VNIVERSIDAD NACIONAL AVFNOMA DE MEXICO

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

INTEGRACIÓN DE UN MODELO CUALITATIVO Y CUANTITATIVO MEDIANTE UNA HERRAMIENTA QUE CONTROLE EL PROCESO DE UN PRODUCTO DE SOFTWARE

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN INGENIERÍA

P R E S E N T A

JORGE MAURICIO ESPINOZA MEJÍA

DIRECTORA DE TESIS: DRA. HANNA OKTABA

Esta tesis corresponde a los estudios realizados con una beca otorgada por el Gobierno de México, a través del Instituto Mexicano de Cooperación Internacional (IMEXCI) de la Secretaria de Relaciones Exteriores.

MÉXICO, D.F. **JULIO, 2002**

Dedicado a mi Familia
y en Memoria de
mi hermano "Chichi"

Agradecimientos:

Gracias Señor por hacer realidad este sueño, por tu amor que sobrepasa todo entendimiento y mostrarme que siempre estás conmigo. Sin ti nada es posible.

A mis Padres y hermanos por hacerme sentir que la distancia no es un obstáculo para saber que están allí. Ustedes son una bendición en mi vida; su amor me inyectó las fuerzas necesarias para llegar a la meta.

Al gobierno de México por conducto del Instituto Mexicano de Cooperación Internacional, quien me brindó el apoyo económico para realizar la Maestría.

A mis maestros, en especial a la Dra. Hanna Oktaba, por su tiempo, apoyo y confianza.

A mis sinodales por sus valiosas sugerencias para la terminación de este trabajo.

A mi amigo Fabián de la Cruz, por su amistad, dedicación y perseverancia, lo cual me motivo a seguir adelante; a Gustavo por sus valiosos consejos técnicos y de manera especial a Jessi por ayudarme en la traducción de los primeros capítulos. Gracias por tu don de servicio.

A la familia Novoa-Medina, en especial a Edwin, por su tiempo y paciencia, a Ceci por hacerme sentir como en casa y a Moshita por sus grandes detalles.

A Lulú, Amalia, Violeta, Juanita, Pedro y Dante, personal administrativo de la maestría por su ayuda y amistad.

Al Ing. Bert De Bièvre por brindarme su apoyo desinteresado en los momentos difíciles. ¡Gracias Bert!.

Y a todos aquellos que de alguna manera contribuyeron para culminar este objetivo.

CONTENIDO

ANEXOS

FIGURAS

TABLAS

Introducción

El desarrollo y mantenimiento de un producto de software resulta a menudo muy complejo por lo que no llevar un control apropiado del mismo llevaría a obtener una inconsistencia entre los requerimientos del producto y el producto propiamente dicho, lo que significa una calidad pobre. Además, los cambios no controlados sobre los productos pueden provocar costo adicional y trabajo vuelto a efectuar.

Los libros de texto de ingeniería de software [Ghezzi91], [Sommerville96], [Hamlet01], los modelos estándar y de referencia [Paulk95], [ISO12207], [ISO15504], [CMMI], sugieren como controlar los cambios y ofrecen una gran variedad de técnicas de verificación/validación para mejorar la calidad de los productos de software. Usualmente, ambos problemas son vistos como separados. Sin embargo, existe una fuerte dependencia entre la calidad y el control de cambios. Esto significa que cada cambio en un producto de software puede causar una revisión (re-verificación/re-validación) del mismo. Por supuesto, las técnicas de revisión dependerán del tipo y severidad del cambio. Esta parte, la cual denominamos *modelo genérico cualitativo multinivel (modelo cualitativo),* fue desarrollado en un trabajo previo [DeJesús01].

Al controlar la calidad y cambios, es posible obtener un producto de software que cumpla los requerimientos establecidos. No obstante, el administrador de proyectos requiere controlar los planes y costos de desarrollo [Humphrey95], lo cual implica implementar las mediciones sobre el proceso y producto de software. La medición juega un rol primordial en el proceso de mejoramiento. El uso de las mediciones implica un control cuantitativo sobre el proceso de desarrollo del software.

Existen muchas propuestas acerca del proceso de medición del software, pero la mayoría de éstas definen "qué" debe ser medido, y no "cómo" y "cuándo" hacerlo. Este trabajo propone un *modelo para el proceso de medición del software (modelo cuantitativo)* basado en [McAndrews93], [Morisio95], [Florac97], [Augustine97], [Jacquet97], [Jones00]; que

sea fácil de entender y provea instrucciones claras acerca de "qué", "cómo" y "cuándo" medir. Este modelo, tiene como alcance ayudar a cualquier organización a describir los productos de software y mejorar el proceso de desarrollo, para lo cual requiere la definición de objetivos que permitan valorar la eficiencia del proceso y descubrir áreas que necesitan ser mejoradas.

De los modelos de proceso de software conocidos internacionalmente Personal Software Process (PSP), es uno de los mas detallados en cuanto a como usar las mediciones como herramienta para evaluar, estimar y controlar el proceso de software. Además PSP posee un conjunto básico de métricas: el tiempo dedicado a una actividad específica, el tamaño del producto, y los defectos encontrados y corregidos durante el proceso de producción. Por estos motivos se escogió PSP como base para la identificación de los objetivos que permita diseñar el *modelo cuantitativo* propuesto a una escala general.

Los procesos presentes dentro de los *modelos cualitativo* y *cuantitativo*, son generalmente manejados en forma independiente, por lo que se propone integrarlos en un solo ambiente de trabajo, pretendiendo que la integración y la interpretación de los resultados que proyecta el uso de éstos pueda provocar un cambio cultural en los desarrolladores de software, una mejora en las organizaciones que apliquen el modelo; contribuyendo al alcance de una disciplina de ingeniería madura.

Objetivos

- Integrar un modelo de control de cambios y calidad (*modelo cualitativo*) a un modelo de manejo de métricas (*modelo cuantitativo*) que permita controlar el proceso de un producto de software.
- Desarrollar una herramienta que soporte el uso del modelo integrado.

La utilización correcta de este modelo permitirá:

A nivel proyecto:

- Valorar el costo de producción de un proyecto de desarrollo de software.
- Identificar los cambios que pueden mejorar la productividad.
- Valorar la efectividad de las revisiones y pruebas.
- Determinar el esfuerzo consumido en trabajo vuelto a realizar.
- Determinar qué fases de desarrollo generan más defectos.
- Determinar qué factores afectan la exactitud de la estimación.

A nivel organización:

- Evaluar de manera cuantitativa los productos, procesos y servicios de software.
- Evaluar el software de productos, procesos y servicios con respecto a los estándares y metas establecidas.
- Estimar atributos como: tamaño, tiempo y recursos necesarios para entidades de software en el futuro.
- Evaluar el nivel de calidad de los productos, así como la calidad que los procesos existentes son capaces de alcanzar.
- Usar las descripciones del producto para valorar la calidad del trabajo e identificar debilidades durante el proceso.
- Efectuar estimaciones reales y planear el tamaño, esfuerzo, y tiempo para proyectos futuros, basados en los datos históricos.
- Conseguir niveles más altos de madurez en CMM y niveles en ISO 15504.

Metodología de la Investigación

En la elaboración de la presente tesis se efectuaron las siguientes actividades:

- Estudio de las métricas de software
- Estudio de los modelos de proceso de software
- Definición de un modelo de proceso de medición: *modelo cuantitativo*
- Diseño del modelo de proceso de medición para una organización
- Presentación del modelo genérico cualitativo multinivel: *modelo cualitativo.*

- Integración entre el *modelo cualitativo y cuantitativo.*
- Estudio de la tecnología actual que permita crear una herramienta que soporte la integración de los *modelos cualitativo y cuantitativo.*
- Análisis y diseño de la herramienta desarrollada.

Estructura de la Tesis

El presente trabajo esta dividido en: aspectos teóricos sobre las métricas de software, definición y diseño de un modelo de proceso de medición: *modelo cuantitativo*, presentación del *modelo cualitativo multinivel*, integración de los modelos en un solo ambiente de trabajo, marco tecnológico para la creación de una herramienta que soporte la integración de los modelos, análisis y diseño de la herramienta y conclusiones.

En el Capítulo I, se presentan algunos conceptos fundamentales sobre las métricas de software, la necesidad de utilizar éstas dentro del proceso de desarrollo, su definición y cuáles deben ser las características ideales que permitan desarrollar modelos que sean aptos para pronosticar el proceso o parámetros del producto. Se describe además, en forma muy general la clasificación de las métricas, así como la escala a utilizar basándose en los tipos de datos de medición seleccionados.

En el Capítulo II, se incorporan las métricas de software dentro de un proceso de medición. Basados en la literatura actual, se propone un modelo de medición que sea fácil de entender y que provea instrucciones claras acerca de "qué", "cómo" y "cuándo" medir. Por último, se detallan los diferentes niveles o escenarios donde una organización puede utilizar el modelo propuesto.

En el Capítulo III, se diseña el modelo de medición propuesto en el capítulo anterior, de tal forma que éste pueda ser aplicado dentro de cualquier organización de software. Valiéndonos de Personal Software Process, uno de los modelos de proceso de software más detallados en cuanto a como usar las mediciones como herramienta para evaluar, estimar y

controlar el proceso, se identifica y definen en forma operacional las medidas que deben ser recolectadas.

