIG(inicia)
Propósito	Brindar al RSIG la posibilidad de tener el control de los usuarios que tienen acceso a las distintas PCs	
Resumen	El caso de uso inicia cuando el RSIG solicita al usuario la inspección de las tecnologías, este hace la revisión y en caso de detectar más de un usuario por máquina se refleja en el registro de incidencia, se le comunica al administrador de la red y finaliza el caso de uso.	
Curso Normal de los eventos		

Acciones del Actor	**Respuesta del proceso de negocio**
1. El RSIG solicita al responsable de la PC revisar los usuarios de la misma.	1.1 El usuario busca el registro de control de acceso a la PC
	1.2 El usuario compara los usuarios existentes con el registro.
3.El RSIG revisa el registro y todos lo usuarios están autorizados	
Curso Alternativo de los eventos	
Acción 2	SI el RSIG encuentra usuarios que no aparecen autorizados en el registro de control de acceso lo refleja en el registro de incidencia
Prioridad	Alta
Mejoras	El proceso se llevará a cabo de forma automatizada pues desde su puesto de trabajo el RSIG tendrá acceso a los usuarios autorizados en las computadoras y a los que realmente tienen cuentas en ese momento

Tabla 7: Descripción del Caso de Uso Solicitar registro de software instalado

Nombre del Caso de Uso	Solicitar registro de software instalado
Actores	RSIG(inicia)
Propósito	Brindar al RSIG la posibilidad de comparar los software instalados en las máquinas y los que están autorizados
Resumen	El caso de uso inicia cuando el RSIG inspecciona los software instalados en las computadoras y le solicita al responsable del equipo el registro de software autorizados en caso de anomalía se procede a reflejarla en el registro de incidencia y se le comunica al administrador de la red.
Curso Normal de los eventos	
Acciones del Actor	**Respuesta del proceso de negocio**
1.El RSIG solicita al responsable de la	1.1 El usuario busca el registro de

PC la revisión de los sistemas del equipo	software
	1.2 El usuario compara el registro con los software instalados
3.El RSIG revisa el registro y todos los sistemas están autorizados	
Curso Alternativo de los eventos	
Acción 2	Si el RSIG encuentra sistemas instalados que no se encuentran en el registro por lo que hace constar el problema en las incidencias e informa al administrador de red.
Prioridad	Alta
Mejoras	El proceso de revisión se llevará a cabo de forma automatizada disminuyendo el tiempo requerido para realizar la revisión

Tabla 8: Descripción del Caso de Uso Realizar controles internos

Nombre del Caso de Uso	Realizar controles internos	
Actores	RSIG(inicia)	
Propósito	Brindar al RSIG la posibilidad de obtener el resultado del control interno de las PC y conocer el estado en que se encuentra la misma	
Resumen	El caso de uso inicia cuando el RSIG solicita al usuario realizar el control interno a la PC con los puntos que va a auditar clasificándolos en varios aspectos: seguridad física, seguridad lógica y seguridad de red, anotando en este registro las deficiencias encontradas en el control y obtiene al final una evaluación, finalizando así el caso de uso	
Curso Normal de los eventos		
Acciones del Actor		**Respuesta del proceso de negocio**
1.El RSIG solicita al usuario la realización del control interno		1.1 El usuario recibe los puntos a auditar

	1.2 El usuario realiza el control interno a la PC
	1.3 El usuario obtiene el resultado y lo entrega al RSIG
3.El RSIG en caso de encontrar anomalías las refleja en el registro de control interno	
4. El RSIG realiza la evaluación	
Curso Alternativo de los eventos	
Acción 3	El RSIG en caso de no encontrar problemas no refleja nada en el registro y pasa al paso 4
Prioridad	Alta
Mejoras	El proceso de revisión se llevará a cabo de forma automatizada disminuyendo el tiempo requerido para realizar la revisión

Tabla 9: Descripción del Caso de Uso Solicitar el registro de incidencias

Nombre del Caso de Uso	Solicitar el registro de incidencias
Actores	Directivo(inicia)
Propósito	Brindar al directivo la posibilidad de revisar el historial de incidencias que han existido en la empresa
Resumen	El caso de uso inicia cuando el directivo solicita al RSIG el registro de incidencias de la unidad
Curso Normal de los eventos	
Acciones del Actor	**Respuesta del proceso de negocio**
1.El directivo decide revisar el historial de incidencias	
2. El directivo solicita el RSIG el registro de incidencias	2.1 El Informático General entrega el registro
Prioridad	Media
Mejoras	El directivo podrá revisar el historial de incidencia de forma automatizada

Diagramas de actividades

El diagrama de actividad es un grafo que contiene los estados en que puede hallarse la actividad a analizar. Cada estado de la actividad representa la ejecución de una sentencia de un procedimiento, o el funcionamiento de una actividad en un flujo de trabajo. En resumen describe un proceso que explora el orden de las actividades que logran los objetivos del negocio. [43]

En los Anexos del 3 al 8 se muestran los diagramas de actividades de los casos de uso del modelo del negocio.

Diagrama de clase del modelo de objetos

Un modelo de objetos del negocio es un modelo interno a un negocio. Describe como cada caso de uso del negocio es llevado a cabo por parte de un conjunto de trabajadores que utilizan un conjunto de entidades del negocio y unidades de trabajo. [43]

Una entidad del negocio representa algo, que los trabajadores toman, examinan, manipulan, crean o utilizan en un caso de uso del negocio. El diagrama de clases del modelo de objeto, es un artefacto que se construye para describir el modelo de objetos del negocio, que en este caso se muestra a continuación.

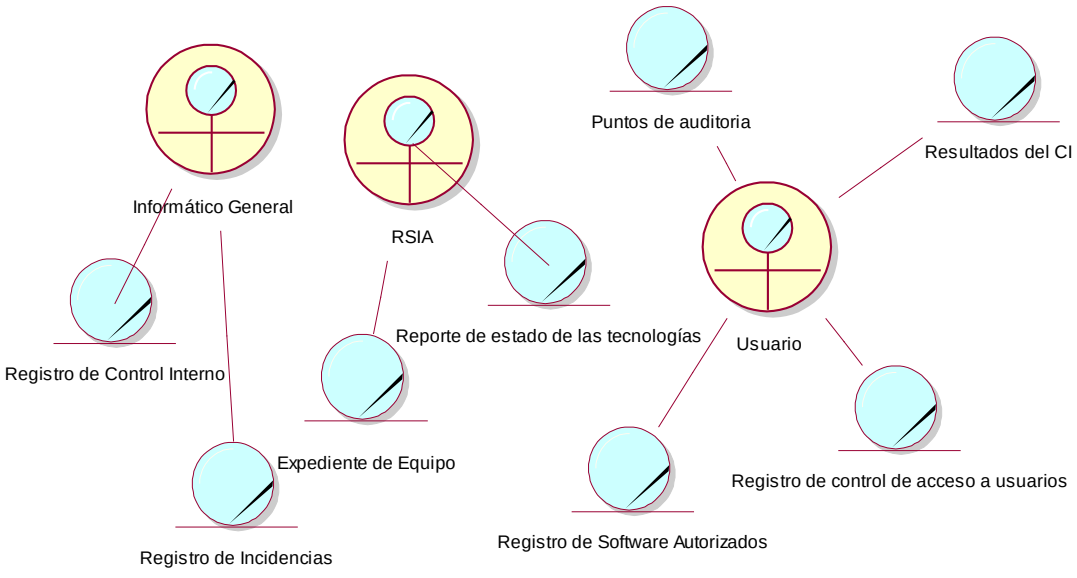


Figura 8: Diagrama de clases del modelo de objetos

III.3.2 Definición de los requerimientos

Requerimientos funcionales

Los requerimientos funcionales permiten expresar una especificación más detallada de las responsabilidades del sistema que se propone. Ellos permiten determinar, de una manera clara, lo que debe hacer el mismo. [43]

