VNIVERSIDAD NACIONAL AVTNOMA DE MEXICO

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

"HERRAMIENTA PARA LA ADMINISTRACIÓN DE LA CONFIGURACIÓN EN TSP"

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN INGENIERÍA (COMPUTACIÓN)

P R E S E N T A:

ALEJANDRO TALAVERA ROSALES

DIRECTORA DE TESIS: M. EN C. MARÍA GUADALUPE ELENA IBARGÜENGOITIA GONZÁLEZ

MÉXICO, D.F. **2008.**

Dedicatorias

A la Universidad Nacional Autónoma de México y al Programa de Posgrado en Ciencia e Ingeniería de la Computación, por haberme dado la oportunidad de desarrollarme profesionalmente.

A la Mtra. Guadalupe Ibargüengoitia por su dirección, apoyo y paciencia para que este trabajo concluyera. Muchas gracias de corazón.

A la Dra. Hanna Oktaba y el Dr. Fernando Gamboa, por sus enseñanzas académicas y profesionales, mismas que trato de compartir y extender en mi ámbito profesional.

A los Maestros Gustavo Arturo Márquez y Reynaldo Alanís, por su interés en este trabajo así como su valioso aporte en mi desarrollo académico.

A Karen y Ara, que desde su llegada a mi vida ésta cambió. Karen, eres mi orgullo y siempre estaré a tu lado. Ara, agradezco tu compañía y apoyo, no solo por el que me brindaste con este trabajo sino por estar cuando más te necesitaba.

A Gigi, Pepe, Misha y Marco, por ser esa familia excepcional que me proporciona todo lo que puedo necesitar: apoyo y cariño.

A Marisa, Amelí y Gustavo, por ser amigos sinceros y que a pesar de que el tiempo avanza siguen estando cada vez más cerca.

A Lupita y Gabriel, por ser mis hermanos mayores y que como tales siempre será un gusto compartirles estos logros.

A Angie, Monse, Miguel Ángel y Miguel, por ser una gran familia que estimo y respeto mucho.

A Martín por ser un amigo honesto y sincero, sé que te dará gusto saber que ya lo logré.

A Sylviane por creer en mi trabajo pero más por ser una amiga y cómplice excepcional.

“La única posibilidad de descubrir los límites de lo posible es aventurarse un poco más allá de ellos, hacia lo imposible.”

Sir Arthur C. Clarke
1917 - 2008

Índice

Introducción

A medida que el desarrollo de software se presenta como una actividad en constante ajuste, se han presentado una gran cantidad de procesos que permiten a los equipos de desarrolladores participar en proyectos de muy distinta naturaleza y objetivos. El presente trabajo se enfocará en el proceso “Team Software Process” (TSP) de Watts S. Humphrey.

TSP se presenta como un proceso que puede ser fácilmente adoptado por cualquier equipo que desarrolla software, ya que se enfoca en dos aspectos claves:

- Es un proceso bien definido, donde se establece el desarrollo en ciclos lógicos y claros
- Establece la estructura del equipo de desarrolladores con base en asignación de roles

Sin embargo el trabajo de Watts S. Humphrey presentado en su libro ‘*Introduction to the Team Software Process*’ presenta problemas para su adopción en la práctica. Por citar algunos de estos problemas mencionamos los siguientes:

- No hay el planteamiento de un seguimiento automatizado de los proyectos con base en tecnologías actuales
- Se carece de guías o herramientas adecuadas para hacer disponibles los reportes y las formas en línea
- La validación de los datos proporcionados por los integrantes del equipo es manual y por tanto susceptible a errores
- No se contempla el uso de una base de datos para registrar y verificar la integración del equipo con el proceso bajo TSP
- La gestión de las formas, definidas en el proceso, es completamente manual

Con base en el análisis del trabajo de Humphrey surgió la inquietud de elaborar una propuesta, que en la medida de lo posible, resuelve algunas de las problemáticas antes señaladas.

Por todo lo anterior el objetivo del presente trabajo es el de mostrar el desarrollo la **Herramienta para la Administración de la Configuración para Equipos de Trabajo** denominada **HACET**.

El trabajo se ha dividido en cinco capítulos mismos que se describen a continuación.

En el Capítulo 1 se presenta los conceptos básicos del Proceso Team Software Process^{SM} (TSP), además de un breve análisis de las herramientas que actualmente se pueden utilizar.

El Capítulo 2 muestra un breve análisis de las tareas más relevantes para el desarrollo de un proyecto de software empleando TSP.

En el Capítulo 3 se describen las tareas en TSP que se asocian con la Administración de la Configuración.

El Capítulo 4 presenta la propuesta de la herramienta para la Administración de la Configuración en TSP que se denominará HACET.

En el Capitulo 5 se presentan las etapas de diseño y desarrollo que se siguieron para construir HACET.

Finalmente se presentan las conclusiones del trabajo y las posibles vertientes del mismo.

Capítulo 1

Conceptos básicos

En este capítulo se presentarán los conceptos básicos en torno al proceso de desarrollo denominado Team Software Process^{SM} (TSP) propuesto por Watts S. Humphrey [WSH 2000]. Este proceso para el desarrollo de software define un conjunto de tareas claramente establecidas y cuyo control es verificado como parte del mismo.

1.1 Técnicas para el desarrollo de software

Desde hace algunos años se han venido desarrollando distintas técnicas para el desarrollo de proyectos de software, las cuales giran en torno a los procesos de producción, según el tipo de producto de software que se obtendrá.

Si bien la definición y seguimiento de dichos procesos resulta vital y de suma importancia para la elaboración de cualquier clase de producto, en la ingeniería de software en particular su puesta en la práctica no resulta tan simple.

Un factor que no hay que olvidar es el hecho de que el software que actualmente se desarrolla es muy variado y sus cualidades pueden ser de distinta índole y naturaleza: sistemas de tiempo real, multimedios, sistemas distribuidos, sistemas de información en línea, entre otros.

Un ejemplo de esto lo tenemos en la definición de los procesos de producción de un CD-ROM interactivo, generado por el Laboratorio de Multimedia que se presenta en la Figura 1.

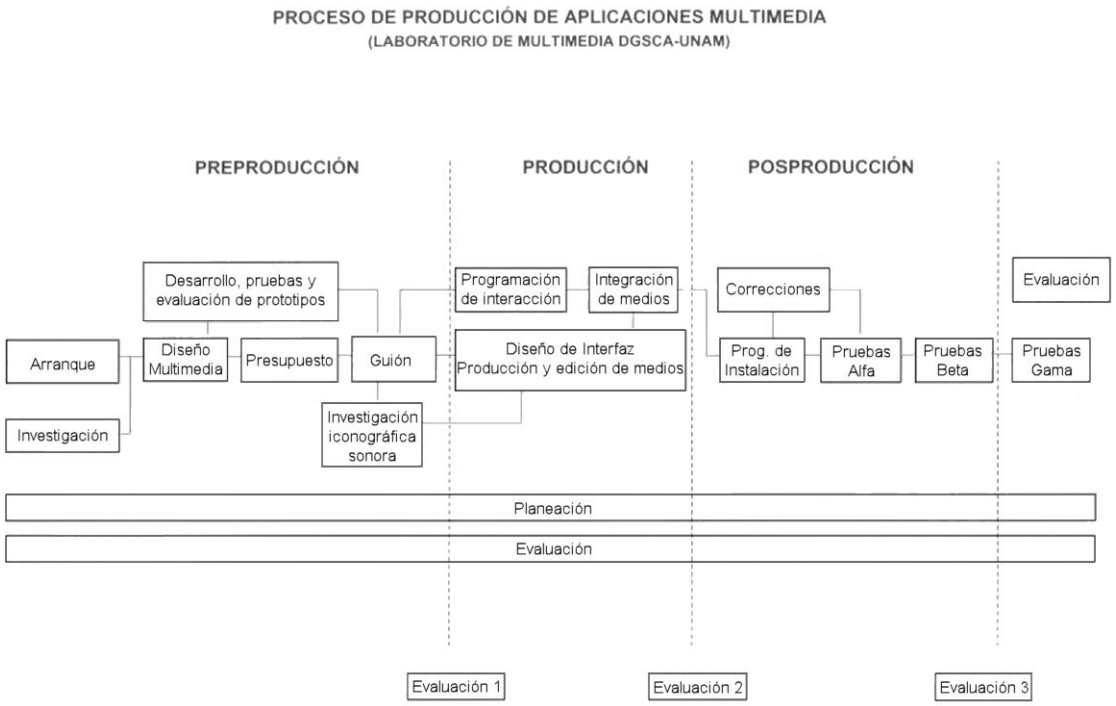


Figura 1 – Proceso de desarrollo del Depto. de Multimedia de la DGSCA

Obviamente el desarrollo de un CD-ROM multimedia es un producto muy especial. Por ejemplo, no pensaríamos que las actividades de Guión e Investigación iconográfica fueran actividades en un proceso de desarrollo para un sistema de consulta a bases de datos mediante un esquema de acceso a la red definido en J2EE.

Sin embargo existen similitudes en técnicas, por ejemplo se hace hincapié en la definición de pruebas de evaluación del software, además de la documentación de las actividades desarrolladas a lo largo de la duración del proyecto.

Existen tanto técnicas personales, como en el caso de Personal Software Process (PSP[1]), en donde el trabajo individual es medido, documentado y refinado mientras se desarrolla el producto. Este tipo de técnicas tiene como objetivo desarrollar una disciplina, en los procesos, en que el proyecto desarrollado esta a cargo de un solo individuo.

Es importante recalcar que si bien técnicas como PSP a la larga forman programadores efectivos, no es suficiente esta formulación para el desarrollo de software a otros niveles de complejidad donde se requiere más personas y la coordinación de cada uno de ellos de forma conjunta.

[1] Definido por Watts S. Humphrey del Software Engineering Institute

Como puede apreciarse, los problemas de desarrollo de software no sólo abarcan aspectos técnicos y de conocimiento de tecnologías, sino que ahora incluyen a la conducción de los esfuerzos individuales en grupo para generar un producto de software.

En un trabajo colaborativo los planteamientos deberían basarse en la definición de tareas y en la delegación de responsabilidades para el monitoreo de las primeras. Es decir, un equipo estará constituido según las tareas de desarrollo y las responsabilidades de cada integrante.

En la siguiente sección se presentarán los conceptos de Rol con las actividades asociadas a los mismos.

1.2 Roles y la formación de un equipo de trabajo

Antes de iniciar un análisis profundo con respecto a los miembros de un equipo de trabajo se dará una primera definición:

"Un equipo de trabajo es un grupo de personas quienes trabajan juntas de forma coordinada para alcanzar un conjunto claro de metas y objetivos."[2]

Tenemos que hacer hincapié en que un equipo de trabajo no sólo es la suma de individuos talentosos, sino que además, cada uno de los integrantes tiene claro el objetivo de su colaboración y que sus esfuerzos están planteados dentro de un proceso de desarrollo plenamente establecido.

De esta percepción del trabajo en conjunto, se tiene la necesidad de establecer las actividades del proceso de desarrollo, mismas que deberán estar asociadas a los miembros del equipo. En otras palabras cada persona desempeña roles.

Un rol es un conjunto de actividades relacionadas. Los roles son frecuentemente etiquetados con nombres comunes que denotan un conjunto de actividades: administrador, analista, diseñador, etc.

Existe también, la necesidad de establecer actividades que definan el rumbo de los esfuerzos, que deciden las tareas a desempeñar, organizar dichas actividades y monitorean sus resultados. Estas actividades son llevados a cabo por los roles de los administradores de los proyectos.

Existen varios tipos de administradores según los procesos y las técnicas empleadas para cada proyecto de software, y las categorías de éstos dependen en mucho del proyecto y las actividades definidas para cada caso. No existe en

[2] Tomado del libro *Succeding with Objects Decisión Framework for Project Management, Adele Goldberg, 1995*

la actualidad un consenso en la forma en que deberían ser definidos los roles y las actividades que desempeñan.

Aún con estas divisiones de trabajo no es posible garantizar el éxito o fracaso para todo proyecto de software. En palabras de DeMarco[3] esto se resume de la siguiente forma:

"Cuando los proyectos de Software fracasan, se debe generalmente a problemas en el equipo de trabajo y no a cuestiones técnicas"

Como cualquier grupo humano, los equipos de desarrollo de software no escapan a estos problemas, mismos que inciden en la efectividad en los procesos de producción y en las cualidades del producto final.

Un equipo de trabajo para un proyecto de software queda definido de la siguiente forma:

Un equipo consiste de:

a) Al menos dos personas
b) Trabajan con miras a una meta/objetivo/misión común, donde
c) Cada uno de los integrantes tiene asignado roles o funciones específicos a desarrollar y donde
d) La conclusión de los objetivos requiere de alguna forma de la dependencia entre los integrantes del equipo.

Para el caso de TSP cada uno de los miembros de un equipo de desarrollo de software conoce y asume uno o varios roles perfectamente definidos. Por lo anterior, cada rol definido en TSP establece actividades concretas.

Ahora bien, todas las actividades definidas en TSP son divididas en roles. Los roles que se definen para esta técnica son los siguientes:

- Líder del equipo
- Administrador de desarrollo
- Administrador de planeación
- Administrador de la calidad
- Administrador de apoyo
- Ingeniero de desarrollo

El **Líder del equipo** tiene como propósito principal el de mantener un equipo productivo en distintos niveles durante el desarrollo del proyecto. Por su parte el **Administrador de desarrollo** tiene como objetivo el de coordinar las tareas que define para el desarrollo del producto final. El **Administrador de la planeación** guía al equipo con un plan detallado para ser llevado a cabo por el equipo de

[3] Tomado del libro *"Introduction to the Team Software Process[SM]",* Watts S. Humphrey, 2000

desarrollo. El **Administrador de la calidad** define técnicas y procedimientos internos que resulten en un producto libre de defectos. Finalmente, el **Administrador de apoyo** tienen como objetivo el de apoyar al equipo en la adquisición de las herramientas adecuadas para el desarrollo del proyecto de software, además de administrar el control de cambios de los documentos durante el desarrollo. El Ingeniero de desarrollo por su parte, participa en las actividades de construcción del producto e interviene activamente en las tareas de los demás.

1.3 Herramientas de apoyo a los equipos de trabajo

Todas las labores que se definen en los procesos de producción de software, requieren de ser apoyadas por herramientas que faciliten su control y seguimiento. Por la diversidad de las tareas, para cada rol se requiere de un tipo particular de herramientas.

La mayoría de las herramientas para el control de los proyectos resultan ser adaptaciones de aquellas utilizadas en otras ingenierías, por ejemplo tenemos el caso de los diagramas de PERT[4] y la técnica de Ruta crítica[5], que pueden ser provechosas para los administradores de planeación y de desarrollo.

Estas dos últimas herramientas son muy utilizadas en la teoría de gráficas y representan problemas cuyos datos de administración son los siguientes: Nombre de la actividad, duración y holgura. Es decir, a partir de estimaciones de tiempo y la interrelación de actividades es posible describir teóricamente el orden de las mismas y cuales de ellas resultarán críticas para el proyecto en general. Estas gráficas pueden realizarse a mano sin mayor problema, aunque también existen herramientas como el software **Project** de la compañía Microsoft® que ha implementado estas gráficas como parte de sus opciones. La Figura 2 muestra la pantalla principal de Microsoft ® Office Project 2003.

[4] Método PERT (Program Evaluation and Review Technique) desarrollo por la Armada de los Estados Unidos de América, en 1957, para controlar los tiempos de ejecución de las diversas actividades integrantes de los proyectos espaciales. El diagrama PERT es una representación gráfica de las relaciones entre las tareas del proyecto que permite calcular los tiempos del proyecto de forma sencilla.

[5] Ruta crítica o CPM por sus siglas en Inglés (Critical Path Method), es uno de los sistemas que siguen los principios de redes, que fue desarrollado en 1957 y es utilizado para planear y controlar proyectos, añadiendo el concepto de costo al formato PERT.

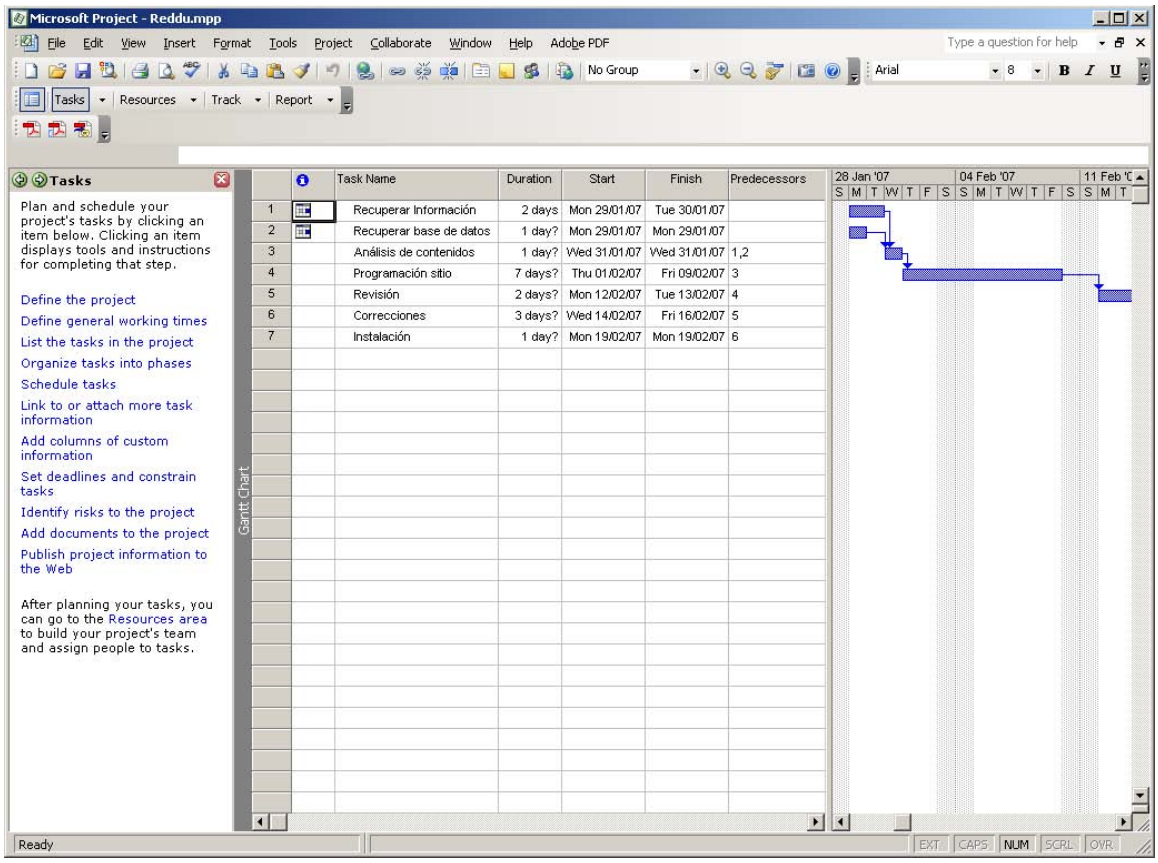


Figura 2 – Vista general de Microsoft® Office Project 2003

Si bien esta herramienta es muy flexible y es fácil de adquirir, fue pensada para apoyar la administración de cualquier tipo de tipo de proyecto a desarrollar, aunque cuenta con opciones de trabajo en equipo para proyectos de software, tales como la consulta y actualización de los reportes y documentos generados por cada miembro del equipo carecen de las opciones adecuadas para adecuarse idealmente a TSP.

Una herramienta útil para las tareas del equipo es el **BSCW**, producto alemán BSCW (Basic Support for Cooperative Work). Dicho software permite la colaboración de grupos de trabajo utilizando la Web. Este software ofrece un sistema de 'espacio compartido', además cuenta con las cualidades para permitir a sus usuarios subir cualquier tipo de documentos a dicho espacio utilizando para esto un navegador común. Permite además, el control de más de un grupo y la notificación automática de cambios. Aunque el control de documentos es una parte importante para la administración del proyecto total, se requiere que existan los procesos para el desarrollo del producto. Es decir, BSCW no define actividad alguna para el desarrollo de software, sólo se encarga de alojar los documentos que se van desarrollando. La Figura 3 muestra la página principal del proyecto BSCW.

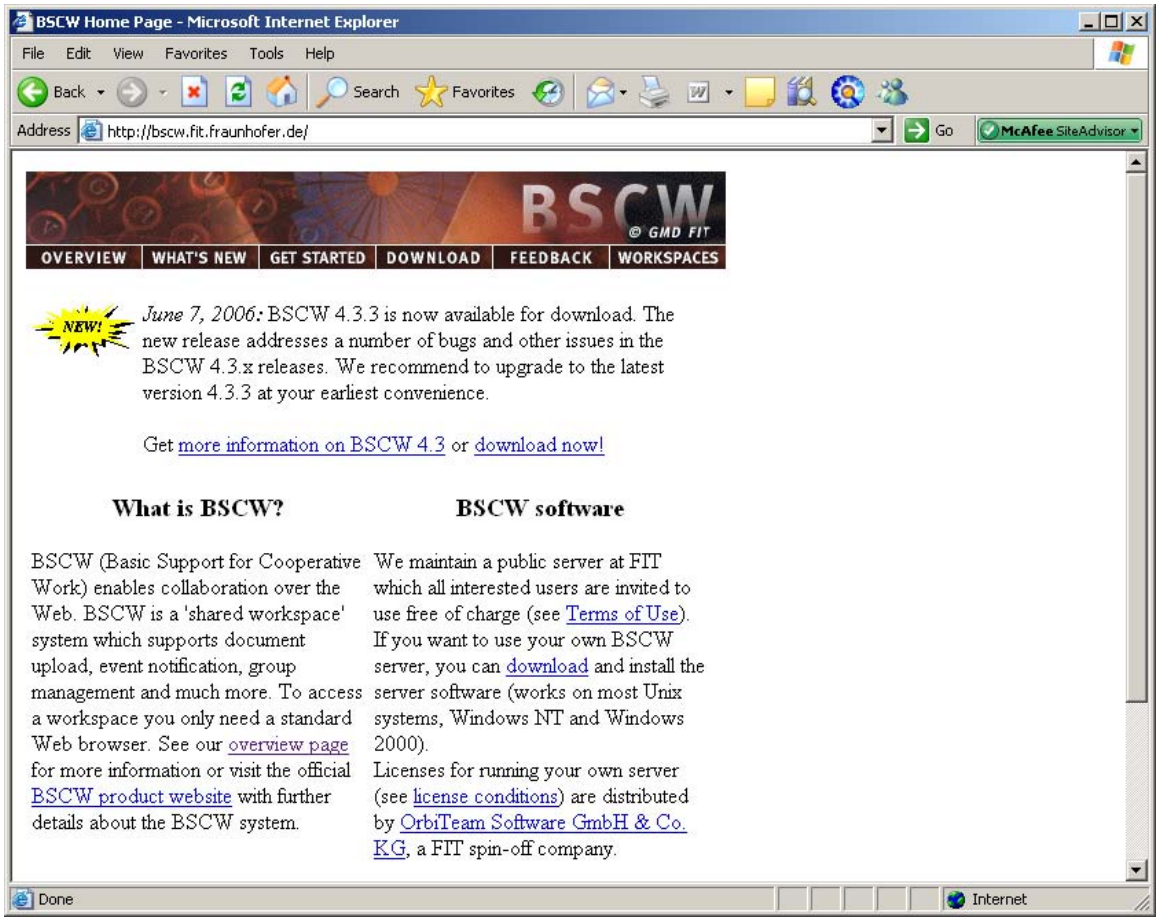


Figura 3 – Página principal del proyecto BSCW

Otro tipo de herramientas que actualmente se utilizan son las denominadas CASE tales como Rational Rose que se muestra en la Figura 4. Esta herramienta por ejemplo, cuenta con opciones para el modelado gráfico de sistemas de información utilizando el lenguaje de modelación UML (Unified Modeling Language). Dicha herramienta permite la generación de documentación para la especificación del producto, por lo que puede ser provechosa para el Ingeniero de desarrollo, pero a pesar de la ámplia gama de diagramas con las que cuenta no tiene herramientas o módulos para la administración de los procesos de producción, ni para la administración de la configuración o control de versiones.

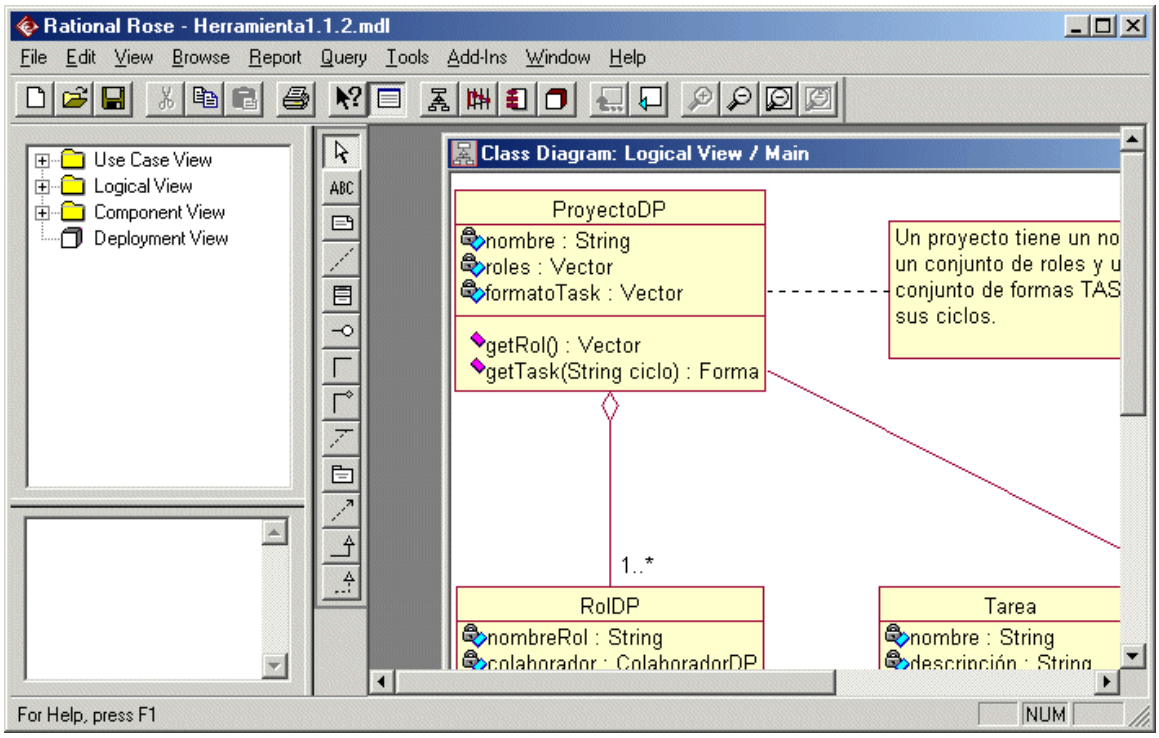


Figura 4 – Vista general de Racional Rose 2003

Por otro lado, se cuenta en la actualidad con herramientas para el control de versiones, de las cuales CVS (Concurrent Versions Systems) ha tenido una gran aceptación en los desarrollos de software libre. CVS es una aplicación que implementa un sistema de control de versiones, por lo que mantiene el registro de todo el trabajo y los cambios en los archivos (código fuente principalmente) que conforman un proyecto y permite que distintos desarrolladores colaboren. Se difunden el sistema bajo la licencia GPL[6]. La Figura 5 muestra la página del proyecto CVS.

