La Gestión de Software aplicada a la Mejora Continua.

Proyecto de Grado

Ingeniería en Computación.

Informe Final:

Departamento de Investigación Operativa,

Instituto de Computación,

Facultad de Ingeniería, UdelaR.

Alumna:

A/C Silvana Deana.

Tutores:

Ing. MSc. Omar Viera, Departamento Investigación Operativa.

A/P Gabriel Ledesma, Jefe del Departamento de Desarrollo de Abitab S.A..

19 de mayo de 2005

Facultad de Ingeniería,
Universidad de la República,
Montevideo,
Uruguay,
2005.

Resumen

Existen varias teorías sobre el desarrollo y la gestión de software. Estas son rechazadas muchas veces por la falta de credibilidad, y/o por su costo en tiempo y dinero. Cuántas veces hemos escuchado "esto siempre lo hemos hecho así y no nos ha ido mal", sin realizar un estudio adecuado de las posibilidades de mejorar, o aún peor, "¿Qué tanto tiempo se puede sobrevivir trabajando del mismo modo?". En realidad, el factor económico tiende a ser determinante en este tipo de decisiones y un cambio en la metodología de trabajo es un riesgo, cuyo costo, muchas veces, es sumamente difícil de financiar. Cuando se quiere realizar una mejora, se deben conocer las facetas que inciden en forma favorable y desfavorable. Para esto se debe registrar lo que se hace, medir los resultados, estudiarlos y realizar un plan de mejora. Los cambios bruscos y exagerados en la metodología de trabajo de una empresa, son naturalmente rechazados, por lo que se deben realizar paulatinamente, tomando medidas para poder estudiar su impacto, constatando el nivel de éxito o fracaso.

La meta es mejorar, bajo las condiciones de mantener el rendimiento en niveles aceptables y sin costo para la empresa. Conocer las aptitudes de sus técnicos de forma objetiva para asignarlos a tareas donde sean más productivos y permita incentivarlos, es parte del camino para lograrlo. Mejorar implica medir, medir implica tomar datos y tomar datos implica organizarse. El objetivo de este proyecto de grado es establecer un proceso de mejora continua, lo cual implica conocer y tener control sobre la construcción del software, para esto se debe medir la productividad. Es decir, tener medidas del tiempo del trabajo y el costo de personal, de los recursos de hardware y software, las restricciones del entorno, el grado de dificultad de los problemas, y el tamaño, las funcionalidades, la confiabilidad y cantidad de defectos de los productos que se construyen. El primer paso de este proceso es tener una herramienta que colecte y analice datos. Se realizó un estudio de herramientas de gestión de software, principalmente de Fuente Abierta[1] dada la restricción de costos. Se llegó a la conclusión de que ninguna de estas se adecuaba completamente a los requerimientos; sólo a un subconjunto de estos. Además, se realizó una investigación de métricas existentes y una selección de las mismas que permitan cumplir con el objetivo planteado. La conclusión final fue realizar una herramienta que tomara datos de la realidad del Departamento y calculara las métricas seleccionadas, para luego realizar un estudio de las mismas, con las cuales elaborar un plan de mejora. Esto lleva a ejecutar ciclos continuos de: recolección de datos relevantes, realización de métricas, planificación de mejora y adaptación de la herramienta. De esta manera la herramienta se irá adaptando al Departamento y no el Departamento a la herramienta.

En este proyecto se construye un prototipo de una herramienta de Gestión de Software nombrada Arena, cuyo objetivo es cumplir con todas las funcionalidades que se consideran necesarias para la mejora del Departamento de Software. La herramienta deberá recolectar los datos para mostrar lo que se hace bien, lo que se hace mal, y los resultados de dichos planes. Organizando a la vez las actividades del Departamento.

[1] Software de Fuente Abierta o Código Abierto, del inglés Open Source Software, se refiere al software cuyo código está disponible públicamente, aunque los términos de licenciamiento específicos varían respecto a sus permisos de utilización.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

La exigencia por parte de los clientes de producir software más rápido, barato y sin defectos es cada día mayor. Establecer un proceso de mejora continua para cualquier empresa con afán de superación, es entonces, una necesidad imperiosa para obtener o mantener una posición competitiva. Los objetivos primordiales y esenciales de este proceso son reducir los tiempos de entrega, aumentar la calidad y productividad, y no sobrepasar los presupuestos preestablecidos.

En la actualidad existen varias metodologías de gestión de software, ver Apéndice A en la página 61. Las recomendaciones para su utilización se basan principalmente en el escenario de aplicación, tomando como criterio principal la cantidad de personas en la empresa. Las diferentes características, especialmente opuestas entre CMM [CMM Software], ver el Punto A.2.2.1 en la página 64, y XP [XP], ver el Punto A.2.3.1 en la página 67, nos sugiere la real importancia del estudio del escenario de aplicación y de las necesidades de las empresas.

Se han construido productos de software para facilitar la aplicación de cualquiera de estas metodologías, o simplemente para gestionar una empresa de esta área. En el Apéndice C en la página 81, se puede ver una síntesis de herramientas de dicha clase que se consideraron más relevantes.

Establecer un proceso de mejora continua en una empresa implica realizar una investigación minuciosa de sus características. Antes que nada se debe saber que es lo que está sucediendo, analizar la realidad, identificando los problemas que impiden llegar a la meta y sus causas. No se puede mejorar lo que no se puede medir, entonces, se debe medir el desempeño, tomando datos relevantes a las metas de la realidad, ver Apéndice B en la página 75. Luego se construyen estadísticas o indicadores de modo de evaluar el desempeño, para finalmente tomar medidas correctivas, i.e., iniciar el proceso de mejora continua.

El Departamento de Software de Abitab S.A.[1], nació como un equipo dedicado a un solo proyecto, evolucionando a lo que es el día de hoy un departamento con múltiples tareas y objetivos cambiantes. Su evolución fue rápida, las tareas se multiplicaron y los objetivos obedecen a las necesidades cambiantes del negocio. La exigencia de la empresa de una adaptación casi instantánea a sus necesidades generó en el departamento soluciones rápidas, que no siempre son las mejores. La realidad ha llevado a gestionar el Departamento con varias herramientas, lo que lleva a una descentralización que se busca eliminar dado que no asegura la coordinación. El exceso de tareas frente a los recursos con que se cuenta hace de su gestión especialmente particular. Se han analizado herramientas incluso desde antes del inicio de este proyecto y ninguna se adapta a su dinámica. Por ejemplo, adoptar una herramienta como XPlanner [XPlanner] sin adoptar XP [XP] no tiene sentido,

[1]Abitab S.A. es una red de prestación de servicios que cuenta con más de 300 locales en todo el país. Entre sus principales servicios se pueden citar: cobranza de servicios públicos, recaudación de impuestos, aportes a la seguridad social, tributos municipales, cobro de suscripciones y cuotas, cobro de tarjetas de crédito y préstamos, pago de jubilaciones, pensiones y salarios, venta de tarjetas telefónicas, de estacionamiento y órdenes de atención médica, así como transferencias y giros. http://www.abitab.com.uy

sin embargo una de sus funcionalidades, la integración de módulos, es una de las requeridas en la herramienta buscada y que no se ha encontrado en otra. Uno de los problemas que requieren de una rápida solución es poder demostrar las dificultades externas que se plantean en el transcurso de la realización de las tareas. La falta de definiciones concretas, los cambios constantes, e incluso la dedicación a productos cuyo negocio finalmente no se concreta, hacen muchas veces que los resultados no concuerden con el esfuerzo realizado. De todas las herramientas analizadas, incluso antes del inicio de este proyecto pueden adaptarse para facilitar su gestión, pero ninguna tiene un análisis de los datos que almacena. Tomar datos a modo de censo no es lo más adecuado y menos si estamos hablando de un ambiente de software. Es indiscutible que para tomar los datos de la realidad, lo mejor es hacerlo por medio de una herramienta, y más aún si queremos construir estadísticas en base a los mismos. Con lo que concluimos que una herramienta que permita gestionar el departamento y analice los datos de sus gestiones es necesaria para un proceso de mejora continua.

Este informe está organizado en diez capítulos y tres apéndices. Se tiene una Bibliografía al final del contenido principal, y una al final de cada apéndice. Existen dos tipos de referencias en el informe: las cruzadas, que citan puntos del mismo junto con la página donde se encuentran, y las bibliográficas que se encuentran entre paréntesis rectos. Este Capítulo (Capítulo 1), brinda una introducción al proyecto. El Capítulo 2 en la página siguiente, describe la historia, dinámica y problemas del Departamento. El Capítulo 3 en la página 14, describe los requerimientos del sistema, los usuarios del sistema y muestra los casos de uso UML [UML] principales de la herramienta. El Capítulo 4 en la página 30, describe los módulos, clases del sistema y sus asociaciones. El Capítulo 5 en la página 42, describe la tecnología utilizada y el plan de implementación. El Capítulo 6 en la página 44 describe los planes de testeo a realizar de la herramienta. En el Capítulo 7 en la página 52, se describe la planificación de la puesta en producción de la herramienta. En el Capítulo 8 en la página 53 se muestran los objetivos a medir y las métricas a utilizar. En el Capítulo 9 en la página 56 se plantean las conclusiones del proyecto. En el Capítulo 10 en la página 57 se plantean el camino a seguir en el futuro. El los apéndices se describe el estado del arte de la gestión de Software (Apéndice A en la página 61), la medición de Software (Apéndice B en la página 75), y las herramientas de gestión de Software (Apéndice C en la página 81).

Capítulo 2

Descripción del problema

Este Capítulo describe los problemas que dieron lugar a este proyecto. En la Sección 2.1 Planteo del problema, se muestra una reseña de las causas de la creación del Departamento (Subsección 2.1.1) y las actividades que se realizan en el mismo (Subsección 2.1.2 en la página siguiente). En la Sección 2.2 en la página 10, se citan las generalidades de la dinámica del negocio y las herramientas de gestión que se utilizan actualmente. El Capítulo termina con un análisis del problema, Sección 2.3 en la página 11.

2.1. Planteo del problema

2.1.1. Antecedentes

Comenzaremos por citar antecedentes que incidieron e inciden sobre la planificación y ejecución de las tareas diarias del departamento de Desarrollo.

La Empresa formalizó el Departamento de Desarrollo de Sistemas hace ocho años. En ese entonces la envergadura de la Empresa era diez veces menor que la actual. El crecimiento exponencial llevaba a que el software se desarrollara siempre con un enfoque parcial de cada servicio o producto que se deseaba incorporar a la Organización. Las aplicaciones se desarrollaban en Cobol [COBOL], C++ [C/C++] y Visual FoxPro [Microsoft Visual FoxPro]. En sus inicios contaba con tres integrantes, durante los primeros años se incorporaron sólo tres personas más.

En el año 2001 la empresa comenzó un emprendimiento cuyo objetivo era realizar las transacciones desde Central, es decir, construir un sistema basado en el modelo cliente servidor [1]. Para cumplir este objetivo era necesario construir un software que lo permitiera, lo que se denominó a nivel interno como "Proyecto On-line". Se organizó un equipo de trabajo con dedicación exclusiva. Constaba de once profesionales informáticos, entre otras disciplinas. Este Equipo se dedicó a planificar y llevar a cabo toda la plataforma actual basada en tecnología J2EE [J2EE].

A fines del año 2002, el Equipo realiza con éxito la puesta en producción de la primera etapa del proyecto. Por lo cual, la Dirección de la Organización decide incorporarlo a la línea de trabajo del Departamento de Desarrollo. Es decir, a partir de ese momento ya no pudo dedicarse a un solo objetivo, por el contrario, a partir de la puesta en producción de toda esta nueva plataforma tecnológica, se abrieron nuevas posibilidades, generando, entre otras, nuevas necesidades, nuevas oportunidades de negocio, y nuevas proyecciones estratégicas.

[1] El modelo cliente servidor describe dos procesos que interactúan entre si por medio de pedidos. El servidor es quien acepta los pedidos de servicio desde la red, lo realiza y devuelve el resultado al solicitante. Los programas que realizan las peticiones y esperan las respuestas son los llamados clientes.

En esta primera etapa, debido a la magnitud de la Empresa, la nueva infraestructura tecnológica no abarcó a todos los sectores. Gran parte de los procesos administrativos y de control de Casa Central no se han cambiado aún. Debido a utilidad vital de estos Sistemas Legacy[2], además de la necesidad de brindarles mantenimiento, se generó la necesidad de crear interfaces para unificar las plataformas tecnológicas.

Estos antecedentes se explicaron porque dieron lugar a una serie de grandes planteamientos y preguntas como las siguientes:

- Planificar un proyecto y trabajar abocados a un solo objetivo sin estar en producción es muy distinto a la situación actual, en la cual siempre hay más de diez objetivos al mismos tiempo.
- Actualmente la puesta en producción de un módulo lleva prácticamente tanto trabajo y tiempo como su desarrollo.
- Los puntos anteriores llevaron a que la Dirección de la Empresa se haga preguntas tales como:
  - Si en un año se pudo hacer el Proyecto más importante de la Empresa, ¿Por qué demoran tanto para hacer un módulo?
  - Antes había mucho menos presupuesto para el Departamento de Desarrollo, ¿Por qué ahora necesitamos mucho más?
  - ¿Que tareas hacen? Necesitamos tener justificado este presupuesto.

Estos son algunos ejemplos de los muchos planteos existentes. Este tipo de preguntas deben ser respondidas con fundamento para que la Organización mantenga la estabilidad y seguridad tecnológica que requiere para continuar brindando a sus clientes y usuarios la confianza y el buen servicio de siempre.

Se debe partir de un principio fundamental: el trabajo de construir software no es tangible. Sólo se percibe cuando pasa a ser parte de un producto rentable. Esto es un piedra angular para cualquier Empresa. Por tal motivo no basta con trabajar bien hay que mostrar y cuantificar lo bien o mal que un Equipo de trabajo hace sus tareas. Quienes tiene el rol de informar a los Directivos de la Organización siempre deben partir de la base que el trabajo creativo, técnico e intelectual no siempre es fácil de comprender para quienes gestionan las finanzas de la empresa.

Cabe resaltar que la principal actividad de la Empresa no es el Desarrollo de Software. Por el tipo de rubro al que pertenece, servicios, es claro que la tecnología juega un rol fundamental, pero no es un objetivo en sí mismo, sino un medio para cumplirlos.

2.1.2. Diversidad de tareas del departamento

Presentado los antecedentes no es difícil imaginar las distintas tareas que se hacen en el Departamento de Desarrollo. Inclusive, hay muchas que ni siquiera son propias de un Equipo de Desarrollo. Y esto último es muy importante de destacar. El Departamento de Desarrollo está inmerso dentro de un Área muy nueva dentro de la Empresa: El Área Tecnológica. Esta Área fue creada en el año 2003 y está comenzando a implementar otros departamentos que probablemente tenga a su cargo varias de las actividades que hoy las realiza Desarrollo.

A continuación se describen brevemente las principales actividades del Departamento:

Relevar y analizar requerimientos: Esta tarea supone reunirse con el Responsable Funcional de un sector y parte de su personal para que estos expresen que funcionalidades requieren de la aplicación

[2]Sistemas Legacy: Aplicaciones de hardware y software en las cuales una empresa ha invertido considerable tiempo y dinero. Típicamente, los Sistemas Legacy realizan operaciones críticas en empresas por muchos años aunque no usen la tecnología actual. Reemplazar Sistemas Legacy requiere de una planificación cuidadosa y un soporte adecuado del sus creadores para su migración. [ISPE]

a desarrollar. En la Empresa cada sector es un cliente interno de Desarrollo. Estos no tienen tiempo para poder llevar a cabo esta tarea de la forma que debería hacerse. A eso se suma que hay varios funcionarios del Equipo de Desarrollo que tiene mucha experiencia en la Empresa y las Áreas delegan parte de la definición de estos requerimientos a los propios Desarrolladores. Tal es así, que Desarrollo hace los requerimientos y luego el Responsable Funcional los valida. Luego de la aprobación del Responsable Funcional se realiza el análisis de requerimientos.

Diseño funcional de las aplicaciones: Implica realizar los casos de uso y describir conjuntamente con el Responsable Funcional cuales serán los requerimientos funcionales.

Análisis y diseño técnico: Esta generalmente es desempeñada por quienes han estado en las tareas anteriores. Significa definir técnicamente la solución de cada requerimiento llegando a la especificación de la forma en la cual realizar la implementación tan detalladamente como sea posible.

Desarrollo de aplicaciones: Con la documentación e información que se obtiene de la tarea anterior se implementan los programas y componentes tecnológicos.

Prueba y puesta a punto de las aplicaciones: Las pruebas a nivel modular de la aplicación sobre la plataforma de desarrollo también son realizadas por integrantes del Equipo de Desarrollo. Pero a su vez, la coordinación, organización de las pruebas integrales sobre producción, e inclusive ejecución de estas, también son menester de integrantes de Desarrollo.

Implantación de aplicaciones: Igualmente que en el punto anterior, Desarrollo interviene directamente en la implantación y puesta en producción de las aplicaciones. Coordinando no sólo aspectos técnicos sino también parte de la logística y coordinación con otros sectores.

Mantenimiento y soporte técnico de los Sistemas Legacy[3]: Como se destacó en la Sección 2.1.1 en la página 7, existen más de veinte aplicaciones que siguen siendo utilizadas en los procesos diarios de la Empresa. El soporte técnico y administración de nuevos requerimientos y solución de errores deben ser atendidos de forma inmediata.

Mantenimiento y soporte técnico de aplicaciones On-line: Estas son las aplicaciones que dan lugar a la actividad principal de la Organización. Debido a que el tipo de servicio que presta la Empresa es en tiempo real, cuando surgen problemas generalmente deben ser solucionados lo antes posible. Una falla en el sistema afecta inmediatamente al servicio que prestan las más de mil terminales conectadas en línea. Solucionar los problemas sin perder el servicio implica estar muy coordinados y ser muy precisos en la soluciones. El soporte técnico de esta nueva plataforma de aplicaciones, muchas veces, es solicitado por los distintos sectores de la Organización, ya que no tienen un conocimiento integrado y preciso de las aplicaciones que se utilizan. No sólo no han podido incorporar aún el cambio de software, tampoco los que han habido en los procesos y las premisas que tiene la Empresa. La mayoría de estas solicitudes de soporte técnico no son errores, ni falta de funcionalidad de la aplicación, igualmente, Desarrollo colabora y responde a las inquietudes que cualquier sector plantee.

Investigación de nuevas tecnologías: Mantenerse como una de las Empresas del medio a la vanguardia de la tecnología es considerado un objetivo importante para la Organización. Por tal motivo, hay muchas actividades que coordinar y abarcar en el terreno de la investigación e incorporación de nuevas tecnologías.

Desarrollo de procedimientos de trabajo: Es parte de quienes organizan y coordinan el Departamento construir y documentar los procedimientos que se utilizan para desarrollar las distintas tareas. En realidad, muchos aún no están documentados, otros deben ser mejorados, y carecen de control, lo cual les resta efectividad. En el futuro, esta tarea corresponderá a otro departamento.

[3]Sistemas Legacy: Aplicaciones de hardware y software en las cuales una empresa ha invertido considerable tiempo y dinero. Típicamente, los Sistemas Legacy realizan operaciones críticas en empresas por muchos años aunque no usen la tecnología actual. Reemplazar Sistemas Legacy requiere de una planificación cuidadosa y un soporte adecuado del sus creadores para su migración. [ISPE]

Administración de la base de datos: Por el momento la administración de la base de datos y todas las maniobras, que deben ser coordinadas con extrema mesura y precisión, son realizadas por el Departamento y los proveedores especialistas en el tema.

Administración del software de base de los servidores de la plataforma: Esta tarea implica tener actualizado el sistema operativo de los servidores centrales y realizar el seguimiento correspondiente.

Capacitación a otras áreas de la Empresa: Cada vez que se libera una nueva versión de una aplicación, se generan las notas de la misma[4] con los cambios y las nuevas funcionalidades. Pero además, se instruye a los sectores directamente involucrados, para asegurar y facilitar el aprendizaje del conocimiento de dichos cambios y funcionalidades.

Cursos de educación continua: En el propio Equipo de trabajo se detectan las debilidades técnicas y se organizan y dictan cursos y seminarios, con el fin de homogeneizar o ampliar los conocimientos.

Asesoramiento y consultoría para negocios B2B[5]: Uno de las principales frentes de negocio es brindar servicios en línea de otras empresas. Esto implica un conexión directa a sus servidores, del término en inglés "Host to Host". El éxito de este emprendimiento, depende del involucramiento del Departamento desde el primer contacto comercial hasta su implantación.

Asesoramiento y consultoría al Área Comercial: Generalmente en cada nuevo negocio que emprende el Área Comercial en alguna instancia aparece el componente tecnológico. En ese momento se cita la consultoría de alguno de los integrantes de Desarrollo para darle viabilidad técnica y contribuir a la planificación de la realización del negocio.

Documentación: Se realiza la documentación de algunos de los procesos de la empresa, manuales de usuario, actas de reunión, y reportes para la gerencia de la empresa. Estos documentos se realizan en base a estándares implantados por la Empresa. Se realiza la administración de versiones de la documentación que genera el Departamento y la otorgada por los clientes para realizar las tareas. Estos documentos son clasificados por área temática o proyecto según corresponda.

2.2. Dinámica del negocio

La dinámica del negocio requiere la capacidad de adaptarse rápidamente al cambio, es una propiedad vital para cualquier sector de la Organización. Sobre todo el saber adaptarse a cualquier cambio en las necesidades planteadas por los clientes externos a la empresa. Esta sinergia de los sectores es un diferencial ante otras empresas del ramo. Las actividades pueden cambiar de prioridad prácticamente a diario. Para poder llevar a cabo con éxito estos cambios tan frecuentes es necesario planificar y volver a planificar. Sobre todo para no perder de vista los objetivos que ayer fueron marcados como los más importantes y hoy son secundarios.

Las herramientas que se utilizan hoy día no han permitido manejar este dinamismo de una forma eficiente. Para intentar lograrlo se utilizan varias técnicas:

- Administradores de correo y agenda
- Cronogramas de proyectos
- Planillas de cálculo
- Notas y apuntes

Sin lugar a dudas que la tarea de documentar, planificar y realizar el seguimiento de estas actividades es muy compleja e implica mucho trabajo. Si a eso le sumamos que las herramientas utilizadas

[4] En inglés, release notes
[5] Del término en inglés: Business to Business, empresa a empresa.

no son las más adecuadas, la complejidad aumenta. Pero sobre todo, es prácticamente imposible obtener datos cuantitativos y estadísticos que permitan medir y analizar resultados.

Otro problema de esta carencia es no poder publicar de forma centralizada la información de la planificación y ejecución de tareas. Cuando se desea hacer un buen trabajo en Equipo es muy importante la comunicación. Una de las cosas más importantes que hay que comunicar son los responsables de cada tarea y su avance en la misma.

Sería oportuno proveer a cada integrante de la información necesaria para que pueda medir su propio rendimiento, permitiendo documentar que fracción de tiempo destina a que actividad, y todos los datos personales y de los equipos necesarios para la mejora continua de la calidad de los productos.

2.3. Análisis de problema

A continuación se enumeran los diferentes problemas que se encontraron en la gestión del Departamento:

1. Unificación Los documentos de requerimientos, diseño, planificación y muestra de resultados, y las actas de reunión, se mantienen separados, por diferentes personas, en diferentes lugares, sin ninguna unión explícita.
2. Seguimiento La comunicación del estado de las tareas se realiza de persona a persona y por requerimiento, en archivos separados por producto, gestionados por una sola persona. La comunicación del estado de las tareas, en algunos casos es indirecta, lo que puede llevar no sólo a la pérdida de la información sino también a su deformación.
3. Pérdida de información La falta de seguimiento y unificación, citadas en los puntos 1 y 2, genera pérdida de la información.
4. Versionamiento El versionamiento se gestiona a nivel de archivos, y de la instalación en el servidor de la integración de los productos. No existe un versionamiento de los productos y sus módulos.
5. Reportes de errores y pedidos de mejora. Estos son tratados como una tarea más, lo que lleva a un informe incorrecto de los tiempos.
6. Clasificación de tareas La clasificación de tareas por tipo, como por ejemplo soporte, implantación, capacitación, o testeo, no se realiza en la creación de las mismas, sino a la hora de generar un informe que lo requiera.
7. Ponderación de tareas No existe ninguna herramienta, que ordene automáticamente las tareas asignadas por prioridad, ni que registre la historia de los cambios de estas prioridades.
8. Integración de Productos La integración de los productos se realiza a partir de las últimas versiones de los archivos que lo componen, por una sola persona. La persona que integra no tiene acceso explícito a todos los cambios realizados en el código, ni de los requerimientos necesarios para la correcta ejecución de los mismos. Esta forma de proceder genera errores que son difíciles de seguir, primero porque la persona que genera la instalación no esta necesariamente al tanto de todos los cambios, y segundo porque las personas que los realizaron están en otra tarea, o en otra fase de la misma, debiendo rastrear los cambios que están incluidos en la integración.
9. Iteraciones Por lo general, al iniciar la creación de una nueva funcionalidad, se identifican todos los requerimientos necesarios, seleccionando subconjuntos por prioridad a ser implementados. Estas iteraciones no se identifican como tales, sino que cada requerimiento es una tarea.

10. Comunicación Algunos de los problemas de comunicación se vieron en los puntos 1, 2, 3, y 8. Una de las soluciones a los problemas es una reunión semanal de todo el Departamento. Dado la dinámica del negocio planteada en la Sección 2.2 en la página 10, esto es casi imposible. Incluso cuando se realizaron reuniones de todo el equipo, estas fueron interrumpidas por consultas y pedidos telefónicos y más de una vez alguno de los integrantes debió retirarse para realizar otra tarea en ese mismo momento.
11. Costos En la actualidad no se genera ningún reporte de costos.
12. Gestión de recursos humanos La gestión de recursos humanos hoy en día se realiza en el Área de Recursos Humanos de la empresa.
13. Rendimiento El rendimiento de cada integrante del equipo, si mejora o empeora es subjetivo.
14. Reportes Los reportes son generados manualmente. Hoy muchas horas son destinadas a lograr conjugar la "muestra de resultados" que deben ser armados como un rompecabezas de información.
15. Documentación La documentación no esta estrictamente ligada a las tareas que se realizan, no se establece cual de las versiones de los documentos fue utilizada.

No todos los problemas que tiene el Departamento de Desarrollo son causa de la gestión de Software, algunos como se vio en la Sección 2.1 en la página 7, tienen que ver con la organización y logística de la empresa, y la falta de conocimiento y adaptabilidad a las nuevas aplicaciones.

Dada la diversidad, magnitud y criticidad de tareas que realiza el Departamento de Desarrollo la administración, planificación, coordinación y ejecución de actividades juega un rol fundamental para que el trabajo sea realizado exitosamente.

La unificación de los datos que hoy están distribuidos en planillas electrónicas en Ms Excel [Microsoft Excel], cronogramas en Ms Project [Microsoft Project], documentos, notas, correo electrónico, tareas planificadas en Ms Outlook [Microsoft Outlook], evitaría la pérdida de información y brindaría una mayor visibilidad del estado del Departamento.

Generar automáticamente el costo de los módulos y sus mantenimientos en base al negocio que hacen posible es una oportunidad que esta perdiendo el Departamento de mostrar una justificación más de ser: su costo para lo que da a ganar.

Un foro de discusión puede mitigar puntos débiles en la comunicación, que permita registrar y mantener ordenadas, públicas, y de fácil acceso, las sugerencias, inquietudes y cuestionamientos de los integrantes del equipo. Un panel de noticias, donde se publiquen por ejemplo las nuevas funcionalidades que pueden ser útiles para otros módulos, contribuyendo a la reutilización de código.

Organizar las tareas en iteraciones y fases, ponderándolas y clasificándolas de manera adecuada, no sólo implica visibilidad y mejora de procedimientos y métodos, sino también genera las posibilidad de crear reportes automáticos con estos datos.

La correcciones de errores debería incluirse en el tiempo de la tarea correspondiente, y registrarse como tal, identificar su causa y documentarla como una forma de evitar volver a cometer el mismo error, y que otros tengan acceso al problema y la forma en la cual fue solucionado.

Los pedidos de mejora, no son tareas que fueron tomadas en cuenta a la hora de realizar la estimación de tiempos iniciales. Se debería dejar constancia de esto, principalmente para quienes realizan este pedido, de manera de poder ver claramente el impacto de los mismos.

Es necesaria la creación de alguna métrica, índice o tendencia para el rendimiento de los integrantes del Departamento. Al menos un registro del tiempo que lleva realizar una tarea, y la cantidad de errores y su tiempo de corrección. Lo que no puede ser medido no puede ser mejorado.

Para hacer los reportes citados en el punto 11 del Capítulo anterior, y para gestionar la disponibilidad de estos recursos, esto es, marcar los días feriados, las licencias, o cualquier día que no puedan

asistir por cualquier otra razón; se debería tener una gestión de Recursos Humanos mínima interna al departamento.

Todos los puntos mostrados anteriormente, con excepción de los citados en el primer párrafo, pueden implementarse en una herramientas de gestión de software.

Capítulo 3

Requerimientos del sistema

En base a los problemas planteados en el Capítulo 2 en la página 7, se decidió realizar una herramienta para gestionar el Departamento, y recolectar los datos relevantes para construir métricas que permitan mostrar su realidad. En este Capítulo se especifican los requerimientos de dicha herramienta.

3.1. Introducción

3.1.1. Objetivo

El objetivo de este Capítulo es especificar los requerimientos de la herramienta de gestión de software a construir como parte del proyecto de grado.

3.1.2. Alcance

El producto a desarrollar se denomina Arena, un gestor de desarrollo de software.

Este producto gestionará recursos humanos, tareas, proyectos, y pedidos de mejora, nuevas funcionalidades, y corrección de errores.

Este software tiene como objetivo organizar y mantener en una misma herramienta las tareas realizadas por el Departamento de Desarrollo, permitiendo obtener y establecer información adecuada para evaluar su rendimiento, causas de retraso, costos y medidas de mejora.

3.1.3. Definiciones, siglas y abreviaturas

Actividades:

Conjunto de tareas. Por ejemplo desarrollo o mantenimiento.

Ítem:

Cada uno de los elementos que forman parte de un dato.

Rama de un proyecto:

Subdivisión de proyecto.

3.1.4. Visión General

Este Capítulo está organizado según IEEE Std 830-1998 (ver [IEEE STD 830-1998]). La Sección 3.2, contiene la descripción general del sistema. En la Sección 3.3 en la página 18, se describen los usuarios del sistema. La Sección 3.4 en la página 18, muestra los casos de uso principales del sistema. Y en la Sección 3.5 en la página 28, se describe la proyección del sistema.

3.2. Descripción General

3.2.1. Perspectiva del producto

Este producto no interactúa con ningún otro sistema.

3.2.2. Requerimientos Funcionales

3.2.2.1. Gestión de recursos humanos

1. El sistema mantendrá los datos de personas, estos son:
   a) Nombres
   b) Apellidos
   c) Teléfonos
   d) Direcciones de correo electrónico
   e) Cédula de identidad
   f) Fecha de nacimiento
   g) Modalidad de contrato (empleado, contratado, proveedor de servicio, etc.)
   h) Fecha de término de contrato
   i) Costo por hora
   j) Disponibilidad

3.2.2.2. Tareas

Estas tareas son trabajos a realizar. Existirán dos clases de tareas activas y efectivas.

1. Las tareas tendrán un usuario responsable en un momento dado.
   a) El cambio de asignación de una tarea deberá justificarse.
   b) Los cambios de asignación quedarán registrados.
2. El acceso a una tarea será restringido.
3. Las tareas podrán registrar tareas predecesoras y sucesoras.

3.2.2.2.1. Tareas activas

1. No tienen alcance definido.
2. Estas estarán asociadas a un proyecto, a una rama de un proyecto, o a otra tarea activa.
3. Tendrán un tiempo asociado que será la suma de los tiempos de sus tareas efectivas descendientes.
4. Tendrán un estado asociado, estos son (ver estados tareas efectivas en esta página):
   - Planificación: No existe ninguna tarea efectiva asociada.
   - Activa: Tiene al menos una tarea efectiva asociada en estado agendada, aceptada, pendiente, en curso, o suspendida.
   - Inactiva: Todas las tareas efectivas asociadas están en estado finalizada o archivada.

3.2.2.2.2. Tareas efectivas

1. Tendrán fecha de inicio y fin, planificado y real.
2. Tienen la especificación detallada de cada punto a realizar: alcance.
3. Tendrán asociado el tiempo que llevo realizarla.
4. Los tiempos registrados en estas tareas serán los tiempos de trabajo asociados al usuario responsable.
5. Estas estarán asociadas a una tarea activa.
6. Las tareas podrán ser de los tipos: nueva tarea, error, mejora, extensión, modificación.
7. La tareas tendrán un estado asociado que son:
   - agendada: La tarea fue asignada no se ha tomado ninguna acción sobre ella.
   - aceptada: El usuario responsable ha entendido y aceptado la tarea.
   - pendiente: El usuario debe comenzar la tarea en cuanto pueda.
   - en curso: El usuario esta resolviendo la tarea.
   - suspendida: Una tarea de mayor prioridad le ha sido asignada al usuario, pero deberá continuar con esta en el futuro.
   - finalizada: El usuario dio por terminada la tarea.
   - archivada: La tarea fue cancelada definitivamente, no se realizarán más acciones sobre esta.

3.2.2.3. Actividades

1. La actividades son clases de tareas. Por ejemplo: mantenimiento, desarrollo, o cambio de herramienta.

3.2.2.4. Proyectos

1. El sistema gestionará proyectos.
2. Los proyectos tendrán asociados una lista de las personas que trabajan en él.
3. El sistema mantendrá la cantidad de horas trabajadas en cada proyecto.
4. Los proyectos contendrán ramas.
5. Ramas
   a) Las ramas tendrán un usuario responsable,
   b) Las ramas podrán contener otras ramas.
   c) Las ramas tendrán prioridad.
   d) Las ramas tendrán estado.
   e) El acceso a cada rama será restringido.
6. Los proyectos tendrán iteraciones.
   a) La iteraciones agrupan las tareas efectivas a ser implementadas en un momento dado.
   b) Las iteraciones tendrán fecha de inicio y fin.
7. Los proyectos podrán tener asociados documentos.
8. Los proyectos podrán tener asociados notas.
9. El acceso a un proyecto será restringido.

3.2.2.5. Gestión de pedidos

1. El sistema gestionará pedidos.
2. Estos tendrán una identificación que será utilizada para su seguimiento. Esta identificación será ingresada por el usuario del Departamento que corresponda, en el momento que lo reciba.
3. Tendrán un título que lo identifique descriptivamente.
4. Los pedidos tendrán mensajes asociados que contendrán:
   a) Usuario que lo envía
   b) Fecha
   c) Descripción
5. Los usuarios podrán ingresar mensajes dejando constancia de una acción tomada al respecto. El primer mensaje corresponderá ser enviado por la persona que reporte el error.

3.2.2.6. Gestión de versiones de productos

1. Los usuarios podrán ingresar notas[1], declarando que nuevas características, cambios y/o correcciones se han agregado al sistema de administración de archivos de la empresa.
2. El acceso a esta gestión será restringido.
3. El usuario que corresponda, al que se le asignarán los permisos adecuados, podrá declarar haber puesto en producción los módulos administrados, pudiendo además, obtener todas las notas de los desarrolladores, desde la versión del producto anterior, en un solo archivo.

[1]En inglés, release notes.

3.2.2.7. Seguridad del sistema

1. La seguridad estará basada en permisos.
2. Los permisos se definirán como asociaciones entre usuarios y los objetos del sistema.
3. El sistema tendrá un usuario Administrador de permisos.

3.2.3. Requerimientos no funcionales

- El sistema será fácil de usar.
- El sistema será un sitio web.
- El usuario sólo deberá poder ingresar al sistema ingresando un usuario y contraseña previamente asignados.
- El sistema verificará usuario y contraseña.
- El sistema soportará múltiples usuarios.

3.3. Usuarios

Se identifican claramente tres tipos de usuarios en el sistema, estos son: Desarrollador, Cliente y Administrador de permisos.

Desarrollador

El desarrollador es el usuario principal del sistema. Es quien lo usara para organizar sus proyectos, tareas y pedidos.

Cliente

Este usuario cataloga a los usuarios externos al sistema. Estos interactúan con la empresa o departamento de software. Este podrá generar los pedidos de mejora, nuevas funcionalidades o corrección de errores, correspondientes a los productos que le provee dicha empresa o departamento.

Administrador de permisos

El sistema contendrá un usuario que será el responsable de asignar los permisos de acceso al resto de los usuarios tanto Clientes como Desarrolladores.

3.4. Casos de uso principales

Esta Sección describe los casos de uso UML [UML] del sistema derivados de los requerimientos establecidos en la Sección 3.2 en la página 15.

3.4.1. Objetos generales

En esta Sección se presentan los casos de uso generales, que son extendidos por los casos de uso principales del sistema.

3.4.1.1. Ítems históricos

Los ítems históricos son todos aquellos objetos del sistema que no son borrados, sino que tienen un período de vigencia. Al crear un ítems histórico el sistema automáticamente generará un registro de alta del mismo. Al modificarlo, el sistema generará un registro de baja, y un nuevo ítem histórico con su correspondiente registro de alta. Si se desea borrar un ítem histórico que no hace referencia a ningún objeto del sistema este se borrará junto con su alta y/o baja asociadas.

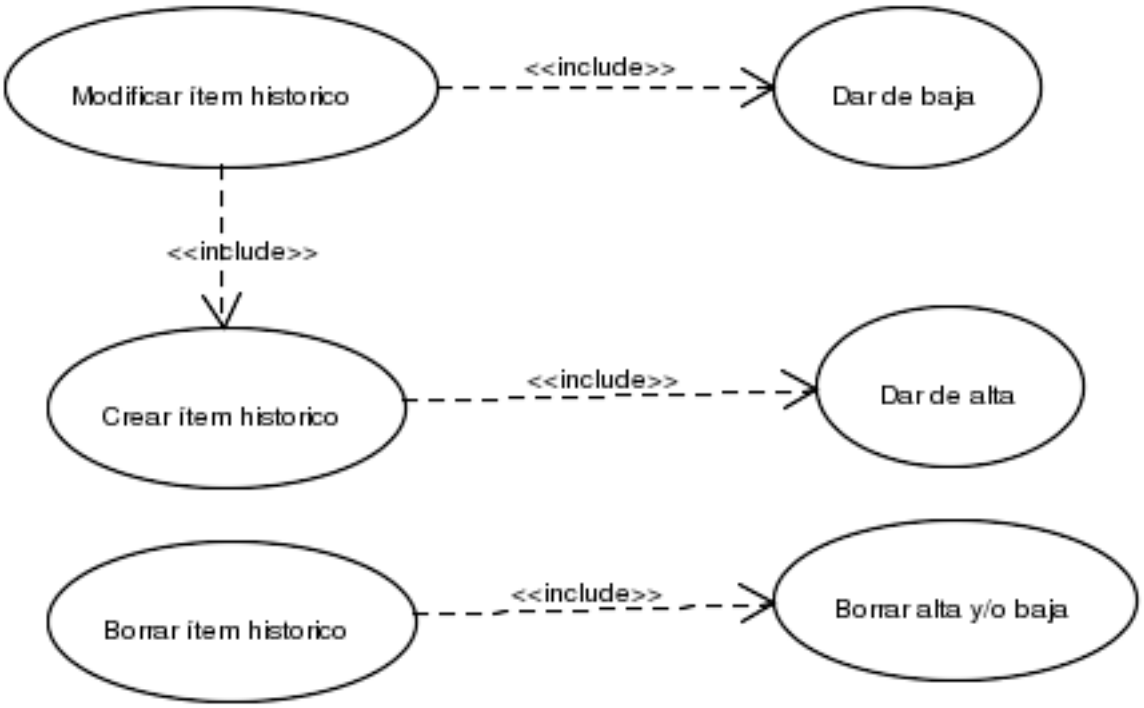

Figura 3.1: Casos de uso de ítems históricos

3.4.1.2. Ítems de trabajo

Los ítems de trabajo son una abstracción de las tareas, proyectos, ramas e iteraciones. Un responsable es asignado a cada ítem de trabajo, esta asignación es un ítem histórico, ver Figura 3.2 en la página siguiente.

