UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

HERRAMIENTAS DE SOFTWARE DE APOYO AL DESARROLLO CON METODOLOGÍAS ÁGILES EN MIRAS A LA CERTIFICACIÓN DE PROCESOS

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN

IGNACIO EMILIO ORTEGA CONCHA

PROFESOR GUÍA:
MARÍA CECILIA BASTARRICA PIÑEYRO

MIEMBROS DE LA COMISIÓN:
AGUSTIN ANTONIO VILLENA MOYA
JOSE ALBERTO PINO URTUBIA

SANTIAGO DE CHILE
MARZO 2008

RESUMEN DE LA MEMORIA
PARA OPTAR AL TITULO DE
INGENIERO CIVIL EN COMPUTACIÓN
POR: IGNACIO ORTEGA C.
FECHA: 07/03/2008
PROF. GUIA: Sra. MARÍA CECILIA BASTARRICA

"HERRAMIENTAS DE SOFTWARE DE APOYO AL DESARROLLO CON METODOLOGÍAS ÁGILES EN MIRAS A LA CERTIFICACIÓN DE PROCESOS"

Cada vez es más frecuente ver que la industria de software se base en estándares y certificaciones de calidad. Esto ha despertado la iniciativa por parte de las empresas a realizar esfuerzos para mejorar sus procesos de desarrollo de forma continua, con la idea de obtener ventajas competitivas, y abarcar una mayor zona de mercado. Sin embargo, para las empresas, el proceso de adoptar las prácticas orientadas a una mejora de procesos resulta tener un elevado costo, que estas no siempre pueden abordar. Es aquí donde surge la idea de utilizar algunas de las prácticas de las metodologías ágiles, principalmente gracias a su nuevo enfoque de cómo lograr proyectos exitosos y además a la poca inversión inicial que requieren. Con el objeto de apoyar una de las iniciativas de la Agenda Digital, ha surgido el Proyecto Tutelkán, cuyo propósito es crear un mecanismo sustentable, que mediante un proceso público de referencia y una comunidad activa de apoyo, permita a las empresas nacionales contar con las herramientas y asistencia necesarias para iniciar la mejora de los procesos de desarrollo en miras a una certificación.

Identificando necesidades reales del ambiente nacional, gracias a la inserción dentro de la industria se construyeron herramientas para el apoyo de administración de requerimientos y de riesgos, que reúnen tanto características provenientes de las metodologías tradicionales como de las metodologías ágiles, formando una plataforma de trabajo extensible e interoperable, gracias a su orientación de servicios. Todo con el fin de apoyar con herramientas de software que agreguen valor a este mejoramiento de procesos en el desarrollo, sin hacerlo más complejo o burocrático. La acogida desde las empresas fue positiva, reconociendo el valor que aporta a mejorar sus procesos y el apoyo que otorga para mejorar la imagen externa.

Agradecimientos

Un especial agradecimiento a mis padres, Ana María Concha y Luis Ortega, que gracias a su esfuerzo y apoyo permitieron el comienzo y desarrollo de este trabajo, mi carrera universitaria.

Agradezco además los comentarios y sugerencias de Roberto Carrasco, quién permitió una buena definición del trabajo.

Índice

Índice de Tablas

Índice de Ilustraciones

1. Introducción

La Agenda Digital es la estrategia país en el ámbito de las tecnologías de información y comunicación con miras al bicentenario. Tiene como propósito incrementar la competitividad y la igualdad de oportunidades en el sector público [1]. Uno de sus objetivos es alcanzar una masa crítica de empresas TIC, capaz de competir internacionalmente; de esta forma, propone como iniciativa el asegurar la calidad a través de la certificación de empresas [1]. A raíz de esta, se ha concebido el Proyecto Tutelkán [49], cuya misión es la "*obtención de altos estándares de calidad en la industria de software nacional, utilizando procesos de desarrollo de referencia*".

1.1. La realidad nacional

La mejora de procesos posibilita el desarrollo de productos de mayor calidad, una mejor imagen y la apertura de nuevos mercados. GECHS [22] en su quinto diagnóstico del año 2006 [23], realizado con el apoyo de más de 40 empresas asociadas, indica que con respecto al año anterior existe un mayor número de empresas exporta productos y servicios, lo que muestra la progresiva salida al exterior que ha experimentado nuestra industria, y que aún sigue en proceso. Según el mismo informe casi el 70% de las empresas opina que el haber implementado procesos formales ha generado mayor satisfacción de clientes, y más del 60% asegura que les ha permitido mejorar la administración. Un aspecto también importante a considerar, es que casi el 40% de las empresas opina que el haber implementado herramientas formales ha permitido mejorar la imagen de la empresa.

Si bien gran parte de las empresas está consciente de que la calidad de los productos es un tema fundamental, un bajo porcentaje incorpora en sus procesos de desarrollo las herramientas y metodologías que apoyen el desarrollo de este tema. El mercado nacional en la industria de software está marcado por la inmadurez con que generalmente se abordan sus procesos de desarrollo. Contamos con innumerables ejemplos de proyectos que fracasan

o que tardan mucho más del tiempo que tenían asignado, lo que merma la imagen de los productos de software nacional, y disminuye la posibilidad de apertura hacia mercados internacionales. De aquí nace la necesidad de contar con mecanismos que nos den una guía de cómo abordar estos procesos, para que nos permitan tener una mayor efectividad y una buena administración.

Sabemos que los modelos de mejora de procesos entregan aquellas mejores prácticas que permitan hacer más eficaces los procesos de desarrollo y tener una mejor administración de los proyectos, pero para la realidad nacional, cuyo mercado se compone principalmente de pequeñas y medianas empresas, esto significa hacer una gran inversión que pocas pueden abordar, por lo que sus elevados costos mitigan la motivación de adoptar estas prácticas. Sin embargo, las metodologías ágiles nos entregan una forma de abordar el proceso de desarrollo con una menor inversión inicial, además de enfatizar el hecho de tomar ventajas sobre las habilidades del equipo humano, y poner en controversia muchos de los supuestos que tiene el desarrollo de software tradicional. Las metodologías ágiles han aumentado su popularidad durante los últimos años, estas han sido desarrolladas con el objetivo de entregar software de forma rápida y que satisfaga las necesidades cambiantes de los clientes. Las propuestas ágiles comparten algunos principios comunes: mejorar la satisfacción del cliente, adaptarse a los cambios en los requerimientos, entregas frecuentes de software funcional, colaboración cercana entre clientes y desarrolladores.

Si bien las empresas han tratado de adoptar estas metodologías y tener en vista los procesos de certificación como forma de mejoramiento de procesos, existe aún un alto desconocimiento de estas prácticas, lo que lleva a pensar en formas de proveer esta información. Según el mismo informe de GECHS, dentro de las normas o procesos menos conocidos por las empresas, el 30% de ellas opina que es el Extreme Programming.

1.2. Proyecto Tutelkán

El propósito del proyecto Tutelkán es crear un mecanismo sustentable que, mediante un proceso público de referencia y una comunidad activa de apoyo, permita a las empresas nacionales contar con las herramientas y asistencia necesarias para iniciar la mejora de los procesos de desarrollo, además de proveer una base de conocimiento y apoyo, para generar competencias en la adopción procesos de desarrollo de software con vistas a la certificación ISO 9001 [27], y/o evaluación CMMI [14].

Inicialmente, el proyecto consiste en apoyar a un grupo de empresas piloto, para adoptar un modelo inicial ya certificado en la empresa Kepler Technologies [35], y luego, a partir de las experiencias recogidas, potenciar y fortalecer el proceso semilla, de forma de hacerlo disponible a otras empresas interesadas en un mejoramiento continuo de sus procesos.

1.3. El Problema

La primera parte del trabajo, fue el participar activamente dentro de la implantación del proceso semilla liderada por Kepler Technologies en las empresas piloto Angecom [5], Intermedia [27], Nectia [39] y Softram [55]. La finalidad de esta participación era identificar áreas dentro del desarrollo de software que manifiesten debilidades dentro del proceso semilla y/o dentro de las empresas, de forma que puedan ser apoyadas a través de herramientas de software.

Las áreas identificadas fueron la administración de requerimientos y la administración de riesgos. La administración de requerimientos es fundamental dentro de todo desarrollo de software debido a que se encarga de establecer de la forma más clara posible cuáles son las necesidades del cliente que pueden satisfacerse a través de una herramienta de software, ya que sabemos que una necesidad no entendida puede llevar al

fracaso del producto por no ser de utilidad o no reflejar el real problema que se intentaba resolver. A su vez el área de administración de riesgos tiene un papel fundamental en identificar y prever cuáles son las amenazas que pueden perjudicar el éxito o el logro de objetivos de un proyecto.

Es de nuestro interés construir herramientas que apoyen estas dos áreas de proceso, y que representen una alternativa ágil en la administración de las áreas, entendiendo ágil, como una herramienta que aporte valor en los procesos en que participe y que permita poner en práctica principios tales como tener un proceso adaptable ante posibles cambios y apoyo a la comunicación entre los participantes. Adicionalmente se pretende que las herramientas provean lo necesario pero de tal forma que el costo de adopción, aprendizaje e incorporación a los procesos estándar, sea mucho menor respecto de los beneficios que proporcionan, además de ofrecer atributos de calidad que las actuales alternativas no poseen, fortaleciendo los procesos internos para llegar a ser un real valor dentro de la organización, en la medida que se ajusten a su realidad y reflejen la forma de actuar dentro de ella. La realidad nacional requiere de herramientas que posibiliten a las organizaciones funcionar de forma productiva, que se alineen a sus objetivos de negocios, y que su uso no signifique un costo adicional, ya sea porque no se ajustan a sus necesidades, posean demasiadas funcionalidades, o debido a que necesitan de capacitación interna.

El trabajo se alinea a los propósitos del proyecto Tutelkán, recogiendo los objetivos de las áreas de proceso que indica el CMMI, en combinación con prácticas provenientes de las metodologías ágiles que mejor se ajustan a la realidad nacional de pequeñas y medianas empresas, para conformar una base conceptual que guíe la construcción de las herramientas de apoyo propuestas.

Es así como el presente trabajo desea entregar un nuevo enfoque en la construcción de las herramientas de apoyo al desarrollo de software, en base a las dos corrientes que actualmente existen, la tradicional y la ágil. Intenta formar una plataforma de trabajo que posibilite y facilite la inserción de características según las necesidades de cada empresa, de

forma que las herramientas se adapten y ajusten a su manera de organizarse y realizar sus actividades.

Dentro de los desafíos interesantes para este trabajo podemos mencionar:

- Combinar ambas perspectivas para potenciar las características de las herramientas. Tomar elementos de ambas perspectivas que aporten valor en el diseño de las herramientas. El desafío está en que ambas perspectivas, en muchos puntos, parecen contradictorias. La idea es reconciliar esos puntos para proveer una base conceptual para el desarrollo y verlos reflejados en las herramientas.

- Asegurar su efectividad y eficiencia. Una de las características fundamentales de toda herramienta es poder asegurar su efectividad y si esta constituye un aporte real de valor para sus usuarios. Así se realizará una evaluación con empresas, de forma que poder verificar en la práctica el aporte de las herramientas.

- Medir el impacto en la práctica. Las empresas que están bajo el proyecto, tienen motivaciones para iniciar acciones de mejora interna dentro la organización a través del proceso de referencia que éste provee. Las herramientas constituyen un apoyo para ciertas áreas que necesitan una mejora, por lo cual se busca determinar el valor aportado de las herramientas respecto de la situación anterior sin el uso de la herramienta.

1.4. Solución

El CMMI representa una guía de cómo las empresas deben desarrollar software, entrega las metas que debe alcanzar una organización en las diferentes áreas, pero no precisa cómo una organización puede alcanzarlas. Las empresas deben interpretar y adaptar a su realidad las áreas de proceso que propone el CMMI para definir su forma particular de acción.

Dado que la forma de accionar es única para cada organización, no se busca generar herramientas que representen la forma de actuar de una sola empresa, sino intenta ser una solución que permita adaptarse o extenderse según el contexto. Busca además satisfacer los objetivos propuestos por CMMI y alinearse con la propuesta ágil de desarrollo, dado que en ella podemos encontrar un método que se adapta al contexto de pequeñas y medianas empresas, que es hacia donde apunta el proyecto Tutelkán.

La solución entonces para el desarrollo de las herramientas será la construcción de una plataforma, donde vivirán las herramientas, de forma que posibilite la adaptación para las necesidades de una organización particular, o para una posible extensión, que cubra áreas relacionadas con las que se abordan en este trabajo.

Las herramientas construidas para el presente trabajo representan una alternativa hecha sobre la Web, y su clave es la arquitectura elegida para el desarrollo de las herramientas, la que provee una serie de características que posibilitan la concreción de la plataforma. El estilo de arquitectura es REST, Representational State Tranfer (Transferencia de Estados Representacionales). REST hace una abstracción de como actualmente está hecha la Web, y reúne todas las cualidades que la han hecho exitosa. Con REST se modela la Web en base a recursos, y se basa en una arquitectura cliente-servidor, donde el servidor no almacena estados, toda la información para entender una petición viene en ella. Su implementación ocupa los estándares HTTP, para la comunicación cliente-

servidor, URI para la identificación de recursos. Una descripción más detallada de REST se encuentra en el anexo 7.3.

Los resultados obtenidos muestran en la práctica que la combinación de los objetivos del CMMI y prácticas ágiles puede darse, lo que en definitiva generó una base conceptual para las herramientas. Por otra parte, las herramientas reflejan la concreción de una plataforma, que presenta las características de extensibilidad e interoperabilidad. La extensibilidad, dada por la adaptación que se hizo para la forma de trabajar de una empresa particular, y la interoperabilidad, que permite que la aplicación no sea vista sólo a través de un cliente Web.

A su vez por parte de las empresas que utilizaron las herramientas destacaron su interfaz clara y simple para alcanzar los objetivos de gestión planteados por ellas.

1.5. Organización del documento

En la siguiente sección presentamos el marco teórico, mostrando las características de los mundos de las metodologías tradicionales y ágiles en el desarrollo de software, presentado en más detalle las perspectivas del CMMI y Extreme Programming. En la sección 3 y 4 presentamos el desarrollo de las herramientas de software y finalmente la sección 5 presenta las conclusiones de este trabajo.

2. Desarrollo de Software

2.1. Modelos de Desarrollo de Software

El desarrollo de software es la traducción de una necesidad del usuario a un producto de software [65]. El conjunto de actividades y resultados asociados que producen un sistema de software, es el proceso de desarrollo de software [56][57]. En la actualidad existen distintos modelos que presentan una descripción de cómo debe ocurrir este proceso de software. En los 60's surge un modelo inicial que define un ciclo de vida del proceso de desarrollo de software, el cual hoy es conocido como el Modelo Cascada [51]. A partir de éste se han presentado otros modelos o variaciones que describen el ciclo de vida del desarrollo de software, entre ellos podemos mencionar el modelo incremental [7] o el modelo espiral [11]. También podemos mencionar aproximaciones que se basan en describir comportamientos de las actividades del desarrollo de software, conocido como Behavoiral Approach [67], o métodos formales [60], que son intentos matemáticos de abordar las actividades que se presentan en el ciclo de vida del desarrollo.

Uno de los enfoques que se ha fortalecido a medida que se presentan modelos descriptivos o prescriptivos de desarrollo, es el basado en la mejora continua de procesos. Uno de los más destacados es el CMM [13], y su evolución, el CMMI [4][12][15]. Sin embargo para pequeñas empresas significa un alto costo el implantarlos [52][58].

Las metodologías ágiles de desarrollo [42] aparecen como una nueva forma de enfrentar un proceso de desarrollo, bajo un manifiesto [3], del cual se extraen los principios que deben gobernar dentro una metodología ágil. Una que se ha propuesto es Extreme Programming [9][10][[40], la cual provee un conjunto de principios, prácticas y valores que deben existir en el desarrollo de software. Es llamada "extreme" pues hace extremo el uso de buenas prácticas en el desarrollo de software [54]. Además está orientado a pequeñas organizaciones, donde el cambio de los requerimientos es una constante. Si bien los beneficios que otorgan las metodologías ágiles son atractivos, las organizaciones deben reflexionar sobre cómo hacer una apropiada integración junto con las prácticas que ya existen [41].

Mejorar la calidad y productividad de los procesos de desarrollo de software se ha convertido en una prioridad para casi todas las empresas que confían en computadores y redes [37]. Pero aún se debe investigar sobre la adaptabilidad de las metodologías ágiles con los estándares de desarrollo de software [61], o propuestas como la que hace el CMMI [8].

Profundizamos a continuación el enfoque de desarrollo en base a procesos propuesto por el CMMI, para luego presentar el enfoque ágil y discutir sus posturas.

2.2. Modelos basados en procesos

Now, more than ever,
companies want to deliver products and services better, faster, and cheaper.
-- CMMI Version 1.2

El enfoque en procesos provee la infraestructura necesaria para tratar con un mundo que cambia constantemente, maximizar la productividad de la gente y el uso de la tecnología para ser más competitivos.

Los procesos ayudan al personal de la organización a encontrar sus objetivos de negocios, haciendo que trabajen de forma más ordenada, sin desperdiciar esfuerzos y con mejoramiento consistente. Los procesos efectivos también proveen un vehículo para introducir y usar nueva tecnología de la mejor forma para alcanzar los objetivos de negocio de la organización.

Entre los diferentes modelos basados en procesos que existen como referencia están:

- ISO 9001 [28]: es un conjunto de estándares internacionales para sistemas de calidad. Diseñado para la gestión y aseguramiento de la calidad, especifica los requisitos básicos para el desarrollo, producción, instalación y servicio a nivel de sistema y a nivel de producto.

- CMM/CMMI: El CMMI es el modelo de madurez de capacidades integrado, que proviene del CMM. El CMMI identifica las áreas de proceso generales que debiesen ocurrir en toda organización y forma un camino evolutivo de desarrollo y mejora de estos procesos a través de niveles de madurez, que en definitiva sirven como indicadores para una organización de su madurez. Un detalle más a fondo sobre CMM y CMMI se puede encontrar en los anexos 7.1.1 y 7.1.2.

- ISO/IEC 15504-2 (Parte 2) 1998 [30], que es compatible con la norma ISO/IEC 12207: 1995/Adm 1. 2002 [29]. La norma establece un marco de referencia común para los procesos del ciclo de vida del software, con una terminología bien definida a la que puede hacer referencia la industria del software. Contiene procesos, actividades y tareas para aplicar durante la adquisición de un sistema que contiene software, un producto software puro o un servicio software y durante el suministro, desarrollo, operación y mantenimiento de productos software.

2.3. Metodologías Ágiles

Las metodologías constituyen un nuevo enfoque sobre el desarrollo de software, y tiene un factor común definido por la Agile Allience [2] en su Agil Manifesto [3], el cual señala:

- Individuos y sus interacciones por sobre procesos y herramientas
- Software funcional sobre documentación exhaustiva
- Colaboración con el cliente por sobre negociación de contratos
- Responder al cambio por sobre seguir un plan

El Manifiesto Ágil nos indica entonces que si bien existe valor en las cosas que se exponen a la derecha, las de mayor valor se encuentran a la izquierda.

Actualmente existes varias metodologías ágiles de desarrollo, como Scrum [53], Lean Develpoment [48], Adaptative Software Development [25] y otras. Sin embargo una de las más populares es Extreme Programming [9][10]. En el anexo 7.2 presentamos una visión general de estas metodologías ágiles y en particular la propuesta hecha por Extreme Programing.