[6] GNU GPL (Licencia Pública General) es una licencia creada por la Free Software Foundation a mediados de los 80, y está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

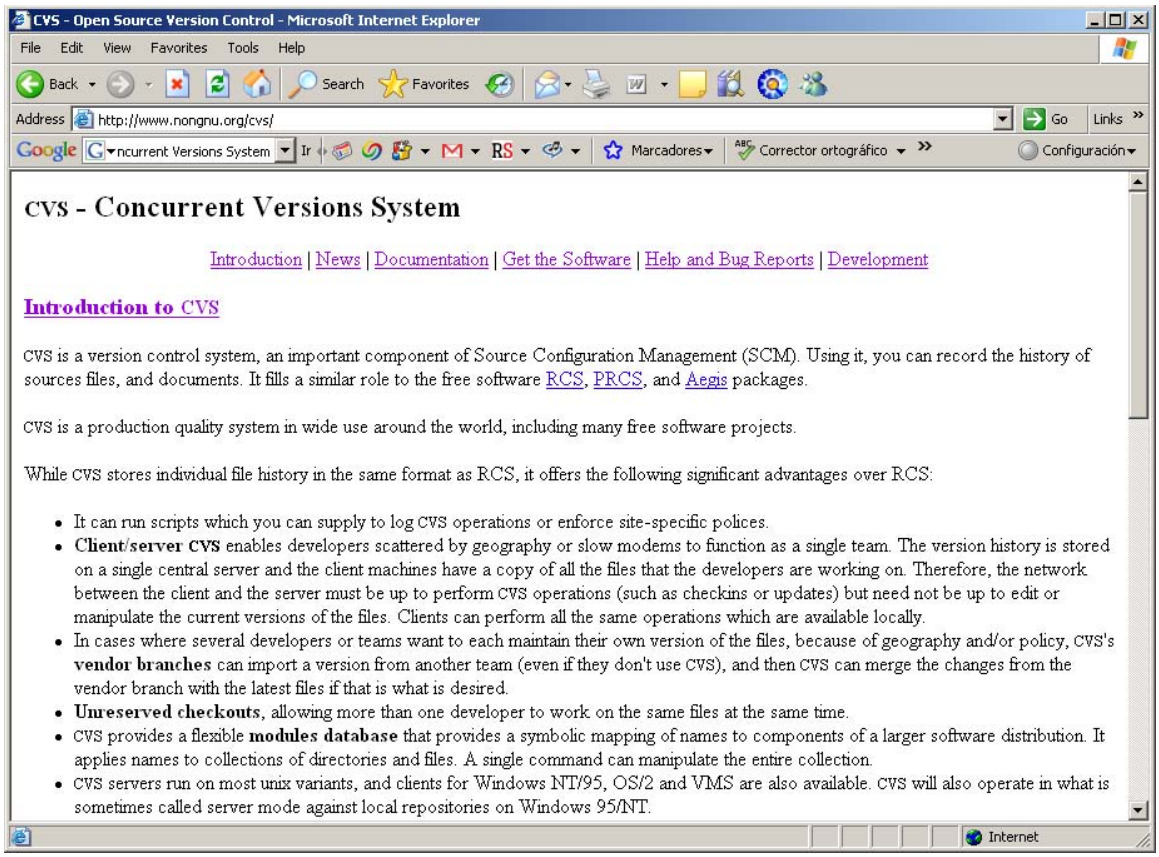


Figura 5 – Sitio oficial de CVS

Si bien CVS puede ser utilizado para la administración de documentos como las formas de TSP, no cuenta con un mecanismo adecuado para validar los datos que se integran.

Ahora bien, existen herramientas comerciales pensadas para el desarrollo de software y que permiten la colaboración de distintos miembros del equipo como es el caso de Borland® Together.

Esta herramienta permite el desarrollo visual de diagramas UML principalmente, para ayudar a la generación de código fuente. Cuenta con opciones para ayudar en la modelación de componentes de software profesionales siguiendo los modelos Enterprise definidas por Borland®. En este sentido Borland® Together apoya principalmente a las fases de desarrollo y de diseño. La Figura 6 muestra el aspecto general de dicha herramienta.

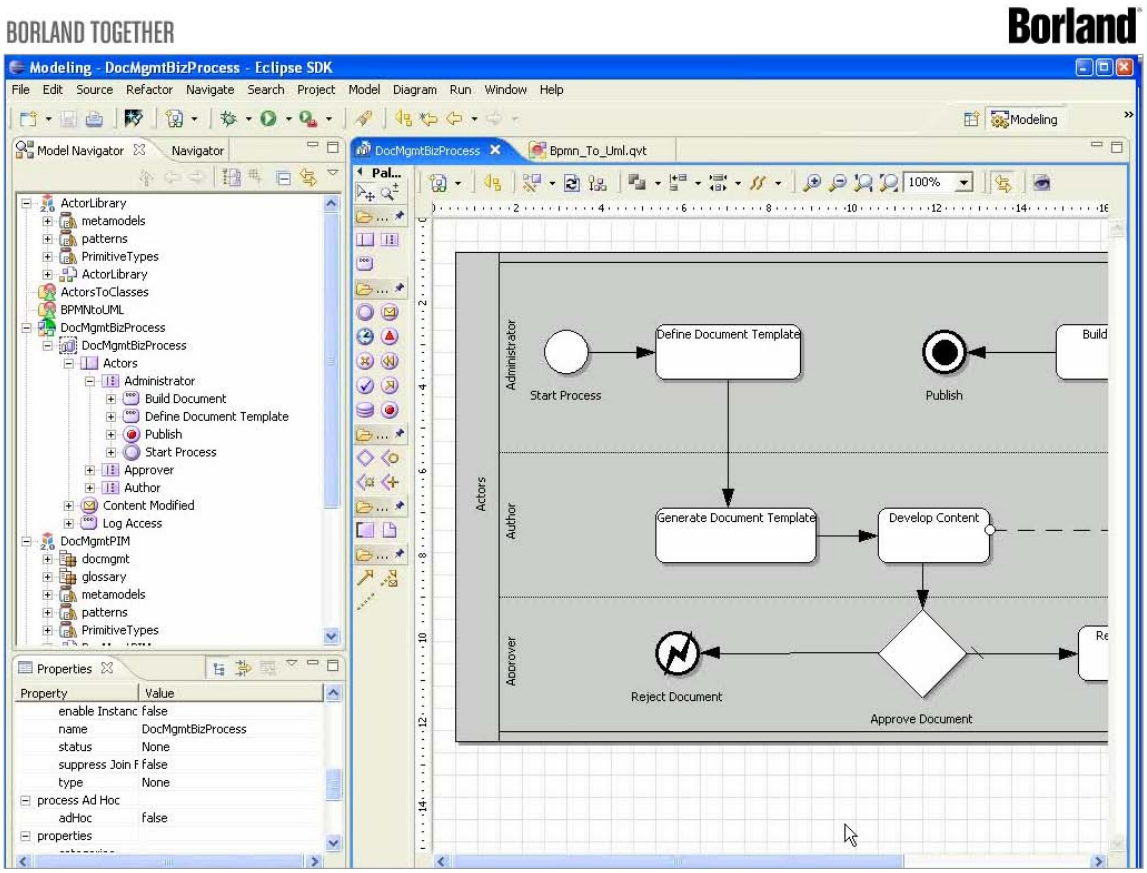


Figura 6 – Vista general de Borland® Together

Como puede apreciarse, no hay una herramienta a la medida que apoye al desarrollo de software, por el contrario, usamos una amplia variedad herramientas desintegradas y las adaptamos y conjuntamos con otras para que cumplan con las tareas que se requieren en el desarrollo.

Además el uso de distintas herramientas genera entre otros problemas la carencia de formatos comunes para los documentos y su control para el almacenamiento por otra parte, se carece de herramientas de apoyo a los roles como el Administrador de calidad y el Líder de proyecto

En el siguiente capítulo, se presentarán las actividades definidas por TSP para el proceso de producción de software, de las cuales se elegirán las que son más susceptibles de automatizar dado su impacto en el proceso en general.

Capítulo 2

Análisis de las tareas a desarrollar por TSP

En el presente capítulo se presentarán las tareas que llevan a cabo cada uno de los roles que define TSP. Además se analizará la administración de proyectos según TSP.

2.1 Planteamiento de un proyecto usando TSP

TSP[1] define un conjunto de tareas, roles y reportes los cuales representan las actividades de desarrollo de software. Las cualidades más importantes de TSP se pueden resumir en la siguiente lista:

1. Proporciona un marco de trabajo de desarrollo.
2. Desarrolla los productos en ciclos.
3. Establece estándares de medición tanto para la calidad del producto como para el desarrollo.
4. Proporciona métricas precisas de trabajo.
5. Utiliza evaluaciones por rol y para el equipo.
6. Requiere de un proceso disciplinado de trabajo.
7. Proporciona orientación a los problemas del equipo.

Estas cualidades son con las que se define el proceso de desarrollo. La idea es que el desarrollo del producto final se divida en etapas muy bien definidas, y en cada una de las mismas se encuentren las mismas actividades. Es decir, la división del proceso en general consistirá en desarrollar una versión cada vez más aproximada del producto final en cada ciclo. La Figura 1, muestra las actividades propuestas por TSP.

[1] Tomado del libro *"Introduction to the Team Software Process^{SM}",Watts S. Humphrey, 2000*

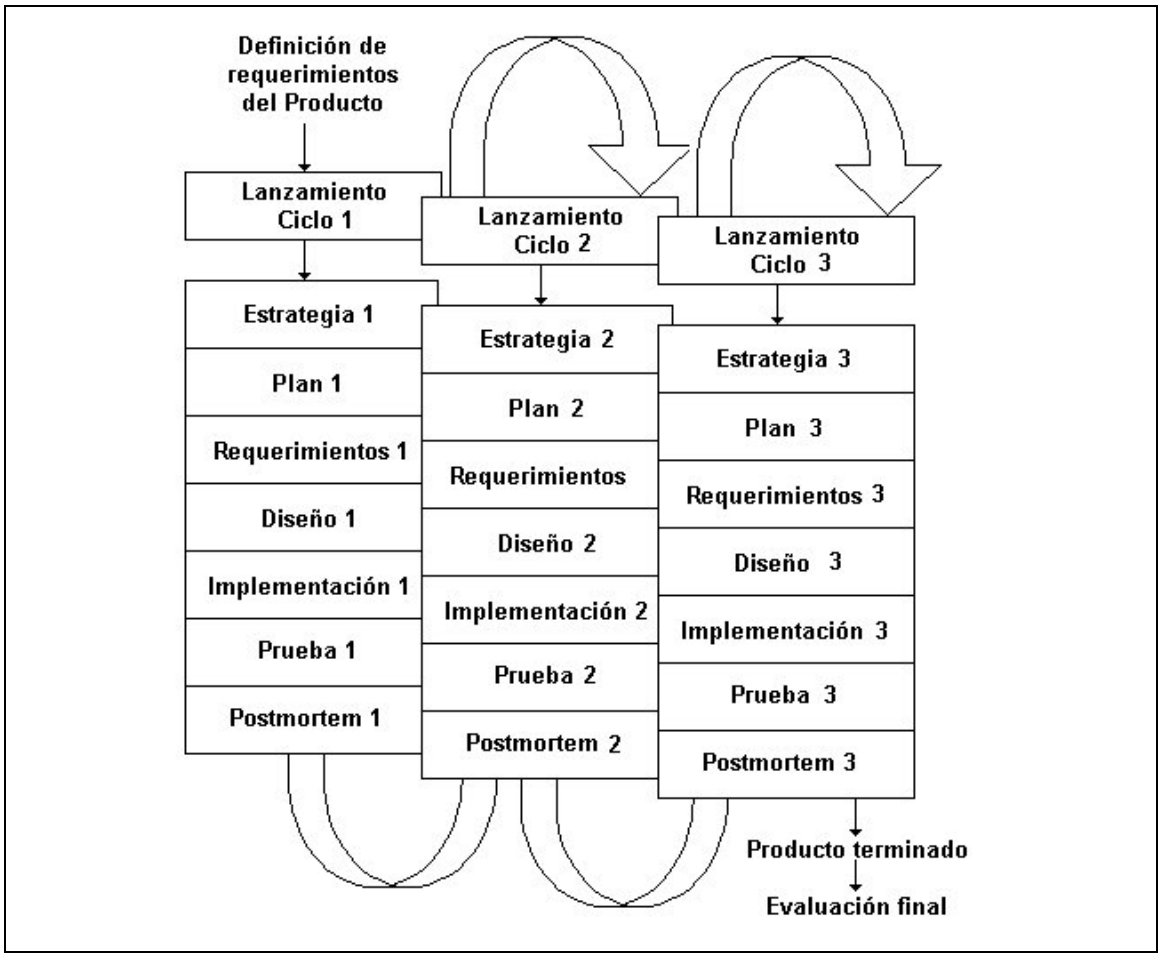


Figura 1 – Actividades propuestas por TSP

Si bien el planteamiento de cada ciclo depende en gran medida del tipo de producto a desarrollarse, es recomendado considerar las siguientes implicaciones de su definición:

1. Cada ciclo deberá producir una versión estable que es un subconjunto del producto final.
2. Cada ciclo deberá ser lo suficientemente pequeño para que sea desarrollado y evaluado en el tiempo disponible.
3. Cuando se combinen los productos de cada ciclo deberán producir el producto final deseado.

El proceso de TSP se inicia considerando que hay equipos que definen una estrategia de desarrollo y donde se establece una base razonable para ser desarrollada en el primer ciclo. Con lo anterior, se fijan estimaciones de las funcionalidades que se esperan añadir en cada ciclo subsecuente.

Es importante destacar que de dichas estimaciones se debería garantizar la terminación de un subconjunto inicial de trabajo del producto final.

Con los datos del primer ciclo se pude ajustar el plan para agregar funcionalidad en los subsecuentes ciclos. En la práctica la información que se va recabando permite la depuración y la mejora de los procesos de producción de los equipos de trabajo.

2.2 Roles en TSP

En TSP cada uno de los integrantes del equipo de desarrollo debe estar convencido de las metas que como equipo deben alcanzar. Es por lo anterior que la división de trabajo debe ser orientada en términos de roles a fin de delegar responsabilidades en núcleos perfectamente definidos tanto en alcance como en su relación con en el equipo.

Como se mostró en el capítulo anterior los roles definidos por TSP son los siguientes:

- Líder del equipo
- Administrador de desarrollo
- Administrador de planeación
- Administrador de la calidad
- Administrador de apoyo

Estos roles cubren un amplio espectro de las actividades del equipo y proveen a cada miembro de responsabilidades específicas mismas que se presentan a continuación para cada uno de ellos.

Líder del equipo

Su objetivo es el contar con un equipo de trabajo efectivo. Esto indica que la persona que desempeña este rol deberá además de crear un equipo mantenerlo en producción de forma continua. Deberá además resolver problemas internos y dar solución a todas las problemáticas que surjan durante un proyecto. Él es el responsable de establecer los contactos con los clientes y presentar los progresos del proyecto. Deberá convocar a reuniones internas o con los clientes. Deberá de motivar a los demás a llevar a cabo sus tareas.

En otras palabras el Líder del equipo cuenta entonces con una visión de lo que se quiere hacer en el proyecto y sirve de guía en todo momento para el equipo.

Las actividades que debe desempeñar de forma continua son las siguientes:

- Elaborar y presentar el reporte semanal de avance ante los responsables del proyecto.
- Notificar al responsable institucional del estado del proyecto.
- Mantener la carpeta del proyecto.

Administrador de desarrollo

Su objetivo es el de guiar al equipo en la elaboración del producto. Hay que recalcar que el producto generado deberá cumplir con una serie de cualidades establecidas desde su origen. Es importante establecer los criterios bajos los cuales se genera un producto de software, y para TSP se han definido los siguientes:

- El producto debe ser documentado en su totalidad.
- Los requerimientos de software, hardware y humanas son contemplados en todas las etapas de desarrollo.
- Toda la documentación cumple totalmente los estándares de producción.
- La implementación refleja la etapa de diseño.
- El producto cumple los criterios de calidad definidos por el equipo.

Debe utilizar los conocimientos y las habilidades de los demás integrantes para el diseño y desarrollo del producto. Por lo anterior interviene en el cálculo del tiempo para las etapas en que se divida el proyecto, así como los productos que se van generando. Así mismo, es responsable de la integración entre los distintos módulos que deban ensamblarse.

Interviene en forma activa en la definición del diseño de alto nivel y en la especificaciones del diseño del software. En otras palabras, sigue los estándares de diseño establecidos para implementar en su totalidad los productos planificados.

Una tarea que tiene que desempeñar es la evaluación de los componentes producidos, generando pruebas, reportes de fallas y la documentación del producto final para el usuario.

Administrador de planeación

Su función principal es la de generar un plan preciso y viable, tanto para el equipo en general como para cada uno de los integrantes del equipo. Establece parámetros de control para el seguimiento del plan durante cada uno de los ciclos del proyecto. Para apoyar al equipo, cada semana deberá de presentar reportes relativos al estado del proyecto tanto en lo general como en lo individual.

Determina las horas semanales que cada ingeniero invertirá en el proyecto, además de obtener los planes detallados para cada miembro del equipo.

Para la elaboración de los reportes sobre el estado y seguimiento del plan en general, requiere de los reportes de los demás integrantes.

TSP al dividir el desarrollo global en ciclos de desarrollo, y para cada uno de ellos el Administrador de planeación genera el correspondiente plan.

Los planes que genere en cada ciclo tienen que ser precisos en las tareas que se llevarán a cabo. Interviene por lo tanto, en la definición de los productos a generarse y en el establecimiento de los tiempos de desarrollo que el equipo consumirá en las tareas asociadas. Además define los tamaños de los productos, y para cada producto estima el tiempo requerido.

Parte de sus actividades son la de monitorear las cargas de trabajo a fin de ir balanceando los planes elaborados y ajustarlos a las necesidades del equipo en todo instante.

Finalmente genera el análisis del desempeño obtenido con respecto al desempeño planificado.

Administrador de calidad

Su función es la de diseñar, establecer y verificar los estándares de calidad en cada uno de los productos del proyecto de software.

Entre sus tareas tiene las siguientes:

- Definie el plan de calidad, así como verificar su cumplimiento.
- Previene al líder del proyecto sobre problemas con la calidad de los productos.
- Apoya al equipo en la definición y documentación de sus procesos.
- Establece y mantiene los estándares de desarrollo del equipo.
- Interviene en la aprobación los productos de línea base.
- En las etapas de prueba, registra los niveles obtenidos libres de errores para el sistema.

Aunque el número de tareas presentadas para el Administrador de calidad parezcan menores a las anteriores, su participación en el proyecto es muy alto, ya que actúa como un inspector con cada miembro del equipo.

Administrador de apoyo

Tiene por objetivo el de apoyar al equipo en la obtención, selección y administración de las herramientas necesarias. Proporcionar al equipo de herramientas y métodos que lo apoyen en el desarrollo de sus actividades. Participa además en la Mesa de Control de Cambios.
Desempeña las siguientes actividades:

- Establece estándares para la reutilización de componentes.
- Mantiene la lista de reutilizables manteniéndolos disponibles a los integrantes de equipo.
- Conseguir las herramientas de software y hardware adecuadas al proyecto.

2.3 Administración del proyecto en TSP

Hasta este punto es posible percibir que TSP se base en las actividades que cada uno de los roles va desempeñando durante el desarrollo de un proyecto de software. La interacción de los miembros del equipo es vital para que todas las tareas en conjunto se reflejen en los productos obtenidos.

Para un equipo en la actualidad su entorno de producción puede ser remoto, es decir, que no necesariamente todos los miembros del equipo laboran en la misma región geográfica, o bien, al mismo tiempo. Esta forma de trabajo es la que con mayor interés se ha venido utilizando y cuyos productos representan el trabajo de equipos completamente heterogéneos.

Un problema que se debe solucionar es la relativa a la disponibilidad de los recursos, tales como la documentación generada por los productos. No siempre son fácilmente accesibles por los desarrolladores, y más aún, cada uno de los integrantes puede tener distintas versiones del mismo documento.

Los siguientes elementos son tareas vitales al nivel de la administración de un proyecto de software usando TSP:

- Generación de bitácoras por rol.
- Generación de reportes individuales y en equipo.
- Consulta de los documentos generados en el proyecto.

Dichas actividades por sí solas presentan un obstáculo para su incorporación en los procesos de producción establecido ya que es grande la carencia de herramientas efectivas que permitan su automatización y asimilación.

Uno de los objetivos secundarios del proceso definido por Watts, es el de presentar un conjunto de técnicas adecuadas que fortalezcan el desarrollo de software en equipo. Pero gran parte del tiempo que cada uno de los integrantes

del equipo debe de invertir está en el análisis y entendimiento de los distintos documentos que corresponden a los roles definidos por TSP.

Ya que parte de las actividades de cada uno de los administradores es la de generar reportes sobre su trabajo y el de equipo de manera continua, se requiere de un mecanismo flexible para la generación de los mismos y su control.

Aunque las formas y los reportes que se generan están por completo documentados, cada uno varía en su objetivo así como su interrelación con los demás. Muchas veces en la documentación no queda claro los objetivos de algunos campos a llenar al igual que el objetivo de registrar algún indicador o su relación con el proyecto.

Por ejemplo el administrador planeación requiere de las formas Week de los demás miembros del equipo a fin de reportar el estado del proyecto y con ello monitorear el estado del mismo. Esto lleva a que algunos indicadores más que ser de utilidad a todos los miembros del equipo sólo tienen sentido al juntarlos y compararlos con los demás.

Es importante que el mecanismo de generación de reportes y su control que se ponga en práctica, cuente al menos con los siguientes elementos:

- Ofrecer los formatos de reportes con base al rol que desempeñe en un proyecto
- Ofrecer un mecanismo de validación de las formas
- Permitir la inspección de los documentos comunes por parte de los miembros del equipo
- Permitir en todo instante el conocimiento del estado del proyecto

A fin de proporcionar una mejor visión de los aspectos funcionales para un mecanismo automatizado de las tareas derivadas de TSP, en el siguiente capítulo se presentan el análisis puntual de las tareas que se requieren llevar según la Administración de la configuración.

Capítulo 3

La Administración de la Configuración en TSP

Previo a la discusión sobre las tareas puntuales para la Administración la Configuración (AdC) de TSP, se debe definir el término junto con sus actividades y sus alcances, para el proceso de producción. Una vez que lo anterior se tiene claro se sientan las bases para mostrar las tareas que en TSP se definen para la AdC.

3.1 Actividades del Administrador de la Configuración

La definición del concepto de la Administración de la Configuración de Software no es única, por lo que nos basaremos en las definiciones establecidas por varios autores para con estas caracterizar sus funciones y actividades.

- La AdC es la disciplina de administrar y controlar la evolución del sistema de software [WHS99].
- La AdC es el proceso de identificar y definir los elementos, las solicitudes de cambios, y verificar la completés y correctez de los elementos de un sistema de software [IEEE – 729 83].
- La AdC está relacionada con el desarrollo de procedimientos y estándares, para manejar la evolución de un producto de software, En esencia está relacionada con el control de cambios, el manejo de sistemas sujetos a cambios y como liberar dichos cambios del sistema a los usuarios [Som 92].
- La AdC es el arte de identificar, organizar y controlar las modificaciones al software construido por un equipo de programadores. El objetivo de la AdC es maximizar la productividad minimizando los errores [Bab 86].

De estas definiciones se presentan las tareas que deben insertarse en el ciclo de producción de software. Las tareas van desde el inicio del proyecto hasta las tareas de mantenimiento del mismo. Bajo este panorama, la AdC tiene como funciones primarias las siguientes tareas:

- Identificación de la configuración
- Definir el proceso de control de la configuración
- Auditoria

Es por dichas tareas que la AdC se vuelve parte integral para el proceso de desarrollo ya que permite una comprensión total del sistema a los desarrolladores, evaluadores, usuarios y personal de mantenimiento del mismo.

3.2 Administración de la configuración en TSP

Para TSP la AdC se definen como el conjunto total de las actividades usadas para administrar el contenido de un producto de software desde el inicio hasta el final del proceso de desarrollo [WSH 2000].

El propósito de la AdC en TSP es el de garantizar que el contenido del producto se conozca y esté disponible en todo momento, además de que las funciones del producto sean rastreables, desde los requerimientos, pasando por el diseño, hasta llegar a la implementación final. Además que todos los contenidos del producto estén propiamente controlados y protegidos.

Para TSP el sistema de la AdC esta enfocado ampliamente al control de cambios de documentos y el seguimiento de los mismos. Estas tareas recaen en el rol del Administrador de Apoyo que se presentó en el Capítulo 2.

En TSP la **Identificación de la configuración** tiene como objetivo garantizar que todos los productos a ser controlados sean nombrados de forma única, en qué momento un producto esta en línea base[1] y que pueda identificar un responsable para cada producto. Los principales elementos a controlar son los siguientes: los requerimientos, el diseño del producto, el código fuente, materiales de prueba y sus resultados y estándares de diseño, formas propias de TSP.

Por su parte, el **Proceso de control de la configuración** tiene como fin el de asegurar que los cambios que se requieran no generen conflictos con los demás productos generados y que ningún programa debería ser cambiado por más de un desarrollador a la vez. Esto último señala que cuando algún desarrollador está alterando un producto en línea base, a dicho producto no se deberá permitir el uso por algún otro que deseé cambiarlo.

Ahora bien, el proceso de **Auditoria** de software se debe ver como un medio por el cual una organización puede garantizar que los desarrolladores han efectuado todo su trabajo de tal forma que satisface cualquier revisión externa [IEEE –1042 87]. De esta especificación se debe considerar que en TSP parte de estas labores recaen sobre el Administrador de la calidad, el Administrador de desarrollo y Administrador de apoyo. Las tareas de estas personas en conjunto permiten establecer y llevar a cabo pruebas y validaciones de los productos que se generan en cada ciclo del proceso. Cada componente está en la versión adecuada

Como puede apreciarse, la conducción de una buena AdC depende de una buena definición de procedimientos cuyo objetivo global sea el de llevar la correcta conducción de las tareas que comprenden el desarrollo de software. La

[1] Se dice que un producto está en **línea base** (una especificación o producto) cuando ha sido revisado y que más tarde servirá de base en el desarrollo y mantenimiento. Puede ser cambiado, pero sólo a través del procedimiento de control de cambios que se ha establecido.

carencia de un conjunto de mecanismos que apoyen las tareas AdC en el proceso de producción de software, resulta en una mala planeación de tiempos, diseños deficientes, así como la corrección de errores puede resultar incompleta, y eventualmente no se implementa el producto realmente requerido.