En el Capítulo IV, se presenta un modelo genérico multinivel de calidad y control de cambios [DeJesús01]. Se inicia identificando los roles que están presentes en el proceso de producción del software. Luego se presentan los diferentes grados de madurez en el *proceso de producción de software* relacionados con el control de calidad de los productos y el control de cambios, con estos modelos se obtienen diversas combinaciones llegando al final a presentar un Modelo Genérico para el Control de Calidad y Cambios en los Productos de Software que incluya todos los roles presentes en el proceso de producción.

En el Capítulo V, se integran el modelo que permite medir el proceso de software (modelo cuantitativo) al modelo genérico de control de calidad y cambios (modelo cualitativo). Se indican las consideraciones que se tomaron en cuenta para la integración y se efectúa la misma.

En el Capítulo VI, se describen los elementos tecnológicos que se tomaron en cuenta para desarrollar una herramienta que soporte la integración de los modelos cualitativo y cuantitativo (Capítulo V). El capítulo inicia con la presentación de la tecnología Java Server Pages (JSP) como medio para implementar la aplicación. Luego se propone el medio para integrar la base de datos dentro de la Web. Enseguida se detallan las características de la herramienta que permite obtener dinámicamente graficas en líneas, barras o sectores a fin de presentar los resultados del registro de métricas. Finalmente se presenta la opción seleccionada para efectuar el intercambio de archivos entre repositorios (local y centralizado).

En el Capítulo VII, se presenta el análisis y diseño orientado a objetos de la herramienta que permite controlar el proceso de un producto de software. Finalmente, se presenta las conclusiones a las que se llegaron durante la elaboración de la tesis, así como los trabajos posteriores sugeridos.

Capítulo 1

Introducción a las métricas de software

Este capítulo trata sobre conceptos fundamentales de las métricas de software, la necesidad de utilizar éstas dentro del proceso de desarrollo, su definición y cuáles deben ser las características ideales que permitan desarrollar modelos que sean aptos para pronosticar el proceso o parámetros del producto. Describiremos además, en forma muy general la clasificación de las métricas, así como la escala a utilizar basándose en los tipos de datos de medición seleccionados.

1.1 La "Crisis del Software"

Debido a que los costos del hardware continúan en declive, las demandas por nuevas aplicaciones de software siguen incrementándose a una alta velocidad. Tanto el inventario de software existente, como el esfuerzo de mantenimiento siguen creciendo. Al mismo tiempo, existe una significativa falta de calidad en los profesionales del software. Mientras tanto, el desarrollo de software a menudo se caracteriza por:

- Calendarios y costos estimados que son bastante inexactos,
- Pobre calidad del software, y
- Un índice de productividad que se está incrementando más despacio que la demanda por el software.

Esta situación ha sido a menudo remitida como la "crisis del software"

1.2 La necesidad de las métricas de software

Para ofrecer una solución a la crisis del software se requiere de más exactitud en el calendario y los costos estimados, mejor calidad en los productos, y una alta productividad.

Todo esto se puede lograr por medio de más eficacia en el manejo del software, el cual, puede ser facilitado por el creciente uso de las métricas del software. El actual manejo del software es ineficaz porque su desarrollo es extremadamente complejo, y porque tenemos pocas medidas bien definidas y confiables de los productos y procesos que sirvan para guiar y evaluar su desarrollo.

De esta manera, exactitud y efectividad estimadas y planeadas, así como el control son casi imposibles de lograr.

El mejoramiento del manejo del proceso depende del incremento de la habilidad para identificar, medir, y controlar los parámetros esenciales del proceso de desarrollo. Esta es la meta de las métricas de software —la identificación y medición de los parámetros esenciales que conforman el desarrollo del software.

Las métricas y modelos de software han sido propuestos y usados por algún tiempo. Las métricas sin embargo, han sido poco usadas en la forma tradicional y metódica. Resultados recientes indican que una implementación y aplicación consciente de un modelo de métricas de software, puede ayudar a lograr mejores resultados en el manejo, en un corto período (para un proyecto específico) y en un período largo (incrementando la productividad de futuros proyectos).

Un mejor uso de las métricas existentes y el crecimiento del desarrollo de las mismas, aparecen como factores importantes en la resolución de la *crisis del software.*

1.3 Definición de las métricas de software

Esencialmente las métricas de software tratan con el manejo del producto de software y el proceso por el cual es desarrollado. El producto de software debería ser visto como un objeto abstracto que se involucra desde un estado inicial de requerimientos hasta terminar un sistema de software, incluyendo código objeto y fuente y la documentación producida durante el desarrollo. Ordinariamente, las mediciones del producto y proceso de software,

son estudiadas y desarrolladas para el uso y modelación del proceso de desarrollo. La información obtenida de las métricas y el modelo, pueden usarse en el manejo y control del proceso de desarrollo, dirección y búsqueda de mejores resultados.

Las métricas deberían facilitar el desarrollo de modelos que sean aptos para pronosticar el proceso o parámetros del producto, y no solo para describirlo. De esta manera las métricas ideales serían:

- simples, definidas en forma precisa —de tal manera que sea claro como será evaluada la métrica;
- objetivas, para una posible extensión;
- obtenibles fácilmente (a un costo razonable);
- válidas —la métrica debe medir lo que se pretende medir; y
- robustas —relativamente insensibles a cambios insignificantes en el proceso o producto.

Además, para una máxima utilidad en los análisis estadísticos, las métricas deberían tener valores de información que pertenezcan a escalas de medición apropiadas.

Ha sido observado que las cualidades fundamentales requeridas de cualquier sistema técnico son:

- funcionalidad – correctés, confiabilidad;
- desempeño – tiempo de respuesta, velocidad, etc.
- economía – efectividad en costos.

El desempeño es ciertamente importante pero generalmente no es incluido en las discusiones de las métricas de software, excepto si se conoce los requerimientos específicos de desempeño para ese producto.

La evaluación del desempeño es a menudo tratado por aquellos comprometidos en los estudios de evaluación del funcionamiento, pero esto generalmente no es incluido en lo que se refiere a las métricas de software.

1.4 Clasificación de las métricas de software

Las métricas de software podrían ser ampliamente clasificadas como: *métricas de producto o métricas de proceso.* Las primeras, son medidas del producto de software en cualquier estado de su desarrollo, desde los requerimientos hasta la instalación del sistema. Las métricas del producto pueden medir la complejidad del diseño de software, el tamaño final del programa (código objeto y fuente) o el número de páginas de la documentación producida. Las métricas del proceso, por otra parte, son medidas del proceso de desarrollo de software, como el tiempo de desarrollo, tipo de metodología usada, o nivel de experiencia del personal de programación.

Además de la distinción entre métricas de producto y proceso, las métricas del software pueden ser clasificadas de otras maneras. Se puede distinguir las propiedades *objetivas* de las *subjetivas*. Las métricas objetivas pueden siempre resultar en valores idénticos para una métrica dada, como medida efectuada por dos o más observadores calificados. Para las métricas subjetivas, aún los observadores calificados pueden medir diferentes valores para una métrica, dado que su juicio subjetivo está involucrado para llegar al valor de la medición.

Para las métricas de un producto, el tamaño del mismo en líneas de código (LOC) es una medida objetiva, para la cual cualquier observador informado trabajando bajo la misma definición de LOC podría obtener el mismo valor de medición que un programa dado.

Un ejemplo de una medida subjetiva de un producto es la clasificación del software como "orgánico", " semi-imparcial", o "implantado", como se requiere en el modelo de costo de estimación COCOMO[Boehm81, Boehm84]. Aunque la mayoría de los programas pueden ser fáciles de clasificar, aquellos que se encuentran en el límite entre categorías pueden ser

clasificados de diferentes formas por observadores con conocimientos diferentes. La Tabla 1.1, muestra los valores para las constantes a y b (dependientes del sistema escogido), utilizadas para calcular el esfuerzo. El tamaño es dado en términos de KDSI –Miles de Líneas de Código Entregadas (Thousands of Delivered Source Instructions) y el esfuerzo es calculado en personas-mes.

Tipo sistema Esfuerzo = a (tamaño) ^{b}	**Constantes dependientes del tipo de sistema**	
	a	**b**
Orgánico: se refiere a un sistema stand-alone.	2.4	1.05
Semi-imparcial: se refiere a sistemas entre orgánico e implantado.	3.0	1.12
Implantado: se refiere a sistemas en tiempo real.	3.6	1.2

Tabla 1.1. Constantes para cálculo del esfuerzo en el modelo COCOMO

Para las métricas de proceso, el tiempo de desarrollo es un ejemplo de una medida objetiva, y el nivel de experiencia del programador es una medida subjetiva.

Otra forma en que las métricas pueden ser categorizadas es: *métricas primitivas o calculadas.*

Las métricas primitivas son aquellas que pueden ser directamente observadas, tales como el tamaño del programa (LOC), el número de defectos observados en una unidad de prueba o el tiempo total de desarrollo de un proyecto. Las métricas calculadas son aquellas que no pueden ser directamente observadas pero son calculadas de alguna manera por otras métricas.

Ejemplos de métricas calculadas son aquellas que comúnmente son usadas para productividad, tales como las LOC que se produce por persona–mes (LOC/persona-mes), o para la calidad del producto tales como el número de defectos por miles de líneas de código (defectos/KLOC). Las métricas calculadas a diferencia de las métricas simples, están combinadas de otros valores métricos y de esta manera proveen un mejor entendimiento y evaluación del proceso de software.

Un gran trabajo ha sido hecho en algunas áreas, tales como las métricas objetivas y menos en otras áreas, como las métricas subjetivas de un producto.