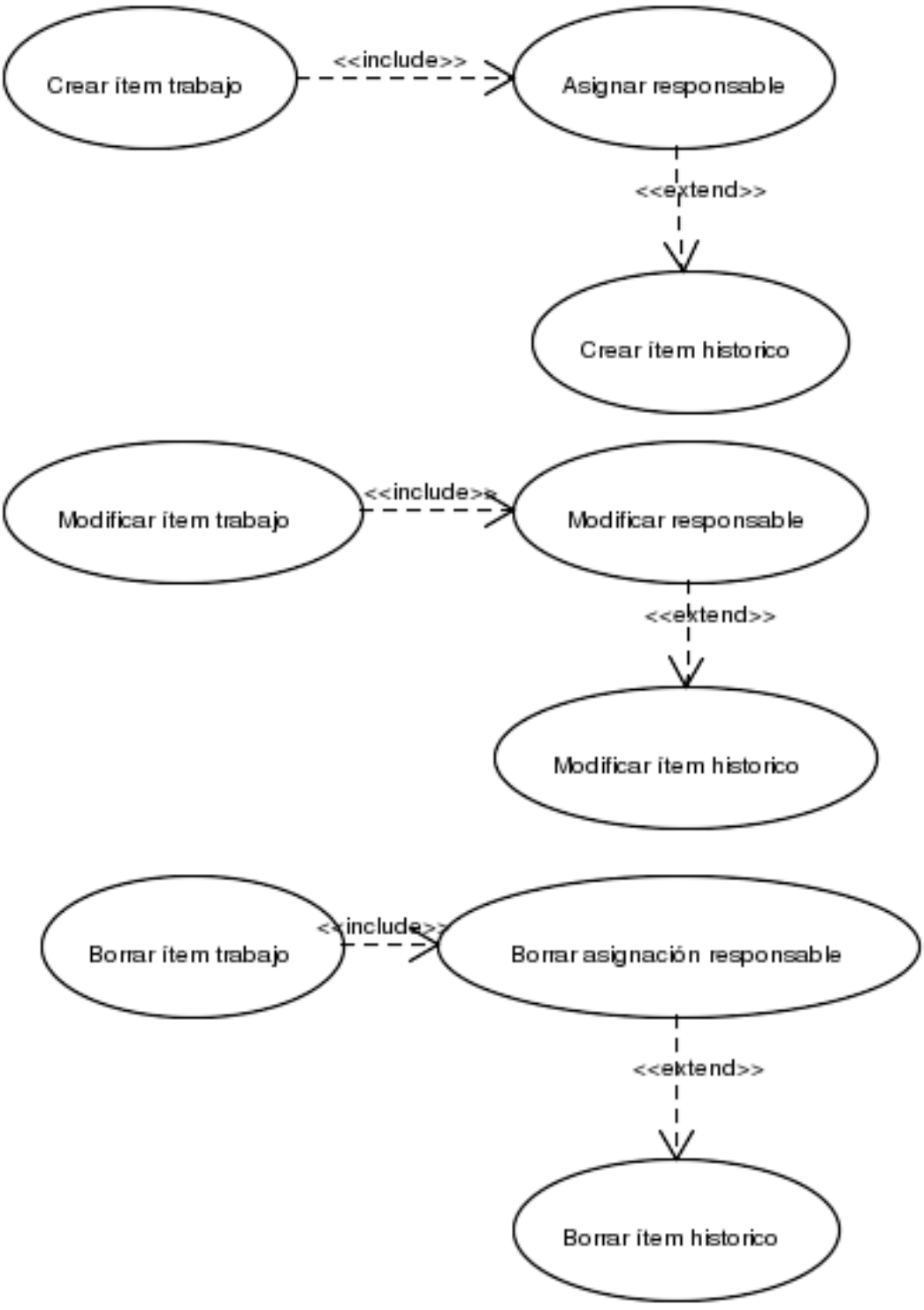

Figura 3.2: Casos de uso ítems de trabajo

3.4.2. Tareas

En esta sección se presentan los casos de uso para crear, modificar y borrar tareas.

Tanto las tareas activas como efectivas son ítems de trabajo, la creación de las mismas extiende el caso de uso “Crear ítem trabajo”, ver Figura 3.2 en la página anterior y 3.3.

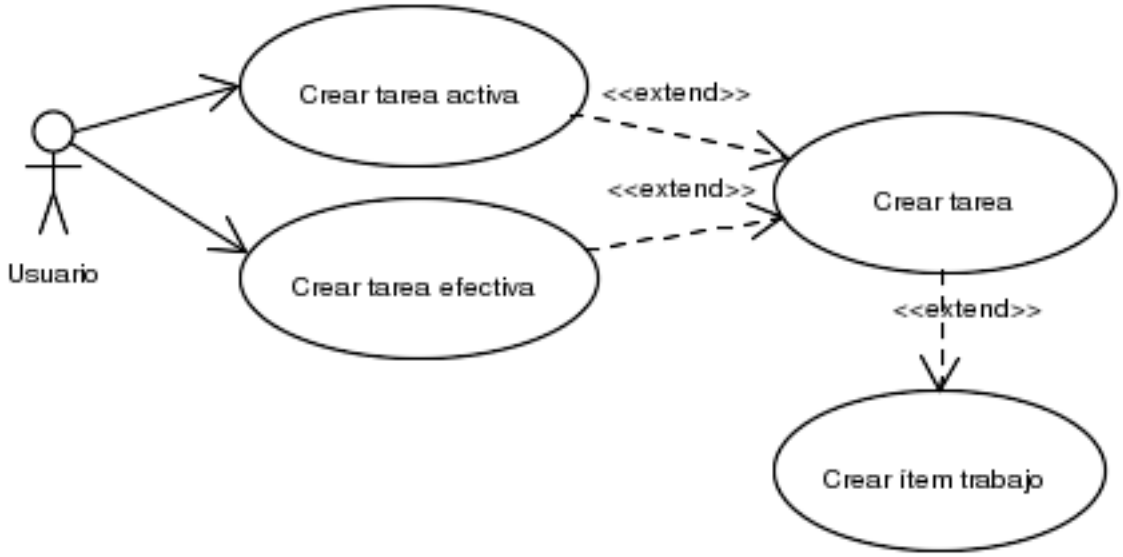

Figura 3.3: Casos de uso para crear tareas

La modificación de tareas extiende el caso de uso “Modificar ítem trabajo”, ver Figura 3.2 en la página anterior y 3.4 en la página siguiente.

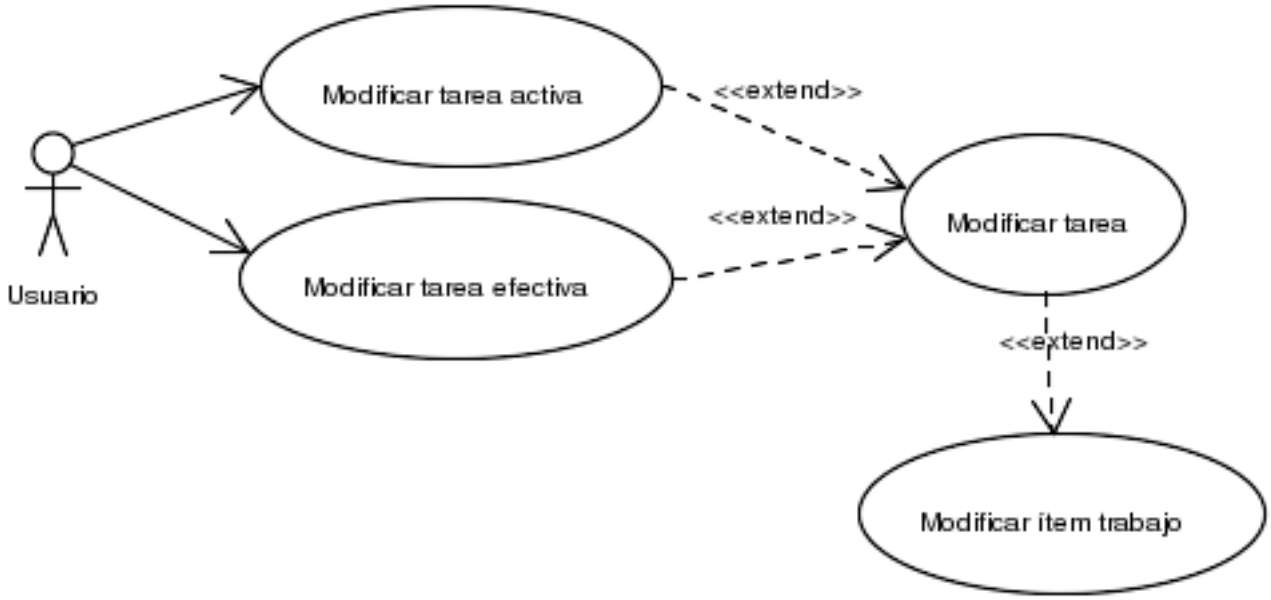

Figura 3.4: Casos de uso de modificación de una tarea

El borrado de tareas extiende el caso de uso “Borrar ítem trabajo”, ver Figura 3.2 en la página 20 y 3.5 en la página siguiente.

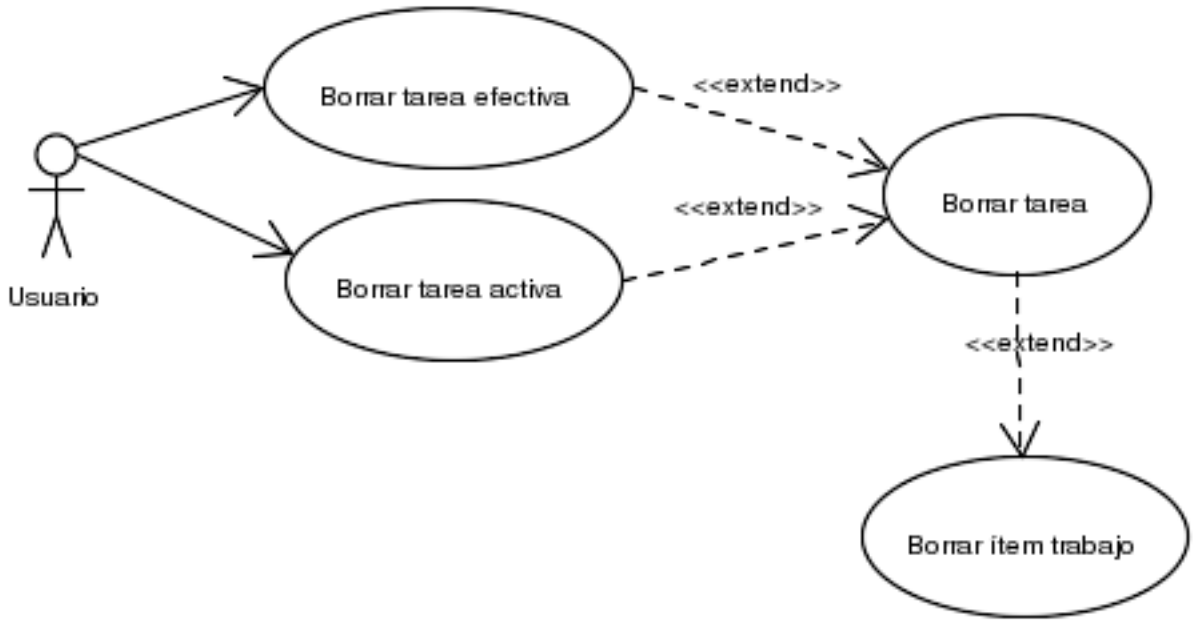

Figura 3.5: Casos de uso de borrado de tareas

3.4.3. Proyectos

La creación, modificación y borrado de iteraciones, ramas y proyectos extienden los casos de uso de creación, modificación y borrado de ítems de trabajo respectivamente, ver figuras 3.6 en la página siguiente, 3.7 en la página siguiente, 3.8 en la página 25, y 3.2 en la página 20.

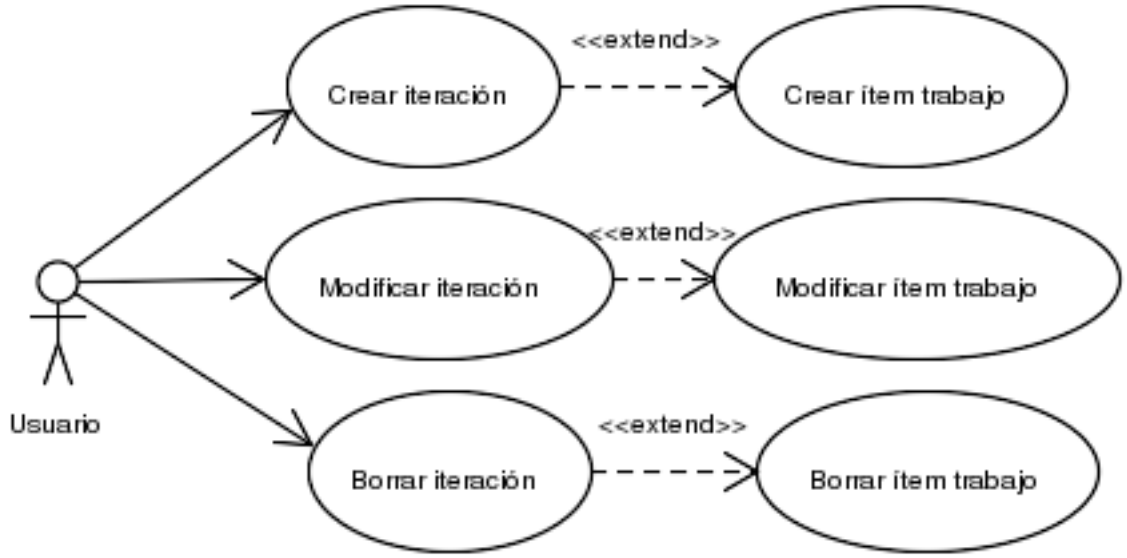

Figura 3.6: Casos de uso de iteraciones

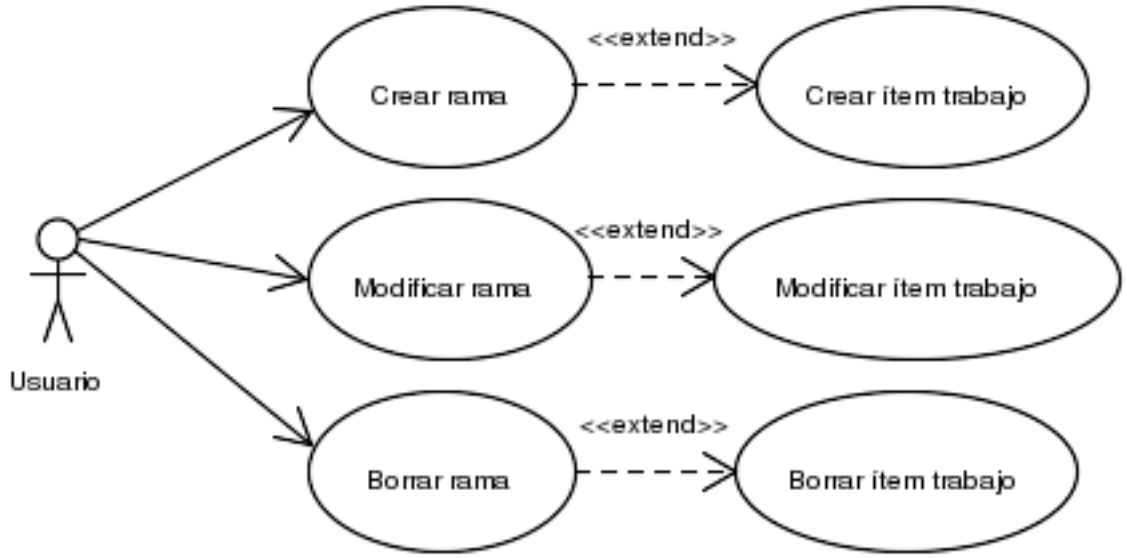

Figura 3.7: Casos de uso de ramas de un proyecto

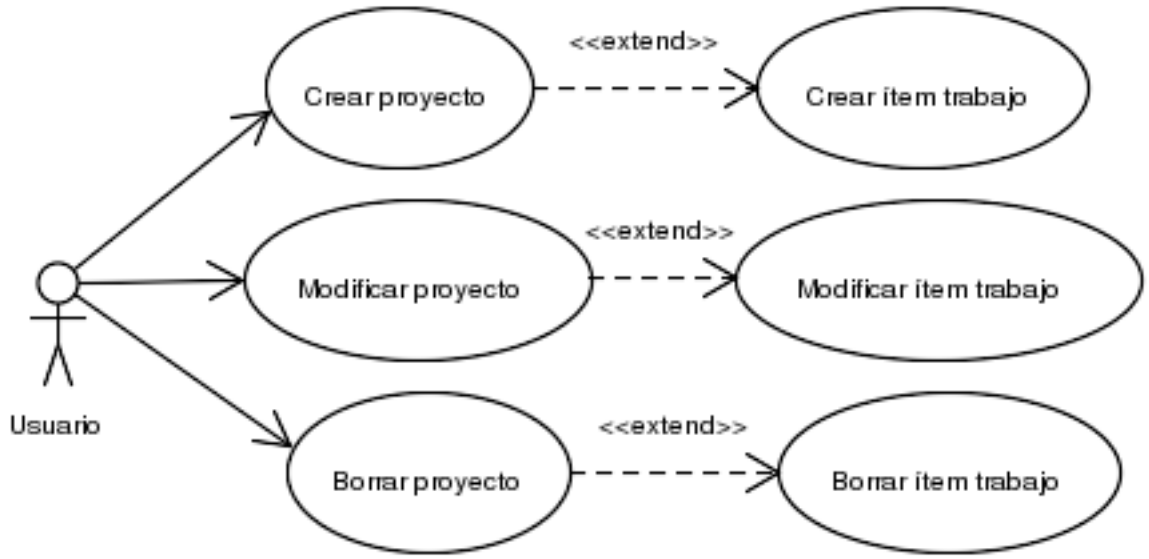

Figura 3.8: Casos de uso de proyectos

3.4.4. Pedidos

La creación de un pedido, implica generar un identificador y crear un mensaje. A un pedido se podrá agregar un nuevo mensaje asociado. Ver Figura 3.9.

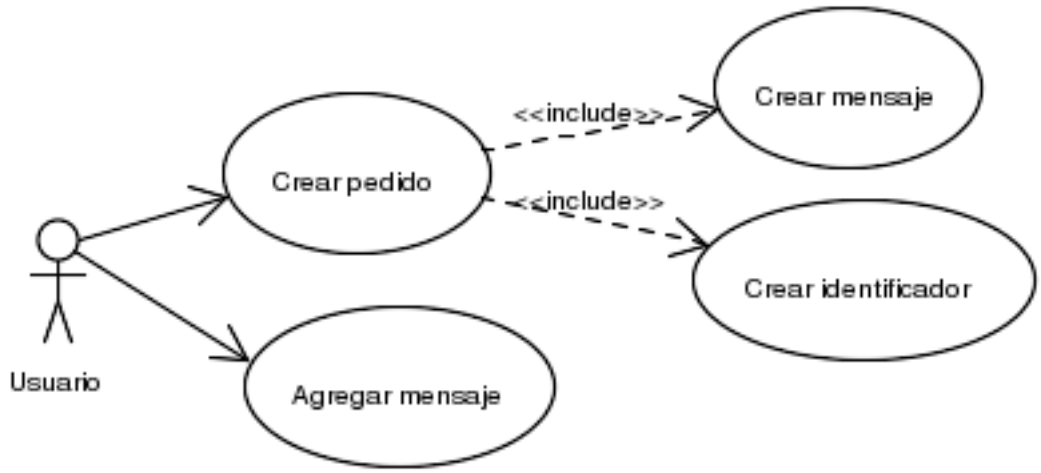

Figura 3.9: Casos de uso pedidos

3.4.5. Recursos Humanos

Al crear, modificar o borrar un desarrollador, se crea, modifica y/o borra una persona, su costo por hora, y su contrato respectivamente. Al crear, modificar o borrar una persona asociada a una

empresa, se crea, modifica y borra una persona respectivamente. Ver Figura 3.10 en la página siguiente.

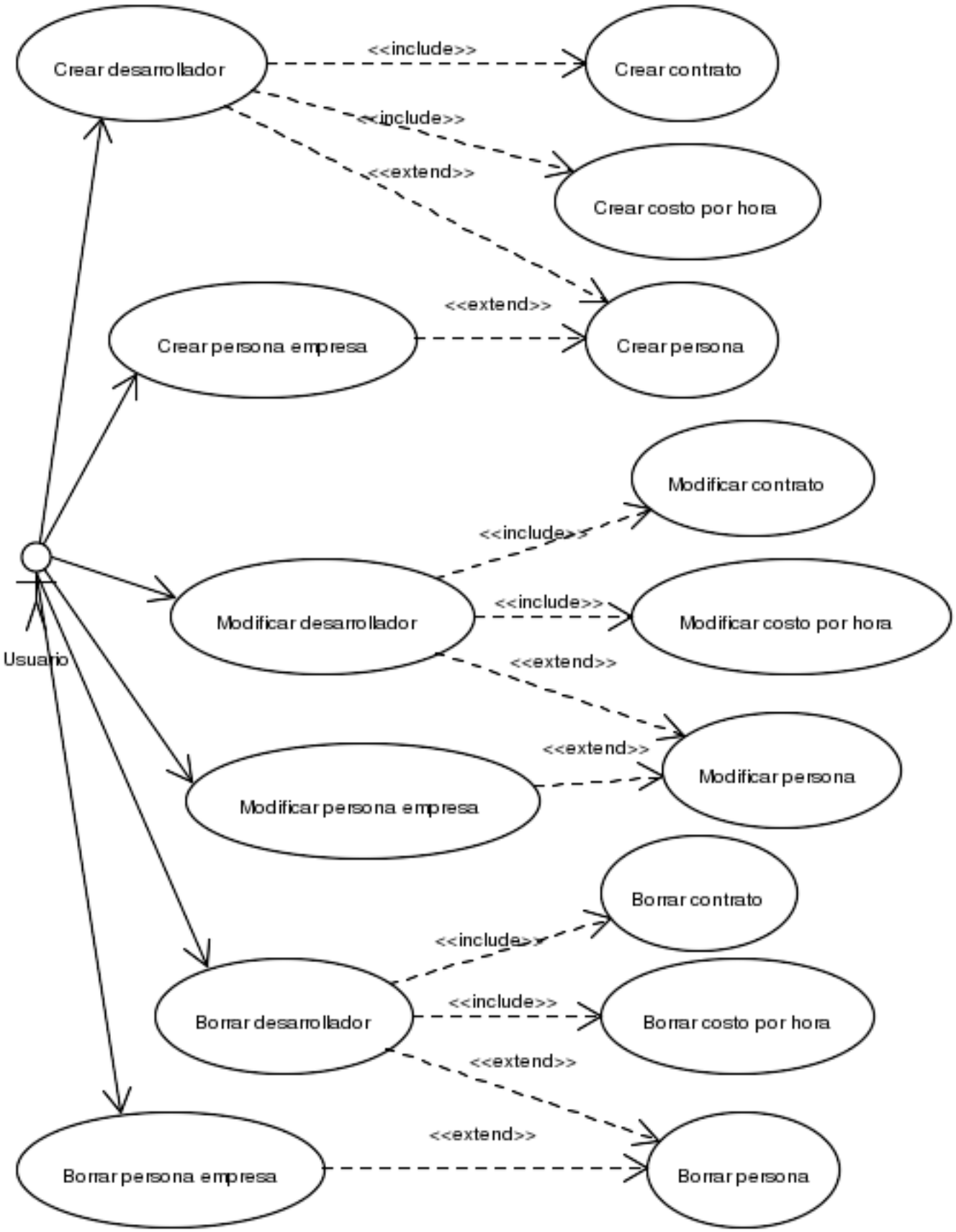

Figura 3.10: Casos de uso de recursos humanos

3.4.6. Seguridad

El sistema tendrá un usuario que podrá administrar los permisos de uso y manipulación de los proyectos, ramas de proyectos, recursos humanos y pedidos, del resto de los usuarios . Ver Figura 3.11.

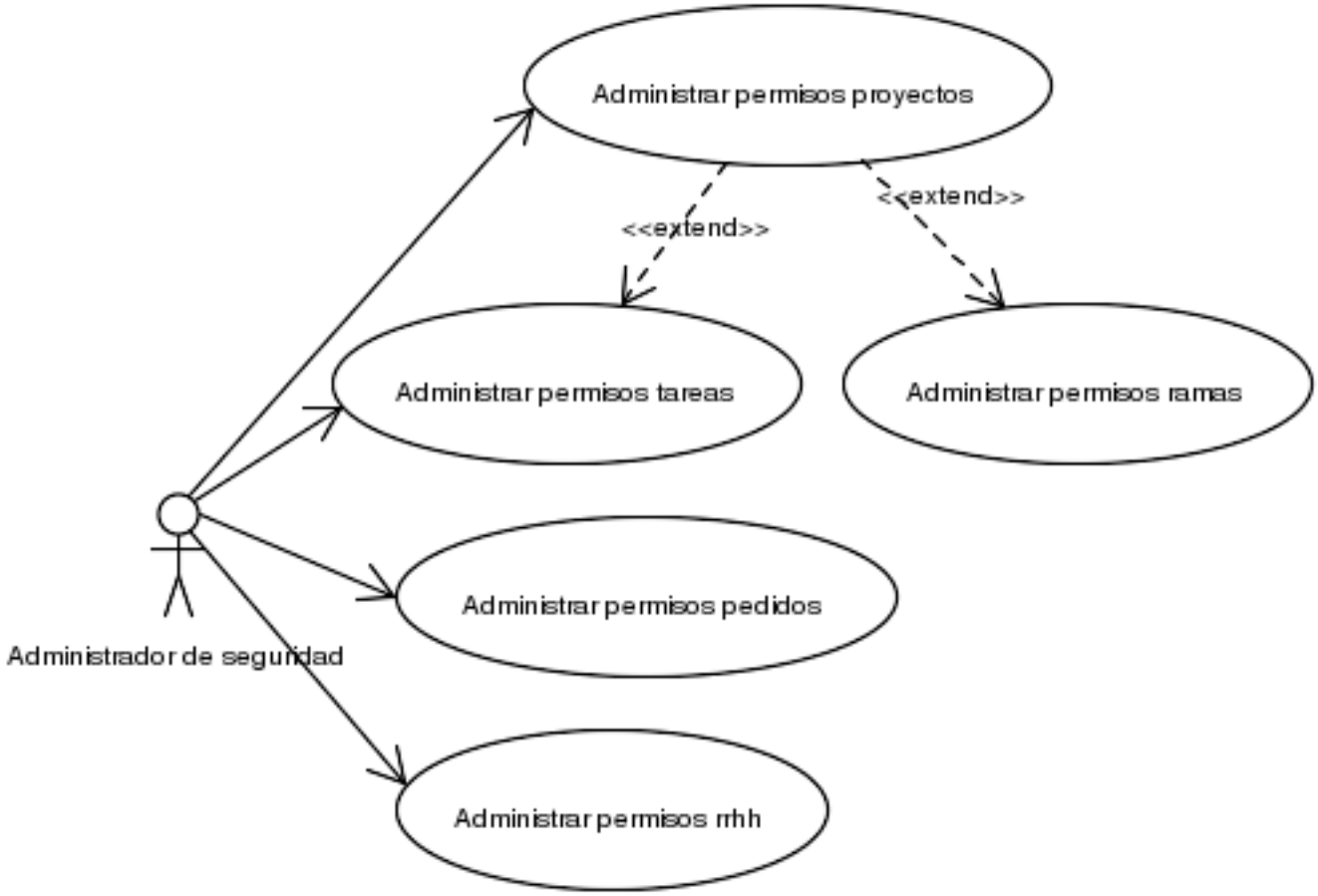

Figura 3.11: Casos de uso de seguridad

3.5. Nota final

Este sistema será utilizado por el Departamento de Desarrollo, y por todas las áreas para las cuales este trabaja mediante interfaces de gestión de pedidos.

Luego de la construcción del prototipo de este sistema como parte del proyecto de grado, este software seguirá evolucionando:

- En la medida de su utilidad.
- De acuerdo a las necesidades que se generen.
- De acuerdo a las necesidades que no haya cubierto la herramienta.
- En base a nuevas funcionalidades que no hubieran sido consideradas y luego de su puesta en práctica se planteasen.

Este sistema será publicado como Software de Fuente Abierta[2] dado el gran aporte que han brindado y brindarán en las etapas siguientes otros productos de esta comunidad, sin los cuales no sería posible la construcción del mismo, además de la posibilidad de crecimiento que esta brinda.

Dado que este producto posee considerables posibilidades de evolución, se considerará de importancia en el diseño tanto la mantenibilidad del mismo como la facilidad de cambio y extensión del sistema.

[2]Software de Fuente Abierta o Código Abierto, del inglés Open Source Software, se refiere al software cuyo código está disponible públicamente, aunque los términos de licenciamiento específicos varían respecto a sus permisos de utilización.

Capítulo 4

Diseño

En este Capítulo se realiza el diseño por medio de diagramas de clase UML[UML] de los requerimientos establecidos en el Capítulo 3 en la página 14.

4.1. Introducción

El diseño del sistema está compuesto por siete módulos, estos son:

- Historia, contiene los objetos y los métodos para registrar la historia de las asociaciones y objetos que correspondan.
- General, contiene las objetos y los métodos que son utilizados por el resto del sistema.
- Tarea, contiene los objetos y los métodos para gestionar tareas.
- Proyecto, contiene los objetos y los métodos para gestionar proyectos.
- Pedido, contiene los objetos y los métodos para gestionar los pedidos de mejora, corrección de errores y nuevas características.
- Recursos Humanos, contiene los objetos y los métodos para gestionar recursos humanos.
- Seguridad, contiene los objetos y los métodos para gestionar la seguridad del sistema.

Los módulos se presentaran en las siguientes secciones según su relacionamiento del resto de los módulos.

4.2. Módulo Historia

Este módulo surgió de la necesidad de conservar todos los datos ingresados al sistema de cierto tipo, es decir, al modificar determinados objetos del sistema, se conservan los datos anteriores, ver Sección 4.3 en la página siguiente. Esto se modeló con tres clases Historico, ItemHistorico y AltaBaja, ver Figura 4.1 en la página siguiente. Historico contiene una lista de referencias a objetos del tipo ItemHistorico, i.e. a todas las instancias persistidas. En otras palabras, una clase que herede de Historico será modificada en forma lógica, físicamente, en lugar de sobrescribir los datos, se dará de baja el ItemHistorico vigente, y se dará de alta un nueva instancia. Estas bajas y altas están

modeladas con el objeto AltaBaja. Dar de alta implica crear un nuevo ItemHistorico con la propiedad vigente en true y la asociación alta. Dar de baja implica establecer la propiedad vigente en false y crear la asociación baja. Las asociaciones de alta y baja se crean con la fecha y justificación correspondientes.

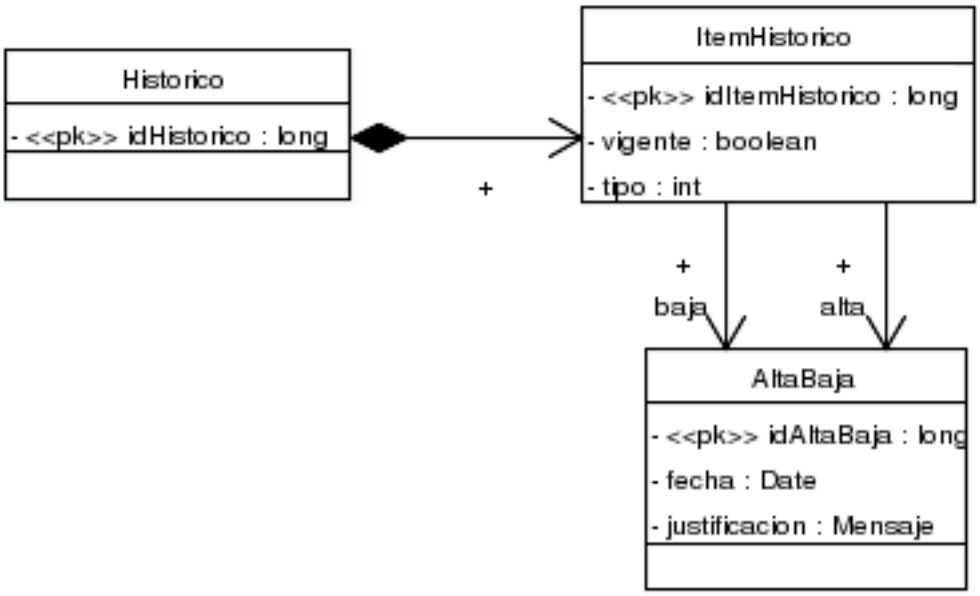

Figura 4.1: Módulo Historia

4.3. Módulo General

Este módulo contiene las clases que modelan las características generales de las tareas, proyectos, ramas, e iteraciones, ver Figura 4.2 en la página siguiente. Todos estos objetos, que se muestran más adelante, ver secciones 4.4 en la página siguiente y 4.5 en la página 34, heredan directa o indirectamente de ItemTrabajo, ya que todos contienen un tiempo, un nombre, una descripción y un usuario responsable. Un ItemTrabajo puede estar agendado, esto está modelado con la clase ItemTrabajoAgendado. Los ítems agendados tiene como atributos las fechas de inicio y fin, planificadas y reales, y los desfases iniciales y finales. Estos últimos son los días para el cual un ítem de trabajo agendado está adelantado o atrasado, siendo negativo para el primer caso, positivos para el segundo y cero en caso de no existir.

La asignación hereda de ItemHistorico, ya que se quiere tener la información histórica de los responsables en el tiempo de un determinado trabajo.

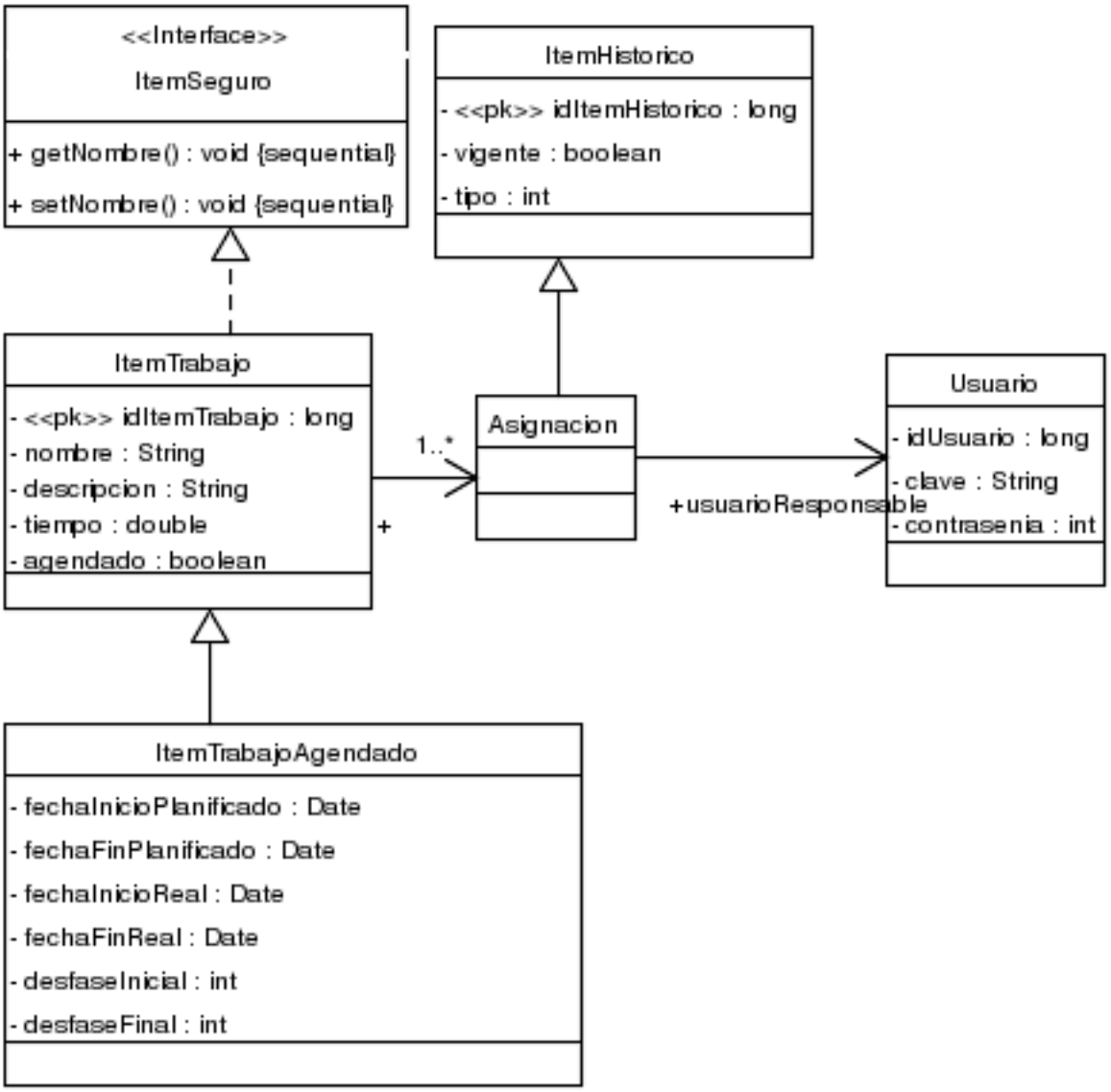

Figura 4.2: Módulo General

4.4. Módulo Tarea

Este módulo contiene todas las clases y métodos para gestionar tareas, ver Figura 4.3 en la página siguiente. Como se vio en 3.2.2.2 en la página 15, existen dos tipos de tareas: las activas y las efectivas, las primeras indican un trabajo a realizar, y las otras la realización del mismo. Toda tarea efectiva está asociada a una actividad que la cataloga.