Al igual que en otros procedimientos de desarrollo, para un esquema de desarrollo como TSP, es importante tener en mente la coordinación de los distintos integrantes del equipo de desarrollo. Si bien, el producto es el mismo, la falta de coordinación puede llegar a generar inconsistencias en el producto durante su desarrollo. Esto representa un factor crítico en TSP ya que los distintos roles van generando documentos y aquellos que resumen el estado del proyecto, depende de los reportes generados por otros roles.

Las actividades que serán descritas a profundidad de la AdC en las siguientes secciones se listan a continuación:

- Control de versiones
- Mesa de control de cambios
- Reportes y formas de la Administración de la Configuración

3.2.1 Control de versiones

El control de versiones es la actividad de administrar la historia de un componente conforme va sufriendo transformaciones [ThMc 93].

Desde el inicio de un proyecto de software se presentan cambios en un componente, ya sean por especificaciones de implementación, variaciones en la especificación inicial, entre otros. Dichos cambios se ven reflejados en los documentos que van generando los distintos roles en TSP.

En este sentido, todo lo que se genere en un proyecto de software es susceptible de mantener un registro de las distintas versiones en las que se ha transformado durante los ciclos de desarrollo: requerimientos, diseños y especificaciones de producción, reportes, programas, bibliotecas, entre otros.

Por lo anterior, todo control de versiones tiene que combinar procedimientos y herramientas que permitan guardar y controlar las distintas versiones de los elementos de interés.

Si bien TSP es un conjunto de buenas prácticas en un proceso de producción efectivo en el desarrollo, deja de lado todas las tareas puntuales que han de realizarse en un proyecto dado. En TSP no se define en ningún momento algún control de versiones explícitamente, pero en la práctica es indispensable y vital. Es común recurrir a dos técnicas de la ingeniería de software que apoyan las actividades de la AdC: la creación de la **carpeta del proyecto**, para los documentos; y la implementación de un **almacén** para productos electrónicos.

La carpeta del proyecto, y se refiere al manejo de material documental tales como planes, reportes semanales, reportes de pruebas, acuerdos, etc. Si bien, es una solución para tener un lugar único para el alojamiento de documentos, requiere de un fuerte compromiso del Administrador de la Configuración para mantenerla al día según el esquema de TSP. Como la documentación y los reportes son generados por los distintos integrantes del equipo de desarrollo en las diferentes fases, se requiere que cada uno de ellos tenga experiencia en el manejo de sus documentos tanto personales como de conjunto. Esto último no es posible de forma inmediata en la mayoría de los casos, sobre todo en aquellos que carecen de experiencia, o bien, están adoptando y/o depurando un proceso ya definido además los problemas de equipos de trabajo a distancia y la validación de una sola carpeta por proyecto.

Un almacén de productos electrónicos, como su nombre lo indica, tiene la finalidad de alojar y administrar los productos electrónicos que se van desarrollando. Por ejemplo: documentos en algún formato electrónico, reportes, diagramas, códigos fuente, bibliotecas liberadas, etc. Esta herramienta es de gran utilidad en la automatización de labores para la administración de versiones, pero no se propone eliminar por completo el uso de la carpeta.

3.2.2 Mesa de Control de la Configuración (MCC)

El establecimiento de procesos, políticas y herramientas enfocadas a la administración del control de cambios, permite al equipo de trabajo manejar las versiones de los productos que se generan. La definición de cada uno de los factores anteriores recae en la Mesa de Control de Cambios (MCC). En TSP las personas que la conforman son aquellas que tienen asignados los roles de Administrador de Apoyo y el de Administrador de desarrollo.

La Mesa de Control de Cambios (MCC) define el documento de solicitud de cambios que da a conocer a los integrantes del equipo de desarrollo. Para este propósito TSP define la forma CCR de sus siglas en inglés *Configuration Change Request*, la cual se muestra a continuación.

Forma CCR

Nombre		Fecha	
Equipo		Instructor	
Parte/Nivel		Ciclo	
Información del Producto			
Nombre		Dueño	
Tamaño del cambio		Medida	
Inspección		Moderador	

Dirección del respaldo:
Información del cambio Razón del cambio:
Beneficios del cambio:
Impacto del cambio:
Descripción
Estado
Aprobado: ☐ Información adicional: ☐ No Aprobado: ☐ Información requerida:
Aprobaciones
Dueño Fecha: Administrador Calidad/Proceso Fecha: MCC Fecha:

Obviamente, no implica que el cambio solicitado se dará en las condiciones que se sugieren en dicho documento, sino más bien, se inicia un proceso de evaluación y consideración de la petición. El proceso definido para los cambios deberá ser propuesto por la MCC y convenido por los integrantes del equipo. En el proceso TSP se sugieren las siguientes consideraciones para el procedimiento de solicitar un cambio:

1. Toda la información necesaria tiene que ser proporcionada con una forma CCR a la MCC.
2. El administrador de desarrollo y el responsable del producto llegan a un acuerdo en relación con el cambio, misma que será notificada a la MCC.

3. En caso de que la MCC determine que el cambio solicitado procede, el Administrador de la calidad tendrá entonces que verifica que el producto se cambia siguiendo el proceso acordado por el equipo y que se siguen los criterios de calidad convenidos.
4. La MCC verifica que el cambio es consistente con las necesidades del cliente o el usuario final y la estrategia de desarrollo.
5. Finalmente la MCC corrobora que los recursos requeridos para el cambio se encuentran disponibles para realizar el cambio.

Otras tareas importantes para la MCC es la definición y uso de procedimientos de respaldo y la generación eventual de reportes.

La capacidad de generar respaldos es esencial incluso para sistemas 'pequeños'. Es conveniente respaldar todos los elementos en línea base.

Por su parte los reportes son necesarios para ayudar al equipo a seguir el estado del producto mientras lo están desarrollando. Aunque los reportes que se emplean en TSP tratan de abarcar todos los aspectos necesarios, es conveniente proporcionar información pertinente sobre los productos en línea base y los cambios. La forma de TSP para el reporte del estado del proyecto es la CSR, de sus siglas en inglés *Configuration Status Report*, misma que se presenta a continuación.

Forma CSR

Nombre		Fecha	
Equipo		Instructor	
Parte/Nivel		Ciclo	

Proceso de Cambio de la Configuración: Actividad

	Semana Actual	Del ciclo a la fecha
CCRs emitidas		
CCRs aprobadas		
CCRs rechazadas		
CCRs reportadas		
CCRs sobresalientes		
CCRs reservadas		

Proceso de Cambio de la Configuración: Estado

Volumen del Producto bajo control	Semana Actual	Desde la semana anterior
Páginas de Texto		
Páginas de diseño		
Líneas de pseudocódigo		
LOC—total		
LOC—nuevo y cambiado		

Prueba del caso LOC		
Páginas de material de prueba		
Páginas de resultados		
Otros		
Otros		
Comentarios		

3.2.3 Reportes y formas bajo la Administración de la Configuración en TSP

Como se ha venido explicando desde el inicio, TSP ofrece un esquema de trabajo con el que se van generando una gran variedad de documentos para la toma de decisiones. Los documentos que se considerarán para su automatización se listan a continuación:

- Forma TASK
- Forma WEEK
- Forma LOGT

Dichos documentos se explican a continuación.

Forma TASK

Esta forma tiene como propósitos principales los siguientes:

- Estimar el tiempo de desarrollo de cada tarea proyectada.
- Calcular el valor planeado para cada una de las tareas definidas.
- Estimar la fecha de terminación de cada tarea.
- Proporcionar un medio básico de seguimiento en la calendarización de tareas, aunque estas no se hayan completado en su totalidad.

Cuando un equipo de desarrollo define el plan general, también se deben desglosar cada una de las tareas que se deben llevar a cabo en cada ciclo.

Esta forma debe ser conocida por todos los miembros del equipo de desarrollo y los Administradores de planeación y de desarrollo tiene una amplia participación en su definición.

La Figura 1 muestra la copia de esta forma.

Nombre _____ Fecha _____
Equipo _____ Instructor _____
Parte/Nivel _____ Ciclo _____

Tarea				Plan en horas							Tamaño del plan/Valor					Actual		
Fase	Parte	Nombre de la tarea	#Ingenieros	Líder del proyecto	Administrador de desarrollo	Administrador de planeación	Administrador de la calidad	Administrador de apoyo	Horas Totales de equipo	Horas acumuladas	Unidades de medida	Tamaño	Semana No.	Valor Planeado (VP)	PV Acumulado	Horas	Horas acumuladas	Semana No.

Figura 1 – Forma TASK

Forma WEEK

El propósito de esta forma es para preparar los reportes semanales acerca de estado del proyecto.

Cada miembro del equipo deben llenar esta forma mostrando el trabajo terminado en la última semana. Sirve también como base para la planificación de la las actividades de la siguiente semana.

Cada semana el Administrador de la planeación prepara una copia de esta forma con un resumen del estado del equipo y los logros alcanzados.

Nombre _____ Equipo:_____
Fecha _____ Ciclo #_____ Semana #: _____

Datos semanales	**Planeados**	**Actuales**
Horas proyectadas para esta semana	_____	_____
Horas proyectadas en el ciclo a la fecha	_____	_____
Valor ganado para esta semana	_____	_____
Valor ganado en este ciclo a la fecha	_____	_____
Horas totales para las tareas en esta fase a la fecha	_____	_____

Datos semanales de cada miembro	Horas planeadas	Horas actuales	Valor ganado	Semana planeada
Instructor				
Ayudante				
Totales				

Tareas terminadas	Horas planeadas	Horas actuales	Valor adelantado	Semana planeada
Totales				

Seguimiento de resultados/riesgos

Nombre	Estado

Otros elementos importantes.

Figura 2 – Forma WEEK

Forma LOGT

Esta forma tiene como propósito llevar una bitácora personal relativa a las tareas y los elementos producidos cada semana de desarrollo. Además de mantener bitácoras separadas, registra el tiempo que se invierte en el proyecto por persona.

TSPi Time Recording Log: Forma LOGT

Nombre _____ Fecha _____
Equipo _____ Instructor _____
Parte/Nivel _____ Ciclo _____

Fecha	Inicio	Fin	Tiempo de interrupción	Tiempo delta	Fase/Tarea	Componente	Comentarios

Figura 3

Por todas estas razones las tareas de la Administración de la Configuración que se considerará para automatizar, usando TSP, serán las siguientes:

- Generación de reportes de cada rol
- Generación de los reportes generales
- Generación y consulta de los planes definidos para un proyecto

En el Capítulo 4 se mostrarán los reportes y formas de TSP que se definen para su automatización.

Capítulo 4

Propuesta de la herramienta para la Administración de la Configuración en TSP

Con base en el análisis de las actividades y procedimientos descritos en el capítulo anterior, se presenta una serie de elementos que deberá cumplir la herramienta para la AdC en TSP a fin de que sea útil y apoye las actividades propuestas.

4.1 Alcances y características de la herramienta

La herramienta de software que se propone tiene como objetivo cubrir los siguientes elementos clave en la administración de la configuración para TSP:

- Control de versiones
- Automatización de la mesa de control de cambios
- Generación de reportes y formas

Dichos elementos se describen a continuación para formular los requerimientos de la herramienta.

Control de versiones

Como se presentó en Capítulo anterior, el control de versiones se requiere para cualquier producto generado por el equipo de desarrollo. Para este módulo los productos generados serán administrados por la herramienta manteniendo los siguientes atributos para cada uno de ellos:

- Identificador.- Es un identificador único para cada producto, y deberá seguir las reglas de nomenclatura definidas por parte del administrador de calidad.
- Nombre.- Nombre del documento o producto
- Ciclo.- Es el ciclo en el cual se generó el producto
- Número de versión.- Número de la versión del documento o producto
- Autor.- Es el autor del documento o producto
- Tipo de documento.- Tipo de producto generado, por ejemplo código fuente, documento del análisis, etc.
- Estado.- indica el estado del documento, el cual puede ser cualquiera de los siguientes: Borrador, Obsoleto, Acabado
- Resumen.- Un pequeño texto que resume el producto o documento. Se utiliza para fines de referencia y/o consulta.
- Proyecto.- Nombre del proyecto al que correspondiente.

- Equipo.- Nombre del equipo

Es decir, la herramienta que se ofrece a los integrantes del equipo de desarrollo cuenta opciones con las cuales se les permitirán registrar y consultar los documentos y otros productos que van generando. Todo lo anterior conforme se progresa en el proyecto y además se registra la información relevante de cada uno de los productos.

Las opciones que la herramienta presenta para cualquier administrador en este caso serán las siguientes: registro de producto, actualización de producto y búsqueda de productos. La Figura 1 muestra los casos de uso derivados de estas opciones.

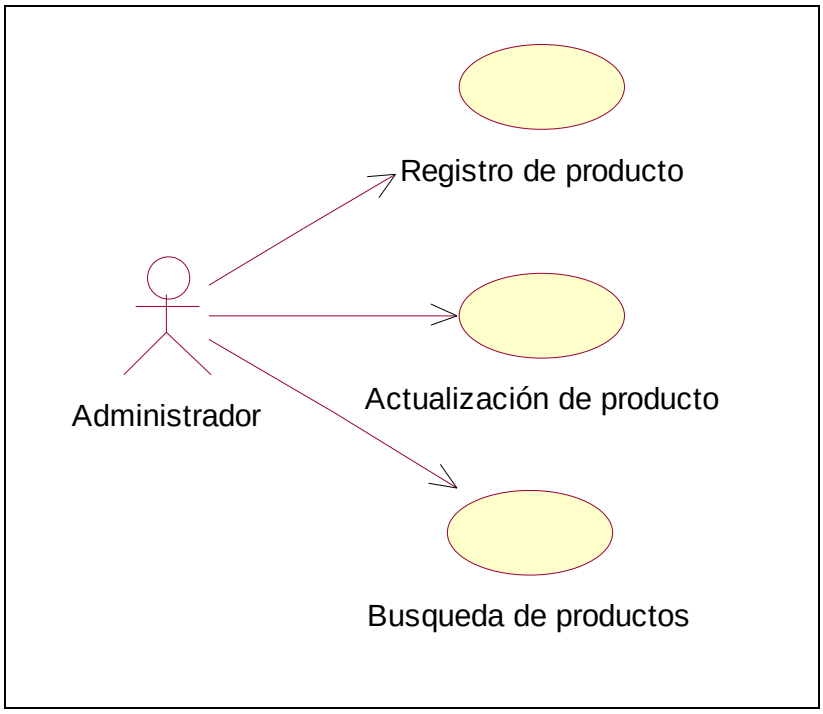


Figura 1 – Casos de uso para el manejo de productos

En este caso todo administrador es capaz de generar, consultar y actualizar (con los permisos correspondientes), los documentos generados en el proyecto. Cabe señalar que la herramienta cuenta con un almacén creciente de documentos.

Automatización de la mesa de control de cambios

La mesa de control de cambios en TSP tiene asociado los siguientes documentos: Forma CCR y Forma CSR, que se presentaron en el Capítulo 3. La primera de estas formas corresponde a la solicitud de cambios generada por un administrador del proyecto. La segunda representa el estado de la configuración, misma que sirve de apoyo para el estado de los documentos presentados cada semana.

La propuesta de la herramienta en este caso es disponer electrónicamente de la Forma CCR. Es decir, todo miembro del equipo puede solicitar cambios para alguno de los productos que se están desarrollando. Ya que el proyecto de software en particular contiene características propias a reportar, se propone que el equipo defina el documento que más le convenga y que éste pueda ser manejado como un documento más de los generados.

Para la solicitud de cambios en un producto, intervienen dos tipos de actores, en primer lugar tenemos a aquellos que solicitan un cambio en alguno de los productos y en segundo lugar tenemos a la mesa de control de cambios que da seguimiento a la solicitud. La Figura 2 muestra los casos de uso de los Administradores y la MCC.

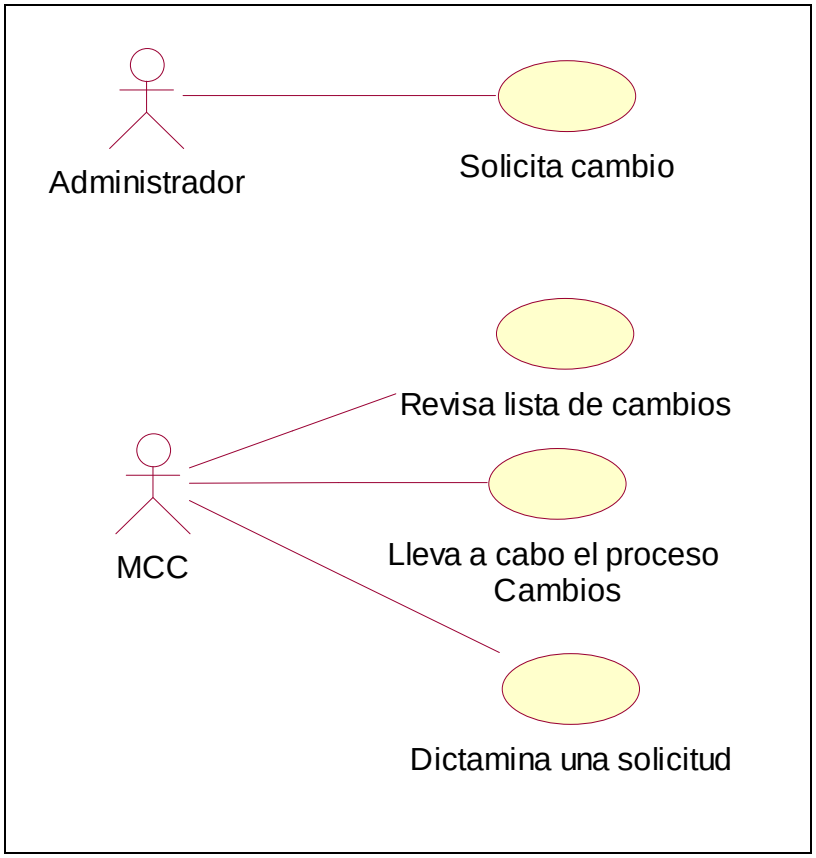


Figura 2 - Casos de uso para los Administradores y la MCC

Para la herramienta se contempla la siguiente interacción:

1. Un administrador solicita un cambio y se registra en el sistema
2. Un miembro de la MCC que atiende la solicitud
3. El miembro de la MCC da respuesta a la solicitud
4. El administrador que solicitó el cambio revisa la respuesta a su solicitud

La Figura 3 resume los puntos anteriores.

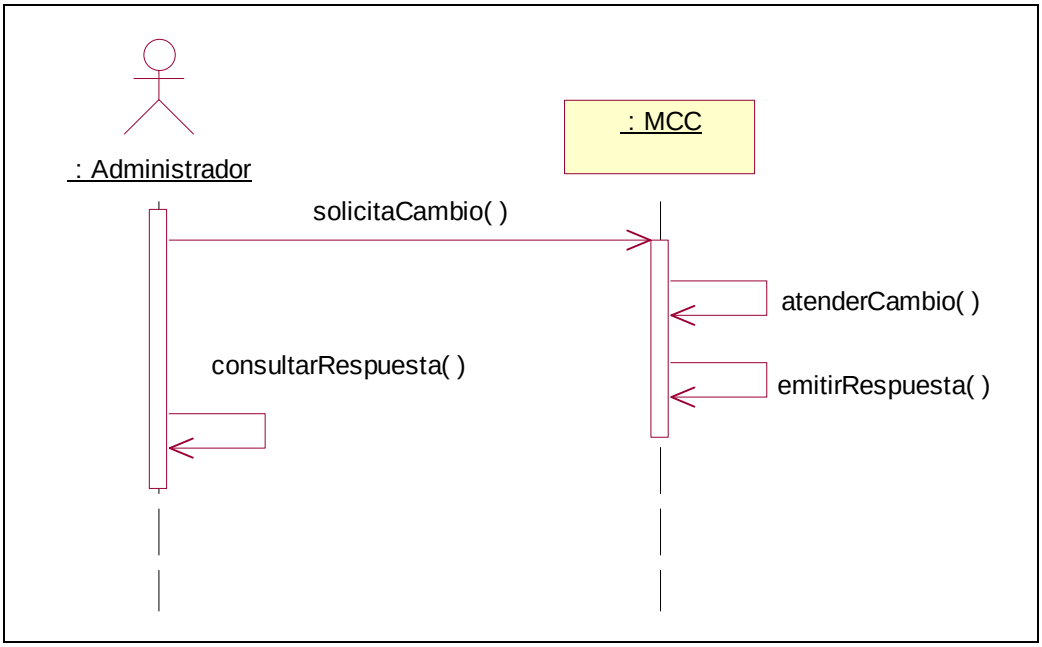


Figura 3 – Solicitud de cambios

Generación de reportes y formas

Como se mencionaba TSP lista un amplio conjunto de formas las cuales también deberán ser llenadas por todos los participantes en el equipo. Las más utilizadas son: Forma TASK, Forma WEEK y forma LOGT, presentadas en el capítulo anterior. Como se indicaba, cada una de estas formas son llenadas y consultadas para la verificación de tiempos, avances y toma de decisiones en el proyecto.

Se propone que la herramienta automatice la generación y consulta de esas formas.

Forma TASK

El responsable de la forma TASK es el administrador de planeación, pero puede ser consultada por los demás administradores en el proyecto. La información que la conforma indica la forma en que las actividades se han planificado y calendarizado. Además, esta forma sirve de base para los reportes semanales que cada uno de los administradores y líder de proyecto llevarán durante el desarrollo. La Figura 4 resume las opciones de la herramienta para la generación y consulta de esta forma.

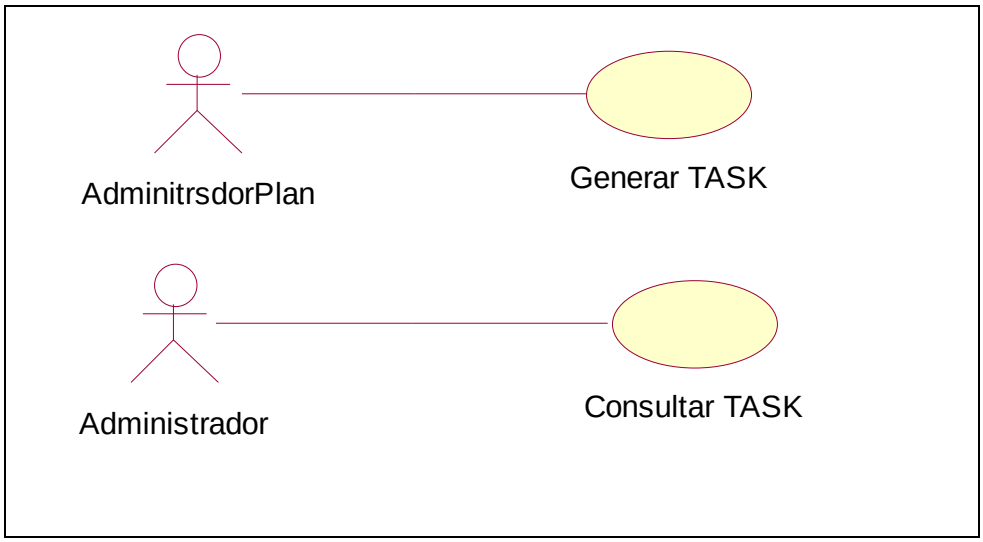


Figura 4 – Casos de uso para la Forma TASK

Es decir, la herramienta debe garantizar que el Administrador de planeación genere la Forma TASK pero a la vez, ésta pueda ser consultada por cualquiera de los administradores.

Forma WEEK

Cada uno de los participantes genera semanalmente un reporte que indica sus avances y en conjunto con los demás. Permiten generar el reporte semanal que se presenta para indicar el estado del proyecto. La Figura 5 resume las opciones que la herramienta usará para esta interacción

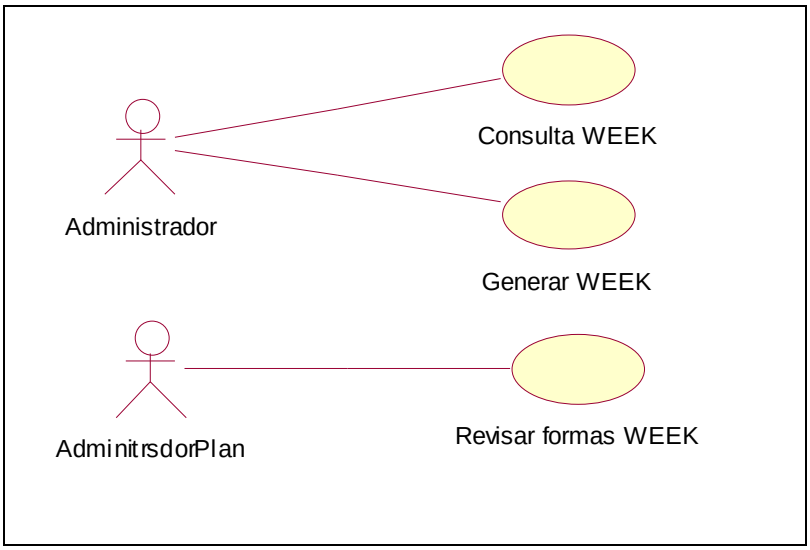


Figura 5- Casos de uso para la forma Week

Entonces cada uno de los participantes puede generar y consultar la Forma WEEK que le corresponde atender y generar, pero el líder de proyecto puede consultar el de los demás para que resuma el estado del proyecto.

Forma LOGT

Esta forma es llenada por cada uno de los participantes los cuales van actualizado la información en ella relativa a las tareas y otros elementos generados en cada una de sus tareas. La Figura 6 resume las opciones para esta forma.

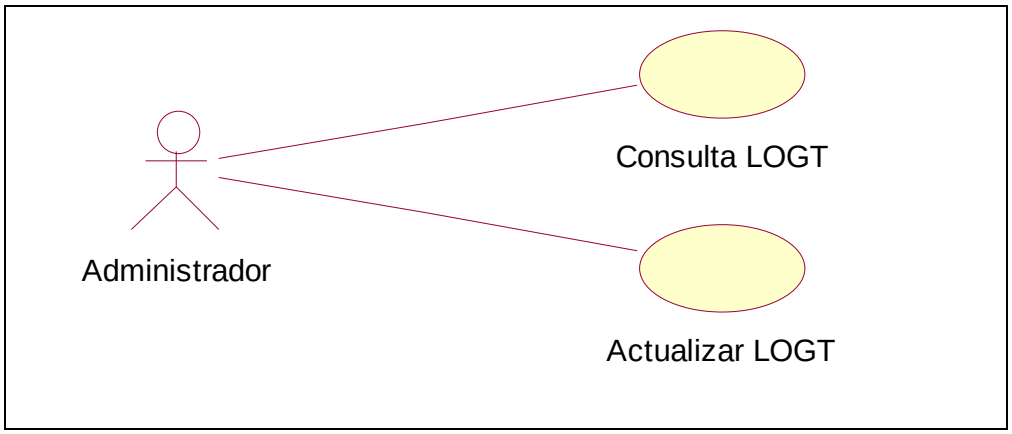


Figura 6 – Casos de uso para la forma LOGT

Esta forma permite medir los esfuerzos que por tarea o producto se utilizan en la generación de los elementos producidos.