1.5 Escalas de medida para las métricas de software

Los datos de las métricas de software deben ser reunidos con un propósito específico. Ordinariamente el propósito es el uso en algunos modelos de proceso y esto puede involucrar el uso de información en otros cálculos o análisis estadístico. Antes de que la información sea reunida y usada, es importante considerar el tipo de información involucrada. Los estadistas reconocen cuatro tipos de datos de medición: nominal, intervalos, ordinal y proporción. Los cuatros tipos de información están descritos en la Tabla 1.2.

Tipo de datos	Operaciones posibles	Descripción de datos
Nominal	$=, \neq$	Categorías
Ordinal	$<, >$	Ranking
Intervalos	+, -	Diferencias
Proporción	/	Cero Absoluto

Tabla 1.2. Tipos de datos de medición

A continuación detallamos algunos ejemplos de métricas de software para cada tipo de dato.

Como un ejemplo de *datos nominales*, uno puede medir el tipo de programa que ha sido producido colocándole una categoría de algún tipo: programa de base de datos, sistema operativo, etc. Para tal información no podemos ejecutar operaciones aritméticas de ningún tipo o categorizar los posibles valores en un "orden natural". La única operación posible es determinar si el programa A es del mismo tipo que el programa B.

Tal información se dice que tiene una escala nominal, y el ejemplo particularmente dado puede ser un importante parámetro en un modelo de proceso de desarrollo de software.

Los datos *ordinales*, en contraste, nos permiten categorizar una variedad de valores de información, aunque las diferencias y proporciones entre los valores no son significativas. Por ejemplo, el nivel de experiencia del programador puede ser medido como *bajo, medio y alto*. (para que esto sea una métrica objetiva, se asume que el criterio para la colocación en las diferentes categorías está bien definido, así que diferentes observadores siempre asignarán un mismo valor para cualquier programador dado.)

Los datos de una escala a *intervalos* no pueden ser categorizados, además es posible exhibir las diferencias significativas entre los valores. La medida de complejidad de McCabe [McCabe76] puede ser interpretada teniendo una escala a intervalos.

Por ejemplo, un programa con un valor de complejidad de 6 es 4 unidades más compleja que un programa con complejidad de 2, pero no es probablemente significativo decir que el primer programa es cuatro veces más complejo que el segundo.

Algunos valores de información están asociados con una escala de proporción, la cual posee un cero absoluto y permiten que proporciones significativas sean calculadas. Un ejemplo es el tamaño del programa, en líneas de código (LOC). Un programa de 2000 líneas puede ser razonablemente interpretado como, dos veces la longitud de un programa de 1000 líneas, y los programas pueden obviamente tener longitud cero de acuerdo a esta medida.

Es importante tener conciencia de que la escala de medición está asociada con la métrica dada. Algunas métricas propuestas tienen valores de escala a intervalos, ordinal, o aún nominal.

Capítulo

Proceso de medición del software: Modelo Cuantitativo

Luego de observar las características y ventajas de las métricas de software, incorporamos éstas dentro de un proceso de medición. El capítulo inicia con la identificación de las necesidades de contar con un modelo de proceso de medición, utilizando UML como lenguaje de modelado. Basados en la literatura actual, se propone un modelo de medición que sea fácil de entender y que provea instrucciones claras acerca de "qué", "cómo" y "cuándo" medir. Por último, se detallan los diferentes niveles o escenarios donde una organización puede utilizar el proceso de medición.

2.1 Introducción

Por todos es reconocido que la mejor forma de evaluar y mejorar el proceso de desarrollo del software es la medición; es decir medir el esfuerzo, costos, duración, defectos y cambios [Fenton91].

A pesar de esto se puede observar que la situación real en la industria de software revela un uso limitado de la medición de procesos. Desde hace unos veinte años, un número significativo de métricas de software ha sido propuesto para mejorar el control de productos y comprender las prácticas de desarrollo de software. Desgraciadamente, muy pocas de estas están estrechamente ligadas a una perspectiva de medición. Basili y Weiss (1984) consideran el proceso de medición y su validación, pero no lo acoplan dentro de un proceso de medición de software. Bache y Bazzana (1994), Hetzel (1994) citan un modelo de procesos pero no lo definen ni lo usan. Pfleeger y McGowan (1990) asocian un conjunto de medidas con los niveles de CMM pero no definen éstas en un modelo de procesos.

Lamentablemente, muchas de las métricas de software propuestas no han seguido un ciclo de diseño, a través de una verificación intensiva y adecuada de los datos. En realidad, muchas métricas de software se han basado en intuiciones y no sobre bases comprobables.

Por lo que, un número significativo de "métricas de software" no pueden ser calificadas como procesos válidos de medición.

Basados en la literatura actual, se propone un modelo de proceso de medición que sea fácil de entender y que provea instrucciones claras acerca de "qué", "cómo" y "cuándo" medir.

2.1.1 Necesidad de un modelo de proceso de medición

En la práctica actual de la ingeniería de software cuando se recolectan las mediciones de un proceso, es posible que tengamos cualquiera de estas situaciones.

1. *El proceso no está definido*, las mediciones del proceso son aplicadas y su definición es tomada de la literatura. La definición de medida lleva a la identificación o definición del atributo de una entidad, o la entidad a ser medida [Fenton91]. Una entidad definida en la literatura puede variar de la entidad equivalente en el contexto del proceso, dando como resultado incoherencias entre la definición de la medida y su aplicación en el contexto del proceso real.

Por ejemplo, ¿para calcular el esfuerzo de diseño, como éste es definido, consiste solamente del diseño inicial? o ¿también del trabajo vuelto a efectuar para corregir fallas?. Y ¿cómo es definida una falla?. ¿Una incoherencia en el documento de diseño es una falla o no?. ¿Una incoherencia en un documento de diseño aún no validado es una falla o no? Y, ¿cuándo un documento es considerado como válido?.

Entidades *estándar*, tal como diseño y falla, dependen del contexto. Dado que las medidas de proceso, son en muchos casos recolectadas manualmente por distintas personas, es factible que se llegue a interpretaciones diferentes, produciendo medidas poco confiables.

2. *El proceso es definido en lenguaje natural*, las mediciones del proceso son aplicadas y su definición es tomada de la literatura. El riesgo de incoherencia existe todavía, pero ésta es baja, dado que la definición del proceso sugiere verificar la coherencia. El problema real

aquí es la complejidad del proceso, el cual no es modelado adecuadamente con lenguaje natural, con la desventaja propia de análisis y comunicación. Esto lleva nuevamente al riesgo de que distintas personas den interpretaciones diferentes, produciendo medidas poco confiables.

3. *El proceso es definido en un lenguaje formal o semi-formal.* Las entidades son definidas en un lenguaje adecuado de representación sin demasiada complejidad. El riesgo de diferentes interpretaciones es reducido ampliamente. El problema ahora, deriva en la separación entre la definición de proceso y la definición de medidas que pueden conducir a incoherencias. Una solución acertada es combinar las medidas y la definición de proceso, es decir definir las medidas como una extensión del modelo de proceso.

2.1.2 Lenguaje para el modelado del proceso de medición

Para la especificación del modelo que se propone en este capítulo, se eligió como lenguaje de modelado a UML (Unified Modeling Language) [Booch99], fundamentalmente porque a mas de ofrecer una descripción sobre la información del proceso, describe el aspecto dinámico del mismo. Además es ampliamente usado y sustentado por muchas herramientas comerciales disponibles.

UML facilita la comunicación, pues cada símbolo tiene una notación con un significado bien definido. Permite construir un modelo de forma precisa, no ambiguo y completo.

2.2 Proceso de medición: Modelo de alto nivel

A nivel general, un proceso de medición puede estar conformado por:
Diseño: Antes de medir, es necesario diseñar un método basado en las necesidades de medición.
Implementación: Las reglas del método de medición son aplicadas al software o pieza de software; esta aplicación produce un resultado, el cual es analizado y puede ser explotado

en un modelo cuantitativo o cualitativo (modelo de calidad, presupuesto, productividad, estimación, etc.).

Evaluación: Con los resultados de la medición se toman decisiones a fin de mejorar el proceso.

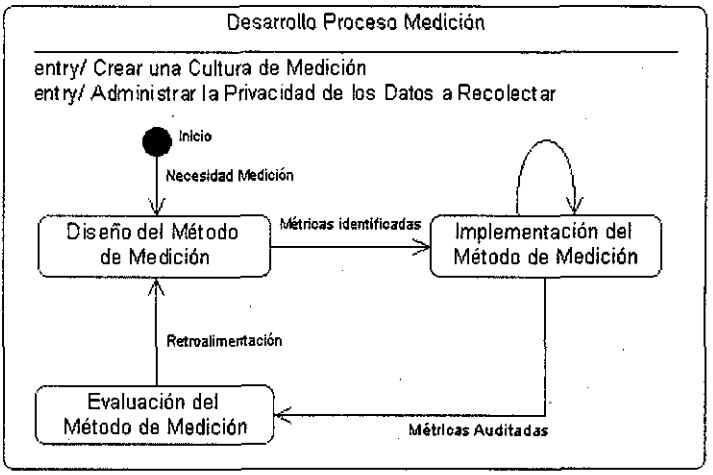


Figura 2.1. Proceso de Medición - Modelo de Alto Nivel

2.3 Proceso de medición: Modelo detallado

El modelo de alto nivel presentado en el punto anterior, así como el estudio de la literatura [McAndrews93], [Morisio95], [Florac97], [Augustine97], [Jacquet97], [Jones00], nos permite identificar los pasos requeridos para proponer un modelo de proceso de medición fácil de entender y que provea instrucciones claras acerca de "qué", "cómo" y "cuándo" medir (Figura 2.2).