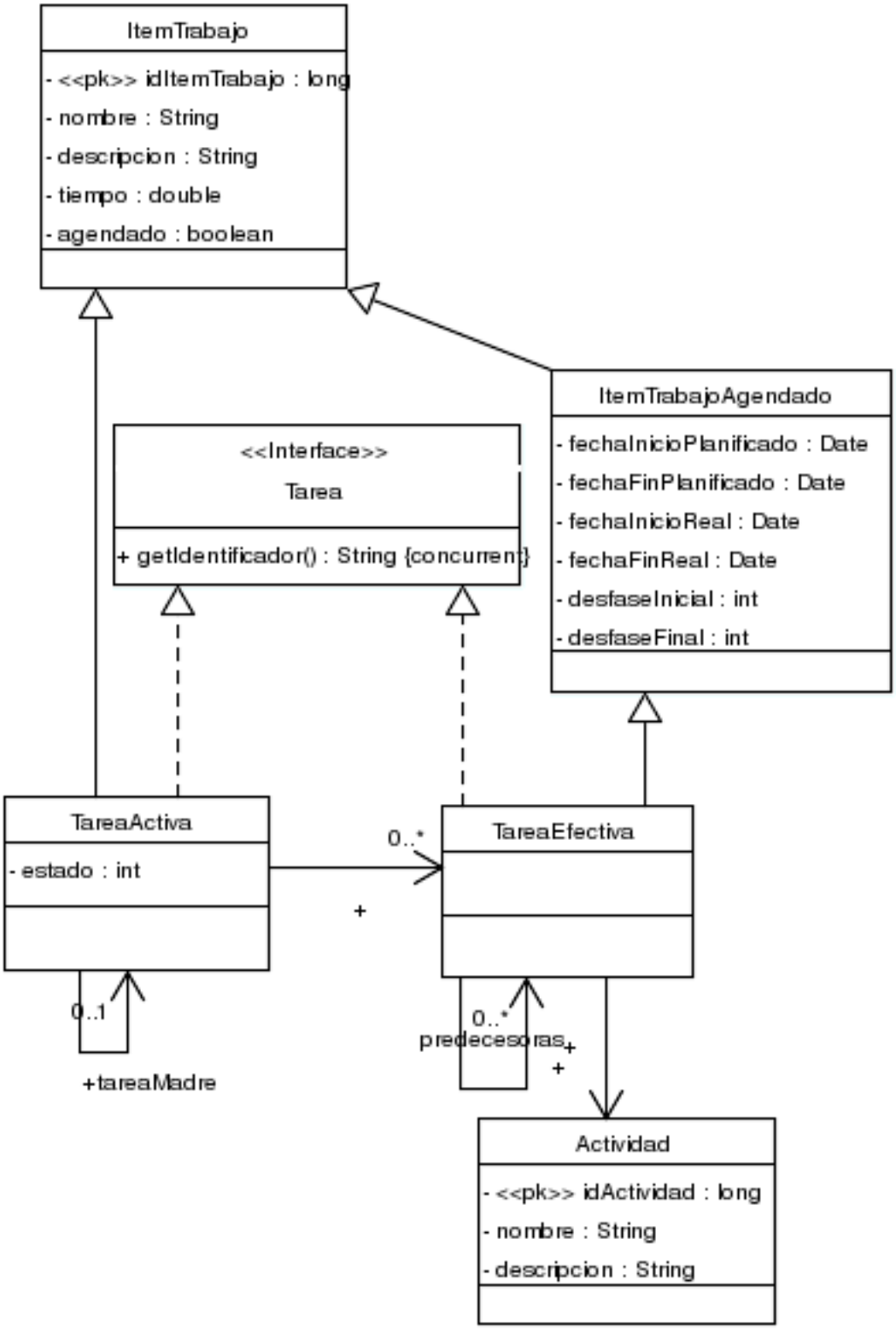

Figura 4.3: Módulo Tarea

4.4.1. Ejemplo

Supongamos una empresa con tres empleados María, Pedro y Juana. La primera coordina las tareas de estos últimos. En la Figura 4.4, se muestra el diagrama de instancias de la siguiente situación:

1. A María le llega como requerimiento realizar un ABM[1] de productos. María crea una tarea activa "Realizar el ABM[2] de productos", y se asigna como responsable.
2. Decide empezar por la interfaz gráfica y que Pedro realice la primer versión. Esto lo establece en el sistema creando una tarea activa "Interfaz gráfica ABM productos", asignándose como responsable, y una tarea efectiva "Interfaz gráfica ABM productos versión 1.0", asignado como responsable a Pedro.
3. Supongamos que al tiempo se descubre un error en dicha interfaz gráfica y decide que lo arregle Juana, entonces crea una tarea efectiva "Corregir error 1 interfaz gráfica ABM productos" con Juana como responsable. La suma de los tiempos de estas dos ultimas tareas será el tiempo de la tarea "Interfaz gráfica ABM productos".

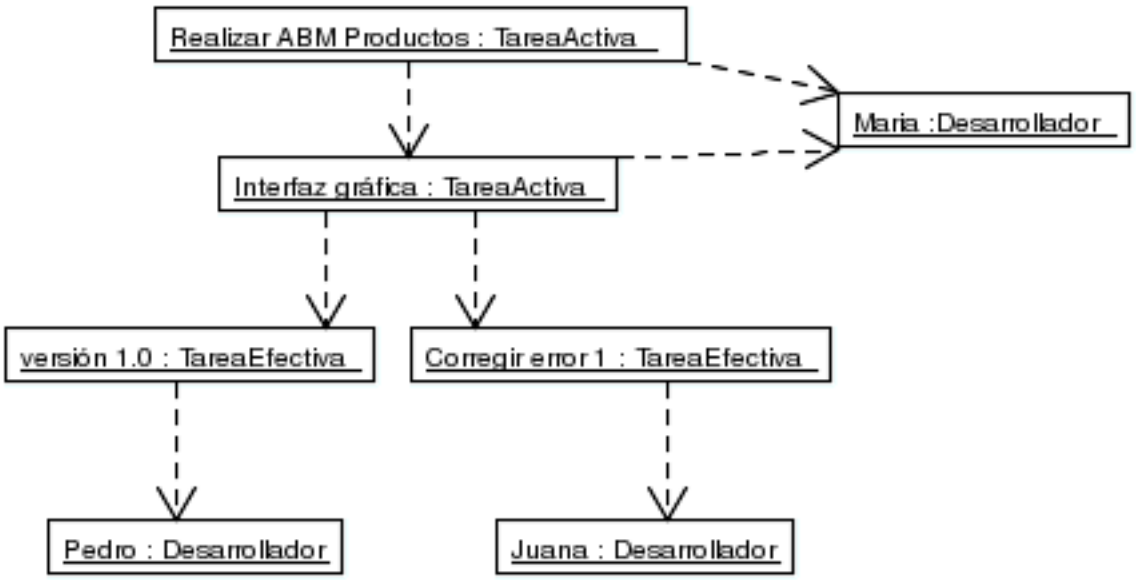

Figura 4.4: Diagrama de dependencias de instancias ejemplo módulo tarea

4.5. Módulo Proyecto

Este módulo tiene todos los objetos y métodos para gestionar proyectos, ver Figura 4.5 en la página siguiente. Este modelo refleja dos partes de la planificación de un proyecto la agendada y la no agendada. La parte de la planificación no agendada, es la que establece la estructura jerárquica de

[1]Alta, baja y modificación.
[2]Alta, baja y modificación.

un proyecto con los trabajos a realizar sin establecer cuando se van a realizar, esto está modelado con las clases Proyecto, Rama y TareaActiva. Por otro lado, las clases Iteracion y TareaEfectiva, modelan los períodos de tiempo en los que se deberán realizar las tareas y conjuntos de estas.

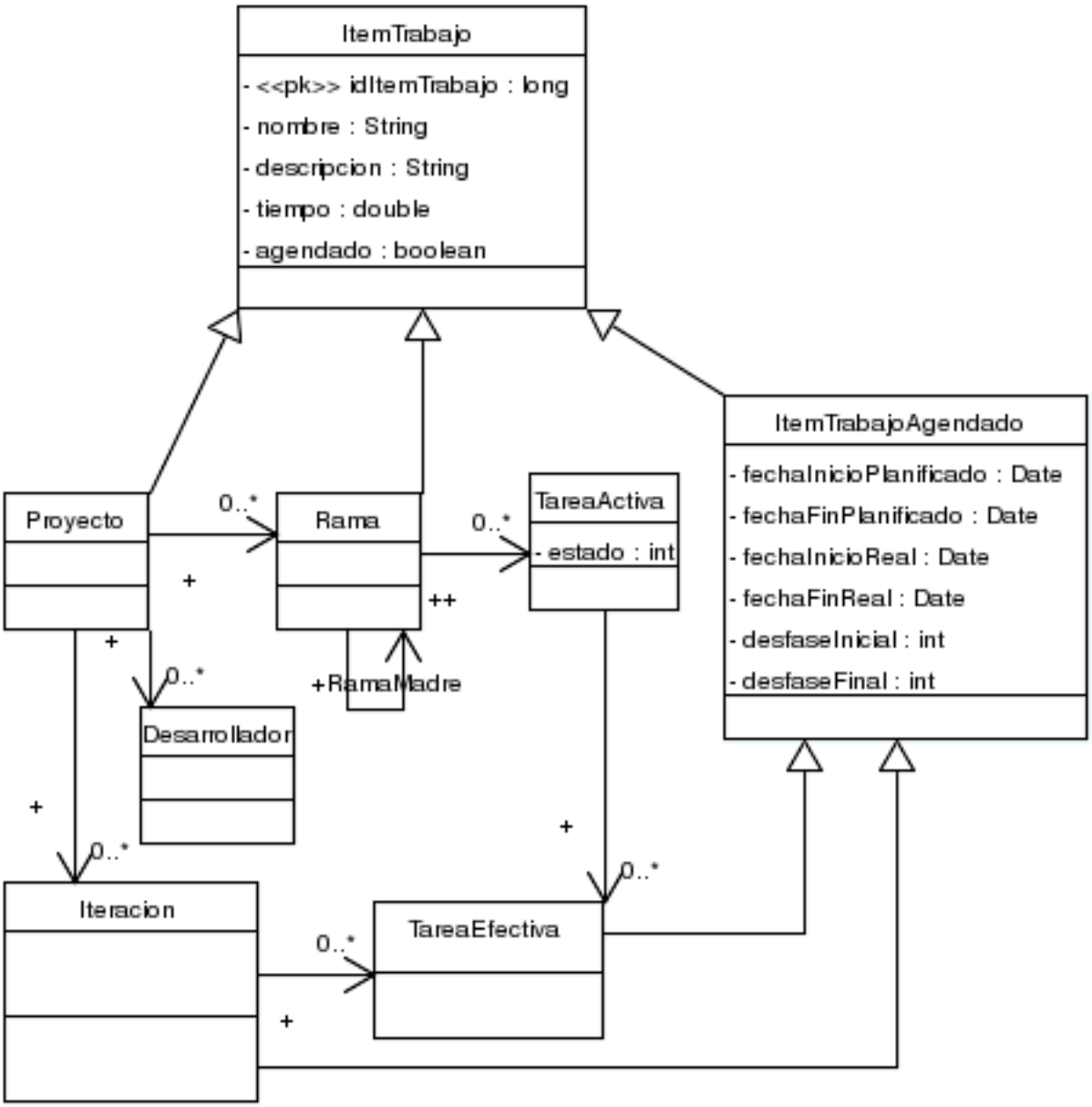

Figura 4.5: Módulo Proyecto

4.6. Módulo Pedido

Este módulo contiene todos los objetos y métodos para gestionar pedidos, ver Figura 4.6. Se pueden realizar pedidos de corrección de errores, nuevas funcionalidades, modificaciones, o extensiones de funcionalidades ya implementadas. La clase mensaje modela todos los mensajes intercambiados por causa de un pedido.

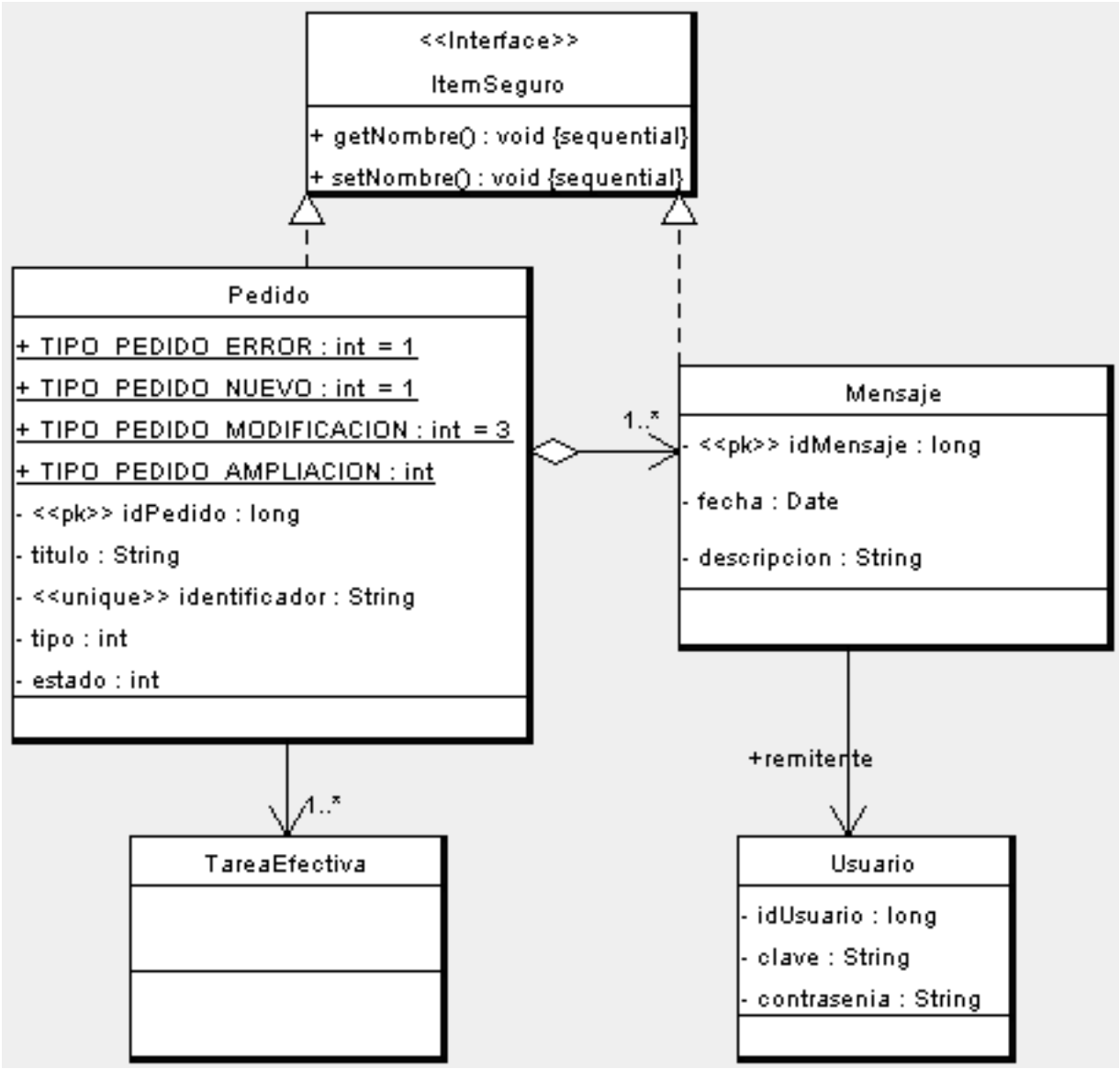

Figura 4.6: Módulo Pedido

4.7. Módulo Recursos Humanos

Este módulo contiene todos los objetos y métodos para la gestión de recursos humanos, ver Figura 4.7. Una empresa contiene secciones que pueden subdividirse. Se mantienen datos de personas que pueden pertenecer a una sección de una empresa. Un desarrollador es una persona que puede tener varios contratos asociados, y varios costos. El sistema deberá chequear que exista un solo contrato y costo vigente a la vez.

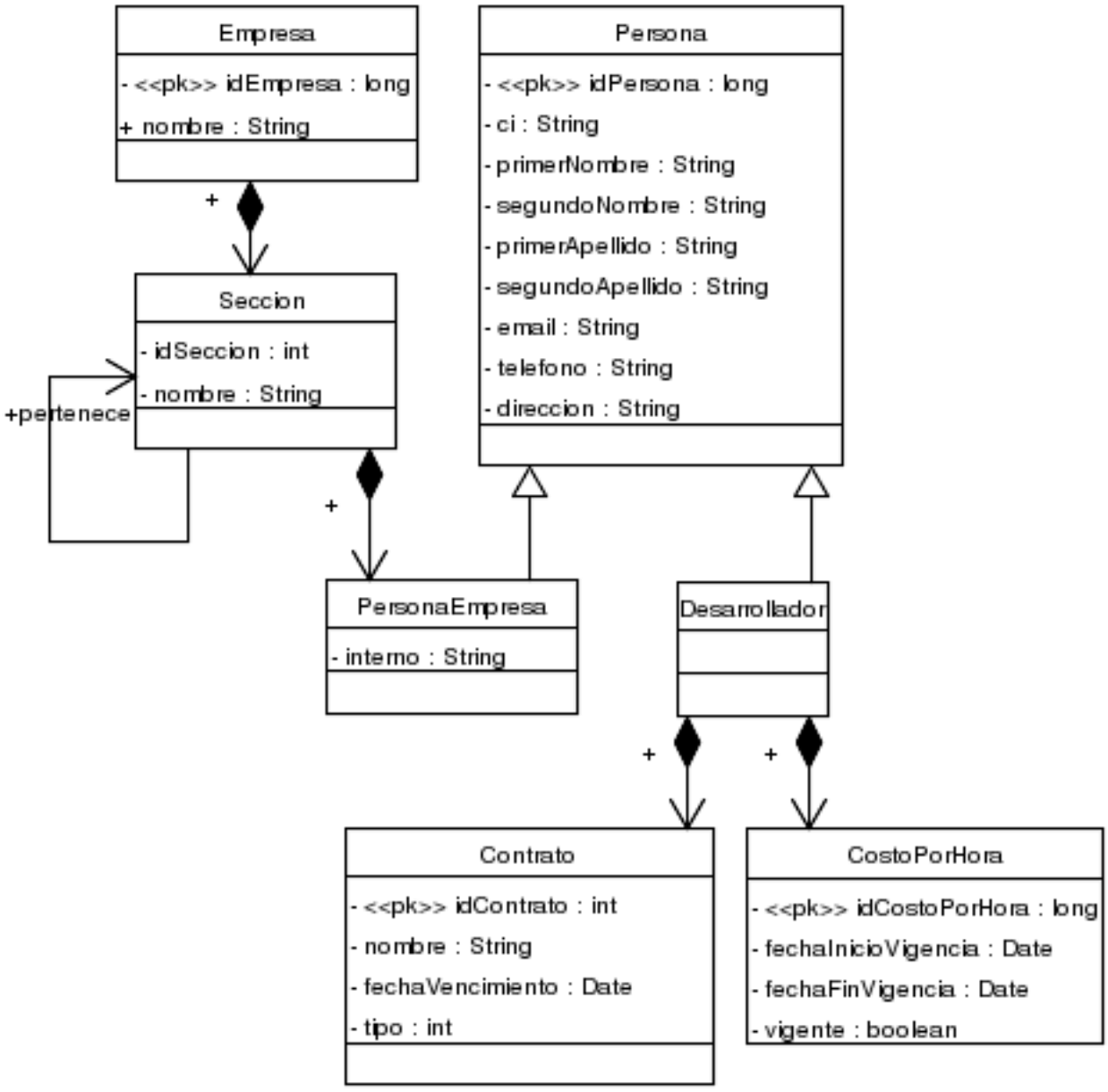

Figura 4.7: Módulo Recursos Humanos

4.8. Seguridad del sistema

Este módulo contiene todos los objetos y métodos para la administración de la seguridad del sistema.

4.8.1. Usuarios

Desde el punto de vista de la seguridad y para contribuir con la expansibilidad del modelo, se diseñaron dos clases de usuarios, ver Figura 4.8 en la página siguiente:

- Usuario Externo: Este usuario accederá al sistema a través de la interfaz de pedidos únicamente.
- Usuario Interno: Este usuario podrá acceder a todo el sistema según los permisos particulares.

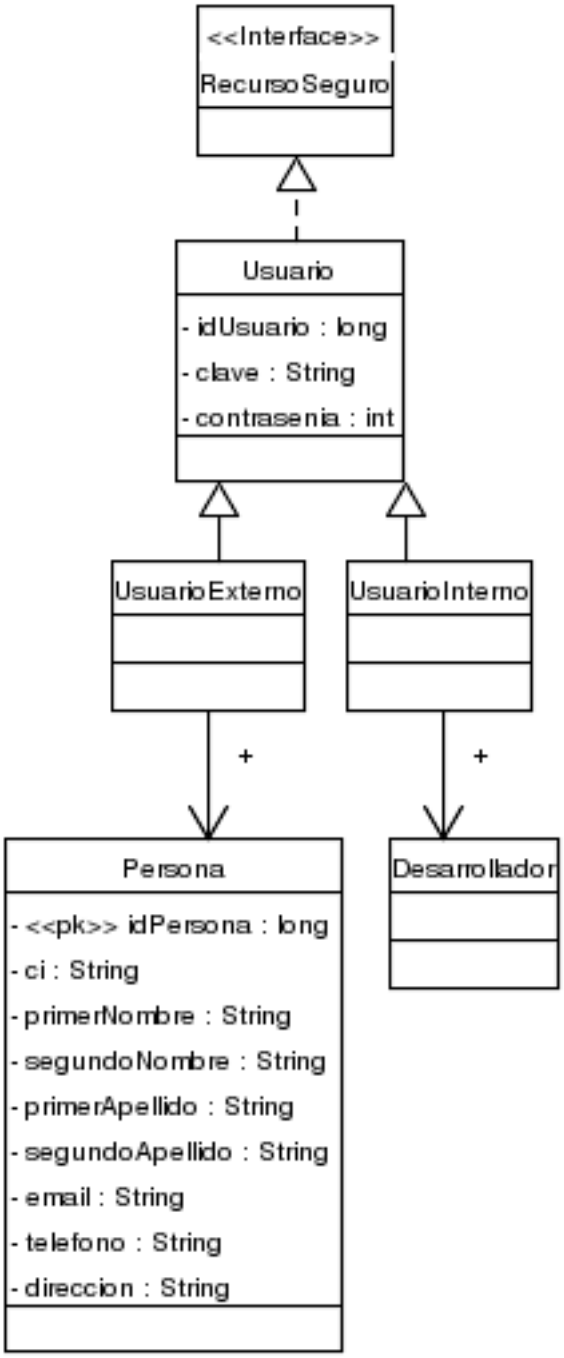

Figura 4.8: Módulo Seguridad: Diagrama de usuarios

4.8.2. Permisos

El sistema tendrá dos grandes tipos de permisos:

- Permisos de clases, tienen como única funcionalidad indicar la autorización de crear instancias de una clase, ver Figura 4.9.
- Permisos de instancias, se refieren a las instancias de clases, por ejemplo el proyecto "A", la tarea "1", etc., ver Figura 4.10 en la página siguiente. Se crearon dos interfaces para clasificar a los objetos del sistema con permisos autónomos: ItemSeguro, de los que no lo son: RecursoSeguro. En la figuras 4.6 en la página 36 y 4.2 en la página 32, se muestran los diagramas con las clases que implementan ItemSeguro, y en la Figura 4.8 en la página anterior, la clase que implementa RecursoSeguro. La clase PermisoInstanciaItemSeguro contiene una lista de objetos de la clase PermisoInstanciaRecursoSeguro, esta es la que permitirá obtener los recursos asignados a un ItemSeguro. Por ejemplo, los usuarios que pueden ser asignados a una tarea por un usuario en particular.

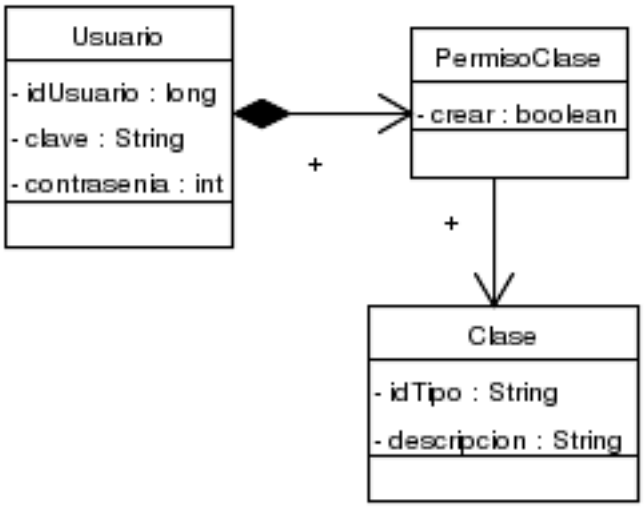

Figura 4.9: Permiso de clases

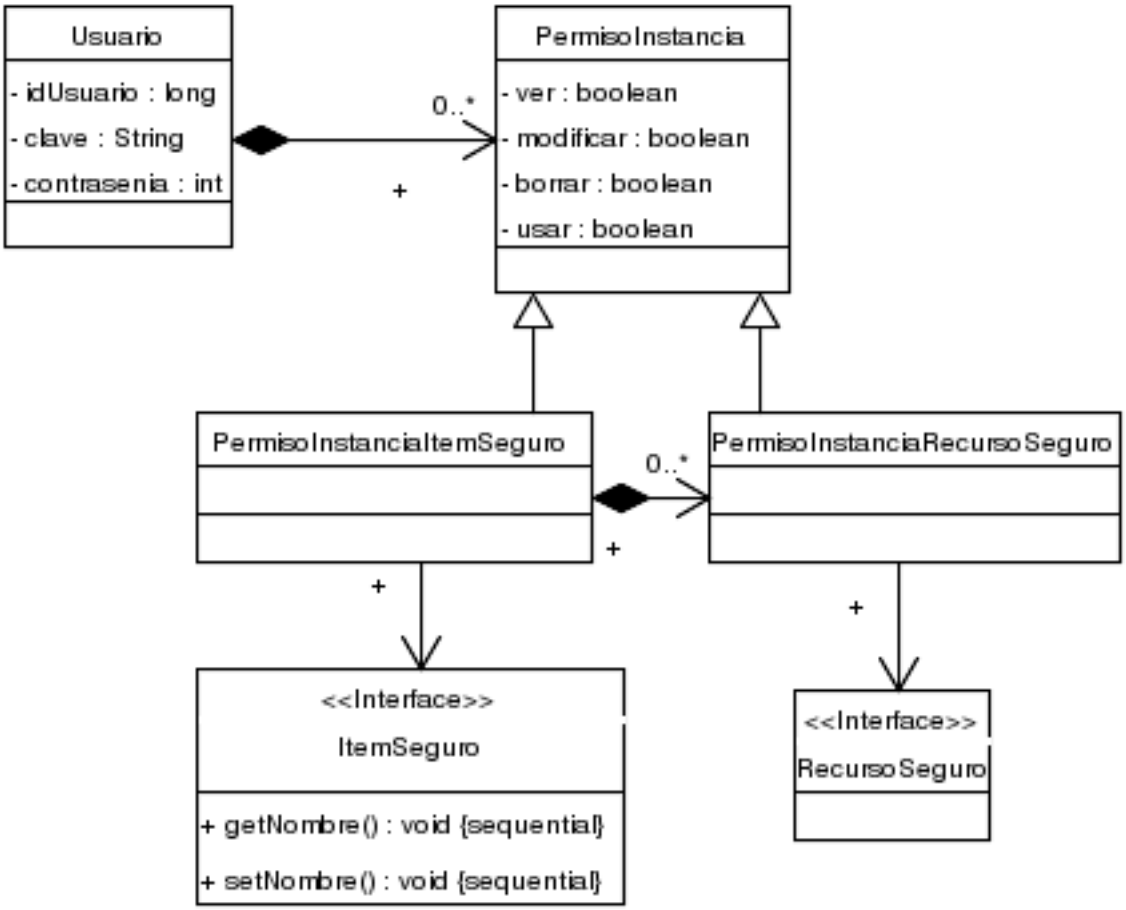

Figura 4.10: Permiso de instancia

Capítulo 5

Implementación

Este Capítulo describe los aspectos más relevantes de las decisiones de implementación y establece un plan para la misma.

5.1. Tecnología

En primer lugar el sistema se decidió realizar como un sitio web. Esto permitirá un fácil acceso ya que lo único necesario para conectarse con el sistema es un navegador[1]. Las actualizaciones se realizan en el servidor con lo que serán efectivas para el usuario de forma inmediata.

La tecnología principal utilizada en el Departamento como se cito anteriormente es J2EE [J2EE], y las aplicaciones web se realizan con el framework Struts [Struts] lo cual lo hace muy fácil de instalar y no requiere la instalación de otro servidor Web. Además este permite interactuar con Hibernate [Hibernate] el cual resuelve el acceso a la base de datos de forma clara y eficiente. J2EE [J2EE], Struts [Struts], e Hibernate [Hibernate] permiten construir un producto de fácil mantenimiento, independiente del sistema operativo, y robusto los cuales favorecen a la buena calidad del producto.

5.2. Plan de implementación

El método para desarrollar Arena es incremental iterativo. El plan de implementación se describe en el Cuadro 5.2 en la página siguiente. En este se especifica la iteración, el módulo a implementar y las figuras donde están los diagramas que muestran las clases que incluyen dichos módulos. La implementación de un módulo incluye la creación de las tablas de la base de datos, y las interfaces gráficas correspondientes.

[1] En inglés Browser.

Iteración	
Módulo	**Diagramas de Clase**
Iteración 1	
Seguridad, ver Sección 4.8 en la página 38.	Figuras: 4.8 en la página 39, 4.9 en la página 40, y 4.10 en la página 41.
Iteración 2	
Historia, ver Sección 4.2 en la página 30.	Figura 4.1 en la página 31.
General, ver Sección 4.3 en la página 31.	General, ver Sección 4.3 en la página 31.
Tarea, ver Sección 4.4 en la página 32 .	Figura 4.3 en la página 33.
Iteración 3	
Pedido, ver Sección 4.6 en la página 36.	Figura 4.6 en la página 36.
Iteración 4	
Proyecto, ver Sección 4.5 en la página 34.	Figura 4.5 en la página 35.
Iteración 5	
Recursos humanos, ver Sección 4.7 en la página 37	Figura 4.7 en la página 37.

Cuadro 5.2: Plan de implementación

Capítulo 6

Testeo y resultados

Para dar por finalizadas las iteraciones del Capítulo 5 en la página 42, se deberán realizar testeos que garanticen su correcto funcionamiento. Los testeos serán realizados al final de cada iteración con el enfoque de caja de negra. Estos se basan en los casos de usos descriptos en la Sección 3.4 en la página 18. Al final de cada testeo se corregirán los errores encontrados y se volverían a realizar los testeos hasta no encontrarse ningún error. Los testeos podrán ser automáticos o manuales. Los casos de testeo de las siguientes iteraciones se planificarán al inicio de las mismas.

6.1. Testeos Iteración 1

6.1.1. Caso testeo: Ingreso al sistema

Se verificará que sólo se pueda ingresar al sistema con el usuario y contraseña correctas.

Entradas:

1. Usuario y contraseña correctas.
2. Usuario correcto, contraseña incorrecta.
3. Usuario incorrecto, contraseña correcta.
4. Usuario y contraseña incorrectas.
5. Usuario vacío y contraseña correcta
6. Usuario vacío y contraseña incorrecta
7. Usuario correcto y contraseña vacía.
8. Usuario incorrecto y contraseña vacío.
9. Usuario y contraseña incorrecta.

Resultados:

En el diagrama de actividades UML [UML] de la Figura 6.1 en la página siguiente indica las validaciones del testeo y los estados esperados. El estado mensaje información indica que deberá mostrarse un mensaje al usuario indicado que debe ingresar los datos. El estado “Error” indica que se desplegará un mensaje de error indicando cual dato es erróneo. Finalmente el estado "Ingreso válido” indica que se deberá ingresar al sitio.

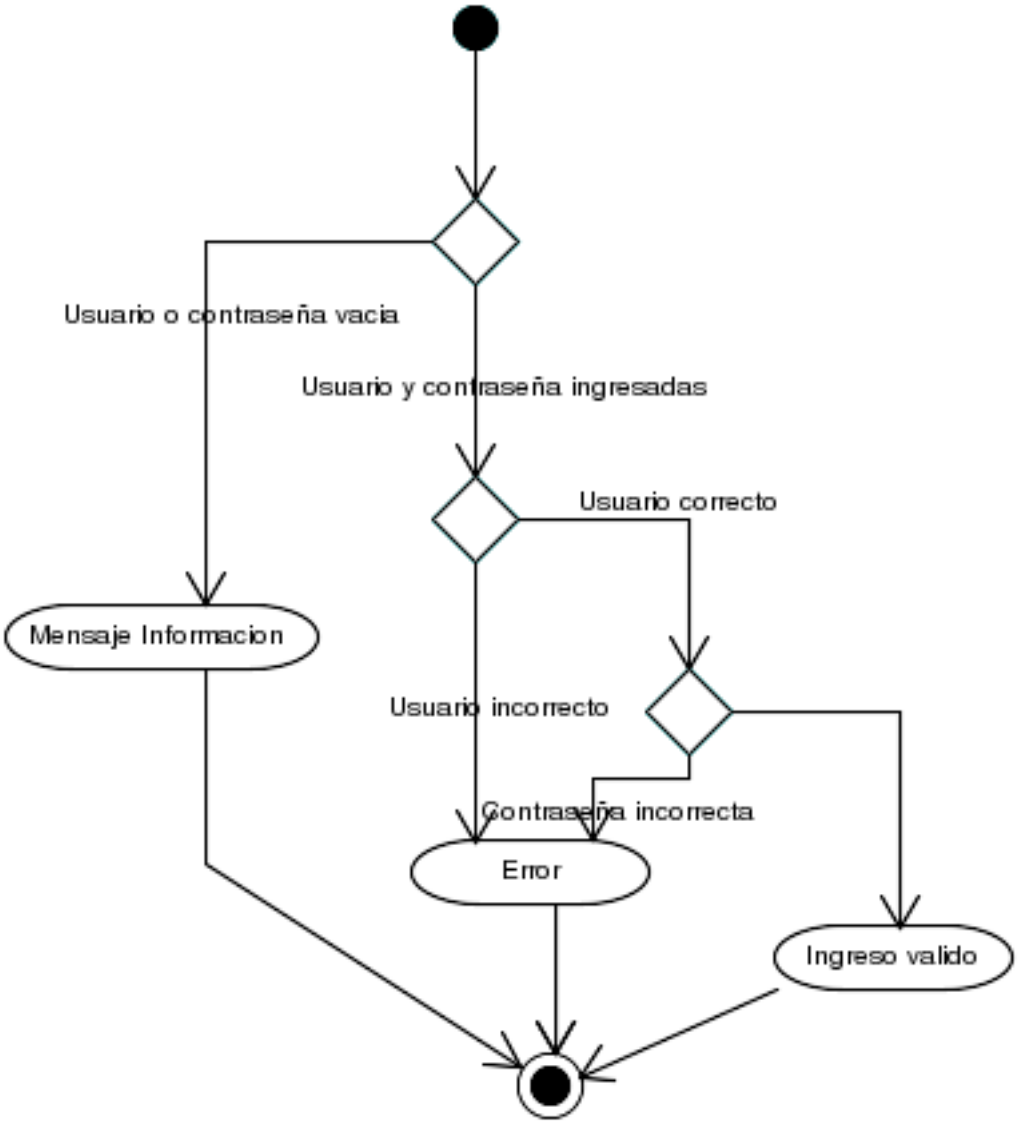

Figura 6.1: Testeo seguridad

6.2. Testeo Iteración 2

6.2.1. Caso Testeo: Creación de una tarea efectiva ya realizada

Procedimiento

1. Ingresar al panel de ingreso de tarea efectiva.
2. Intentar ingresar el objeto:
   a) Si se muestra un mensaje indicando ingresar el nombre, es correcto seguir con el siguiente punto.
   b) sino es incorrecto.
3. Ingresar el nombre.
4. Intentar ingresar el objeto:
   a) Si se muestra un mensaje indicando ingresar la descripción, es correcto, seguir con el siguiente punto.
   b) sino es incorrecto.
5. Ingresar una descripción.
6. Ingresar los valores de las fecha de inicio y fin planificadas anteriores a la fecha actual.
7. Intentar ingresar el objeto.
   a) Si se muestra un mensaje de indicando ingresar las fechas de inicio y fin reales, es correcto, seguir con el paso siguiente.
   b) sino es incorrecto.
8. Intentar ingresar el objeto.
   a) Si se muestra un mensaje indicando ingresar la actividad de la tarea, es correcto, seguir con el siguiente paso.
   b) sino es incorrecto.
9. Asignar una actividad.
10. Intentar ingresar el objeto.
    a) Si se muestra un mensaje indicando ingresar el usuario responsable, es correcto, seguir con el siguiente paso.
    b) sino es incorrecto.
11. Asignar un usuario responsable.
12. Ingresar el objeto
13. Verificar
    a) Los datos ingresados son los grabados.
    b) El calculo de desfases es correcto.
    c) Existe un itemHistorico.

Condiciones

Deberán existir actividades ingresadas en la base de datos.

6.2.2. Caso Testeo: Creación de una tarea efectiva no realizada aún

Procedimiento

1. Ingresar al panel de ingreso de tarea efectiva.
2. Intentar ingresar el objeto:
   a) Si se muestra un mensaje indicando ingresar el nombre, es correcto seguir con el siguiente punto.
   b) sino es incorrecto.
3. Ingresar el nombre.
4. Intentar ingresar el objeto:
   a) Si se muestra un mensaje indicando ingresar la descripción, es correcto, seguir con el siguiente punto.
   b) sino es incorrecto.
5. Ingresar una descripción.
6. Ingresar los valores de las fecha de inicio y fin planificadas, la fecha de fin posterior a la actual.
7. Intentar ingresar el objeto.
   a) Si se muestra un mensaje indicando ingresar la actividad de la tarea, es correcto, seguir con el siguiente paso.
   b) sino es incorrecto.
8. Asignar una actividad.
9. Intentar ingresar el objeto.
   a) Si se muestra un mensaje indicando ingresar el usuario responsable, es correcto, seguir con el siguiente paso.
   b) sino es incorrecto.
10. Asignar un usuario responsable.
11. Ingresar el objeto
12. Verificar que los datos ingresados son los grabados.

Condiciones

Deberán existir actividades ingresadas en la base de datos.

6.2.3. Caso Testeo: Consulta de una tarea efectiva

Procedimiento

1. Ingresar tareas efectivas.
2. Ingresar al panel de consulta y verificar que muestre las tareas ingresadas.

6.2.4. Caso Testeo: Histórico tarea efectiva

Procedimiento

1. Crear una tarea efectiva.
2. Ingresar al panel de modificación de la tarea efectiva.
3. Cambiar el usuario responsable.
4. Ingresar al panel de consulta de tareas efectivas.
5. Buscar la tarea modificada
6. Verificar la creación del histórico del usuario responsable.

6.2.5. Caso Testeo: Modificación de una tarea efectiva

Procedimiento

1. Ingresar al panel de modificación de tarea efectiva.
2. Modificar datos.
3. Chequear que las modificaciones se reflejen en la base de datos.

6.2.6. Caso de Testeo: Eliminación de una tarea efectiva

Procedimiento

1. Eliminar una tarea efectiva.
2. Verificar que la tarea no exista en la base de datos.

6.2.7. Caso Testeo: Ingreso de una tarea activa

Procedimiento

1. Ingresar al panel de creación de tarea activa.
2. Intentar ingresar el objeto:
   a) Si se muestra un mensaje indicando ingresar el nombre, es correcto seguir con el siguiente punto.
   b) sino es incorrecto.
3. Ingresar el nombre.
4. Intentar ingresar el objeto:
   a) Si se muestra un mensaje indicando ingresar la descripción, es correcto, seguir con el siguiente punto.
   b) sino es incorrecto.

5. Ingresar una descripción.
6. Intentar ingresar el objeto.
   a) Si se muestra un mensaje indicando ingresar el usuario responsable, es correcto, seguir con el siguiente paso.
   b) sino es incorrecto.
7. Asignar un usuario responsable.
8. Verificar que los datos ingresados son los grabados.

6.2.8. Caso de Testeo: Eliminación de una tarea activa

Procedimiento

1. Eliminar una tarea activa.
2. Verificar que la tarea eliminada no exista en la base de datos.

6.2.9. Caso Testeo: Modificación de una tarea activa

Procedimiento

1. Ingresar al panel de modificación de tarea activa.
2. Modificar datos.
3. Chequear que las modificaciones se reflejen en la base de datos.

6.2.10. Caso Testeo: Histórico de una tarea efectiva

Procedimiento

1. Crear una tarea efectiva.
2. Ingresar al panel de modificación de la tarea efectiva.
3. Cambiar el usuario responsable.
4. Ingresar al panel de consulta de tareas efectivas.
5. Buscar la tarea modificada
6. Verificar la creación del histórico del usuario responsable.