2.4. Discusión

El enfoque de los métodos tradicionales comparte la visión de que el proceso de desarrollo de software es análogo a un proceso industrial, buscan ser predictivos, repetibles y estables. Los sistemas son completamente especificables desde un comienzo, por lo que se genera una planificación que debe seguirse a través de la vida del proyecto. Sin embargo, sabemos que el cambio es una constante en el mundo de hoy, además del simple hecho que predecir desde un comienzo ya es de por sí una tarea muy difícil de concretar. No podemos determinar qué es lo que puede cambiar en el futuro. El tradicional desarrollo de software guiado por la planificación carece de flexibilidad para adaptarse dinámicamente y ajustar su proceso de desarrollo. Por su parte, las metodologías ágiles cuestionan la suposición de que el cambio y la incertidumbre pueden ser controlados con altos niveles de formalización.

En general, tanto para CMMI como para XP, el diagnóstico de los problemas del desarrollo de software es similar: los proyectos de software están casi siempre retrasados, por encima del presupuesto y poco o casi nada funcionales, por lo que buscan producir productos y/o servicios de calidad, crear valor a sus clientes y mejorar su satisfacción. Sin embargo, el enfoque de solución es distinto. CMMI ve que la calidad de los productos está en directa relación con la calidad de los procesos dentro de la organización, lo que genera que las personas, el principal componente dentro del desarrollo, se vean como recursos, pero, sabemos que bajo las mismas especificaciones o bajo el mismo proceso y con personas distintas no se produce el mismo resultado. Por el contrario, el enfoque ágil asume la importancia de las personas dentro del desarrollo y busca una solución en torno a generar equipos autoorganizados en torno a un conjunto de valores, principios y prácticas comunes.

Los métodos ágiles están centrados en las personas, reconociendo el valor que traen sus competencias y sus relaciones en el desarrollo de software [40].

CMMI ha mostrado ser un apoyo para las empresas que lo han adoptado, hay resultados del éxito que ha tenido y los beneficios que en consecuencia ha aportado a las organizaciones. Su enfoque es a nivel organizacional a través de la mejora continua de procesos y a través de sus niveles provee de un indicador de madurez organizacional, a diferencia de las metodologías ágiles que se centran en el proceso de desarrollo. Es por esto que la comparación podría ser inapropiada y podríamos decir que son propuestas que se pueden complementar para potenciarse. De hecho, ya existen varios intentos que buscan analizar el grado de cumplimiento que posee XP en relación a las áreas de proceso del CMM-CMMI, según Paulk [47] XP puede ser usado para alcanzar muchas de las prácticas de nivel 2 y 3 del CMM, a su vez similar diagnóstico se hace en [21] respecto del CMMI. Las principales debilidades exhibidas por XP se refieren a las áreas de proceso de enfoque organizacional, que se encuentran en los niveles 4 y 5.

Existe un amplio interés por la adopción de las metodologías ágiles, incluso por aquellas que ha optado por el enfoque de mejora de procesos como el CMMI. Sin embargo las organizaciones deben ser cautelosas a la hora de abordar este tipo de cambios. Los cambios en el proceso de desarrollo de software representan un complejo fenómeno de cambio organizacional, y no puede ser llevado simplemente reemplazando las actuales herramientas por las nuevas. Según Boehm [6] las organizaciones deben evolucionar cuidadosamente hacia el mejor balance de los métodos ágiles y aquellos guiados por una planificación que mejor se adecuen a su organización.

Nerur [41] resume las principales diferencias de ambas perspectivas en la Tabla 1.

	Tradicional	**Ágil**
Supuestos fundamentales	Los sistemas son completamente especificables, predecibles y pueden ser construidos a través de una cuidadosa planificación	El software de alta calidad y adaptable puede ser desarrollado por equipos pequeños usando los principios de mejoramiento de diseño continuo y basado en el testing con una rápida retroalimentación y cambio.
Control	Centrado en el proceso	Centrado en las personas
Estilo de Administración	Comandar y controlar	Liderazgo y colaboración
Administración del Conocimiento	Explícito	Tácito
Asignación de roles	Individual, a favor de la especialización	Equipos auto organizados, promueve el intercambio de roles
Comunicación	Formal	Informal
Rol del Cliente	Importante	Critico
Ciclo del Proyecto	Guiados por tareas o actividades	Guiado por funcionalidades
Modelo de Desarrollo	Modelo de Ciclo de Vida (Cascada, Espiral, etc.)	Modelo de entrega evolutiva
Estructura organizacional deseada	Mecánica (burocrática con alta formalización)	Orgánica (flexible y participativa, promoviendo la acción cooperativa)

Tabla 1: Diferencias entre la perspectiva tradicional y la ágil

3. Herramientas de Software

La motivación detrás de la construcción de herramientas de software es proveer al proyecto Tutelkán de herramientas que apoyen dos áreas en particular, definidas por el CMMI, en el cual está basado su proceso semilla. La idea de introducir conceptos que provienen de las metodologías ágiles nace dado que el contexto que poseen las empresas usuarias del proyecto responde al contexto al que van dirigidas las metodologías ágiles.

En lo que sigue presentaremos la situación actual respecto a las herramientas que existen y además mostramos las visiones del CMMI y de las metodologías ágiles sobre las áreas en que se focalizará el desarrollo de las herramientas, otorgando la base conceptual para su desarrollo.

3.1. Administración de Requerimientos

Los requerimientos de cliente a menudo cambian sustancialmente durante el desarrollo, con la consecuencia de que los desarrolladores tengan que trabajar con una siempre cambiante especificación. La mejor manera de trabajar con esto es un proceso adaptativo e iterativo permitiendo que el código de desarrollo cambie para satisfacer los requerimientos [50]. En el proceso tradicional de ingeniería de software, la ingeniería de requerimientos tiene como objetivos identificar, analizar, documentar y validar requerimientos para el sistema a construir [45].

3.1.1. Situación Actual

Las herramientas comerciales para la administración de requerimientos pueden ser divididas en dos categorías [50]:

- Centradas en el documento: las cuales soportan la vista de texto de los requerimientos. DOORS, RTM, Document Director y otras.

- Orientadas al modelo: que modelan los requerimientos como información estructurada. (Analyst, CORE, Caliber RDD-100)

En general, las herramientas comerciales proveen un gran conjunto de funcionalidades, lo que las hace complejas y conllevan un gran costo de aprendizaje el cual debe abordarse en base a la capacitación, esto justifica el hecho de construir una herramienta con el espíritu de hacer sencilla la labor de administración de requerimientos.

El proceso semilla además provee una herramienta de administración de requerimientos que estará disponible dentro del proyecto Tutelkán, si bien es más sencilla, hace difícil la posibilidad de un ambiente compartido y colaborativo de trabajo y puede tener serias falencias en el manejo de los datos, además de ocupar una herramienta propietaria como plataforma de trabajo.

3.1.2. Administración de Requerimientos según el CMMI.

Según lo que presenta el CMMI, los objetivos de la Administración de Requerimientos son:

- Administrar los requerimientos, los productos y sus componentes (que corresponden a los productos de trabajo a ser entregados al cliente o al usuario final),
- Identificar las inconsistencias entre dichos requerimientos y los planes del proyecto y productos de trabajo.

A continuación presentamos los objetivos específicos y las prácticas específicas que el CMMI define para esta área.

Objetivos Específicos (OE) y Resumen de Prácticas Específicas (PE).

OE 1: Administrar Requerimientos

Los requerimientos son gestionados, y las inconsistencias entre los planes del proyecto y los productos relacionados deben ser identificados.

PE 1.1: Obtener entendimiento de los requerimientos: Desarrollar un entendimiento con los proveedores de los requerimientos sobre el significado de los mismos. Esto se hace a través de la definición, por ejemplo, de los proveedores de requerimientos válidos, criterios de evaluación y aceptación de requerimientos. Una forma de evaluar el entendimiento sobre un requerimiento es preguntar si: ¿está completo?, ¿es consistente con el resto?, ¿se puede implementar?, ¿es verificable?, ¿es trazable?

PE 1.2: Obtener Compromiso en los requerimientos: Obtener compromiso con los requerimientos de los participantes del proyecto.

PE 1.3: Administrar los cambios en los requerimientos: Administrar los cambios en los requerimientos debido a su evolución durante el transcurso del proyecto.

PE 1.4: Mantener la trazabilidad bidireccional de los requerimientos: Mantener la trazabilidad bidireccional (corresponde a una asociación entre dos o más entidades lógicas que se puede establecer en ambas direcciones) entre los requerimientos y cada nivel de descomposición del producto. La idea es mantener la trazabilidad entre el requerimiento origen hacia el nivel más bajo y viceversa. El propósito de la trazabilidad bidireccional es ayudar a determinar si todos los requerimientos han sido completados y pueden ser trazados a un requerimiento de origen válido.

PE 1.5: Identificar inconsistencias entre el trabajo del proyecto y los requerimientos: Esta práctica identifica las inconsistencias entre los requerimientos, con los productos de trabajo y los planes del proyecto, e inicia las acciones correctivas para superarlas.

3.1.3. Administración de Requerimientos Ágil

Las metodologías ágiles tienen como características ser un proceso adaptable y orientado a las personas. El proceso adaptable en un sentido se entiende como la capacidad que se tiene ante los cambios en los requerimientos, a diferencia de las metodologías tradicionales éstos no se definen por completo al comienzo de un proyecto, sino que se permite que se modifiquen de acuerdo a la necesidad planteada por el cliente. Es por este motivo que la relación y comunicación entre ambas partes es de suma importancia para concretar objetivos de forma de aportar el mayor valor posible al cliente. Un cambio de perspectiva sustancial es el de establecer claramente la imposibilidad de definir en un comienzo el tiempo, precio y alcance de un proyecto, por lo que en general toma el enfoque de fijar el tiempo y precio, dejando como variable el alcance del proyecto. Se entiende entonces que los requerimientos definidos desde un comienzo no conforman parte del contrato y se establece el concepto de contrato de alcance variable, fundamentado siempre bajo el contexto donde los requerimientos o necesidades pueden cambiar conforme el avance del proyecto. De este último nacen las responsabilidades que deben tener tanto desarrolladores como clientes en un proyecto administrado de forma ágil, que impacta la forma de administrar los requerimientos dentro de un proyecto.

Las responsabilidades por parte del cliente son la de maximizar el valor obtenido por cada entrega, a su vez que el desarrollador maximiza la calidad de su trabajo. El cliente define que es lo que será implementado y en que prioridad según sus necesidades, y el desarrollador estima honestamente cuanto tardará en realizar el trabajo. Por último al cliente se le permite cambiar funcionalidades por otras equivalentes, a su vez que al

desarrollador se le permite cambiar sus estimaciones a medida que su conocimiento del problema a resolver aumente. La Tabla 2 resume estas responsabilidades [62].

	Cliente	Desarrollador
Desea Maximizar	Valor recibido por entrega del desarrollo	Calidad del trabajo realizado
Puede definir	Qué será implementado y en que prioridad, según las necesidades de su negocio	Cuánto estima que se demorará una tarea.
Puede cambiar	Funcionalidades solicitadas por otras no implementadas de costo equivalente	Sus estimaciones en base a nuevos descubrimientos

Tabla 2: Responsabilidades entre Cliente y Desarrolladores

Los equipos de desarrollo ágil aceptan el cambio, tomando la idea que los requerimientos evolucionan durante la vida del proyecto. Los métodos ágiles entienden que los requerimientos evolucionan con el tiempo, por lo que una inversión temprana en una descripción detallada de ellos sería una pérdida. Lo que hacen entonces, será un modelamiento de requerimiento inicial, para identificar el alcance del proyecto y desarrollar una planificación de alto nivel, y luego estimar, qué es lo que se necesita en una etapa temprana del proyecto, por lo que eso es lo que se debería hacer. Durante el desarrollo se realiza un detalle más fino, en la medida de lo necesario.

La forma que se utiliza para especificar requerimientos bajo la perspectiva ágil son las *"historias de usuarios"* [17]. Una historia de usuario describe la funcionalidad deseada desde la perspectiva cliente o usuario, quién la quiere y cómo y porqué la funcionalidad será usada.

Los componentes básicos de una historia de usuario son a veces conocidos como las tres C [33], Card, Conversation, Confirmation:

- Card: la descripción escrita de la historia, que sirve como una identificación, recordatorio y también ayuda en la planificación.
- Conversación - es el dialogo es llevado con los usuarios, notas, documentos intercambiados.
- Confirmación - El criterio de test de aceptación que el usuario usara para confirmar que la historia esta completa.

Una historia de usuario no es técnica. Una buena historia de usuario sigue el modelo INVEST: Independent, Negotiable, Valuable, Estimable, Small, Testable.

- Independent: una historia de usuario debe ser independiente de otra, las cuales deben planificarse, priorizarse y la estimación debe ser difícil. A menudo, dependencias pueden ser reducidas mediante la combinación de historia dentro de una o dividiendo las historias.
- Negociable: una historia de usuario es negociable. La historia es solo una descripción breve de la historia la cual no incluye detalles. Los detalles son trabajados dentro la conversión con el cliente.
- Valuable: Cada historia tiene que tener un valor para el cliente (tanto para el usuario como para el comprador). Una buena forma de valorar las historias es hacer que el mismo cliente la escriba.
- Estimable: Los desarrolladores necesitan ser capaces de estimar una historia de usuario para permitir la priorización y el plan de la historia.
- Small: Una buena historia de usuario debe ser pequeña en esfuerzo, típicamente representando no más del esfuerzo de dos o tres personas para una semana.
- Testable: Una historia de usuario necesita ser testeable para que la confimación se haga efectiva. No se desarrolla lo que no se puede testear. Si no se puede hacer test entonces nunca sabrá lo que se está haciendo. Un ejemplo de una historia que no se puede testear: "Software debe ser fácil de usar".

¿Cómo se maneja el cambio desde una perspectiva ágil?

Los métodos ágiles desarrollan software que es de alta calidad y de alto valor, y la mejor manera de construir software de alto valor deforma temprana es construir los requerimientos de alta prioridad primero.

La idea está en manejar una pila de requerimientos, la cual contiene al tope de la lista los requerimientos de alta prioridad y por los cuales se trabajará en la iteración, los que deberían expresarse de forma más detallada. Dado que las metodologías ágiles adoptan el cambio de forma natural, en cualquier momento durante el proyecto puede ingresarse un nuevo requerimiento, el cual entra en la pila de acuerdo a la priorización hecha por el cliente. Además, en cualquier instante se puede cancelar un requerimiento y eliminarlo de la pila. El cambio en los requerimientos se maneja como el ingreso de un nuevo requerimiento. La Ilustración 1, resume lo anterior.

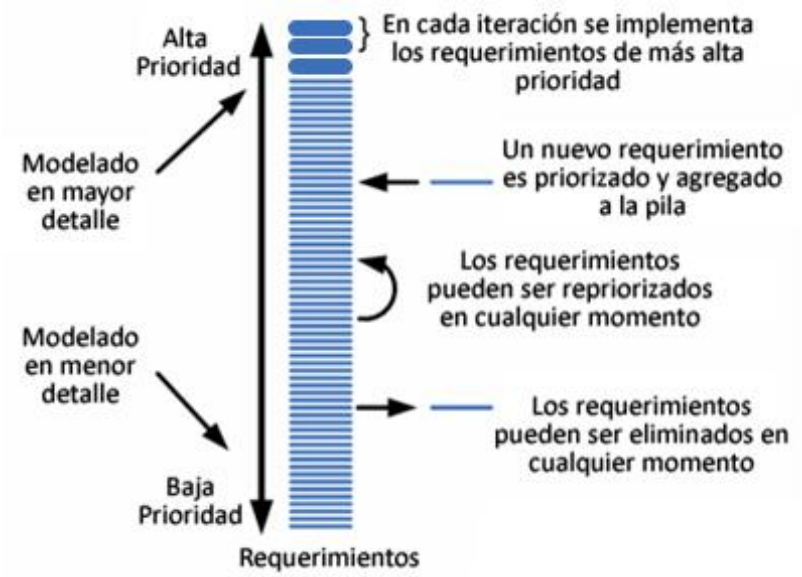


Ilustración 1: Administración de Requerimientos Ágil

3.1.4. Discusión

Los requerimientos describen qué es lo que se va a hacer pero no cómo serán implementados [45]. Una de las más visibles diferencias entre el desarrollo de software ágil y las metodologías tradicionales: “la urgencia en realizar la documentación de los requerimientos debe ser transformada en una urgencia por colaborar cercanamente con los clientes para entonces crear software funcional basado en lo que te digan”.

Desde el punto de de la ingeniería de software, las principales debilidades de XP son sus problemas de mantenibilidad, como resultado de su limitada documentación (XP se basa en el código y los casos de prueba simplemente) y falta de una visión amplia del sistema a ser construido [38]. Entonces, la pregunta que se plantea: ¿Es posible tener una metodología ágil y requerimientos documentados?

Nuestra propuesta para la herramienta se basa en que si se necesita la documentación de los documentos y a diferencia de XP que se basa en ocupar herramientas más sencillas como las “*story card*s” escritas en papel, se proveerá de una herramienta de software debido a la facilidad que entrega en la organización y búsqueda de entrega. Sin embargo, no se renuncia al principio ágil de que la comunicación sea personal, la herramienta debiese ser usada con el ánimo de evaluar que los requerimientos sean entendidos y para que se establezcan las condiciones de satisfacción de los requerimientos. Además se adhiere a la posición de Steve [59], que indica que la administración de los requerimientos es un proceso político y no técnico, y su verdadera importancia radica en controlar expectativas. Se necesita administración de requerimientos para proteger a los administradores de proyectos por falsas acusaciones de promesas rotas cuando salga la primera versión del producto.

La Tabla 3 resume las diferencias entre la perspectiva tradicional y la ágil sobre la administración de requerimientos.

	Tradicional	**Ágil**
Especificación de requerimientos	Se realiza completamente al comienzo del proyecto y generalmente constituye el contrato del proyecto	Se hace en la medida de lo necesario, especificando con mayor detalle las funcionalidades de mayor valor para el cliente bajo un enfoque de adaptación frente a los cambios en los requerimientos
Compartir conocimiento	Centrada en la documentación	Comunicación interpersonal
Priorización	Establece el orden en cuanto a cuando la funcionalidad es implementada	Establece el orden según el valor que le otorga al cliente.
Involucración del cliente	En la etapa de especificación de requerimientos	Durante todo el proceso de desarrollo

Tabla 3: Diferencias entre la perspectiva tradicional y la ágil en la administración de requerimientos

3.2. Administración de Riesgos

El desarrollo de software está a menudo plagado de problemas que no se anticipan, lo que produce que los proyectos no terminen a tiempo, se excedan de su original presupuesto, o que entreguen productos no satisfactorios. Si bien estos problemas no se pueden erradicar, algunos de ellos pueden controlarse mediante la toma de acciones preventivas. La administración de riesgos es un área de la administración de proyectos que trata con estas amenazas antes de que ocurran.

El propósito de la Administración de Riesgos es identificar los problemas potenciales antes de que ocurran, de forma que las actividades de manejo de riesgos pueden ser planeadas e invocadas según sea la necesidad, a través de la vida del producto o proyecto, para mitigar impactos adversos en el logro de objetivos.

3.2.1. Situación Actual

Muchas de las herramientas que actualmente existen para la administración de riesgos se caracterizan por su complejidad. En el ámbito comercial existe una amplia variedad de herramientas y están más bien orientadas a la gestión de riesgos a nivel organizacional. A su vez es difícil encontrar alternativas libres o de código abierto.

En particular, la empresa Kepler Technologies posee una herramienta construida en MS Word, que constituye el estándar de la organización, esta se tomará como base para la propuesta de nuestra administración de riesgos, pero al igual que la herramienta de administración de riesgo hace difícil el trabajo colaborativo.