4.2 Incorporación de la herramienta en el proceso de desarrollo

Los procedimientos automatizados que se incorporan en la herramienta, permiten que sus procesos se encuentren regidos por los flujo de información que TSP sugiere desde un inicio y dan una guía en cuanto a los fines que se siguen con cada uno de ellos.

Si bien, TSP nos presenta un marco de trabajo bien definido y claramente estructurado, resulta ser lo suficientemente flexible para poder incorporarse en casi cualquier proyecto de software. Por consiguiente una herramienta que apoye al conjunto de técnicas utilizadas en el proceso de desarrollo deberá permitir al equipo la incorporación de sus propios documentos y productos generados durante los ciclo de desarrollo.

El trabajo de Humphrey [WSH 2000], nos señala aspectos que debemos seguir en lo general, pero es importante recordar que cada uno de los proyectos de

software que se generan cuenta con sus características y necesidades propias para cumplir con sus objetivos y necesidades particulares.

Teniendo en mente lo anterior, la herramienta deberá integrar las siguientes características para su fácil adopción y puesta en funcionamiento por parte de un equipo de trabajo:

- Interfaz gráfica de usuario
- Opciones de impresión de formas y documentos
- Validación de la información siguiendo las consideraciones de cada forma
- Tiempos de capacitación breves
- Registro confiable de productos
- Consulta a través de la red de los documentos disponibles
- Trabajo colaborativo del equipo de desarrollo

Cualquiera de los elementos anteriores ofrece un conjunto de opciones y consideraciones para la operación de la herramienta. Es importante recalcar que el objetivo de la herramienta es el de facilitar la incorporación de la misma, a fin de ayudar a sus usuarios a entender y aprovechar TSP en el desarrollo de proyectos de software.

Por su parte la herramienta establece varios tipos de usuarios para su operación, tanto para la definición de proyectos, como de los roles de cada uno de ellos. Los tipos de usuarios definidos por la herramienta son los siguientes:

- Administrador del sistema
- Administrador de proyecto

A continuación se describen las cualidades de cada uno de ellos.

Administrador del sistema

Este usuario estará encargado de las actividades de administración de la herramienta, no necesariamente del proceso de algún proyecto. Debe quedar claro que las actividades del Administrador del sistema no se describen en TSP, sino más bien se establece para definir un responsable de las propias operaciones de la herramienta.

Dichas actividades de forma puntual son las siguientes:

- Alta, bajas y cambios de los colaboradores, es decir de los miembros de los equipos de desarrollo. Ver Figura 7.

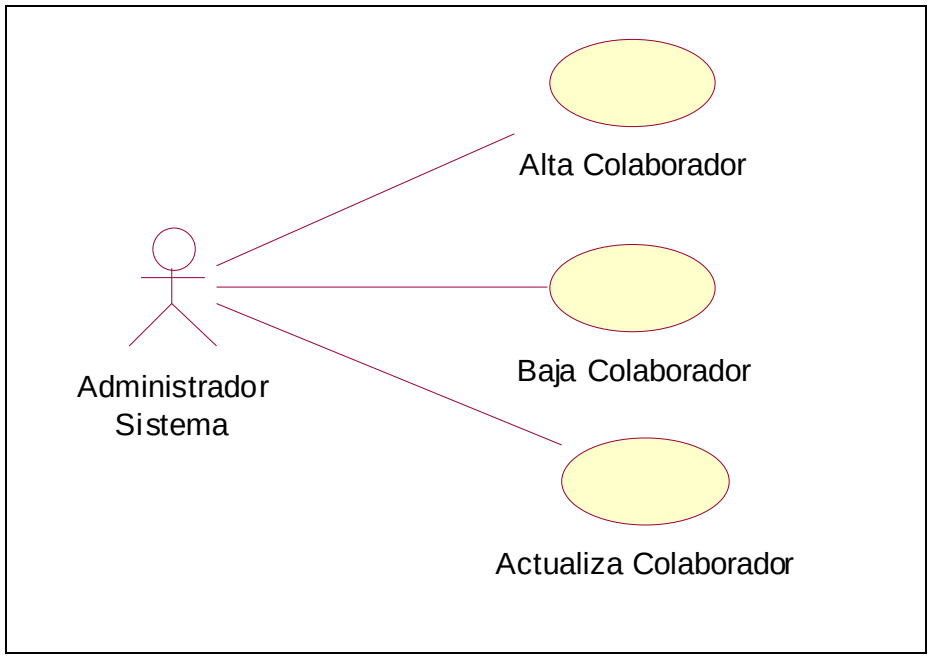


Figura 7 – Casos de uso del administrador del sistema

- Alta, bajas y cambios de los proyectos a desarrollar. Esto se lleva a la par de la asignación de roles para el mismo proyecto. Ver Figura 7.
- Puede generar un reporte sobre los desarrolladores de un proyecto de software. Ver Figura 8.

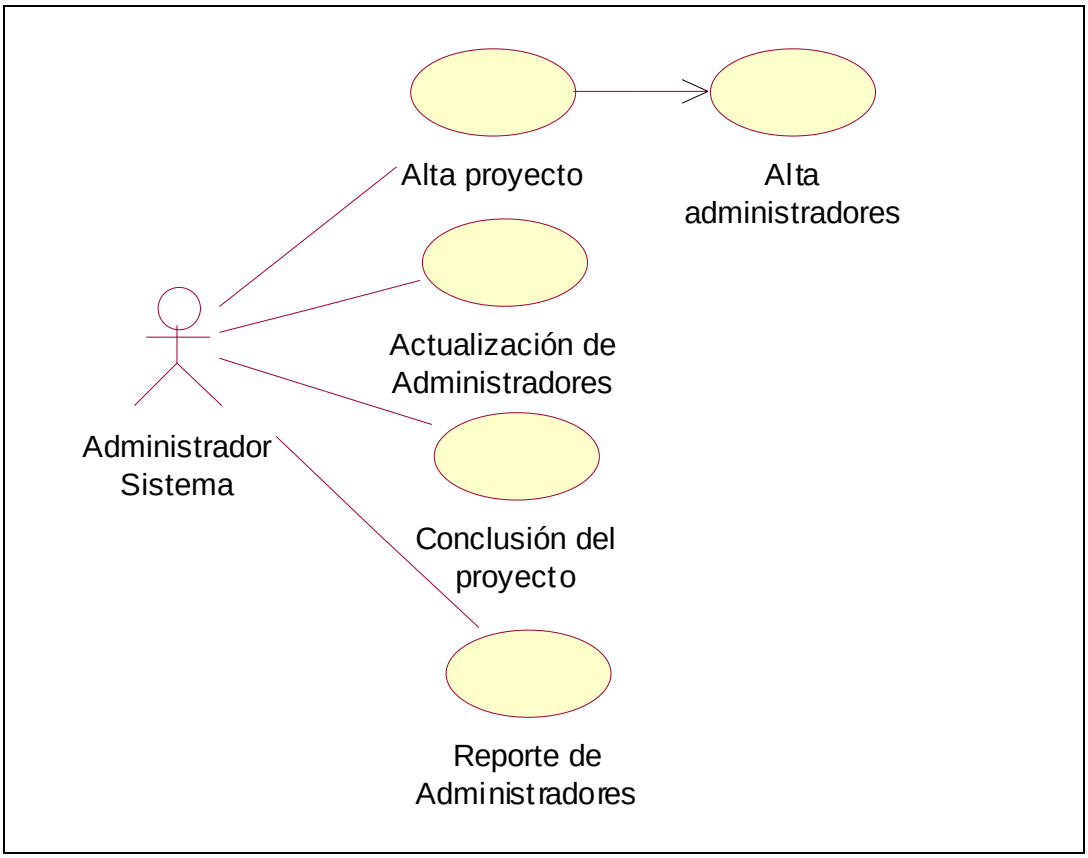


Figura 8 – Casos de uso del administrador del sistema

Administrador de proyecto

Como puede apreciarse en la Figura 9, al dar de alta un proyecto se dan de alta Administradores. Dichos administradores son los que TSP define y que se han venido explicando en el presente trabajo.

En resumen, la herramienta requiere de una división del trabajo con base en los roles definidos por TSP y que se van incorporando al proyecto, de esta forma desde un inicio se definen los integrantes del equipo y los roles que cada uno de ellos tienen asignados para cada proyecto.

4.3 Seguimiento del trabajo colaborativo con la herramienta

Una vez que la herramienta es adoptada por los desarrolladores de cada proyecto de software, comienza la generación de reportes y es aquí donde el seguimiento de los proyecto comienza a registrarse en la herramienta.

Las actividades de cada uno de los roles en TSP exige que los administradores generen, en primer lugar, un plan de trabajo conjunto. Dicho plan deberá seguirse y podrá ser consultado por todos los integrantes del equipo. A la par del

desarrollo, cada administrador llena su parte de la forma WEEK y van dando lugar al registro de sus logros y los alcances de sus actividades en el proyecto. Esto último permita la verificación del estado del proyecto con respecto a lo propuesto en el plan de trabajo. Esta interacción se resume en la Figura 9.

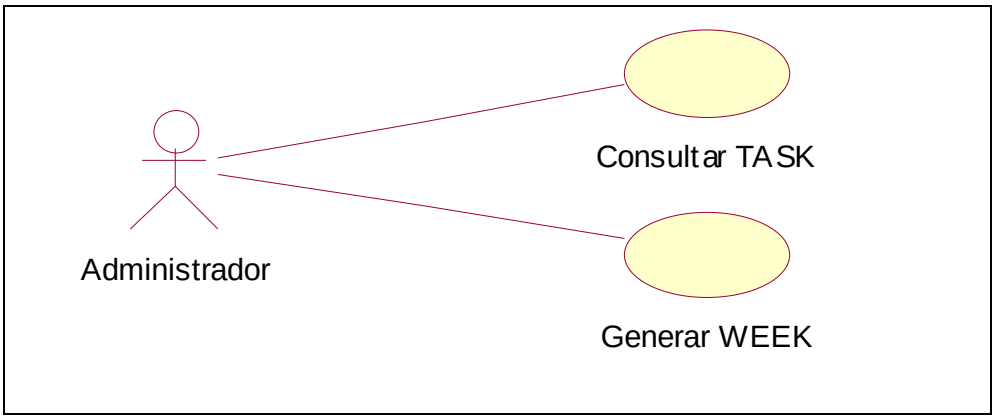


Figura 9 – Casos de uso para el seguimiento de proyectos

Para el proceso de implementación tanto el Administrador de desarrollo como el Administrador de calidad requieren de la revisión de las formas LOGT que cada uno de los ingenieros van generando. En TSP la división del trabajo trae consigo la generación de diversos productos, mismos que deben ser planificados y reportados, además de indicar la división de tareas para el proceso mismo.

Si bien dichas tareas son definidas por cada uno de los administradores de TSP, es necesario que dicha forma pueda ser consultada y verificada por la herramienta para su seguimiento y control.

4.4 Administración de documentos y formas

Cada una de las formas propuestas por TSP cuenta con sus reglas y consideraciones para cumplir con su objetivo. Los puntos de interés ha reflejarse en la herramienta se listan a continuación:

- Validación de todas las formas, con base a las especificaciones establecidas en TSP. Ver Figura 10.

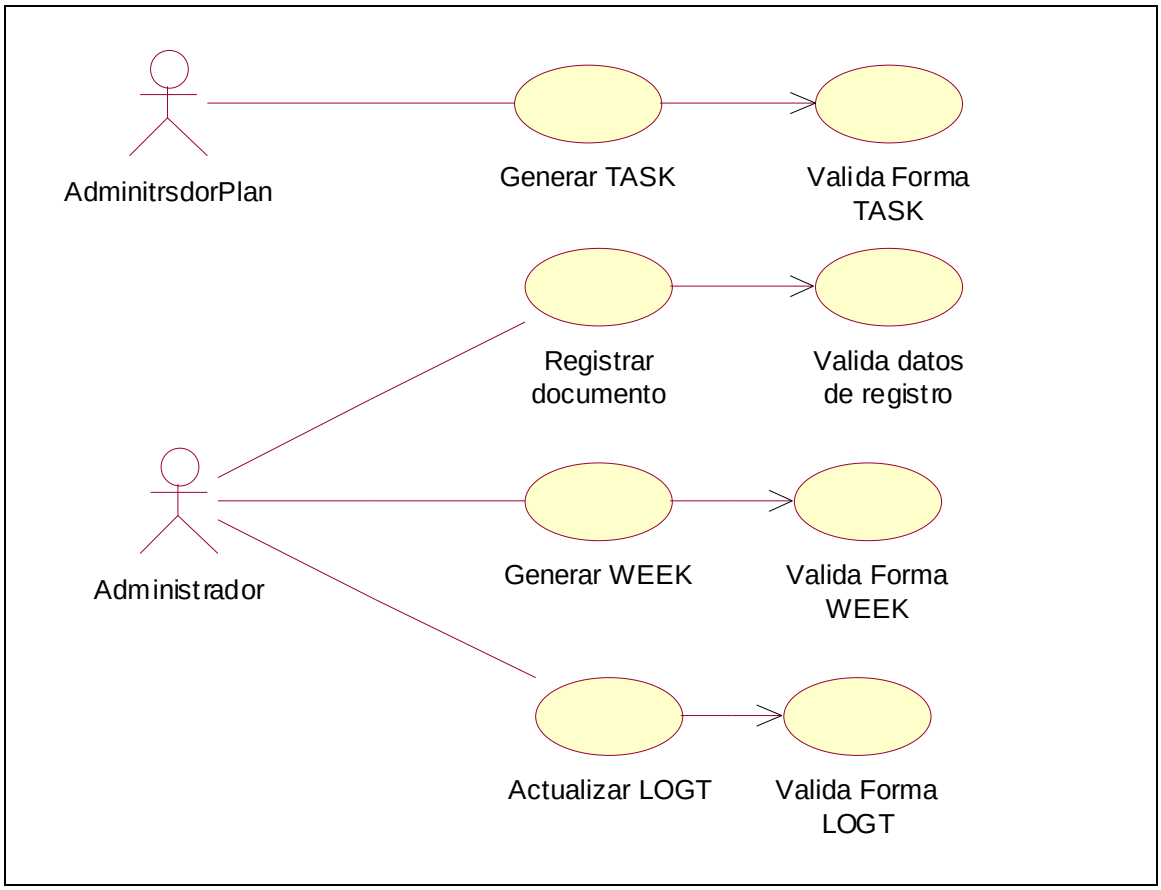


Figura 10 – Casos de uso del Administrador de planeación y participantes

- Consulta de las formas por cada uno de los administradores del proyecto
- Mantenimiento de versiones y cambios de productos y documentos mediante la herramienta
- Manejo de cualquier tipo de documentos y productos generados en el proyecto
- Almacenamiento de productos y documentos

Es decir, se contará con la capacidad de alojar todos aquellos productos y documentos que el equipo considere necesarios para el proyecto que está desarrollando.

En el siguiente capítulo se presenta la herramienta tanto a niveles técnicos así como los funcionales que ofrece para su buena utilización e incorporación a los ciclos de producción propuestos en TSP.

Capítulo 5

Diseño y desarrollo de la herramienta HACET[1]

Como se ha venido presentando en los capítulos anteriores, la herramienta tiene como objetivo principal apoyar el proceso de desarrollo TSP. En la actualidad no existen herramientas adecuadas que apoyen el proceso, o bien, que cuenten con las cualidades prácticas y útiles para su fácil incorporación en los procesos de producción de equipos de desarrollo. Esta carencia de herramientas puede ser el resultado de que lo importante es el proceso en sí, y no tanto los elementos tecnológicos que puedan apoyar las actividades y las tareas definidas por él para un proyecto de software.

James W. Over, siguiendo las especificaciones de Humphrey, desarrolló una herramienta denominada ***Support Tool***, para el manejo de formas y validación de datos. Dicha herramienta consta de un conjunto de hojas electrónicas elaboradas con Microsoft Excell ®. Ver Figura 1

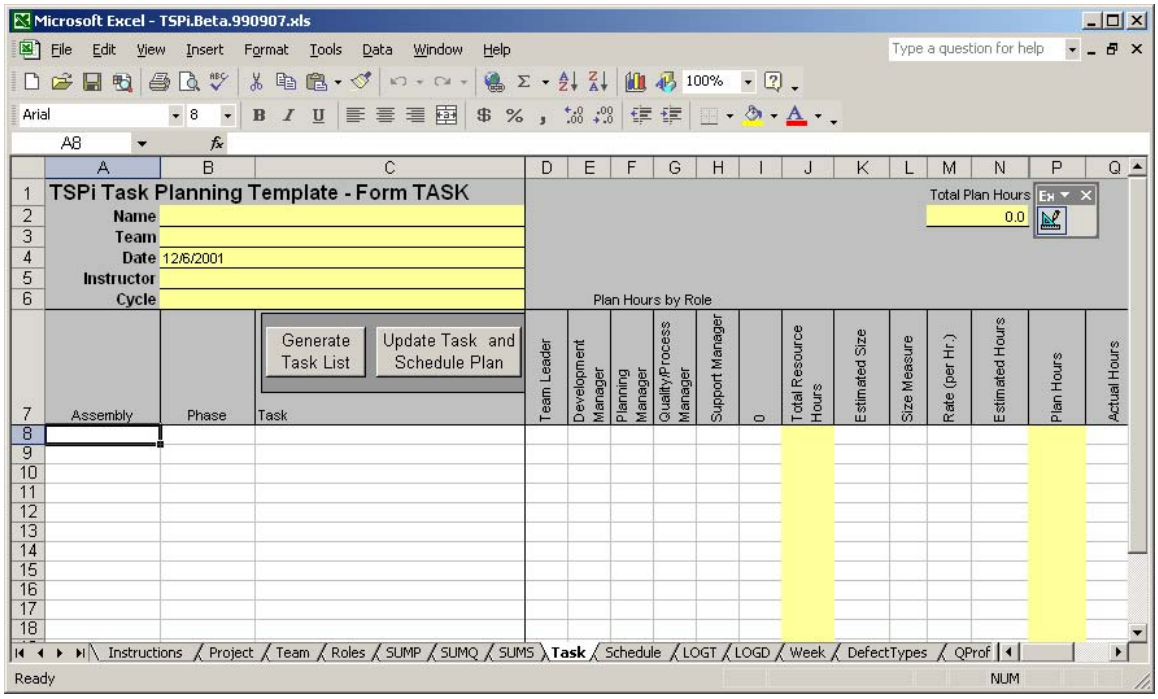


Figura 1 – Support Tool

El principal objetivo de la herramienta de Over es el de apoyar el curso de capacitación para TSP, por lo que se omiten cualidades funcionales y técnicas deseables en proyectos de software reales, tales como el manejo de datos

[1] Abreviatura de Herramienta para la Administración de la Configuración para Equipos de Trabajo

personales de los integrantes de cada equipo así como los roles que desempeñan, el trabajo colaborativo, el manejo de varios proyectos de forma automatizada, el almacenaje de documentos y otros productos del desarrollo.

En el presente capítulo se definen los requerimientos de una herramienta para TSP además de su diseño e implementación. Finalmente se evaluarán las cualidades de la herramienta generada a fin de validar que las mismas apoyan el proceso definido por TSP.

5.1 Requerimientos funcionales de la herramienta HACET

La herramienta que se propone en este trabajo, llamada HACET (Herramienta para la Administración de la Configuración para Equipos de Trabajo), tiene por objetivo:

- Apoyar todo el proceso propuesto por TSP con sus formas y productos.
- Llevar el control de versiones de las formas TASK, LOGT y WEEK, y demás documentos generados por el desarrollo
- Automatizar la mesa de control de cambios (MCC) facilitando la generación de la forma CCR
- Capacidad de extender su funcionalidad acorde a las necesidades particulares de los equipos de desarrollo

Con base en las observaciones realizadas en el Capítulo 4, una herramienta que apoye las actividades y tareas para un proyecto usando TSP, deberá satisfacer los siguientes puntos:

- Administración de los datos de los ingenieros de software
- Administración de proyectos de software
- Administración de documentos generados en cada proyecto
- Administración de roles por proyecto
- Administración de la configuración: control de versiones

Dichos puntos se describen a continuación.

Administración de los datos de los ingenieros de software
Aunque esta especificación es necesaria y al mismo tiempo básica, se debe ofrecer un mecanismo de identificación y recuperación de datos personales de los miembros de cada grupo de desarrollo, tales como nombre completo, su teléfono, e-mail, dirección, entre otros; a fin de tenerlos siempre disponibles para su consulta durante la vida del proyecto.

Administración de proyectos de software
El núcleo central de la herramienta serán los proyectos de software tal y como se mencionó en el capítulo anterior. Parte del diseño será el de asociar un

proyecto de software con el grupo de desarrollo que le dará seguimiento. A cada uno de los miembros del equipo se le denominará **colaborador** para la herramienta.

La asociación anterior no se resumirá meramente en generar y mantener la lista de los colaboradores en el desarrollo de cada proyecto, sino que además quedarán registrados los roles de cada uno de ellos, según la especificación de TSP. La Figura 2 resume la relación entre proyectos, roles y colaboradores que la herramienta utiliza.

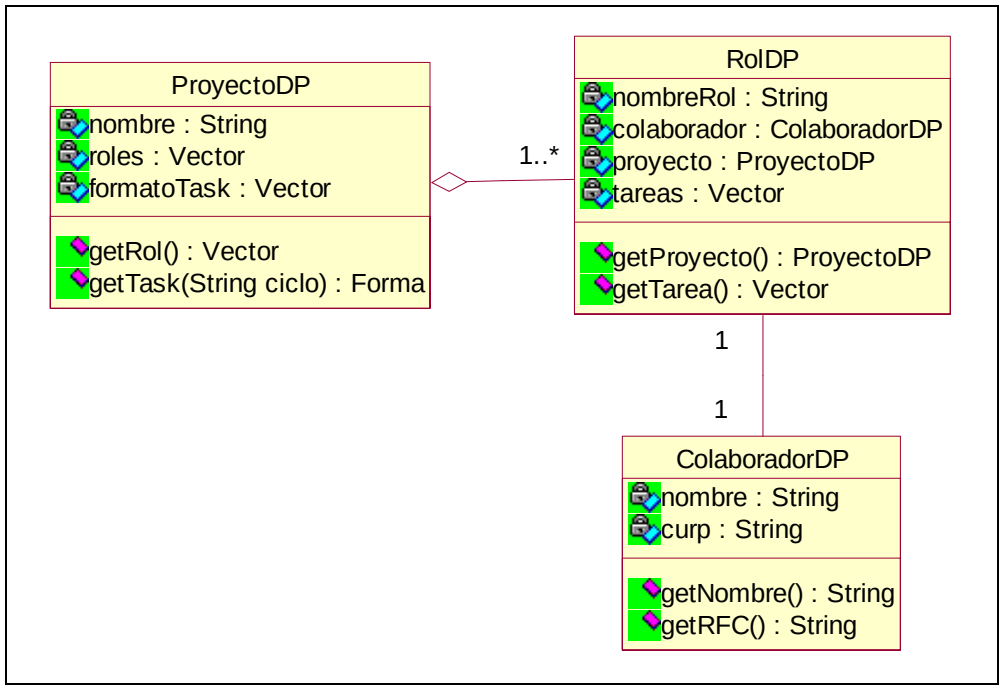


Figura 2 – Diagrama de clases para la relación Proyecto-Rol-Colaborador

Ahora bien, la herramienta que se ha definido tiene por objetivo apoyar las tareas derivadas de TSP, por lo que los roles son los mismos que se definen en dicho proceso, pero cuenta con la opción de incluir nuevos a un proyecto. Ver Figura 3.

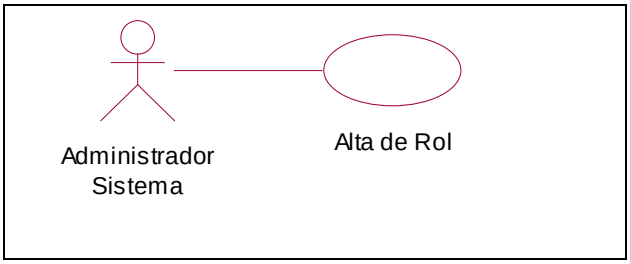


Figura 3 – Caso de uso de alta de Rol

Administración de documentos generados en cada proyecto
En TSP ya existe una amplia variedad de documentos que deben generarse por los integrantes del equipo. La herramienta por lo tanto debe contemplar la integración de las mismas en sus opciones de administración. Cada formato contempla una serie de datos que se deben de proporcionar a fin de identificar varios aspectos de interés, tales como el administrador que genera los documentos, el nombre del equipo, el proyecto del cual se genera el proyecto, fecha de creación, el ciclo de desarrollo en el que se generó el documento, entre otros.

Una forma práctica de reunir los elementos de dichos documentos es definiendo una interfaz para garantizar el funcionamiento e integración de las mismas aún para documentos nuevos que requieran ser incorporados. La Figura 4 muestra la relación de la interfaz **Forma** con los documentos que definen cada uno de las formas de TSP.

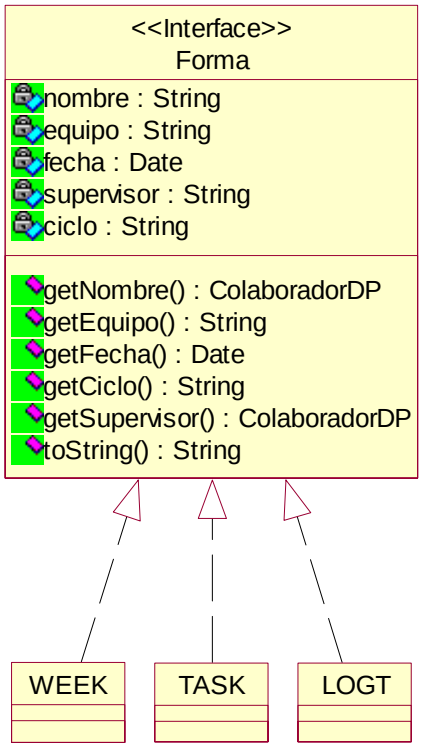


Figura 4 – Interfaz Forma

Este mecanismo permite la integración de nuevos formatos con sólo implementar la interfaz. La ventaja de este hecho permite que cada uno de los documentos determine su propia línea de herencia.

Para el caso de documentos no contemplados por TSP, pero adoptados por el equipo de desarrollo, se permite la trasmisión de éste al servidor, a fin de registrarlo y tenerlo disponible por la herramienta.