Los requerimientos funcionales del sistema propuestos son los siguientes:

1. Insertar usuario
2. Editar usuario
3. Eliminar usuario
4. Realizar control interno
5. Imprimir modelo de control interno
6. Insertar codificador de cargos
7. Insertar codificador de compañías
8. Insertar codificador de componentes
9. Insertar codificador de sistemas operativos
10. Insertar codificador de software autorizados
11. Insertar codificador de tipo de equipo
12. Insertar codificador de tipo de software
13. Crear criterios de cuestionario
14. Crear cuestionarios
15. Editar detalles de cuestionario
16. Visualizar detalles de cuestionario
17. Cambiar estado de cuestionario
18. Realizar búsqueda
19. Insertar datos de PC
20. Editar datos de PC
21. Ejecutar baja técnica
22. Insertar componente
23. Editar componente
24. Eliminar componente
25. Visualizar resultados de controles por microcomputadoras

26. Visualizar resultados por área
27. Visualizar historial de incidencias por PC
28. Actualizar historial de incidencias por PC
29. Imprimir historial de incidencias por PC
30. Visualizar bajas técnicas
31. Visualizar registro de acceso
32. Insertar registro de acceso
33. Imprimir registro de acceso
34. Visualizar expediente de equipo
35. Imprimir expediente de equipo
36. Visualizar registro tecnológico
37. Visualizar recomendaciones de controles
38. Visualizar registro de software autorizados
39. Insertar registro de software autorizados
40. Eliminar registro de software autorizados
41. Imprimir registro de software autorizados

Requisitos no funcionales

Los requerimientos no funcionales especifican cualidades, propiedades del sistema; como restricciones del entorno o de la implementación, rendimiento, dependencias de la plataforma, etc. [43]

Los requerimientos no funcionales del sistema propuesto son los siguientes.

Requisitos de interfaz.

El sistema debe tener una interfaz sencilla, amigable, muy legible y simple de usar, el producto debe ser autoritario e interactivo para que los usuarios se sientan confiados.

Se debe informar al usuario dónde está y qué puede hacer desde allí, al proporcionar señales de navegación que conduzcan al usuario hasta el contenido que desea y evitarle navegar a través de muchas áreas para ello. El contenido debe ser mostrado de manera comprensible para el usuario.

Requerimientos de Usabilidad

El sistema estará dirigido a RSI. El mismo será utilizado solo por usuarios registrados, Súper-Administrador y Administrador de Área. Si el usuario es Súper-Administrador tiene acceso a todos los módulos del sistema y el Administrador de Área tendrá acceso solo al módulo Registro de Controles. El sistema esta diseñado para ser utilizado por personas con mínimos conocimientos en el manejo de la computadora y el ambiente Web en sentido general, debido a que contará con una ayuda a fin de documentar al usuario en su utilización. La ejecución de los comandos debe ser posible por el uso del teclado u otros dispositivos de entrada como el Mouse.

Los mensajes de error deben ser reportados por la propia aplicación en la medida de las posibilidades y no por el Sistema Operativo. Los mensajes del sistema deben estar en el idioma apropiado (español).

Requerimientos de Seguridad

La información estará protegida contra accesos no autorizados utilizando mecanismos de autenticación y autorización que puedan garantizar el cumplimiento de esto: cuenta, contraseña y nivel de acceso, de manera que cada uno pueda tener disponible solamente las opciones relacionadas con su actividad y tenga datos de acceso propios, garantizando así, la confidencialidad. Se usarán mecanismos de encriptación de los datos que por cuestiones de seguridad no deben viajar al servidor en texto plano, como es el caso de las contraseñas. Se guardará encriptada esta información en la base de datos.

Se harán validaciones de la información tanto en el cliente como en el servidor.

Se crearán usuarios con diferentes niveles de acceso al sistema. Se limitarán los permisos de los usuarios que ejecutan sentencias o consultas SQL. Se utilizarán, además, procedimientos almacenados dado que el modo en que se pasan los parámetros, evita el uso de inyección de código SQL. No obstante, los usuarios accederán de manera rápida y operativa al sistema sin que los requerimientos de seguridad se conviertan en un retardo para ellos.

Requerimientos de Software

La aplicación debe poderse ejecutar en entornos Windows, Linux*, etc.* (Multiplataforma), para su ejecución del lado del servidor necesita MySQL como sistema gestor de base de datos y Apache como servidor Web, del lado del cliente cualquiera de los exploradores Web existentes en el mercado.

Requerimientos de Hardware

Se requiere de una computadora como servidor de base de datos con los requerimientos de hardware que necesita MySQL. Las terminales clientes solo requerirán de una computadora conectada a la red, para poder ejecutar los navegadores de Web al menos deben cumplir los requisitos mínimos (que requiera el navegador en cuestión)

III.3.3 Modelo del sistema

Descripción General del Modelo de Sistema

Modelo de Casos de Uso del sistema

El modelo de casos de uso permite que los desarrolladores del software y los clientes lleguen a un acuerdo sobre los requisitos, es decir, sobre las condiciones y posibilidades que debe cumplir el sistema. Describe lo que hace el sistema para cada tipo de usuario. [23]

Actores del sistema

Un actor no es más que un conjunto de roles que los usuarios de Casos de Uso desempeñan cuando interaccionan con estos Casos de Uso. Los actores representan a terceros fuera del sistema que colaboran con el mismo. Una vez que hemos identificado los actores del sistema, tenemos identificado el entorno externo del sistema [24].

Tabla 10: Actores del sistema

Actores	*Justificación*
Administrador	Es el encargado de gestionar la cuentas de los usuarios del sistema, realizar el control interno, gestionar lo codificadores y los cuestionarios, dar bajas técnicas a equipos y gestionar los componentes de los mismos

RSIA	Es el encargado de realizar búsquedas, gestionar los datos de las PC.

Casos de Uso del Sistema

La forma en que interactúa cada actor del sistema con el sistema se representa con un Caso de Uso. Los Casos de Uso son “fragmentos” de funcionalidad que el sistema ofrece para aportar un resultado de valor para sus actores. De manera más precisa, un Caso de Uso especifica una secuencia de acciones que el sistema puede llevar a cabo interactuando con sus actores, incluyendo alternativas dentro de la secuencia [15].