3.2.2. Administración de Riesgos según el CMMI.

Esta área de proceso pertenece al nivel 3 de madurez definido por el CMMI. Las características de esta son:

- Proceso continuo y de visión hacia adelante.
- Debe dirigir temas que ponen en peligro el logro de objetivos críticos.
- Una administración continua de riesgos es aplicada para anticipar efectivamente y mitigar los riesgos que tienen impacto crítico para el proyecto.
- La administración efectiva de riesgos incluye la identificación temprana y agresiva de los riesgos, a través de colaboración e involucración de los principales interesados.
- La administración de requerimientos puede ser dividida en tres partes: definir una estrategia para la administración de riesgos, identificar y analizar riesgos, tratar los riesgos identificados, incluyendo la implementación de los planes para mitigarlos cuando se necesite.

Objetivos Específicos (OE) y Resumen de Prácticas Específicas (PE).

OE 1: Preparase para la Administración de Riesgos

Establece y mantiene una estrategia para identificar, analizar y mitigar los riesgos.

PE 1.1: Determinar las fuentes de riesgo y categorías: Se identifican las fuentes de riesgos tanto internas como externas y se establecen las categorías de riesgos de forma de proveer un mecanismo para recolectar y organizar los riesgos.

PE 1.2: Definir los parámetros de los riesgos: Se definen los parámetros para analizar y categorizar riesgos, y los parámetros para controlar la administración de riesgos.

PE 1.3: Establecer una estrategia de administración de requerimientos: Se establece una estrategia para administración de requerimientos que puede contemplar el definir el

alcance en la administración de riesgos, herramientas a ser usadas, como los riesgos serán organizados o categorizados, o las técnicas para la mitigación de ellos.

OE 2: Identificar y Analizar Riesgos

Los riesgos son identificados y analizados para determinar su importancia.

PE 2.1: Identificar Riesgos: Se identifican las posibles amenazas, peligros o vulnerabilidades que pueden afectar negativamente en el logro de objetivos, para luego documentar los riesgos encontrados de forma concisa que precise su contexto, condiciones y consecuencias.

PE 2.2: Evaluar, Categorizar y priorizar los riesgos: Se evalúan y categorizan los riesgos según los parámetros y categorías definidas para luego determinar su prioridad.

OE 3: Mitigar Riesgos

Los riesgos son manejados y mitigados cuando es apropiado, de forma de reducir el impacto en el logro de objetivos.

PE 3.1: Desarrollar los planes de mitigación de riesgos: Se desarrollan planes de mitigación para los riesgos más importantes para el proyecto de acuerdo a la estrategia de mitigación definida.

PE 3.2: Implementar los planes de mitigación de riesgos: Se monitorea el estado de los riesgos periódicamente y se implementan los planes de mitigación cuando sea apropiado.

La Ilustración 2 muestra un esquema que sintetiza las actividades involucradas en la administración de riesgos de acuerdo al CMMI.

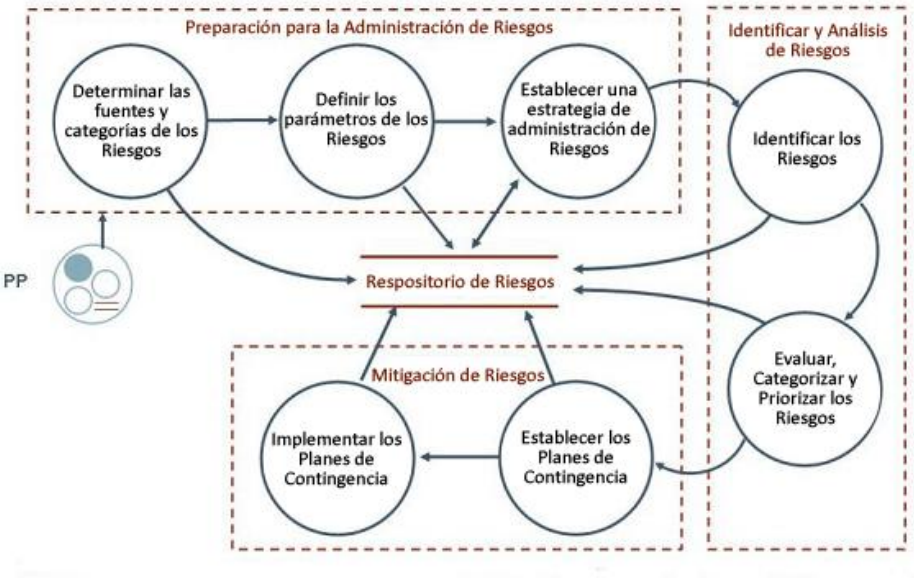


Ilustración 2: Esquema de Administración de Riesgos según el CMMI

3.2.3. Riskit Method

Riskit Method [32] es un método de administración de requerimientos que intenta evitar los problemas que presentan otras propuestas de gestión de riesgos en la ingeniería de software. Riskit Method nos provee de un proceso definido de administración de requerimientos que en términos generales es similar al definido por el CMMI. El valor que aporta para este trabajo es la conceptualización que se hace sobre el riesgo. El Riskit Method define al riesgo como una posibilidad de pérdida, la pérdida misma, o una característica, objeto o acción que está asociado con tal posibilidad. De acá define los elementos que componen a un riesgo:

- Factor de Riesgo: que representa un hecho conocido que influye en la ocurrencia del riesgo.
- Evento de Riesgo: que es la ocurrencia de un incidente con consecuencias negativas.
- Salida de Riesgo: es la situación resultante después de ocurrido el riesgo pero antes de que cualquier reacción haya sido realizada.

- Reacción: que representan las acciones correctivas que se toman una vez ocurrido el riesgo.
- Efecto del Riesgo: es la combinación del impacto del riesgo junto con reacciones que produce sobre el proyecto.
- Pérdida de Utilidad: que corresponde al daño que sufre el cliente o interesado del proyecto.

Por ejemplo, en el desarrollo de software un factor de riesgo puede ser el uso de nuevas tecnologías o la inestabilidad de los requerimientos. El evento de riesgo entonces puede corresponder a que el sistema falla, o que hay que redefinir los requerimientos, así la salida del riesgo sería una salida del sistema de producción, o que se necesita rehacer trabajo. La reacción entonces puede darse a través de un respaldo de los datos, para dejar el sistema operacional más tarde cuyo efecto pudiese resultar en un costo adicional o la pérdida de ciertas funcionalidades.

3.2.4. Administración de Riesgos ágil

Según la perspectiva ágil, el desarrollo de software debiese ser guiado por los riesgos y debe ser el mayor interés en decidir que lo que se realizará dentro de la siguiente iteración dentro de un proyecto, pues las funcionalidades que poseen un mayor riesgo debiesen realizarse primero una vez que el cliente determine las de mayor valor. Sin embargo, los métodos ágiles son poco explícitos a la hora de decir como se enfrentan, priorizan o se determinan los planes de acción frente a un riesgo.

La efectiva administración de riesgos involucra:

- Identificar los riesgos
- Analizar cada riesgo y determinar su severidad
- Priorizar los riesgos identificados basado en su severidad.
- Crear planes de acción para los riegos de más alta prioridad

- Monitorear y prever continuamente para asegurar que los planes de acción efectivamente mitigan los riesgos.

Para Ron Jeffries [34] la gestión de riesgos es una forma de apoyo a la priorización de las historias de usuario. En definitiva un riesgo es una historia de usuario que no ha sido establecida y que tiene un gran valor de negocio, pues permite la estimación de otras tareas, en otras palabras, permite saber cuanto se va a demorar el proyecto de mejor manera.

En la gestión de riesgos ágil se debe posibilitar la más amplia participación a la hora de identificar riesgos, lo que minimiza que hayan riesgos no identificados y tienen una mejor opción para ser aceptados. Además, el equipo entero es el que decide cuando un riesgo es eliminado. Para la evaluación de la severidad de un riesgo, la propuesta ágil se basa en la experiencia y conocimiento del equipo para analizar y priorizar los riesgos, usando sólo sus percepciones para determinar la severidad de cada riesgo.

3.2.5. Discusión

La administración de riesgos ha sido introducida en varias organizaciones bajo sus propios puntos de vista, sin embargo estos han estado basados en la intuición o la iniciativa individual, lo cual la hace inefectiva e inconsistente [32]. Según Kontio [32], los siguientes factores contribuyen al poco uso de métodos de administración de riesgos es que el riesgo como concepto es abstracto y confuso, los métodos de administración del riesgo esta basados en la cuantificación del riesgo para su análisis y los usuarios no son capaces de proveer una estimación suficientemente precisa, los riesgos tienen distintas implicaciones para los distintos interesados y pocos métodos toman en cuenta este aspecto, cada riesgo puede impactar al proyecto en distintas maneras.

En general, la administración de riesgo es considerada compleja o demasiado costosa, lo merma la posibilidad para las pequeñas y medianas empresas el considerarlas dentro de la organización. Sin embargo, la importancia que tiene es tan vital que puede determinar el éxito o el fracaso de un proyecto.

La administración de riesgo es una actividad que debe realizarse de forma obligatoria dentro de cada proyecto tal como lo indica el estándar de la IEEE 1074, para el ciclo de vida del desarrollo de software, y debe ser una actividad que debe estar presente a través de todo el proyecto.

En general, el proceso para la administración de riesgos está bien definido y es similar para ambas perspectivas.

4. Desarrollo de las herramientas

Las herramientas construidas tienen como propósito ser una alternativa que permita el trabajo compartido y colaborativo a través de una interfaz Web, tendrán las características de extensibilidad, para hacer una posible adaptación a la realidad de cada empresa a su forma de trabajo, e interoperabilidad, entendida como la capacidad para que distintas aplicaciones puedan acceder a los servicios expuestos por las herramientas. Las herramientas se construirán sobre una base de datos relacional, para evitar los problemas de manejo de datos que actualmente posee la alternativa de la empresa Kepler Technologies y se ocuparán herramientas de código abierto para que quede disponible al público en general y en particular al proyecto Tutelkán.

Se ha escogido la alternativa de ocupar un framework de desarrollo para la implementación de las herramientas debido a que permite heredar las buenas prácticas con que fue construido, lo que permite construir un mejor código para la aplicación y debido a que en general disminuye el tiempo de programación para el desarrollador. En el anexo 7.4, se presentan las distintas alternativas analizadas.

Para el desarrollo se presentarán las funcionalidades que poseen las herramientas, su diseño e implementación, para finalmente mostrar los resultados obtenidos junto con su evaluación. La evaluación se hará analizando el grado de cumplimiento que tienen las herramientas con respecto a los objetivos planteados por el CMMI junto con las prácticas ágiles que se decidieron adoptar mostrando las ventajas que tienen con respecto a las alternativas que ya existen. Adicionalmente se mostrará la evaluación con empresas que usaron las herramientas presentando el valor generado para ellas.

4.1. Administración de Requerimientos

Uno de los principales puntos de conflicto entre las metodologías tradicionales y las ágiles es la documentación. Para nuestra aplicación el fundamento de tenerla es que nos posibilita la administración de forma más eficiente, tener los requerimientos dentro de una aplicación nos hace más fácil la visualización de todos ellos en conjunto, su búsqueda y organización que definidas en papel. Además nos permite un canal compartido para realizar evaluación, en el sentido que lo escrito dentro de la aplicación refleje lo entendido por ambas partes y que se definan de forma explícita las condiciones de satisfacción para cada requerimiento.

La herramienta de administración de requerimientos se basa en la administración ágil de requerimientos, esto es, maneja simplemente una lista de requerimientos donde los primeros que aparecen representan los requerimientos de mayor prioridad por el cliente y más abajo lo de menor prioridad. En cualquier momento el cliente puede agregar, eliminar o repriorizar un requerimiento. Además se seguirá el modelo INVEST definido por Cohn [17], donde las historias de usuario (para nuestro caso haremos un símil entre estas y un requerimiento) deben ser Independientes, Negociables, Valiosas, Estimables, Small (Pequeñas), Testeables, por lo que no se manejarán relaciones entre las historias de usuario, presentarán un atributo en que se puede definir el esfuerzo para realizarla y se relacionarán con uno o más tests de aceptación definidos por el cliente.

4.1.1. Funcionalidades

Según Fiksel [20] los criterios que debe reunir cualquier sistema de administración de requerimientos son:

- Que sirva la necesidad de múltiples perspectivas y roles concurrentemente
- Soporte de múltiples formas de representaciones de requerimientos
- Soporte de conversión de documentación para importar y exportar

- Soporte para la ubicación de productos y casos de prueba derivados de los requerimientos
- Soporte para el control de cambios y administración de versiones
- Operar en múltiples plataformas
- Soporta integración con arquitecturas y plataformas emergentes

Dada la arquitectura que hemos escogido, muchas de las características anteriormente mencionadas se tienen, por ejemplo la de soporte de múltiples representaciones, operar en múltiples plataformas e integración con arquitecturas y plataformas emergentes. La parte de soporte de documentación para importar y exportar nuestra herramienta se entiende como distintos clientes que intentan utilizar la interfaz definida por los servicios. Sin embargo para esta primera versión, no se considera el soporte para el control de cambios, si bien puede representar una característica importante, vimos que el manejo de cambios para las metodologías ágiles se hace de forma sencilla, agregando o eliminando requerimientos de la lista.

La herramienta presentará entonces, las siguientes funcionalidades:

- Se soportará administración de requerimientos para múltiples proyectos.
- Visualizar los requerimientos definidos para un proyecto, de forma que los requerimientos de mayor prioridad se presentarán al comienzo de la lista.
- Crear, eliminar, modificar requerimientos.
- Para cada requerimiento se podrán crear uno o más test de aceptación definidos por el cliente.
- Se podrán visualizar los tests de aceptación asociados a un requerimiento.
- Se podrá modificar y eliminar tests de aceptación asociados a un requerimiento.

4.1.2. Diseño

Para nuestra aplicación tenemos dos elementos de datos que modelar. Estos son los requerimientos de cliente y los tests de aceptación.

Los atributos de un requerimiento de cliente se describen en la Tabla 4.

Nombre Atributo	**Descripción**
Nombre	Indicará el nombre del requerimiento, que debe describir la funcionalidad pedida por el cliente.
Descripción	Constituye una descripción más detallada del requerimiento, la cual puede ser modificada en la medida en que el equipo de desarrollo conozca más del problema de negocio que se está resolviendo
Prioridad	Este atributo permite determinar el valor que aporta al cliente el requerimiento
Impacto	Será la medida del impacto que tendrá el requerimiento
Estimación	corresponde a la estimación de esfuerzo por parte del equipo de desarrollo para completar el requerimiento
Estado	Los estados se mostrarán la evolución que tiene un requerimiento dentro del desarrollo del proyecto
Tipo	El tipo indicará una característica del requerimiento. Por defecto se tendrán los tipos Funcional y No funcional
Pedido Por	Identifica el cliente que ha pedido el requerimiento
Responsable	Indica quién será el responsable dentro del equipo de desarrollo para la concreción del requerimiento

Tabla 4: Modelo para un requerimiento de cliente

Los atributos para representar los test de aceptación se describen en la Tabla 5:

Nombre Atributo	**Descripción**
Nombre	Nombre del test de aceptación
Descripción	Indica una descripción de la prueba por el cliente que desea realizar.
Configuración	Indica al ambiente que debe existir para que la prueba se realice
Entrada	Indica los datos que serán ingresados al sistema bajo el ambiente especificado. Esto determina el flujo de acciones que puede realizar un usuario sobre el sistema junto con los datos
Salida Esperada	Indica la salida esperada por el sistema una vez ingresado los datos de entrada

Tabla 5: Modelo para un test de aceptación

Diseño Arquitectónico

El diseño arquitectónico estará dado por el estilo de arquitectura REST, el cual describe un sistema de red en términos de elementos de datos (recurso, identificador del recurso, representación), conectores (cliente, servidor, cache) y componentes (servidor de origen, gateways, proxy).

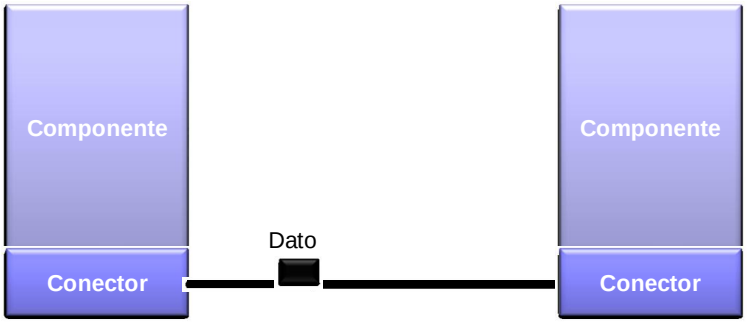


Ilustración 3: Esquema REST

Las principales características de REST son:

- Los elementos de datos son accesados a través una interfaz estándar
- Los componentes se comunican a través de la transferencia de *representaciones* (documentos que efectivamente contienen la información) de los *recursos* (fuente específica de información) a través de esta interfaz en vez de operar directamente con el recurso.
- Los conectores presentan una interfaz abstracta de comunicación, escondiendo los detalles de implementación y mecanismos de comunicación.
- Los componentes usan conectores para acceder, proveer acceso o mediar el acceso a los recursos
- Todas las peticiones hechas a través de conectores deben contener toda la información necesaria para el entendimiento de dicha petición, sin depender de alguna petición anterior. Esto es lo que le da el carácter de "Sin Estado".

Para realizar un desarrollo basado en la arquitectura REST debemos entonces responder las siguientes preguntas: ¿Cuáles son los recursos?, ¿Cuáles son las representaciones?, ¿Cuáles son los métodos soportados para cada recurso?, ¿Cuáles son los códigos de estado de respuesta como retorno? Para mayor información, consultar el Anexo 7.3.

Dado que ocuparemos el estilo de arquitectura REST, se hablará de servicios expuestos y clientes que consumen esos servicios. La comunicación se dará tanto entre cliente y servicio, como entre servicios, a través del protocolo HTTP y los servicios se comunicarán con una base de datos relacional para el almacenamiento de los datos.

Para la herramienta de administración de requerimientos existirán los servicios de:

- Proyectos: Encargado de la administración de proyectos. Esto es agregar, modificar, eliminar y ver los proyectos de una organización.

- Requerimientos de Cliente: Encargado de la administración de los requerimientos dentro de un proyecto y que provee las mimas características de agregar, modificar, eliminar y visualizar los requerimiento de un proyecto. Adicionalmente a través de este servicio podrá crear un proyecto y relacionarlo directamente con un proyecto.

- Test de Aceptación: Los tests de aceptación corresponden a la descripción de las condiciones de satisfacción que deben cumplir los requerimientos por parte del cliente para ser aceptados. Su servicio es similar al de requerimientos.