Como se presentó en el capítulo anterior, los datos que se deben mantener para cada documento o producto de software del proyecto para la herramienta serán los siguientes:

- Identificador
- Nombre
- Ciclo
- Número de versión
- Autor
- Tipo de documento
- Estado
- Resumen
- Proyecto
- Equipo

La Figura 5 resume la relación que se emplea en la herramienta:

materiales		
PK	**material_id**	**LONG**
FK1	**material_proyecto_id**	**LONG**
FK2	**material_elaboro_id**	**LONG**
	material_nombre	**TEXT(50)**
	material_ciclo	**CHAR(10)**
	material_version	**TEXT(30)**
	material_tipo	**VARCHAR(20)**
	matrial_estado	**VARCHAR(20)**
	material_formato	**TEXT(30)**
	material_fecha_in	**DATETIME**
	material_archivo	VARBINARY(255)

Figura 5 – Relación para los datos de un producto

Administración de Roles por proyecto
Si bien la asignación roles se encuentra relacionada con cada proyecto y el equipo de trabajo, se debe contar con un registro de la misma. Con ello se deja registro de las actividades encargadas a cada miembro del equipo de desarrolladores.

Administración de la configuración: control de versiones
Para contar con una herramienta confiable es necesario que HACET cuente con las opciones de control de versiones sobre los productos que se generan, siendo estos almacenados en un espacio del sistema y permitiendo su consulta y recuperación para todos los miembros del equipo.

La forma de transferencia será por la red, por lo que estos datos son indispensables para su posterior recuperación con la herramienta.

En la siguiente sección se presenta la arquitectura de desarrollo de la herramienta HACET.

5.2 Especificación de la arquitectura de HACET

Para la organización de los distintos elementos de la herramienta y su implementación se utiliza la arquitectura J2EE (Java 2 Plataform Enterprise Edition) propuesta por Sun Microsystems para el desarrollo de aplicaciones distribuidas. La Figura 4 resume la especificación de dicha arquitectura.

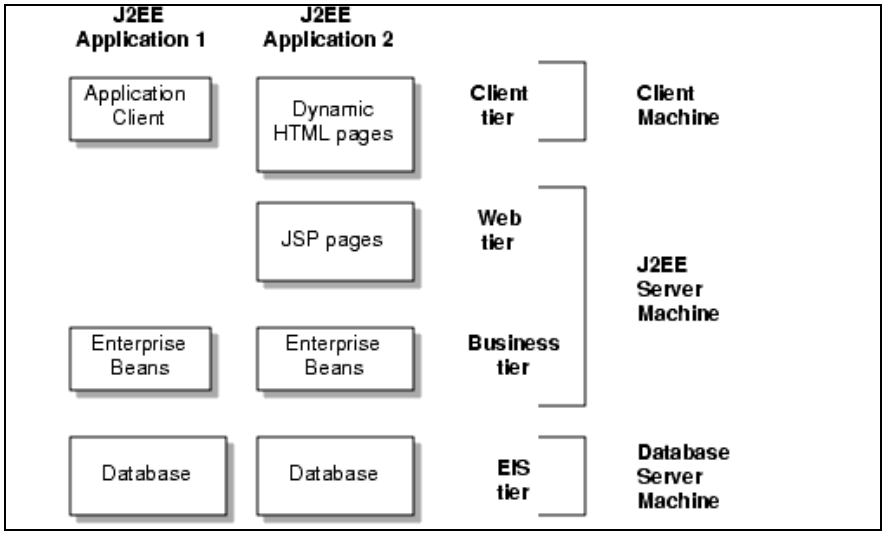


Figura 6 – Arquitectura Java 2 Plataform Enterprise Edition

Como parte de la implementación de la arquitectura anterior, se utilizó Apache Tomcat en sus versiones 4.x, 5.x y 6.x.

Con base en la definición de la especificación del J2EE de SUN, la arquitectura de HACET estará conformada en tres capas: **Capa de cliente**, **Capa de lógica del sistema** y **Capa de base de datos**. Dicha propuesta resulta ser una especialización del patrón de diseño **Modelo-Vista-Controlador**, donde cada capa representa cada uno de los componentes de dicho patrón. La Figura 7 muestra la estructura general de cada elemento de dicho patrón.

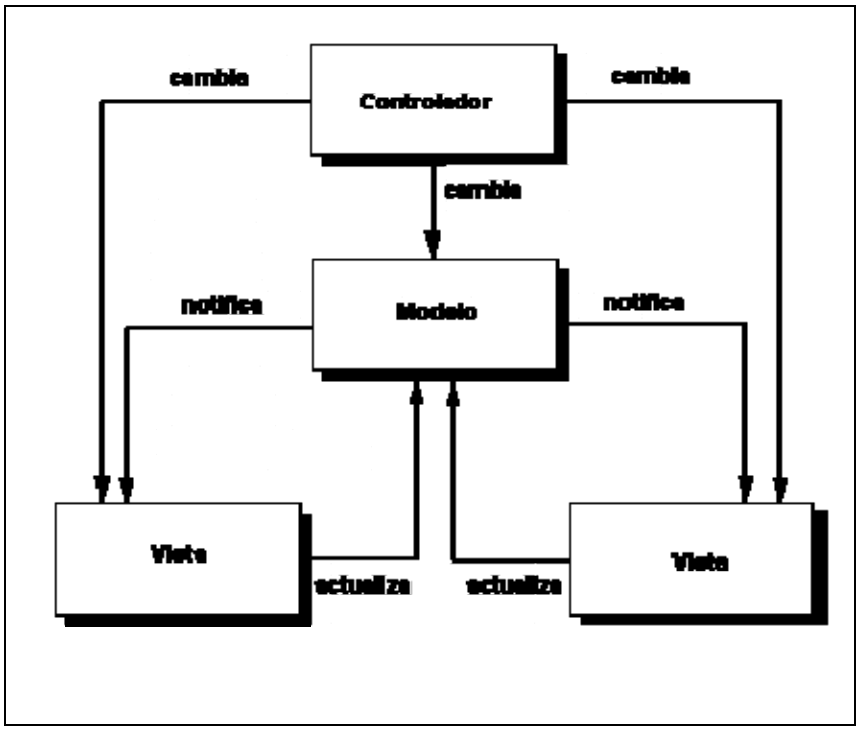


Figura 7 – Arquitectura de MVC

En el caso particular de HACET, la implementación del patrón se conforma de en un conjunto paquetes, las cuales alojan clases, interfaces y otros elementos que las integran.

Los paquetes de la **Capa de cliente** vienen a conformar la **Vista**, es decir, al conjunto de objetos que permiten al usuario observa la información en cada instante. Con esta concepción podemos generar clientes gráficos o modo texto sin que la funcionalidad se sujete a ésta. Clases que pertenecen a esta capa tenemos a **ClienteAdmin** y **Client** del paquete **atr.cliente**, donde el primero es un cliente gráfico y el segundo es modo texto.

Los paquetes de la **Capa lógica** conforman al **Controlador,** es decir, se integra por aquellos paquetes que implementan las reglas del negocio. En esta capa se encapsula la funcionalidad real del sistema además de que se define un conjunto de servidores para que los clientes se conecten a estos últimos y se definan los criterios de seguridad y solicitud de manipulación de los datos. Clases que pertenecen a esta capa tenemos a **Server** y **Servicio** del paquete **atr.servidor**, los cuales integran al servidor principal de HACET.

Los paquetes del **Capa de base de datos** definen al **Modelo**, es decir, se conforman de los paquetes que tienen interacción con la base de datos. En esta capa se efectúan las operaciones de modificación y consulta de la base de datos. Clases que pertenecen a esta capa tenemos a la clase **ManejaDB**, cuyos objetos permiten la manipulación de la base de datos.

Los paquetes anteriores se resumen en la Figura 8.

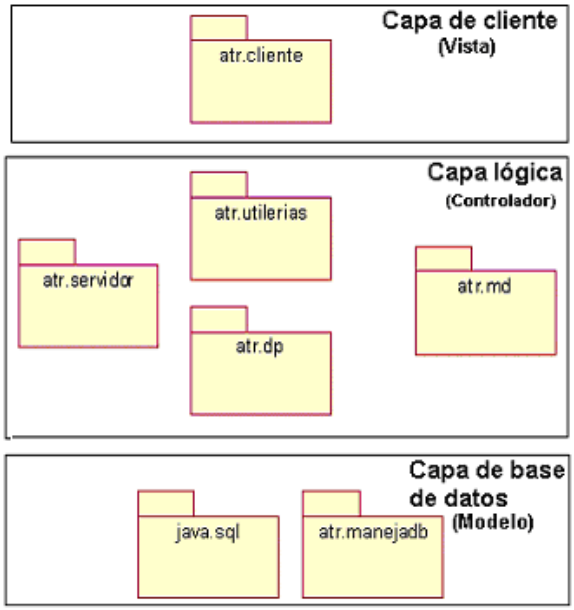


Figura 8 – Paquetes de HACET

La comunicación entre la capa lógica y la capa de base de datos se define con base en un segundo patrón denominado **Command** (Instrucción). La Figura 9 muestra a detalle este último patrón.

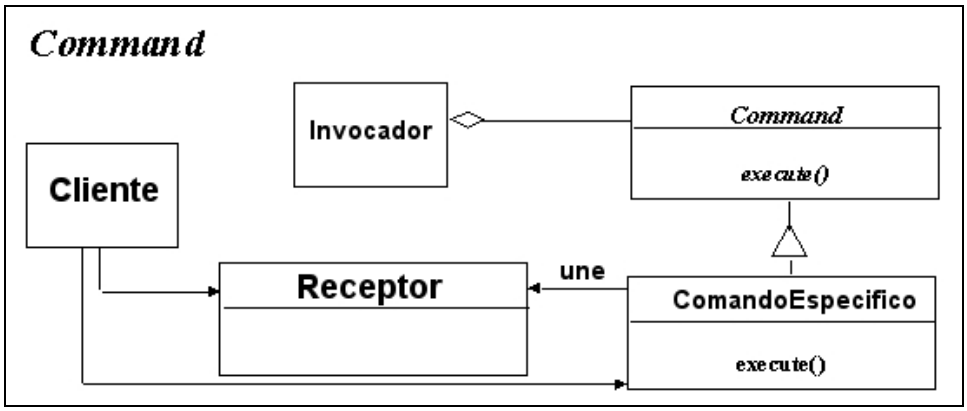


Figura 9 – Patrón Command

Las actividades a desarrollar son las siguientes:

1. El cliente crea un objeto de la clase **ComandoEspecifico** y lo une a él con un **Receptor**.
2. El cliente entrega el **ComandoEspecifico** al **Invocador**, el cual lo alacena.
3. El **Invocador** tiene la responsabilidad de realizar la instrucción.

En la siguiente sección se indicarán como es que cada uno de los patrones se implementaron para la generación de la herramienta.

Aunque la complejidad que se describe hasta el momento podría suponer que la distribución de HACET estará conformada por un amplio número de elementos, esta se ha simplificado en tres elementos claramente identificados: Servidor Tomcat, la base de datos y un equipo en Internet. Ver Figura 10.

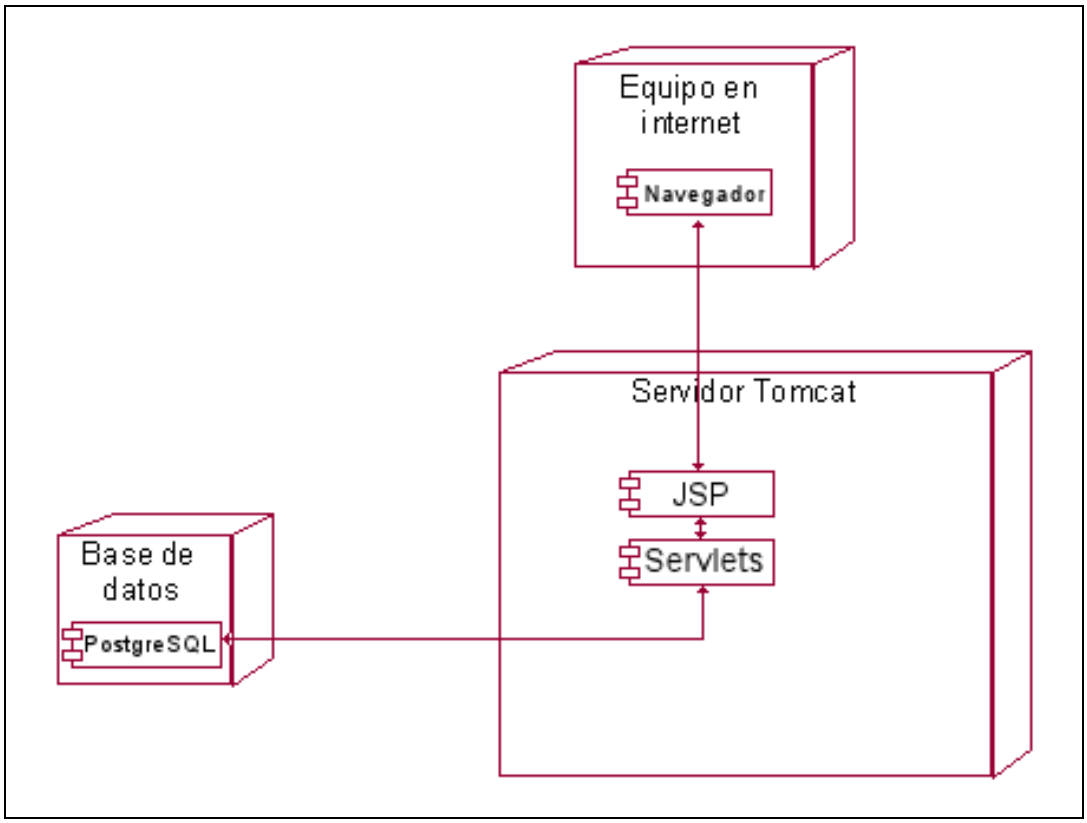


Figura 10 – Diagrama de distribución de Hacet

5.3 Implementación

La implementación de HACET se desarrolló a tres niveles: clientes, servidores y base de datos, según la especificación del J2EE de SUN. A continuación se presentan las consideraciones de implementación de la herramienta.

Clientes

Los clientes tienen como objetivo enviar instrucciones a los servidores de la capa lógica. Como se definió anteriormente, los clientes se van a implementar bajo el patrón **Command**, en el caso particular de HACET, las actividades que desarrolla un cliente serán las siguientes:

1. El cliente crea un objeto de la clase **Instruccion**, según la opción que haya utilizado el usuario.

2. El cliente envía el objeto anterior al servidor.
3. El servidor evalúa la operación.
4. El servidor recibe un objeto **Resultado** como respuesta a la instrucción enviada
5. El cliente procesa dicho resultado.

Un ejemplo concreto de esta funcionalidad se muestra en la Figura 11, con el uso de la instrucción **Insertar colaborador**.

El usuario utiliza la interfaz gráfica en este caso, por lo que las operaciones de modelo delegacional de Java se implementan, es decir, los métodos **actionPerformed** se implementan en cada clase.

Cuando el usuario interactúa con el **ClienteAdmin** se crea en ese instante un objeto Instrucción y se utiliza como parámetro del método enviaInstrucción que regresa un objeto **Resultado**. Cualquiera que sea el resultado se despliega una ventana de diálogo que indica el resultado de la operación. En caso de ser correcta la operación los datos se borran llamando al método **limpiaCampos**.

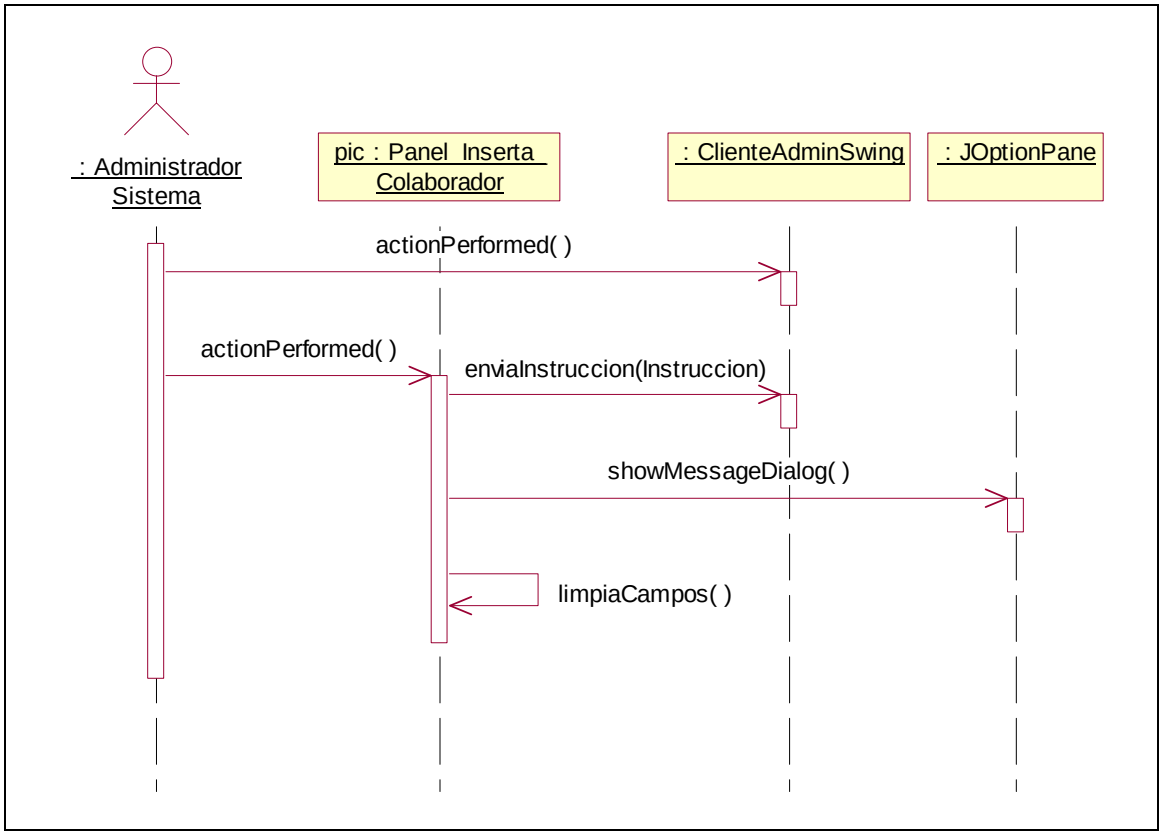


Figura 11 – Diagrama de interacción para insertar un coalborador

Cabe señalar que la implementación del patrón **Command** incluye la definición de un protocolo de red, por lo que el método **enviaInstruccion** encapsula dicha

funcionalidad. Para el caso del cliente modo texto, la lógica de este último método se implementa tal cual.

Servidores

Los servidores implementados incorporan la parte del patrón **Command** que les corresponde, además de efectuar las llamadas a la base de datos según la instrucción que se ha solicitado.

Las actividades que se requieren para cumplir con el patrón **Command** son las siguientes:

1. Establecer una conexión con el cliente.
2. Recibir una instrucción.
3. Validar al usuario y los parámetros de la misma.
4. En caso de ser una instrucción válida se procesa.
5. Se genera un objeto Resultado que refleje la condición de la instrucción procesada.
6. Envía el objeto del punto anterior al cliente.

Los primeros 4 pasos se muestran en la Figura 12.

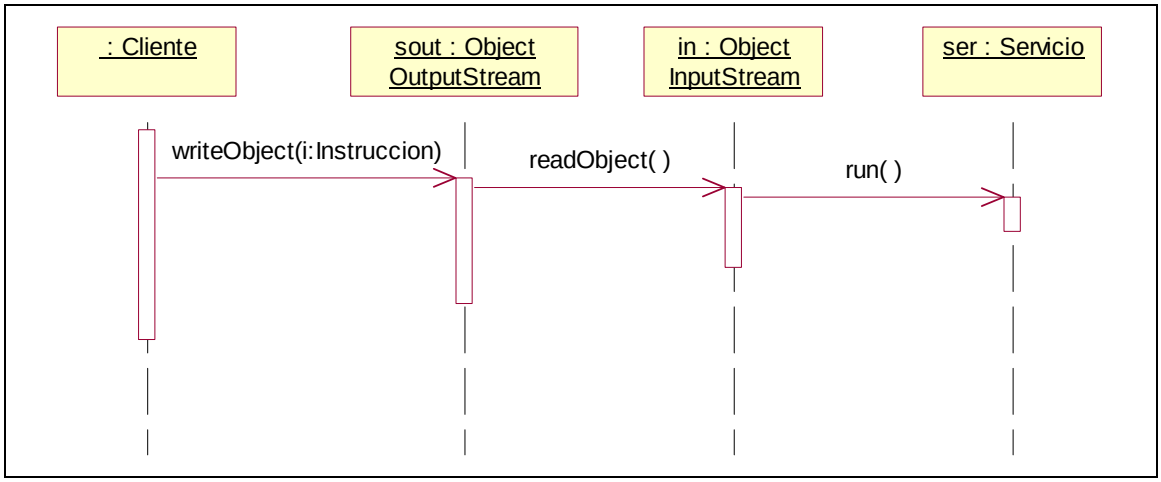


Figura 12 – Diagrama de interacción para enviar instrucción

Los pasos 5 y 6 se muestran en la Figura 13.

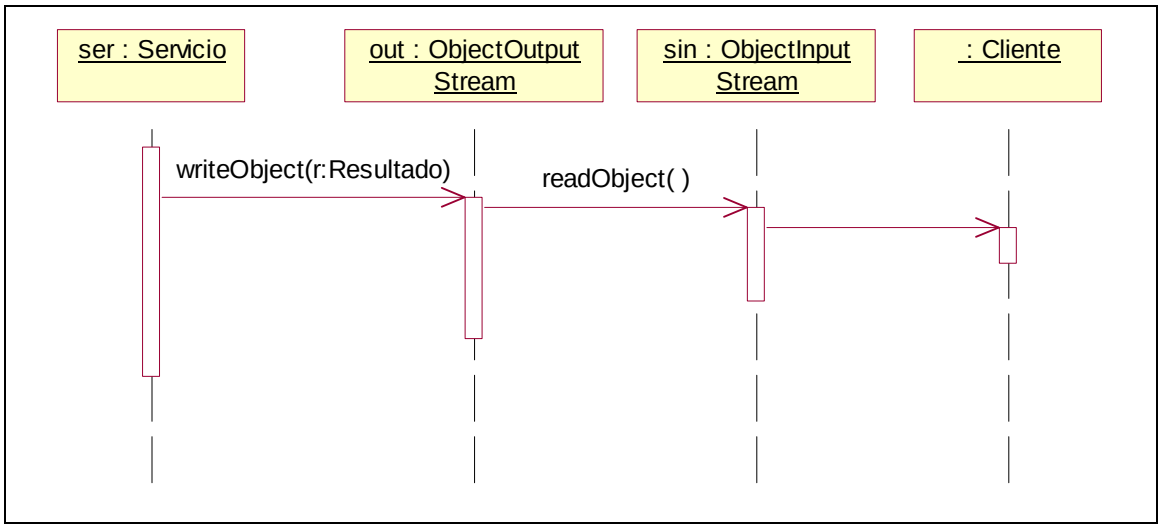


Figura 13 – Diagrama de interacción para enviar un resultado

Para cada caso de uso definido para la herramienta se han desarrollado un conjunto de instrucciones atómicas, por lo que las instrucciones programadas alcánza la cifra de 80 y se tienen al momento unos 20 mensajes de operación.

Base de datos

Para la base de datos se definió un componente de software para la explotación de la misma. La clase desarrollada es **ManejaDB**, cuya funcionalidad se muestra en la Figura 14.

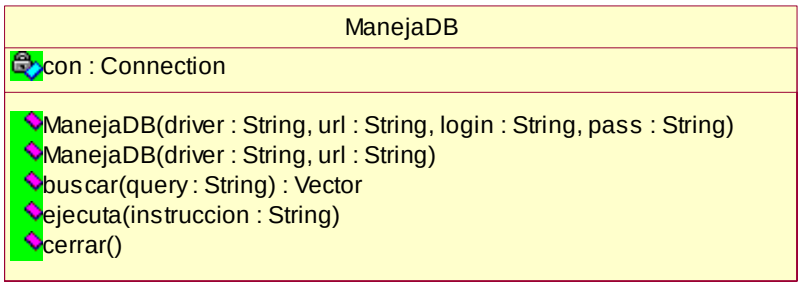


Figura 14 – La clase ManejaDB

Los métodos constructores utilizan las especificaciones del JDBC™ de SUN.

En caso de una consulta utilizando un objeto de esta clase, se utiliza el método **busca()** que recibe la instrucción SELECT particular para el manejador de base de datos que la procesará. Si existen resultados en la base estos se regresan en un **Vector** que contiene la información de cada registro representado por un objeto **Hastable**.

Si se desean efectuar operaciones de alteración a la base de datos se utilizará el método **ejecuta()**, cuyo parámetro es la instrucción a ejecutarse en la base.

El método **cerrar()** cierra la conexión con la base de datos.

Ahora bien, como un resultado de la arquitectura de desarrollo, la independencia conceptual de cada una de las capas permitió también la independencia de plataforma y de manejadores de bases de datos.

La definición de la base de datos adecuada para la funcionalidad de la herramienta se muestra a continuación en las Figuras 15 y 16.