6.3. Testeo Iteración 3

6.3.1. Caso Testeo: Creación de un pedido por un usuario interno

1. Ingresar al panel de creación de pedidos con un usuario interno (Se deberá haber ingresado a la aplicación con dicho usuario).
2. Intentar dar de alta el pedido
   a) Si muestra un mensaje indicando ingresar el título, es correcto, seguir en el paso siguiente.
   b) sino es incorrecto.
3. Intentar dar de alta el pedido
   a) Si muestra un mensaje indicando ingresar el tipo, es correcto, seguir en el paso siguiente.
   b) sino es incorrecto.
4. Ingresar el tipo.
5. Intentar dar de alta el pedido
   a) Si muestra un mensaje indicando ingresar la descripción, es correcto, seguir en el paso siguiente.
   b) sino es incorrecto.
6. Ingresar la descripción.
7. Ingresar el pedido.
8. Verificar que los datos ingresados sean los grabados, y la fecha, hora y usuario los correctos.

6.3.2. Caso Testeo: Creación de un pedido por un usuario externo

Este caso es igual al anterior a excepción de que hay que controlar que no este disponible el ingreso del identificador.

6.3.3. Caso Testeo: Modificación de un pedido completo

1. Ingresar al panel de edición de pedidos con un usuario interno.
2. Modificar el título.
3. Modificar el tipo.
4. Modificar el identificador.
5. Modificar el estado.
6. Intentar grabar el pedido.
   a) Si muestra un mensaje indicando ingresar una nueva descripción, es correcto, seguir en el paso siguiente.
   b) sino es incorrecto.

7. Ingresar una descripción.
8. Grabar el pedido.
9. Verificar que los datos fueron cambiados efectivamente, y que existe un mensaje nuevo asociado al pedido con el usuario y fecha correctos.

6.3.4. Caso Testeo: Búsqueda de un pedido

1. Ingresar al panel de búsqueda
2. Ingresar los datos para la búsqueda del pedido.
3. Verificar que los pedidos buscados sean los mostrados.

6.3.5. Caso Testeo: Ingreso de un mensaje

1. Ingresar al panel de búsqueda de mensajes.
2. Seleccionar agregar pedidos.
3. Ingresar una descripción.
4. Ingresar nuevamente al panel de búsqueda de mensajes y verificar que el mensaje este asociado correctamente al pedido.

Capítulo 7

Puesta en producción

7.1. Introducción

Este Capítulo describe la planificación para la puesta en producción según las iteraciones citadas en el Capítulo 5.

El producto tendrá las siguientes versiones al final de cada iteración a partir de la iteración 2 que se muestran en la tabla 7.1.

Versión	Hito	Nuevas funcionalidades
Arena 1.0	Final de la iteración 2	Control de ingreso al sitio, gestión de tareas activas y efectivas.
Arena 1.1	Final de la iteración 3	Gestión de pedidos.
Arena 1.2	Final de la iteración 4	Gestión de proyectos, ramas e iteraciones
Arena 1.3	Final de la iteración 5	Gestión de recursos humanos.

Cuadro 7.1: Primeras versiones del producto

7.2. Plan de implantación

Arena 1.0

Se seleccionará un Grupo A de personas del Departamento. Se les asignará usuarios con permisos para gestionar tareas. Serán seleccionadas según el criterio del Jefe del Departamento.

Arena 1.1

Para esta versión se seleccionará el Grupo B. Estará compuesto por personas del Grupo A encargadas de gestionar pedidos. Se les asignarán permisos para gestionar pedidos. Una vez considerada estable para las personas del Grupo B. Se extenderán los permisos del resto del Grupo A para gestionar pedidos, y luego para el resto del Departamento. Finalmente se irán creando usuarios para personas de otros Departamentos de la Empresa para generar pedidos externos.

Versiones posteriores a Arena 1.1

A partir de esta versión los permisos para la utilización de nuevas funcionalidades serán otorgados a todos los miembros del Departamento según lo considere el Jefe del Departamento. Esto será después de una evaluación apropiada de las nuevas funcionalidades.

Capítulo 8

Métricas

> “No se puede controlar lo que que no se puede medir”
>
> -De Marco, 1982

El primer paso para la mejora es organizarse y evaluar el estado actual, adquirir control sobre lo que se hace y se deja de hacer para luego poder tomar decisiones de mejora. La herramienta a desarrollar provee las funcionalidades para organizar las tareas al mismo tiempo que proporcionará datos para calcular los indicadores que se mencionarán en las siguientes secciones. En este capítulo se hará una descripción de los indicadores a utilizar, ver Apéndice B en la página 75, para cumplir con los tres objetivos principales:

1. Controlar los costos
2. Controlar los requerimientos
3. Controlar los errores

8.1. Tiempos de construcción y costos

Se quiere medir los tiempos de dedicación por tarea y actividad. Las actividades que identificamos en la construcción de un producto son concepción, diseño, implementación, testeo, puesta en producción, soporte y mantenimiento.

Tanto los tiempos agendados como reales junto con los desfases finales e iniciales se pueden obtener o inferir de las tareas, proyectos, ramas, y/o iteraciones. Para calcular los costos se utilizará el indicador citado en el punto B.2.2 en la página 77.

8.2. Requerimientos

El segundo de los objetivos principales es analizar el flujo de los requerimientos. Se quiere tener medidas sobre la cantidad de requerimientos en los diferentes estados a través del tiempo. Se quiere tener respuesta a preguntas como: ¿Cuántos requerimientos están pendientes? ¿Cuántos requerimientos se cumplieron en el último año? ¿Cuántas modificaciones se realizaron al requerimiento

inicial? ¿La realización del requerimiento cumplió con las necesidades del usuario? ¿Cuántos requerimientos se agregaron a la planificación inicial? Las especificaciones: ¿son correctas? ¿están completas?

Los requerimientos en la herramienta a desarrollar se modelaron como pedidos, ver Sección 4.6 en la página 36, estos tienen un atributo de nombre tipo que tendrá los siguientes valores para realizar su seguimiento:

1. Ingresado, el requerimiento está pendiente de evaluación
2. Aceptado, el requerimiento fue evaluado y se va a realizar.
3. Rechazado, el requerimiento fue evaluado y no se va a realizar.
4. En construcción, se están realizando las tareas necesarias para construir las funcionalidades establecidas en el requerimiento.
5. Testeo, se está realizando un testeo integral de todas las funcionalidades que componen el requerimiento
6. Para deploy, la construcción de las funcionalidades del requerimiento han sido realizadas, pero están pendientes del deploy
7. En producción inestable, el usuario ya puede utilizar las funcionalidades requeridas pero se detectan errores frecuentemente.
8. En producción estable, el usuario ha utilizado las funcionalidades requeridas, estas funcionan correctamente, y los errores son poco frecuentes.

Se calcularán los costos de la producción de un requerimiento con el indicador citado en el punto B.2.2 en la página 77. Los tiempos dedicados al requerimiento se obtendrán de las tareas asociadas al mismo y los costos por períodos de los datos del módulo de recursos humanos.

Para determinar si un requerimiento es estable, se fijará un valor aceptable de densidad de errores y se calculará su correspondiente densidad con la fórmula citada en el punto B.2.3.2.1 en la página 78.

8.3. Errores

Medir la cantidad de errores y el tiempo que lleva corregirlos así como su clasificación, ver Apéndice en la página 75, se consideran primordiales a la hora de medir la calidad del producto y de un Analista o Ingeniero.

Se quiere tener respuestas a preguntas como: ¿Cuántos errores se detectaron en la fase de testeo? ¿Cuánto errores se detectaron después de la puesta en producción? ¿Cuál es la relación tiempo de construcción sobre tiempo de corrección de errores? ¿Cuántos productos son estables?

Para contestar estas preguntas se utilizarán los indicadores de densidad de errores, ver puntos B.2.3.2.1, B.2.3.2.2, y B.2.3.3.2 en la página 78.

8.4. Nota final

El cálculo de estos indicadores brindarán pautas no sólo de mejora sino de otros factores a medir. Los cambios, como ya se mencionó, para que sean estables deben ser progresivos, sobre todo cuando pueden afectar la confianza de quienes son evaluados. El seguimiento detallado de los requerimientos es el objetivo más importante al día de hoy para el Departamento.

La métricas seleccionadas con excepción de la densidad de errores, ver punto B.2.3.2.1 en la página 78, se consideran totalmente confiables. La métrica de densidad de errores, citada en el punto B.2.3.2.1 en la página 78 no es sólo muy compleja de medir, sino también subjetiva ya que debería existir una persona que pondere cada línea de código o reglas para su ponderación, con lo que hace de esta poco recomendable.

Capítulo 9

Conclusiones

La mejora continua requiere de un estudio minucioso de los objetivos y de los recursos que se cuenta para realizarla. La tarea implicará la aceptación del equipo para su cumplimiento. Las etapas más interesantes de este no pudieron ser alcanzadas y los resultados de mejora real no podrán medirse sino en el largo plazo.

Los indicadores que se planifican calcular brindarán control sobre el departamento y llevarán a conclusiones que sin estos no se pueden aseverar.

La herramienta a desarrollar junto con el análisis y la evaluación que se ha realizado y permitirá realizar serán un aporte a la mejora de la calidad del Departamento y se espera que también a sus metodologías.

La parte sin duda más interesante de este proyecto será su aplicación, los cambios para organizarse mejor, seguir creciendo, y poder mostrar información más completa hacia el resto de la empresa. Igualmente se lograron los objetivos de diseño de una herramienta que se ajustará a las necesidades particulares de la gestión de la empresa, así como el estudio de indicadores sobre factores que se quiere controlar y del estado del arte de las metodologías y herramientas de gestión de software.

Capítulo 10

Trabajo Futuro

En el futuro se implantará el sistema en todo el Departamento y se estudiarán los datos para sacar métricas que permitan evaluar los diferentes aspectos a mejorar. La herramienta se adaptará en caso de ser necesario a los nuevos datos que necesite.

Se construirá un motor de reportes que permita de forma automática generar los reportes que son requeridos por la gerencia. Se agregarán las funcionalidades de calendario pudiendo agendar reuniones y convocatorias a todos los participantes con pedido de confirmación. Tendrá un responsable de construir un acta la cual se ingresará por medio del sistema.

Se adaptará la herramienta para que colabore con la gestión de cualquier Departamento de la Empresa permitiendo así una globalización de la gestiones.

Se publicará la herramienta como de Fuente Abierta[1] favoreciendo su evolución, y el aporte de diferentes puntos de vista.

[1]Software de Fuente Abierta o Código Abierto, del inglés Open Source Software, se refiere al software cuyo código está disponible públicamente, aunque los términos de licenciamiento específicos varían respecto a sus permisos de utilización.

Bibliografía

[C/C++] Deitel, H., Deitel P. (1995). Como programar en C/C++ . Segunda edición. Prentice Hall. ISBN 9-6888-0471-1.

[CMM Software] Chrissis, M. B., Curtis, B., Paulk, M. C., Weber, C. V. (1993). Capability Maturity Model^{SM}for Software, Version 1.1. Software Engineering Institute.

[COBOL] Cobol en español. http://www.escobol.com/ Última fecha de visita 02/10/2004.

[Hibernate] Hibernate, relational persistence for idiomatic Java. http://www.hibernate.com Última fecha de visita 27/09/2004.

[IEEE STD 830-1998] IEEE Recommended Practice for Software Requierements Specificactions. IEEE Std 830-1998 (Revision of IEEE Std 830-1993). Software Engineering Standards Committee of IEEE Computer Society. Aprobado el 25/06/1998. ISBN 0-7381-0332-2. http://kybele.escet.urjc.es/Documentos/ISI/IEEE-STD-830-1998.pdf Última fecha de visita 13/11/2004.

[ISPE] ISPE International Society for Pharmaceutical Engineering Glossary: Resources and glossaries. Michele Gonzales. http//www.ispe.org/ Última fecha de visita 03/10/2004.

[J2EE] Java Technology. Java 2 Platform, Enterprise Edition (J2EE). http://java.sun.com/j2ee/index.jsp Última fecha de visita 03/10/2004.

[Microsoft Excel] Microsoft Excel Home Page. http://www.microsoft.com/office/excel/default.asp Última fecha de visita 04/10/2004.

[Microsoft Outlook] Microsoft Outlook Home Page. http://office.microsoft.com/outlook Última fecha de visita 04/10/2004.

[Microsoft Project] Microsoft Project Home Page. http://office.microsoft.com/project Última fecha de visita 04/10/2004.

[Microsoft Visual FoxPro] Microsoft Visual FoxPro Home Page. http://www.microsoft.com/latam/vfoxpro/default.asp Última fecha de visita 03/10/2004.

[Struts] The Apache Struts Web Application Framework. http://struts.apache.org Última fecha de visita 27/09/2004.

[UML] G. Booch, I. Jacobson, J. Rumbaugh. The Unified Modeling Language User Guide. Addison-Wesley. ISBN 0-201-57168-4.

[XP] eXtreme Programming. http://www.extremeprogramming.org/ Última fecha de visita: 29/08/2004

[XPlanner] XPlanner. http://www.xplanner.org/ Última fecha de visita: 22/08/2004

Apéndices

Apéndice A

Estado del arte de gestión de proyectos de software

Este apéndice es la base teórica para todo el desarrollo de este informe, expone algunas metodologías de gestión en general y de software. Se recomienda especialmente, la lectura de las secciones A.1 y A.3 en la página 70, aún para aquellas personas con conocimiento en los temas tratados.

A.1. Introducción

Si uno quiere construir algo tan simple como la casilla de un perro, entonces sólo necesita clavos, maderas cortadas, un martillo, y un poco de imaginación y voluntad. En un par de días está construida. Si en cambio uno quiere construir una casa o algo más grande como un edificio, entonces la construcción no es tan simple. Hay que realizar estudio del terreno, los planos, contratar al personal adecuado, analizar, listar, y presupuestar los materiales necesarios, etc.

En el caso de los proyectos de software las técnicas no son tan directas. Se asume que no se puede construir un software con la metodología de la casilla de un perro, pero tampoco se pueden aplicar siempre las mismas técnicas y herramientas como en la construcción de un edificio. Las dificultades en el desarrollo de un proyecto de software, y los objetivos del mismo no son tan claros para los clientes. La planificación de los proyectos dependen fuertemente de los escenarios en los que se plantean, las personas que lo desarrollan, los objetivos y expectativas de los clientes, usuarios y quienes financian los proyectos, los tiempos de terminación que se esperan, los cambios, el agregado de nuevas funcionalidades, etc.

Se han desarrollado metodologías para los diferentes escenarios de construcción de software, desde el departamento de defensa de Estados Unidos, donde se espera una fuerte regulación y documentación, hasta empresas donde los clientes cambian frecuentemente los requerimientos y están trabajando junto con los programadores, diseñadores y realizadores de testeos.

Este Apéndice muestra las características generales de las diferentes metodologías de gestión de software de la actualidad.

En el Capítulo A.2 en la página siguiente, se citan metodologías gestión de proyectos. Este comienza mostrando PMBOK [PMBOK], una guía de gestión general de proyectos, siguiendo con metodologías de desarrollo de software tradicionales y ágiles. En el Capítulo A.3 en la página 70, se realiza una comparación entre las metodologías de software presentadas.

Al final del Apéndice B.2.3.3.2 en la página 80, se citan todos los vínculos de la bibliografía con un comentario de su contenido.

La bibliografía tiene dos clases de referencias las citadas en el texto, y la consultadas en la investigación que no son referenciadas están marcadas con *.

A.2. Metodologías de gestión de proyectos

A.2.1. Gestión general de proyectos

A.2.1.1. A guide to the project management body of knowledge[1] (PMBOK)

El objetivo principal de PMBOK [PMBOK] es identificar y describir el subconjunto de conocimientos y prácticas que son aplicables a la mayoría de los proyectos, en casi su totalidad, en la mayoría de los casos, y cuyo valor y utilidad son ampliamente conocidos. No es intención de este Documento que dicho conocimiento y prácticas sean aplicadas uniformemente a todos los proyectos, el equipo de gestión del proyecto es el responsable de determinar que es lo apropiado para cada proyecto. Este Documento brinda una referencia básica para cualquier persona interesada en la profesión de la gestión de proyectos.

Este documento es también usado por Project Management Institute para brindar una estructura consistente a sus programas de desarrollo profesional, inclusive:

- Certificación de profesionales de gestión de proyectos (Certification of project Management Proffessionals, PMPs)
- Acreditación de grado de programas de educación en gestión de proyectos (Accreditation of degree-granting educational programs in project management)

A.2.1.2. Proceso

Los procesos de gestión de un proyecto pueden organizarse en cinco grupos de uno o más procesos cada uno, estos son:

- Procesos de inicialización: Procesos de reconocimiento que un proyecto o fase debe comenzar y realización del compromiso a realizarlo.
- Procesos de planificación.
- Procesos de ejecución.
- Procesos de control.
- Procesos de cierre.

[1] No se encontró una traducción exacta del título “A guide to the project management body of knowledge” la más correcta sería “Una guía a una gran cantidad de conocimiento de la gestión de proyectos”.

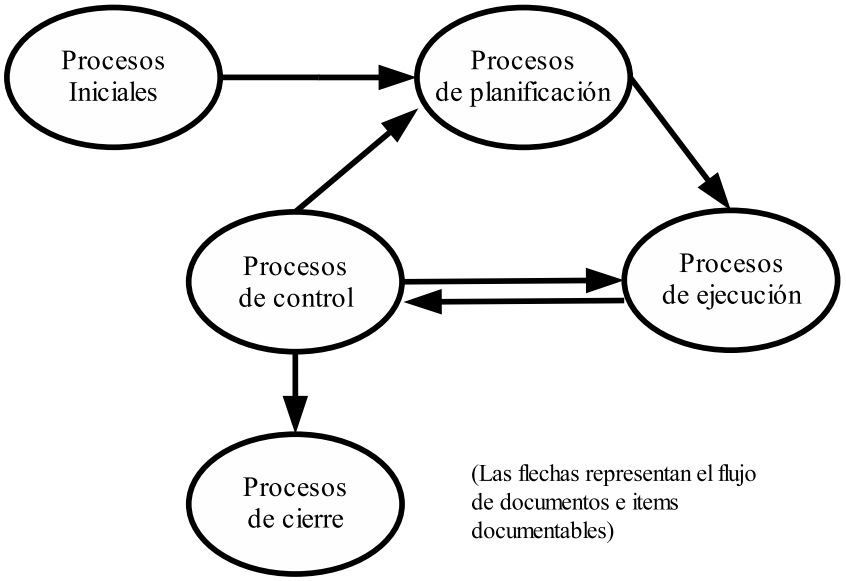

Figura A.1: Relaciones entre grupos de procesos en una fase

Los procesos que componen estos conjuntos están organizados por áreas, de la siguiente manera:

- Gestión de integración del proyecto
  - Desarrollo del plan del proyecto
  - Ejecución del plan del proyecto
  - Control de cambios globales
- Gestión de alcance del proyecto
  - Inicialización
  - Planificación de alcance
  - Definición de alcance
  - Verificación del alcance
  - Control de cambios del alcance
- Gestión de tiempo del proyecto
  - Definición de las actividades
  - Secuenciamiento de las actividades
  - Estimación de la duración de las actividades
  - Desarrollo del calendario
  - Control del calendario
- Gestión de costo del proyecto
  - Planificación de recursos
  - Estimación de costos
  - Estimación del presupuesto

    - Control de costos
- Gestión de calidad del proyecto
    - Planificación de la calidad
    - Aseguramiento de la calidad
    - Control de la calidad
- Gestión de recursos humanos
    - Planificación de la organización
    - Adquisición del personal
    - Desarrollo de equipos
- Gestión de comunicación del proyecto
    - Planificación de las comunicaciones
    - Distribución de la información
    - Reportes de rendimiento
    - Cierre administrativo
- Gestión de riesgos del proyecto
    - Identificación de riesgos
    - Cuantificación de riesgos
    - Desarrollo de respuestas a los riesgos
    - Control de respuestas a los riesgos
- Gestión de abastecimiento del proyecto
    - Planificación de abastecimiento
    - Planificación de solicitudes
    - Solicitudes
    - Selección de proveedores
    - Administración de contrato
    - Terminación de contrato

A.2.2. Metodologías Tradicionales de gestión de software

A.2.2.1. CMM [CMM Software]

CMM [CMM Software] define la madurez de un proyecto de software como una extensión para la cual un proceso específico esta explícitamente definido, gestionado, medido, controlado y es efectivo. Esta implica que la productividad y calidad resultante del proceso de una organización de software puede ser mejorada en el tiempo a través de ganancias consistentes en la disciplina lograda usando este proceso de software.

CMM [CMM Software] cataloga organizaciones de software en cinco niveles de madurez.

- Nivel 1: Inicial. Procesos ad hoc.
- Nivel 2: Repetible. Procesos de seguimiento de costos, calendario y funcionalidades definidos.
- Nivel 3: Definido. Las actividades de gestión e ingeniería de los procesos de software están documentadas, estandarizadas, e integradas en un proceso de software estándar de la organización. Todos los proyectos usan un proceso estándar de la organización aprobado para el desarrollo y mantenimiento de software.
- Nivel 4: Gestionado. Los productos y procesos de software están cuantitativamente entendidos y controlados.
- Nivel 5: Optimizante. Las organizaciones están enfocadas en la mejora continua. Tiene como objetivo prevenir la ocurrencia de defectos, identificando los puntos fuertes y débiles del proceso pro-activamente. Mejoran también incorporando las nuevas tecnologías y métodos.

Cada nivel implica los anteriores, es decir, si se está en el nivel 4, por ejemplo, cumple con todas las características de los niveles 3 y 2. Para pasar de un nivel a otro se deben cumplir objetivos, estos son secuenciales. Esto establece prioridades y orden en las características de los procesos.

CMM [CMM Software] no establece como lograr estos niveles, sólo brinda pautas que se deben cumplir.

Cada nivel de madurez esta compuesto por varias áreas claves del proceso. Estas están organizadas en cinco secciones llamadas características comunes. Las características comunes especifican las prácticas claves que llevan a cumplir los objetivos de las áreas claves.

Los niveles de madurez indican la capacidad del proceso y contienen Áreas Claves del proceso. Estas Áreas cumplen objetivos y están organizadas por Características comunes.

A.2.2.1.1. Esquema de los niveles de madurez CMM [CMM Software]

- Inicial
- Repetible
  - Gestión de requerimientos
  - Planificación de proyecto de software
  - Seguimiento y verificación del proyecto de software
  - Gestión de configuración de software
  - Asesoramiento de calidad de software
  - Gestión de subcontratación de software
- Definido
  - Foco en el proceso de la organización
  - Definición del proceso de la organización
  - Programa de entrenamiento
  - Gestión de software integrado
  - Ingeniería del producto de software
  - Coordinación entre grupos
  - Revisión por pares

- Gestionado
  - Gestión cuantitativa del proceso
  - Gestión de calidad de software
- Optimizante
  - Prevención de defectos
  - Gestión de cambios de la tecnología
  - Gestión de cambios del proceso

A.2.2.1.2. Roles en la organización

- Gerente (Manager)
- Gerente superior (Senior manager)
- Gerente del proyecto (Project manager)
- Gerente del proyecto de Software (Project software manager)
- Gerente de software de primera linea (First-line software manager)
- Líder de tarea de software (Software task leader)
- Personal, personal de ingeniería de software, individuos. (Staff, software engineering staff, individuals)

A.2.2.1.3. Grupos

- Grupo de ingeniería de software (Software engineering group)
- Grupos relacionados con software (Software-related groups)
- Grupo de ingeniería de proceso de software (Software engineering process group)
- Group de ingeniería de sistema (System engineering group)
- Grupo de testeo del sistema (System test group)
- Grupo de asesoramiento de calidad de software (Software quality assurance group)
- Grupo de gestión de configuración de software (Software configuration management group)
- Grupo de entrenamiento (Trainning Group)

A.2.3. Metodologías Ágiles de gestión de software

Los procesos ágiles son una respuesta o reacción a la imposibilidad de realizar los procesos tradicionales, bajo determinados escenarios.

Enfoques metodologías		
Enfoque	**Metodologías tradicionales**	**Metodologías ágiles**
Política	Individuos e iteraciones	Procesos y herramientas
Entregas	Software que funciona	Documentación completa
Cliente	Colaboración con el cliente	Negociación de un contrato
Organización	Responder al cambio	Seguir un plan

Cuadro A.2: Enfoques metodologías tradicionales y ágiles

A.2.3.1. eXtreme Programming (XP [3. XP])

XP [3. XP] nació en respuesta a los problemas causados por procesos tradicionales, en un entorno donde los proyectos son vagos y cambiantes. La metodología XP [3. XP] es un conjunto de principios y prácticas llevados a niveles extremos.

La programación se realiza de a pares; mientras uno programa, el otro chequea que el enfoque sea el más adecuado y está realizado de la forma más simple. Estos pares y la propiedad de las clases son dinámicas. Se utilizan estándares de codificación. El refactoring es continuo, ya que el sistema debe permanecer simple, para minimizar el impacto de los cambios. Cada módulo es desarrollado junto con un testeo automático del mismo. La integración se realiza, lo antes posible, al menos una vez al día, esto está soportado por los testeos de integración automáticos. Cada persona en el equipo no debe trabajar más de 40 horas semanales, al menos, no debe realizar horas extra más de 2 semanas seguidas.

El cliente es parte del equipo, es quien escribe el conjunto de "historias". Estas son características del sistema a ser construidas, escritas en tarjetas, en lenguaje natural. El equipo estima el esfuerzo necesario para construirlas, mientras que el cliente le da prioridad según su valor para el negocio. En base a esto, el cliente, selecciona un subconjunto de "historias" a ser implementadas en iteraciones menores a 30 días. La idea es realizar las funcionalidades con más valor para el cliente lo antes posible, lo que lleva a una retroalimentación temprana.

A.2.3.1.1. Roles

- Programador
- Cliente
- Testeador
- Consultor de testeos del cliente
- Encargado del seguimiento
- Árbitro
- Consultor
- Gerente general

A.2.3.2. Scrum [Scrum]

El significado de la palabra Scrum, en inglés, además de ser una posición de los jugadores de rugby, se traduce como un conjunto de personas que se están empujando unas a las otras para tratar de obtener algo.

Los equipos en esta metodología se llaman "scrums", están liderados por un jefe[2], y no tienen más de 10 personas. En caso de ser más personas, se organizan "scrums de scrums", donde los jefes forman otro scrum, y así sucesivamente. Los integrantes de los estos equipos trabajan tiempo completo.

Los requerimientos del sistema se mantienen en una lista llamada "BackLog List", cuyo subconjunto "Sprint BackLog" son las funcionalidades a desarrollar en la siguiente iteración. La iteraciones van desde una semana hasta un mes, y se llaman "sprints". La funcionalidades construidas en los sprints no pueden ser modificadas hasta el siguiente, pero si pueden ser reducidas ya que la fecha de entrega es fija. Los scrums mantienen reuniones de avance diarias, intensivas, de no más de 20 minutos, lideradas por el jefe del scrum.

El cliente es cooperativo, y es quien fija las prioridades de la tareas de la lista "BackLog List". Se realizan reuniones de planificación donde participan todos los scrums, la gerencia y los clientes. En esta se evalúan los resultados y el cliente modifica las funcionalidades y prioridades en la "BackLog List".

A.2.3.2.1. Roles

- Jefe del scrum
- Dueño del producto
- Equipo del scrum
- Cliente
- Gerencia

A.2.3.3. Familia de metodologías Crystal

Crystal es un conjunto de metodologías. Para cada proyecto en particular se selecciona la más adecuada según su criticidad y tamaño. La criticidad se refiere a la perdida potencial que causaría una falla en el sistema, y se catalogan en: Comodidad, Dinero discrecional, Dinero esencial, y Vida. Cada miembro de la familia Crystal es marcado con un color que indica el “peso” de la metodología; es decir, cuanto más oscuro es el color de la metodología, más crítico y más grande es el proyecto indicado para usarla. Todas se basan en ciclos de desarrollo incremental, intentan reducir el trabajo de productos intermedios, y requieren que las personas sean colaborativas y comunicativas. Existen tres metodologías Crystal: Crystal Clear, Crystal Orange, y Crystal Orange Web. Las dos primeras han sido implementadas.

A.2.3.3.1. Crystal Clear y Crystal Orange [Crystal methods]

Aplicabilidad Crystal Clear [Crystal methods] puede ser aplicado para proyectos de hasta 5 personas, con criticidad Dinero Discrecional. Mejorando los aspectos de testeo y comunicación puede ser aplicado a proyectos de hasta 8 personas con criticidad Dinero Esencial o a proyectos de hasta 10 personas con criticidad Dinero Discrecional.

Crystal Orange [Crystal methods] puede ser aplicado a proyectos entre 10 y 40 personas con criticidad Dinero Discrecional. Mejorando los procesos de verificación se puede aplicar a proyectos de hasta 50 personas con criticidad Dinero Esencial.

[2]En inglés, Scrum Master.

Políticas Estándar La políticas son las prácticas necesarias para ser aplicadas durante el proyecto de desarrollo. Estas pueden ser reemplazadas por políticas similares de XP [3. XP].

Ambas metodologías comparten la mayoría de las políticas. Requieren testeos funcionales automáticos, dos revisiones del usuario en cada entrega y un involucramiento directo del mismo. Se enfocan en tomar más decisiones y en generar menos documentos. Se fijan hitos, y se hace un seguimiento en base a estos. Las entregas son incrementales sobre una base regular. La iteraciones Crystal Clear [Crystal methods] son de 2 a 3 meses, mientras que en Crystal Orange [Crystal methods] duran hasta 4 meses.

A.2.3.4. Feature Driven Development[3] (FDD) [3. FDD]

FDD [3. FDD] fue creado por Peter Coad, cuyo trabajo fue extendido junto con Jeff De Luca y Stephen Palmer sobre la base del trabajo en un gran proyecto. Se enfoca en iteraciones de 2 días hasta 2 semanas. FDD [3. FDD] tiene cinco procesos.

- Desarrollar un modelo global (Develop an overall model)
- Construir una lista de características (Build a features list)
- Planear por característica (Plan by feature)
- Diseñar por característica (Design by feature)
- Construir por característica (Build by feature)

Los tres primeros se hacen al principio del proyecto, y los dos últimos son las fases de las iteraciones.

A.2.3.4.1. Roles

- Claves
  - Gerente del proyecto (Project manager)
  - Jefe de arquitectura (Chief architect)
  - Gerente de desarrollo (Development manager)
  - Jefe de programadores (Chief programmer)
  - Dueño de clase (Class owner)
  - Experto de dominio (Domain expert)
- De Soporte
  - Gerente de entregas (Release manager)
  - Abogado/Gurú del lenguaje (Language lawyer/language guru)
  - Ingeniero de construcción (Build Engineer)
  - Desarrollador de software de soporte (toolsmith)
  - Administrador del sistema (System administrator)
- Adicionales
  - Realizadores de testeos (testers)
  - Encargado de instalar nuevas versiones (Deployers)
  - Escritores técnicos (technical writers)

[3] En español sería “Desarrollo guiado por características”

A.3. Comparación de metodologías de gestión de software.

A.3.1. Equipos y Roles

Al iniciar un proyecto una de las cosas necesarias para evaluar el presupuesto es la cantidad y calidad del personal necesario para implementarlo. Aquí se plantea una comparación entre los diferentes roles y equipos de las metodologías planteadas en el capítulo anterior.

A.3.1.1. Escalabilidad

En la Figura A.2, se muestra la máxima cantidad de personas, desarrolladores, analistas, programadores, etc., que pueden trabajar en un proyecto, según cada metodología.

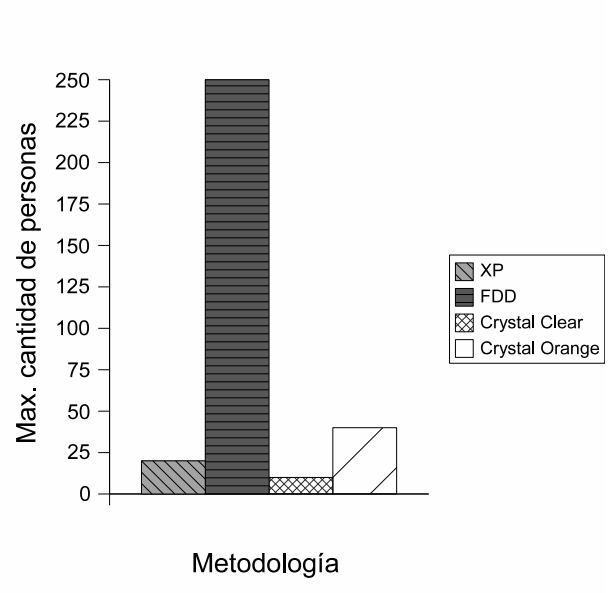

Figura A.2: Escalabilidad procesos

CMM [CMM Software] y Scrum [Scrum] son totalmente escalables.

A.3.1.2. Rol del cliente

En la Figura A.3, se muestra el porcentaje de participación del cliente en diferentes etapas del proceso. El cliente es un rol de XP [3. XP] y Scrum [Scrum], en las demás sólo participa en la relevamiento de requerimientos.

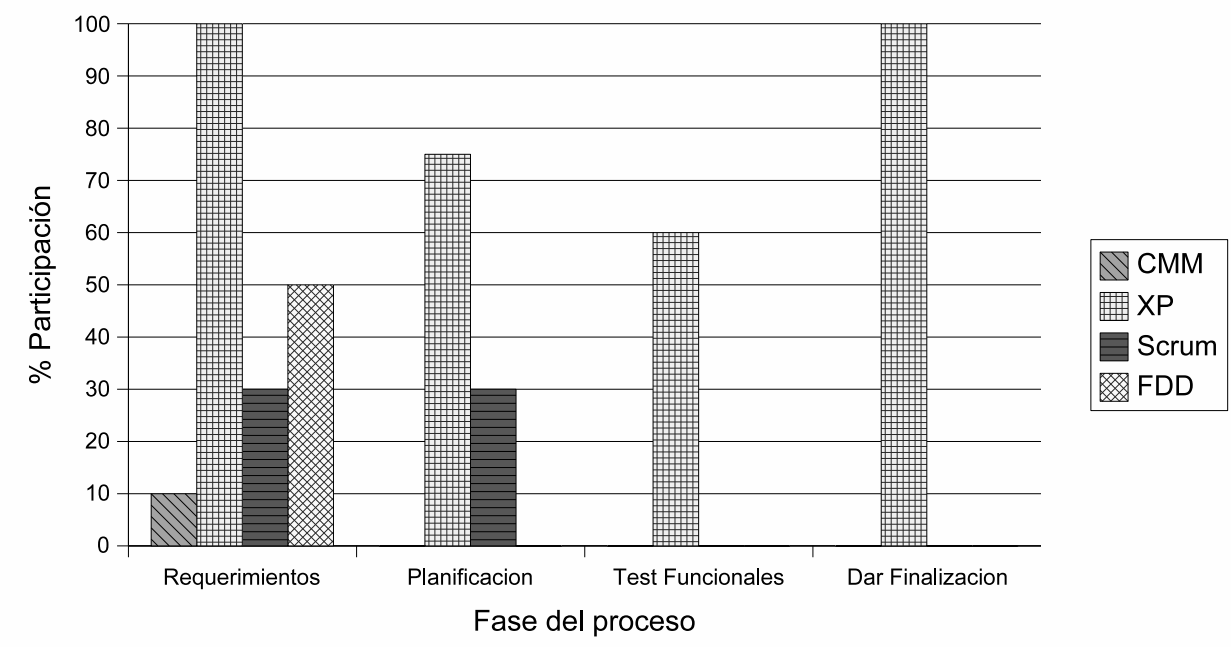

Figura A.3: Participación del cliente en diferentes etapas del proceso

A.3.2. Comparación por factores

Factor	**Metodologías Tradicionales**	**Metodologías Ágiles**
Alcance (Requerimientos)	Bien conocidos, Tamaño bien entendido No van a cambiar	Inciertos No se conoce el alcance Sujeto al cambio
Recursos (dinero, infraestructura, personas)	Aprobados y disponibles Han sido hecho antes Presupuesto es suficiente y esta fundamentado Personas son familiares con las tareas y las herramientas	No están completamente aprobados o disponibles Se necesita dar prueba de su necesidad Dinero esta controlado Presupuesto incierto Se necesitan nuevas habilidades
Tiempo	Claramente definido Hitos definidos	No está bien definido/está abierto Hitos poco claros Sujeto a cambios
Riesgos	Bien entendido Impacto mínimo	Nuevas tecnologías Riesgos desconocidos Impacto alto

Bibliografía

[CMM Software] Chrissis, M. B., Curtis, B., Paulk, M. C., Weber, C. V. (1993). Capability Maturity Model^{SM}for Software, Version 1.1. Software Engineering Institute.

[CMM Software: Key Practices] Bush, M., Chrissis, M. B., Garcia, S.M., Paulk, M. C., Weber, C. V. (1993). Key Practices of the Capability Maturity Model^{SM}, Version 1.1. Software Engineering Institute.

[Crystal methods] Crystal Methods. http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/heterodox.asp#8. Última fecha de visita 26/03/2005.

[FDD Coad - Palmer] Coad, P., Palmer, S.(2002). A Practical Guide to Feature-Driven Development (Presentación del libro, ISBN 0-13-067615-2).

[Manifiesto Agil] Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R.C., Mellor, S., Schwaber, K., Sutherland, J., Thomas, D. (2001). Manifiesto for Agile Software Development. http://www.agilemanifesto.org/. Última fecha de visita 26/03/2005.

[PMBOK] Duncan, W.R. (1996). A guide to the Project Management Body Of Knowledge. Project Management Institute.

[Scrum] Scrum. Proceso de desarrollo de Software. http://www.dcc.unicamp.br/~ra022247/Arquivos/scrum.pdf. Última fecha de visita 26/03/2005.

[Sw. Ágil Abrahamsson - Salo - Ronkainen - Warsta] Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J.(2002). Agile Software Development Methods, Review and Analysis. VTT Electronics Publications.

[XP Perez] Perez, B. (2003). Extreme Programming.

[* Calidad de Software - Gilles] Gilles, A.G. (1992) Software Quality. Theory and management. Chapman & Hall Computing. ISBN 0-4124-5130-1

[* Calidad de software - Horch] CHorch, J.W. (2003) Practical Guide to Software Quality Management. Segunda edición. Artech House. ISBN 1-5805-3527-5

[* Calidad de software - Cavanagh et.al.] Cavanagh, R.R., Neuman, R.P., Pande, P.S. (2000) The Six Sigma Way. McGraw-Hill.

[* Calidad de Software - Gilles] Gilles, A. G. (1992) Software Quality. Theory and management. Chapman & Hall Computing. ISBN 0-4124-5130-1

[* Calidad de software - Horch] CHorch, J.W. (2003) Practical Guide to Software Quality Management. Segunda edición. Artech House. ISBN 1-5805-3527-5

[* Costos - Isakowitz] CIsakowitz, J.I. (2002) Cost estimation handbook. NASA HQ. (196 páginas)

[* Desarrollo ágil - Highsmith] DHighsmith, J. (2002) What is agile software development? CrossTalk The Journal of Defense Software Engineering.

[* Desarrollo ágil - Paetsch] Paetsch, F. (2003) Requirements enginnering in agile software development. Diploma thesis. University of Calgary.

[* eXtreme programming - Beck, Fowler] eBeck, K., Fowler, M. (2000). Planning extreme programming. Primera edición. Addison Wesley. ISBN 0-2017-1091-9 (160 páginas)

[* eXtreme programming - Burke, et.al.] eBurke, E.M., Coyner, B.M. (2003) Java extreme programming cookbook. O'Reilly ISBN 0-5960-0387-0 (288 páginas)

[* eXtreme Programming - Hightower et.al] Hightower, R., Lesiecki, N. (2002) Java tools for extreme programming - Mastering open source tools including Ant, JUnit, and Cactus. Wiley computer publishing ISBN 0-4712-0708-X

[* FDD Nebulon] Nebulon Pty. Ltd. FDD Certified. http://www.nebulon.com Info Última fecha de visita 28/03/2005.