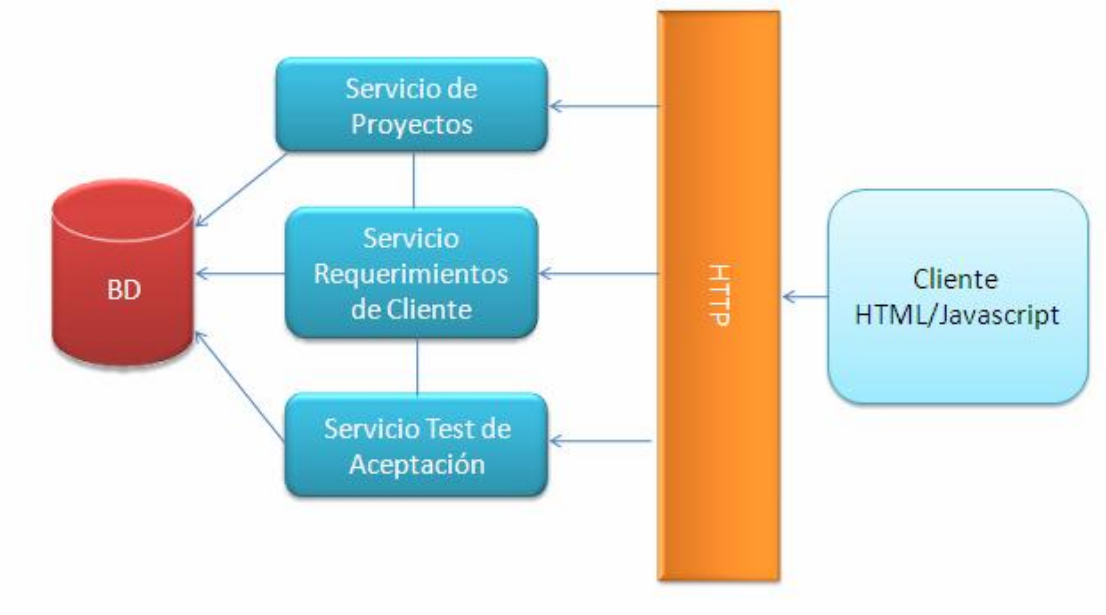


Ilustración 4: Arquitectura para la herramienta de Administración de Requerimientos

El cliente será HTML/Javascript el contendrá la interfaz de usuario y consumirá los servicios antes mencionados.

4.1.3. Implementación

Detallaremos la implementación respondiendo las preguntas que se deben hacer cuando se realiza una aplicación siguiendo la arquitectura REST.

¿Cuáles son los recursos?

Los recursos serán los proyectos, requerimientos de cliente y los test de aceptación.

¿Cuáles son las representaciones?

Inicialmente todos los servicios provistos por la aplicación transmitirán representaciones en formato XML, el cual deberá ser consumido por las aplicaciones clientes.

¿Cuáles son los métodos?

Como regla general, los servicios tendrán soporte para los cuatro verbos base que provee el protocolo HTTP (GET, POST, PUT, DELETE). Los métodos y las URI de acceso a los recursos de esquematizan en las tablas 1, 2 y 3.

URI (Identificador)	Método	Representación	Descripción
/projects	GET	XML	Obtiene todos los proyectos que residen en la aplicación
/projects/:id	GET	XML	Obtiene el proyecto que posee el identificador :id
/projects	POST	XML	Agrega un proyecto a la aplicación
/projects/:id	PUT	XML	Realiza una modificación al proyecto con identificador :id
/projects/:id	DELETE	-	Elimina el proyecto con identificador :id de la aplicación

Tabla 6: API REST para la manipulación de Proyectos

URI (Identificador)	Método	Representación	Descripción
/clientrequirements	GET	XML	Obtiene todos los requerimientos de cliente de la aplicación.
/:project/clientrequirements	GET	XML	Obtiene todos los requerimientos asociados al proyecto :Project
/clientrequirements/:id	GET	XML	Obtiene el requerimiento de cliente que posee el id :id
/:project/clientrequirements/:id	POST	XML	Agrega un requerimiento de cliente asociado a un proyecto
/clientrequirements/:id	PUT	XML	Modifica un requerimiento de cliente con identificador :id

Tabla 7: API REST para la manipulación de requerimientos de cliente

URI (Identificador)	Método	Representación	Descripción
/acceptancetests	GET	XML	Obtiene todos los test de aceptación dentro de la aplicación
/acceptancetests/:id	GET	XML	Obtiene el test de aceptación con identificador :id
/:project/acceptancetests	GET	XML	Obtiene todos los requerimientos de cliente asociados a un proyecto
/:project/:clientrequirement/acceptancetests	GET	XML	Obtiene todos los tests de aceptación del proyecto :Project asociados al requerimiento de cliente :clientrequeirement
/:project/acceptancetests/:id	POST	XML	Agrega un test de aceptación asociado al proyecto :Project

Tabla 8: API REST para la manipulación de Test de Aceptación

¿Cuáles son los códigos de estado de respuesta?

Los códigos de estado para las respuestas serán:

- 200 OK: Para cuando se ocupe el método GET, o para indicar el éxito al agregar un recurso
- 204 No content: Cuando se elimine algún recurso de forma exitosa.
- 404 Not Found: Cuando no se encuentre un recurso dentro de la aplicación o no se referencia a una URI de la API del sistema.
- 401 Bad Request: Cuando se intente agregar un recurso y no se adhiera al formato que entiende la aplicación
- 500 Internal Server Error: Cuando ocurra algún error dentro del servidor.

4.1.4. Resultados

Gracias a la elección de la arquitectura tenemos una herramienta de administración de requerimientos que tiene como principales características de implementación su extensibilidad y su interoperabilidad. En los resultados presentamos las interfaces, que muestran en la práctica cómo la extensibilidad y la interoperabilidad se dan en nuestra plataforma.

4.1.4.1. Interfaz de Proyectos

La interfaz principal muestra la lista de proyectos agrupados por el atributo cliente. La interfaz permite además agrupar por cualquier atributo de proyecto. En la parte superior, se muestran las acciones sobre un proyecto. En este caso, son visitar, que permite visualizar los requerimientos y eliminar el proyecto seleccionado.

En el costado derecho se muestra el formulario que permite crear un nuevo proyecto.

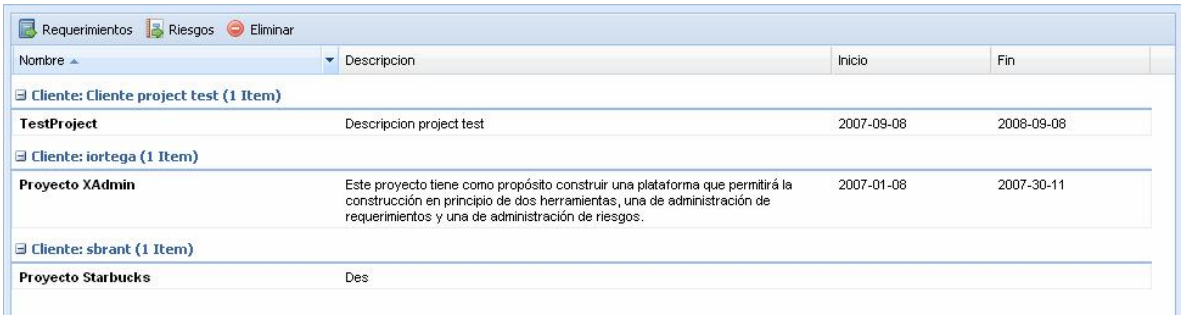


Ilustración 5: Lista de Proyectos agrupados por cliente.

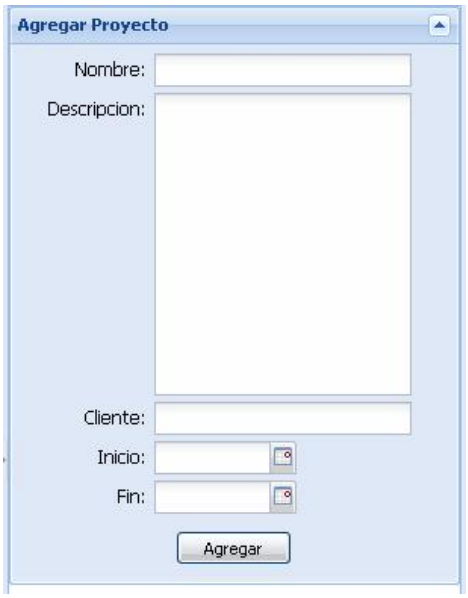


Ilustración 6: Formulario que permite agregar un proyecto

4.1.4.2. Interfaz de Requerimientos

En la interfaz principal de requerimientos se muestra la lista de requerimientos agrupados por prioridad, de forma que queden los de más alta prioridad al comienzo. En el ejemplo se muestran los requerimientos del proyecto XAdmin que corresponde a esta aplicación. Las operaciones que se pueden realizar sobre los requerimientos en la cabecera de la lista y el proyecto seleccionado en la cabecera de la aplicación.

En el costado derecho, se muestran indicadores sobre el actual estado de los requerimientos. Por ejemplo, el número total de requerimientos, o el número de requerimientos que no tienen un test de aceptación asociado.

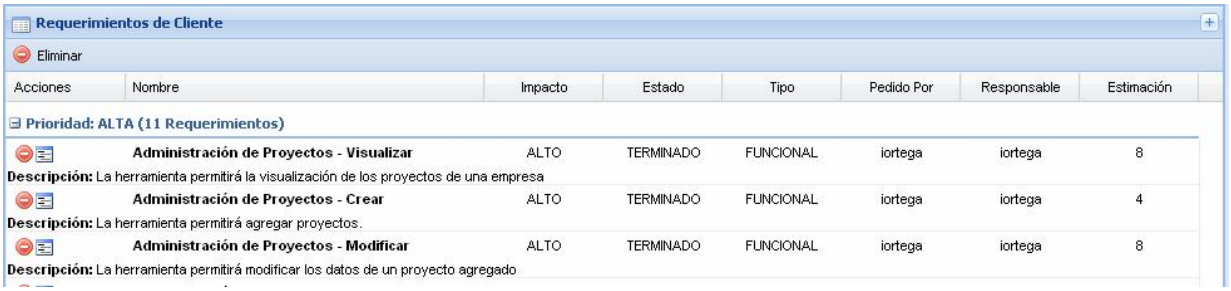


Ilustración 7: Lista de Requerimientos

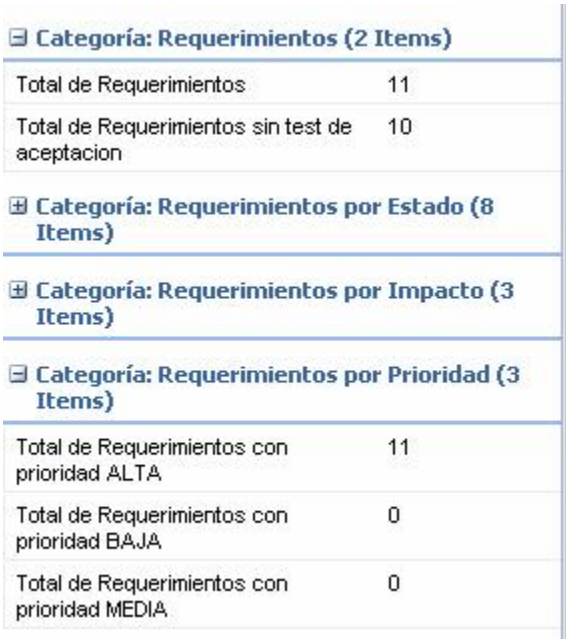


Ilustración 8: Estado de los requerimientos

Una vez seleccionado un requerimiento se puede ver su detalle donde es posible su edición y donde se muestran los elementos relacionados con este.

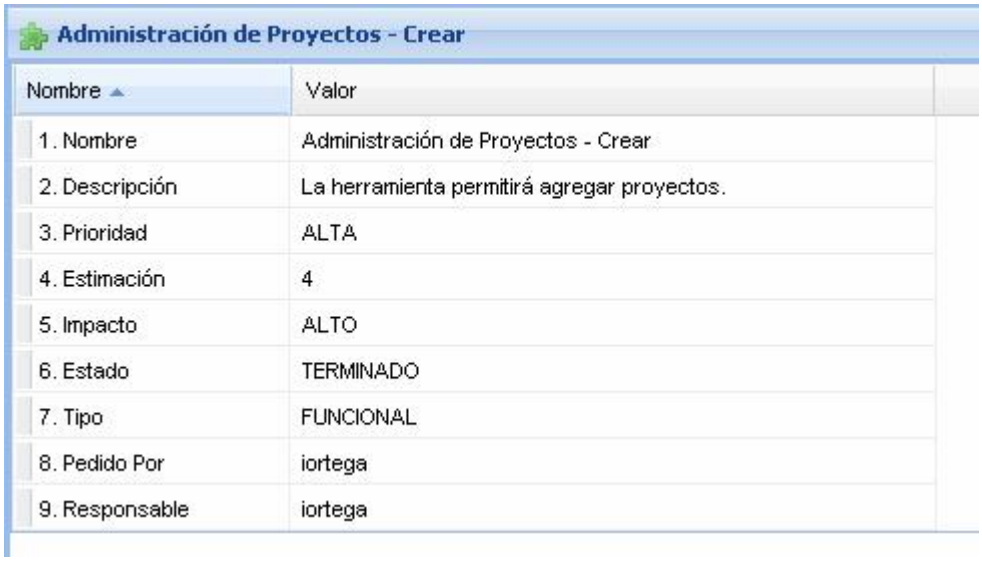


Ilustración 9: Detalle de un Requerimiento

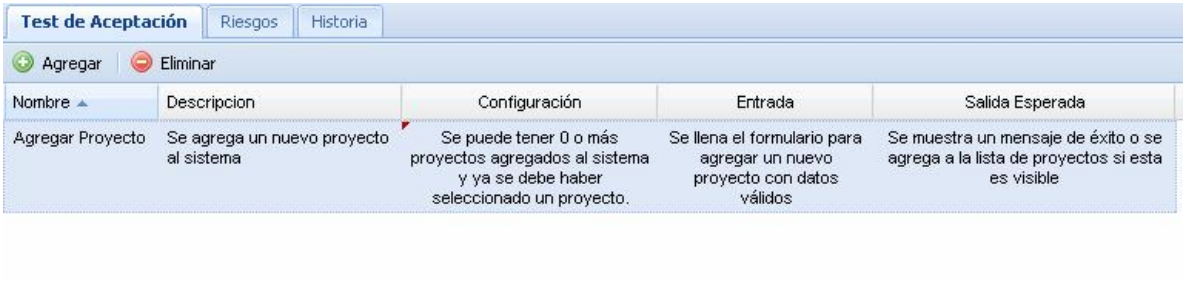


Ilustración 10: Elementos asociados a un requerimiento

4.1.4.3. Interoperabilidad

Mostramos como dos aplicaciones diferentes se pueden comunicar a través la API REST definida. Estas dos aplicaciones corresponden al mismo cliente Web mostrado anteriormente y a un cliente en Excel que permite la visualización de los requerimientos.

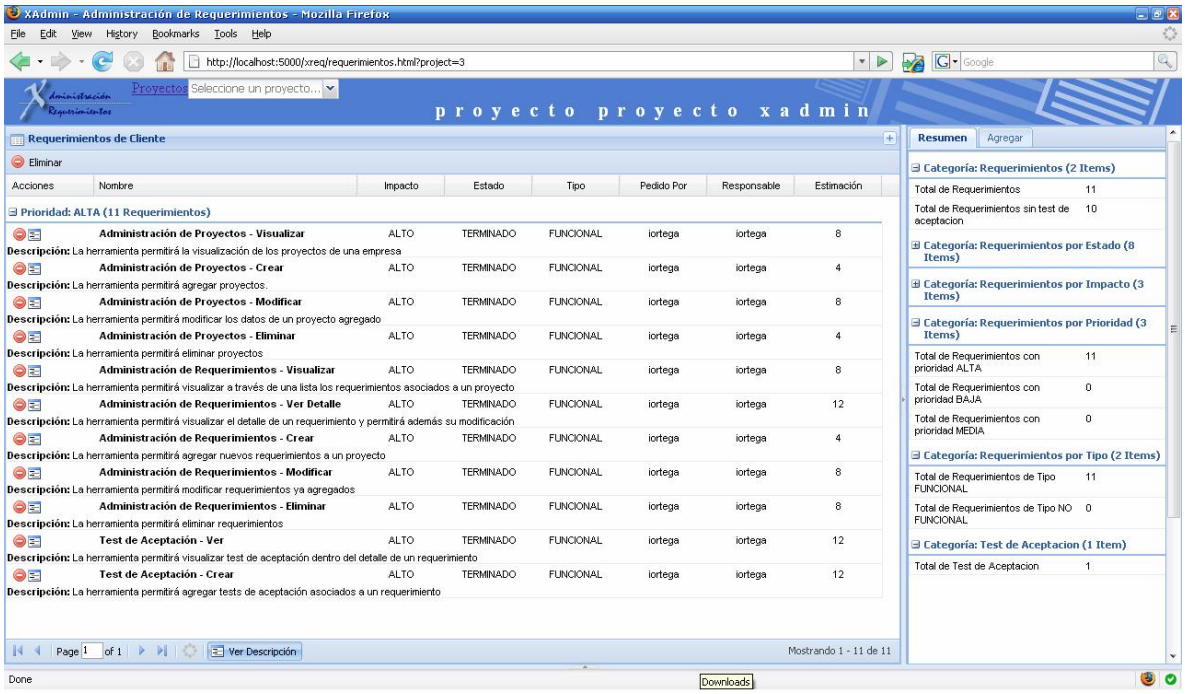
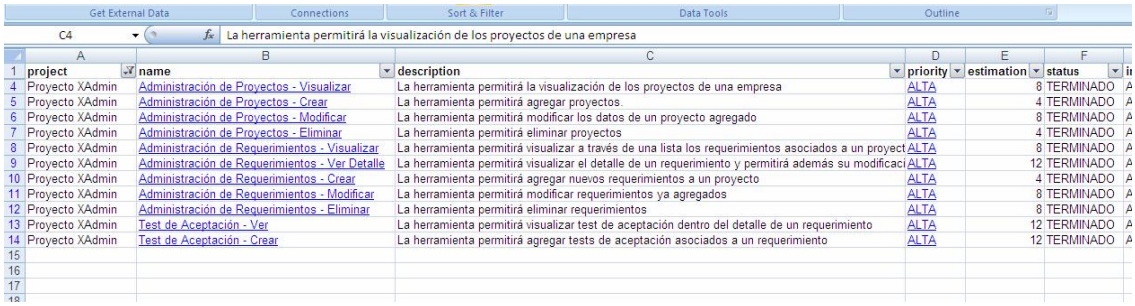


Ilustración 11: Cliente Web. Herramienta de Administración de requerimientos



	project	name	description	priority	estimation	status
4	Proyecto XAdmin	Administración de Proyectos - Visualizar	La herramienta permitirá la visualización de los proyectos de una empresa	ALTA	8	TERMINADO
5	Proyecto XAdmin	Administración de Proyectos - Crear	La herramienta permitirá agregar proyectos.	ALTA	4	TERMINADO
6	Proyecto XAdmin	Administración de Proyectos - Modificar	La herramienta permitirá modificar los datos de un proyecto agregado	ALTA	8	TERMINADO
7	Proyecto XAdmin	Administración de Proyectos - Eliminar	La herramienta permitirá eliminar proyectos	ALTA	4	TERMINADO
8	Proyecto XAdmin	Administración de Requerimientos - Visualizar	La herramienta permitirá visualizar a través de una lista los requerimientos asociados a un proyect	ALTA	8	TERMINADO
9	Proyecto XAdmin	Administración de Requerimientos - Ver Detalle	La herramienta permitirá visualizar el detalle de un requerimiento y permitirá además su modificaci	ALTA	12	TERMINADO
10	Proyecto XAdmin	Administración de Requerimientos - Crear	La herramienta permitirá agregar nuevos requerimientos a un proyecto	ALTA	4	TERMINADO
11	Proyecto XAdmin	Administración de Requerimientos - Modificar	La herramienta permitirá modificar requerimientos ya agregados	ALTA	8	TERMINADO
12	Proyecto XAdmin	Administración de Requerimientos - Eliminar	La herramienta permitirá eliminar requerimientos	ALTA	8	TERMINADO
13	Proyecto XAdmin	Test de Aceptación - Ver	La herramienta permitirá visualizar test de aceptación dentro del detalle de un requerimiento	ALTA	12	TERMINADO
14	Proyecto XAdmin	Test de Aceptación - Crear	La herramienta permitirá agregar tests de aceptación asociados a un requerimiento	ALTA	12	TERMINADO

Ilustración 12: Cliente Excel. Herramienta de Administración de Requerimientos

4.1.4.4. Extensibilidad

La extensibilidad de nuestra herramienta se manifiesta en la medida que se construyan nuevas API que provean de nuevos servicios y clientes que consuman estos nuevos servicios, todo esto con el fin de poder adaptar la forma de trabajo que tienen o que pretenden tener las empresas, puesto que la premisa es que las herramientas se adaptan y no

los usuarios a las herramientas. Un ejemplo que construimos fue a partir de la herramienta que posee actualmente la empresa Kepler Technologies.