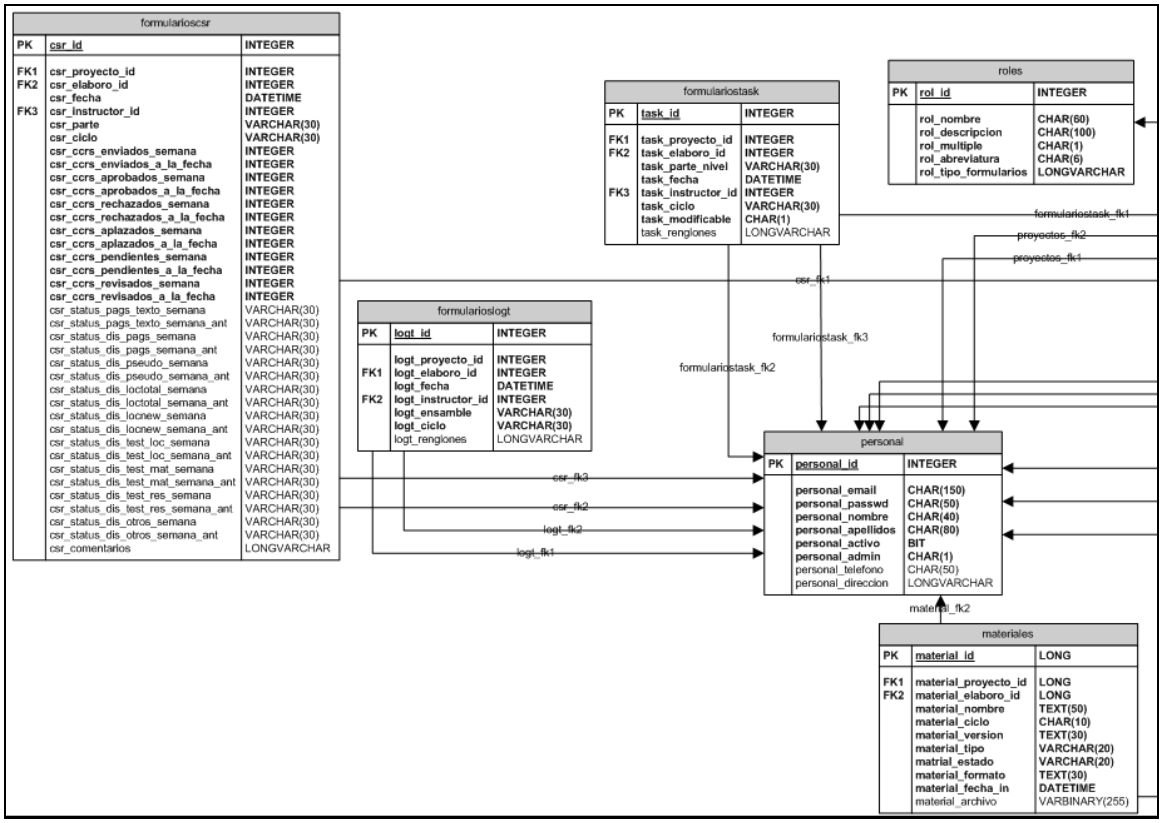


Figura 15 – Diagrama relación entidad (parcial)

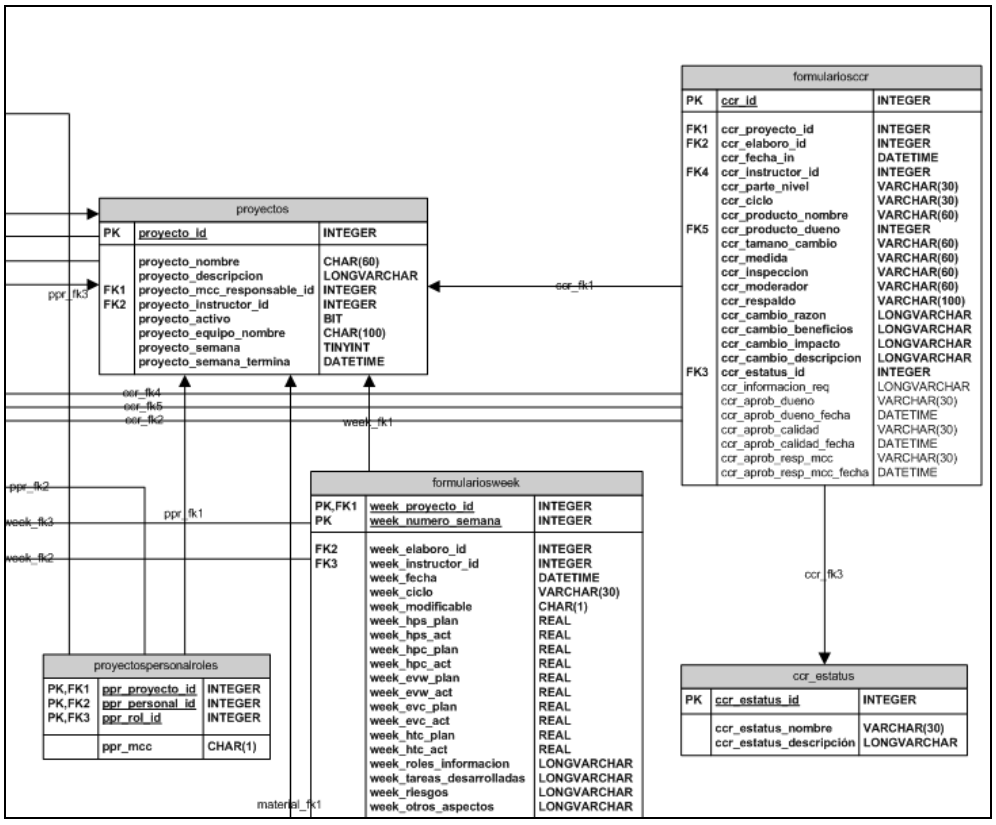


Figura 16 – Diagrama relación entidad (parcial)

La herramienta se ha programado actualmente para dos bases de datos: PostgreSQL versión 8.2 y Microsoft® Access™. Para incorporar la funcionalidad de la herramienta con una nueva base de datos se requieren de dos condiciones básicas:

- Contar con el controlador de JDBC™ para la base de datos en particular
- Construir el espacio de la base de datos de la herramienta respetando los nombres, tipos de datos y relaciones entre las tablas que se presentaron previamente.

La Figura 17 muestra la disposición las tablas en Microsoft® Access™.

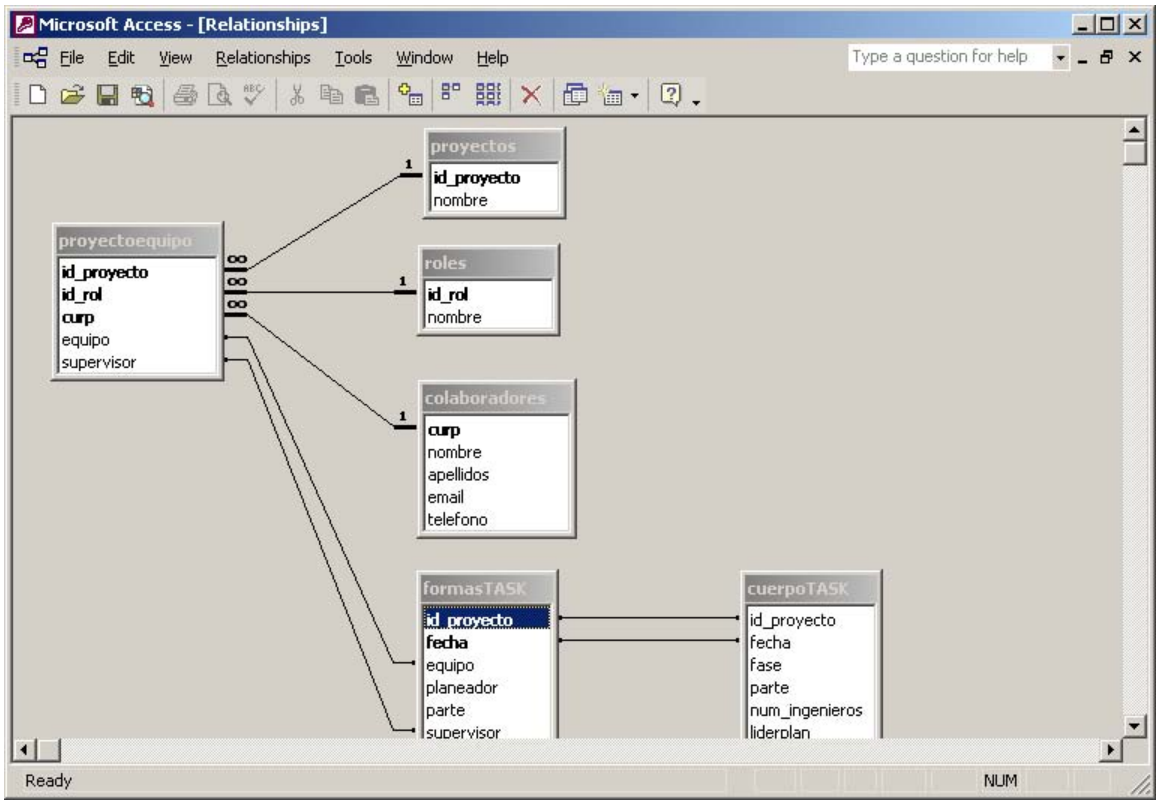


Figura 17 – Implementación de la base de datos con Microsoft™ Access© (Parcial)

5.4 Descripción de los módulos de la aplicación

A fin de evaluar a HACET como una herramienta que cumple plenamente con las especificaciones presentadas hasta el momento, se presentarán las características en los siguientes rublos:

- Opciones del administrador
- Generación de la forma TASK
- Generación de reportes Week
- Generación de la bitácora LOGT

Opciones del administrador
La pantalla principal de HACET se presenta en la Figura 18.

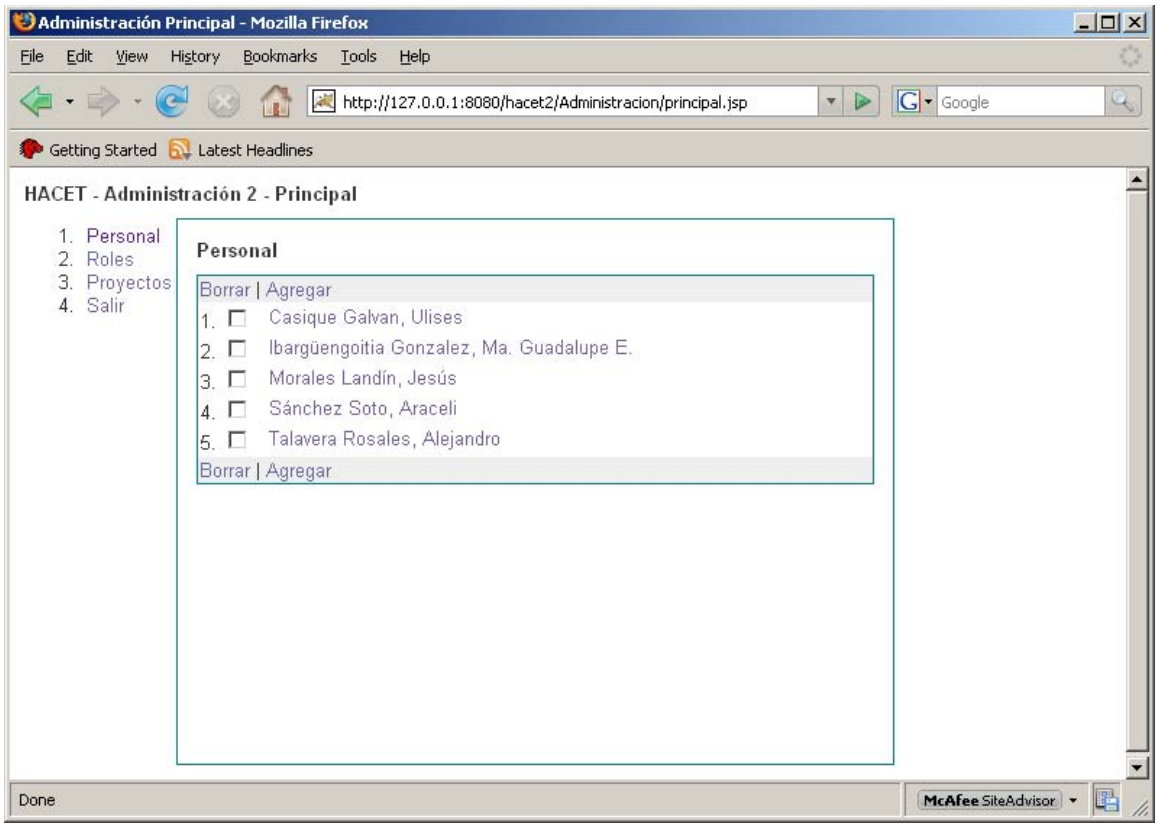


Figura 18 – Pantalla principal para el administrador de HACET

Un administrador de HACET tiene a su cargo las siguientes funciones de altas, bajas y cambios para colaboradores, proyectos y roles.

Para dar de alta un colaborador, por ejemplo, HACET cuenta con su propia forma, tal y como se muestra en la Figura 19.

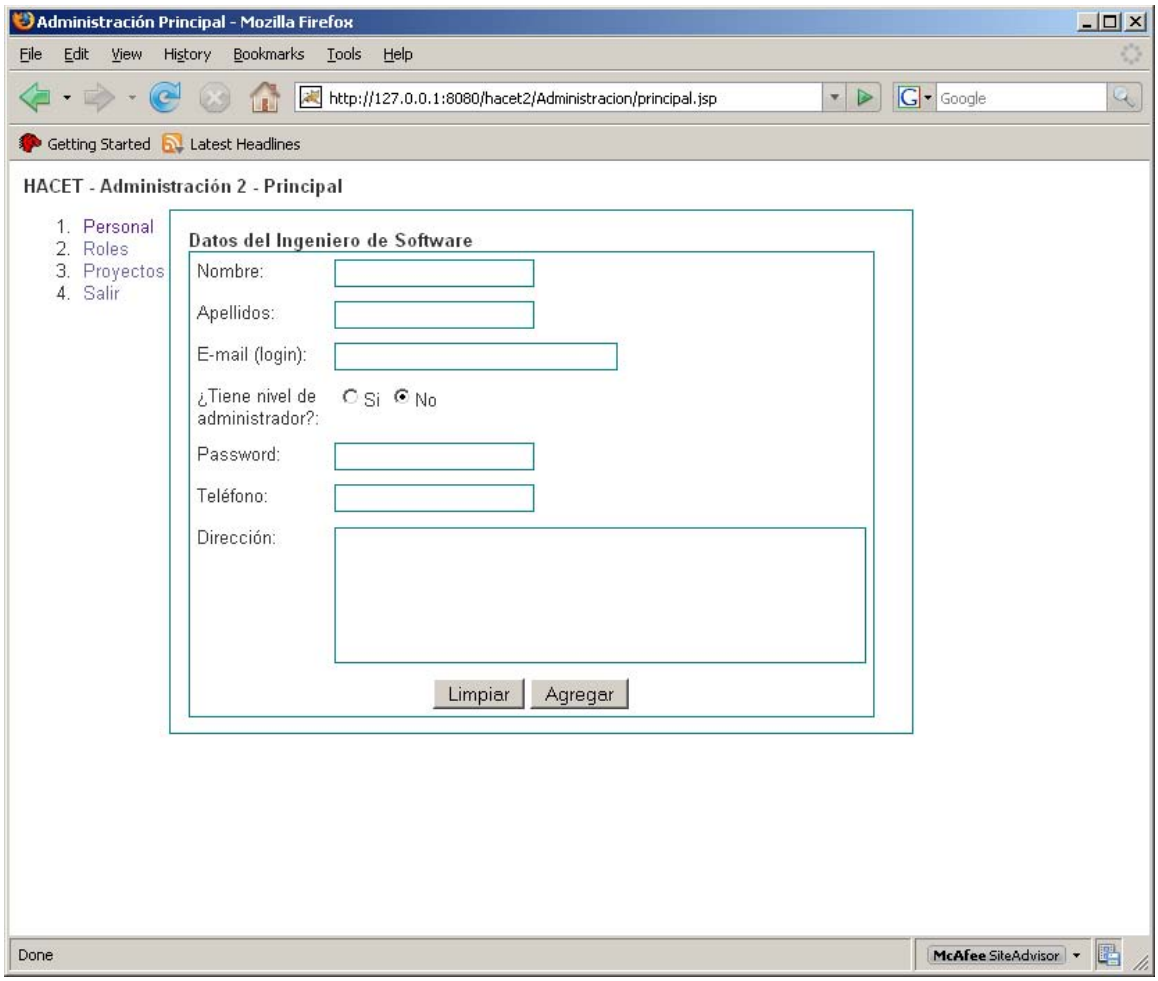


Figura 19 – Forma para alta de colaborador

Si bien las opciones de un administrador de HACET resultan pocas con respecto a los roles, esto se debe a que participa a que no tiene un papel definido en el proceso de producción. Aunque con lo anterior de él dependen tareas como la asignación de roles en los proyectos que maneja el sistema. La Figura 20 muestra esta opción.

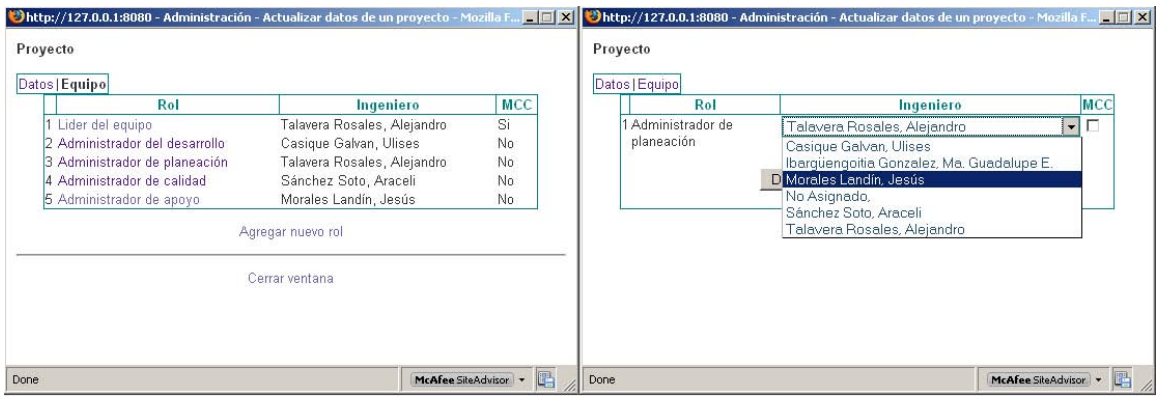


Figura 20 – Asignación de roles en un proyecto

Generación de la forma TASK

El administrador de planeación tiene a su cargo la generación del la forma TASK, que como se mencionó en el capítulo 4, los demás administradores pueden consultarla en su momento. La Figura 21 muestra la ventana de la forma TASK para el administrador de planeación, mientras que la Figura 22 muestra la opción de consulta para cualquier otro colaborador.

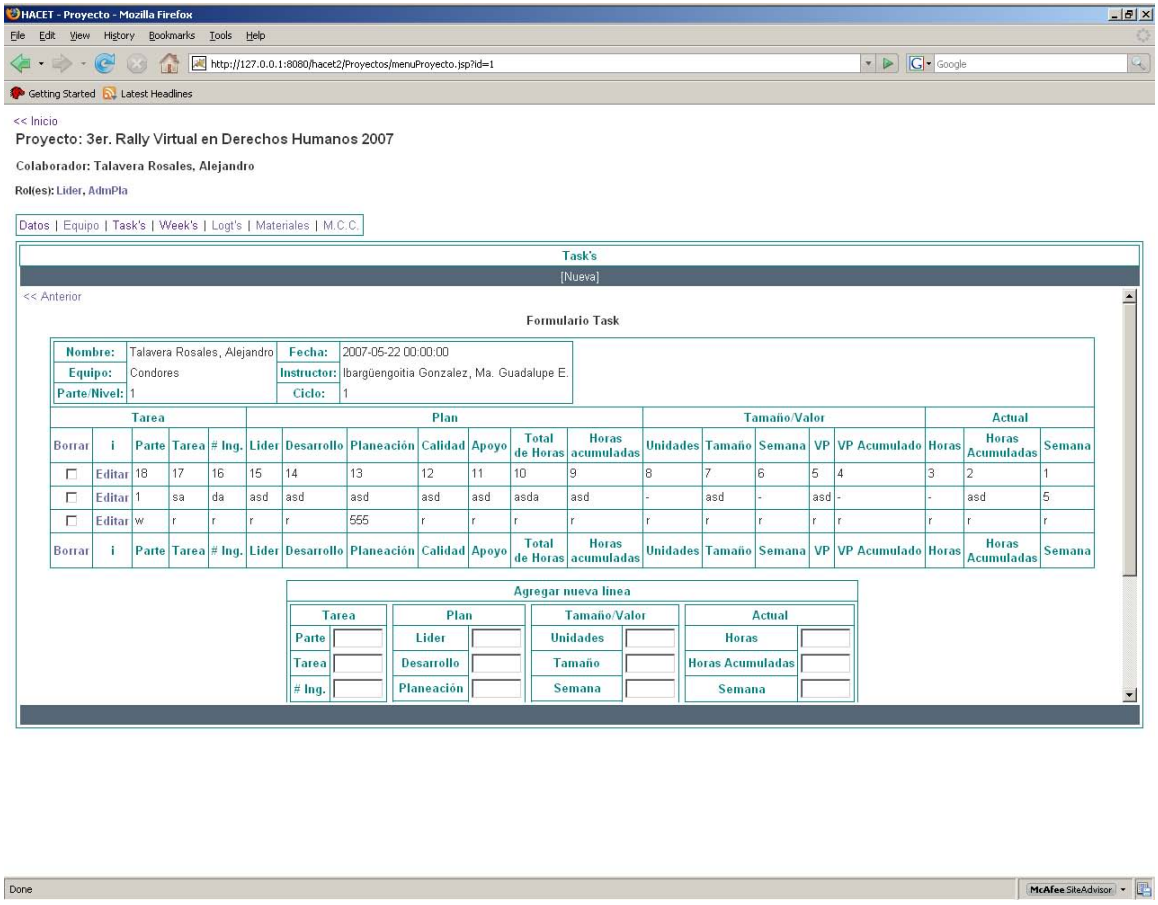


Figura 21 – Formato de llenado de la Forma TASK

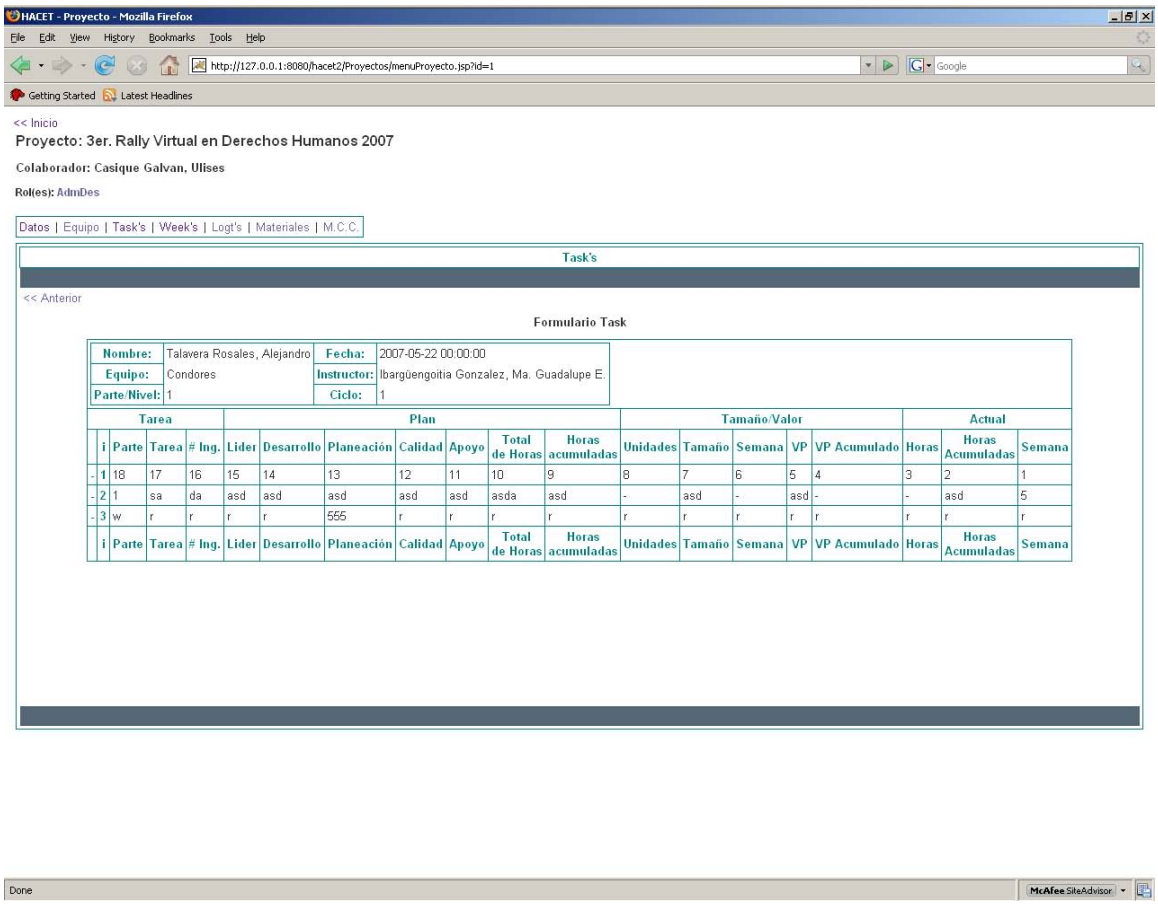


Figura 22 – Consulta de la Forma TASK

Generación de reportes Week

Todos los colaboradores tienen que generar su reporte semanal usando para ello una forma WEEK. La Figura 23 muestra la interfaz de la forma WEEK para cualquier colaborador.

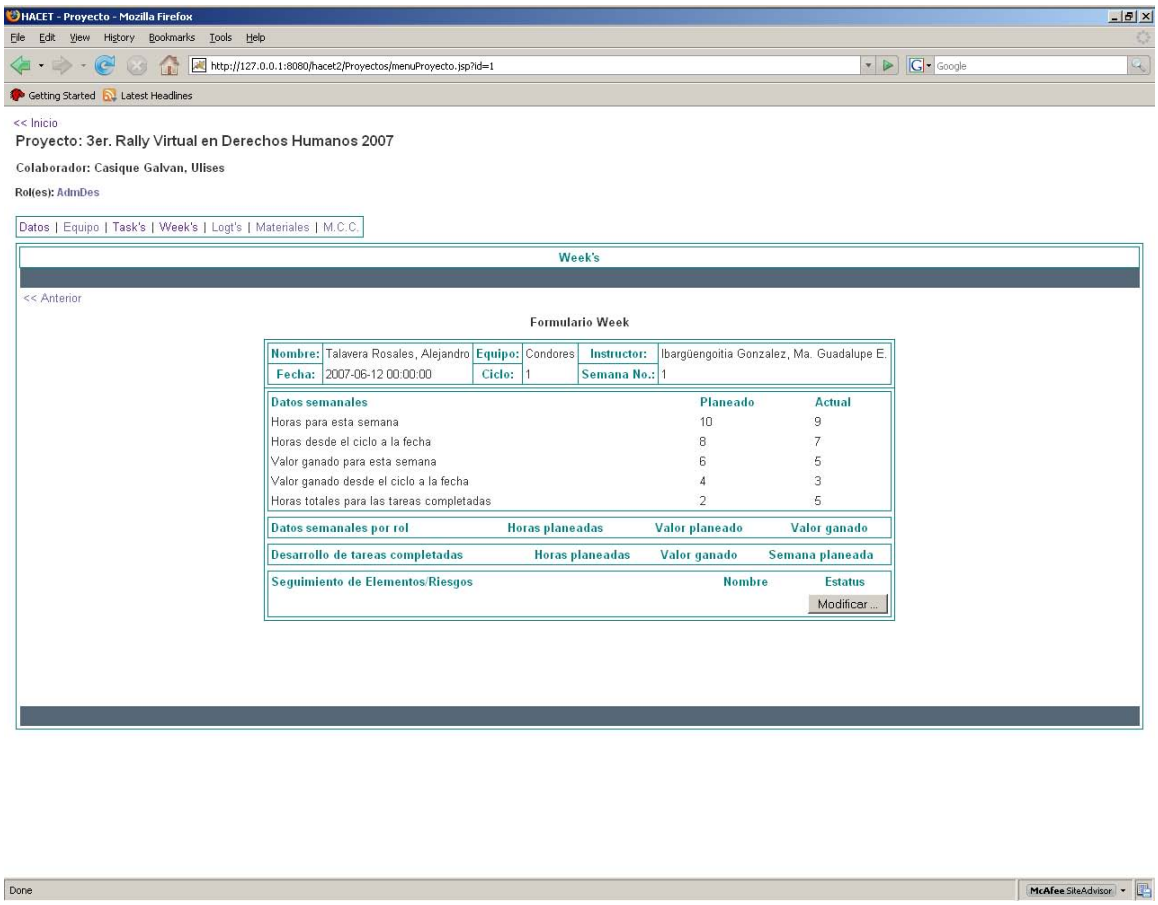


Figura 23 – Forma WEEK

Generación de la bitácora LOGT

Todos los colaboradores pueden generar su reporte de progreso en sus tareas usando para ello una forma LOGT. La Figura 24 muestra la interfaz de la forma WEEK para cualquier colaborador.