[* Ingeniería de software - Bryan, Siegel] Bryan, W.L., Siegel, S. (1988) Software product Assurance - Techniques for Reducing Software Risk. Elsevier

[* Ingeniería de Software - Pfleeger] Pfleeger, S.L. (1991) Software Engineering. The production of quality Software. Segunda edición. Maxwell Macmillan International. ISBN 0-0294-6488-9

[* RUP - Kruchten] /Kruchten, P. (1999) The Rational Unified Process. An Intoduction. Addison-Wesley. ISBN 0-2016-0459-0 (255 páginas)

[* RUP - NIJR] NIJR (2007). Presentación dX Rup. http://web.nijt.edu/~gblank/cis683/dX%20RUP.ppt Última fecha de visita 02/10/2004

[* Software Management - Royce] Royce, W. (1998) Software project mangement. A unified framework. Addison Weslwy Longman, Inc. ISBN 0-2013-0958-0 (406 páginas)

Bibliografía comentada Apéndice A

[CMM Software] Este trabajo fue financiado por el Departamento de Defensa de Estados Unidos. Creado en la Instituto de Ingeniería de Software, Universidad de Carnegie Mellon. Fue iniciado en respuesta al pedido de brindar al gobierno federal de Estados Unidos un método de asesoría de la capacidad de sus contratados de software. Este Apéndice presenta una introducción a CMM y sus productos asociados.

[CMM Software: Key Practices] Este trabajo es una ampliación de [CMM Software] describe practicas claves de las áreas clave de los niveles de madurez CMM. Estas descripciones establecen: los objetivos, las acciones necesarias para asegurar que el proceso esta establecido y perdurara, las precondiciones que deben existir en el proyecto o en la organización para implementar una área clave del proceso, las necesidades y análisis de medición, los pasos para asegurar que las actividades están siendo realizadas de acuerdo al proceso establecido.

[FDD Coad - Palmer] Diapositivas de la Presentación del libro A Practical Guide to Feature-Driven Process. Presenta los principios, definiciones y estrategias de calidad, enfoques, prácticas, y procesos de FDD. Es una excelente introducción a FDD, escrita por sus propios desarrolladores.

[Manifiesto Agil]Términos en los que se basa los procesos ágiles.

[Mediciones: Munson]Este texto describe como gestionar sistemas de desarrollo de mediciones de software, como construir herramientas de mediciones y estándares, y como construir experimentos controlados usando herramientas de medición estándar.

[PMBOK] Este documento trata sobre gestión general de proyectos. Establece definiciones de términos comúnmente usados, y describe: influencias de la estructura organizacional, habilidades generales clave, ciclos de vida, fases y procesos.

[Sw. Ágil Abrahamsson - Salo - Ronkainen - Warsta]Describe las metodologías ágiles. Los procesos, roles, responsabilidades , prácticas, ámbito de uso y las investigaciones actuales del los métodos ágiles existentes. Estos son: Extreme programming, Scrum, Crystal family of methodologies, Feature driven development, The Rational Unified process, Dynamic Systems Development Method, Adaptative Software Development, Open Source Software development. Termina con una comparación de todos estos métodos.

[XP Perez] Es una presentación donde muestra el manifiesto ágil y sus principios, y las variables, los principios, valores y practicas de extreme programming.

Apéndice B

Medición de Software

> "Sin la información adecuada, tú eres simplemente otra persona con una opinión."
>
> -Tracy O'Rouke, CEO[a] de Allen-Bradley

[a] Chief Executive Officer, se refiere a quien es responsable máximo operativo de una organización.

Supongamos que un fabricante de autos quiere construir un auto económico, que rinda por lo menos x km/litro. Entonces se le indica al equipo de ingenieros que diseñen un motor económico. Estos comienzan un proceso iterativo de diseño, construcción y prueba. En cada nuevo diseño miden el rendimiento del motor lo que les indica si el nuevo diseño es más económico que los anteriores, si están mejorando o no, y toman decisiones basándose en la mediciones. Lo que busca el fabricante es el atributo de calidad económico, y los ingenieros van mejorando la calidad de los motores tomando mediciones.

La calidad en un proceso es producto del esfuerzo y no de la casualidad, incluso en productos de software. La calidad comienza por la institución de un programa de medición. Este permite comparar atributos de manera de lograr aprender de los errores cometidos y de los éxitos logrados. Se mide el pasado, y se lo compara con el presente para mejorar el futuro.

B.1. Medición

La medición es obtener datos de manera estándar para poder comparar. Las estadísticas convierten dichos datos en información. Permiten entender que indican los datos sobre los procesos. Una definición formal de medición es: Medición es la correspondencia entre números o símbolos y atributos de objetos de acuerdo a una regla predeterminada.

Uno de los problemas en la medición de software es la falta de estándares para medir, por ejemplo, la productividad, el código, o la sobrecarga de un sistema operativo. Se debe evitar sacar malas conclusiones, lo que lleva a tener especial cuidado cuando se quiere medir personas.

Lo primero que hay que hacer al iniciar un proceso de medición es establecer que es lo que se quiere medir, luego analizar cuales son los factores que influyen en lo que se quiere medir, para luego recolectar datos de los mismos y crear y calcular los indicadores que respondan a lo que se estableció en el primer paso que se quería medir. Finalmente se realiza la verificación de dichos indicadores.

La medición de software sigue el método científico ya que se debe elaborar una hipótesis, experimentar y validar. Los errores en la medición pueden ser causa de las herramientas que se utilizan, inherentes al método o debido a redondeos en el procesamiento de datos en una computadora.

B.2. Medición de software

La medición de software depende de los indicadores de negocio, proyecto, y producto, y de los objetivos de los procesos. Existen varias propuestas para medir software, en esta sección se muestran las que se consideraron más relevantes para el Proyecto.

Generalmente se tiende a atribuir la extensión de los plazos de la construcción de software a la productividad de los ingenieros o analistas, esto no es necesariamente así. La construcción de software depende desde la especificación de los requerimientos, pasando por su diseño, implementación, testeo y puesta en producción, hasta el uso del sistema por los usuarios y su mantenimiento. Estas etapas a su vez dependen del proceso de software, la disponibilidad de recursos y el entorno. Determinar cuales son los factores que extienden los plazos y los costos, es sólo posible midiendo todos los factores que influyen en la construcción de un software. Algunos de estos factores son las características que determinan la calidad de un software, según ISO 9126 (Figura 2-2 [GQM - van Solingen, Berghout]), que son: funcionalidad, confiabilidad, facilidad de uso, eficiencia, facilidad de mantenimiento, y capacidad de operar en diferentes entornos. Un producto de buena calidad reduce los costos y el esfuerzo total de construcción.

GQM [GQM - van Solingen, Berghout], The Goal/Question/Metric Method, en español: Método Objetivo/Pregunta/Métrica, plantea una metodología para construir métricas orientado en objetivos con un "enfoque de arriba hacia abajo"[1]. El paradigma GQM[GQM - van Solingen, Berghout] GQM define un objetivo, aclara dicho objetivo con preguntas, y define métricas que deberían dar información de las respuestas a dichas preguntas. Se tomará en cuenta este enfoque en este Proyecto.

B.2.1. Productividad

Lo primero que se quiere medir es la productividad, sin embargo, en esta investigación no se ha encontrado una medida confiable de la misma.

Una de las medidas más conocidas y más utilizadas para medir el volumen del trabajo generado son las líneas de código, desde ahora en adelante “LOC”. Entre sus variaciones se presentan incluir o no las líneas en blanco, los comentarios o las declaraciones de variables. A pesar de encontrarse en la mayoría de los libros y los artículos, su uso no es recomendado ampliamente, debido a que no todas las líneas de código tienen la misma complejidad: no toman en cuenta ni la redundancia, ni la funcionalidad, ni el esfuerzo real, ni su reutilización. Una misma lógica se puede implementar en diferente cantidad de líneas de código, siendo muchas veces más fácil de mantener con menos líneas de código. Entonces el volumen de software se podría medir como la sumatoria de las complejidades de cada línea de código realizada, es decir:

$$v = \sum_{l \in \Omega} c(l)$$

donde v es el volumen, Ω representa el conjunto de líneas de código escritas, y c la función de complejidad antes mencionada. Teniendo el valor del volumen de software v se tienen los siguientes indicadores:

$$p_{programador} = \frac{v}{e_{meses}}$$

donde $p_{programador}$ es la productividad del programador, v es el volumen de software, y e_{meses} el esfuerzo en meses del programador, i.e., la cantidad de meses que llevo construir el software.

[1] El "enfoque de arriba hacia abajo”, de su término original en inglés "top-down approach", se trata de enfocar un problema comenzado por los conceptos más generales, ”desde arriba”, los cuales se van dividiendo “hacia abajo” en partes más concretas y específicas hasta lograr el objetivo deseado.

B.2.2. Indicador de costo del personal

El costo del personal se puede medir como:

$$costo_{personal} = \sum_{p \in Personal} (t_{normal}(p) \star c_{normal}(p) \dagger t_{extra}(p) \star c_{extra}(p))$$

donde $t_{periodo}(p)$ es cantidad tiempo normal trabajado en unidades de tiempo, $c_{periodo}(p)$ es costo de la unidad de tiempo de $t_{periodo}(p)$, $t_{extra}(p)$ es cantidad tiempo extra trabajado en unidades de tiempo, $c_{extra}(p)$ es costo de la unidad de tiempo extra de $t_{extra}(p)$ y $Personal$ es el conjunto de personas dedicadas a construir el software cuyo costo se quiere calcular. Las unidades de tiempo pueden ser cualquiera según corresponda.

B.2.3. Mantenimiento

Según [Fundamentals of Software Engineering - Ghezzi et.al.] el costo del mantenimiento es a menudo más del 60 % del total del costo de la construcción de un sistema de software, y el 20 % del mantenimiento se atribuye a corregir errores, y adaptar la aplicación a los cambios de entorno, y el 50 % se atribuye a mejorar, cambiar, y agregar nuevas funcionalidades y cualidades. Tomando en cuenta esto, en las siguientes secciones se mostrarán métricas que permitan verificar o descartar este enunciado.

B.2.3.1. Clasificación de errores

No todos los errores tienen el mismo impacto, ni la misma forma de evitarlos. Por esto se consideró necesario una clasificación que permita distinguirlos. Se definieron dos grandes clases de errores que se nombraron fallas y desaciertos. Las fallas se refieren a comportamiento inesperado del sistema, y los desaciertos a los errores humanos que llevan a un comportamiento incorrecto del sistema.

Se identificaron tres clases de desaciertos: desacierto con el cliente, desacierto operacional y desacierto interno. Los primeros dos se refieren a la comunicación entre el personal de software y los clientes y el personal de software y los usuarios respectivamente. Desacierto interno se refiere a la comunicación dentro del equipo de construcción de software.

1. Desacierto con el cliente. La comunicación con el cliente fue desacertada. Se refiere a los requerimientos mal especificados por el cliente, o mal interpretado por la empresa. Este se puede dar en cualquier etapa, y lleva a un comportamiento incorrecto del software, i.e. la funcionalidad se realiza pero de manera errónea .
2. Defecto requerimiento. El requerimiento no fue correctamente especificado por el cliente, normalmente implica volver a definirlo.
3. Defecto operacional. Se refiere a la mala interpretación de la manipulación del sistema por el usuario. Por ejemplo, en los datos de una persona se ingresa la dirección de correo electrónico de forma incorrecta y se reporta una falla en el envío automático de correos electrónicos a la misma.
4. Falla de código. El diseño es correcto, el desarrollador interpretó correctamente el requerimiento, pero el código no se comporta como se esperaba. Este tipo de falla debería encontrarse en la etapa de testeo.
5. Falla de recursos/sistemas internos. Se refiere a los recursos y/o sistemas con los que se interactúa cuya responsabilidad es propia.
6. Falla de recursos/sistemas externos. Se refiere a los recursos y/o sistemas con los que se interactúa cuya responsabilidad es de terceros.

B.2.3.2. Métricas de mantenimiento

B.2.3.2.1. Densidad de errores La densidad de errores es una métrica que determina la cantidad de errores encontrados según el tamaño del módulo o producto:

$$\rho_{errores} = \frac{\sharp err}{v_{modulo}}$$

donde $\rho_{errores}$ es la densidad de errores, $\sharp err$ la cantidad de errores, y v_{modulo} el volumen del módulo.

B.2.3.2.2. Eficiencia de detección de errores La eficiencia de detección de errores edd se define como la cantidad de defectos detectados en la fase de testeo, $\sharp err_{detectados}$, sobre el total de encontrados hasta el momento, $\sharp err$:

$$ederr = \frac{\sharp err_{detectados}}{\sharp err}$$

Esta métrica varía en función del tiempo y debería tender a estabilizarse.

B.2.3.2.3. Eficiencia de testeo La eficiencia de testeo $eficiencia_{testeo}$ determina el porcentaje de ítems testeados, $\sharp items_{testeados}$, respecto a la cantidad de ítems a testear, $\sharp items$:

$$eficiencia_{testeo} = \frac{\sharp items_{testeados}}{\sharp items}$$

B.2.3.2.4. Porcentaje de tiempos de mantenimiento Para comprobar el estudio de Ghezzi [Fundamentals of Software Engineering - Ghezzi et.al.] citado al principio, se deben calcular los siguientes porcentajes:

$$factor\,costo\,mantenimiento\,costo\,total = \frac{costo_{mantenimiento}}{costo_{total}}$$

donde la fórmula de costos es la planteada en el punto B.2.2 en la página anterior. Se puede calcular otros porcentajes según la clasificación realizada en el punto B.2.3.1 en la página anterior.

B.2.3.3. Calidad

B.2.3.3.1. Precisión de la estimación de un plazo La precisión de la estimación de un plazo se puede medir como:

$$PEP = \frac{duración_{real}}{duración_{estimada}}$$

B.2.3.3.2. Calidad La calidad de un producto puede medirse en función de tiempo dedicado a la corrección de errores en función del tiempo total dedicado.

$$Calidad = \frac{tiempo_{corrección\,errores}}{tiempo_{total}}$$

Bibliografía

[Mediciones: Ishigaki - Jones] Ishigaki, D., Jones C. (2003). Practical Measurement in the Rational Unified Process.

[Mediciones: Munson] Munson, J.C. (2003). Software Engineering Measurement. Auerbach Publications.

[Fenton-Pfleeger Software Metrics] Fenton, E., Pfleeger, S.L. (1997). Software Metrics.

[Fundamentals of Software Engineering - Ghezzi et.al.] Ghezzi, C., Jazayeri, M., y Mandrioli, D.. (1991). Fundamentals of Software Engineering.

[GQM - van Solingen, Berghout] van Solingen, R., Berghout, E. (1999) The goal/question/metric method, a practical method for quality improvement of software development. McGraw-Hill. ISBN 007-709553-7.
https://www.gqm.nl/. Última fecha de visita 24/04/2005.

[GQM Guide - Florac et.al.] Florac, W.A., Goethert, W.B., Park, R.E. (1996) Goal-Driven Software Measurement, A Guidebook.
http://www.sei.cmu.edu/pub/documents/96.reports/pdf/hb0
Última fecha de visita 24/04/2005

Bibliografía comentada
Apéndice B

[Mediciones: Munson]Este texto describe como gestionar sistemas de desarrollo de mediciones de software, como construir herramientas de mediciones y estándares, y como construir experimentos controlados usando herramientas de medición estándar.

Apéndice C

Evaluación de herramientas

Este apéndice muestra herramientas de gestión de software disponibles. En el Capítulo C.1, describe las características generales de una herramienta general de proyectos de software. En los Capítulos C.2 en la página 100 y C.3 en la página 110, describen una herramienta para proyectos que siguen la metodología XP [3. XP] y FDD [3. FDD] respectivamente. En el Capítulo C.4 en la página 123, cita dos herramientas más de gestión general de proyectos de software. En el Capítulo C.5 en la página 127, realiza una comparación entre las herramientas anteriores. Finalmente en el Capítulo C.6 en la página 129, se realizan las aclaraciones finales y se muestran las conclusiones.

C.1. TUTOS

TUTOS de sus siglas en inglés, The Ultimate Team Organization Software, es el software fundamental para la organización de equipos. Es una aplicación web para ser usada por grupos de trabajo de software, o sistema ERP[1]/CRM[2]/PLM[3]. Esta herramienta fue evaluada en su versión 1.2.

El sistema es orientado a objetos, y estos son: citas, contactos (datos básicos de personas), compañías, departamentos de compañías, direcciones, proyectos o productos, tareas, recursos, notas, documentos, registros de tiempos, facturas[4] y permisos. Se accede a estos por medio de enlaces, o paneles de búsqueda, y están controlados por permisos. Los permisos en el sistema son cuatro: ver, ver y usar, ver, usar, y modificar, y ver, usar, modificar y borrar. Todos poseen una interfaz para su creación, visualización, modificación, y borrado.

Se ingresa al sitio con un usuario y contraseña, a la vista del calendario con las tareas y citas del usuario.

[1]ERP: De sus siglas en inglés: Enterprise Resource Planning, planificación de recursos de una empresa.
[2]CRM: De sus siglas en inglés: Customer Relationship Management, gestión de relacionamiento con el cliente.
[3]PLM: De sus siglas en inglés: Product Lifecycle Management, gestión del ciclo de vida de un producto.
[4]Las facturas no serán tratadas porque no son relevantes para el tema en estudio

C.1.1. Directorio

El directorio es un administrador de contactos (personas), y compañías. La compañías a su vez pueden tener departamentos. A cualquiera de estos puede asignarse una dirección.

El panel principal del directorio es un panel de búsqueda, ver figura C.1 en la página siguiente. Este presenta tres opciones, buscar entradas en el directorio, buscar en Internet o buscar compañías. Desde este panel pueden crearse compañías, departamentos o contactos.

El panel de visualización general del Directorio, ver figura C.2 en la página 84, muestra una tabla con nombre, apellido, ubicación (directorio), correo electrónico y teléfono.

En la figura C.3, se muestra el panel de permisos de un contacto. Los permisos donde se seleccionan quienes tienen permiso para ver, para ver y modificar, y para ver, ver y usar, ver, usar, y modificar, y ver, usar, modificar y borrar. Todos los objetos que tienen permisos, tienen acceso a este panel.

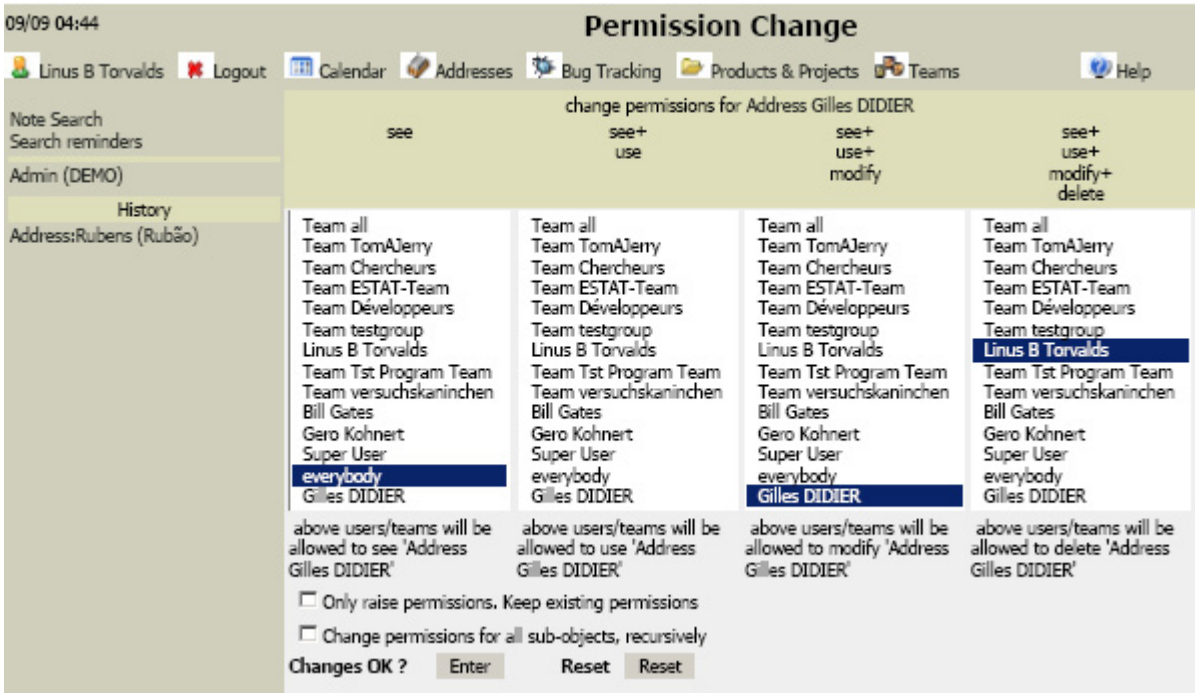

Figura C.3: TUTOS: Panel de gestión de permisos de un contacto

C.1.2. Calendario

El calendario es una vista de citas y tareas semanal, subdividida en días. Este es cargado por defecto al ingresar al sitio con las tareas y citas del usuario, ver figura C.4 en la página 85. También se puede ver el calendario con las tareas y citas de los equipos, de otros miembros del equipo, según los permisos, ver figura C.5 en la página 86.

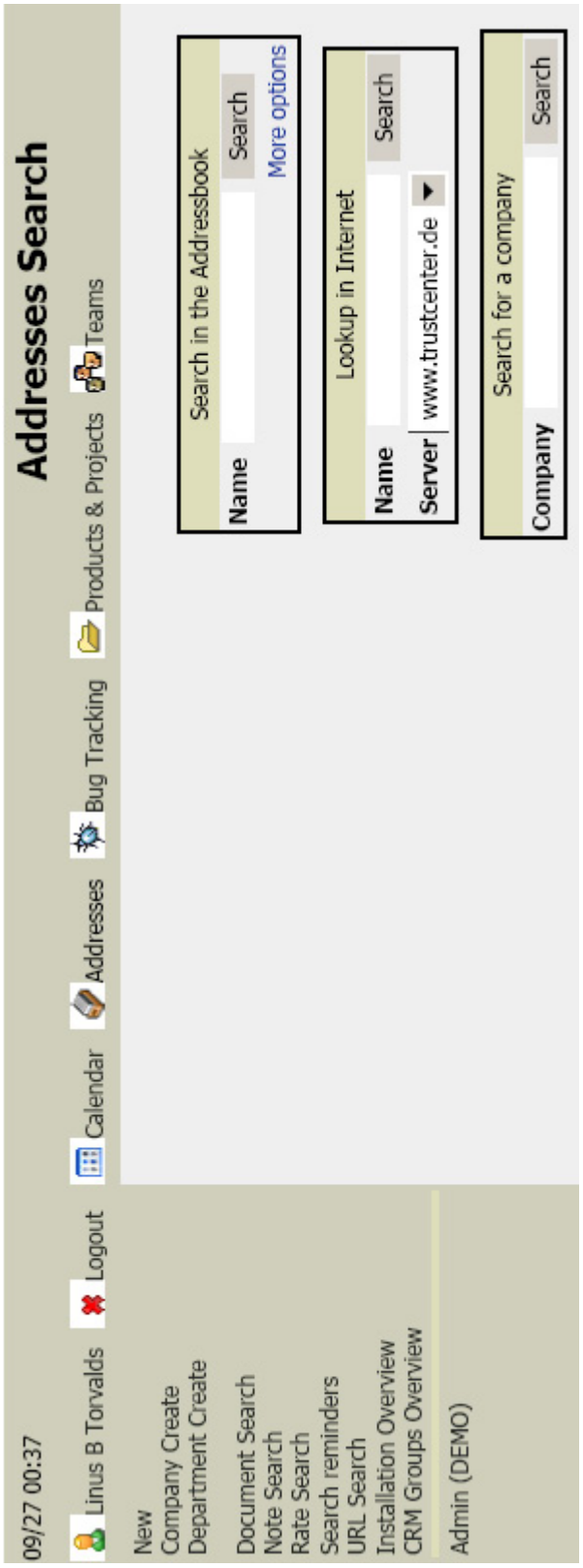

Figura C.1: TUTOS: Panel de visualización del directorio

Figura C.2: TUTOS: Panel de visualización general del directorio

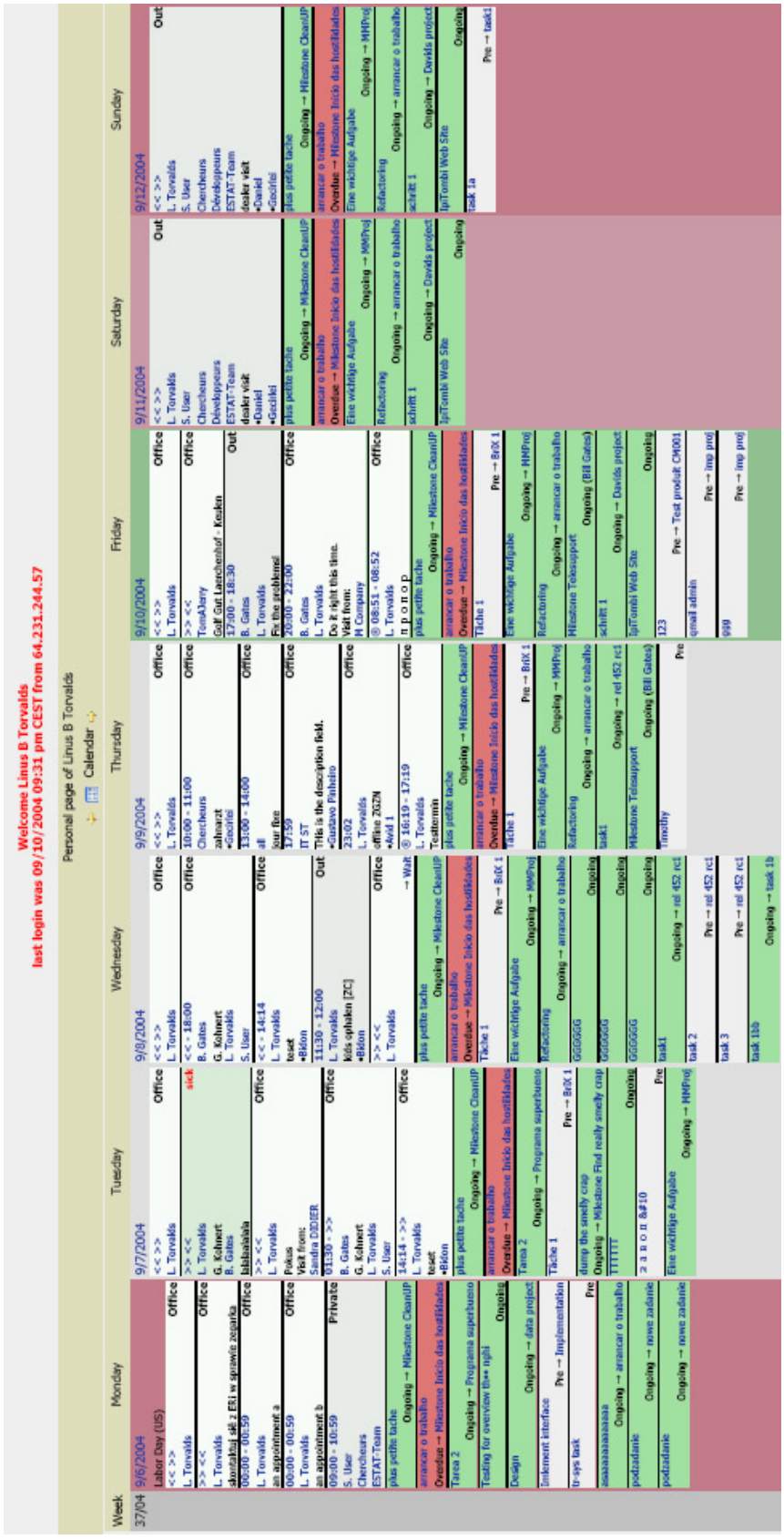

Figura C.4: TUTOS: Vista General del calendario del usuario

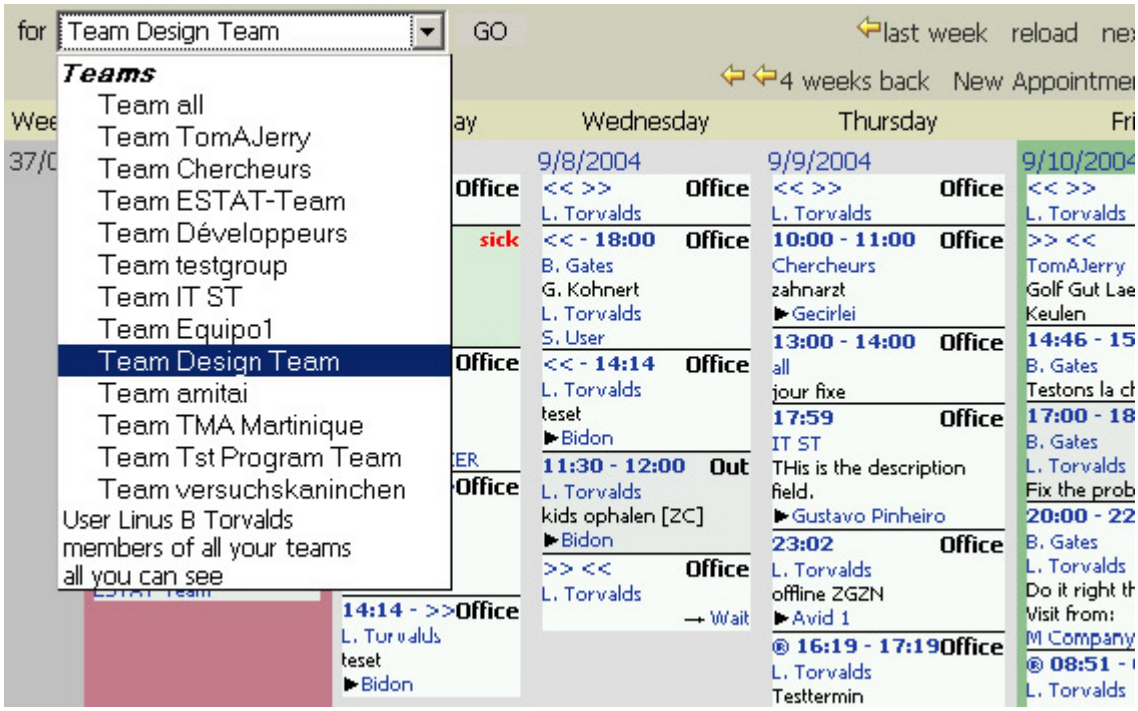

Figura C.5: TUTOS: Vista de filtros del calendario del usuario

C.1.3. Citas

La citas tienen como atributos obligatorios a sus participantes, y como atributos opcionales fecha y hora de comienzo y fin, lugar, recursos que se van a utilizar, como una sala de reuniones, el cliente (si esta presente), y el producto del cual trata la cita. Existen dos clases de citas, las puntuales, registradas para una sola fecha y las repetitivas, como por ejemplo, reunión semanal de los viernes, ver figura C.7 en la página 88. El sistema registra la historia de las mismas en caso de tener modificaciones. Tiene dos paneles de visualización general de citas, uno igual al calendario, pero sólo muestra citas, y otro que lista las citas del usuario como una lista ordenada por identificación de la misma, ver figura C.6 en la página siguiente.

ID	( )Start	End	®	Description	Location	Product/Project	Participants	☐
1	8/31/2004	9/10/2004			Office		Linus B Torvalds	☐
2	9/1/2004			Meeting	Out	TUTOS 1.2.20040901	Linus B Torvalds Gero Kohnert	☐
3	9/1/2004			Count my money	Office		Bill Gates	☐
4	9/2/2004				Office		Linus B Torvalds	☐
5	09/02/2004 10:11 am CEST			going to the chale and we're gonna get married	Out		Linus B Torvalds	☐
6	09/02/2004 01:00 pm CEST	09/02/2004 02:00 pm CEST		Comentar las herramientas CRM consultadas	Office		Linus B Torvalds	☐

Figura C.6: TUTOS: Listado de citas

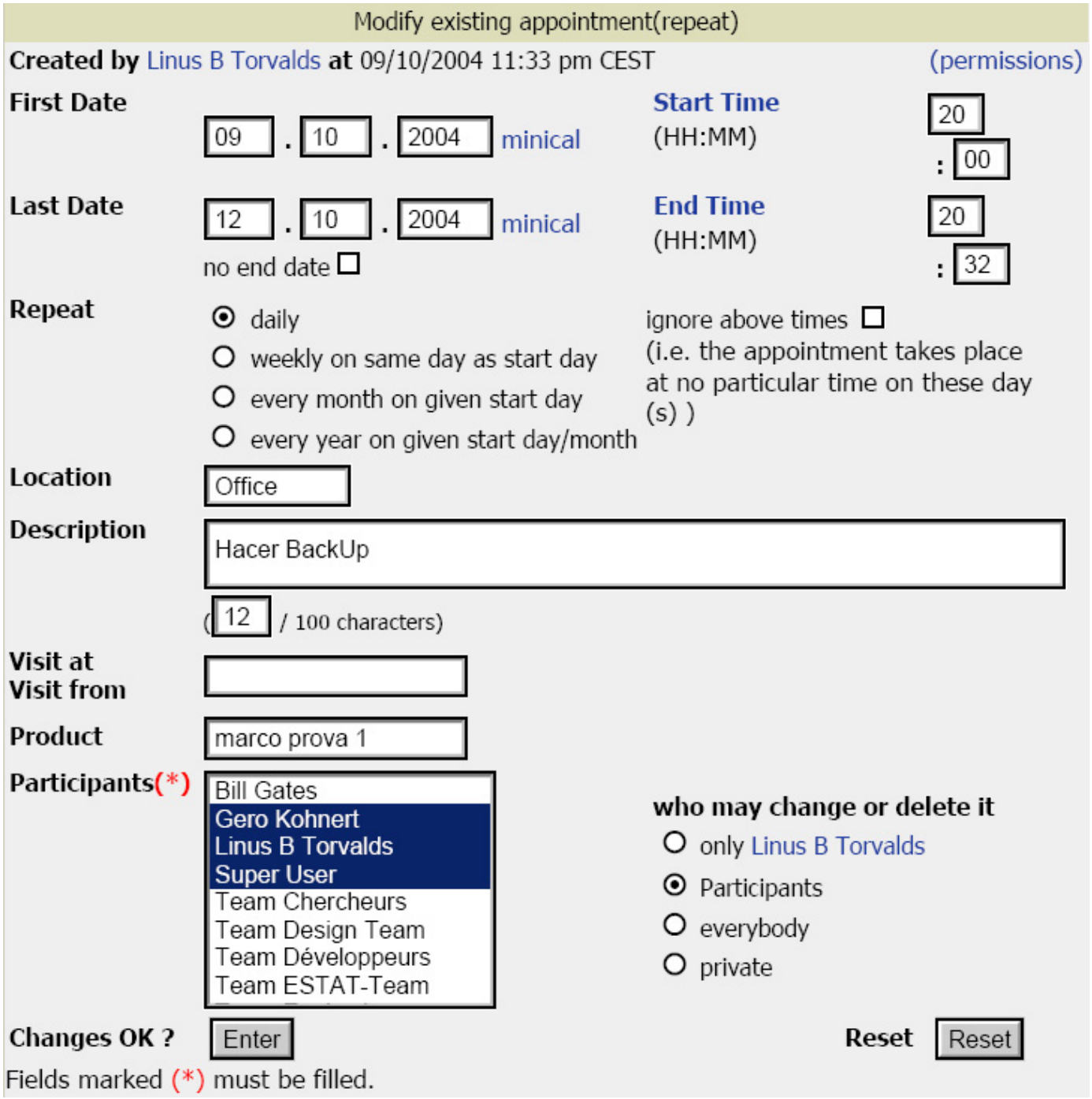

Figura C.7: TUTOS: Vista de panel de edición de tarea repetitiva

C.1.4. Recursos

El sistema administra recursos, estos son objetos simples con sólo dos atributos nombre y descripción. El la figura C.8 en la página siguiente, se puede ver el panel de visualización general de recursos.

Figura C.8: TUTOS: Panel de visualización general de recursos

C.1.5. Proyectos/Productos

El sistema maneja productos o proyectos indistintamente, y se identifican por su nombre y versión. Contienen un registro de errores, ver sección C.1.7 en la página 91, enlaces a sitios web, citas, notas, documentos, y tareas asociadas.

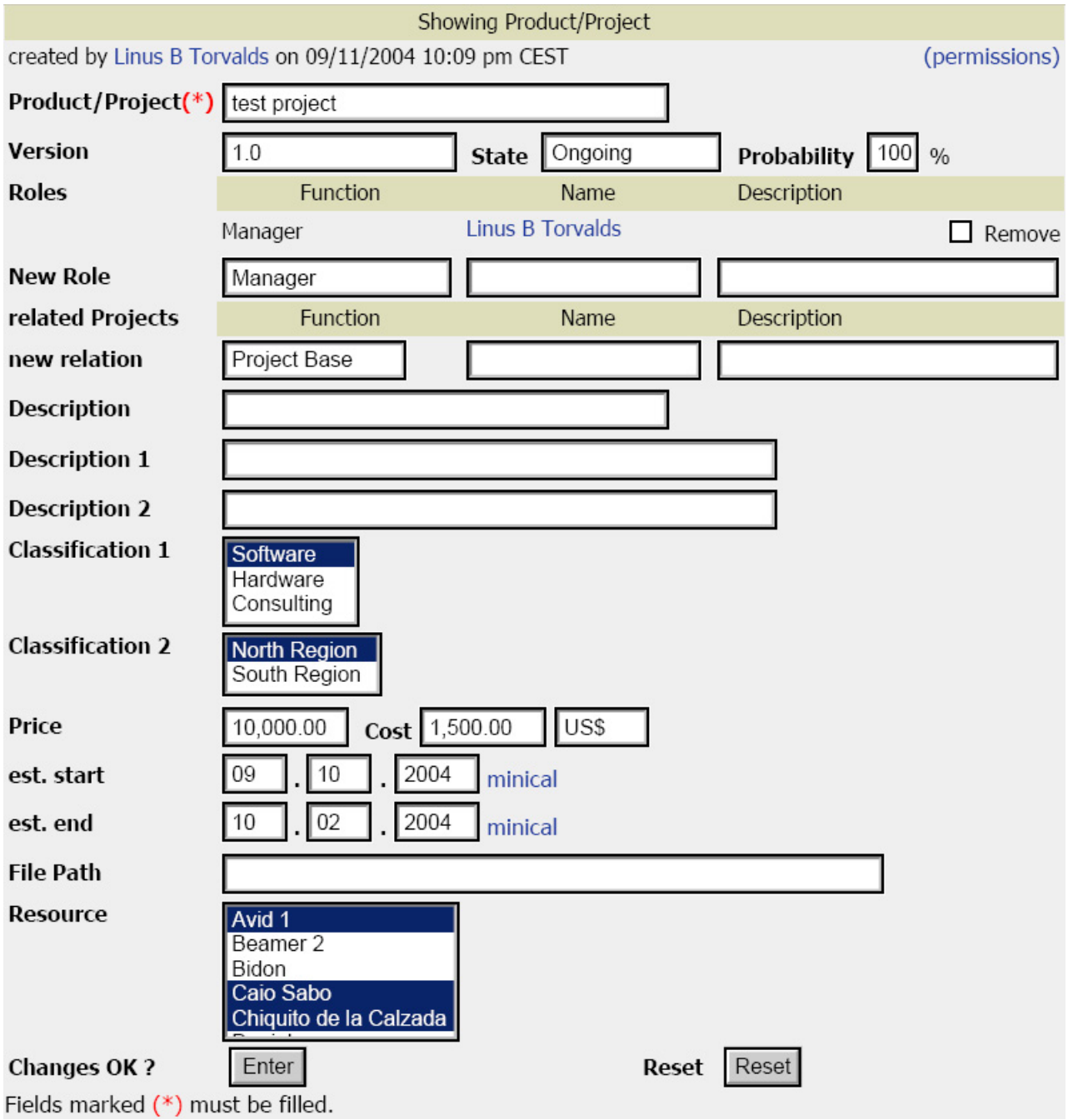

Figura C.9: TUTOS: Panel de edición de un proyecto/producto

C.1.6. Tareas

Tienen como atributos su nombre, estado, proyecto/producto al cual pertenecen, lista de ítems para hacer, tiempos agendados y reales, estimación del volumen de horas, subtareas, y porcentaje de completitud.