Atendiendo a las funcionalidades de los casos de uso del sistema estos se pueden agrupar en paquetes como se muestra a continuación:

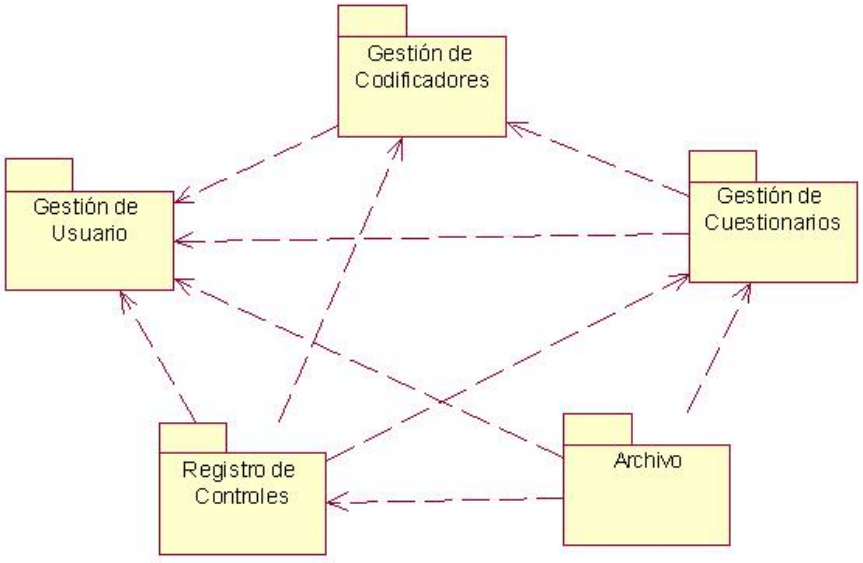


Figura 9: Diagrama de Casos de Uso por paquetes

Por la complejidad de los casos de uso del paquete Registros de Controles en este trabajo se realizará el análisis de los casos de uso que lo componen.

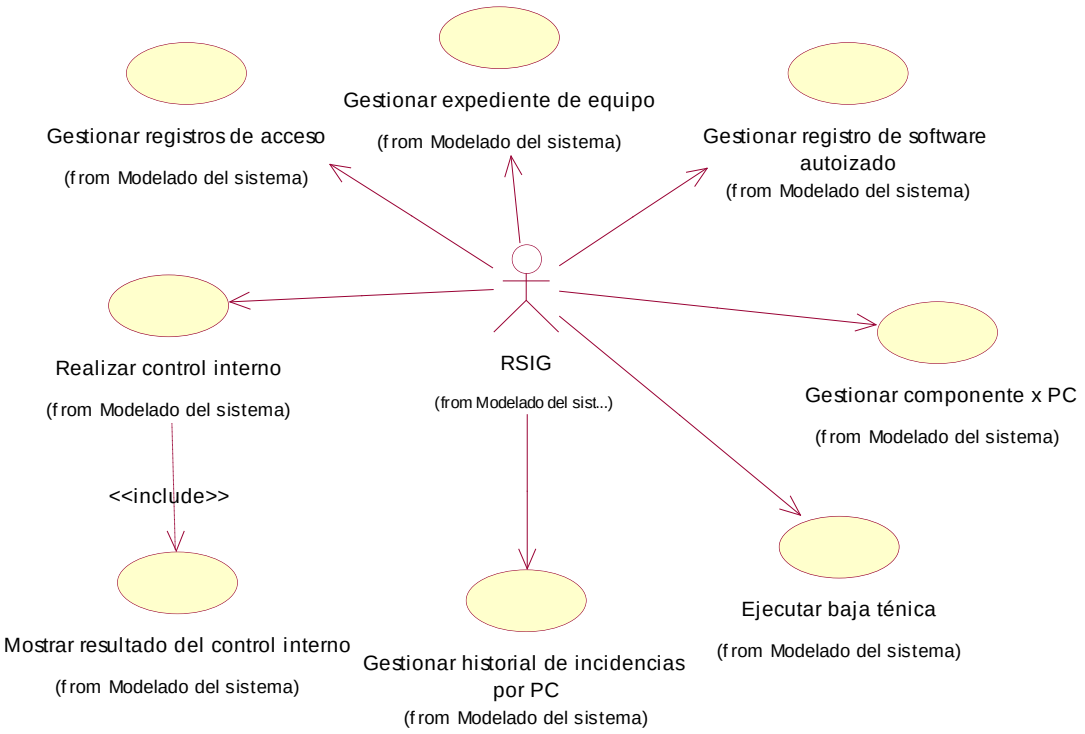


Figura 10: Diagrama de Casos de Uso del Paquete Registro de controles

Descripción de los casos de uso del paquete Registros de Controles

Tabla 11: Descripción del Caso de Uso Realizar control interno

Caso de uso	Realizar control interno
Actores	Administrador(inicia)
Propósito	Realizar el control interno a una PC y visualizar el resultado
Resumen: El caso de uso inicia cuando administrador desea realizar el control interno de una PC en un departamento de una compañía. El sistema permite evaluar los aspectos del control interno y mostrar e imprimir el resultado del mismo. El caso de uso finaliza con el control interno de una computadora realizado.	
Referencias	R4, R5, Mostrar resultado del control interno <include>
Precondiciones	Tiene que existir la computadora en el departamento seleccionado
Post-condiciones	Queda realizado el control interno en la computadora Si acción: imprimir se imprime el resultado del mismo

Prototipo	Ver Anexo 9

Tabla 12: Descripción del Caso de Uso Gestionar historial de incidencias

Caso de uso	Gestionar historial de incidencias por PC
Actores	Administrador(inicia)
Propósito	Insertar, editar y eliminar una incidencia del historial
Resumen: El caso de uso comienza cuando se desea consultar el historial de incidencia que tiene una computadora en un departamento de una compañía. El sistema permite añadir una incidencia, editarla y eliminarla. El caso de uso finaliza con el historial actualizado.	
Referencias	R26, R27, R28
Precondiciones	Tiene que existir la computadora en el departamento seleccionado
Post-condiciones	Se actualiza el historial de incidencias Si acción: insertar, se inserta una incidencia en el historial Si acción: editar, se modifica una incidencia en el historial Si acción: eliminar se elimina una incidencia del historial
Prototipo	Ver Anexo 10

Tabla 13: Descripción del Caso de Uso Gestionar registro de control de acceso

Caso de uso	Gestionar registro de control de acceso
Actores	Administrador(inicia)
Propósito	Insertar, editar y eliminar un acceso en el registro
Resumen: El caso de uso comienza cuando el administrador desea consultar el registro de control de acceso a una computadora en un departamento de una compañía. El sistema permite insertar, editar y eliminar un acceso. El caso de uso finaliza con el registro actualizado.	
Referencias	R30, R31, R32
Precondiciones	Tiene que existir la computadora en el departamento seleccionado
Post-condiciones	Se actualiza el registro de control de acceso Si acción: insertar se inserta un acceso en el registro

	Si acción: editar se modifica un acceso en el registro Si acción: eliminar se elimina un acceso en el registro
Prototipo	Ver Anexo 11

Tabla 14: Descripción del Caso de Uso Visualizar expediente de equipo

Caso de uso	Gestionar expediente de equipo
Actores	Administrador(inicia)
Propósito	Visualizar el expediente de un equipo
Resumen: El caso de uso comienza cuando el administrador desea consultar el expediente de un equipo de un departamento de una compañía. El sistema permite imprimir el expediente.	
Referencias	R33, R34, R35
Precondiciones	Tiene que existir la computadora en el departamento seleccionado
Post-condiciones	Se visualiza el expediente de la PC Si acción: imprimir se imprime el expediente
Prototipo	Ver Anexo 12