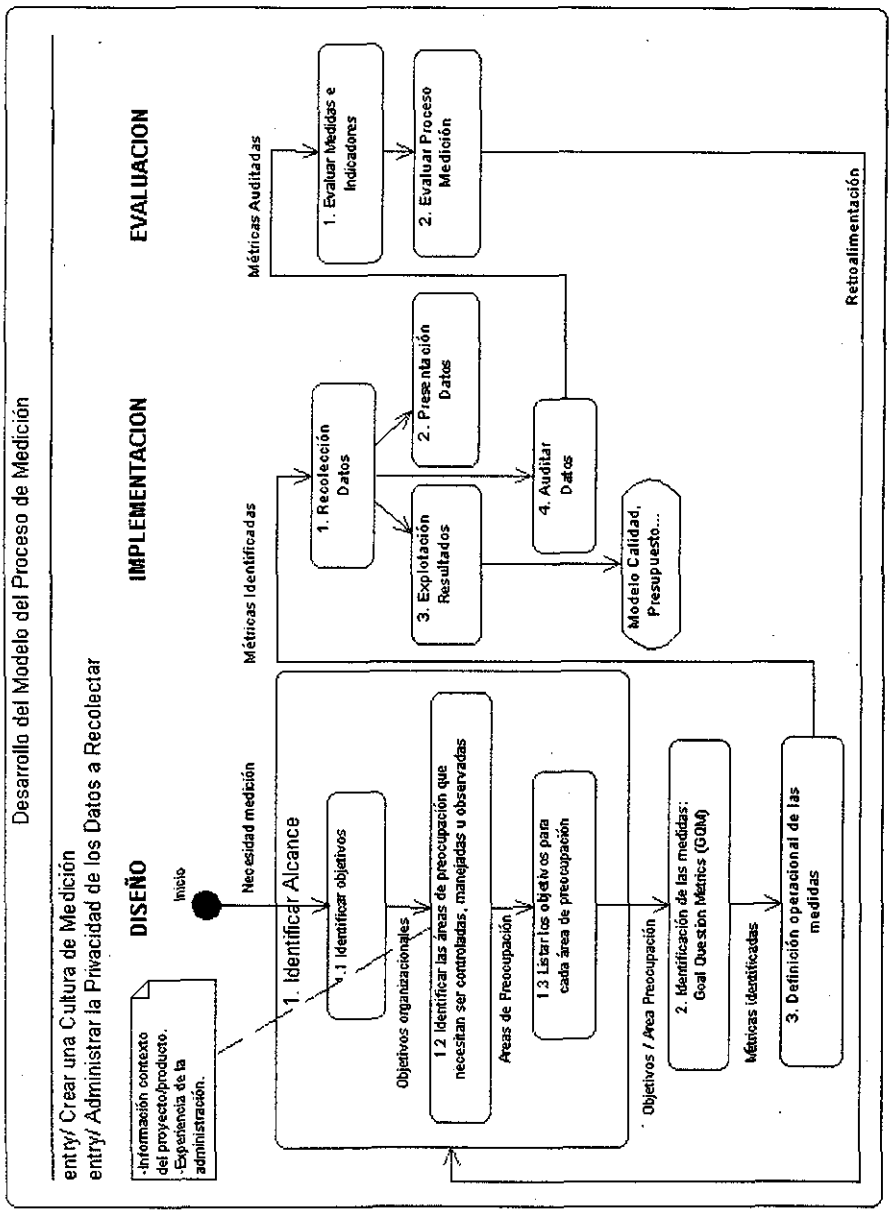


Figura 2.2. Modelo detallado del proceso de medición

Implementar un proceso de medición dentro de una organización es similar a implementar cualquier iniciativa nueva. La medición usualmente representa un cambio significativo en cómo la organización controla su negocio, para lo cual se requiere:

Crear una cultura de medición

El miedo es a menudo la primera reacción de un profesional de software ante un programa de métricas. Las personas tienen miedo que los datos sean usados contra ellos, que tome demasiado tiempo reunir y analizar la información, o que el equipo fije la mirada en conseguir los números correctos antes que en construir buen software.

Para ayudar al equipo a superar el miedo, se les debe educar sobre las métricas a utilizar. Indicarles porque la medición es importante y cómo se intenta usar los datos. Dejar claro que nunca se usarán los datos de métricas para castigar o premiar individuos.

Un administrador de software competente no necesita métricas individuales para distinguir los contribuyentes efectivos del equipo de los haraganes.

Administrar la privacidad de los datos que serán recolectados

Respecto a la privacidad de los datos, es más duro abusar de la información si los administradores no conocen su fuente.

Se pueden clasificar los datos en uno de estos tres niveles de privacidad:

- *Individuo*: sólo el individuo quién reúne los datos sobre su propio trabajo conoce esta información, aunque se puede juntar con los datos de otros individuos a fin de tener un perfil completo del proyecto.
- *Equipo de proyecto*: los datos son privados a los miembros del proyecto, aunque se puede juntar con datos de otros proyectos para proveer un perfil organizativo completo.
- *Organización*: los datos pueden ser compartidos entre todos los miembros de la organización.

Como un ejemplo, si se está reuniendo datos de distribución de esfuerzo de trabajo, el número de horas invertidas por cada individuo en el desarrollo o mantenimiento durante una semana, es privado para ese individuo. La distribución total de horas de todos los miembros del equipo es privada al equipo del proyecto, y la distribución a través de todos los proyectos es pública a toda la organización.

A continuación se presentan las diferentes actividades para diseñar, implementar y evaluar el método de medición propuesto.

2.3.1 Diseño del método de medición

El proceso de medición se origina con una necesidad de medir. La necesidad puede ser grande o pequeña. Por ejemplo: una organización puede necesitar medir el rendimiento de un proyecto corporativo; un proyecto pequeño puede necesitar valorar y evaluar sus inspecciones de diseño y código.

Antes de diseñar un proceso de medición, es importante entender cuáles son las necesidades de medir. Una vez que la necesidad de medición se ha establecido, la primera actividad del proceso debe estar dirigido a *identificar el alcance*. Esta actividad ayuda a establecer el propósito de la medición.

Una vez establecidos los objetivos, mediante el método GQM (Goal-Question-Metric) [Basili 84] se identifica qué métricas son apropiadas para alcanzar las metas propuestas.

Finalmente mediante la actividad *definición operacional de las medidas*, se define las medidas operacionalmente, de forma que todos entiendan completamente lo que los valores medidos representan. La Figura 2.3, detalla esta actividad.

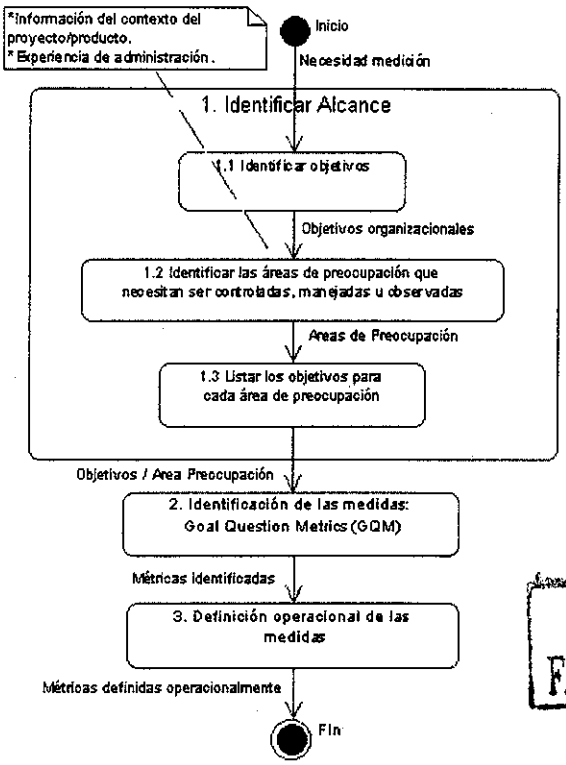


Figura 2.3. Diseño del método de medición

1. Identificar el Alcance

1.1 Identificar los objetivos.

Antes de diseñar un método de medición es importante conocer cómo los objetivos del negocio están relacionados al proceso de medición. Debemos saber, entre otras cosas, qué deseamos medir (qué tipo de software, qué atributos, etc.), desde que punto de vista se utilizará el método (usuario de software, diseñador de software, etc.), y que uso brindará el método de medida. Todos estos criterios tienen una fuerte influencia en el diseño del método de medición.

En la mayoría de los casos las metas y los objetivos de los negocios son referentes al costo, calidad o tiempo, que pueden ser combinados con facilidad al proceso de software apropiado.

1.2 Identificar las áreas de preocupación que necesitan ser controladas, manejadas u observadas.

Un proceso de medición efectivo ayuda al administrador del proyecto a identificar y manejar riesgos que podrían impactar el proyecto. Esta tarea permite identificar las "*preocupaciones*" que pueden impactar en la consecución de los objetivos del proyecto. Estas *preocupaciones* pueden ser identificadas con la información del contexto del proyecto/producto (objetivos, estrategias técnicas, etc.), requerimientos organizativos generales o experiencia de la administración en proyectos/productos anteriores.