En la figura C.10 en la página 92, se muestra el panel de edición e ingreso de una tarea. En figura C.11 en la página 93, se muestra el panel de visualización de una tarea, que contiene además

de los datos del panel anterior, una barra de progreso de la tarea. Esta última indica con color verde el porcentaje completado, y en rojo el porcentaje a completar. En la figura C.12 en la página 94, se muestran todas las tareas del proyecto, junto con sus barras de progreso. Finalmente en la figura C.13 en la página 95, se muestra el diagrama de Gantt de un proyecto.

C.1.7. Errores

El sistema gestiona errores. Estos tienen una identificación, el producto/proyecto al cual pertenece, y el usuario asignado. Posee dos descripciones una corta para identificarlo, y una larga para los detalles del error. Tiene dos atributos para su clasificación: estado y clase. Los valores para estado son:

- Abierto: el error fue ingresado al sistema, pero no se ha comenzado a resolverlo.
- En curso: se está resolviendo.
- Resuelto: está resuelto pero sin archivar.
- Cerrado: no se va a realizar ninguna acción más, queda archivado.

Los valores para clase son:

- Liviano: de poca importancia.
- Pesado: muy importante.
- Peligroso.
- Sugerencia: pedido de mejora.

Cualquier modificación que se realice en el error es registrado en la historia del mismo.

En la figura C.14 en la página 96, se muestra el panel de visualización de un error.

C.1.8. Registros de tiempos

Se tienen registros de tiempos asociados al proyecto/producto. Sus atributos obligatorios son: el volumen de horas dedicadas, la persona quien las trabajo, la fecha del día correspondiente, costo por hora de la persona, monto y moneda, una descripción y el estado. Los valores posibles para el estado son: ningún estado, verificado, facturado (al cliente), pagado (por el cliente), no facturable. Se tiene un panel que muestra los porcentajes de dedicación de personas a un proyecto, ver figura C.15 en la página 96.

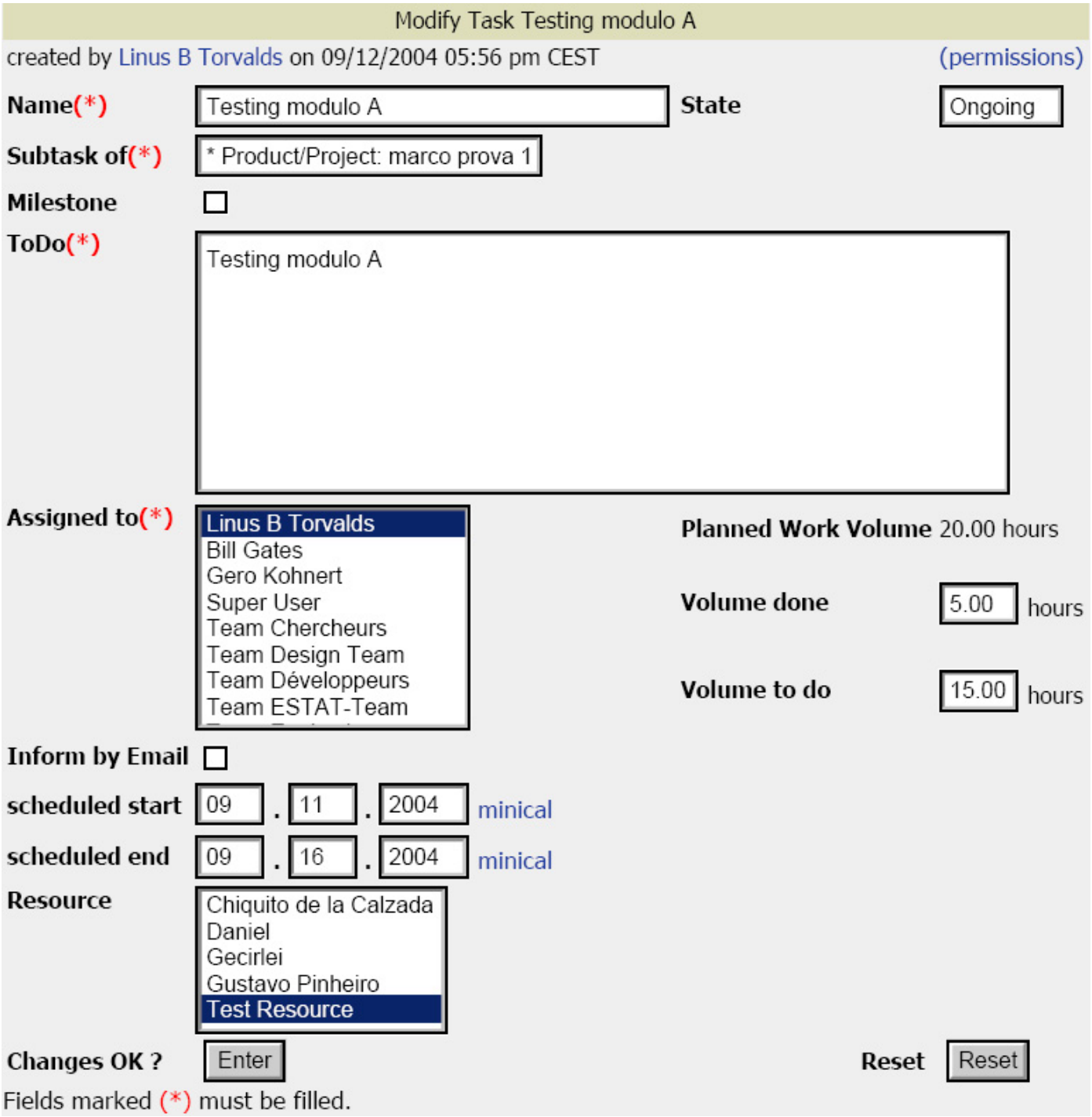

Figura C.10: TUTOS: Panel de ingreso y edición de tarea

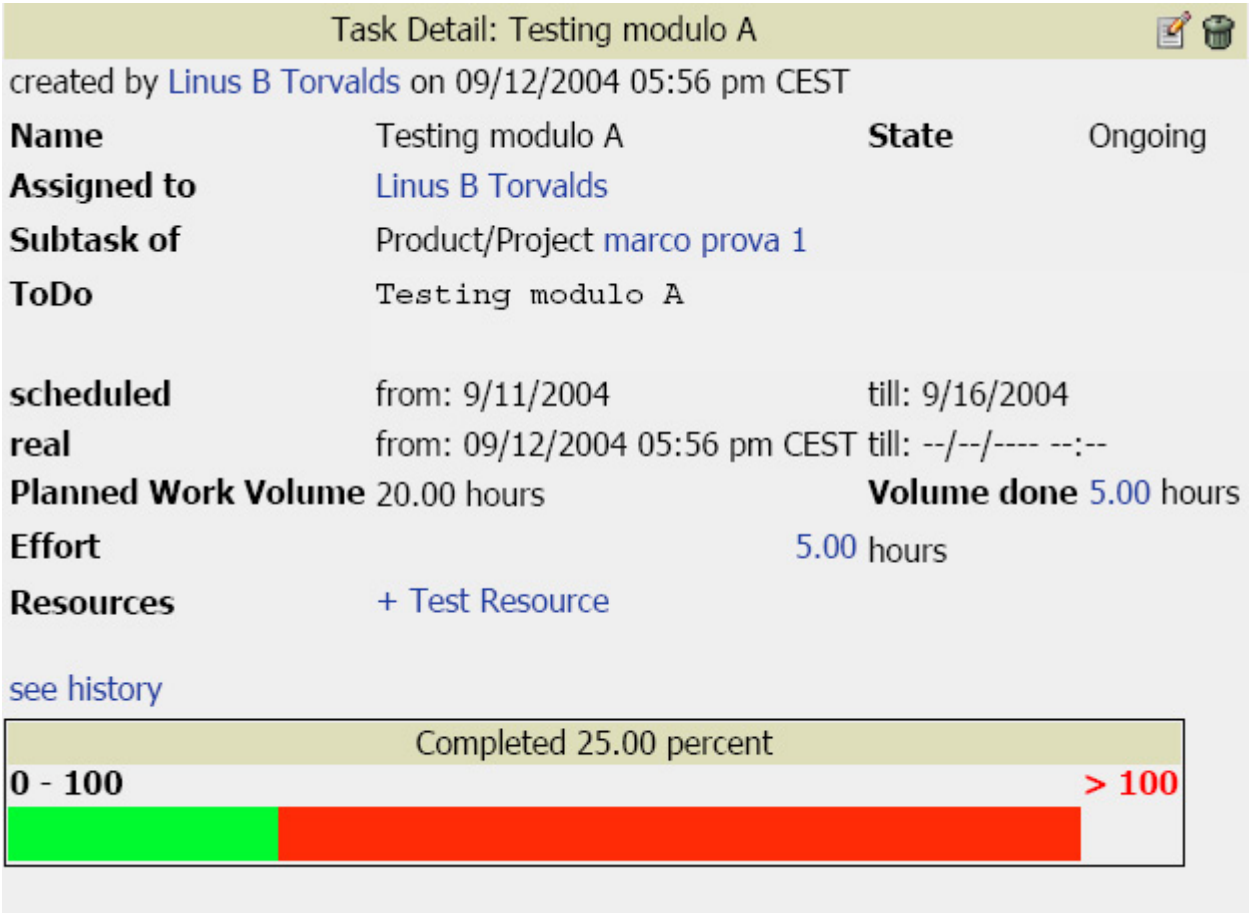

Figura C.11: TUTOS: Panel de visualización de tarea

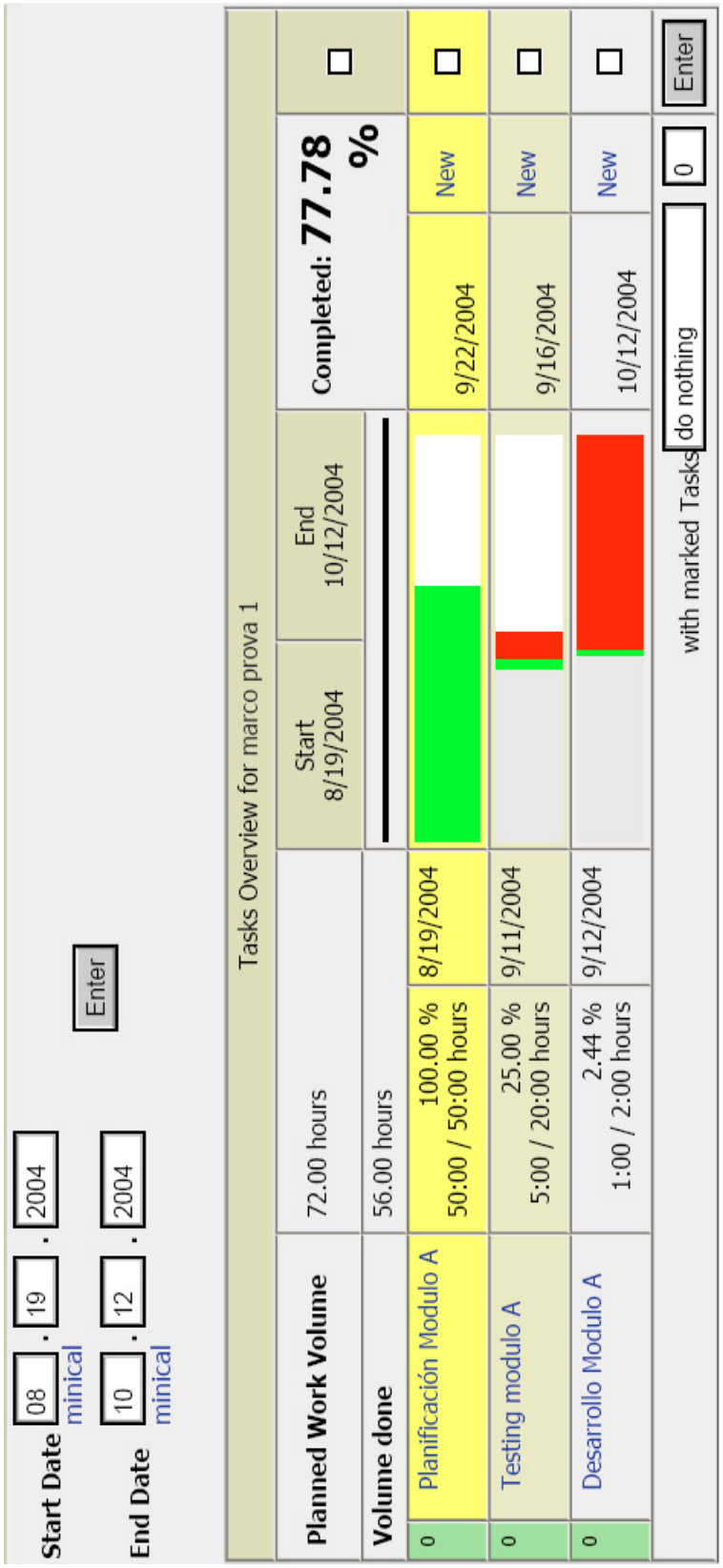

Figura C.12: TUTOS: Panel de visualización de tareas de un proyecto

Task [Assigned to]	Planned Work Volume	scheduled start	scheduled end
task2 [Linus B Torvalds]	**2:00**	**9/10/2004**	**9/15/2004**
task1 [Bill Gates]	**10:00**	**9/13/2004**	**9/22/2004**
task 1b [Linus B Torvalds]	10:00	9/12/2004	9/15/2004
task 1c [Linus B Torvalds]	10:00	9/12/2004	9/13/2004
Pietje Puk [Gero Kohnert]	8:00	9/12/2004	9/12/2004
task 1a [Linus B Torvalds]	5:00	9/17/2004	9/28/2004
task 1aa [Linus B Torvalds]	12:00	9/10/2004	9/10/2004
task3 [Linus B Torvalds]	**4:00**	**9/15/2004**	**9/20/2004**
test [Super User]	0:00	9/10/2004	9/10/2004

Figura C.13: TUTOS: Diagrama de Gantt

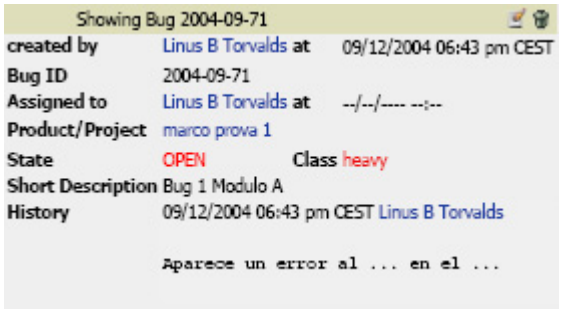

Figura C.14: TUTOS: Panel de visualización de un error

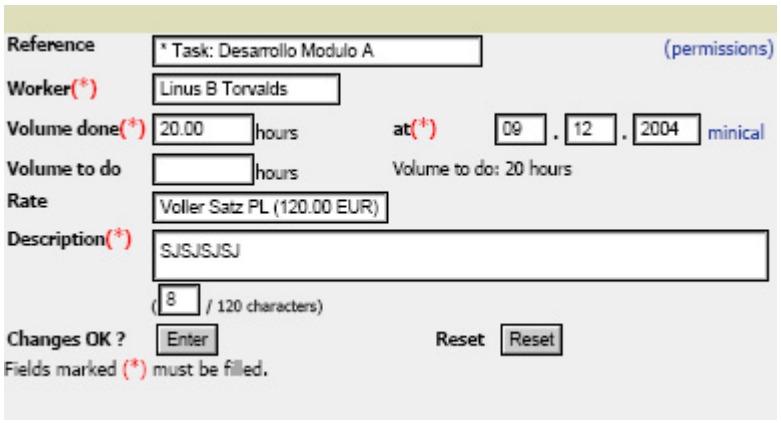

Figura C.15: TUTOS: Panel de ingreso y modificación de ingreso de tiempo

C.1.9. Notas

Como se ve en los diferentes paneles, el sistema permite agregar notas a las compañías, proyectos, contactos, errores, tareas, citas, documentos, y recursos. Y estas son mostradas y se pueden acceder desde las correspondientes interfaces de visualización. Sus atributos requeridos son título y descripción. El sistema registra a su creador, hora de creación y referencia. Se puede indicar un recordatorio de la nota en el calendario. En la figura C.16 en la página siguiente, se muestra el panel de visualización de una nota.

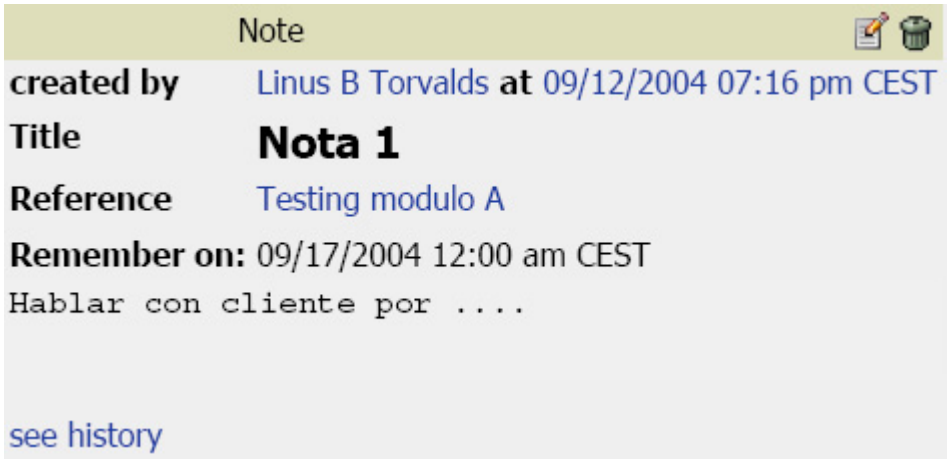

Figura C.16: TUTOS: Panel de visualización de una nota

C.1.10. Documentos

El sistema permite asociar documentos existentes a contactos, compañías, proyectos/productos, o errores. Su único atributo requerido es el nombre y los opcionales son versión, referencia a producto/proyecto, autor, nombre original, sistema de versionamiento, y su acceso, que puede ser total o restringido por el usuario que la creó. El sistema registra la fecha de creación del documento en TUTOS [1. TUTOS], su tamaño en bytes, su tipo, por ejemplo "application/ms word", y lleva un registro de su historia. En la figura C.17, se muestra el panel de visualización de un documento.

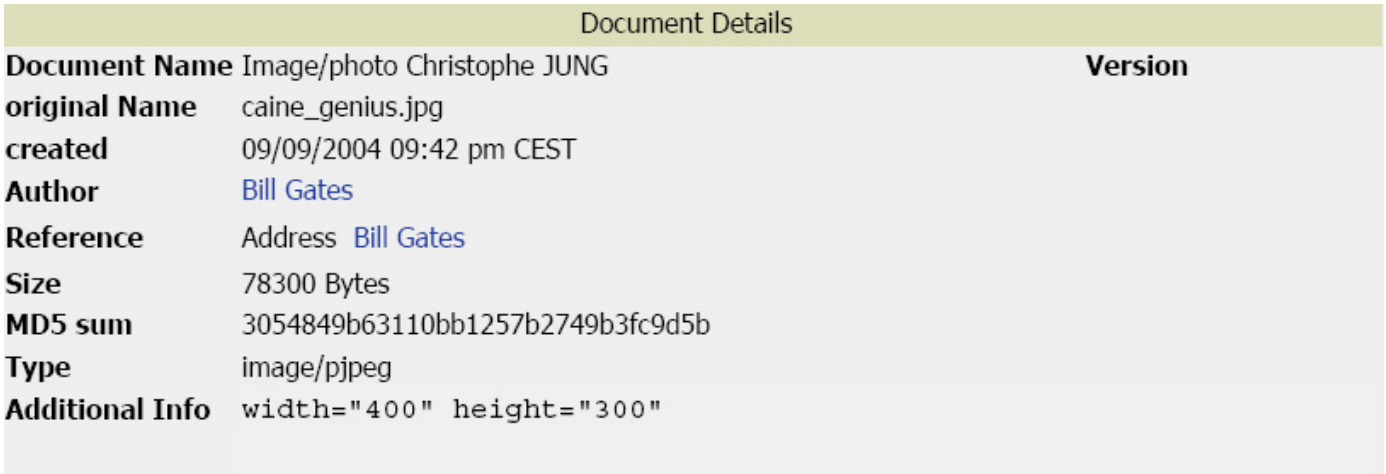

Figura C.17: TUTOS: Panel de visualización de un documento

C.1.11. Estadísticas

TUTOS genera cuatro reportes con estadísticas de las tareas de todos los proyectos:

- Cantidad de tareas por estado: Este reporte muestra el porcentaje y la cantidad de tareas en los distintos estados, ver figura C.18.
- Cantidad de tareas según clasificación 1: Este reporte muestra el porcentaje y la cantidad de tareas en los elementos de la clasificación 1, ver figura C.19.
- Cantidad de tareas según clasificación 2: Ídem anterior, pero de la clasificación 2, ver figura C.20 en la página siguiente.
- Cantidad de tareas asignadas por gerente: Muestra el porcentaje y la cantidad de tareas asignadas a los gerentes, ver figura C.21 en la página siguiente.

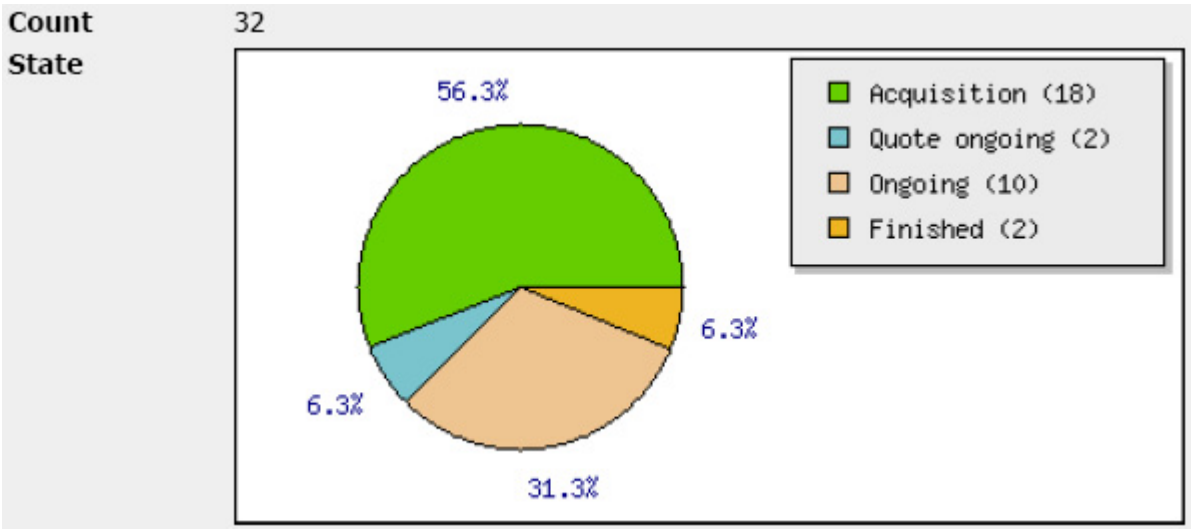

Figura C.18: TUTOS: Tareas por Estado

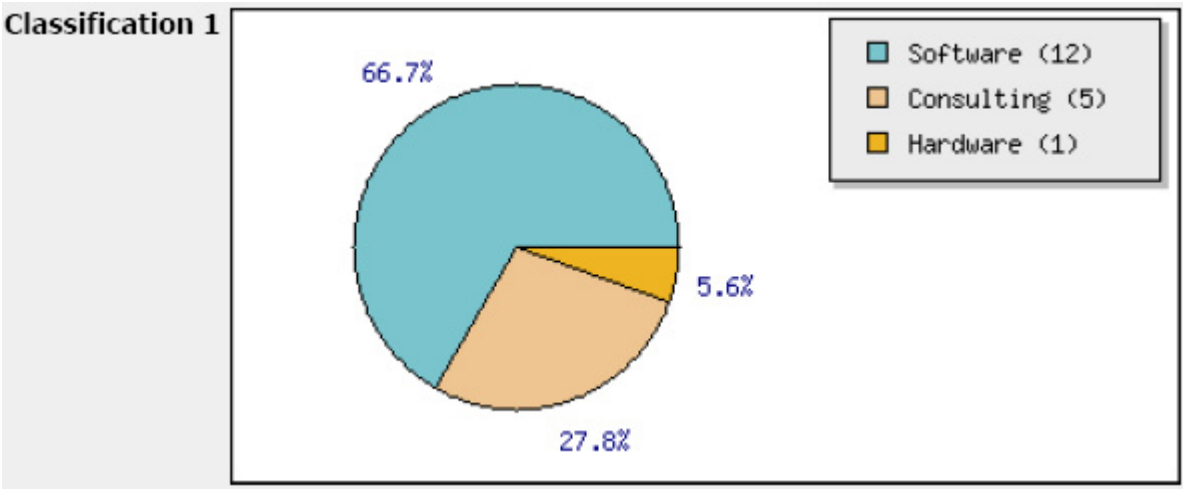

Figura C.19: TUTOS: Tareas por clasificación 1

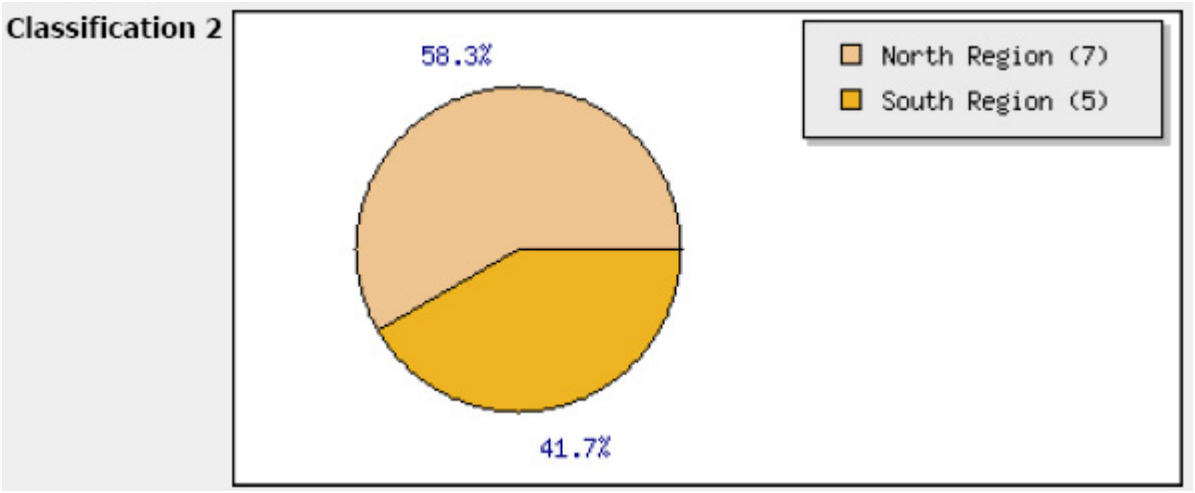

Figura C.20: TUTOS: Tareas por clasificación 2

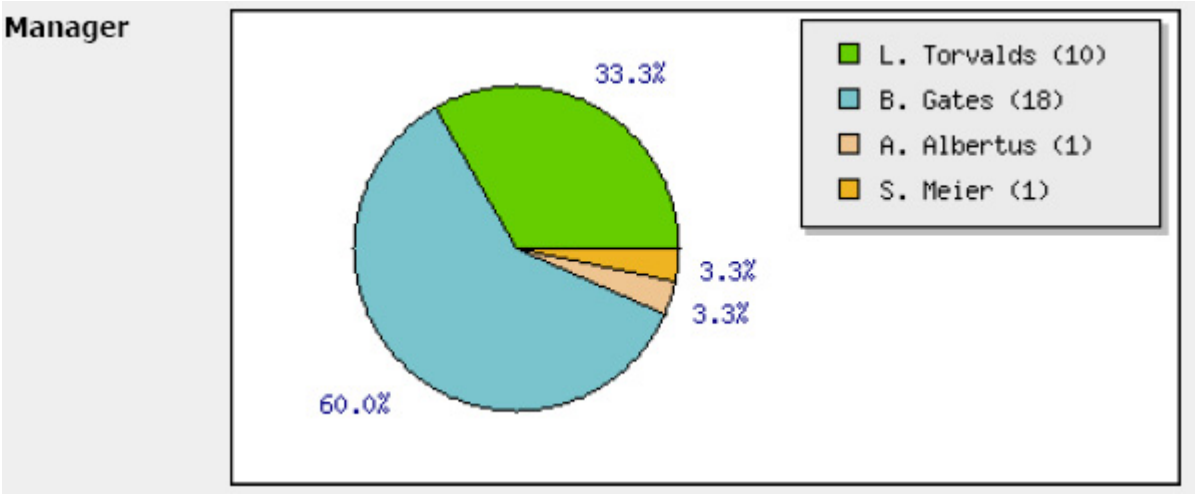

Figura C.21: TUTOS: Tareas por gerente

C.2. XPlanner

XPlanner

Es un proyecto de Fuente Abierta[5] para web, de planificación y seguimiento de proyectos para equipos de desarrollo ágil, actualmente XP [3. XP]. Esta implementado usando Java [3. Java], JSP [3. JSP], Struts [3. Struts], Hibernate [3. Hibernate] y MySQL [3. MySQL]. Esta herramienta fue evaluada en su versión 0.6.2.

El sistema, siguiendo las líneas de desarrollo XP [3. XP], se basa en iteraciones [3. XP Iteraciones], ver figura C.22. Estas iteraciones implementan requerimientos del cliente, que son llamados Historias [3. XP Historias]. Cada historia es seleccionada según la importancia dada por el cliente y el tiempo estimado de construcción por los desarrolladores. Para la construcción de historias se generan tareas.

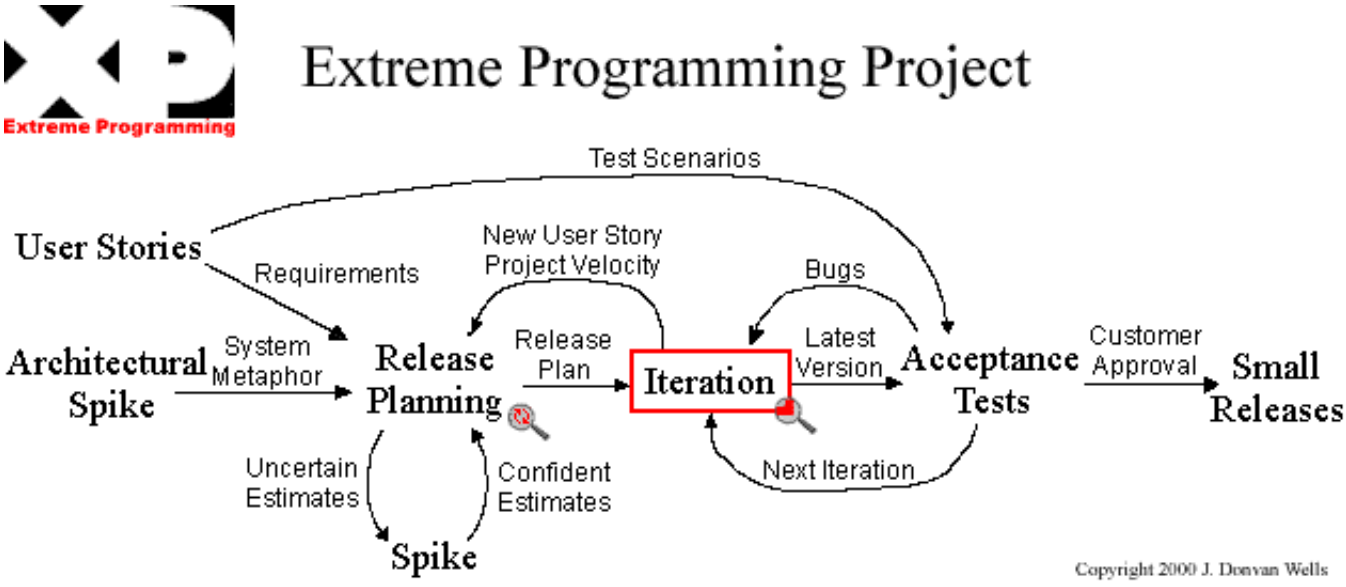

Figura C.22: Proyecto XP

C.2.1. Iteraciones

Cada proyecto se divide en iteraciones [3. XP Iteraciones], estas se muestran en la figura C.24. En este, ver figura C.23, se pueden ver todas las historias que contiene, ordenadas por prioridad; junto con su progreso, y tiempo estimado de construcción.

[5]Software de Fuente Abierta o Código Abierto, del inglés Open Source Software, se refiere al software cuyo código está disponible públicamente, aunque los términos de licenciamiento específicos varían respecto a sus permisos de utilización.

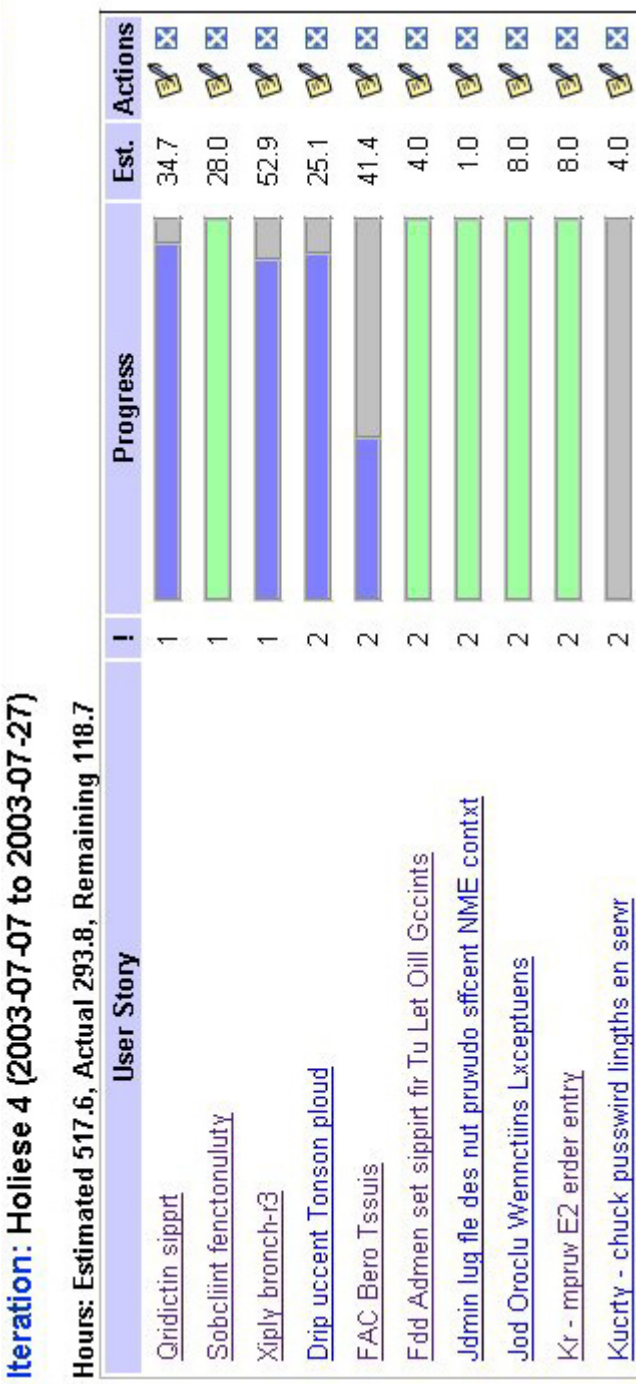

Figura C.23: XPlanner: Visualización de una iteración

En la figura C.24 en la página siguiente se muestra el diagrama de flujo de una iteración XP [3. XP].

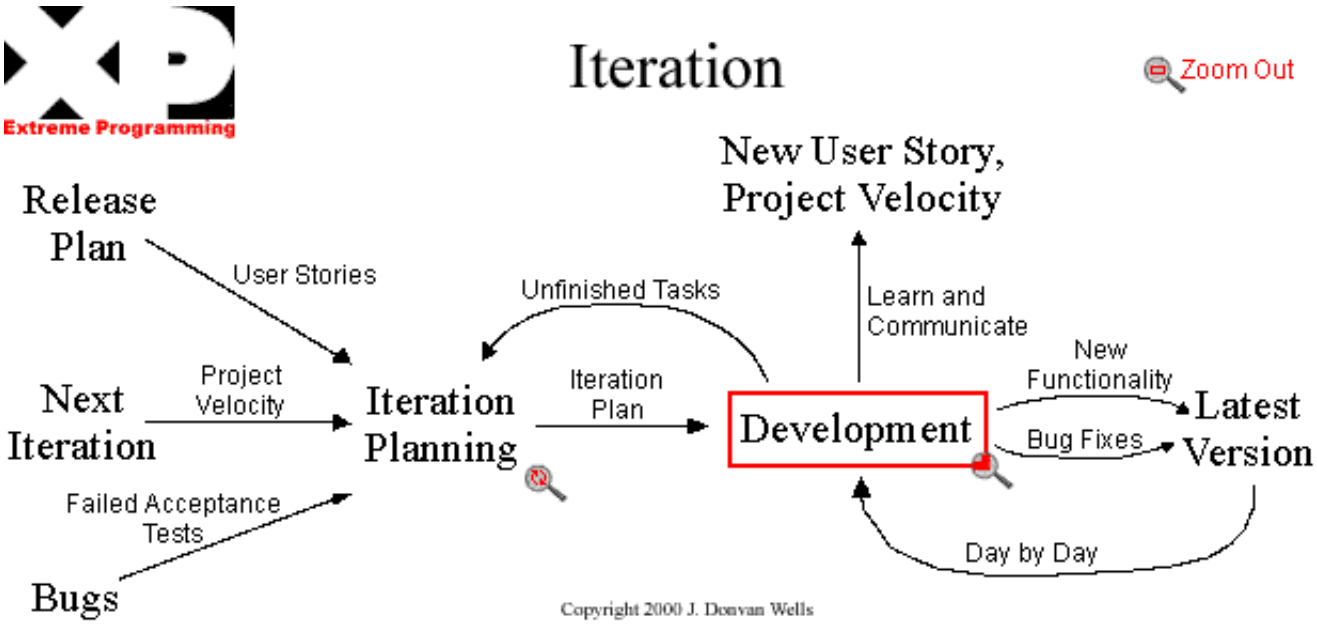

Figura C.24: Diagrama de flujo de una iteración XP

C.2.2. Historias

La historias son creadas desde el panel de la iteración a la cual pertenece.

En el panel de visualización global de una historia , ver figura C.25 en la página siguiente, se muestra el nombre de la historia, barra de progreso, los requerimientos a implementar (features), enlaces a documentos asociados, su prioridad, las horas de construcción estimadas, y las dedicadas. Además contiene una tabla con todas las tareas asociadas. Cada línea de la tabla representa una tarea, y muestra su nombre, tipo, barra de progreso, cantidad de horas estimadas para su construcción, estado y notas.