La forma de administrar requerimientos de Kepler es a través de la definición de tres niveles: requerimientos de cliente, requerimientos de sistema y productos, que corresponden al software que se produce durante el desarrollo. La aplicación que poseen administra además las relaciones que se dan entre estos niveles, en un curso normal se deberían definir los requerimientos de cliente para luego traducirlos en especificaciones más técnicas, que corresponde a los requerimientos de sistema, los que a su vez generan los productos a ser entregados al cliente. Además posee la capacidad de mostrar la trazabilidad que existe entre ellos a través de una matriz y que constituye un elemento importante en la evaluación de impacto cuando se hace algún cambio en el requerimiento. Actualmente ellos no definen explícitamente los tests de aceptación para los requerimientos de cliente dentro de su aplicación, ni tampoco los casos de prueba para sus requerimientos de sistema.

La extensión que se hizo de nuestra herramienta contempla no sólo los tres niveles que define la empresa para administrar sus requerimientos y la trazabilidad entre ellos, sino también provee de la definición explícita de los tests de aceptación y casos de prueba. Además que provee la interfaz para administrar requerimientos para múltiples proyectos.

No se detalla la API de los servicios agregados ya que son muy similares a las anteriores y presentamos en las siguientes ilustraciones las interfaces del cliente Web.

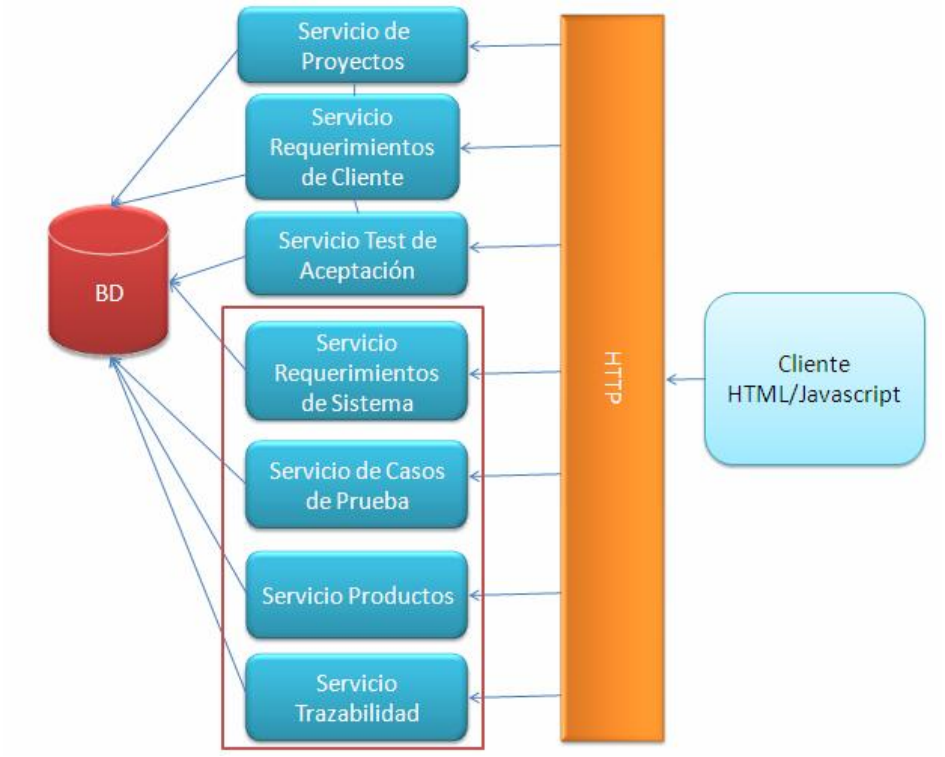


Ilustración 13: Servicios agregados a la aplicación

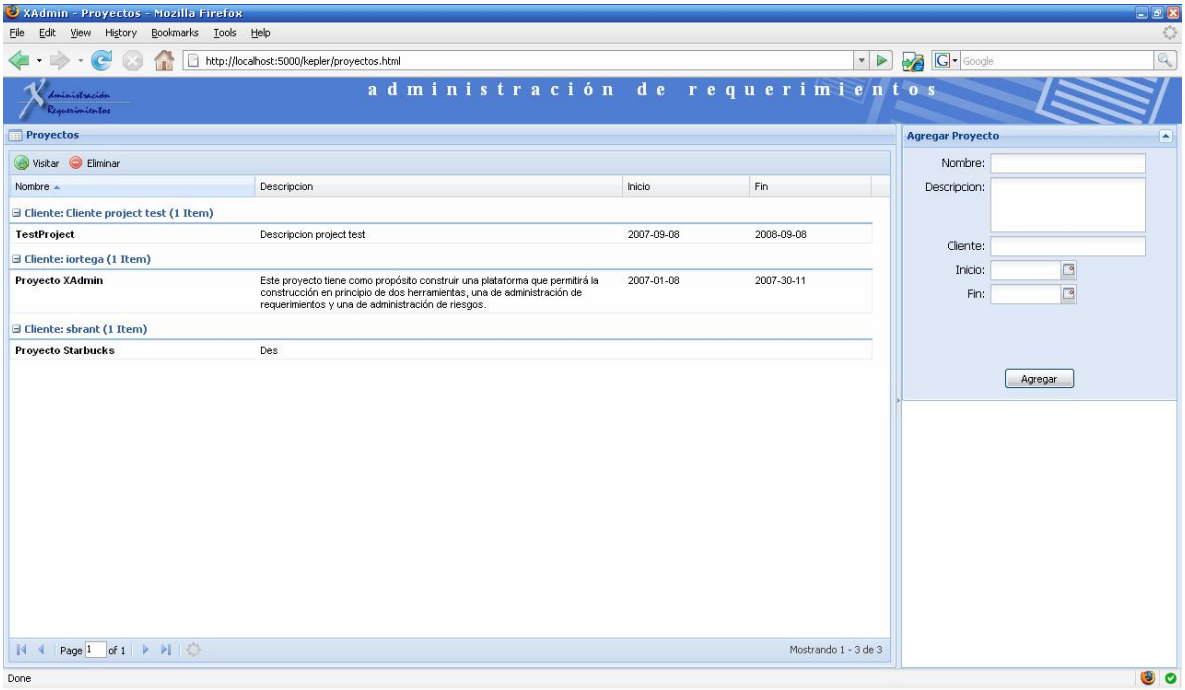


Ilustración 14: Interfaz de administración de proyectos

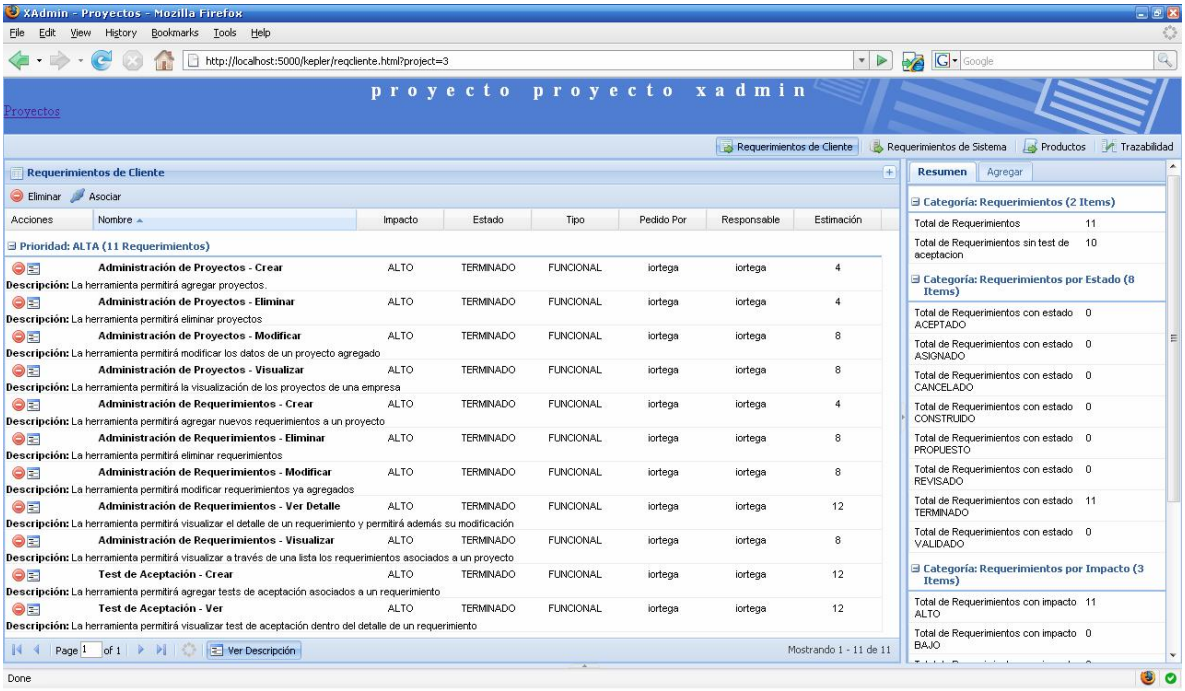


Ilustración 15: Interfaz de Administración de Requerimientos de Cliente

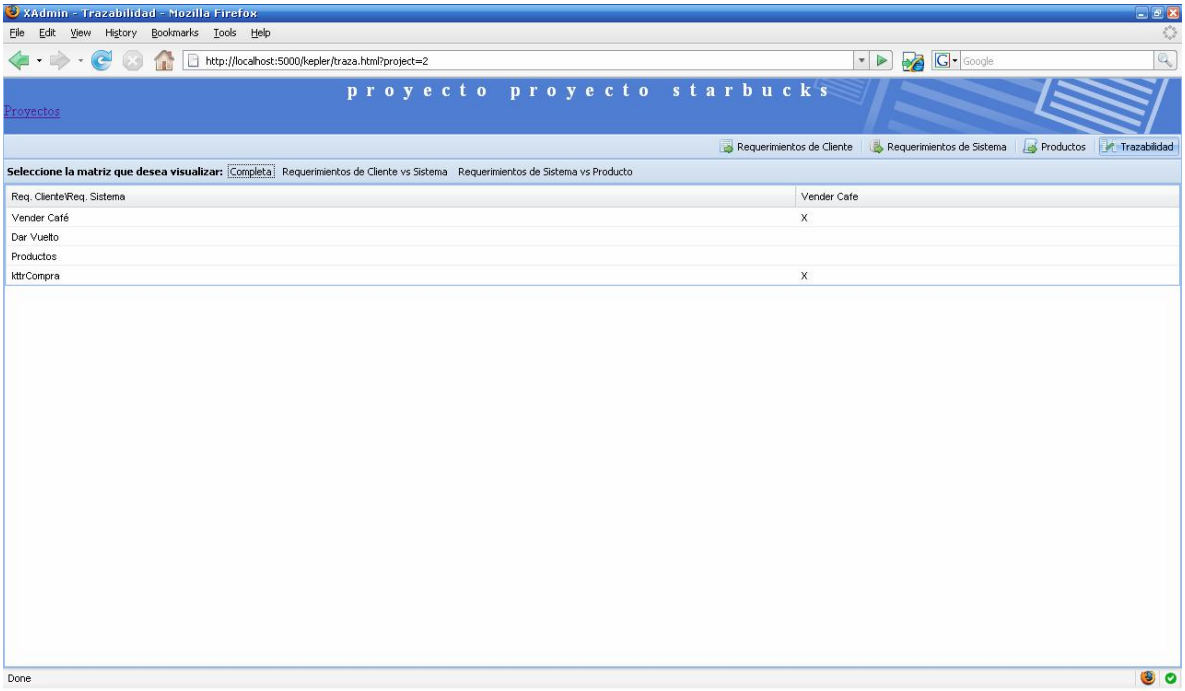


Ilustración 16: Interfaz que muestra la trazabilidad entre requerimientos y productos

4.1.5. Evaluación

Nuestra primera evaluación consistirá el grado de cumplimiento de la herramienta con respecto a los objetivos definidos por el CMMI en el área de administración de requerimientos para luego presentar la evaluación hecha por las empresas usuarios de la herramienta.

4.1.5.1. Evaluación con respecto a los objetivos

Objetivo	Justificación	Cumplimiento
OE 1: Preparase para la Administración de Riesgos		
PE 1.1: Obtener entendimiento de los requerimientos	La herramienta permite visualizar los requerimientos de forma que los de mayor prioridad se encuentran al principio y muestra sus detalles, además permite que se puedan ingresar los tests de aceptación de los requerimientos, que representarán las condiciones de satisfacción del requerimiento. Esto debiese ser una buena base para revisar si los requerimientos se han entendido por el equipo de desarrollo.	ALTO
PE 1.2: Obtener Compromiso en los requerimientos	El compromiso según las metodologías ágiles se hace de forma que para un tiempo determinado se entrega el software que entrega el mayor valor al cliente y esto se hace de continua y frecuente. El software que entrega mayor valor está determinado por la prioridad que le da el cliente al requerimiento y la aplicación muestra los de mayor prioridad al comienzo, lo que permite que el equipo tengo buena visibilidad por los requerimientos que debiesen ser implementados primero y de esa forma obtener el compromiso con el cliente.	MEDIO
PE 1.3: Administrar los cambios en los requerimientos	Desde una perspectiva ágil de desarrollo hemos visto que los cambios en los requerimientos simplemente se manejan de forma se agregan a la pila de requerimientos según la prioridad determinada por el cliente, el cliente entonces debe priorizar los requerimientos de acuerdo al marco de tiempo que tiene el equipo de desarrollo para trabajar.	MEDIO
PE 1.4: Mantener la trazabilidad bidireccional de los requerimientos	La trazabilidad que mantiene la herramienta es la dada por los requerimientos y sus tests de aceptación. Para el CMMI esto debiese darse hasta el nivel de productos, que corresponden al entregable dado al cliente. La extensión hecha de la aplicación que se adecua a la forma de trabajo de la empresa Kepler Technologies proporciona una implementación de cómo esto pudiese ocurrir.	BAJA
PE 1.5: Identificar inconsistencias entre el trabajo del proyecto y los requerimientos.	Las inconsistencias a identificar según el CMMI se dan a nivel de requerimientos con productos de trabajo y planes del proyecto. La herramienta se enfoca a una administración de requerimientos que debiese ser la base para la planificación de un proyecto, sin embargo no	BAJA

	contempla la parte de generar o proveer una interfaz para realizar la planificación del proyecto y tampoco sobre los productos de trabajo que corresponde al software construido por el equipo de desarrollo y que debiese estar relacionada al plan de trabajo. Son dos partes que el CMMI cubre con otras áreas de proceso y que no constituye el alcance de este trabajo.	

Tabla 9: Evaluación con respecto a los objetivos del CMMI.

Por otra parte podemos ver como la herramienta se adhiere al modelo INVEST antes mencionado. Recordemos que INVEST es acrónico de Independent (independientes), Negotiable (negociables), Valuable (valorables), Estimable (estimables), Small (pequeñas), Testable (testeables), que son las características que debiese reunir toda historia de usuario en un desarrollo ágil.

Objetivo	**Justificación**	**Cumplimiento**
Independientes	Los requerimientos debiesen ser escritos de forma independiente conceptualmente de forma que se permita implementar y planificar de forma independiente. La herramienta posibilita esto debido a que no permite relaciones entre los requerimientos, si bien esto no se niega el valor que esto podría tener en ciertas circunstancias, las organizaciones debiesen ver la necesidad de agrupar los requerimientos bajo algún criterio. En general, escoger un criterio de agrupación depende de quién especifica un requerimiento, por lo que a priori la herramienta no provee de ningún tipo de relación entre requerimientos. Sin embargo, como posible extensión o trabajo futuro se puede otorgar un sistema que permita la relación de requerimientos definida por la necesidad propia de una organización.	ALTO
Negociables	Esto significa que los requerimientos no son un contrato y que su detalle se determinará durante el desarrollo. La aplicación permite que el detalle sobre los requerimientos se hagan a medida que avanza el trabajo y el equipo conoce más sobre el problema de negocio a resolver.	ALTO
Valiosos	El sentido de valor está determinado por el cliente y para nuestra aplicación se manifiesta por la prioridad que le otorga el cliente.	ALTO
Estimable	Los requerimientos son estimables, esto se materializa a través de unos de los atributos de los requerimientos que permite medir el esfuerzo sobre el requerimiento.	ALTO

Pequeñas	Los requerimientos debiesen ser pequeños, representando pocas horas de trabajo para los desarrolladores, esto se puede verificado a través de las estimaciones que se hagan sobre los requerimientos.	ALTO
Testeables	La herramienta permite que esto se haga explicito durante la escritura de los requerimientos, puesto que permite asociar uno o más test de aceptación a un requerimiento.	ALTO

Tabla 10: Evaluación respecto al modelo INVEST.

4.1.5.2. Evaluación con usuarios

La evaluación con los usuarios consistió en establecer cuál era el estado de la administración de requerimientos por parte de las empresas, que tipo de herramientas ocupaban y cuáles eran las características que más le satisfacían y cuáles eran las que necesitaban. Luego se presenta la herramienta, para su uso, para finalmente hacer una evaluación de las características de la herramienta.

Las empresas que evaluaron la herramienta concluyeron que la interfaz presentada es clara e intuitiva, permite la gestión de requerimientos de acuerdo a sus objetivos y además destacaron el valor que tenía la definición explícita de los test de aceptación.

Una de las posibles mejora que detectaron las empresas usuarias es la posibilidad de manejar control de versiones de los requerimientos, para poder llevar de forma clara un historial de cambios, y la posibilidad de definir distintos usuarios y roles para poder definir interfaces de acuerdo a ellos, por ejemplo para el jefe de proyecto, tener una interfaz que muestre sólo los proyectos en que él este a cargo. Estas posibles son aspectos que se tuvieron en consideración pero que simplemente se decidió por no abordarlas para este trabajo.

4.2. Administración de Riesgos

En general, la administración de riesgos es un proceso mejor definido que la administración de requerimientos, y que tanto para la propuesta de CMMI, Riskit Method y la perspectiva ágil reúnen características comunes que se ocuparán para el modelo de un riesgo.

Para la aplicación, a cada riesgo se le asociará uno o más impactos, que indicarán los efectos que tiene el riesgo una vez que ocurra, uno o más indicadores, que representan las situaciones en que se puede manifestar un riesgo, una o más estrategias de mitigación, que son acciones disminuir la probabilidad de que el riesgo ocurra y uno o más planes de contingencia, que indican las acciones correctivas una vez que el riesgo se ha manifestado.