Preocupaciones comunes del proceso

Todos los procesos tienen al menos tres, y a menudo cuatro, características en común. La experiencia muestra que las *preocupaciones* de la mayoría de proyectos pueden ser agrupadas en "*áreas de preocupación*" básicas que son comunes a casi todos los proyectos. Estas son:

- calidad del producto (defectos)
- tiempo del proceso o duración (calendarios y progreso)
- tamaño del producto y estabilidad
- recursos y costo del proceso

Estas *preocupaciones* están estrechamente unidas a los atributos de rendimiento del proceso que una organización desea seleccionar para controlar o mejorar.

Las *preocupaciones* —factores importantes para la mayor parte de las organizaciones y comunes a todos los procesos de software— se discuten brevemente a continuación.

Calidad del producto (defectos)

La excesiva variabilidad y los procesos sin objetivo son las causas principales de los defectos. Los defectos introducidos en un producto tienen múltiples efectos. Estos requieren el esfuerzo de un personal hábil para detectar, remover, reparar, y volver a probar el producto. Los defectos también incrementan el costo del proceso, tiempo, y complejidad. Además, los defectos que escapan a la detección y reparación antes que el producto es liberado al cliente, reducen la satisfacción del usuario con el producto. Las actividades que disminuyen la introducción de defectos o aumentan la detección temprana de los mismos, son los objetivos primarios para medir la eficacia del proceso.

Duración del proceso (calendarios y progreso).

La duración es una característica que se traduce directamente como un atributo del proceso. Es el tiempo transcurrido desde el principio del proceso hasta que el producto está disponible al usuario. El usuario en este caso no es necesariamente el cliente o "usuario final". Puede ser una persona o equipo relacionado con las actividades subsecuentes del proceso. Un proyecto que tiene problemas con el horario (calendarios) puede tener que eliminar la funcionalidad o sacrificar la calidad para mantener el calendario de entrega. El flujo de trabajo a través de un proceso dado, a menudo puede ser mejorado proveyendo mejores herramientas, usando mejor tecnología, haciendo más efectivo el uso del tiempo, eliminando pasos innecesarios, y mejorando el entrenamiento.

Tamaño del producto y estabilidad.

Esta preocupación se relaciona con el tamaño del producto. La estabilidad incluye cambios en el alcance o cantidad. Un incremento en el tamaño del sistema, usualmente requiere un incremento en los recursos o la ampliación de los calendarios del proyecto.

Recursos y costo del proceso.

Esta preocupación está relacionada al balance que debe existir entre el trabajo a ser ejecutado y el recurso de personal asignado al proyecto. Como en cualquier otra actividad, el costo de un proceso de software tiene muchos elementos y es afectado por una variedad de factores. Ejemplos incluyen la naturaleza del producto, destrezas y experiencia del

personal, gastos generales, etc. La clave para manejar los costos del proceso, es identificar los elementos del costo que se pueden controlar o afectar y buscar oportunidades para reducirlos o eliminarlos.

1.3 Listar los objetivos para cada área de preocupación.

Listar los objetivos en términos del rendimiento del proceso ayuda a identificar áreas de preocupación potenciales. El mejor camino para hacer esto, es desde el punto de vista de los atributos de los productos o procesos que se desea controlar o mejorar.

2. Identificación de las medidas

El aspecto más difícil de la medición es decidir qué medir. El método GQM (Goal-Question-Metric) es una buena técnica para decidir que métricas son apropiadas para alcanzar las metas de un esfuerzo específico [Basili 84]. Para cada objetivo, se identifica un conjunto de preguntas cuyas respuestas indican si la meta está siendo alcanzada. Por ejemplo, la Figura 2.4 ilustra como algunas métricas pueden ser generadas a partir de un objetivo específico.

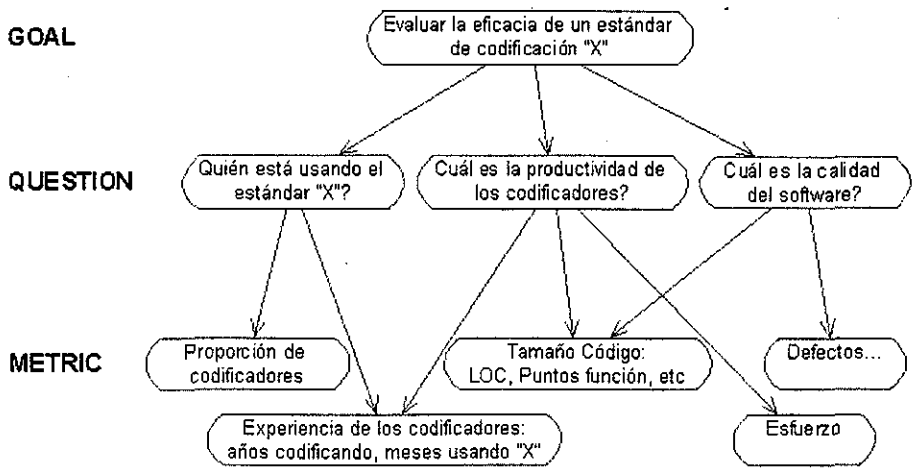


Figura 2.4. Ejemplo de métricas derivadas a partir de metas (goals) y preguntas (questions)

En la figura se observa que la meta es evaluar la eficacia de usar un estándar de codificación "X". Para decidir si el estándar es efectivo, varias preguntas deben ser efectuadas. En primer lugar, es importante saber quién está usando el estándar, de modo que se pueda comparar la productividad de los codificadores que están utilizando el estándar con aquellos que no. Así mismo, probablemente se desea comparar la calidad del código producido con la utilización del estándar con aquel que no utilizo el mismo. Para resolver estos asuntos, es importante hacer preguntas sobre productividad y calidad.

Una vez que las preguntas son identificadas, se debe analizar cada pregunta para determinar lo que se debe medir a fin de responder las preguntas. Por ejemplo, para comprender quien está usando el estándar, es necesario saber que proporción de codificadores está usando la norma. Sin embargo, es importante también tener un perfil de la experiencia de los codificadores, aclarando cuánto tiempo ellos han trabajado con la norma, el entorno, el lenguaje, y otros factores que ayuden a evaluar la eficacia del estándar. La pregunta sobre la productividad requiere una definición de la misma, la cual se define normalmente como la división entre cierta medida del esfuerzo para cierta medida del tamaño del producto (ver actividad selección y definición de las medidas más adelante). Como se muestra en la figura, la métrica puede estar en términos de líneas de código, puntos de función, o cualquier otra métrica que pueda ser utilizada. De manera similar, la calidad se puede medir desde el punto de vista del número de errores encontrados en el código.

De este modo, se generán solamente las medidas que están relacionadas con las metas. Hay que notar que, en muchos casos, varias medidas pueden necesitar responder una sencilla pregunta. Igualmente, una medida sencilla puede aplicar a más de una pregunta.

3. Definición operacional de las medidas

La identificación y definición de las medidas son dos actividades diferentes pero estrechamente relacionadas. Una cosa es identificar una medida como el número de defectos reportados, que puede ser usado para caracterizar un proceso o producto, y otra

crear una definición operacional de una medida seleccionada.[1] Por ejemplo, una definición operacional para el número de defectos encontrados podría incluir el tipo de defecto, cuando ocurrió, cuando fue detectado, la actividad de descubrimiento, severidad, y una descripción de los procesos usados para encontrar, clasificar, y contar las ocurrencias del defecto.

Para tener una definición operacional, se debe conocer lo suficiente acerca de lo que representan los datos y cómo éstos son recolectados, a fin de:

- asegurar que personas diferentes implementen las medidas en forma correcta y consistente; y
- para interpretar los datos correctamente.

Las actividades de identificación y definición de las medidas están acopladas en la práctica porque no es inusual tener que reconsiderar una medida debido a la incapacidad de la organización para aplicarlo fácilmente o consistentemente, una vez que ha sido operacionalmente definido.

Definiendo las medidas del proceso

Una vez que se han identificado las medidas, se las debe definir; los nombres no son suficientes. El concepto a ser medido debe ser claramente definido. Por ejemplo, si consideramos dos distancias, somos capaces de decir si éstas son iguales o no, y, de lo contrario, indicar cuál es más grande. Somos capaces de hacer esto sin usar números, sin medir, porque el concepto de distancia entre dos puntos es claro y preciso. Además debemos poder decir exactamente a otras personas, cómo es obtenida cada medida, a fin de que se pueda coleccionar e interpretar los valores correctamente.

Una solución es, la utilización de formas o tablas que ayuden a definir, implementar, y comunicar las definiciones operacionales. El asunto fundamental no es establecer si la

[1] las definiciones operacionales le dicen a las personas cómo las medidas son hechas, y proporcionan suficiente detalle, tal que otros obtengan los mismos resultados si siguen los mismos procedimientos.

definición para una medida es correcta, sino más bien que todos entiendan completamente lo que los valores medidos representan. Sólo así podemos esperar que las personas recolecten los valores consistentemente, pudiendo además tener otros intérpretes cuya aplicación de resultados nos permitirá alcanzar conclusiones válidas.

Criterios para las definiciones operacionales.