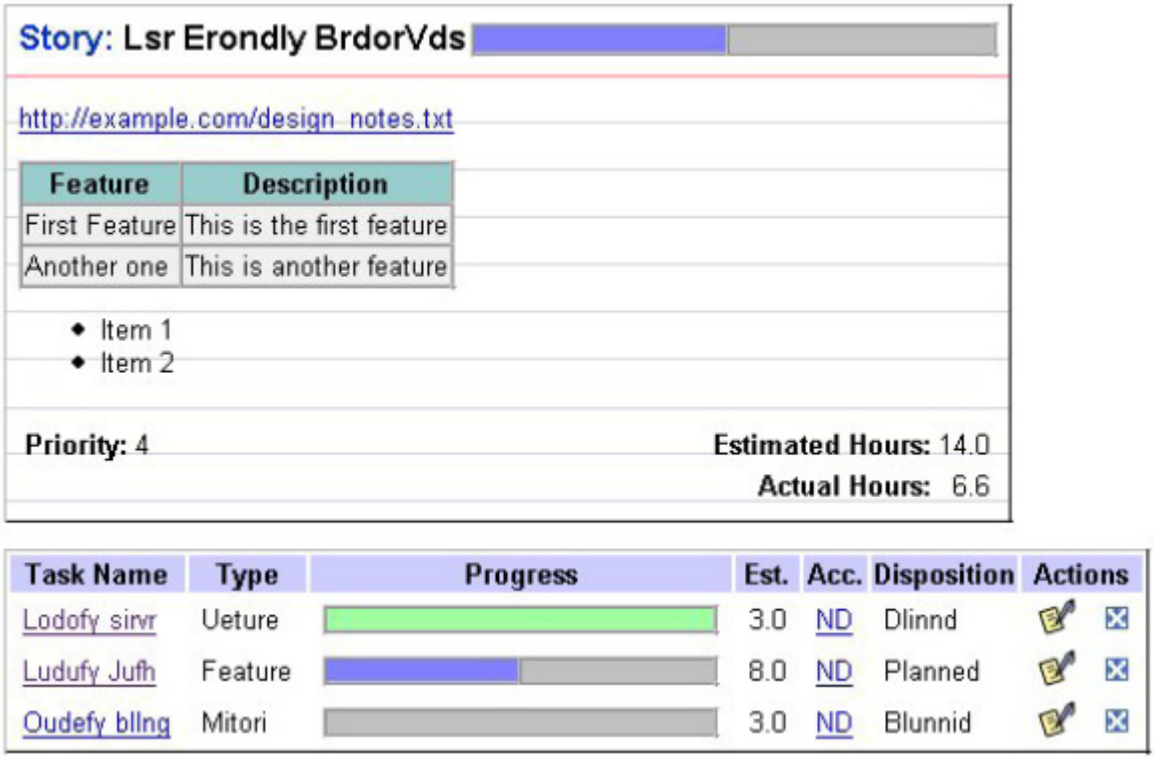

Figura C.25: XPlanner: Panel de visualización global de una historia

C.2.3. Tareas

Las historias se dividen en tareas. Estas son mostradas en un panel de visualización de tarea, ver figura C.26 en la página siguiente, que muestra su nombre, la barra de progreso, los registros de tiempos de dedicación y las notas asociadas. Las notas son mostradas con el tema que trata, su autor y fecha de creación.

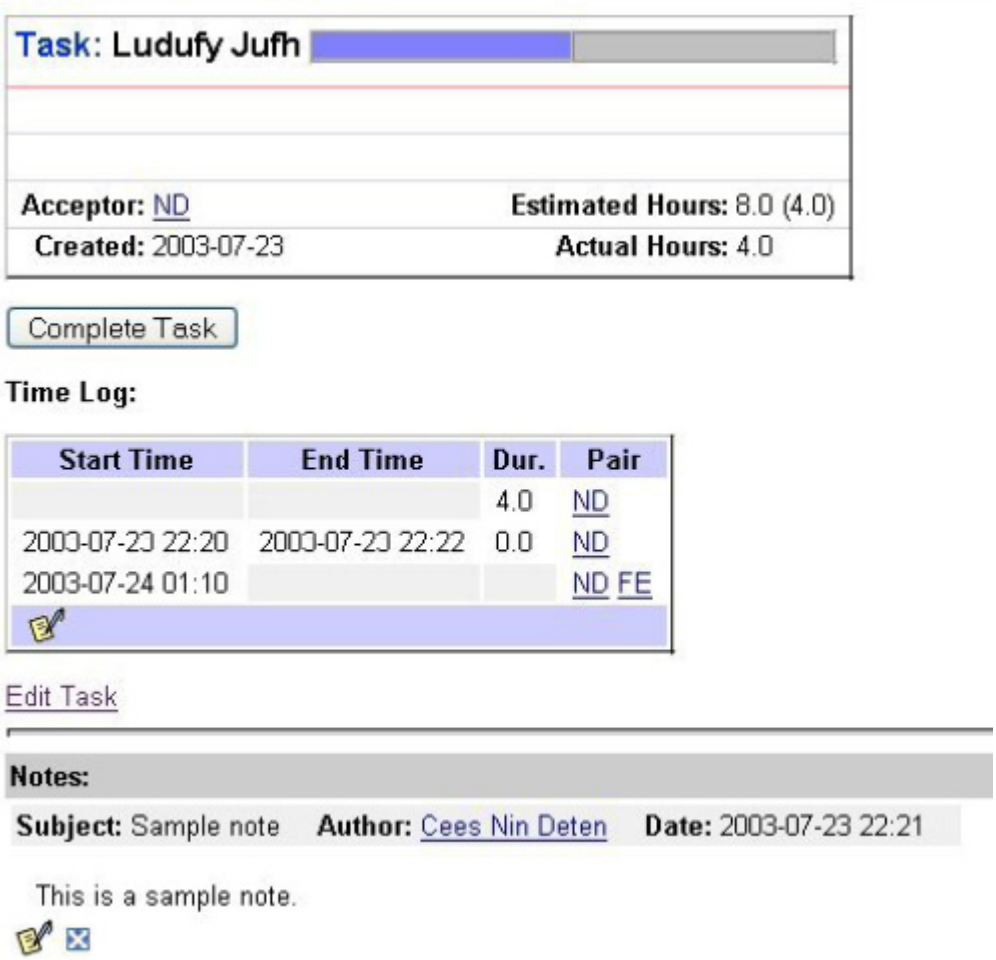

Figura C.26: XPlanner: Panel de visualización de una Tarea

C.2.4. Métricas de Iteraciones

El sistema genera métricas que se muestran en un panel de visualización de métricas, ver figura C.27 en la página siguiente. Este panel muestra la cantidad de horas dedicadas y asignadas por el desarrollador en una iteración.

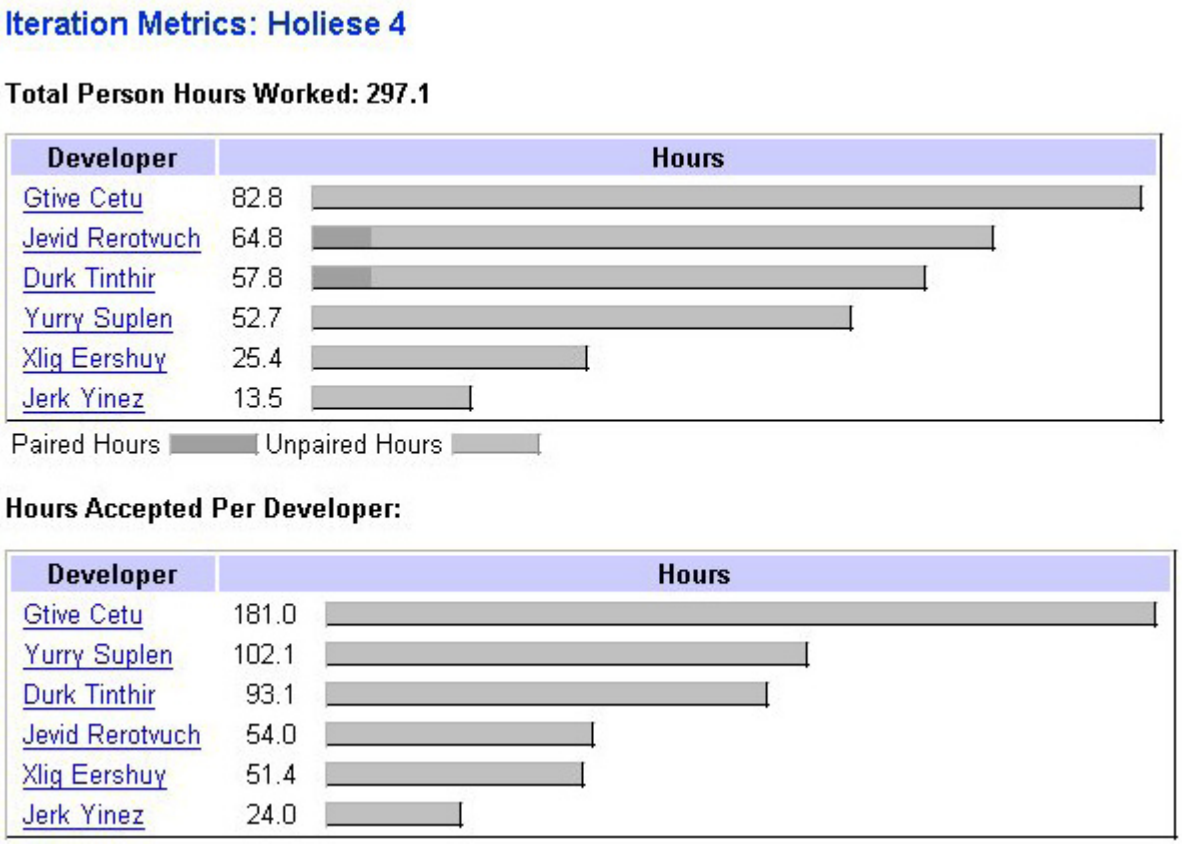

Figura C.27: XPlanner: Panel de visualización de métricas de iteraciones

C.2.5. Estadísticas de las iteraciones

El sistema genera estadísticas por iteración. En el panel de visualización de estadísticas por iteración, ver figuras C.28 en la página siguiente y C.29 en la página 107, muestra:

Velocidad del proyecto: Compara el esfuerzo estimado originalmente requerido con el completado.

- Todas las tareas por tipo
- Todas las tareas por estado
- Horas estimadas completadas por tipo
- Horas estimadas por tipo

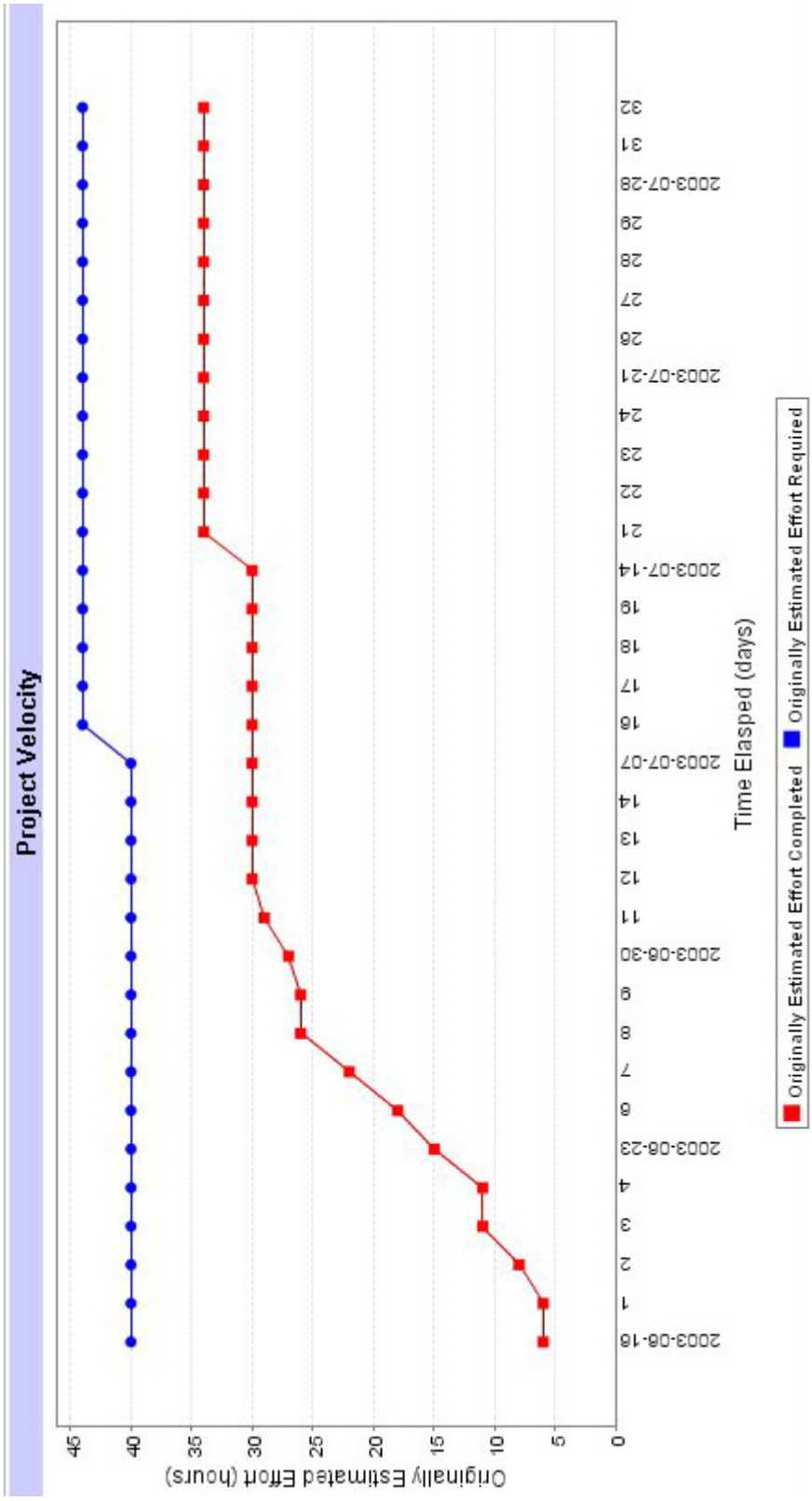

Figura C.28: XPlanner: Panel de visualización de estadísticas de una iteración

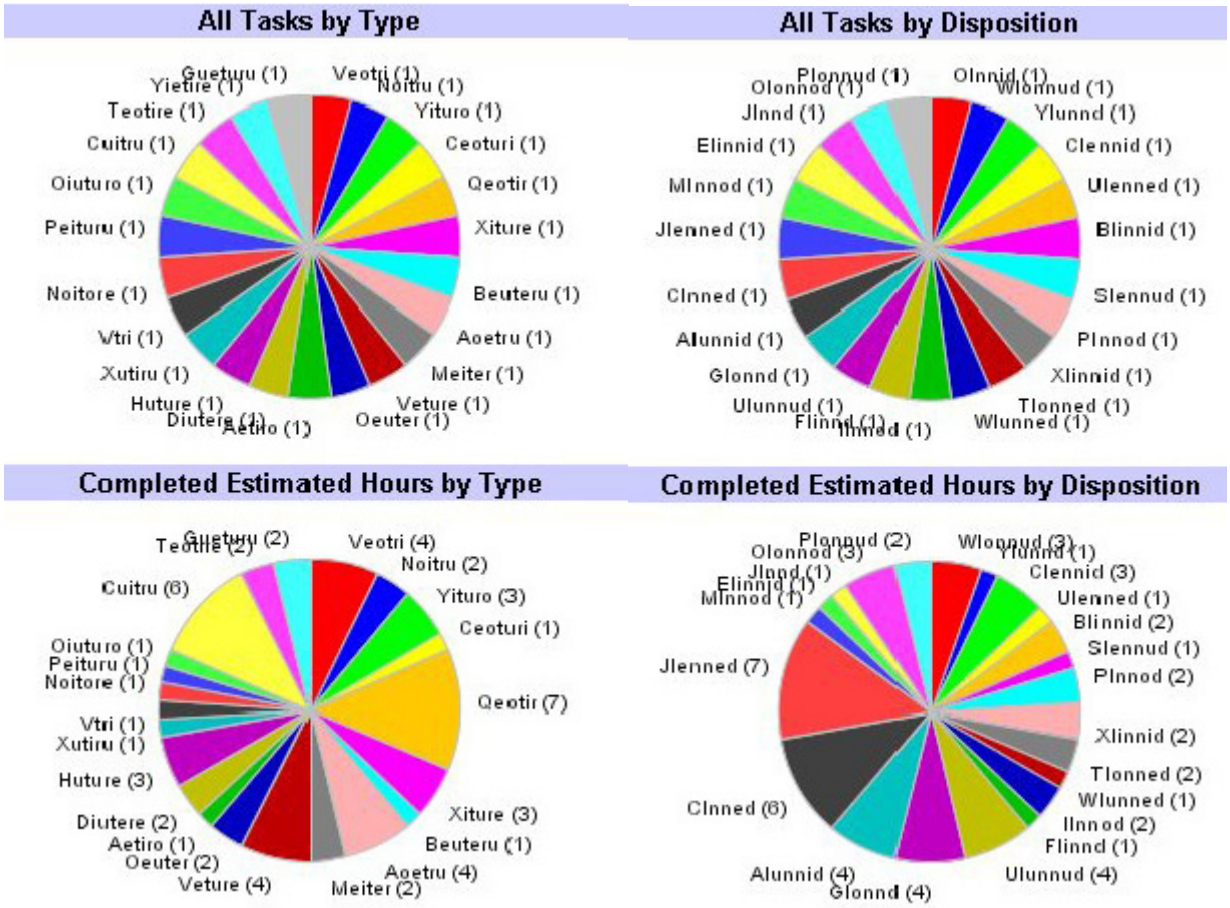

Figura C.29: XPlanner: Panel de visualización de estadísticas de una iteración (cont.)

C.2.6. Estado del desarrollador

En este panel, ver figura C.30 en la página siguiente, se muestra el nombre del desarrollador, y los datos de contacto: teléfono y correo electrónico. Además contiene las siguientes tablas:

- Tareas que se están desarrollando actualmente
- Tareas asignadas pendientes
- Tareas terminadas

Todas estas tareas se muestran con la historia a la cual pertenecen.

Name: Durk Tinthir

Contact Info:

Email: morkp@example.com
Phone: 214-555-1212

Tasks in progress:

Story	Task	Acceptor?
LHSL Xhungis 28-AMU-03	Kmplmont Loqoerid BUTW chingus	Yes
Xiply bronch-r3	Oitubse Wegrutn scrpts	Yes
Qridictin sipprt	GQ2 KHJG Yroblom	Yes
Lsr Erondly BrdorVds	Ludufy Jufh	Yes

Unstarted Tasks:

Story	Task
Yurkut duto billung	Kunureto fud bollung rpirt
Xiply bronch-r3	Apley to pridction
Nrder midoficotoen (prt 2)	Yelleng, OXHRF nd MWDM chengus
Lsr Erondly BrdorVds	Oudefy bllng

Closed Tasks:

Story	Task
Qridictin sipprt	Jrdur Vourch woth RWF/NAUF ruutos selictd roturns nithing
Yurkut duto billung	Brevde blling doto fer murket feds

Figura C.30: XPlanner: Panel de visualización estado del desarrollador

C.2.7. Entrega de software e integración

Este panel, ver figura C.31 en la página siguiente, muestra la integración actual, la lista de espera de integraciones, y las integraciones recientes.

Los datos de la integración actual son la persona que la está realizando, la fecha de comienzo y los módulos que esta integrando. Es desde este panel, que se da por finalizada dicha integración.

La lista de espera para integrar contiene: la persona que quiere integrar, la fecha en la cual ingreso a la lista, y que es lo que va a integrar. Se tiene la opción de salir o unirse a la lista.

Finalmente se muestra una lista de las integraciones recientes, que muestra el nombre de la persona que la realizó, fecha de comienzo y fin, duración, estado y que fue lo que integró.

Figura C.31: XPlanner: Panel de integración

C.3. FDDTracker

FDDTracker es una herramienta comercial, desarrollada por ITPS [3. ITPS], para gestión de proyectos de software que siguen la metodología FDD [3. FDD]. En la figura C.32, se muestra el ciclo de desarrollo FDD.

En su versión 1.0, tiene tres ediciones estándar, profesional, y empresarial. Además existe una versión reducida con licencia gratuita. La versión evaluada, es el producto demostrativo, equivalente a la edición empresarial de la versión 1.0.

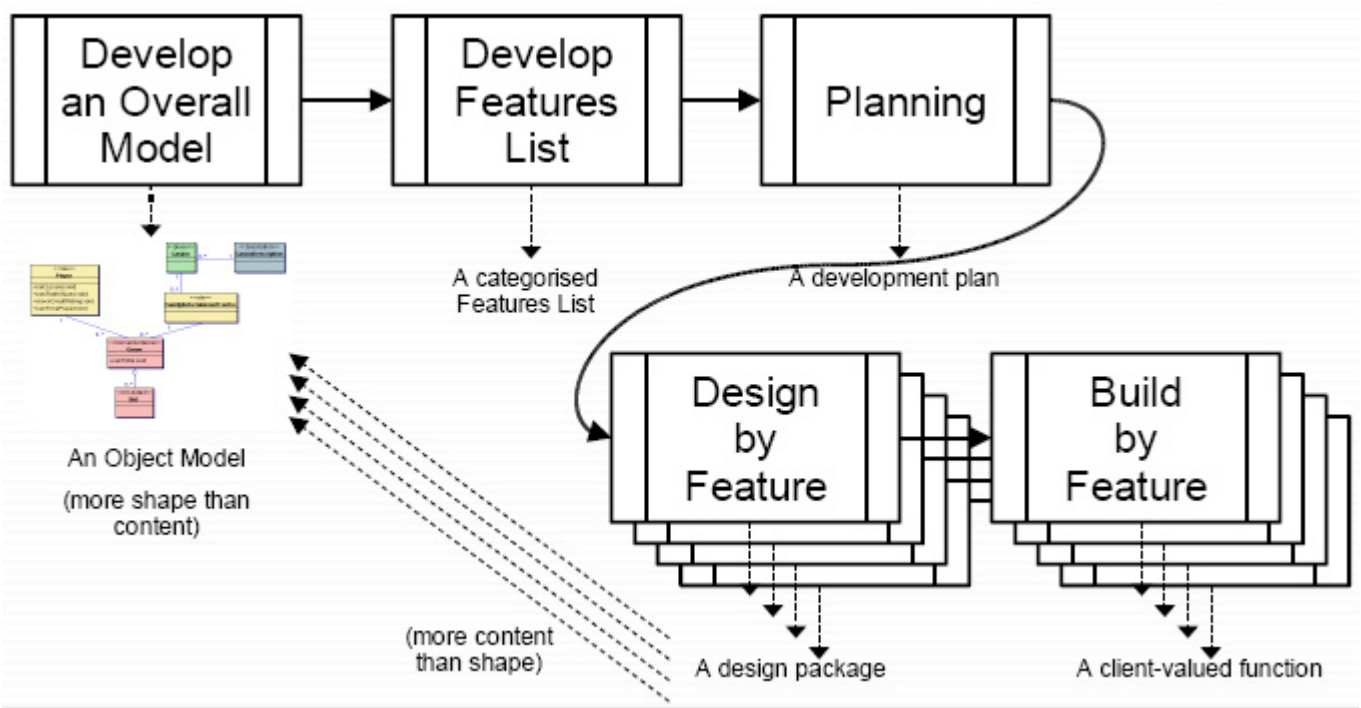

Figura C.32: Ciclo de desarrollo FDD

Las funcionalidades del producto se dividen en cuatro módulos: proyecto, reportes, mantenimiento y administración. El acceso a cada una de estas funcionalidades es por compañía o proyecto. En la figura C.33 en la página siguiente se muestra el panel de selección de la compañía o producto que será gestionado.

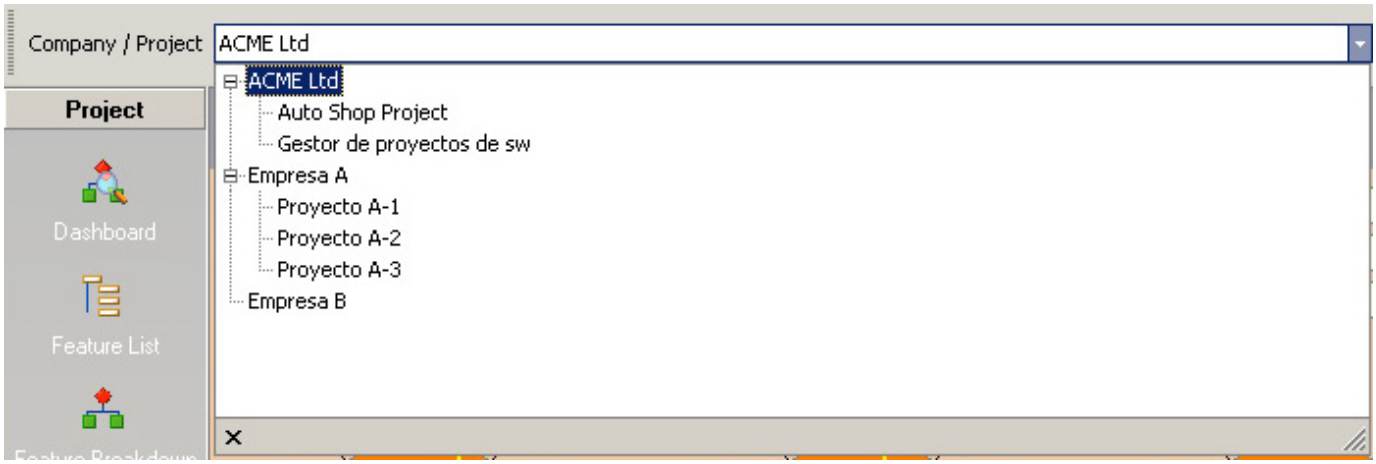

Figura C.33: FDDTracker: Panel de selección de compañía o proyecto a gestionar

C.3.1. Administración

En este módulo se gestionan las compañías, sus departamentos, los usuarios, y sus permisos.

Los usuarios, además de por su nombre, se identifican por el departamento, y la compañía a la cual pertenecen. Los permisos son estáticos y asignados a nivel de usuario. En la figura C.34 se muestran las relaciones y los atributos de estos objetos.

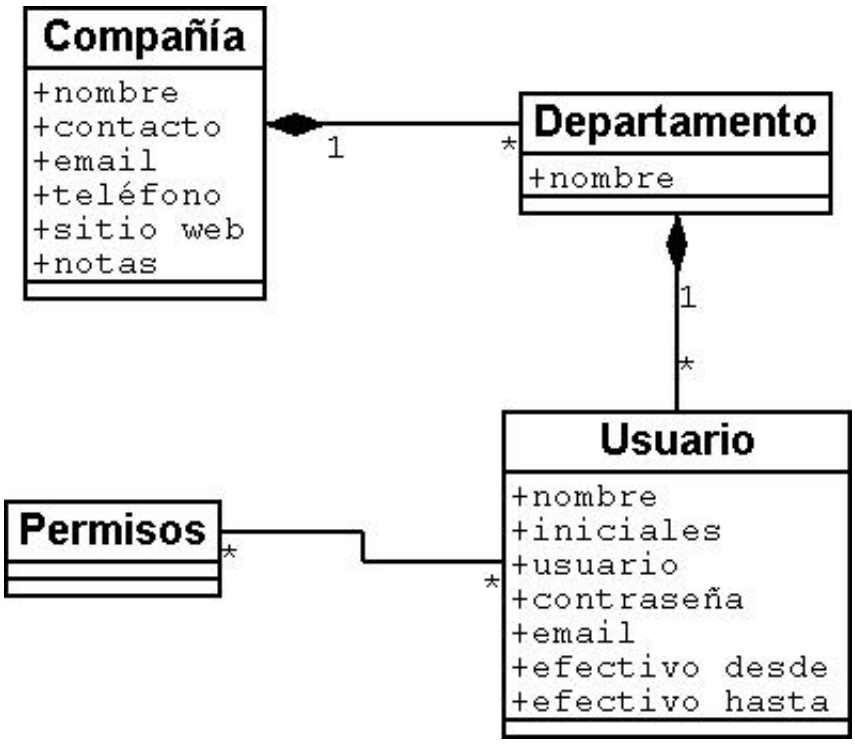

Figura C.34: FDDTracker: Relaciones de objetos del módulo de administración

C.3.2. Mantenimiento

En este módulo se gestionan los proyectos, sus equipos, áreas temáticas (major feature set, ver sección 3 de [3. JMCUML]), y actividades de negocio (feature set, ver sección 3 de [3. JMCUML]).

C.3.2.1. Proyectos

Los proyectos están asociados a una empresa y pueden clasificarse como internos o externos. Tienen como atributos, ver figura C.36 en la página siguiente, un identificador, un nombre, fecha de comienzo y entrega, el monto del costo, una descripción general y otra del caso de uso asociado. En la figura C.35, se muestra el panel de visualización de proyectos, donde se muestran el o los proyectos de la empresa seleccionada en el panel de selección de empresa/proyecto, figura C.33 en la página anterior. Y en la figura C.36, se muestra el panel de ingreso y edición de un proyecto.

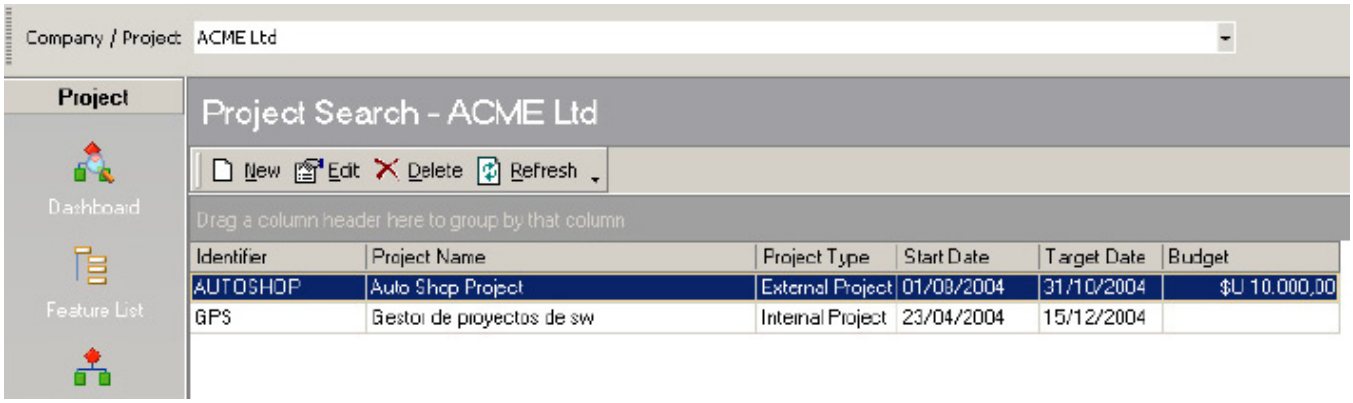

Figura C.35: FDDTracker: Panel de visualización de proyectos

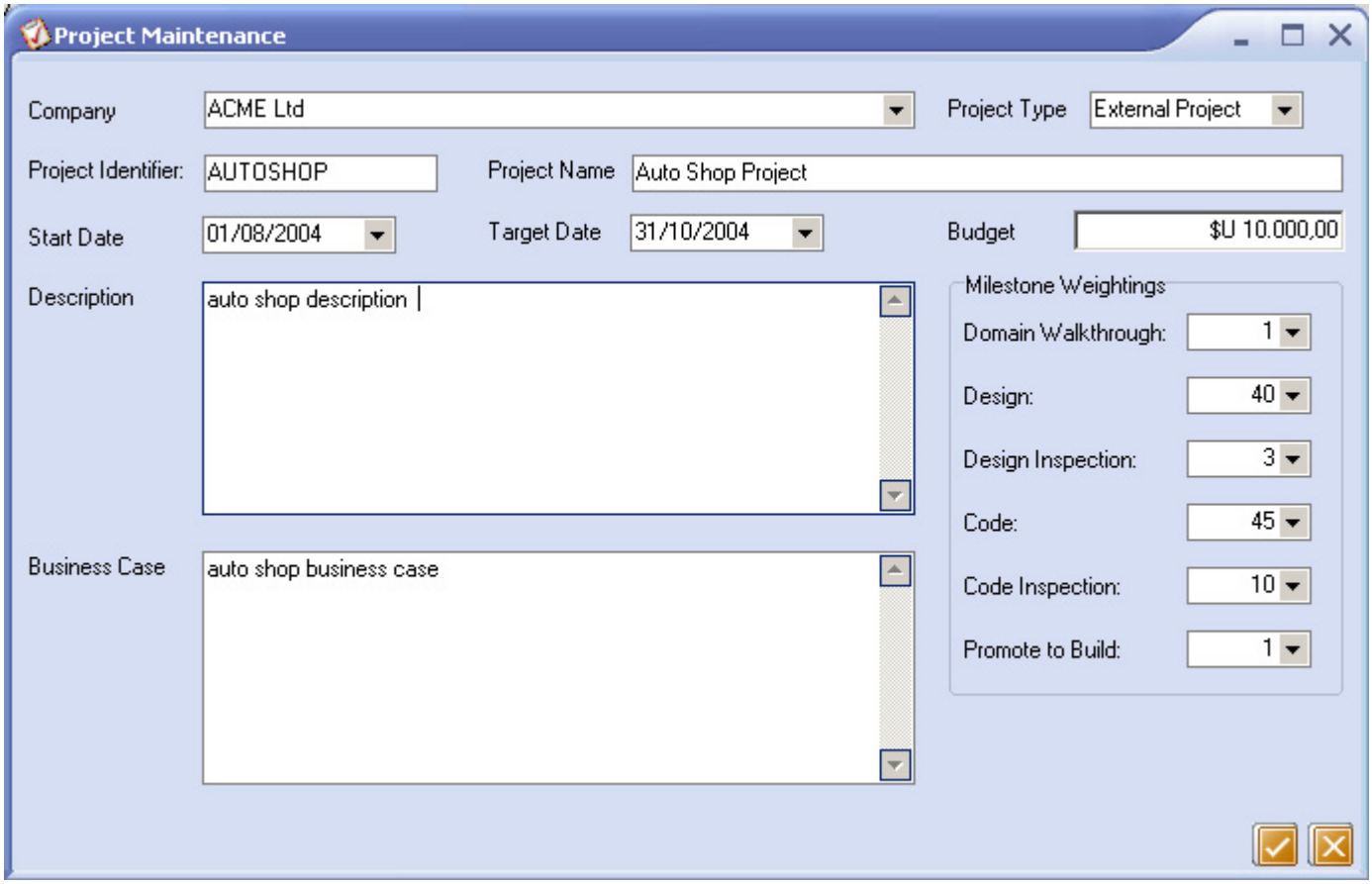

Figura C.36: FDDTracker: Panel de ingreso y modificación de un proyecto

C.3.2.2. Equipos

Los equipos se arman con los usuarios a los cuales se les asigna un rol en el mismo. Dichos roles son los roles FDD [3. FDD]: sponsor de proyecto, gestor de proyecto, jefe de arquitectos, gestor de desarrollo, jefe de programadores, dueño de clase, experto de dominio, y gestor de dominio. Además, se establece el período de disponibilidad en el equipo.

C.3.2.3. Área temática (Subject Area)

Las áreas temáticas (major feature set, ver sección 3 de [3. JMCUML]) son la primer subdivisión del proyecto y debe darse de alta al menos una. Luego se ingresan las actividades de negocio, para finalmente poder ingresar los rasgos a implementar en el proyecto. Estas siguen el patrón de área temática de FDD [3. FDD]: gestión de <objeto>, del inglés, <object> management; por ejemplo, gestión de ventas.

C.3.2.4. Actividad de negocio

Las actividades de negocio (feature set, ver sección 3 de [3. JMCUML]) son la segunda subdivisión de un proyecto. Estas siguen el patrón de actividad de negocio FDD [3. FDD]: <acción> <-ando/-endo> un <objeto>, del inglés, <action><-ing> a(n) <object>; por ejemplo, facturando un servicio.

C.3.3. Proyecto

C.3.3.1. Lista de rasgos

La lista de rasgos se presenta en un panel, ver figura C.37, agrupada por proyectos y actividades de negocio. Este muestra las fechas planificadas y reales de cada ciclo del rasgo (ver figura 6-7 de [3. JMCUML]).

Project Name | Business Activity

				Walkthrough		Design		Design
Project Name	Identifier	Feature Name	Chief Prog.	Planned	Actual	Planned	Actual	Planne
Project Name : Auto Shop Project								
Business Activity : SCHEDULESERVICE (Estimate completion: 27/08/2004, 7 Features)								
Auto Shop Project	SCHEDUL	Schedule a service for a car	glc	02/08	02/08	03/08	03/08	04/08
Auto Shop Project	SCHEDUL	Edit customer detail for a Customer List	glc	09/08	09/08	10/08	10/08	11/08
Auto Shop Project	SCHEDUL	Edit the Service Schedule for a car	glc	16/08	16/08	16/08	16/08	16/08
Auto Shop Project	SCHEDUL	Edit a Service Schedule for a car model	glc	16/08	16/08	16/08	16/08	16/08
Auto Shop Project	SCHEDUL	Edit the tasks list of a service schedule	glc	20/08	20/08	20/08	20/08	20/08
Auto Shop Project	SCHEDUL	Edit the parts lists of a service description		22/10		22/10		24/10
Auto Shop Project	SCHEDUL	Reserve the list of part for a service		22/10		22/08		
Project Name : Gestor de proyectos de sw								
Business Activity : Estado_arte (1 Features)								

Figura C.37: FDDTracker: Lista de rasgos

C.3.3.2. Tablero

El tablero, ver figura C.38 en la página siguiente, muestra la cantidad total de áreas temáticas, actividades de negocio, y rasgos. Junto con gráficas con los porcentajes de trabajo completado, y rasgos atrasados, en desarrollo, completados, y no comenzados aún. En los cuatro últimos paneles también muestra la cantidad de rasgos correspondientes.

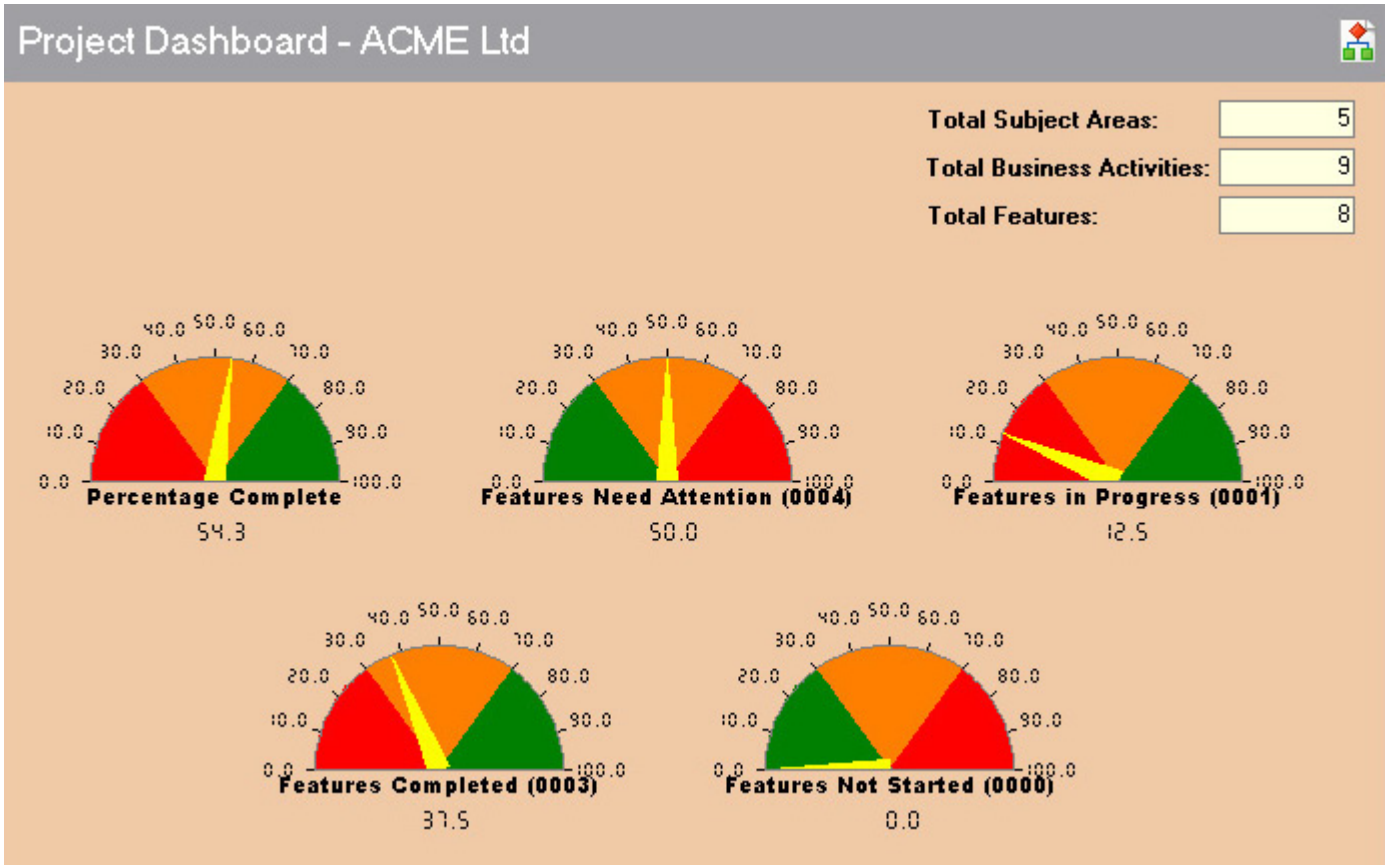

Figura C.38: FDDTracker: Tablero

C.3.3.3. Árbol de rasgos (Feature Breakdown Structure)

El árbol de rasgos muestra jerárquicamente los proyectos y sus componentes. La raíz es un nodo vacío, cuyos hijos son los proyectos, luego siguen las área temáticas, las actividades de negocio, y los rasgos que son las hojas del árbol.