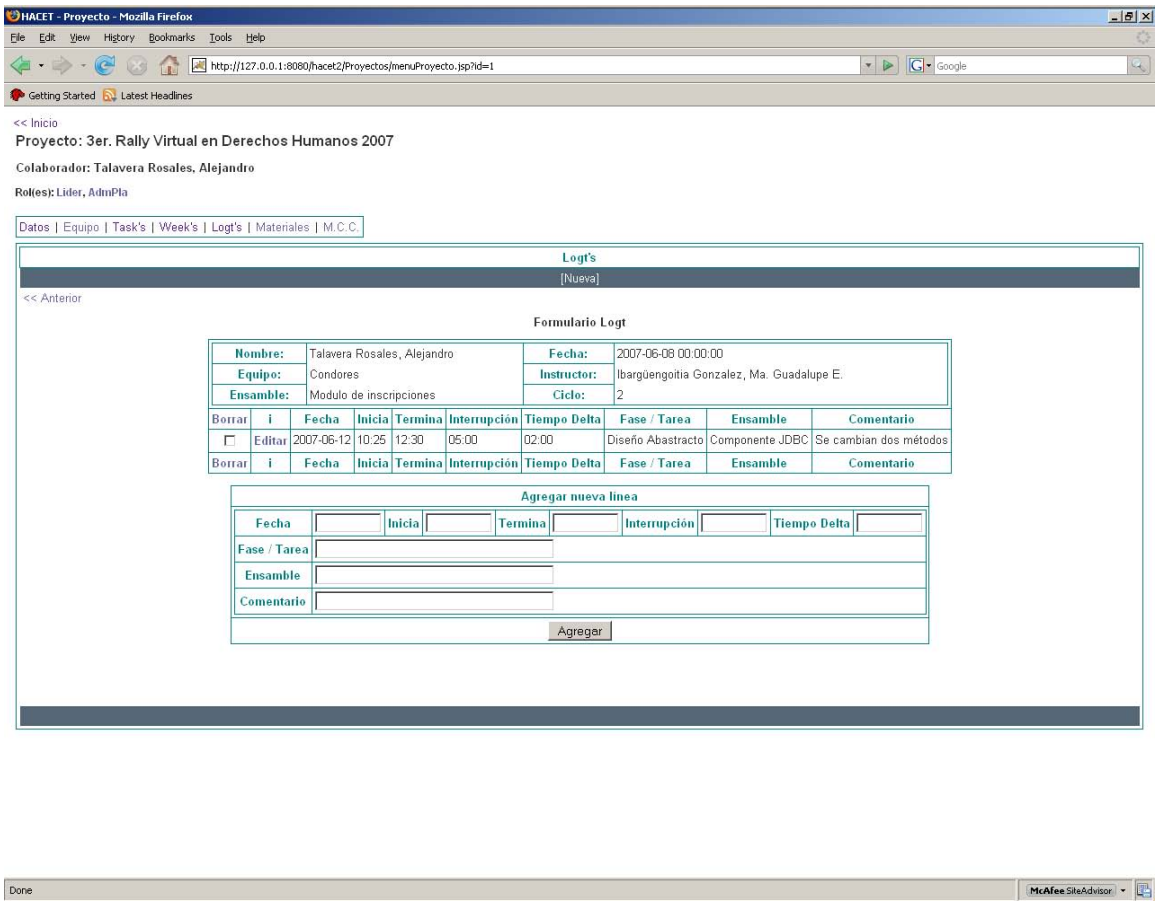


Figura 24 – Forma LOGT

Con este conjunto de formas automatizadas, los miembros de un equipo cuentan con las estructuras de validación y seguimiento de documentos.

Cabe señalar que los clientes de HACET pueden ser creados con la funcionalidad que requieren con solo limitar el conjunto de instrucciones con las que se programen. Esto mismo ocurre con el conjunto de instrucciones de la capa de las reglas del negocio, misma que puede ser incrementada según las necesidades de los proyectos que se van administrando.

Así mismo, no es un requisito indispensable el uso de navegadores o programas con interfaz gráfica, sino más bien el uso correcto de los protocolos de transferencia de instrucciones y respuestas.

5.3 HACET y la Mesa de Control de Cambios (MCC)

Como se indicó anteriormente, uno de los principales objetivos de HACET es apoyar las tareas de la Mesa de Control de Cambios (MCC) por lo existen opciones ya definidas en la herramienta.

Las opciones de la sección MCC de la herramienta permiten dar un seguimiento desde la solicitud de cambio por parte de cualquier miembro del equipo de desarrollo, hasta el dictamen de la mesa en su caso. La Figura 25 muestra las opciones para la MCC en la herramienta.

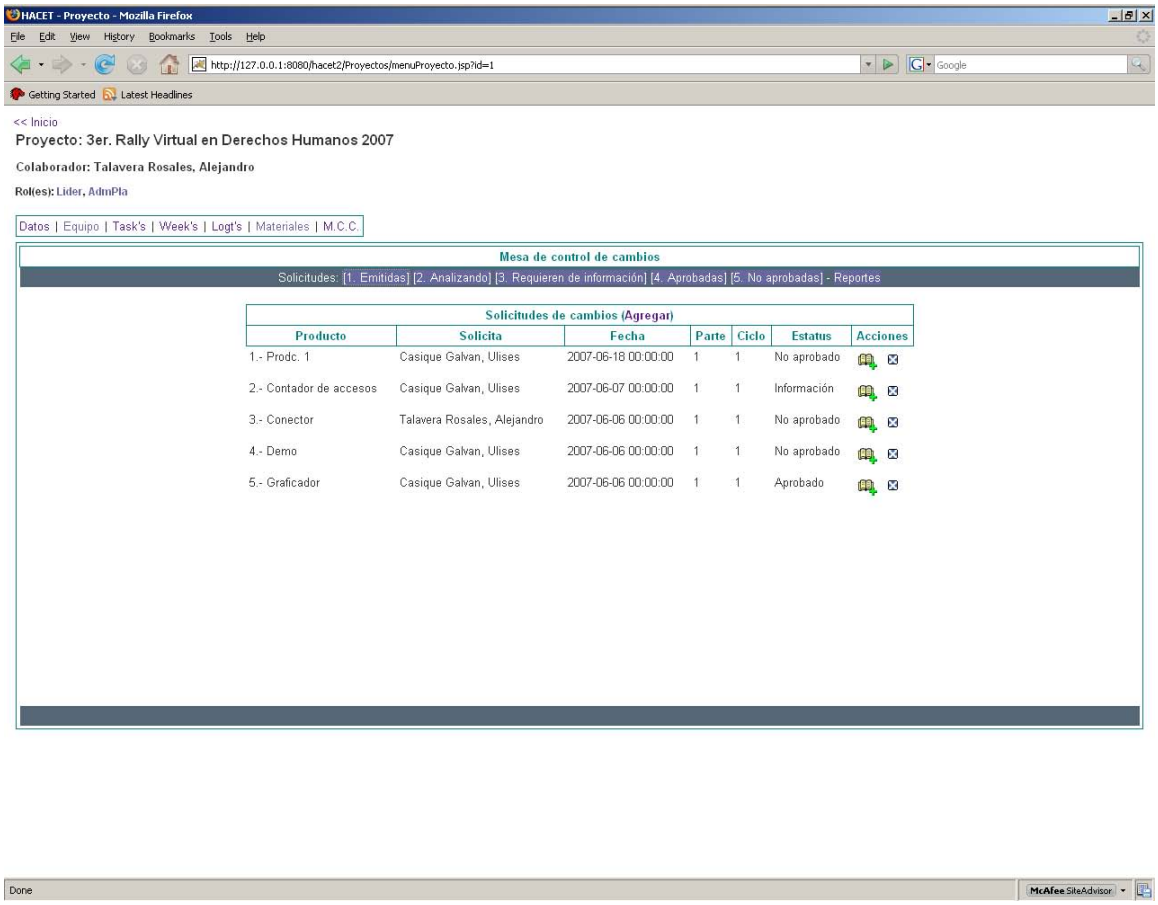


Figura 25 – Opciones para MCC en HACET

En primer lugar, todo miembro del equipo de desarrollo puede solicitar un cambio a la mesa por medio de la opción **Agregar**. La Figura 26 muestra la Forma CCR correspondiente a la solicitud de cambio.

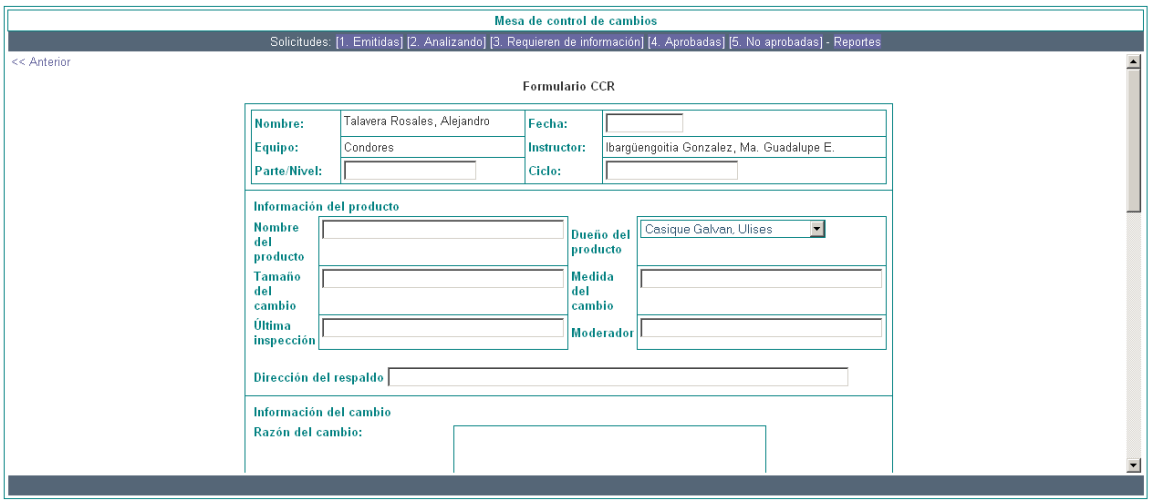


Figura 26 – Forma CCR para solicitar cambios a la MCC

Por su parte la MCC, a través de un responsable, puede analizar las solicitudes que llegan con base en la opción de **Emitidas**. La Figura 27 muestra la lista de forma CCR que se han emitido.

Solicitudes de cambios (Agregar)

Producto	Solicita	Fecha	Parte	Ciclo	Estatus	Acciones
1.- Prodc. 1	Casique Galvan, Ulises	2007-06-18 00:00:00	1	1	No aprobado	
2.- Administrador de preguntas	Talavera Rosales, Alejandro	2007-06-18 00:00:00	1	1	Solicitado	
3.- Contador de accesos	Casique Galvan, Ulises	2007-06-07 00:00:00	1	1	Información	
4.- Conector	Talavera Rosales, Alejandro	2007-06-06 00:00:00	1	1	No aprobado	
5.- Demo	Casique Galvan, Ulises	2007-06-06 00:00:00	1	1	No aprobado	
6.- Graficador	Casique Galvan, Ulises	2007-06-06 00:00:00	1	1	Aprobado	

Figura 27 – Lista de solicitudes a la MCC (forma CCR)

Cada una de estas solicitudes guarda su información particular y al ser inspeccionada por el responsable de la MCC cambia su estado a **Analizando**. En caso de que se requiera mayor información por parte del solicitante el estado cambia a **Información**. Si la MCC resuelve una solicitud esta puede ser **Aceptada** o **Rechazada**. La Figura 28 muestra los estados de una solicitud dentro de la MCC.

La Figura 28 muestra las opciones del responsable de la MCC cuando analiza una solicitud de cambios.

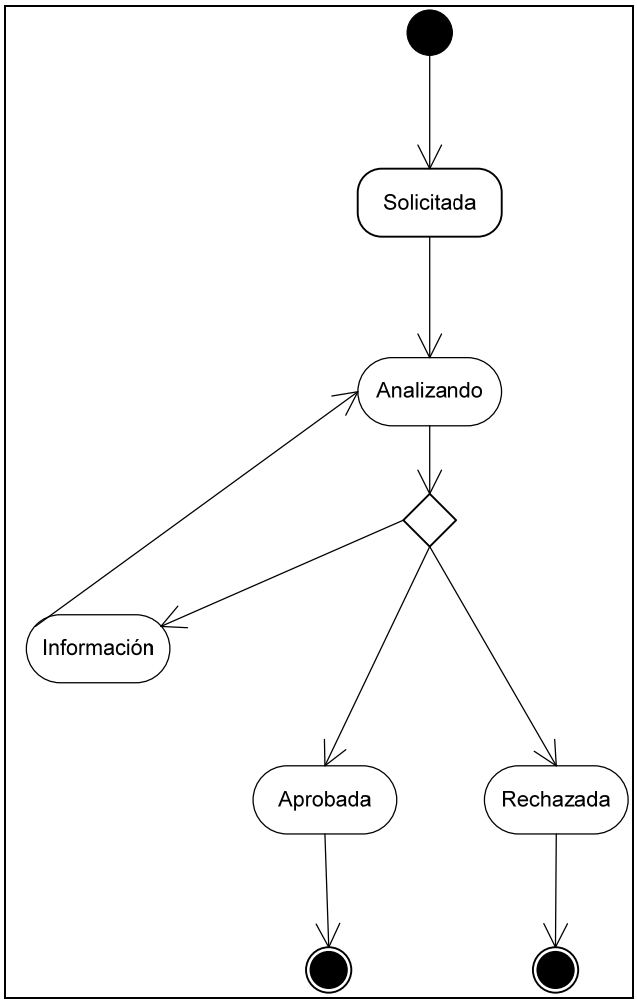


Figura 28 –Estados de una solicitud a la MCC

Por su parte HACET ofrece a los responsables de la MCC's las opciones correspondientes al cambio del estado de una solicitud, además de hacer visibles a todos los solicitantes los distintos estados por los que pasan en la MCC, así como los dictámenes finales. La Figura 29 muestra las opciones para la actualización del estado de una solicitud.

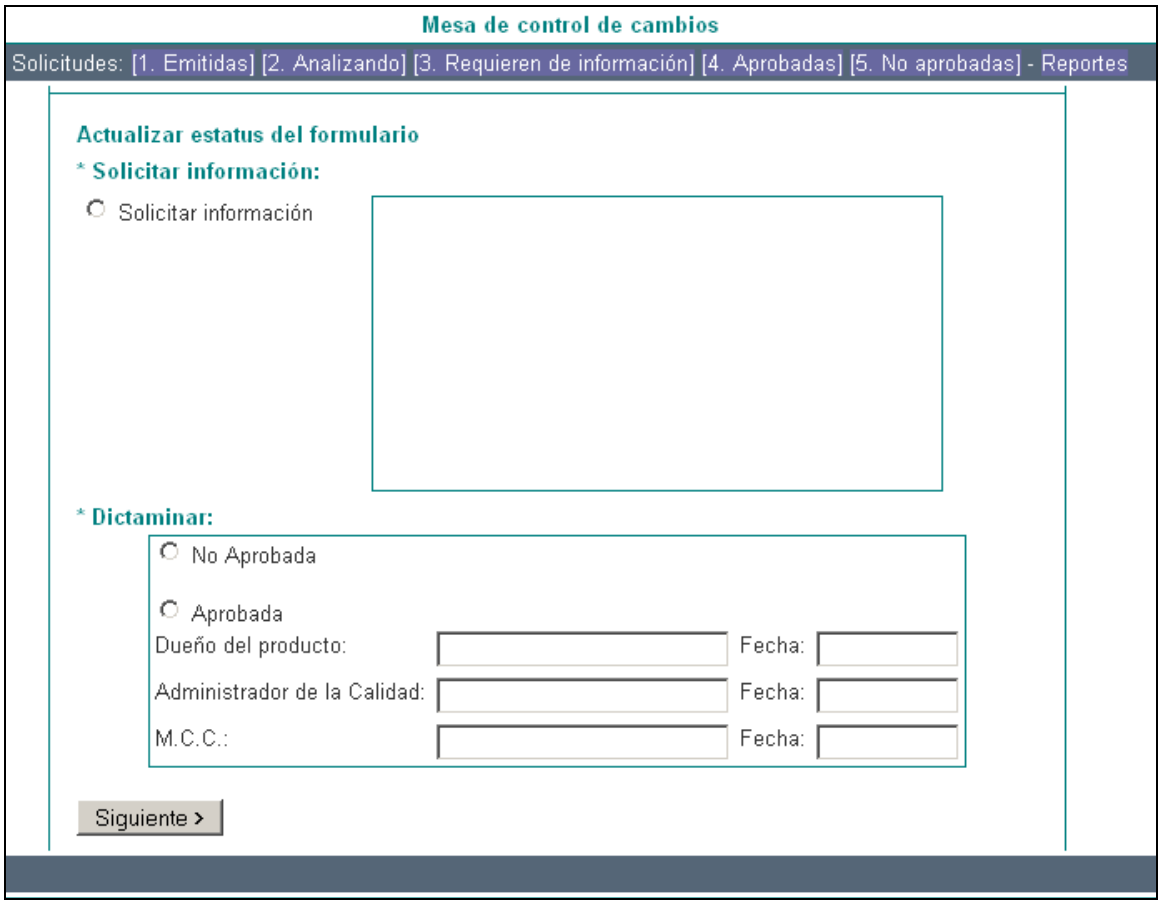


Figura 29 – Opciones para actualizar el estado de una solicitud de cambio

Finalmente los responsables de la MCC pueden generar los reportes que reflejen el trabajo que vienen realizado dentro un proyecto determinado. La Figura 30 muestra la opción de **Reportes**.

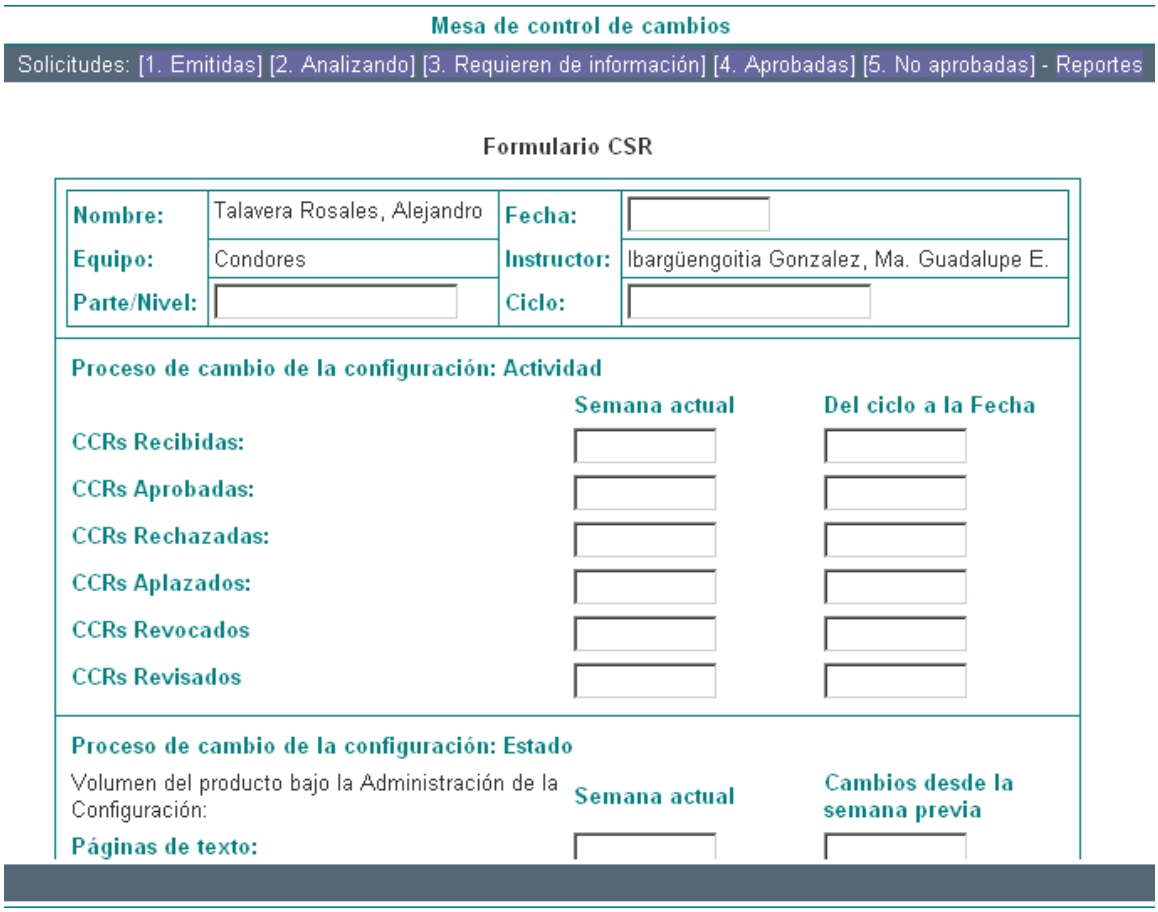


Figura 30 – Reportes de la MCC (formularios CSR)

Estas opciones de automatización para algunas de las tareas de la MCC permiten sentar las bases de una mejor administración de los proyectos de desarrollo así como el control de los distintos roles que participan.

Conclusiones

El desarrollo de software por parte de un equipo de trabajo no solo conlleva la integración de profesionistas en las distintas áreas en torno al proyecto. Se requiere además de una correcta estructura organizacional y la definición de tareas puntuales para cada uno de los miembros del equipo de desarrollo.

TSP reúne una gran parte de las tareas deseadas en cualquier desarrollo, pero su aprendizaje e integración en un equipo de trabajo se ve favorecida con la automatización de las actividades de generación y control de documentos.

Si bien, TSP define las labores de desarrollo y los roles que cada uno de los miembros del equipo de trabajo desempeñará, no se contaba con una herramienta que automatizara dichas tareas.

La herramienta HACET desarrollada en el presente trabajo, está construida expresamente para apoyar la capacitación e integración de TSP en los procesos de desarrollo de Software. HACET fue desarrollada con base a un extenso análisis de las tareas relacionadas con la Administración de la configuración bajo la definición de TSP, además de las cualidades funcionales requeridas en el campo de trabajo.

Las características más importantes con las que se desarrolló HACET se pueden resumir en la siguiente lista:

- Automatización de las tareas definidas por TSP para el desarrollo de software
- La portabilidad lógica y física de los componentes que la integran
- Diseño de mecanismos para la extensión de su funcionalidad
- Apoyo a equipos de trabajo remotos
- Definición de un protocolo propio de comunicación en red
- Persistencia de objetos por la red
- Independencia de la plataforma
- Aplicación de patrones de diseño funcionales

La herramienta actualmente esta en su primera versión, por lo que es importante recalcar que se requiere de un evaluación de su desempeño con equipos en tareas de producción reales, a fin analizar su impacto en los procesos de producción.

Así mismo, es necesario verificar su funcionamiento con varios equipos de desarrollo, además de evaluar las posibles fallas en distintos rublos, como lo son: seguridad, carga de trabajo, carga y flujo de materiales al sistema, entre otros.

Trabajos futuros

A fin de que la herramienta aquí propuesta presente más opciones útiles se ha pensado en incluir las siguientes cualidades:

- Incorporación de un mayor número de formas de TSP y de otros procesos actuales.
- Incorporar mecanismos de validación automática en cada forma que se incorpore a la herramienta.
- Manejo de bitácoras de cambios de roles por proyecto.
- Recuperación de reportes y formas en lugares sin comunicación de red.
- Extensión de las tareas permitidas por parte de la Mesa de control de cambios.
- Reforzamiento de las actividades de las actividades de la administración de la configuración.
- Evaluar aspectos de seguridad con los que se cuenta.

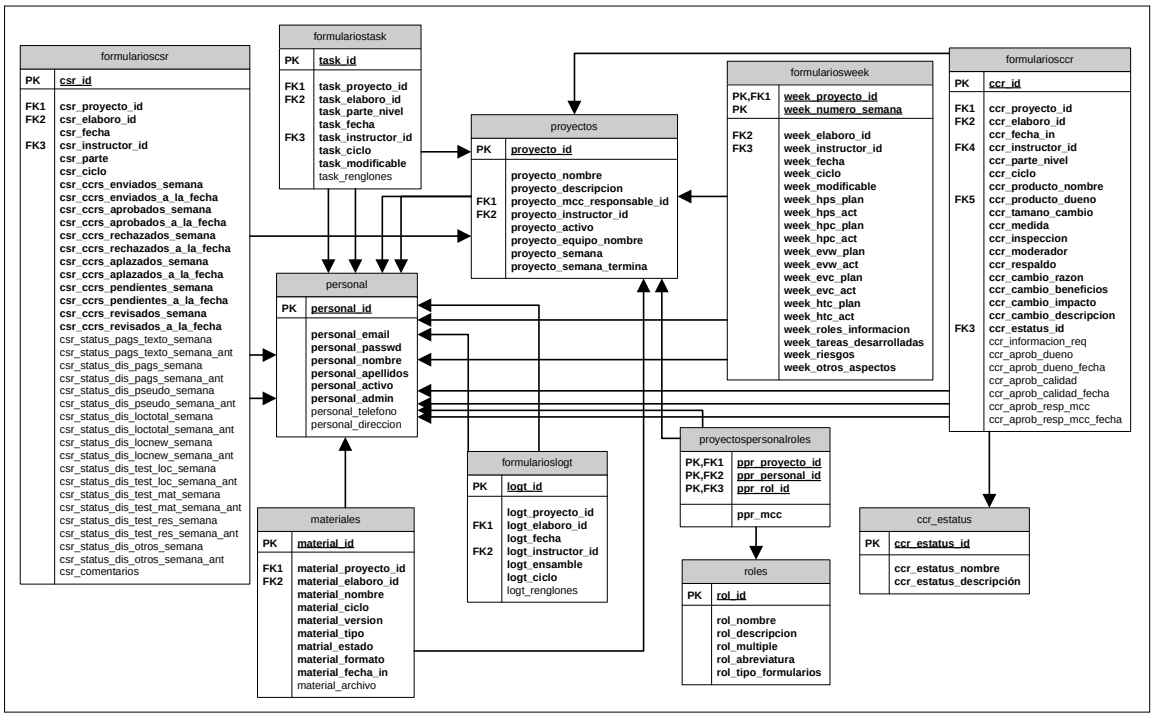
Apéndice A

Diseño de la base de datos

A continuación se presenta el diagrama completo Entidad/Relación mismo que resume el diseño conceptual de la base de datos que se presentó de forma parcial en el Capítulo 5 – Diseño y desarrollo de la herramienta HACET.