Tabla 15: Descripción del Caso de Uso Gestionar registro de software autorizado

Caso de uso	Gestionar registro de software autorizado
Actores	Administrador(inicia)
Propósito	Insertar y eliminar un software del registro
Resumen: El caso de uso comienza cuando el administrador desea consultar el registro de software autorizado para una PC del un departamento de una compañía. El sistema permite imprimirle registro e insertar o eliminar software del mismo. El caso de uso finaliza con la actualización del registro de software autorizado.	
Referencias	R38, R39, R40, R41
Precondiciones	Tiene que existir la computadora en el departamento seleccionado
Post-condiciones	Se visualiza el expediente de la PC Si acción: imprimir, se imprime el registro

	Si acción: insertar, se inserta un software Si acción: eliminar, se elimina un software
Prototipo	Ver Anexo 13

Tabla 16: Descripción del Caso de Uso Gestionar componente de PC

Caso de uso	Gestionar componente x PC
Actores	Administrador(inicia)
Propósito	Insertar y eliminar los componentes de una PC
Resumen: El caso de uso inicia cuando el administrador desea consultar los componentes de una PC de un departamento de una compañía. El sistema permite que se pueda adicionar componentes, eliminarlo o editarlo. El caso de uso termina con la actualización de todos los componentes de la PC	
Referencias	R21, R22, R23
Precondiciones	Tiene que existir la computadora en el departamento seleccionado
Post-condiciones	Se actualizan los componentes a visualizar Si acción: insertar, se insertan los componentes Si acción: eliminar, se eliminan los componentes de la PC Si acción: editar, se modifican los componentes
Prototipo	Ver Anexo 14

Tabla 17: Descripción del Caso de Uso Dar Baja técnica

Caso de uso	Ejecutar Baja técnica
Actores	Administrador(inicia)
Propósito	Cambiar el estado del equipo a baja técnica
Resumen: El caso de uso inicia cuando el administrador necesita dar la baja técnica a un equipo en un departamento de una compañía. El sistema permite cambiar el estado del equipo a baja técnica.	
Referencias	R20
Precondiciones	Tiene que existir la computadora en el departamento seleccionado
Post-condiciones	Queda actualizado el estado del equipo

Prototipo	Ver Anexo 15

III.3.4 Diseño e implementación del sistema

Principios de Diseño del Sistema

El diseño de sistemas se define como el proceso de aplicar ciertas técnicas y principios con el propósito de definir un dispositivo, un proceso o un sistema, con suficientes detalles como para permitir su interpretación y realización física.

Modelo de Clases Web

Un diagrama de clases Web representa las colaboraciones que ocurren entre las páginas, donde cada página lógica puede ser representada como una clase. Al tratar de utilizar el diagrama de clases tradicional para modelar aplicaciones Web surgen varios problemas, por lo cual los especialistas del Rational plantearon la creación de una extensión al modelo de análisis y diseño que permitiera representar el nivel de abstracción adecuado y la relación con los restantes artefactos de UML.

Tabla 18: Diagramas de clases web: Paquete Registro de controles

Casos de Uso	*Diagrama de Clases WEB*
Realizar control interno	Ver Anexo 16
Gestionar historial de incidencias	Ver Anexo 17
Gestionar registro de control de acceso	Ver Anexo 18
Gestionar expediente de equipo	Ver Anexo 19
Gestionar registro de software autorizado	Ver Anexo 20
Gestionar componente x PC	Ver Anexo 21
Ejecutar baja técnica	Ver Anexo 22

Diseño de la Base de datos

Por la importancia de los datos manejados en el módulo Registro de controles de la gerencia de Copextel es necesario lograr un buen diseño de la información almacenada.

A continuación se muestra el diseño de la base de datos del sistema propuesto a través de los diagramas de clases persistente y el esquema de la base de datos generados a partir de este, con el modelo de datos.

El diagrama de clases persistentes muestra todas las clases capaces de mantener su valor en el espacio y en el tiempo. Se muestra en el Anexo 23.

El modelo de datos que muestra la estructura física de las tablas de la base de datos, obtenido a partir del diagrama de clases persistentes, es mostrado en el Anexo 24.

El modelo de implementación describe cómo los elementos del modelo de diseño se implementan en términos de componentes. Describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados y cómo dependen los componentes unos de otros [43]. Este modelo se muestra en el Anexo 25

III.4 Aplicación de Scrum

Para comprobar eficientemente la efectividad de la metodología Scrum se decide aplicar dicha metodología al Módulo de Registro de Controles que responde a las condiciones necesarias para utilizar una metodología ágil. En el Anexo 26 se muestra una guía de los pasos a seguir para aplicar eficientemente dicha metodología.

El equipo de trabajo está conformado por 3 miembros, un diseñador y dos desarrolladores.

Cuando se trabaja con la metodología Scrum primeramente se ordena el Product Backlog que es el inventario de funcionalidades, mejoras, tecnología y corrección de errores que deben incorporarse al producto a través de las sucesivas iteraciones de desarrollo.

Representa todo aquello que esperan los clientes, usuarios, y en general los interesados en el producto. Todo lo que suponga un trabajo que debe realizar el equipo tiene que estar reflejado en el backlog.

Tabla 19: Product Backlog

Id	*Descripción*	*Nro It*	*Módulo del Sistema*
1	Realizar el control interno de una computadora	2	Registro de controles
2	Obtener resultado del control interno	2	Registro de controles
3	Dar baja técnica a un equipo	1	Registro de controles
4	Añadir un componente a un equipo	1	Registro de controles
5	Modificar un componente de un equipo	1	Registro de controles
6	Eliminar un componente de un equipo	1	Registro de controles
7	Visualizar el historial de incidencias	3	Registro de controles
8	Actualizar el historial de incidencias	3	Registro de controles
9	Imprimir el historial de incidencias	3	Registro de controles
10	Visualizar registro de control de acceso a un equipo	3	Registro de controles
11	Insertar registro de control de acceso a un equipo	3	Registro de controles
12	Imprimir registro de control de acceso a un equipo	3	Registro de controles
13	Visualizar expediente de un equipo	1	Registro de controles
14	Imprimir expediente de un equipo	1	Registro de controles

Luego en la reunión de planificación del Sprint se define el Sprint Backlog que es la lista que descompone las funcionalidades del Product Backlog en las tareas necesarias para construir un incremento: una parte completa y operativa del producto.

En el Sprint Backlog se asigna a cada tarea la persona que la va a llevar a cabo, y se indica el tiempo de trabajo que se estima, aún falta para terminarla.

Tabla 20: Sprint Backlog de la iteración 1

Requisito	*Tarea*	*Persona asignada*
3	Dar baja técnica a un equipo	Desarrollador
4	Añadir un componente a un equipo	Desarrollador
5	Modificar un componente de un equipo	Desarrollador
6	Eliminar un componente de un equipo	Desarrollador
13	Visualizar expediente de un equipo	Desarrollador
14	Imprimir expediente de un equipo	Desarrollador

En el transcurso de la iteración se realizan reuniones de monitorización de Sprint en la que los miembros expresan las tareas en las que están trabajando, si han encontrado o prevén encontrarse algún impedimento y actualizan sobre el Sprint Backlog las tareas ya terminadas o los tiempos de trabajo que le queda.

Utilizando la herramienta Sprintometer se da seguimiento al proyecto y al trabajo que llevan a cabo los integrantes del equipo. Este software brinda la posibilidad de obtener los gráficos burndown que permiten ver el estado en que se encuentra el proyecto, la velocidad del equipo de trabajo, el trabajo que falta por realizar y el que se ha completado.