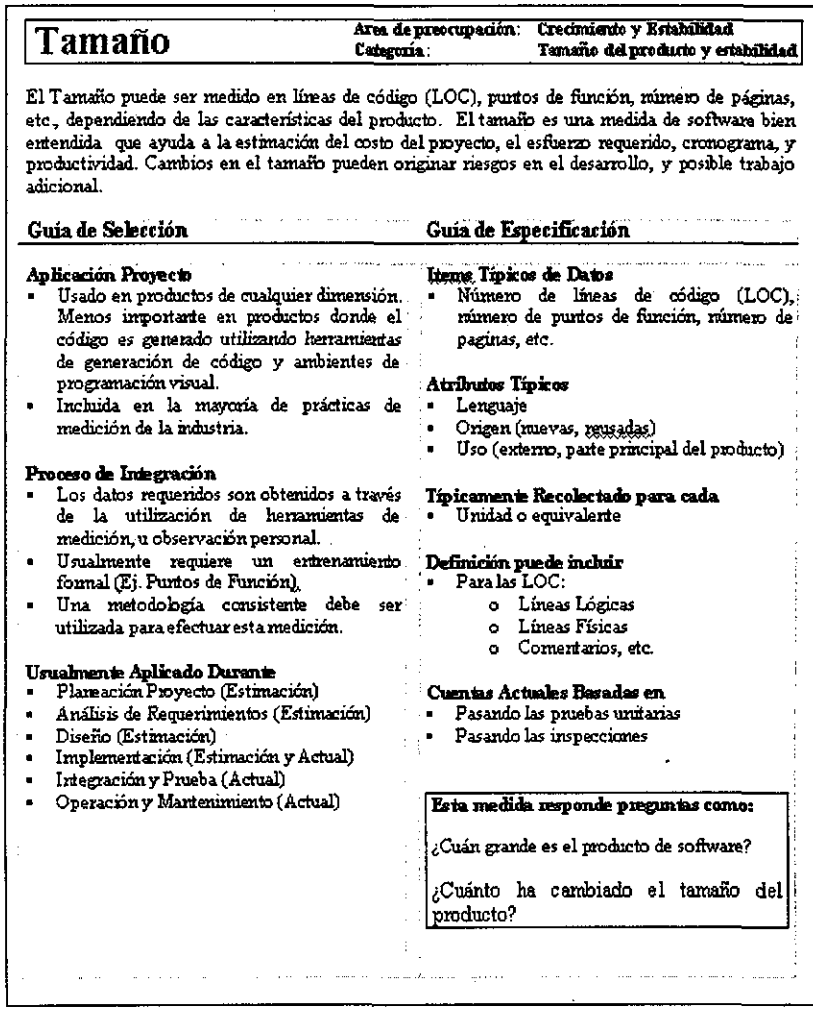
Las definiciones operacionales deben satisfacer dos criterios importantes [Park 92]:

- *Comunicación*: Si alguien usa la definición como una base para medir o describir el resultado de una medición, entonces otros deben conocer en forma precisa ¿qué ha sido medido, cómo fue medido, y qué ha sido incluido y excluido?
- *Repetibilidad*: ¿Pudieron otras personas, basados en la definición, repetir las medidas y obtener los mismos resultados?

Estos criterios están estrechamente relacionados. De hecho, si no se puede comunicar exactamente lo que fue realizado para coleccionar un conjunto de datos, no se podrá comunicar a alguien más cómo hacerlo. Muchas organizaciones proponen definiciones de medida sin determinar primero qué desean conocer los usuarios acerca de los valores medidos. De esta forma las medidas son a menudo recolectadas inconsistentemente y en contradicción con las necesidades de los usuarios. En lo que se refiere a la implementación, reglas como, " contar líneas de código sin comentarios, declaraciones sin líneas en blanco" o "contar todos los problemas abiertos" están expuestas a interpretaciones que obtengan resultados repetibles.

Ejemplo de definiciones operacionales

Una forma clara de explicar cómo utilizar las medidas identificadas, es la utilizada por el Departament of Defense and US Army en su método PSM (Practical Software and Systems Measurement) [Jones00]. PSM dispone de plantillas para definir cada una de las mediciones, tanto en sus características en función del proyecto y producto a desarrollar, como en los procedimientos para recopilación y análisis. Estas plantillas han sido adoptadas como parte de este trabajo y se detallan en el Anexo A. La Figura 2.5, muestra un ejemplo de una tabla que describe el tamaño del producto.



Tamaño

Área de preocupación: Crecimiento y Estabilidad
Categoría: Tamaño del producto y estabilidad

El Tamaño puede ser medido en líneas de código (LOC), puntos de función, número de páginas, etc., dependiendo de las características del producto. El tamaño es una medida de software bien entendida que ayuda a la estimación del costo del proyecto, el esfuerzo requerido, cronograma, y productividad. Cambios en el tamaño pueden originar riesgos en el desarrollo, y posible trabajo adicional.

Guía de Selección

Aplicación Proyecto
- Usado en productos de cualquier dimensión. Menos importante en productos donde el código es generado utilizando herramientas de generación de código y ambientes de programación visual.
- Incluida en la mayoría de prácticas de medición de la industria.

Proceso de Integración
- Los datos requeridos son obtenidos a través de la utilización de herramientas de medición, u observación personal.
- Usualmente requiere un entrenamiento formal (Ej. Puntos de Función).
- Una metodología consistente debe ser utilizada para efectuar esta medición.

Usualmente Aplicado Durante
- Planeación Proyecto (Estimación)
- Análisis de Requerimientos (Estimación)
- Diseño (Estimación)
- Implementación (Estimación y Actual)
- Integración y Prueba (Actual)
- Operación y Mantenimiento (Actual)

Guía de Especificación

Items Típicos de Datos
- Número de líneas de código (LOC), número de puntos de función, número de páginas, etc.

Atributos Típicos
- Lenguaje
- Origen (nuevas, reusadas)
- Uso (externo, parte principal del producto)

Típicamente Recolectado para cada
- Unidad o equivalente

Definición puede incluir
- Para las LOC:
  - Líneas Lógicas
  - Líneas Físicas
  - Comentarios, etc.

Cuentas Actuales Basadas en
- Pasando las pruebas unitarias
- Pasando las inspecciones

Esta medida responde preguntas como:

¿Cuán grande es el producto de software?

¿Cuánto ha cambiado el tamaño del producto?

Figura 2.5. Ejemplo de una tabla que describe el tamaño del producto (PSM)

2.3.2 Implementación del método de medición

Una vez que el método de medición se ha diseñado, éste puede ser implementado. Para efectuar esta tarea se debe ejecutar las siguientes tareas:

- recolección de datos;
- presentación de los resultados de medida;
- explotación de los resultados; y
- auditoría de los datos.

Cada una de estas tareas se describe a continuación. La Figura 2.6 detalla esta actividad.

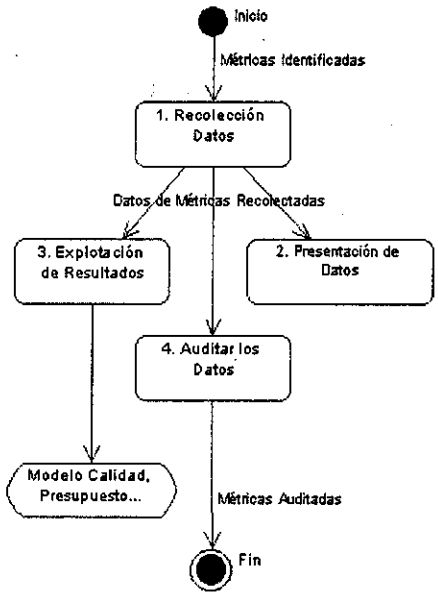


Figura 2.6. Implementación del método de medición

1. Recolección de Datos

Esta actividad implementa los procedimientos de recolección definidos en la actividad *definición operacional de las medidas.*

Usando la definición operacional de los procedimientos de medida, los datos son coleccionados, registrados, y almacenados. Además, los datos son validados y los procedimientos son revisados para adecuación.

2. Presentación de los resultados de medida.

La implementación del método de medición permite que un resultado sea obtenido. Para que esta información resulte efectiva, los datos deben ser presentados en un formato que brinde una interpretación clara. El resultado debe ser diseñado para:

- Soportar el tipo de análisis requerido (estimación, factibilidad, o rendimiento).
- Proveer un nivel apropiado de detalle.
- Proveer información puntual a fin de efectuar decisiones y tomar acciones.

Estos indicadores son a menudo representados como una gráfica o tabla. A continuación se provee un ejemplo de cómo pueden ser representados los resultados del proceso de medición. La figura 2.7, muestra la historia acumulada, semana-a-semana del estado de los problemas encontrados durante la integración y pruebas del sistema.

Esta gráfica permite identificar el número de problemas encontrados hasta la fecha, la tasa de ocurrencia, y el tiempo invertido para corregir los problemas dado un conjunto predeterminado de casos o métodos de prueba. También proporciona un indicador referente al progreso de la integración y prueba. Además, muestra el total acumulado de problemas reportados hasta la fecha, y el número de problemas abiertos y cerrados para cada semana del período de prueba. La línea discontinua muestra el número de problemas abiertos aún no evaluados.

Además los datos le indican al administrador del proyecto, que la tasa de problemas localizados esta alrededor de ocho por semana, mientras que los problemas cerrados están alrededor de cuatro por semana. El tiempo de descubrimiento antes de dar por cerrado un problema es cerca de siete semanas, las cuales podrían tener un impacto en la capacidad de ejecutar casos de prueba si la ejecución es dependiente de la solución de los problemas abiertos.