Así mismo se ofrece como guía las instrucciones SQL empleadas con el sistema relacional de base de datos PostgreSQL para implementar el diseño.

Diagrama


formulariosscr
PK csr_id
FK1 csr_proyecto_id
FK2 csr_elaboro_id
csr_fecha
FK3 csr_instructor_id
csr_parte
csr_ciclo
csr_ccrs_enviados_semana
csr_ccrs_enviados_a_la_fecha
csr_ccrs_aprobados_semana
csr_ccrs_aprobados_a_la_fecha
csr_ccrs_rechazados_semana
csr_ccrs_rechazados_a_la_fecha
csr_ccrs_aplazados_semana
csr_ccrs_aplazados_a_la_fecha
csr_ccrs_pendientes_semana
csr_ccrs_pendientes_a_la_fecha
csr_ccrs_revisados_semana
csr_ccrs_revisados_a_la_fecha
csr_status_pags_texto_semana
csr_status_pags_texto_semana_ant
csr_status_dis_pags_semana
csr_status_dis_pags_semana_ant
csr_status_dis_pseudo_semana
csr_status_dis_pseudo_semana_ant
csr_status_dis_loctotal_semana
csr_status_dis_loctotal_semana_ant
csr_status_dis_locnew_semana
csr_status_dis_locnew_semana_ant
csr_status_dis_test_loc_semana
csr_status_dis_test_loc_semana_ant
csr_status_dis_test_mat_semana
csr_status_dis_test_mat_semana_ant
csr_status_dis_test_res_semana
csr_status_dis_test_res_semana_ant
csr_status_dis_otros_semana
csr_status_dis_otros_semana_ant
csr_comentarios
formulariostask
PK task_id
FK1 task_proyecto_id
FK2 task_elaboro_id
task_parte_nivel
task_fecha
FK3 task_instructor_id
task_ciclo
task_modificable
task_renglones
proyectos
PK proyecto_id
proyecto_nombre
proyecto_descripcion
FK1 proyecto_mcc_responsable_id
FK2 proyecto_instructor_id
proyecto_activo
proyecto_equipo_nombre
proyecto_semana
proyecto_semana_termina
formulariosweek
PK,FK1 week_proyecto_id
PK week_numero_semana
FK2 week_elaboro_id
FK3 week_instructor_id
week_fecha
week_ciclo
week_modificable
week_hps_plan
week_hps_act
week_hpc_plan
week_hpc_act
week_evw_plan
week_evw_act
week_evc_plan
week_evc_act
week_htc_plan
week_htc_act
week_roles_informacion
week_tareas_desarrolladas
week_riesgos
week_otros_aspectos
formulariosccr
PK ccr_id
FK1 ccr_proyecto_id
FK2 ccr_elaboro_id
ccr_fecha_in
FK4 ccr_instructor_id
ccr_parte_nivel
ccr_ciclo
ccr_producto_nombre
FK5 ccr_producto_dueno
ccr_tamano_cambio
ccr_medida
ccr_inspeccion
ccr_moderador
ccr_respaldo
ccr_cambio_razon
ccr_cambio_beneficios
ccr_cambio_impacto
ccr_cambio_descripcion
FK3 ccr_estatus_id
ccr_informacion_req
ccr_aprob_dueno
ccr_aprob_dueno_fecha
ccr_aprob_calidad
ccr_aprob_calidad_fecha
ccr_aprob_resp_mcc
ccr_aprob_resp_mcc_fecha
personal
PK personal_id
personal_email
personal_passwd
personal_nombre
personal_apellidos
personal_activo
personal_admin
personal_telefono
personal_direccion
formulariosslogt
PK logt_id
FK1 logt_proyecto_id
logt_elaboro_id
logt_fecha
FK2 logt_instructor_id
logt_ensamble
logt_ciclo
logt_renglones
proyectospersonalroles
PK,FK1 ppr_proyecto_id
PK,FK2 ppr_personal_id
PK,FK3 ppr_rol_id
ppr_mcc
roles
PK rol_id
rol_nombre
rol_descripcion
rol_multiple
rol_abreviatura
rol_tipo_formularios
ccr_estatus
PK ccr_estatus_id
ccr_estatus_nombre
ccr_estatus_descripción
materiales
PK material_id
FK1 material_proyecto_id
FK2 material_elaboro_id
material_nombre
material_ciclo
material_version
material_tipo
matrial_estado
material_formato
material_fecha_in
material_archivo

Implementación para PostgreSQL

Creación de las tablas e inserción de datos

```
CREATE TABLE roles (
        rol_id          SERIAL    NOT NULL,
        rol_nombre      CHAR(60)  NOT NULL,
        rol_descripcion  CHAR(100) NOT NULL,
        rol_multiple    BOOLEAN   NOT NULL,
        rol_abreviatura  CHAR(6)   NOT NULL,
        rol_tipo_formularios TEXT     NOT NULL,
        CONSTRAINT roles_pk PRIMARY KEY (rol_id)
);

INSERT INTO roles (rol_id, rol_nombre, rol_descripcion, rol_multiple, rol_abreviatura, rol_tipo_formularios)
VALUES (0,'Sin rol', 'Sin rol', true, '-----', '');
INSERT INTO roles (rol_nombre, rol_descripcion, rol_multiple, rol_abreviatura, rol_tipo_formularios)
VALUES ('Lider del equipo', 'Según las especificaciones de TSP', false, 'Lider', '');
INSERT INTO roles (rol_nombre, rol_descripcion, rol_multiple, rol_abreviatura, rol_tipo_formularios)
VALUES ('Administrador del desarrollo', 'Según las especificaciones de TSP', false, 'AdmDes', '');
INSERT INTO roles (rol_nombre, rol_descripcion, rol_multiple, rol_abreviatura, rol_tipo_formularios)
VALUES ('Administrador de planeación', 'Según las especificaciones de TSP', false, 'AdmPla', '');
INSERT INTO roles (rol_nombre, rol_descripcion, rol_multiple, rol_abreviatura, rol_tipo_formularios)
VALUES ('Administrador de calidad', 'Según las especificaciones de TSP', false, 'AdmCal', '');
INSERT INTO roles (rol_nombre, rol_descripcion, rol_multiple, rol_abreviatura, rol_tipo_formularios)
VALUES ('Administrador de apoyo', 'Según las especificaciones de TSP', false, 'AdmApo', '');
INSERT INTO roles (rol_nombre, rol_descripcion, rol_multiple, rol_abreviatura, rol_tipo_formularios)
VALUES ('Ingeniero de software', 'Según las especificaciones de TSP', true, 'Ing', '');
INSERT INTO roles (rol_nombre, rol_descripcion, rol_multiple, rol_abreviatura, rol_tipo_formularios)
VALUES ('Instructor', 'Según las especificaciones de TSP', true, 'Instr', '');

CREATE TABLE personal (
        personal_id     SERIAL    NOT NULL,
        personal_email   CHAR(150) NOT NULL,
        personal_passwd   CHAR(50),
        personal_nombre   CHAR(40)  NOT NULL,
        personal_apellidos CHAR(80)  NOT NULL,
        personal_activo   BOOLEAN   NOT NULL,
        personal_admin    BOOLEAN   NOT NULL,
        personal_telefono  CHAR(50),
        personal_direccion TEXT,
        CONSTRAINT personal_pk PRIMARY KEY (personal_id)
);

INSERT INTO personal ( personal_id, personal_email, personal_passwd, personal_nombre,
personal_apellidos, personal_activo, personal_admin, personal_telefono, personal_direccion ) VALUES (0,
'admin', 'ADMIN', '', 'No Asignado', true, true, '', '');

CREATE TABLE proyectos (
        proyecto_id             SERIAL    NOT NULL,
        proyecto_nombre         CHAR(60)  NOT NULL,
        proyecto_descripcion    TEXT      NOT NULL,
        proyecto_mcc_responsable_id INT       NOT NULL,
        proyecto_instructor_id   INT       NOT NULL,
        proyecto_activo         BOOLEAN   NOT NULL,
        proyecto_equipo_nombre   CHAR(100) NOT NULL,
```

```
        proyecto_semana          SMALLINT  NOT NULL,
        proyecto_semana_termina    TIMESTAMP NOT NULL,
        CONSTRAINT proyectos_pk  PRIMARY KEY (proyecto_id),
        CONSTRAINT proyectos_fk1 FOREIGN KEY (proyecto_mcc_responsable_id) REFERENCES
personal (personal_id),
        CONSTRAINT proyectos_fk2 FOREIGN KEY (proyecto_instructor_id)      REFERENCES personal
(personal_id)
);

CREATE TABLE proyectospersonalroles (
        ppr_proyecto_id INT     NOT NULL,
        ppr_personal_id INT     NOT NULL,
        ppr_rol_id      INT     NOT NULL,
        ppr_mcc         BOOLEAN NOT NULL,
        CONSTRAINT ppr_pk  PRIMARY KEY (ppr_proyecto_id, ppr_personal_id, ppr_rol_id),
        CONSTRAINT ppr_fk1 FOREIGN KEY (ppr_proyecto_id) REFERENCES proyectos (proyecto_id),
        CONSTRAINT ppr_fk2 FOREIGN KEY (ppr_personal_id) REFERENCES personal  (personal_id),
        CONSTRAINT ppr_fk3 FOREIGN KEY (ppr_rol_id)      REFERENCES roles     (rol_id)
);

CREATE TABLE formulariostask (
        task_id          SERIAL      NOT NULL,
        task_proyecto_id   INT         NOT NULL,
        task_elaboro_id    INT         NOT NULL,
        task_parte_nivel   VARCHAR(30) NOT NULL,
        task_fecha         TIMESTAMP   NOT NULL,
        task_instructor_id INT         NOT NULL,
        task_ciclo         VARCHAR(30) NOT NULL,
        task_modificable   BOOLEAN     NOT NULL,
        task_renglones     TEXT,
        CONSTRAINT formulariostask_pk  PRIMARY KEY (task_id),
        CONSTRAINT formulariostask_fk1 FOREIGN KEY (task_proyecto_id)    REFERENCES proyectos
(proyecto_id),
        CONSTRAINT formulariostask_fk2 FOREIGN KEY (task_elaboro_id)     REFERENCES personal
(personal_id),
        CONSTRAINT formulariostask_fk3 FOREIGN KEY (task_instructor_id) REFERENCES personal
(personal_id)
);

CREATE TABLE formularioslogt (
        logt_id          SERIAL      NOT NULL,
        logt_proyecto_id   INT         NOT NULL,
        logt_elaboro_id    INT         NOT NULL,
        logt_fecha         TIMESTAMP   NOT NULL,
        logt_instructor_id INT         NOT NULL,
        logt_ensamble      VARCHAR(30) NOT NULL,
        logt_ciclo         VARCHAR(30) NOT NULL,
        logt_renglones     TEXT,
        CONSTRAINT logt_pk  PRIMARY KEY (logt_id),
        CONSTRAINT logt_fk1 FOREIGN KEY (logt_elaboro_id)    REFERENCES personal  (personal_id),
        CONSTRAINT logt_fk2 FOREIGN KEY (logt_instructor_id) REFERENCES personal  (personal_id)
);

CREATE TABLE formulariosweek (
        week_proyecto_id         INT         NOT NULL,
        week_numero_semana       INT         NOT NULL,
        week_elaboro_id          INT         NOT NULL,
        week_instructor_id       INT         NOT NULL,
        week_fecha               TIMESTAMP   NOT NULL,
        week_ciclo               VARCHAR(30) NOT NULL,
        week_modificable         BOOLEAN     NOT NULL,
```

```
        week_hps_plan           REAL       NOT NULL,
        week_hps_act            REAL       NOT NULL,
        week_hpc_plan           REAL       NOT NULL,
        week_hpc_act            REAL       NOT NULL,
        week_evw_plan           REAL       NOT NULL,
        week_evw_act            REAL       NOT NULL,
        week_evc_plan           REAL       NOT NULL,
        week_evc_act            REAL       NOT NULL,
        week_htc_plan           REAL       NOT NULL,
        week_htc_act            REAL       NOT NULL,
        week_roles_informacion  TEXT       NOT NULL,
        week_tareas_desarrolladas TEXT     NOT NULL,
        week_riesgos            TEXT       NOT NULL,
        week_otros_aspectos     TEXT       NOT NULL,
        CONSTRAINT week_pk  PRIMARY KEY (week_proyecto_id, week_numero_semana),
        CONSTRAINT week_fk1 FOREIGN KEY (week_proyecto_id) REFERENCES proyectos
(proyecto_id),
        CONSTRAINT week_fk2 FOREIGN KEY (week_elaboro_id) REFERENCES personal
(personal_id),
        CONSTRAINT week_fk3 FOREIGN KEY (week_instructor_id) REFERENCES personal
(personal_id)
);

CREATE TABLE ccr_estatus (
        ccr_estatus_id      SERIAL     NOT NULL,
        ccr_estatus_nombre  VARCHAR(30) NOT NULL,
        ccr_estatus_descripción TEXT   NOT NULL,
        CONSTRAINT ccr_estatus_pk  PRIMARY KEY (ccr_estatus_id)
);

INSERT INTO ccr_estatus (ccr_estatus_nombre, ccr_estatus_descripción) VALUES ('Solicitado', 'Se ha
solicitado el cambio');
INSERT INTO ccr_estatus (ccr_estatus_nombre, ccr_estatus_descripción) VALUES ('Analizando', 'Se esta
analizando el cambio');
INSERT INTO ccr_estatus (ccr_estatus_nombre, ccr_estatus_descripción) VALUES ('Información', 'Se
solicita información');
INSERT INTO ccr_estatus (ccr_estatus_nombre, ccr_estatus_descripción) VALUES ('Aprobado', 'Se aprobó
el cambio');
INSERT INTO ccr_estatus (ccr_estatus_nombre, ccr_estatus_descripción) VALUES ('No aprobado', 'No se
aprobó el cambio solicitado');


CREATE TABLE formulariosccr (
        ccr_id              SERIAL     NOT NULL,
        ccr_proyecto_id     INT        NOT NULL,
        ccr_elaboro_id      INT        NOT NULL,
        ccr_fecha_in        TIMESTAMP  NOT NULL,
        ccr_instructor_id   INT        NOT NULL,
        ccr_parte_nivel     VARCHAR(30) NOT NULL,
        ccr_ciclo           VARCHAR(30) NOT NULL,

        ccr_producto_nombre  VARCHAR(60)  NOT NULL,
        ccr_producto_dueno   INT          NOT NULL,
        ccr_tamano_cambio    VARCHAR(60)  NOT NULL,
        ccr_medida           VARCHAR(60)  NOT NULL,
        ccr_inspeccion       VARCHAR(60)  NOT NULL,
        ccr_moderador        VARCHAR(60)  NOT NULL,
        ccr_respaldo         VARCHAR(100) NOT NULL,

        ccr_cambio_razon     TEXT NOT NULL,
        ccr_cambio_beneficios TEXT NOT NULL,
```

```
        ccr_cambio_impacto      TEXT NOT NULL,
        ccr_cambio_descripcion  TEXT NOT NULL,

        ccr_estatus_id          INT  NOT NULL,
        ccr_informacion_req     TEXT,

        ccr_aprob_dueno         VARCHAR(30),
        ccr_aprob_dueno_fecha   TIMESTAMP,
        ccr_aprob_calidad       VARCHAR(30),
        ccr_aprob_calidad_fecha TIMESTAMP,
        ccr_aprob_resp_mcc      VARCHAR(30),
        ccr_aprob_resp_mcc_fecha TIMESTAMP,

        CONSTRAINT ccr_pk  PRIMARY KEY (ccr_id),
        CONSTRAINT ccr_fk1 FOREIGN KEY (ccr_proyecto_id)  REFERENCES proyectos
(proyecto_id),
        CONSTRAINT ccr_fk2 FOREIGN KEY (ccr_elaboro_id)   REFERENCES personal
(personal_id),
        CONSTRAINT ccr_fk3 FOREIGN KEY (ccr_estatus_id)   REFERENCES ccr_estatus
(ccr_estatus_id),
        CONSTRAINT ccr_fk4 FOREIGN KEY (ccr_instructor_id) REFERENCES personal
(personal_id),
        CONSTRAINT ccr_fk5 FOREIGN KEY (ccr_producto_dueno) REFERENCES personal
(personal_id)
);

CREATE TABLE formulariosCSR (
        csr_id                  SERIAL      NOT NULL,

        csr_proyecto_id         INT         NOT NULL,
        csr_elaboro_id          INT         NOT NULL,
        csr_fecha               TIMESTAMP   NOT NULL,
        csr_instructor_id       INT         NOT NULL,
        csr_parte               VARCHAR(30) NOT NULL,
        csr_ciclo               VARCHAR(30) NOT NULL,

        csr_ccrs_enviados_semana    INT     NOT NULL,
        csr_ccrs_enviados_a_la_fecha INT    NOT NULL,
        csr_ccrs_aprobados_semana   INT     NOT NULL,
        csr_ccrs_aprobados_a_la_fecha INT   NOT NULL,
        csr_ccrs_rechazados_semana  INT     NOT NULL,
        csr_ccrs_rechazados_a_la_fecha INT  NOT NULL,
        csr_ccrs_aplazados_semana   INT     NOT NULL,
        csr_ccrs_aplazados_a_la_fecha INT   NOT NULL,
        csr_ccrs_pendientes_semana  INT     NOT NULL,
        csr_ccrs_pendientes_a_la_fecha INT  NOT NULL,
        csr_ccrs_revisados_semana   INT     NOT NULL,
        csr_ccrs_revisados_a_la_fecha INT   NOT NULL,

        csr_status_pags_texto_semana    VARCHAR(30),
        csr_status_pags_texto_semana_ant VARCHAR(30),
        csr_status_dis_pags_semana      VARCHAR(30),
        csr_status_dis_pags_semana_ant  VARCHAR(30),
        csr_status_dis_pseudo_semana    VARCHAR(30),
        csr_status_dis_pseudo_semana_ant VARCHAR(30),
        csr_status_dis_loctotal_semana  VARCHAR(30),
        csr_status_dis_loctotal_semana_ant VARCHAR(30),
        csr_status_dis_locnew_semana    VARCHAR(30),
        csr_status_dis_locnew_semana_ant VARCHAR(30),
        csr_status_dis_test_loc_semana  VARCHAR(30),
        csr_status_dis_test_loc_semana_ant VARCHAR(30),
```

```
        csr_status_dis_test_mat_semana     VARCHAR(30),
        csr_status_dis_test_mat_semana_ant VARCHAR(30),
        csr_status_dis_test_res_semana     VARCHAR(30),
        csr_status_dis_test_res_semana_ant VARCHAR(30),
        csr_status_dis_otros_semana        VARCHAR(30),
        csr_status_dis_otros_semana_ant    VARCHAR(30),

        csr_comentarios                    TEXT,

        CONSTRAINT csr_pk  PRIMARY KEY (csr_id),
        CONSTRAINT csr_fk1 FOREIGN KEY (csr_proyecto_id)  REFERENCES proyectos
(proyecto_id),
        CONSTRAINT csr_fk2 FOREIGN KEY (csr_elaboro_id)   REFERENCES personal  (personal_id),
        CONSTRAINT csr_fk3 FOREIGN KEY (csr_instructor_id) REFERENCES personal  (personal_id)
);

CREATE TABLE materiales (
        material_id         SERIAL      NOT NULL,
        material_proyecto_id INT        NOT NULL,
        material_elaboro_id  INT        NOT NULL,
        material_nombre      VARCHAR(50) NOT NULL,
        material_formato     VARCHAR(30) NOT NULL,
        material_version     VARCHAR(30) NOT NULL,
        material_fecha_in    TIMESTAMP   NOT NULL,
        material_archivo     BYTEA,
        material_descripcion TEXT,
        CONSTRAINT material_pk  PRIMARY KEY (material_id),
        CONSTRAINT material_fk1 FOREIGN KEY (material_proyecto_id)  REFERENCES proyectos
(proyecto_id),
        CONSTRAINT material_fk2 FOREIGN KEY (material_elaboro_id)   REFERENCES personal
(personal_id)
);
```

Apéndice B

Instalación de HACET

Requisitos

Al desarrollarse HACET con estándares libres para su instalación requiere que se cuente con:

- Un sistema manejador de bases de datos
- Un contenedor de servlets propiamente configurado
- Controladores JDBC para el manejador de la base de datos

La herramienta se ha probado con los siguientes sistemas manejadores de base de datos:

- PostgreSQL
- MySQL

El contenedor de servlets donde se ha probado el funcionamiento de HACET es Tomcat en distintas versiones: 3.x, 4.x, 5.x y 6.x, pero pueden emplearse otros contenedores que implementen la referencia oficial de Sun Microsystems para las tecnologías Java Servlet y JavaServer Pages.

Los controladores JDBC dependen enteramente de la base de datos que se va a utilizar. Los controladores que han utilizado exitosamente para PostgreSQL[1] son:

- 7.3 Build 113
- 7.4 Build 216
- 8.0 Build 320
- 8.1 Build 410

[1] Para mayor información consultar la página http://jdbc.postgresql.org/download.html

- 8.2 Build 506.

Para MySQL se han utilizado los siguientes controladores[2]:

- Connector/J 3.0
- Connector/J 3.1
- Connector/J 5.0
- Connector/J 5.1

En caso de utilizar otro sistema manejador de base de datos se requiere de consultar con el proveedor a fin de obtener el controlador JDBC correcto y seguir los pasos que se indique para la configuración del sistema.

Pasos de la instalación

El proceso de instalación se ha dividido en los siguientes pasos:

1. Creación de la Base de datos
2. Instalar el archivo hacet2.war
3. Configurar los archivos *variables_db.jsp*

A continuación se explican a detalle cada uno de los pasos anteriores.

1. Creación de la Base de Datos

En los Sistemas Manejadores de Base de Datos (SMBD) mencionados en el punto de **Requisitos** se requiere de contar con un espacio de trabajo y los permisos correspondientes para poder generar las tablas definidas en el diseño. En el Apéndice A se presentan los comandos SQL para la creación de la base.

Como ejemplo de la creación de la base se ponen a continuación los comandos utilizados en PostgreSQL.

[2] Para mayor información consultar la página http://www.mysql.com/products/connector/j/

Creación del usuario en PostgreSQL

```
createuser -P -l tspi2
Enter password for new role: Jkll89ysv
Enter it again:
Shall the new role be a superuser? (y/n) n
Shall the new role be allowed to create databases? (y/n) n
Shall the new role be allowed to create more new roles? (y/n) n
CREATE ROLE
```

Creación de la base de datos en PostgreSQL

```
createdb -E LATIN1 -O tspi2 tspi2
CREATE DATABASE
```

Una vez creada la base de datos falta construir las tablas de la base. Para este punto consultar el Apéndice A, donde aparecen las instrucciones para crear las tablas y los datos iniciales de las mismas.

Si se emplea otro SMBD distinto a PostgreSQL, el proceso puede variar completamente. Se recomienda leer la documentación del SMDB correspondiente y/o solicitar el apoyo del área de administración de bases de datos.

2. Instalar el archivo hacet2.war

El archivo **hacet2.war** es la distribución de la herramienta HACET y que puede ser instalado en un contenedor de servlets. En este archivo se encuentran compactados todos los archivos del sistema. Para obtener una copia del mismo solo se requiere bajar de la siguiente página: http://www.hacet.dgsca.unam.mx/.

El proceso de instalación de archivos WAR varía de contenedor a contenedor y de las herramientas que se emplean con el mismo. Por lo anterior hay que consultar la documentación propia del contenedor de servlets que seesta empleando.

Para ejemplificar el proceso de instalación se presenta los pasos de instalación del archivo **hacet2.war** con un contenedor Tomcat 6:

a. Parar Tomcat
b. Borra cualquier instalación previa de HACET
c. Copiar **hacet2.war** en el directorio $CATALINA_HOME/webapps/
d. Iniciar Tomcat

Al iniciar Tomcat, automáticamente se instala el archivo **hacet2.war** en el contenedor.

3. Configurar los archivos *variables_db.jsp*

Antes de comenzar a utilizar HACET se debe de modificar dos archivos nombrados *variables_db.jsp*. Estos archivos se localizan en los siguientes directorios del contenedor:

- $CATALINA_HOME/webapps/hacet2/Administracion
- $CATALINA_HOME/webapps/hacet2/Proyectos

Una vez ubicado estos archivos los cambios que deben efectuarse son los siguientes:

- $CATALINA_HOME/webapps/hacet2/Administración/variables_db.jsp, actualizar las siguientes líneas con los datos correspondientes al sistema:

```
String base_driver       = "org.postgresql.Driver";
String base_url          = "jdbc:postgresql://132.248.150.193/tspi2";
String base_login        = "tspi2";
String base_passw        = "Jkll89ysv";
String admin_login       = "admin";
String admin_passwd      = "ADMIN";
```

- $CATALINA_HOME/webapps/hacet2/Proyectos/variables_db.jsp
  actualizar las siguientes líneas con los datos correspondientes al sistema:
  String base_driver = "org.postgresql.Driver";
  String base_url = "jdbc:postgresql://132.248.150.193/tspi2";
  String base_login = "tspi2";
  String base_passw = "Jkll89ysv";

Bibliografía

Libros
[Bab 86] Babich W.A. Software Configuration Management, Addison-Wesley, 1996.
[IEEE – 729 83] ANSI/IEEE Std 729-1983, Glossary of Software Engineering Terminology, IEEE, New York, 1983
[IEEE –1042 87] ANSI/IEEE Std 1042-1987, Guide to Software Configuration Management, American National Standard / IEEE, New York, 1990
[Som 92] Sommerville IAN, Software Engineering/IAN Sommerviller, Addison Wesley, 1992.
[ThMc 93] Richard H. Thayer and Andrew D. McGettrick, Software Engineering A European Perspective, IEEE Computer Society Press, 1993
[WHS 99] William J. Brown, Hays W. “Skip” McCormick III, Scott W. Thomas, AntiPatterns and Patterns in Software Configuration Management, Willey Computer Publishing, 1999.
[WSH 2000] Watts S. Humphrey, Introduction to the Team Software Process, Addison-Wesley, 2000
[JZ 2006] Jeffrey Zeldman, Designing With Web Standards, Peachpit Press, 2006
[BH 2003] Brian Hochgurtel, Cross-Platform Web Services Using C# & JAVA, Charles River Media, 2003

Sitios
[SUN-MS DEAJ2PEE, 2001] Designing Enterprise Applications with the Java[tm] 2 Platform, Enterprise Edition, 2nd Edition
http://java.sun.com/blueprints/guidelines/designing_enterprise_applications_2e_draft/blueprintsTOC.html
[JCh 2000] J. Chavez, Multi-tier Internet Architecture with Java, UML and OOA & D
http://www.adass.org/adass/proceedings/adass99/P1-59/