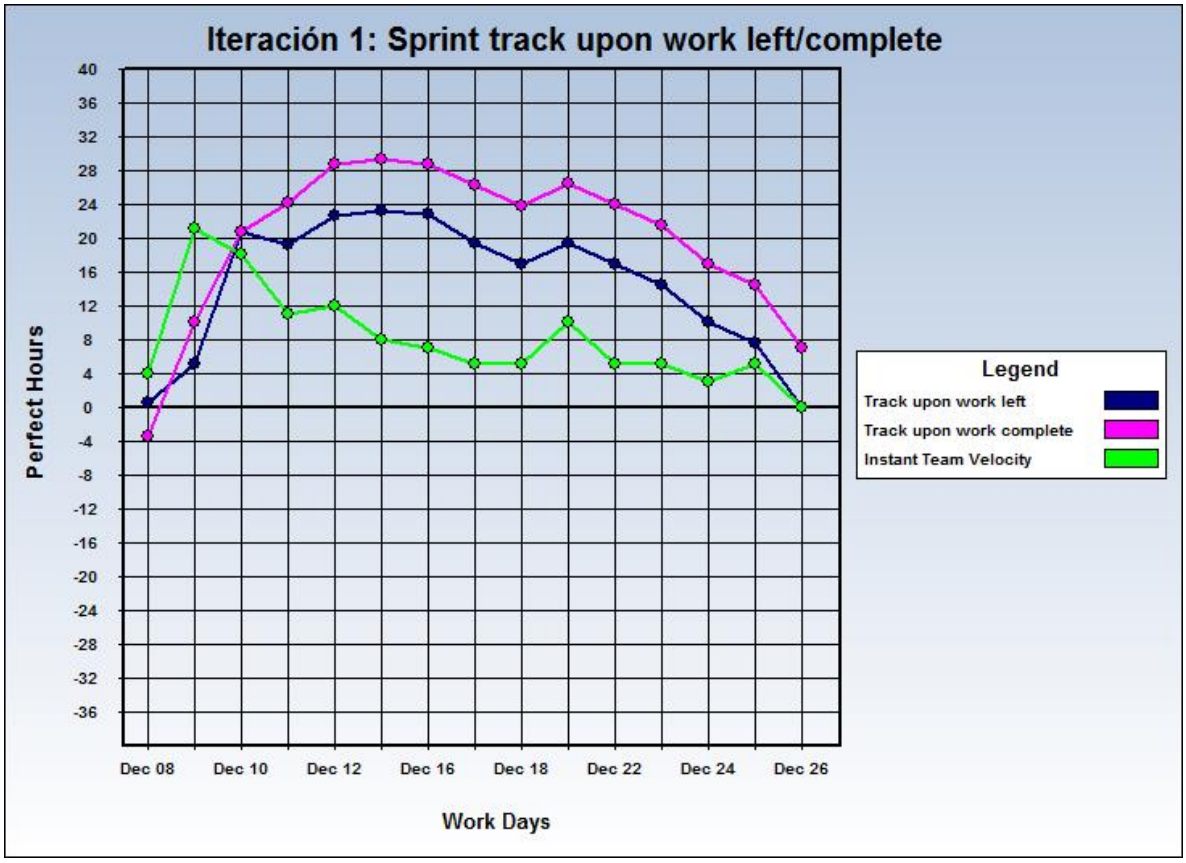


Figura 11: Gráfico del curso de la iteración 1

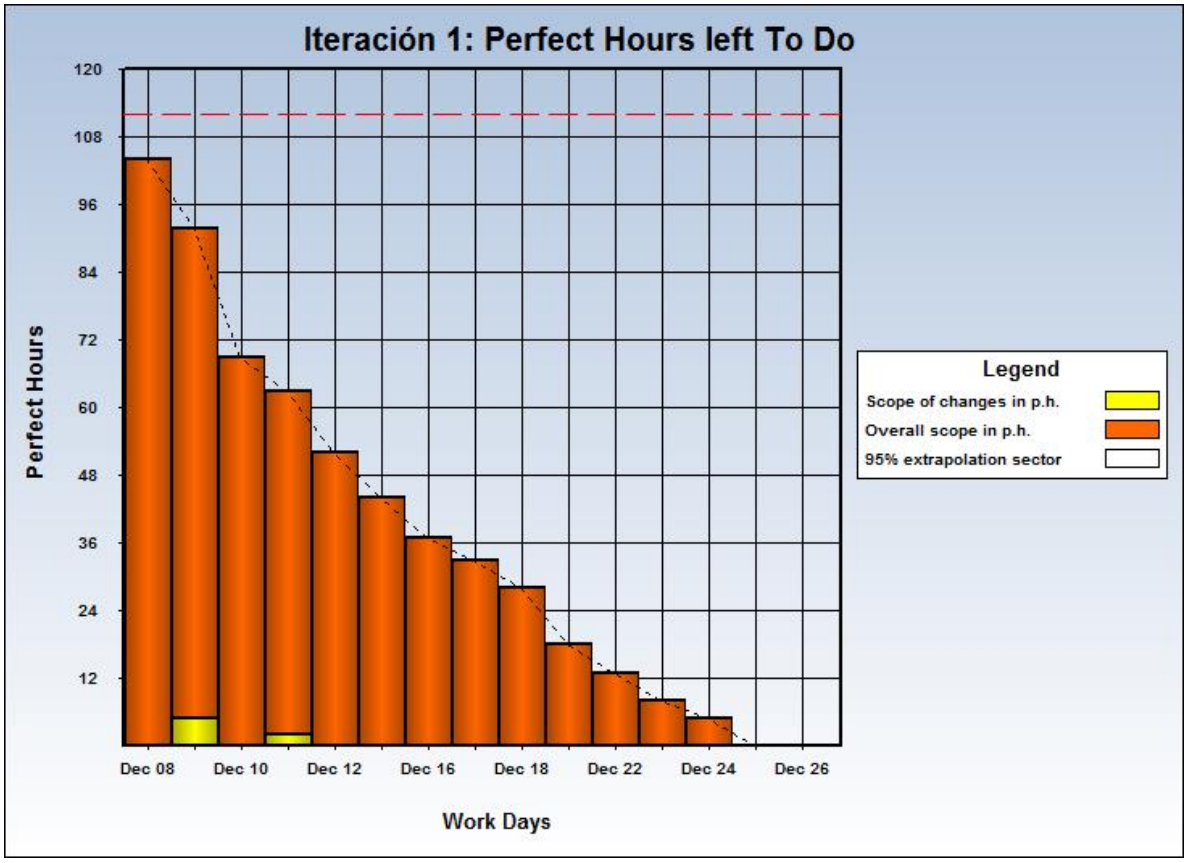


Figura 12: Gráfico de alcance de la iteración 1

Cuando finaliza la iteración se lleva a cabo la reunión de revisión del Sprint donde el equipo presenta al cliente la parte del producto desarrollado en el transcurso de la iteración. En esta reunión se define también las tareas que van a integrar la próxima iteración.

Tabla 21: Sprint Backlog de la iteración 2

Requisito	*Tarea*	*Persona asignada*
1	Realizar el control interno de una computadora	Desarrollador
2	Obtener resultado del control interno	Desarrollador

Para ver los gráficos de las iteraciones 2 y 3 ver los Anexos 27, 28, 29 y 30.