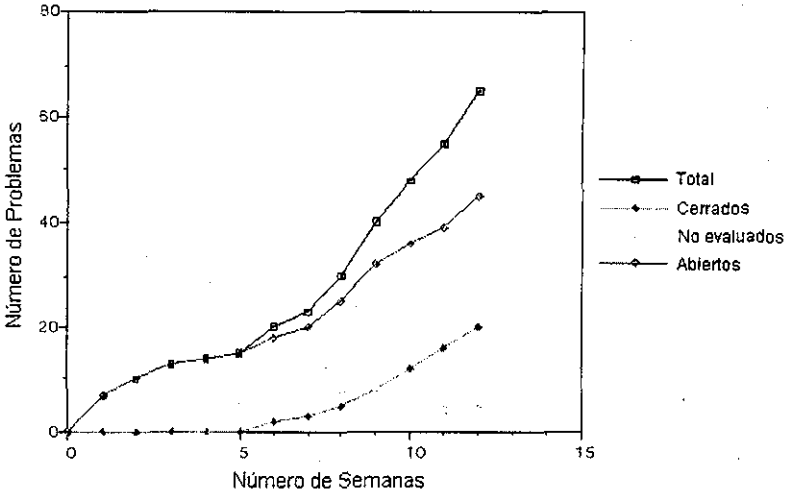


Figura 2.7. Ejemplo de cómo representar los resultados del proceso de medición.

3. Explotación del resultado

El resultado de la aplicación del método de medición puede ser usado de diferentes formas, muchas de las cuales posiblemente no fueron previstas en la fase de diseño.

Usos potenciales de un resultado de medida de software pueden ser: modelos de calidad, modelos de presupuesto, o un proceso de estimación basado en un modelo de productividad y estimación.

4. Auditar los resultados

Los resultados se deben revisar mediante diferentes métodos para averiguar su calidad. Por ejemplo, las regiones compuestas de cálculos matemáticos se deben verificar.

Los resultados pueden ser comparados con otros valores bien conocidos a fin de verificar su validez.

2.3.3 Evaluación del método de medición

Esta actividad es usada para mejorar el proceso y requiere tomar decisiones basándose en los datos de las mediciones. Estas decisiones pueden involucrar replanificación, ejecución de acciones correctivas, o simplemente continuar el proceso sin cambio. El resultado de esta actividad es:

- El reporte final de medición es aceptado para el período actual; y
- Una realimentación es obtenida para desarrollar los planes y procedimientos de medición.

Cualquier realimentación de los tipos mencionados enseguida, debe ser efectuada durante el siguiente ciclo de reportes:

- Los procedimientos de medida definidos pueden necesitar ser actualizados como resultado de revisar los métodos de recolección.
- Los planes de medida pueden necesitar actualizarse para reflejar asuntos adicionales presentados durante el análisis.
- No existió realimentación del reporte de medida, puesto que el mismo no se utilizó.

Las tareas necesarias para ejecutar esta actividad se detallan en la Figura 2.8.

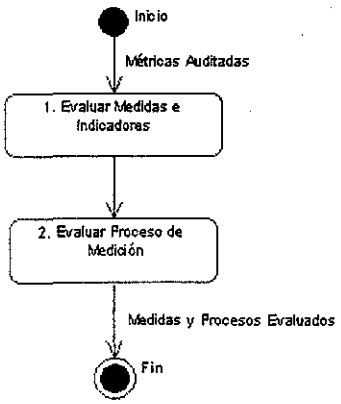


Figura 2.8. Evaluación del método de medición

1. Evaluar las medidas e indicadores

Para saber si los datos e indicadores satisfacen las necesidades de información de los administradores del proyecto, es necesario evaluar los productos del proceso de medición: medidas, indicadores y análisis de resultados. La efectividad de estos productos de medición debe ser evaluada con criterios predefinidos. El borrador del estándar ISO/IEC 15939 (Software Measurement Process), provee una serie de criterios subjetivos que permiten evaluar los productos del proceso de medición, así por ejemplo:

- Uso de los productos de medición
- Confianza en los resultados de la medición.
- Adaptabilidad de los resultados con el propósito de la medición.
- Comprensibilidad de los resultados.
- Exactitud de la medida.
- Fiabilidad de la medida.

2. Evaluar el proceso de medición

Aún un programa de medida basado en medidas e indicadores apropiados y probados puede resultar ineficaz debido a la ineficacia del proceso de medición. Es posible evaluar el proceso desde estas tres perspectivas:

Desempeño –medición de las entradas, salidas y efectos del proceso de medición. El proceso de medición por si mismo está sujeto a medición. Los criterios que pueden ser usados para evaluar el desempeño del proceso de medición son:

- Puntualidad;
- Eficiencia;
- Contención de defectos; y
- Satisfacción del cliente.

Conformidad –comparación entre el proceso de medida y el uso pretendido.

Capacidad –comparación entre el proceso de medida y un estándar externo de madurez de proceso.

2.4 Niveles de utilización del modelo de medición

Existen diferentes niveles o escenarios donde la organización puede aplicar el modelo de medición detallado anteriormente. El propósito de este trabajo de tesis pretende que inicialmente sirva para *describir productos* y *mejorar el proceso*. La Figura 2.9, ilustra algunos usos de las medidas.

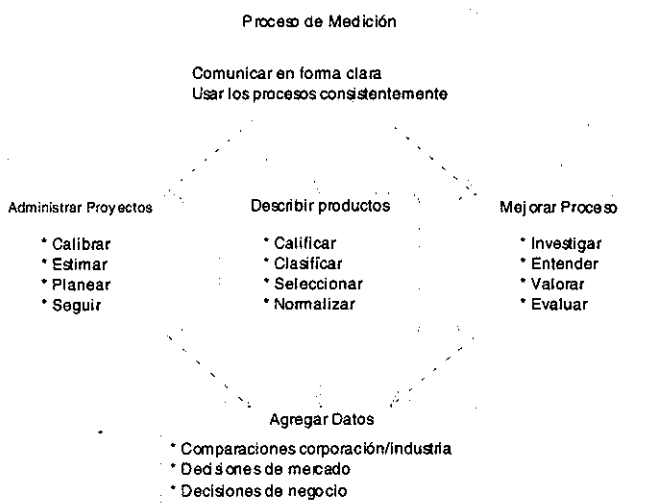


Figura 2.9. Usos del modelo de medición

Proceso de Medición

Un programa de medición efectivo inicia con un *proceso de medición* que puede ser usado consistentemente por todas las personas encargadas de tomar decisiones a través de la organización.

Los métodos descritos en el modelo propuesto, pueden ser usados para diseñar un *proceso de medición* que incluya:

- objetivos comunes de gestión;
- medidas básicas (ejemplo: tamaño, esfuerzo, tiempo, defectos);
- definiciones no ambiguas;
- procedimientos de recolección;
- técnicas de análisis;
- mecanismos de evolución (ejemplo: revisiones del progreso mensual); y
- roles y responsabilidades.

Con un proceso de medición definido, es posible establecer una colección de medidas sumamente sólida. Como las medidas básicas evolucionan, sus definiciones y usos a menudo se expanden, por ejemplo:

- Los reportes de problemas pueden expanderse a estado de seguimiento, tipo, severidad, y prioridad;
- Los atributos de tamaño pueden ser rastreados por el lenguaje, plataforma, estado de desarrollo, origen, y método de producción;
- Los atributos de esfuerzo pueden agregarse para rastrear la clase laboral, fase, y actividades realizadas; y
- El tiempo puede ser rastreado por fechas y criterios de terminación.

Estas medidas básicas pueden forjarse a fin de ser aún más efectivas en *proyectos directivos, descripción de productos, y mejora de procesos*. Estos usos de las medidas se discuten a continuación.

Administrar Proyectos

Con un proceso de medición repetible, los administradores pueden comenzar a confiar en las medidas básicas que usan para manejar sus proyectos. Usando los métodos del modelo, un elemento del proceso de medición puede ser definido para:

- Recolectar las medidas básicas en proyectos completos, a fin de que una base de datos pueda ser establecida para ayudar a entender la capacidad del proceso de software actual;
- Usar los datos históricos para calibrar los modelos de estimación del software;
- Efectuar estimaciones reales y planear el tamaño, esfuerzo, y tiempo para proyectos futuros, basados en los datos históricos;
- Usar los datos actuales versus los planeados para evaluar: cuánto se ha completado del proyecto, cuánto queda por hacer, y cuándo el proyecto estará completo; éstas evaluaciones permitirán efectuar decisiones suponiendo recursos y progreso; y
- Volver a planear proyectos basados en desviaciones del estado, progreso, o renegociación de requisitos.

Describir Productos

Una vez que el administrador entiende las medidas básicas que le permiten administrar el desempeño y capacidad del proyecto, el siguiente nivel involucra la descripción de los productos en términos de estas medidas básicas.

Como una organización define los procesos que serán usados a través de todos los proyectos, éstos pueden iniciar el enfoque de medición con la calidad y descripciones del producto. Las descripciones del producto se usan para calificar qué tan bueno es, clasificar sus características, y seleccionarlo de acuerdo a las especificaciones del usuario. Algunos ejemplos de descripciones del producto incluyen mantenibilidad, confiabilidad, y densidad de problemas. Existen algunas formas de describir productos en términos de las relaciones existentes entre las medidas básicas (ejemplo, la densidad de defectos = defectos/tamaño).

Algunos ejemplos de cómo diferentes usuarios pueden usar el proceso de medición para describir productos incluyen:

- Los administradores pueden entender el nivel de calidad de sus productos, así como la calidad que los procesos existentes son capaces de alcanzar;

- Los desarrolladores pueden usar las descripciones del producto para entender la calidad de su trabajo e identificar potencialidades y debilidades durante el proceso; y
- Los clientes pueden describir los productos en sus especificaciones de requisitos para indicar el nivel deseado de calidad.