C.3.3.4. Inspecciones

El módulo de proyectos también gestiona las inspecciones. Para las cuales se registra su fecha, tiempos, ubicación, rasgo que se inspeccionó, el tipo y sus resultados, ver figura C.39 en la página siguiente.

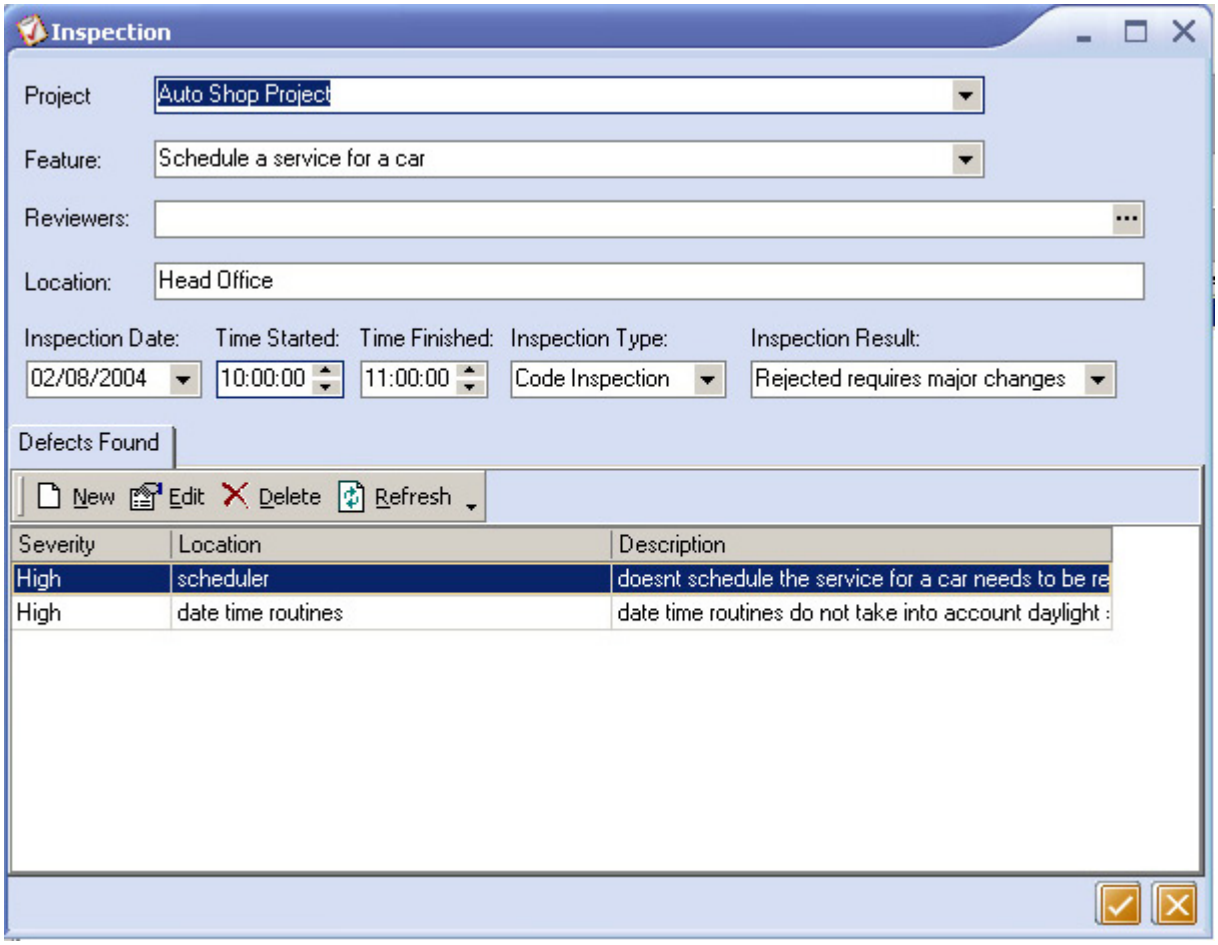

Figura C.39: FDDTracker: Inspección

C.3.3.5. Notas

Se puede agregar notas a los proyectos, la cuales tiene una fecha, un autor (usuario), un tipo que puede ser privado o público, y el detalle de la nota, ver figura en la página siguiente.

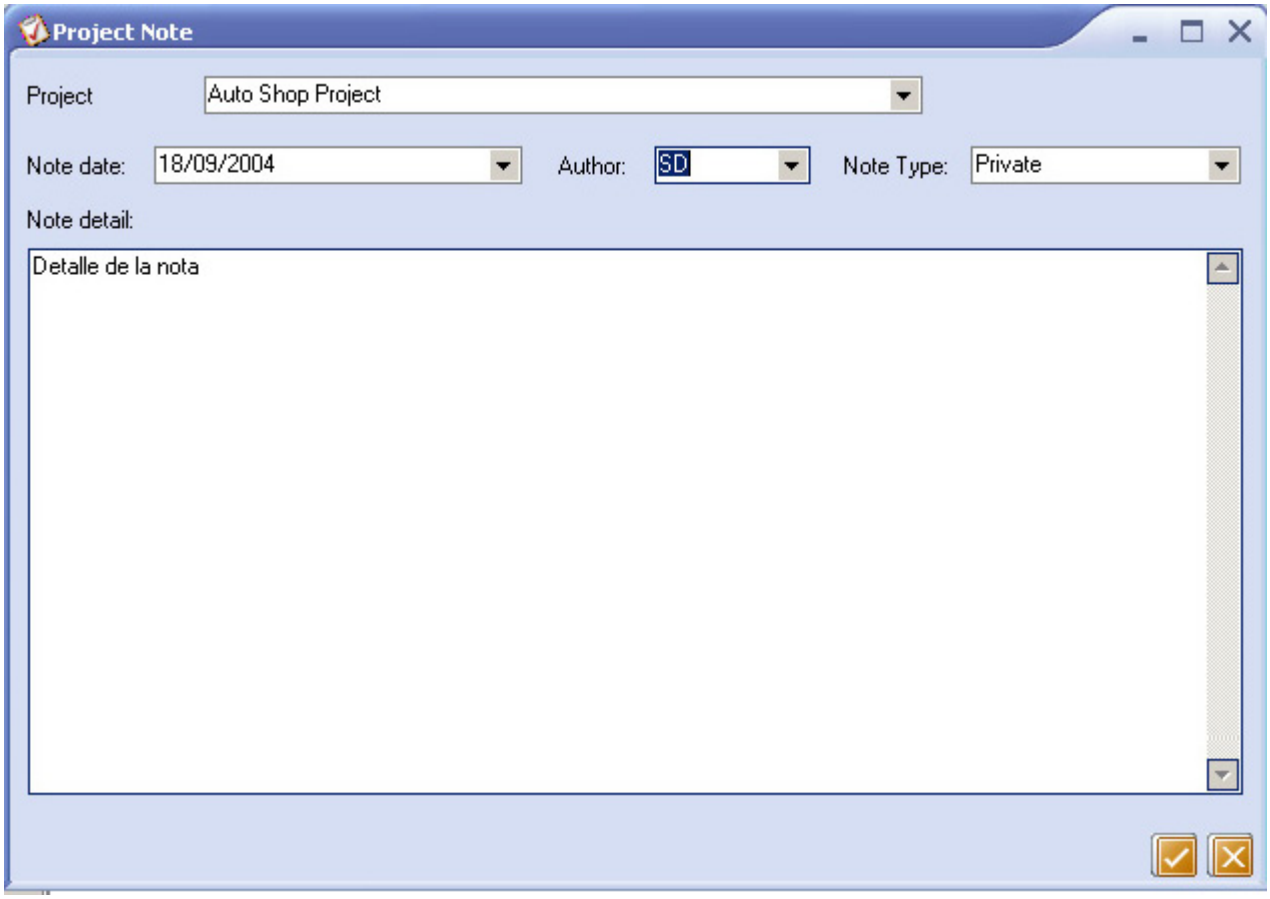

Figura C.40: FDDTracker: Nota

C.3.4. Reportes

Existen tres clases de reportes: diagrama de estacionamiento, visualización del plan, gráfico de tendencias, y de defectos. En la versión de evaluación no están disponibles estas funcionalidades, las figuras está incompletas, o fueron sacadas de las impresiones de pantalla[6] de la página web de FDDTracker [1. FDDTracker].

C.3.4.1. Diagrama de estacionamiento

Este diagrama muestra las actividades de negocio, agrupadas por área temática. De cada actividad de negocio se muestra el mes de terminación planificado, su porcentaje de completitud, la cantidad de rasgos que posee, las iniciales del jefe de programadores encargado y esta pintados con el color de su estado, rojo: atrasado, amarillo: en desarrollo, verde: completado y celeste: no comenzado aún. Ver la figura C.41 en la página siguiente, y las figuras 6-9 y 6-10 de [3. JMCUML].

[6]Impresiones de pantalla: del inglés Screenshots

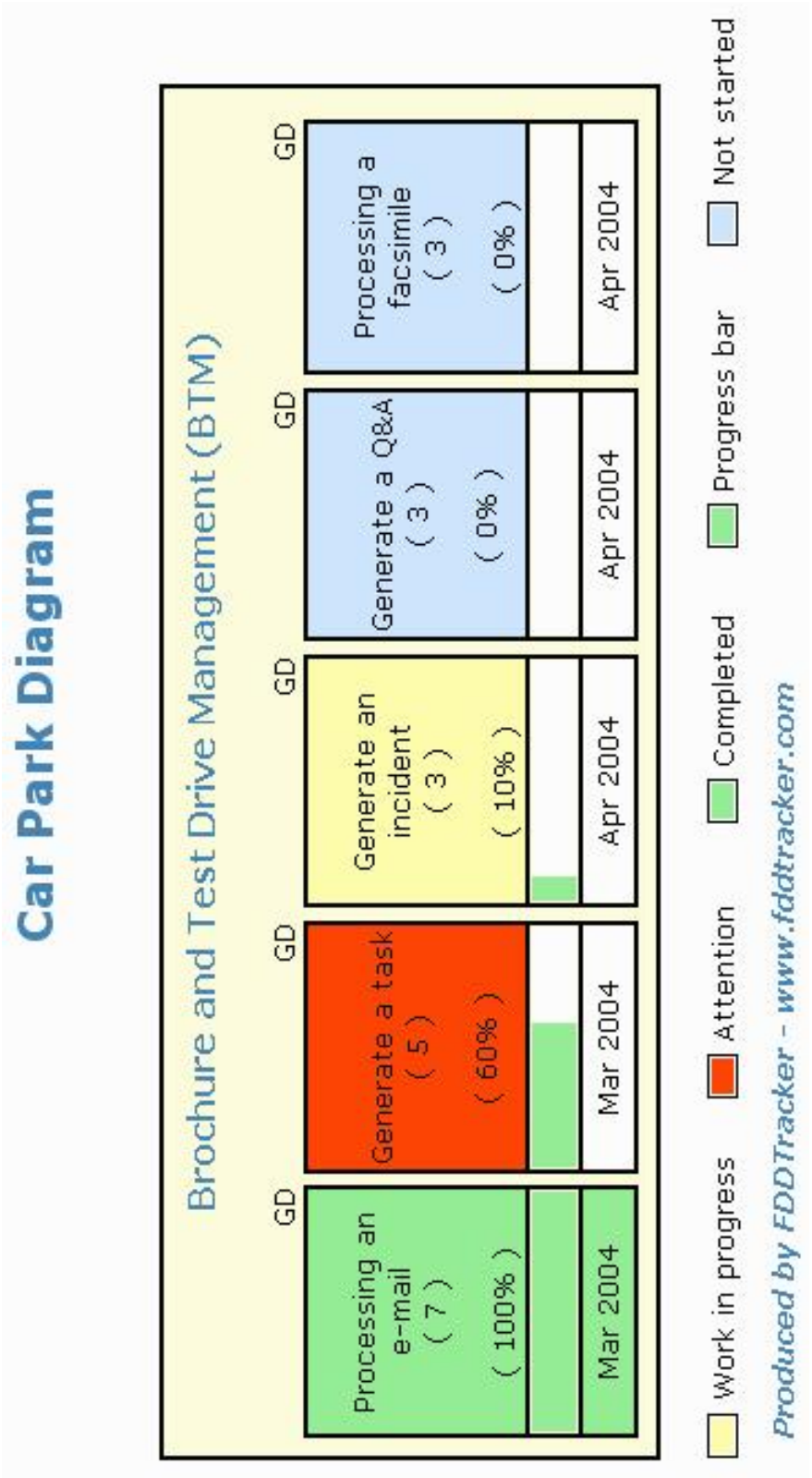

Figura C.41: FDDTracker: Diagrama de estacionamiento.

C.3.4.2. Visualización del plan

Esta tabla, ver figura C.42 en la página siguiente, muestra todos los rasgos, agrupados por actividad de negocio,

detallados de la misma forma que en la sección C.3.3.1 en la página 114.

TASK - Generate a task (5)

Id	Feature name	Chief Prog.	Design by Feature								Build by Feature	
			Walkthrough		Design		Design Inspection		Code		Code Inspection	Promote
			Plan	Actual	Plan	Actual	Plan	Actual	Plan	Actual	Plan	Plan
TA001	Set parameters for "Could not create Policy Holder" task in stored procedure	GD	13-04		13-04		13-04		13-04		13-04	13-04
TA002	Set parameters for "Could not create Q&A" task in stored procedure	GD	13-04		13-04		13-04		13-04		13-04	13-04
TA003	Set parameters for "Could not send dealer fax" task in stored procedure	GD	13-04		13-04		13-04		13-04		13-04	13-04
TA004	Set parameters for "Create follow up" task in stored procedure	GD	13-04		13-04		13-04		13-04		13-04	13-04
TA005	Calling the Stored proc to create a task structure	GD	13-04		13-04		13-04		13-04		13-04	13-04
Progress sum for feature set Generate a task:									Estimated completion date:			

INCIDENT - Generate an incident (3)

Id	Feature name	Chief Prog.	Design by Feature								Build by Feature	
			Walkthrough		Design		Design Inspection		Code		Code Inspection	Promote
			Plan	Actual	Plan	Actual	Plan	Actual	Plan	Actual	Plan	Plan
IN001	Processing the XML data for the Brochure request	GD	14-04		14-04		14-04		14-04		14-04	14-04
IN002	Processing the XML data for the Test Drive request	GD	14-04		14-04		14-04		14-04		14-04	14-04
IN003	Calling the Stored proc to create Incident structure	GD	14-04		14-04		14-04		14-04		14-04	14-04
Progress sum for feature set Generate an incident:									Estimated completion date:			

QANDA - Generate a Q&A (3)

Id	Feature name	Chief Prog.	Design by Feature								Build by Feature	
			Walkthrough		Design		Design Inspection		Code		Code Inspection	Promote
			Plan	Actual	Plan	Actual	Plan	Actual	Plan	Actual	Plan	Plan
QA001	Processing the XML data for the Brochure request	GD	15-04		15-04		15-04		15-04		15-04	15-04
QA002	Processing the XML data for the Test Drive request	GD	15-04		15-04		15-04		15-04		15-04	15-04
QA003	Calling the Stored proc to create Q & A structure	GD	15-04		15-04		15-04		15-04		15-04	15-04

Figura C.42: FDDTracker: Visualización del plan

C.3.4.3. Gráfica de tendencias

La gráfica de tendencias, muestra la cantidad de rasgos terminados por semana real y planificada, ver figura C.43.

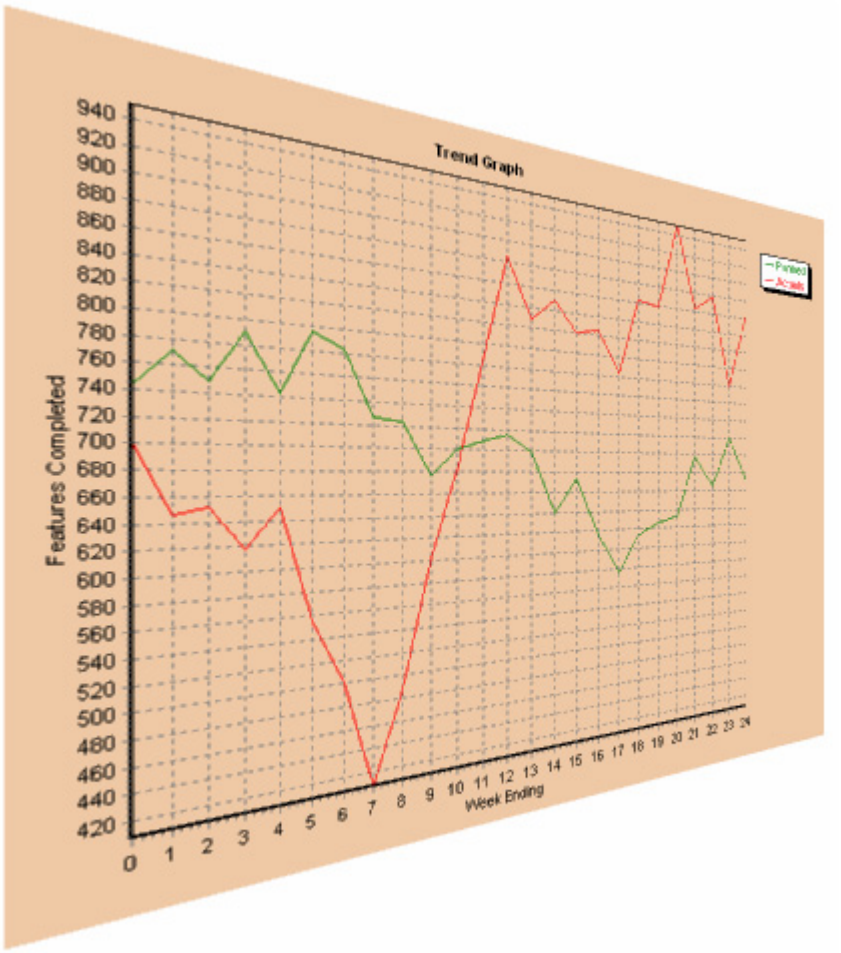

Figura C.43: FDDTracker: Impresión de pantalla del gráfico de tendencias

C.3.4.4. Gráfica de defectos

Esta gráfica muestra la cantidad de defectos de severidad alta, media, y baja por semana.

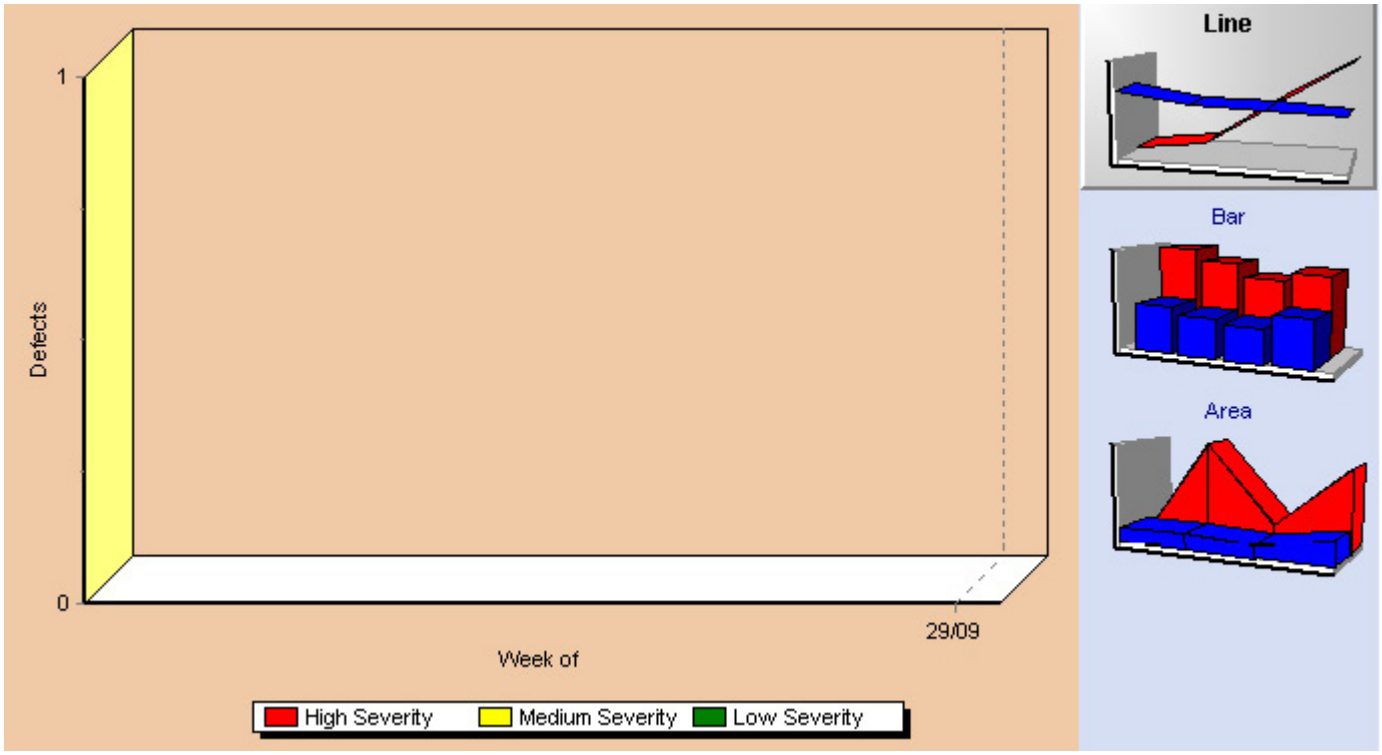

Figura C.44: FDDTracker: Gráfica de defectos

C.4. Otras herramientas

Esta sección explica brevemente herramientas que fueron evaluadas en sus aspectos generales, que serán comparadas con las que se describieron en los Capítulos C.1, C.2, y C.3, en el Capítulo C.5.

C.4.1. dotProject

dotProject.net FREE SOFTWARE Es un entorno de gestión de proyectos para web con tecnología PHP [3. PHP]. Incluye módulos para compañías, proyectos, tareas (con diagramas de Gantt), foros, archivos, calendarios, contactos, tickets/helpdesk, soporte de lenguaje múltiple, permisos y vistas por usuarios y módulos.

Esta herramienta sigue las líneas generales de TUTOS [1. TUTOS], ver Capítulo C.1 en la página 81, y fue evaluada en su versión 1.0.2. En la figura C.45 en la página siguiente, se muestra el panel de gestión de tareas de la herramienta.

Figura C.45: dotProject: Panel de gestión de tareas

C.4.2. NetOffice

Es un gestor de proyectos en línea que incluye: colaboración entre equipos, gestión de usuarios, diferentes niveles de acceso, tareas, proyectos, y registro de tiempos, historia de cambios en las tareas, seguimiento de aprobación de documentos, notas, sitios para los clientes de los proyectos, CRM[7], y diagramas de Gantt.

Esta herramienta sigue la líneas generales de TUTOS [1. TUTOS], ver Capítulo C.1 en la página 81, y fue evaluada en su versión 2.6.0. En la figura C.46 en la página siguiente, se muestra la página inicial de la herramienta.

[7]CRM: De sus siglas en inglés: Customer Relationship Management, gestión de relacionamiento con el cliente.

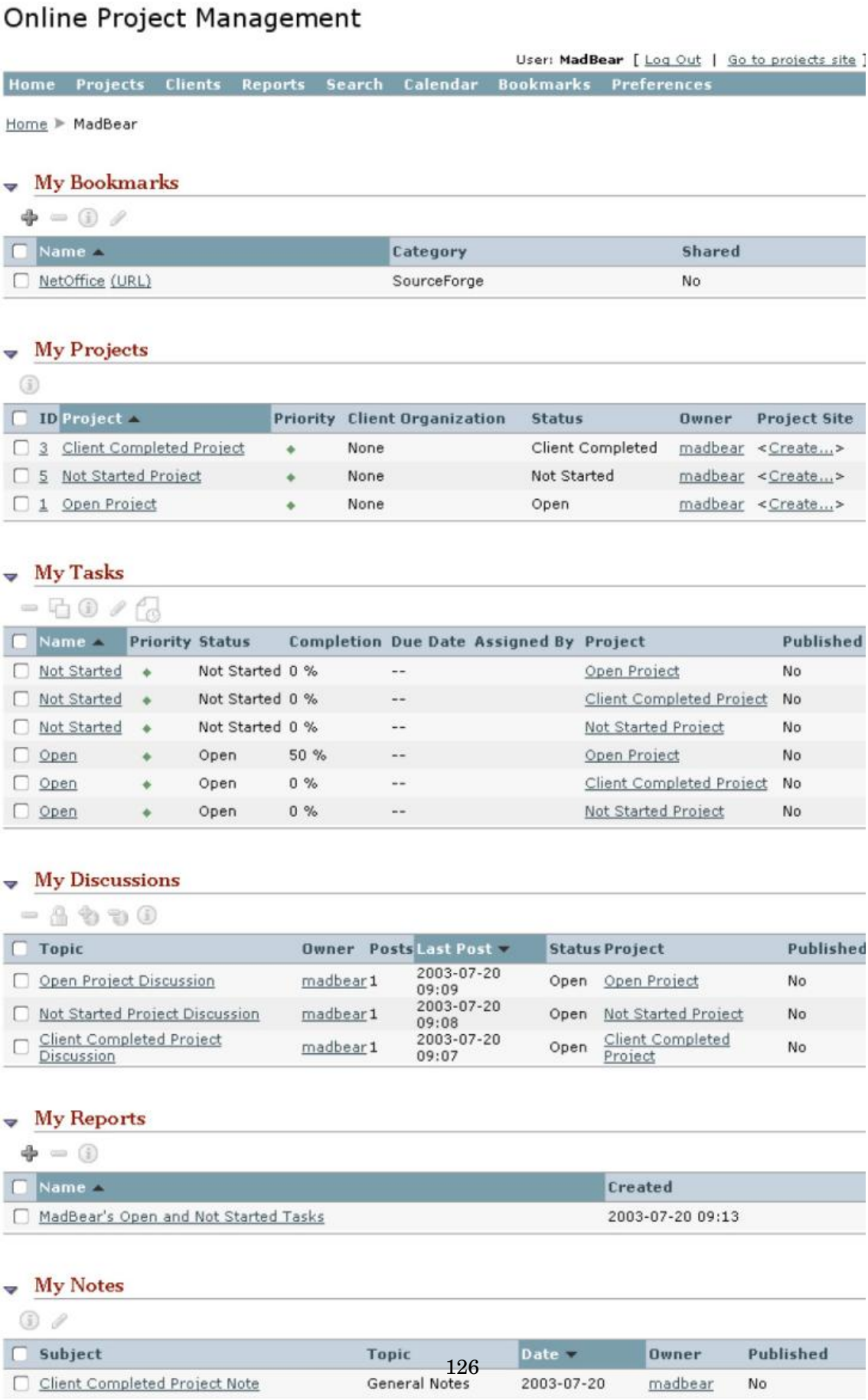

Figura C.46: NetOffice: Página inicial

C.5. Comparación de herramientas

Este Capítulo compara características de las herramientas citadas en los Capítulos: C.1 en la página 81, C.2 en la página 100, C.3 en la página 110, y C.4 en la página 123.

C.5.1. Generalidades

En el cuadro C.1 en la página siguiente se muestra una comparación de los atributos generales de las herramientas evaluadas. dotProject [2. dotProject], TUTOS [1. TUTOS] y FDDTracker [1. FDDTracker] gestionan empresas, mientras que NetOffice [2. NetOffice] y XPlanner [1. XPlanner] no. Todas las herramientas gestionan clientes, la diferencia es que dotProject[2. dotProject] y FDDTracker [1. FDDTracker], las gestionan como una empresa, y XPlanner [1. XPlanner] como una persona. Sólo dotProject [2. dotProject] y NetOffice [2. NetOffice] tienen foros de discusión. Todas tiene reportes, que son los siguientes:

- dotProject [2. dotProject]
  - Horas asignadas por usuario
  - Horas trabajadas de las horas asignadas
- NetOffice [2. dotProject], tiene reportes de tareas, que se pueden salvar, por los siguientes parámetros seleccionados por el usuario:
  - cliente(s)
  - proyecto(s)
  - asignación(es)
  - rango de fechas de finalización de tareas:
    - real
    - planificada
  - estado(s)
  - prioridad(es)
- TUTOS [1. TUTOS], ver sección C.1.11 en la página 97.
- FDDTracker [1. FDDTracker], ver sección C.3.4 en la página 117.
- XPlanner [1. XPlanner], ver secciones C.2.5 en la página 105 y C.2.4 en la página 104.

NetOffice [2. NetOffice], TUTOS [1. TUTOS], y FDDTracker [1. FDDTracker], gestionan equipos, mientras que dotProject [2. dotProject] y XPlanner [1. XPlanner] no. En dotProject [2. dotProject], NetOffice [2. NetOffice], y XPlanner [1. XPlanner] son registradas sólo la cantidad de horas dedicadas a las tareas, en TUTOS [1. TUTOS] el registro de tiempos es más completo ver sección C.1.8 en la página 91, y en FDDTracker [1. FDDTracker] no se registran las horas. Las citas sólo se gestionan en TUTOS [1. TUTOS], dotProject [2. dotProject] y NetOffice [2. NetOffice], en estos dos últimos se les llama eventos. Finalmente, la integración sólo se gestiona en XPlanner [1. XPlanner].

Herramienta	Empresas	Clientes	Calendario	Foros	Reportes	Equipos	Tiempos	Citas	Integración
dotProject [2. dotProject]	√	√	√	√	√		√	√	
Netoffice [2. NetOffice]		√	√	√	√	√	√	√	
TUTOS [1. TUTOS]	√	√	√		√	√	√	√	
FDDTracker [1. FDDTracker]	√	√			√	√			
XPlanner [1. XPlanner]		√			√		√		√

Cuadro C.1: Comparación de aspectos generales

C.5.2. Tareas

En el cuadro C.2 se muestra la comparación de elementos de las tareas. En dotProject [2. dotProject] las dependencias se manejan a nivel de tarea padre, y lista de tareas de las cuales depende, mientras en TUTOS [1. TUTOS] las dependencias son sólo jerárquicas. Los tareas tienen un estado en dotProject [2. dotProject]: activa e inactiva, NetOffice [2. NetOffice]: no comenzada, abierta, terminada, y atrasada, y en TUTOS [1. TUTOS]: no comenzada, abierta, suspendida, completa, y completa para el cliente. dotProject [2. dotProject], NetOffice [2. NetOffice], y XPlanner [1. XPlanner] asignan prioridades para las tareas, y estas son en dotProject [2. dotProject]: baja, normal y alta, en NetOffice [2. NetOffice]: muy alta, alta, media, baja, muy baja, y en XPlanner [1. XPlanner] es un entero, la prioridad 1 es la más alta. En TUTOS [1. TUTOS] y XPlanner [1. XPlanner] se puede asignar una nota a las tareas. Los documentos pueden ser asignados a una tarea en dotProject [2. dotProject], NetOffice [2. NetOffice] y TUTOS [1. TUTOS].

Herramienta	Dependencias	Estado	Prioridades	Notas	Documentos
dotProject [2. dotProject]	√	√	√		√
Netoffice [2. NetOffice]		√	√		√
TUTOS [1. TUTOS]	√	√		√	√
FDDTracker [1. FDDTracker]					
XPlanner [1. XPlanner]			√	√	

Cuadro C.2: Comparación de características de tareas

C.5.3. Proyectos

La única de las herramientas evaluadas que no tiene permisos a nivel de proyecto es XPlanner [1. XPlanner]. NetOffice [2. NetOffice], TUTOS [1. TUTOS], y XPlanner [1. XPlanner] permiten asociar notas a sus proyectos. En dotProject [2. dotProject], NetOffice [2. NetOffice], y TUTOS [1. TUTOS], se puede asignar documentos a proyectos. Los proyectos tienen un estado en todas las herramientas evaluadas, con excepción de FDDTracker [1. FDDTracker]. Los diagramas de Gantt, de las tareas de los proyectos, es generado en dotProject [2. dotProject], NetOffice [2. NetOffice], y TUTOS [1. TUTOS].

Herramienta	Permisos	Notas	Documentos	Estado	Gantt
dotProject [2. dotProject]	√		√	√	√
Netoffice [2. NetOffice]	√	√	√	√	√
TUTOS [1. TUTOS]	√	√	√	√	√
FDDTracker [1. FDDTracker]	√			√	
XPlanner [1. XPlanner]		√			

Cuadro C.3: Comparación de características de proyectos

C.6. Nota final y Conclusiones

C.6.1. Nota final

Al iniciar la evaluación de las herramientas, una de las herramientas que se comenzó a evaluar fue Microsoft Project 2000 [5. Microsoft]. Junto con otras herramientas se realizó su evaluación preliminar, y se vio que no se podían manejar más de un proyecto a la vez, ni compartir recursos entre sí, como otras. Esto llevó a que el documento estuviera centrado en las Herramientas que se vieron. A pesar de que este producto por sí solo no cumple con esas funcionalidades, casi en la culminación de esta investigación, se encontró una propuesta de Microsoft [5. Microsoft] comparable a las herramientas vistas, esta es Ms Enterprise Project Management 2003 [5. Ms EPM 2003]. Por cuestiones de tiempo, y accesibilidad a la demostración de este producto, no se pudo evaluar esta propuesta de gestión de proyectos. Un vídeo demostrativo de este producto puede verse en [5. Ms EPM Demo 2003].

La bibliografía está organizada según las siguientes categorías:

1. Herramientas de gestión de software evaluadas que se mostraron detalladamente.
2. Herramientas de gestión de software evaluadas que se citaron en sus generalidades
3. Enlaces citados en el documento que no se refieren a ninguna herramienta de gestión de software.
4. Herramientas de gestión de software no evaluadas.
5. Otras referencias citadas en el documento.

C.6.2. Conclusiones

Existen varias propuestas de herramientas de gestión de software. Algunas muy completas como dotProject [2. dotProject], NetOffice [2. NetOffice], o TUTOS [1. TUTOS]. Aunque, existen funcionalidades muy buenas, que están en unas pero no en otras. Por ejemplo, TUTOS registra la historia de todos sus objetos, pero no tiene foros de discusión como dotProject [2. dotProject] o NetOffice [2. NetOffice].

FDDTracker [1. FDDTracker] representa fielmente la metodología FDD [3. FDD], lo que la convierte en la herramienta ideal para una empresa que siga esta metodología.

La integración de módulos es la característica más interesante de XPlanner [1. XPlanner]. La cual permite una organización muy buena a la hora de llevar a producción los productos, ya que mantiene la información exacta de las funcionalidades disponibles.

Salvo necesidades particulares, se considera que NetOffice [2. NetOffice] o TUTOS [1. TUTOS], son herramientas con el nivel necesario de madurez para gestionar los proyectos de cualquier empresa de software.

Bibliografía

[1. FDDTracker] FDDTracker, Feature Driven Development Tracking Software. Versión 1.0 http://www.fddtracker.com Última fecha de visita 21/09/2004.

[1. TUTOS] TUTOS. The Ultimate Team Organization Software. http://www.tutos.org/homepage/index.html Última fecha de visita: 22/08/2004

[1. XPlanner] XPlanner. http://www.xplanner.org/ Última fecha de visita: 22/08/2004

[2. dotProject] dotProject.net. Free Software. http://www.dotproject.net/ Última fecha de visita: 21/08/2004

[2. NetOffice] NetOffice Project Management http://netoffice.sourceforge.net/index.php Última fecha de visita: 22/08/2004

[3. FDD] Feature-Driven Development. http://www.featuredrivendevelopment.com Última fecha de visita 19/09/2004

[3. Hibernate] Hibernate, relational persistence for idiomatic Java. http://www.hibernate.com Última fecha de visita 27/09/2004.

[3. ITPS] IT Project Services, Business Systems Consultants. http://www.itps.com.au/ Última fecha de visita 19/09/2004.

[3. Java] Java. http://java.sun.com Última fecha de visita 27/09/2004.

[3. JMCUML] Java Modeling in color with UML. Capítulo 6: Feature Driven Development. http://www.pcoad.com/download/bookpdfs/jmcuch6.pdf Última fecha de visita 19/09/2004.

[3. JSP] JavaServer Pages Technology. http://java.sun.com/products/jsp/ Última fecha de visita 27/09/2004.

[3. MySQL] MySQL database server. http://www.mysql.com Última fecha de visita 27/09/2004.

[3. PHP] PHP: Hypertext Preprocessor. http://www.php.net/ Última fecha de visita 27/09/2004.

[3. Struts] The Apache Struts Web Application Framework. http://struts.apache.org Última fecha de visita 27/09/2004.

[3. XP] eXtreme Programming. http://www.extremeprogramming.org/ Última fecha de visita: 29/08/2004

[3. XP Historias] Historias XP. http://www.extremeprogramming.org/rules/userstories.html Última fecha de visita: 29/08/2004

[3. XP Iteraciones] Desarrollo XP basado en iteraciones http://www.extremeprogramming.org/rules/iterative.html Última fecha de visita: 29/08/2004

[4. FDD Tools Project] FDD Tools Project. Version 0.7. http://fddtools.sourceforge.net Última fecha de visita 14/09/2004

[4. JETeam] JETeam. Versión 0.0.4. http://jeteam.sourceforge.net Última fecha de visita 14/09/2004

[4. phpaga] phaga. Version 0.2 http://phpaga.sorceforge.net Última fecha de visita 14/09/2004

[5. Microsoft] Microsoft. http://www.microsoft.com Última fecha de visita 27/09/2004.

[5. Ms EPM 2003] Microsoft Enterprise Project Management 2003. http://www.microsoft.com/spain/Office/project/epmoverview.asp Última fecha de visita 27/09/2004.

[5. Ms EPM Demo 2003] Demostración de Microsoft EPM 2003. http://www.microsoft.com/office/project/prodinfo/demo.mspx Última fecha de visita 22/09/2004.

[5. Ms Project Standard] Microsoft Project Standard 2003. http://www.microsoft.com/spain/Office/project/standoverview.asp Última fecha de visita 27/09/2004.