Tabla 22: Sprint Backlog de la iteración 3

Requisito	*Tarea*	*Persona asignada*

7	Visualizar el historial de incidencias	Desarrollador
8	Actualizar el historial de incidencias	Desarrollador
9	Imprimir el historial de incidencias	Desarrollador
10	Visualizar registro de control de acceso a un equipo	Desarrollador
11	Insertar registro de control de acceso a un equipo	Desarrollador
12	Imprimir registro de control de acceso a un equipo	Desarrollador

III.5 Resultados

Al inicio del desarrollo del Sistema de Gestión de Información del Control Interno Informático (SGICII) fue estimado el tiempo de culminación del proyecto y el costo de producción. Se planificó concluir en 12 meses el proyecto. Para comprobar la diferencia de la aplicación de las dos metodologías, Scrum y RUP, ambas se aplicaron al módulo Registro de Controles, obteniéndose como resultado que trabajando un diseñador y dos desarrolladores, 6 horas diarias con RUP se tardó 2 meses en concluir la primera versión del módulo, mientras que utilizando Scrum con la misma cantidad de personas y el mismo tiempo de trabajo se finalizó en 1 mes y medio, fundamentalmente debido a la disminución del tiempo de análisis pues esta metodología no genera gran cantidad de documentación y además es un proceso más flexible, lo cual evidencia que en los casos donde la complejidad del proceso es baja es más factible usar la metodología ágil en lugar de una tradicional.

III.6 Conclusiones del capítulo

En este capítulo se aplicaron las metodologías RUP y Scrum al Proyecto Sistema de Gestión de Información del Control Interno Informático que se desarrolla en Copextel, para ello se utilizaron las herramientas Rational Rose y Sprintometer.

En el caso de RUP se aplicó siguiendo los flujos de trabajo de esta metodología, se definieron actores, trabajadores y casos de uso del negocio y se llevó a cabo el análisis, diseño e implementación del sistema.

En el caso de Scrum se definió el Product Backlog y los Sprint Backlog de cada una de las iteraciones. Se hizo un seguimiento del proyecto utilizando los gráficos del curso y alcance de las iteraciones.

Conclusiones Generales

En el desarrollo de este trabajo se llevó a cabo un estudio de las metodologías de desarrollo existentes y especialmente para aplicaciones web, se describieron las mismas, quedando plasmado sus características principales sus beneficios y desventajas.

Como resultado de la investigación realizada se concluye que no existe una metodología universal para hacer frente con éxito a cualquier proyecto de desarrollo de software. Cada equipo de trabajo debe escoger la metodología que mejor se adapte a sus características y las del proyecto.

Las metodologías tradicionales juegan un importante papel en la realización de grandes proyectos donde se exige un alto grado de ceremonia del proceso, en los cuales es crucial esclarecer bien las necesidades del cliente y las características deseadas del sistema que ayuden a comprender mejor el mismo. Sin embargo cuando se trabaja en proyectos pequeños, este esquema tradicional resulta poco práctico pues se pierde mucho tiempo en realizar tareas que no apoyan en gran medida la construcción de la aplicación. Es por ello que muchos desarrolladores en la actualidad se inclinan por utilizar las metodologías ágiles, pero es válido puntualizar que cuando se trata de proyectos de mediano tamaño resulta de gran ayuda determinar las características del problema que se va a automatizar, es decir, modelar el negocio objeto de estudio para dejar plasmado las particularidades del problema a resolver pues en la mayoría de las empresas los procesos que se pretenden automatizar no cuentan con una documentación formal de estos, y regularmente se llevan a cabo, con base en lo que el personal involucrado con más experiencia establece a su buen juicio.

Por todo lo antes expuesto se seleccionó para aplicar en la construcción de aplicaciones web en la Universidad de Cienfuegos la metodología de desarrollo ágil Scrum y la metodología tradicional RUP, esta propuesta fue validada aplicándose al proyecto Sistema de Gestión de Información del Control Interno Informático, lográndose disminuir el tiempo de entrega de este.

Recomendaciones

A pesar de que los objetivos trazados con la realización de este trabajo fueron cumplidos, la autora del mismo sugiere tomar esta propuesta solo como la primera fase de un estudio mucho más ambicioso y continuar esta investigación.

Se recomienda entonces:

- Continuar el estudio de las metodologías y proponer metodologías de desarrollo de otros tipos de sistemas que se construyen en la Universidad de Cienfuegos
- Aplicar las metodologías propuestas al desarrollo de aplicaciones web de la Universidad de Cienfuegos.
- Desarrollar una propuesta metodológica que se adapte a proyectos de mediano tamaño, que incluya modelado del negocio y principios ágiles.

Referencias Bibliográficas

[1] Mas Camacho, María Rosa y Febles Rodríguez, Juan Pedro, “Experiencias de la aplicación de la ingeniería de software en sistemas de gestión,” *Revista Cubana de Informática Médica*; www.cecam.sld.cu/pages/rcim/revista_1/articulos_pdf/r0100a01.pdf.

[2] Alvarez, Juana, “Ingeniería de Software,” Jul. 2007; http://www.educando.edu.do/educanblog/index.php?blogId=435 .

[3] Crawford Labrin Broderick, “Métodos Ágiles: Enfatizando el Tratamiento del Conocimiento Tácito sobre el Explícito,” 2005; www.frcu.utn.edu.ar/deptos/depto_3/34JAIIO/34JAIIO/asse/asse09.pdf.

[4] Flores, Antonio, “Historia Ingeniería de Software”; http://www.um.es/docencia/barzana/IAGP/IAGP2-Ingenieria-software-introduccion.html .

[5] Herrera Urive, Eliécer y Valencia Ayala, Luz Estela, “Del Manifiesto Ágil sus Valores y Principios,” *Scientia et Technica*, May. 2007; http://redalyc.uaemex.mx/redalyc/pdf/849/84934064.pdf.

[6] Gacitúa Bustos, Ricardo A, “Métodos de desarrollo de software: El desafío pendiente de la estandarisación,” 2003.

[7] Gabardini, Juan y Campos, Lucas, “Balanceo de Metodologías Orientadas al Plan y Ágiles. Herramientas para la Selección y Adaptación”; http://www.rmya.com.ar/Download/Paper_BMOPA.pdf.

[8] Díaz, María Irma, “La incertidumbre y la ingeniería de software”; http://www.acis.org.co/fileadmin/Revista_102/dos.pdf.

[9] Mendinilla, N. , “En busca de respuestas para la ingeniería de software,” 2005; http://is.ls.fi.upm.es/udis/docencia/proyecto/ docs/FilosofiaIS.pdf.

[10] Canós José H., Letelier Patricio, y Penadés, Mª Carmen , “Métodologías Ágiles en el Desarrollo de Software”; http://www.willydev.net/descargas/prev/TodoAgil.pdf.

[11] Orjuela Duarte, Ailin y Rojas, Mauricio, “Las Metodologías de Desarrollo Ágil como una Oportunidad para la Ingeniería de Software Educativo,” 2008; http://pisis.unalmed.edu.co/avances/archivos/ediciones/Edicion%20Avances%202008%202/21.pdf.

[12] Figueroa, Roberth G. y Solís, Camilo J, "Metodologías Tradicionales Vs. Metodologías Ágiles."