4.2.1. Funcionalidades

Las funcionalidades para las herramientas de administración de riesgos serán:

- Visualizar la lista de riesgos asociados a un proyecto.
- Agregar, eliminar y modificar riesgos.
- Para cada riesgo:
    - Visualizar lista de impactos asociados. Agregar, eliminar y modificar impactos.
    - Visualizar lista de indicadores asociados. Agregar, eliminar y modificar indicadores.
    - Visualizar lista de estrategias de mitigación. Agregar, eliminar y modificar estrategias de mitigación.
    - Visualizar lista de planes de contingencia, Agregar, eliminar y modificar planes de contingencia.
- Mostrar un resumen de los riesgos. Esto es presentar información agregada sobre los riesgos.

4.2.2. Diseño

El modelo del riesgo se describe en la Tabla 11:

Nombre Atributo	Descripción
Nombre	Nombre del riesgo
Descripción	Constituye una descripción más detallada del riesgo
Probabilidad	Corresponde a la probabilidad que se estima la ocurrencia del riesgo. El rango de valores corresponde al intervalo [0,1]
Impacto	Corresponde a una estimación cuantitativa del impacto si llegase a ocurrir el riesgo. El rango de valores corresponde al intervalo [0,10]
Exposición	Será el producto entre la probabilidad y el impacto y será utilizado como elemento de priorización para la asignación de recursos para la gestión del riesgo
Impactos	Corresponden a la descripción de los impactos que tiene el hecho de que ocurriese el riesgo
Indicadores	Corresponden a una lista de indicadores que señalan cuando se puede manifestar el riesgo
Estrategia de Mitigación	Corresponden a las estrategias a llevar para que el riesgo no ocurra
Medidas de Contingencia	Son las medidas de acción cuando ha ocurrido el riesgo

Tabla 11: Modelo para un riesgo

La arquitectura para esta aplicación presenta los servicios de riesgos, que permite visualizar la lista o un riesgo, además de agregar, eliminar y modificar un riesgo y los de impactos, indicadores, estrategias de mitigación y planes de contingencia, que permiten visualizar la lista de ellos asociados a un riesgo, además de agregar, eliminar y modificarlos. La Ilustración 17 muestra la esquematización de arquitectura de la herramienta.

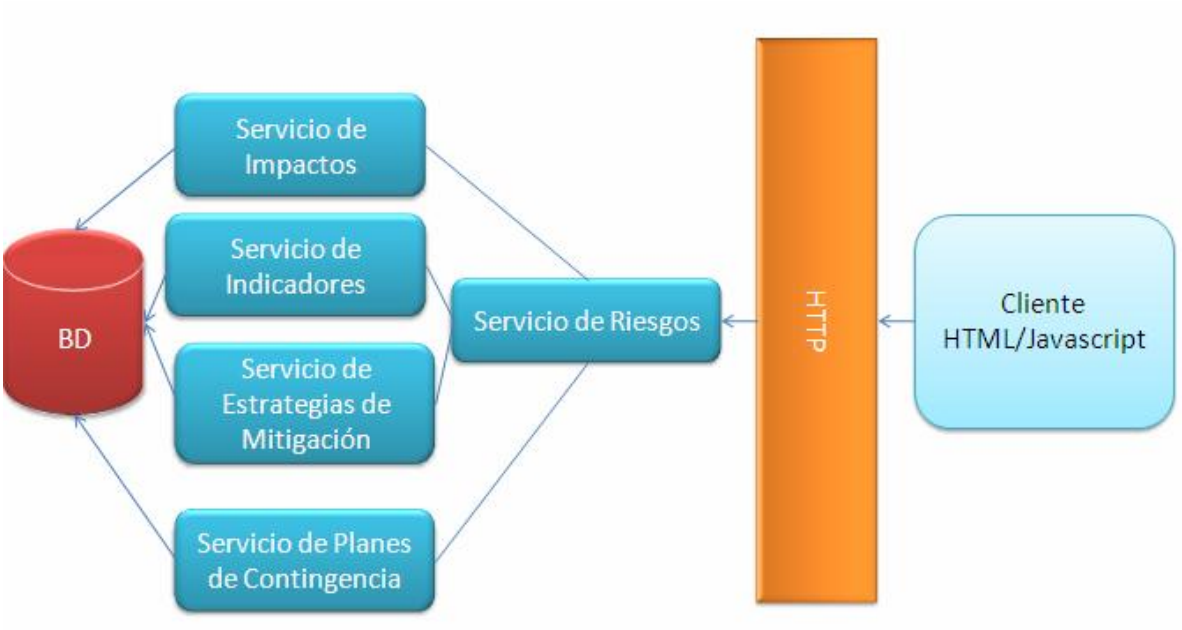


Ilustración 17: Arquitectura de la herramienta de Administración de Riesgos

4.2.3. Implementación

Mostramos a continuación el detalle de la arquitectura respondiendo las preguntas para adherirse a la arquitectura REST.

¿Cuáles son los recursos?

Los recursos son los riesgos, impactos, estrategias de mitigación, indicadores y medidas de contingencia.

¿Cuáles son las representaciones?

Al igual que la herramienta anterior para las representaciones se ocupará el formato XML.

¿Cuáles son los métodos?

Presentamos el detalle de los métodos con las URIs en las tablas 7, 8, 9, 10 y 11.

URI (Identificador)	Método	Representación	Descripción
/risks	GET	XML	Obtiene todos los riesgos que residen en la aplicación
/risks/:id	GET	XML	Obtiene el riesgo que posee el identificador :id
/risks	POST	XML	Agrega un riesgo a la aplicación
/risks/:id	PUT	XML	Realiza una modificación al riesgo con identificador :id
/risks/:id	DELETE	-	Elimina el riesgo con identificador :id de la aplicación

Tabla 12: API REST para Riesgos

URI (Identificador)	Método	Representación	Descripción
/impacts	GET	XML	Obtiene todos los impactos de cliente de la aplicación.
/:risk/impacts	GET	XML	Obtiene todos los impactos asociados al riesgo :risk
/impacts/:id	GET	XML	Obtiene el impacto que posee el identificador :id
/:risk/impacts	POST	XML	Agrega un impacto asociado a un riesgo
/impacts/:id	PUT	XML	Modifica un impacto con identificador :id

Tabla 13: API REST para los impactos asociados a un riesgo

URI (Identificador)	Método	Representación	Descripción
/indicators	GET	XML	Obtiene todos los indicadores de cliente de la aplicación.
/:risk/indicators	GET	XML	Obtiene todos los indicadores asociados al riesgo :risk
/indicators /:id	GET	XML	Obtiene el indicador que posee el identificador :id
/:risk/indicators	POST	XML	Agrega un indicador asociado a un riesgo
/indicators /:id	PUT	XML	Modifica un indicador con identificador :id

Tabla 14: API REST para los indicadores asociados a un riesgo

URI (Identificador)	Método	Representación	Descripción
/strategies	GET	XML	Obtiene todos los estrategias de cliente de la aplicación.
/:risk/strategies	GET	XML	Obtiene todos los estrategias asociados al riesgo :risk
/strategies/:id	GET	XML	Obtiene la estrategia que posee el identificador :id
/:risk/strategies	POST	XML	Agrega la estrategia asociado a un riesgo
/strategies/:id	PUT	XML	Modifica una estrategia con identificador :id

Tabla 15: API REST para las estrategias de mitigación asociadas a un riesgo

URI (Identificador)	Método	Representación	Descripción
/actions	GET	XML	Obtiene todos los planes de contingencia de cliente de la aplicación.
/:risk/actions	GET	XML	Obtiene todos los planes de contingencia asociados al riesgo :risk
/actions/:id	GET	XML	Obtiene el plan de contingencia que posee el identificador :id
/:risk/actions	POST	XML	Agrega un plan de contingencia asociado a un riesgo
/strategies/:id	PUT	XML	Modifica una estrategia con identificador :id

Tabla 16: API REST para los planes de contingencia asociados a un riesgo

¿Cuáles son los códigos de estado de respuesta?

La convención de códigos son las mismas que para la herramienta de administración de requerimientos.

4.2.4. Resultados

Presentamos a continuación los resultados obtenidos de la implementación de la herramienta de Administración de Riesgos mostrando algunas de sus interfaces.

Al comenzar la aplicación se muestran los riesgos de forma jerárquica y en su parte superior aparecen las operaciones que se pueden realizar con ellos. Adicionalmente, se muestra un resumen de los riesgos de acuerdo a una escala de exposición definida en intervalos. Los riesgos de exposición Alta serán aquellos que pertenecen al intervalo (6.4,10.0], Significativa al intervalo (3.6,6.4], Moderada al intervalo (1.6,3.6], Inferior al intervalo (0.4,3.6] y Baja al intervalo [0,0.4].

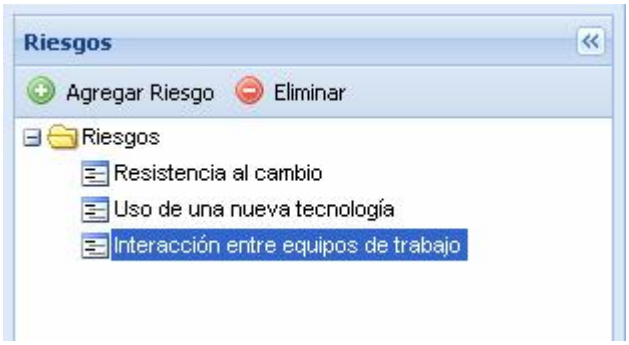


Ilustración 18: Riesgos de un proyecto

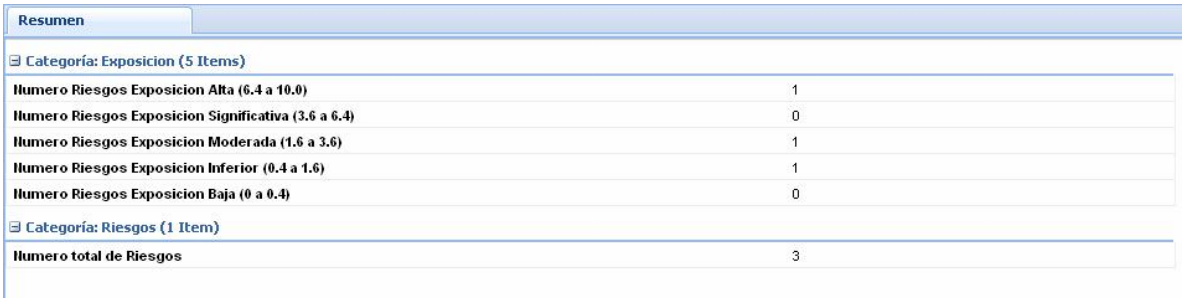


Ilustración 19: Resumen de Riesgos según Exposición

El detalle de un riesgo se muestra al hacer clic sobre uno de ellos, es en esta interfaz donde se puede editar un riesgo, además de agregar, eliminar, editar y visualizar los impactos, estrategias de mitigación, indicadores y medidas de contingencia asociadas al riesgo.

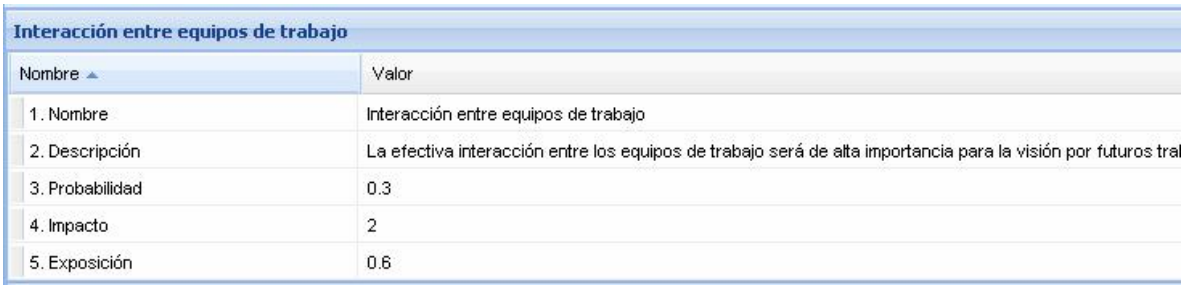


Ilustración 20: Detalle de un Riesgo

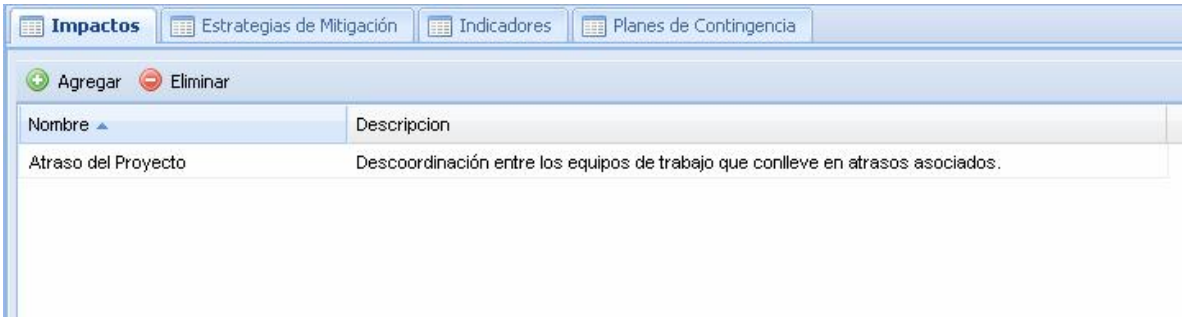


Ilustración 21: Impactos, Indicadores, Estrategias de Mitigación y Planes de Contingencia asociados a un riesgo

4.2.5. Evaluación

Análogamente a la evaluación hecha para la herramienta de administración de requerimientos, revisamos los objetivos que define el CMMI en el área de administración de riesgos evaluando el grado de cumplimiento o apoyo que ofrece la herramienta para finalmente presentar la evaluación hecha por los usuarios.

4.2.5.1. Evaluación con respecto a los objetivos

Objetivo	Justificación	Cumplimiento
OE 1: Preparase para la Administración de Riesgos		
PE 1.1: Determinar las fuentes de riesgo y categorías	En general, las fuentes de riesgos dependerán del contexto particular de cada proyecto y sus categorías pueden depender de este. Las empresas pueden optar por disponer de un estándar o una taxonomía para la clasificación de riesgos. La herramienta simplemente agrupa los riesgos de acuerdo al proyecto al que pertenece y no presenta ninguna otra categoría particular.	MEDIO
PE 1.2: Definir los parámetros de los riesgos	La herramienta define claramente los parámetros que serán usados para analizar y priorizar los riesgos, definiendo su probabilidad de ocurrencia y su impacto bajo una escala definida y además se puede hacer una descripción escrita de los impactos, los indicadores que señalan cuando puede ocurrir un riesgo, las estrategias de mitigación para que el riesgo no ocurra y los planes de contingencia una vez que ocurra el riesgo.	ALTO
PE 1.3: Establecer una estrategia de administración de requerimientos	La herramienta permite apoyar las estrategias tomadas por las organizaciones, puesto que ya trae consigo cuales serán los parámetros para los riesgos y ya define una forma en como se evaluarán, analizarán y priorizarán los riesgos.	ALTO
OE 2: Identificar y Analizar Riesgos		
PE 2.1: Identificar Riesgos	La herramienta es un apoyo a la hora de identificar los riesgos, y además constituye una forma de documentación.	ALTO
PE 2.2: Evaluar, Categorizar y priorizar los riesgos.	Los riesgos dentro de la herramienta presentan su "exposición", que corresponde al producto de la probabilidad del riesgo con su impacto, el cual será el elemento para determinar la importancia del riesgo.	ALTO
OE 3: Mitigar Riesgos		
PE 3.1: Desarrollar los planes de mitigación de riesgos	La herramienta permite definir planes de contingencia asociados a los riesgos.	ALTO
PE 3.2: Implementar los planes de mitigación de riesgos	Según lo anterior la herramienta permite definir planes de contingencia asociados a los riesgos de forma que sienta la base para posibilitar su implementación.	ALTO

Tabla 17: Evaluación con respecto a los objetivos CMMI

4.2.5.2. Evaluación con usuarios

Para la herramienta de administración de riesgos la evaluación por parte de las empresas fue similar en el aspecto de la claridad, simpleza de la interfaz y logro de objetivos de gestión para el área.

Una de las empresas notó que el valor que le aportaba era el cálculo automático del estado de los riesgos de acuerdo a la definición de una escala en base a intervalos, cálculo que ellos deben hacer manualmente.

5. Conclusiones

Tanto los modelos tradicionales de desarrollo de software basados en la mejora de procesos como las metodologías ágiles han ganado terreno en el mundo de desarrollo de software y son dos enfoques que están en constante discusión y evolución. El presente trabajo tiene una línea principal dada por el marco del proyecto Tutelkán, el cual busca generar un modelo de referencia y busca brindar a las empresas la oportunidad de evaluarse CMMI a través de ese modelo. Así es como las herramientas buscan alinearse con los objetivos que este propone en las dos áreas que fueron elegidas: administración de requerimientos y de riesgos.

El CMMI entrega de forma clara los objetivos dentro de las áreas, sin embargo no especifica una forma precisa de cómo conseguir esos objetivos. Por el contrario, las metodologías ágiles y en particular XP, una de las más populares, entregan prácticas concretas que las empresas pueden adoptar y que en general se alinean de buena forma para las áreas de nivel 2 y 3 que posee CMMI, que es donde pertenecen las áreas escogidas para apoyarlas con herramientas de software.

Las áreas de requerimientos y riesgos representan un desafío para toda empresa de desarrollo de software. La administración de requerimientos es vital visualizar el problema que se está resolviendo, y la administración de riesgos ayuda a la vislumbrar cuáles son los posibles problemas que pueden afectar el éxito de un proyecto. Estas dos áreas debiesen complementarse y realizarse de forma conjunta. La administración de requerimientos es una actividad ineludible en el desarrollo de software, en cambio la de administración de riesgos no siempre se lleva a cabo, y es un desafío para las empresas visualizar el valor que tiene llevar ambas actividades de forma conjunta.

La implementación de las herramientas también constituyó un desafío, en particular la de administración de requerimientos, puesto que en el mercado ya existen muchas herramientas que apoyan esta actividad. Se buscó una alternativa simple en su uso, además de poseer características provenientes de las metodologías ágiles. Así es como se manifiesta

la posibilidad de especificar de forma explícita los tests de aceptación por parte del cliente, la estimación por parte de los desarrolladores, cosas que forman parte fundamental en XP para el Planning Game. De igual forma la herramienta de administración de riesgos implementada provee una estrategia concreta de administración de riesgos, concepto importante en XP y que sin embargo no es siempre claro cómo abordarlo.

La arquitectura escogida para la construcción de las herramientas nos permitió tener un enfoque de servicios, lo que permitió concretar el hecho de crear una plataforma de trabajo, además de tener características de extensibilidad e interoperabilidad. La extensibilidad permite hacer una adaptación para las empresas a su forma de trabajar y la interoperabilidad permite tener herramientas, no necesariamente Web, que puedan ser clientes de nuestros servicios, lo que se corroboró en la práctica.

De esta forma las herramientas fueron evaluadas dentro de la empresa creadora del proceso semilla de Tutelkán, donde fueron satisfactoriamente calificadas. La empresa destacó la simplicidad y claridad con que las herramientas logran el objetivo de gestión para ambas áreas comparativamente con las actuales herramientas que poseían. Además visualizaron en las herramientas oportunidades de apoyo a las iniciativas que tienen dentro de la organización, como la mejora de procesos, y la exportación de software gracias a la posibilidad de trabajar colaborativamente.

6. Referencias

[1] Agenda Digital. http://www.agendadigital.cl. 2004. Último acceso, 04/07

[2] Agile Alliance. http://www.agilealliance.org/. Último acceso, 10/07.