[13] Pérez Sánchez, Jesús , "Metodologías ágiles: la ventaja competitiva de estar preparado para tomar decisiones los mas tarde posible y cambiarlas en cualquier momento"; http://www.willydev.net/descargas/prev/metodologiasagiles.pdf.

[14] Letelier, Patricio y Penadés, Mª Carmen, "Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP)"; www.willydev.net/descargas/masyxp.pdf .

[15] Cáceres, Paloma y Marcos, Esperanza, "Procesos Ágiles para el Desarrollo de Aplicaciones Web"; http://www.dlsi.ua.es/webe01/articulos/s112.pdf .

[16] Conallen, J, *Building Web applications with UML*, Addison Wesley, 2000.

[17] Rayas, L. y Abreu, J. L., "Modelo para la exposición de la materia de ingeniería de software I," *International Journal of Good Conscience.*, vol. 3, Mar. 2008.

[18] Delgado Expósito, Erly, "Metodologías de desarrollo de software. ¿Cuál es el camino?
, *Revista de Arquitectura e Ingeniería*; www.empai-matanzas.co.cu/revista%20EMPAI/REVISTA3/articulo3.htm .

[19] Sosa López, Daylin y Hector Ortiz, Kadir, "Metodologías de Desarrollo de Software."

[20] Díaz Antón, Maria Gabriela y Angélica Pérez, Maria, "Propuesta de una metodología de desarrollo de software educativo bajo un enfoque de calidad sistemática"; http://www.academia-interactiva.com/ise.fdf .

[21] Mendoza Sánchez, María A, "Metodologías de Desarrollo de Software," Jun. 2004; http://www.willidev.net/InsiteCreation/v1.0/descargas/cualmetodología.pdf.

[22] "Disciplina para la Administración de Proyectos MSF"; http://www.planetacursos.com/curso/MjkSMA.

[23] "Introducción al modelo Scrum para desarrollo de Software"; http://es-es.start2.mozilla.com/firefox?client=firefox-a&rls=org.mozilla:es-ES:official.

[24] Beck, K, *Extreme Programming Explained. Embrace Change*, Pearson Education, 1999.

[25] de San Martin Oliva, Carla Rebeca Patricia, “Uso de la metodología ICONIX”; http://www.unsj-cuim.edu.ar/portalzonda/seminario08/archivos/UsodeICONIX.pdf .

[26]Calderón, Amaro et al., “Metodologías ágiles”; http://www.seccperu.org/files/Metodologías%Agiles.pdf.

[27]Letelier, Patricio, “Proceso de desarrollo de software”; https://pid.dsic.upv.es/C1/Material/Documentos%2520Disponibles/Introducci%C3%B3n%2520Proceso%2520de%2520Desarrollo%2520de%2520SW.doc.

[28] Shenone Marcelo Hernán, “Diseño de una Metodología Ágil de Desarrollo de Software”; http://materias.fi.uba.ar/7500/shenone-tesisdegrado.pdf.

[29]“the Agil Unified Process Home Page”; http://www.ambysoft.com/unifiedprocess/agilUP.html.

[30]“DSDM Consortium”; http://www.dsdm.org.

[31]“XBreed”; http://agile.csc.ncsu.edu/xbreed.html.

[32]“Lean Development”; http://agile.csc.ncsu.edu/lean.html.

[33]“Win-Win Spiral Model ”; http://agile.csc.ncsu.edu/winwinspiral.html.

[34] Cáceres, Paloma y Marcos, Esperanza, “Hacia un proceso metodológico dirigido por modelos para el desarrollo ágil de sistemas de información Web,” 2003; http://www.willydev.net/InsiteCreation/v1.0/descargas/prev/agilweb.pdf.

[35]“Cápitulo 6 Criterios de Usabilidad”; http://www.tdr.cesca.es/TESIS_UPC/AVAILABLE/TDX-0609104-120415//07Jctp7de20.pdf.

[36] Mingenz Sans, Daniel y García Morales, Emilio José, “Metodologías para el Desarrollo de Aplicaciones Web: UWE”; http://www.eici.ucm.cl/academicos/ygomez/descargas/Ing-Sw2/apuntes/DASBDMetodolog-UWE.pdf.

[37] Shewabw, Daniel y Rossi, Gustavo, “The Object-Oriented Hypermedia Design Model”; http://www.telemidia.puc-rio.br/oohdm/oohdm.html.

[38] Barreiro Alonso, Enrique, “Otros conceptos de la ingeniería del software.”

[39] Cañadas, José Joaquín, “Tema 1. Introducción a los Métodos Formales en Ingeniería del Software”; www.ual.es/~jjcanada/mfis/mfis03_04/teoria/Clase1.pdf.

[40] Passerini, Nicolás y Brey, Gustavo A., "Metodologías Iterativas de Desarrollo," 2005; http://apit.wdfiles.com/local--files/start/01_apit_metodologias.pdf.

[41] "Las Personas en las Metodologías de Ingeniería del Software"; http://www26.brinkster.com/lwelicki/articles/PersonasMetodologiaJIS04.pdf .

[42] Daniele, Marcela, "Análisis y Diseño de Sistemas (3303)," 2007; http://dc.exa.unrc.edu.ar/nuevodc/materias/sistemas/2007/TEORICOS/TEORIA_1_Introduccion_AyDS2007.pdf.

[43] Jacobson, I, Rumbaugh, J., y Booch, G. , *El Proceso Unificado de Desarrollo del Software*, Addison-Wesley, 2000.

[44] Lowe, D. y Hall, W., *Hypermedia & the Web. An Engineering Approach*, Wiley and Sons, .

[45] Fowler, M, Beck, K., y Brant, J., *Refactoring: Improving the Design of Existing Code*, Addison-Wesley, 1999.

[46] Kruchten Philippe, *The Rational Unified Process: an introduction*, Addison-Wesley, 2000.

[47] Overmyer, S. P., *What's Different about Requirements Engineering for Web Sites? Requirements Engineering*, London: Springer-Verlag, ; http://es-es.start2.mozilla.com/firefox?client=firefox-a&rls=org.mozilla:es-ES:official.

[48] "Proyectos Ágiles.org como gestionar proyectos con Scrum"; http://www.proyectosagiles.org/beneficios-de-scrum.

[49] "Rational Rose"; http://www.rational.com.

[50] "Sprintometer User Guide"; http://sprintometer.com.

Bibliografía

[1] V. Rodríguez Montequín, “Adaptación De Las Metodologías De Análisis Y Diseño Tradicionales Para El Desarrollo De Proyectos En Internet”; www.aeipro.com/congresos/2000_1/pdf/FB02.pdf.

[2] Schwaber K., Beedle M., y Martin R.C., “Agile Software Development with SCRUM,” 2001; http://es-es.start2.mozilla.com/firefox?client=firefox-a&rls=org.mozilla:es-ES:official.

[3] Vigil Regalado, Yamila y Fouces Cabana, Erich, “Análisis De La Necesidad De Una Metodología De Evaluación De Arquitectura De Software”; www.gestiopolis.com/administracion-estrategia/arquitectura-de-sofware-como-disciplina-cientifica.htm.

[4]Rueda Chacón, Julio César, “Aplicación De La Metodología Rup Para El Desarrollo Rápido De Aplicaciones Basado En El Estándar J2EE,” Mar. 2006; http://biblioteca.usac.edu.gt/tesis/08/08_7691.pdf.

[5] Silva, Darío Andrés y Mercerat, Bárbara, “Construyendo aplicaciones web con una metodología de diseño orientada a objetos,” 2001; www.lifia.info.unlp.edu.ar/papers/2001/Silva2001.pdf .

[6] López Villatoro, Marco René, “Desarrollo de software utilizando proceso unificado y extreme programming , Ene. 2009; http://www.revistaciencia.info/papers/v01n01_02.pdf.

[7] Goncalves, Matias, “Desarrollo de un Nuevo Modelo de Estimación Basado en Metodología Agil de Desarrollo y Generadores de Aplicaciones,” 2005; http://www.i-sol.com.ar/plan_de_tesis_final.pdf.

[8] Ferrá Grau, Xavier, “Desarrollo orientado a objetos con UML”; http://www.clikear.com/manuales/uml/introduccion.asp.

[9] Torres Flores, Carmina Lizeth, y Alférez Salinas, Germán Harvey, “Establecimiento de una Metodología de Desarrollo de Software para la Universidad de Navojoa Usando OpenUP”; http://fit.um.edu.mx/departamentodeinvestigacion/publicaciones/COMP-004-2008%20Establecimiento%20de%20una%20Metodolog%C3%ADa%20de%20Desarrollo%20de%20Software%20para%20la%20Universidad%20de%20Navojoa%20Usando%20OpenUP.pdf .

[10] Palacio Juan, “Flexibilidad con Scrum .”

[11] Charalambos, Jean Pierre y Martínez, José Andrés, “GAC: Una metodología para

la creación de sitios web de contenido dinámico," Ago. 2005; http://www.scielo.org.co/pdf/iei/v25n2/v25n2a05.pdf .

[12]Palacio, Juan, "Gestión de proyectos ágil: conceptos básicos," 2006; www.navegapolis.net/files/s/NST-003_01.pdf.

[13]Alejandro Martínez y Raúl Martínez, "Guía a Rational Unified Process"; http://www.info-ab.uclm.es/asignaturas/42551/trabajosAnteriores/Trabajo-Guia%20RUP.pdf.

[14] Moreno Cadavid, Julián, Velásquez Henao, Juan David, y Ovalle Carranza, Demetrio, "Hacia una Metodología para la Construcción de Modelos de Simulación Basados en Sistemas Multi-Agente," Nov. 2005; http://pisis.unalmed.edu.co/avances/archivos/ediciones/2005/moreno_etal05.pdf.....moreno_etal05.

[15]Meza Martínez, Jorge Iván, "Introducción a la implementación de Scrum"; http://www.jorgeivanmeza.com/.

[16]"Introducción al modelo Scrum de desarrollo de Software"; http://www.navegapolis.net/files/s/NST-010_01.pdf.

[17]Marcos, Esperanza, "Investigación en Ingeniería del Software vs. Desarrollo Software"; http://www.ciencias.holguin.cu/2008/Abril/articulos/ARTI1.htm.

[18]Medín, Cristian Fernando, "Investigación exploratoria de metodologías de desarrollo ágiles"; http://www.cedis-it.com.ar/images/Documentos_base/est2008_metodologiasagiles.pdf.

[19] Díaz, Luis Carlos, Carrillo, Angela y Alvarado, Deicy, "IS, RUP y UML en el Contexto de ADOO"; http://sophia.javeriana.edu.co/~lcdiaz/ADOO2008-1/IngSoftwareEnADOO(IS-RUP-UML).pdf.

[20] De Luca J., Coad P, y Lefebvre E, "Java Modeling In Color With UML: Enterprise Components and Process," 1999.

[21] Zulueta Véliz, Yeleny, Hernández Alba, Yailien, y Fernández Pérez, Leonel Duvier, "La Gestión De Riesgos en las Metodologías de Desarrollo de Software ."

[22] Aguilar Sierra, Alejandro, "Las Metodologías Ágiles en la Enseñanza de la Ingeniería de Software," Sep. 2003.

[23]"Medición: Usos y herramientas"; http://www.scrummanager.net.

[24]“Metodología ICONIX”; www.unsj-cuim.edu.ar/portalzonda/seminario08/archivos/MetodologiaICONIX.pdf.

[25] Letelier, Patricio, “Metodologías Ágiles para el Desarrollo y Pruebas del Software ; www.redit.es/pdfs/Jornada%2520sobre%2520Testeo%2520de%2520Software%2520(8%2520y%25209%2520de%2520mayo).pdf.

[26] Fernández y Fernández, Carlos Alberto, “Modelado Visual con UML”; http://www.utm.mx/~caff/doc/ModeladoVisualconUML.pdf.

[27] Molpeceres, Alberto, “Procesos de desarrollo: RUP, XP y FDD”; http://www.javahispano.org/contenidos/archivo/71/metodos_desarrollo.pdf.

[28] Montero, Yusef Hassan y Martín Fernández, Francisco Jesús, “Propuesta De Adaptación De La Metodología De Diseño Centrado En El Usuario Para El Desarrollo De Sitios Web Accesibles, Jul. 2004; http://es-es.start2.mozilla.com/firefox?client=firefox-a&rls=org.mozilla:es-ES:official.

[29]“Rational Unified Process (RUP)”; www.utim.edu.mx/~mgarcia/DOCUMENTO/ADSI2/RUP.pdf .

[30] Bosch, Mela y Thompson, Héctor, “Reconocedor-Asignador para semantización en hipertexto (RA)”; http://eprints.rclis.org/archive/00006068/01/BoschThompsonVCongreso_ISKO.pdf. .....BoschThompsonVCongreso_ISKO.

[31]Díaz Flores, Mirian Milagros, “RUP vs. XP”; http://www.usmp.edu.pe/publicaciones/boletin/fia/info49/articulos/RUP%20vs.%20XP.pdf.

[32] Alferez Salinas, German Harvey y Soto Romero, Omar Otoniel, “Scrum, ¿Un Paradigma de Administración de Proyectos que Cumple lo que Promete?”; http://fit.um.edu.mx/departamentodeinvestigacion/publicaciones/TechnicalReportCOMP-021-2009.pdf .

[33]“Tema II: una Metodología para el desarrollo de BD,” Ene. 2005; http://basesdatos.uc3m.es/fileadmin/Docencia/FSE/Conceptos_Teoricos/Tema_II__Metodolog_a_para_el_desarrollo_de_BD_.pdf.

[34] García Avila Lourdes F., “Tesis para optar por el título estatal de Máster en Informática Aplicada a la Ingeniería y la Arquitectura. “Metodología para evaluar la calidad de la etapa de análisis de proyectos informáticos orientado a objetos (CAOOSI)”; www.cecam.sld.cu/pages/rcim/revista_1/articulos_htm/mariarosa.htm.

[35]Fowler,Martin, “The New Methodology”; http://martinfowler.com/articles/newMethodology.html.

[36] Fraternali, P., “Tools and Approaches for Developing Data-Intensive Web Applications: a Survey”; http://eprints.kfupm.edu.sa/72183/.

[37] Anaya, Víctor y Letelier, Patricio, “Trazabilidad de Requisitos Adaptada a las Necesidades del Proyecto: Un Caso de Estudio Usando Alternativamente RUP y XP”; www.willydev.net/InsiteCreation/v1.0/descargas/prev/traza.pdf.

[38] Begel, Andrew y Nagappan, Nachiappan, “Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study”; http://research.microsoft.com/hip/papers/AgileDevAtMS-ESEM07.pdf.