[3] Agile Manifesto. http://agilemanifesto.org/. Último acceso, 10/07.

[4] Ahern, Turner, R., and Clouse, A. 2003 CMMI Distilled: a Practical Introduction to Integrated Process Improvement. Addison-Wesley Longman Publishing Co., Inc.

[5] Angecom. http://www.angecom.cl. Último acceso, 06/07.

[6] Barry Boehm, "Get Ready for Agile Methods, with Care," Computer, vol. 35, no. 1, pp. 64-69, Jan., 2002.

[7] Basili, V. R., and A. J. Turner, Iterative Enhancement: A Practical Technique for Software Development, IEEE Trans. Software Engineering, 1,4, 390-396, 1975.

[8] Bastarrica C., Hurtado J. Implementing CMMI using a combination of Agile Methods.

[9] Beck, K. 2000 Extreme Programming Explained: Embrace Change. Addison-Wesley Longman Publishing Co., Inc.

[10] Beck, K. and Andres, C. 2004 Extreme Programming Explained: Embrace Change (2nd Edition). Addison-Wesley Professional.

[11] Boehm, B. W. 1988. A spiral model of software development and enhancement. IEEE Computer, 21, 5, 61–72.

[12] Chrissis, M. B., Konrad, M., and Shrum, S. 2003 CMMI Guidelines for Process Integration and Product Improvement. Addison-Wesley Longman Publishing Co., Inc.

[13] CMM. Capability Maturity Model. http://www.sei.cmu.edu/cmm/. Último acceso, 10/07

[14] CMMI for Development, Version 1.2. CMU/SEI-2006-TR-008 ESC-TR-2006-008. SEI. Carnegie Mellon University.

[15] CMMI. Capability Maturity Model Integration. http://www.sei.cmu.edu/cmmi/. Último acceso, 04/07

[16] Cockburn, A. 2004. Crystal Clear: A Human-powered Methodology for Small Teams. Addison-Wesley.

[17] Cohn, M. 2004 User Stories Applied: for Agile Software Development. Addison Wesley Longman Publishing Co., Inc.

[18] Dynamic Software Development Method. http:// www.dsdm.org/. Último acceso 10/07.

[19] Fielding, Roy Thomas. Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation, University of California, Irvine, 2000.

[20] Fiksel, J. Dunkle, M. "Principles of requirement management automation", 1992.

[21] Fritzsche, M., Keil, P. Agile Methods and CMMI: Compatibility or Conflict? Technische Universität München. 2007.

[22] GECHS. Asociación Gremial de las empresas chilenas desarrolladoras de software. http://www.gechs.cl/. Último acceso 11/07.

[23] GECHS. Quinto diagnostico Industria Nacional de Software y Servicios. 2007.

[24] Herbsleb, J., Zubrow, D., Goldenson, D., Hayes, W., and Paulk, M. 1997. Software quality and the Capability Maturity Model. Commun. ACM 40, 6 (Jun. 1997), 30-40.

[25] Highsmith, J. A. 2000 Adaptive Software Development: a Collaborative Approach to Managing Complex Systems. Dorset House Publishing Co., Inc.

[26] Intermedia. http://www.intermedia.cl. Último acceso, 06/07.

[27] ISO 9000. International Organization for Standardization. http://www.iso.org/. Último acceso, 04/07

[28] ISO 9001:2000. Quality Management Systems - Requirements. 2000.

[29] ISO/IEC 12207 Amendment 1: Information Technology - Software Life Cycle Processes Amendment 1.

[30] ISO/IEC 15504-2:1998. Information technology - Software process assessment - Part 2: A reference model for processes and process capability.

[31] ITDA-KP. Information Technology Development Area – KEPLER Process. http://www.keplertech.com/Site/productos_servicios/productos_servicios_itdakp.html. Último acceso, 04/07

[32] J. Kontio. The Riskit Method for Sofiware Risk Management, version I.00 CS-mT3782 I MEACSTR-97-38, 1997. Computer Science Technical Reports. University of Maryland. College Park, MD.

[33] Jeffries, R. Essential XP: Card, Conversation, Confirmation. http://www.xprogramming.com/xpmag/expCardConversationConfirmation.htm. Último acceso 10/07.

[34] Jeffries, R. Risk Management. http://www.xprogramming.com/xpmag/risklog.htm. Último acceso 10/07.

[35] Kepler Technologies. http://www.keplertech.com/. Último acceso 11/07.

[36] Martin, J. 1999 Rapid Application Development. Maxwell Macmillan International.

[37] Myerson, M. 1996 Risk Management Processes for Software Engineering Models. 1st. Artech House, Inc.

[38] Nawrocki, J. R., Jasiñski, M., Walter, B., and Wojciechowski, A. 2002. Extreme Programming Modified: Embrace Requirements Engineering Practices. In Proceedings of the 10th Anniversary IEEE Joint international Conference on Requirements Engineering (September 09 - 13, 2002). RE. IEEE Computer Society, Washington, DC, 303-310.

[39] Nectia. http://www.nectia.cl. Última acceso, 06/07.

[40] Nerur, S. and Balijepally, V. 2007. Theoretical reflections on agile development methodologies. Commun. ACM 50, 3 (Mar. 2007), 79-83.

[41] Nerur, S., Mahapatra, R., and Mangalaraj, G. 2005. Challenges of migrating to agile methodologies. Commun. ACM 48, 5 (May. 2005), 72-78.

[42] Newkirk, J. 2002. Introduction to agile processes and extreme programming. In Proceedings of the 24th international Conference on Software Engineering (Orlando, Florida, May 19 - 25, 2002). ICSE '02. ACM Press, New York, NY, 695-696.

[43] Nielsen J. Designing Web Usability: The Practice of Simplicity.

[44] Orhan KALAYCI. CMMI versus XP

[45] Paetsch, F., Eberlein, A. and Maurer, F. (2003) "Requirements Engineering and Agile Software Development" in Proceedings of the International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, Linz, Austria, 308--313. http://citeseer.ist.psu.edu/article/paetsch03requirements.html

[46] Palmer, S. R. and Felsing, M. 2001 A Practical Guide to Feature-Driven Development. 1st. Pearson Education.

[47] Paulk, M.C. Extreme Programming from a CMM Perspective. XP Universe Conference Papers, Raleigh, NC, 23-25 July 2001.

[48] Poppendieck, M. and Poppendieck, T. 2003 Lean Software Development: an Agile Toolkit. Addison-Wesley Longman Publishing Co., Inc.

[49] Proyecto Tutelkán. “Obtención de altos estándares de calidad en la industria de software nacional, utilizando procesos de referencia”. http://www.tultekan.org. 2006. Último acceso, 04/07

[50] Reed, K., Damiani, E., Gianini, G., and Colombo, A. 2004. Agile management of uncertain requirements via generalizations: a case study. In Proceedings of the 2004 Workshop on Quantitative Techniques For Software Agile Process (Newport Beach, California, November 05 - 05, 2004). QUTE-SWAP '04. ACM, New York, NY, 40-45.

[51] Royce, W. W. 1970. Managing the development of large software systems: Concepts and techniques. In Proceedings of IEEE WESTCON. (Los Angeles, CA), 1–9.

[52] Saiedian, H. and Carr, N. 1997. Characterizing a software process maturity model for small organizations. SIGICE Bull. 23, 1 (Jul. 1997), 2-11.

[53] Schwaber, K. 2004 Agile Project Management with Scrum. Microsoft Press.

[54] Smith, S. and Stoecklin, S. 2001. What we can learn from extreme programming. J. Comput. Small Coll. 17, 2 (Dec. 2001), 144-151.

[55] Softram. http://www.softram.cl. Último acceso, 06/07.

[56] Sommerville, I. 1996. Software process models. ACM Comput. Surv. 28, 1 (Mar. 1996), 269-271.

[57] Sommerville, I. 2004 Software Engineering (7th Edition). Pearson Addison Wesley.

[58] Staples, M., Niazi, M., Jeffery, R., Abrahams, A., Byatt, P., and Murphy, R. 2007. An exploratory study of why organizations do not adopt CMMI. J. Syst. Softw. 80, 6 (Jun. 2007), 883-895

[59] Steve Andriole, "The Politics of Requirements Management," IEEE Software, vol. 15, no. 6, pp. 82-84, Nov/Dec, 1998.

[60] Sutton, W. L. 1988. Advanced models of the software process. In Proceedings of the 4th international Software Process Workshop on Representing and Enacting the Software Process (Devon, United Kingdom). C. Tully, Ed. ACM Press, New York, NY, 156-158.

[61] Theunissen, W. H., Kourie, D. G., and Watson, B. W. 2003. Standards and agile software development. In Proceedings of the 2003 Annual Research Conference of the South African institute of Computer Scientists and information Technologists on Enablement Through Technology (September 17 - 19, 2003). J. Eloff, A. Engelbrecht,

P. Kotzé, and M. Eloff, Eds. ACM International Conference Proceeding Series, vol. 47. South African Institute for Computer Scientists and Information Technologists, 178-188.

[62] Villena, Agustín. Apuntes del curso CC62V: Taller de Metodologia Agiles de Desarrollo de Software.

[63] Wikipedia. Hypertext Transfer Protocol. http://es.wikipedia.org/wiki/Hypertext_Transfer_Protocol. Último acceso, 10/07.

[64] Wikipedia. Service Oriented Architecture Protocol. http://en.wikipedia.org/wiki/SOAP. Último acceso, 10/07.

[65] Wikipedia. Software Development. http://en.wikipedia.org/wiki/Software_development. Ultimo acceso, 06/07.

[66] Wikipedia. Word Wide Web. http://en.wikipedia.org/wiki/World_Wide_Web. Último acceso, 10/07.

[67] Williams, L. G. 1988. A behavioral approach to software process modelling. In Proceedings of the 4th international Software Process Workshop on Representing and Enacting the Software Process (Devon, United Kingdom). C. Tully, Ed. ACM Press, New York, NY, 167-170.

7. Anexos

7.1.1. CMM

Capability Maturity Model (CMM), también conocido como Capability Maturity Model for Software (SW-CMM) es un modelo de referencia para evaluar la madurez de un proceso de software desarrollado por el Software Engineering Institute (SEI), y un modelo normativo para ayudar a organizaciones de software a progresar sobre un camino evolucionario, desde un proceso ad-hoc, caótico, hacia uno maduro y disciplinado.

El CMM fue originalmente desarrollado para asistir al Departamento de Defensa de los Estados Unidos (DoD) para la adquisición de software [24]. A través de los esfuerzos del SEI para obtener una amplia participación en el desarrollo y el mejoramiento del CMM, el modelo ganó visibilidad en una extensa comunidad de la ingeniería de software. Gradualmente, organizaciones comerciales comenzaron a adoptar el CMM como un framework para sus iniciativas de mejoramiento internas.

El CMM fue definido en cinco niveles de madurez. Actualmente el CMM ha sido retirado por el SEI y lo ha reemplazado el CMMI [13].

7.1.2. CMMI

"The quality of a system or product is highly influenced by the quality of the process used to develop and maintain it"
CMMI. Versión 1.2.

A partir de CMM se comenzaron a desarrollar otros modelos para otras disciplinas, de esta forma nacieron los modelos P-CMM (People CMM), para mejorar las capacidades de las personas, SA-CMM (Software Adquisition CMM), para mejorar el proceso de adquisición de software, SE-CMM (Systems Engineering CMM), para mejora de los procesos de sistemas de ingeniería, IPD-CMM (Integrated Product Development CMM), que mejora procesos organizacionales y de administración de proyectos, con el enfoque de involucrar a todas las disciplinas que participen en el desarrollo de un producto o servicio y otros.

Lo anterior era fuente de confusión cuando se utilizaba más de un modelo, era difícil de integrar y combinar en un programa de mejoramiento, debido a sus distintas estructuras, formatos, términos y formas de medir la madurez. Es así como aparece el Capability Maturity Model Integration (CMMI) en el 2005, que integra disciplinas de software y sistemas en un solo framework de mejora de procesos, y provee formas de ingresar nuevas disciplinas cuando las necesidades llegan.

El CMMI es un modelo de madurez de mejoramiento de procesos para el desarrollo de productos y servicios. Consiste en las mejores prácticas dirigidas a las actividades de desarrollo y mantención, que cubren el ciclo de vida del producto desde su concepción hasta su entrega [14].

Dentro del CMMI, el SEI ha encontrado varias dimensiones en las cuales una organización puede enfocarse para mejorar su negocio. Las tres críticas son: la gente, procedimientos y métodos, y las herramientas y equipamiento (**Error! Reference source not found.**). Estas tres dimensiones se mantienen unidas a través de los procesos que se dan dentro la organización. De esta forma, el SEI ha tomado la siguiente premisa de

administración de procesos: "*la calidad de un sistema o producto está altamente influida por la calidad de los procesos usados para desarrollarla y mantenerla*".

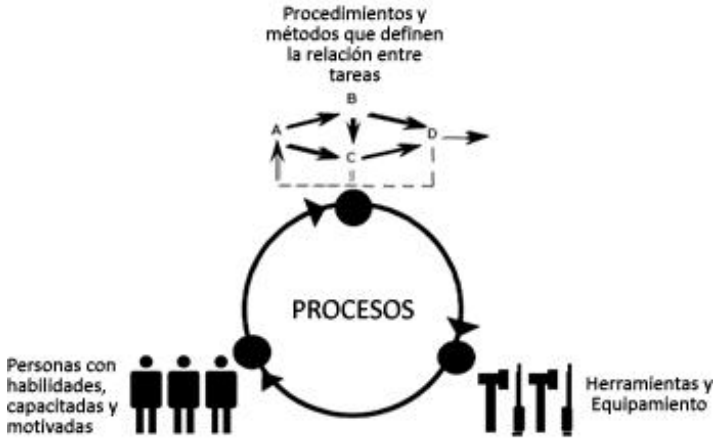


Ilustración 22: Las tres dimensiones críticas

El CMMI define áreas de procesos, (process areas, PA) para representar un agrupamiento de prácticas relacionadas en un área que, cuando es implementada en conjunto, satisface un grupo de objetivos considerados importantes para realizar un mejoramiento en dicha área.

El CMMI permite el mejoramiento de procesos usando dos representaciones, llamadas continua y por etapas (continuous and staged). La representación continua usa niveles de capacidad, para caracterizar el mejoramiento relativo para un área de proceso individual (o un grupo de áreas de proceso). A su vez, la representación por etapas usa un conjunto predefinido de áreas de proceso, que define un camino de mejoramiento para la organización. Todas las áreas de proceso del CMMI son comunes para ambas representaciones y en su versión 1.2 existen 22 áreas de proceso.

7.1.2.1. Niveles CMMI

Los niveles dentro del modelo son usados para describir un camino evolutivo recomendado para una organización que quiere mejorar los procesos que usa, de forma de desarrollar y mantener sus productos y servicios. Existen dos caminos, que se asocian a las dos representaciones que tiene CMMI. Para la representación continua se le denominan "Niveles de Capacidad" y para la escalonada "Niveles de Madurez". Para alcanzar un nivel, la organización debe alcanzar todos los objetivos de un área de proceso o de un conjunto de área de procesos, independientemente de si es un nivel de capacidad o madurez. La **Error! Reference source not found.** entrega una breve descripción de los niveles de madurez definidos por el CMMI.

Nivel de Madurez	Características
1.- Inicial	El proceso de software es caracterizado como ad-hoc, e incluso caótico. Pocos procesos están definidos y éxito depende del esfuerzo individual.
2.- Repetible	Se establecen procesos de administración de proyectos básicos para realizar su seguimiento de costo, planificación y funcionalidad. La disciplina de proceso necesaria está en el lugar para repetir de forma temprana el éxito en proyectos de aplicaciones similares.
3.- Definido	El proceso de software para actividades de administración e ingeniería está documentado, estandarizado e integrado en un proceso estándar para la organización. Los proyectos usan una aprobada, adaptada versión del proceso estándar de la organización para el desarrollo y mantención del software.
4.- Administrado	Se recolectan métricas detalladas para el proceso de software y la calidad del producto. Ambos, el proceso de software y los productos son cuantitativamente entendidos y controlados.
5.- Optimizado	Proceso continuo de mejora es facilitado por retroalimentación cuantitativa del proceso, ideas innovadoras y nuevas tecnologías.

Tabla 18: Niveles de Madurez del CMMI

7.1.2.2. Áreas de Proceso

Presentamos una breve descripción de las áreas de cada nivel del CMMI en Tabla 19.

Nivel	Acrónimo	Área de Proceso	Descripción
2	REQM	Administración de Requerimientos	Administración de los requerimientos del proyecto, productos y sus componentes para identificar las inconsistencias entre los requerimientos, planes del proyecto y productos de trabajo.
	PP	Planificación del Proyecto	Establece y mantiene los planes que definen las actividades del proyecto
	PMC	Monitoreo y Control del Proyecto	Provee de una visualización del avance del proyecto para tomar acciones correctivas si existen desviaciones significativas con respecto al plan.
	SAM	Administración de Acuerdos con Proveedores	Administra la adquisición de productos por parte de proveedores con los cuales existe un acuerdo formal.
	MA	Métricas y Análisis	Desarrolla y sostiene la capacidad de medir para apoyar las necesidades de información de gestión.
	PPQA	Aseguramiento de calidad de procesos y productos	Provee de personal para obtener entendimiento de los procesos y sus productos asociados de forma de poder evaluar el cumplimiento de ellos.
	CM	Administración de la Configuración	Establece y mantiene la integridad los productos de trabajo.
3	RD	Levantamiento de Requerimientos	Produce y analiza los requerimientos de cliente
	TS	Solución Técnica	Diseño, desarrollo e implementación de la solución de los requerimientos.
	PI	Integración de Productos	Ensambla los productos y sus componentes. Asegura que los productos funcionan correctamente integradamente para su entrega.
	VER	Verificación	Se verifica que el producto satisface los requerimientos
	VAL	Validación	Se demuestra que el producto cumple con su propósito cuando es puesto en su ambiente.
	OPF	Enfoque en Procesos Organizacionales	Planificación e implementación de mejora de procesos organizacional basado en las fortalezas y debilidades de los procesos de la organización
	OPD	Definición del Procesos Organizacionales	Establece y mantiene un conjunto de procesos a nivel organizacional
	OT	Capacitación Organizacional	Desarrollo de las habilidades y conocimiento del personal para que puedan ejercer sus roles de forma efectiva y eficiente.
	IPM	Administración de Proyectos Integrada	Establecer y administrar la involucración de los interesados del proyecto de acuerdo a un proceso definido integral que se adapta a los estándares de

			procesos organizacionales
	RM	Administración de Riesgos	Identifica los potenciales problemas antes de que ocurran, de forma de planificar y ejecutar actividades para mitigar los impactos adversos en el logro de los objetivos
	DAR	Análisis y Toma de Decisiones	Analizar las posibles decisiones usando una evaluación formal, identificando alternativas y evaluándolas contra los criterios establecidos
4	OPP	Desempeño de Procesos Organizacionales	Establecer y mantener un entendimiento cuantitativo del desempeño de los procesos estándar de la organización de acuerdo a los objetivos de calidad y desempeño, para proveer datos de forma de administrar cuantitativamente los proyectos.
	QPM	Administración de Proyecto Cuantitativa	Cuantitativamente administrar los procesos definidos por el proyecto para alcanzar los objetivos de calidad y desempeño.
5	OID	Innovación Organizacional y Despliegue	Seleccionar y desplegar mejoras incrementales e innovadoras que mejoran los procesos y tecnologías de la organización de forma cuantitativa de acuerdo a los objetivos de negocio organizacionales.
	CAR	Análisis y Resolución Causal	Identifica las causas o defectos y otros problemas para tomar medidas para prevenir una ocurrencia futura.

Tabla 19: Áreas de Proceso del CMMI

7.1.2.3. Componentes del Modelo

Los componentes del modelo describen cómo será descrita un área de proceso y se dividen en tres categorías: requeridos, esperados e informativos, que indican como interpretarlas.

- **Componentes Requeridos:** Los componentes requeridos describen lo que una organización debe lograr para satisfacer un área de proceso. El logro debe ser visiblemente implementado en los procesos de la organización. Los componentes requeridos en CMMI corresponden a los objetivos generales y específicos.

- **Componentes Esperados:** Los componentes esperados describen que una organización puede implementarlos para lograr los componentes requeridos. Los componentes esperados incluyen las prácticas genéricas y específicas.

- **Componentes Informativos:** Los componentes informativos proveen detalles que ayudan a la organización a pensar en una estrategia de cómo realizar los componentes requeridos y esperados. Subprácticas, productos de trabajo típico, amplificaciones, referencias son ejemplos de los componentes informativos del modelo.

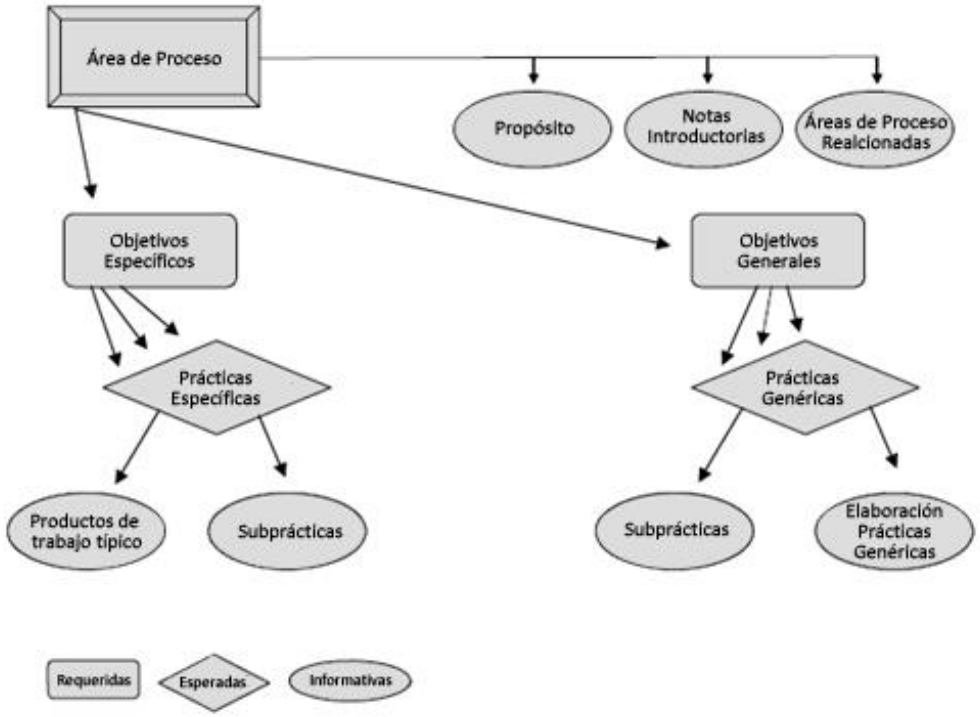


Ilustración 23: Esquema de los componentes del CMMI.

7.2. Metodologías Ágiles

"Software development fails to deliver, and fails to deliver value. This failure has huge economic and human impact. We need to find a new way to develop software"
-- Kent Beck.

Las metodologías ágiles aparecen como una nueva propuesta de cómo realizar el desarrollo de software. Toda metodología ágil tiene como factor común un conjunto de valores y principios que la Agile Alliance [2] expresó en el Manifesto Ágil [3], el cual señala donde está el mayor valor en el desarrollo de software:

- Individuos y sus interacciones por sobre procesos y herramientas
- Software funcional sobre documentación exhaustiva
- Colaboración con el cliente por sobre negociación de contratos
- Responder al cambio por sobre seguir un plan

Actualmente existen una serie de metodologías ágiles, dentro de las cuales destacan:

7.2.1. Scrum

Scrum [53], lleva su nombre de la formación que se realiza en el Rugby. Fue inicialmente desarrollada por Ken Schwaber y Jeff Sutherland, con colaboraciones posteriores de Mike Beedle. Scrum provee un framework de administración de proyectos que enfoca el desarrollo en ciclos Sprint de 30 días de duración, en los cuales una serie de características, registradas en el Backlog, son entregadas. La principal práctica en Scrum es el uso de reuniones de equipo que duran 15 minutos para la coordinación y la integración. Scrum ha estado en uso por casi diez años y ha sido usado para la entrega exitosa de una amplia gama de productos.

7.2.2. Dynamic Systems Development Method (DSDM)

El Dynamic Systems Development Method [17] fue desarrollado en Gran Bretaña, a mediados de los 90. Es fruto, y una extensión, de las prácticas de Rapid Application Development (RAP) [36]. DSDM ostenta el mayor soporte para su capacitación y documentación, al menos en Europa. DSDM posee nueve principios que incluyen: involucración activa con el usuario, entregas frecuentes, decisiones tomadas por el equipo, testing integrado a través del ciclo de vida del proyecto, y cambios reversibles en el desarrollo.

7.2.3. Crystal Methods

Alistair Cockburn es el autor de la familia de métodos centrado en las personas, "Crystal" [16]. Alistair es un "arqueólogo de metodologías", que ha entrevistado a docenas de equipos a través del mundo, tratando de separar que es lo que realimente funciona, de lo que la gente dice que debería funcionar. Alistair, y Crystal, se centran en los aspectos personales en el desarrollo – colaboración, cooperación. Alistair usa el tamaño del proyecto, criticidad, y objetivos para configurar de la mejor forma las prácticas para cada miembro de las metodologías de la familia Crystal.

7.2.4. Feature Driven Development

Jeff De Luca y Peter Coad colaboraron en el Feature-Driven Development [46]. FDD consiste en un proceso minimalista de cinco pasos, que se centra en desarrollar un modelo de objetos global, construyendo una lista de características (features), para luego planificar por característica, y seguir un diseño por característica iterativo, y finalmente construir por pasos cada característica. Los procesos del FDD son breves (cada uno descrito en una página), y existen dos roles clave en FDD, que son el "chief architect" y el "chief programmer".

7.2.5. Lean Development

Desarrollada por Bob Charette, Lean Development [48] fue derivada de los principios que introdujo la reestructuración de la industria automovilística Japonesa en los 80's. En LSD, Charette extiende la visión de las metodologías tradicionales sobre el cambio, como un riesgo de pérdida a ser controlado con la administración de prácticas restrictivas, hacia un cambio visto como oportunidad.

7.2.6. Adaptive Software Development

Adaptive Software Development [25], desarrollado por Jim Highsmith, fue una contribución al movimiento ágil, que provee un fondo filosófico para los métodos ágiles, mostrando como el desarrollo de software en las organizaciones puede responder a la turbulencia del actual clima de negocio, aprovechando más que evitando el cambio. ASD contiene ambas prácticas- desarrollo iterativo, planificación basada en característica, revisiones con el cliente a través de focus group- y una "ágil" filosofía de gestión basada en el liderazgo y la colaboración.

7.2.7. Extreme Programming

Extreme Programming (XP) [9], fue desarrollada por Kent Beck, Ward Cunningham y Ron Jeffries. XP predica los valores de la comunicación, simplicidad, retroalimentación y coraje. Los aspectos importantes de XP son alterar el punto de vista del costo del cambio para un desarrollo, y énfasis en el perfeccionamiento técnico a través de la refactorización y el Test Driven Development. XP provee un sistema dinámico de prácticas, cuya integración como unidad integral ha sido probada. XP es una de las más populares metodologías ágiles y es por esto que profundizaremos en sus características.

En su primera propuesta Kent Beck presentó a XP como: una liviana, eficiente, de bajo riesgo, flexible, predecible, científica, y divertida forma de desarrollar software [9].

XP considera el desarrollo de software como un sistema de control de cuatro variables: Costo, Tiempo, Alcance, Calidad. La primera indica la cantidad de recursos invertidos, la segunda determina cuando el sistema estará listo, la tercera cuanto será hecho y la última la correctitud del sistema. Además XP define cuatro valores Comunicación, Simplicidad, Retroalimentación, Coraje. La comunicación entendida como un factor fundamental en el éxito del proyecto, la que permite la transmisión de ideas, funcionalidades y diseño entre el cliente y el equipo de desarrollo. Simplicidad, entendida como la maximización de trabajo que no se hace. Retroalimentación, de forma de descubrir de forma temprana las necesidades y los problemas en el desarrollo, y Coraje como una actitud agresiva hacia la construcción del software, para hacer extremas las prácticas.

De los valores se derivan una serie de principios. Se juega a ganar, enseñando a aprender, donde debe darse siempre una comunicación abierta y honesta, de forma de obtener una retroalimentación rápida, que favorezca el aprendizaje. Se enseña a aprender, asumiendo siempre simplicidad y realizando cambios paso a paso para atacar los problemas urgentes, dejando la mayor cantidad de opciones abiertas, realizando experimentos concretos de forma de hacer siempre un trabajo de calidad. Se trabaja con los instintos de las personas, no contra ellos, los cuales aceptan su responsabilidad (no se les asigna) y miden honestamente, para que en definitiva el equipo siempre viaje con poco equipaje, simple y valioso.

XP se puede ver como un proceso definido dividido en tres ciclos, el cual se puede apreciar en la Ilustración 24: Ciclos de XP. El de más afuera representa la interacción que tiene el equipo con el cliente, que corresponde a la gestión de valor. Las prácticas corresponden al Planning Game (Juego de Planeación), que cuenta con la activa participación del cliente para escribir y priorizar las historias de usuario, funcionalidades del software, donde además se cuenta con el equipo completo para estimar dichas historias de forma de definir el alcance de la siguiente entrega, las cuales deben ser pequeñas y frecuentes. Así se obtiene feedback del cliente lo más tempranamente posible. Adicionalmente el cliente debe especificar las condiciones de satisfacción para que el equipo sepa cuando una funcionalidad está completa.

En el ciclo del medio se muestran las prácticas para la gestión del equipo de desarrollo, el cual debe alcanzar un ritmo sostenido de trabajo, donde no se permiten las horas extra de trabajo, para que mantenga su motivación. Además se debe dar una propiedad colectiva del código generado, basado en estándares de codificación, de forma que todo miembro del equipo pueda aportar valor en el desarrollo y no dependa de personas en particular. Al final, cuando se termine una funcionalidad, el equipo debe hacer la integración con el resto del código generado, la cual debe hacerse de forma continua para detectar errores o problemas tempranamente.

Finalmente, el ciclo interno determina como debe realizarse el desarrollo dentro del equipo apuntando siempre a la obtención de un código de calidad. Se tiene la práctica de programación de a pares, dos miembros del equipo se sientan juntos a programar, lo que disminuye los errores en la programación, desarrollo que debe ser guiado por tests, donde se diseña e implementa la prueba que debe satisfacer la funcionalidad, para posteriormente diseñar e implementar el código que pase las pruebas. El diseño debe ser lo más simple posible, se debe diseñar de forma de obtener la cosa más simple que pueda funcionar, además de siempre refactorizar el código una vez que sea necesario, esto es, reestructurar el código de forma de no alternar su funcionamiento externo, para mejorar su diseño e estructura interna.

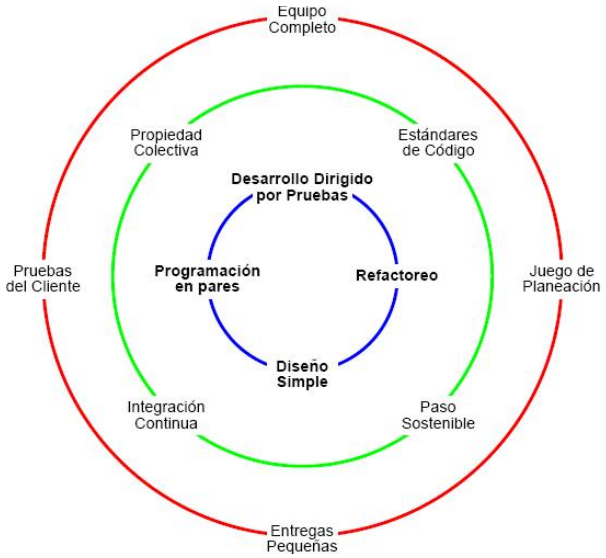


Ilustración 24: Ciclos de XP

7.3. REST

La Internet ha generado nuevas formas de comunicarse, congregarse, y compartir información, no cabe duda de la revolución que ha provocado y el impacto que ha tenido en diversas áreas, se ha convertido en una red de redes que ha sobrepasado su fin inicial de compartir información científica.

Hoy en día, la Web no sólo entrega recursos sino también servicios, esto ha obligado a diseñar pensando en la Web como orientada a servicios. El protocolo SOAP (Service Oriented Arquitecture Protocol [64]) es un ejemplo de esto, SOAP utiliza mensajes XML para la comunicación de las peticiones a los servidores usando HTTP y así proveer de más herramientas a la Web para que esta pudiese proveer servicios tal como son desarrollados para las estaciones de trabajo local y ser usados desde cualquier aplicación remota.

Roy Fielding, principal desarrollador de la comunidad Apache y del protocolo HTTP, en su investigación para su PhD, propone una arquitectura llamada por él REST [18] (Representational State Transfer, en español, Transferencia de Estado Representacional) basada en la simplicidad y en el uso de la Web tal como está construida, a diferencia de SOAP que provee de mecanismos sobre lo ya construido, perdiendo simplicidad en la hora de exponer y consumir servicios.

REST describe un sistema de red en base a:

- Elementos de Datos: Recursos, Identificadores, Representaciones
- Conectores: Cliente, Servidor, Cache, Resolver, Tunnel
- Componentes: Servidor de origen, Gateway, Proxy, User Agents

Donde sus características principales son:

- Los elementos de datos son accesados a través una interfaz estándar
- Los componentes se comunican a través de la transferencia de representaciones de los recursos a través de esta interfaz en vez de operar directamente con el recurso.
- Los conectores presentan una interfaz abstracta de comunicación, escondiendo los detalles de implementación y mecanismos de comunicación.
- Los componentes usan conectores para acceder, proveer acceso o mediar el acceso a los recursos
- Todas las peticiones hechas a través de conectores deben contener toda la información necesaria para el entendimiento de dicha petición, sin depender de alguna petición anterior. Esto es lo que le da el carácter de "Sin Estado"

El estilo de arquitectura REST posee varios atributos que lo hacen interesante a la hora de optar por él, podemos mencionar el hecho de que posee escalabilidad al aumentar el número de clientes, permite la transmisión de datos bajo distintos tipos, dado que se puede agregar soporte para nuevos tipos de contenidos sin reducir o eliminar los tipos de contenidos antiguos, soporta intermediarios (proxies y gateways) y gracias que soporta caching posee mejores tiempos de respuesta y mejores características de carga del servidor. Sin embargo, dentro de las razones más poderosas que nos insta a optar por esta arquitectura, es que hace posible la interoperabilidad de nuestras herramientas, esto es gracias a que a través de la definición de una interfaz de acceso común nos permite tener varias aplicaciones cliente que consuman esas interfaces. Así en la práctica podemos tener no tan sólo una aplicación Web como aplicación cliente sino que también otras aplicaciones que consuman los servicios ofrecidos por esa interfaz. Por otra parte tenemos la extensibilidad como características de las herramientas, ya que podemos agregar nuevas interfaces que provean de nuevos servicios, esto en la práctica nos permitirá tener la posibilidad de extender las capacidades de las herramientas adaptándose a la forma de trabajo que posean las empresas.

La forma de desarrollo bajo esta arquitectura, de forma simple, debe responder las siguientes preguntas:

- ¿Cuáles son los recursos? Los recursos representan la abstracción de la información. Cualquier información que pueda ser nombrada puede corresponder un recurso: un documento, una imagen, un servicio, una colección de recursos, etc. Un recurso es un mapping conceptual a un conjunto de entidades.

- ¿Cuáles son las representaciones? Las representaciones corresponden a los datos y los metadatos, datos sobre los datos.

- ¿Cuáles son los métodos soportados para cada recurso? Los métodos representan las operaciones que se pueden realizar sobre los recursos. Para HTTP los métodos más conocidos corresponden al GET, para la obtención de recursos, POST, para agregar recursos, PUT, para modificar recursos y DELETE para eliminar recursos.

- ¿Cuáles son los códigos de estado de respuesta como retorno? Estos representan las posibles respuestas que puede enviar el servidor a partir de una petición y que HTTP ya tiene bien definidos.

7.4. Framework de Desarrollo

Se ha escogido la alternativa de ocupar un framework de desarrollo para la implementación de las herramientas debido a que permite heredar las buenas prácticas con que fue construido, lo que permite construir un mejor código para la aplicación y debido a que en general disminuye el tiempo de programación para el desarrollador.

Se analizaron distintas alternativas de framework para el desarrollo web en distintos lenguajes: python, php, ruby y java. El proceso de evaluación consistió en:

- Elaborar una lista de criterios de evaluación.
- Elegir los frameworks de desarrollo Web para cada uno de los lenguajes de programación.
- Realizar su instalación y configuración.
- Realizar uno o más tutoriales provistos en la documentación del framework.
- Analizar la experiencia a través de la lista de criterios de evaluación, para escoger la alternativa de desarrollo.

Se consideraron los siguientes criterios de evaluación para los frameworks:

- Instalación: analizar cuán sencillo y simple es realizar la instalación del framework.
- Configuración: analizar cuán fácil es configurar el framework, su puesta en producción.
- Documentación: analizar cual es el estado de la documentación del framework, para abordar el desarrollo sobre una buena base y disminuir la curva de aprendizaje.
- Características propias: analizar cuales son las características que ofrece el framework y que posibiliten un desarrollo acorde a las necesidades de las aplicaciones que se construirán. En particular se favoreció aquellos frameworks que faciliten el trabajo bajo la arquitectura definida.

Finalmente, se estableció una escala sencilla: ALTA, MEDIA, BAJA, para medir el grado de satisfacción de los criterios establecidos. El resultado se presenta en la Tabla 20.

Lenguaje	Python	Php		Ruby	Java
Criterios\Framework	Pylons	Symfony	CakePhp	Ruby on Rails	Grails
Instalación	ALTA	MEDIA	ALTA	ALTA	MEDIA
Configuración	ALTA	MEDIA	ALTA	ALTA	MEDIA
Documentación	ALTA	ALTA	ALTA	ALTA	MEDIA
Características	ALTA	ALTA	MEDIA	MEDIA	MEDIA

Tabla 20: Cuadro comparativo de frameworks de desarrollo Web.

En general, las alternativas de frameworks de desarrollo en el lenguaje PHP poseen favorables condiciones para su instalación y configuración debido a que es una plataforma que penetrado en el mercado y que es fácil de encontrar básicamente en cualquier servicio de hosting. Las alternativas en lenguajes Python o Ruby proveen de servicios de instalación y configuración sencillos y simples en comparación a las alternativas PHP. A su vez el framework en lenguaje Java se consideró inmaduro, lo cual sería un riesgo abordar el proceso de desarrollo con esa plataforma.

La alternativa escogida será entonces el framework de desarrollo en lenguaje Python, Pylons, por cumplir los criterios de evaluación de forma satisfactoria destacando su facilidad en su instalación y configuración y proveer de características que se adaptan a la arquitectura definida.