Mejorar el proceso

Otro objetivo común de los administradores es mejorar la capacidad de la organización [Humphrey89]. Una vez que los administradores entienden las medidas básicas, usan éstas para manejar los productos y procesos existentes, pueden iniciar el enfoque con la mejora de procesos. Aquí, las medidas son usadas para investigar y entender, y para enfocar la atención en áreas problemáticas a mejorar. Por ejemplo, si en los proyectos se localiza un alto porcentaje de problemas a partir de las pruebas de sistema o unitarias, entonces la organización puede necesitar implementar un proceso de inspección de código y diseño más riguroso, o proveer entrenamiento en las unidades de prueba (ejemplo, enfocar mejoras en fases más tempranas del proceso de desarrollo).

Agregar Datos

Los métodos descritos en el modelo propuesto, pueden usarse para identificar y definir medidas que ayuden a tomar decisiones con relación a los objetivos y metas organizacionales. Con las medidas se puede entender el proceso del software y las capacidades organizativas, e involucrarse con los aspectos comerciales del software. Esto típicamente implica que algunos datos de los proyectos se agreguen dentro de la organización para ensamblar información referente a: productividad (tamaño/esfuerzo), calidad (antes y después de la liberación), y previsibilidad [Grady87].

A menudo, las organizaciones de software están obligadas a reportar las mediciones como parte de su política. Para que los proyectos den cuenta y reflejen su progreso en forma más exacta, estos deben tener planes y procedimientos documentados que comuniquen

exactamente lo que representan los datos. Sin esta consistencia, el uso de la agregación resulta cuestionable.

Resumen:

En este capítulo se propuso un modelo de proceso de medición basado en el estudio de las propuestas identificadas de la literatura actual. Como lenguaje de modelado, se utilizó UML. Además se detalló cada una de las actividades para implementar el proceso de medición, el cual debe iniciar con un cambio cultural dentro de la organización que desea implementar el modelo. Finalmente, se expusieron los diferentes niveles o escenarios donde una organización puede utilizar el proceso de medición descrito.

Capítulo 3

Diseño del Modelo Cuantitativo: Aplicación a escala general

Una vez identificado el método de medición propuesto en el capítulo anterior, es necesario diseñarlo de forma que pueda ser aplicado dentro de cualquier organización de software. Valiéndonos de Personal Software Process, uno de los modelos de proceso de software más detallados en cuanto a como usar las mediciones como herramienta para evaluar, estimar y controlar el proceso, se identifica y definen en forma operacional las medidas que deben ser recolectadas para obtener un producto de calidad.

3.1 Introducción

Cualquier cambio o mejora en el proceso de software, debe iniciar con objetivos reales y claramente identificados. Estos objetivos pueden ser el resultado de reportes y recomendaciones de evaluación del proceso o algunas otras actividades de mejoramiento del mismo.

La crisis de los productos de software tanto en tiempo como en calidad, obliga a los administradores a utilizar técnicas cada vez más certeras para asegurar el cumplimiento de los objetivos. Las mediciones de software han probado ser una herramienta efectiva al momento de colaborar en la toma de decisiones de los administradores. Cuando las mediciones están basadas en los objetivos de la organización y/o producto, y son integradas en todo el proceso de desarrollo, proveen la información necesaria para identificar y controlar muchas de las áreas de preocupación a los que se enfrenta un administrador de proyectos. Esto facilita la identificación de riesgos, el seguimiento efectivo de problemas específicos, la evaluación del impacto de los riesgos y problemas en el cumplimiento de los objetivos en lo que se refiere a costos, cronogramas, productividad y calidad del producto final; las mediciones también permiten evaluar de una manera más objetiva, alternativas para confrontar las dificultades y seleccionar la solución más adecuada.

El método de medición descrito en el capítulo anterior, tiene como alcance ayudar a cualquier organización a describir los productos de software y mejorar el proceso de desarrollo, para lo cual requiere la definición de objetivos que permitan valorar la eficiencia del proceso y descubrir áreas que necesitan ser trabajadas.

De todos los modelos de proceso de software conocidos internacionalmente, Personal Software Process (PSP) [Humphrey89] es uno de los más detallados y cuyos objetivos están dentro del alcance que se pretende alcanzar con este trabajo. Además PSP posee un conjunto de métricas que a pesar que no cubren todos los aspectos de calidad de un producto de software, al menos se concentran en asegurar fundamentalmente un componente: *funcionalidad.* Este conjunto básico de medidas unidas a otros procedimientos y métricas (definidas al final de este capítulo), permitirán obtener productos con mejor calidad.

Por estos motivos y gracias a algunas ventajas adicionales que se justifican en el punto 3.2.4; PSP servirá como base (identificación de los objetivos) para el diseño a escala general del modelo cuantitativo.

3.2 Personal Software Process (PSP)

El Instituto de Ingeniería de Software (SEI) desarrolló CMM® (Capability Maturity Model), el cual brinda una serie de ideas para incluir las prácticas necesarias para optimizar los procesos. La desventaja de este modelo es que dice muy poco acerca de cómo estas prácticas deben ser implementadas.

Para hacer CMM más tangible, Watts Humphrey de SEI, ofrece un punto de vista diferente desde otra perspectiva. El resultado, Personal Software Process (PSP), un enfoque práctico para mejorar el proceso de software, el cual dirige los hábitos de trabajo de un ingeniero de software y procura llevarlo hasta un nivel 5 de CMM, manejando profundamente la medición.

Para obtener mejoras en el proceso mediante PSP, se requiere una actitud honesta y humilde, y una verdadera motivación para mejorar. Una administración impuesta con PSP es probable que de lugar a una calidad pobre de los datos, pues habría una tendencia a mostrar solamente "buenas" figuras.

3.2.1. Aspectos de Calidad

Antes de examinar Personal Software Process, se presentan algunas definiciones y aspectos de calidad relacionados con éste.

¿Qué es Calidad?

La calidad no es un atributo de un producto o servicio que puede tener un valor absoluto, es relativo a las percepciones y necesidades individuales de los usuarios. Los aspectos de calidad tienen diferentes énfasis de acuerdo a quien lo valora y para qué se lo utiliza.

"Existen esencialmente dos puntos de vista que pueden ser seguidos para asegurar la calidad de un producto, uno es asegurando el proceso mediante el cual el producto es desarrollado, y el otro obteniendo la evaluación del producto final.

Ambas opciones son importantes aunque ambas requieren la presencia de un sistema de administración de calidad" [ISO9126]. Por supuesto ambos aspectos de calidad existen.

Para los productos, estos aspectos son capturados por los modelos de calidad. Para los procesos, existe un conjunto de atributos para estimar la calidad del proceso.

Software de Calidad

Algunos modelos de calidad para software han sido propuestos desde 1970. El ISO 9126 para la calidad del software es un intento para armonizar los conceptos y terminología; éste define a un software de calidad como "la totalidad de rasgos y características de un producto de software que llevan a satisfacer las necesidades" [ISO 9126].

Proceso de Software

El proceso de software es la secuencia de pasos requeridos para producir y mantener software. Éste incorpora los métodos, herramientas, personal y otros recursos envueltos en la producción del software. Todas las organizaciones y equipos manejan procesos, ya sea que estos estén conscientemente definidos o no.

El proceso de desarrollo de software individual, ha sido raramente definido y modelado. Humprey (1990) sugirió que los procesos individuales deben ser comprendidos y modelados de mejor forma, y entonces combinados con procesos organizacionales. PSP procura definir procesos individuales que puedan ser hechos a la medida y que cumplan con los requerimientos.

Calidad del Proceso

La calidad del proceso de software puede ser ampliamente definido como la utilización de los mejores procesos para desarrollar un producto de trabajo. La medida más simple es la calidad del producto. Algunas medidas de procesos miden la calidad del producto antes que éste se concluya, tal que la administración pueda ver si el proceso está produciendo software de calidad.

Además de productos de calidad, la administración usualmente desea al menos estimar y controlar el costo y planes de desarrollo [Humphrey95]. La medición juega un papel central en la mejora de procesos. Sin ésta, es difícil valorar la eficiencia del proceso y descubrir áreas que necesitan ser mejoradas.

Algunas métricas de calidad de procesos nos dan una estimación precisa del costo y el cronograma, el número de defectos encontrados en las diferentes fases del proceso y las fases donde los defectos fueron insertados. A nivel individual, muy pocos atributos son medidos, además de la productividad y el tiempo de desarrollo. Los programas de medición personal pueden ser difíciles de introducir, si el ingeniero piensa que serán usadas para ajustar la supervisión o para efectuar comparación entre empleados.

3.2.2 Descripción general de Personal Software Process (PSP)

La motivación detrás de PSP en la perspectiva de los gerentes de la organización, es el hecho de que el ingeniero de software tiene en gran medida la mayoría de influencia en la calidad. Personal Software Process, se caracteriza por una planificación y administración deliberada, una estimación cuantitativa de recursos y seguimiento, revisiones individuales altamente estructuradas de los productos de trabajo de software, y mediciones frecuentes del producto y proceso, con la meta explícita de *MEJORAR EL DESEMPEÑO DEL PROCESO DE DESARROLLO, Y OBTENER COMO RESULTADO PRODUCTOS DE SOFTWARE DE ALTA CALIDAD.* La Figura 3.1, muestra la estructura de PSP dentro del proceso de desarrollo de un producto de software.

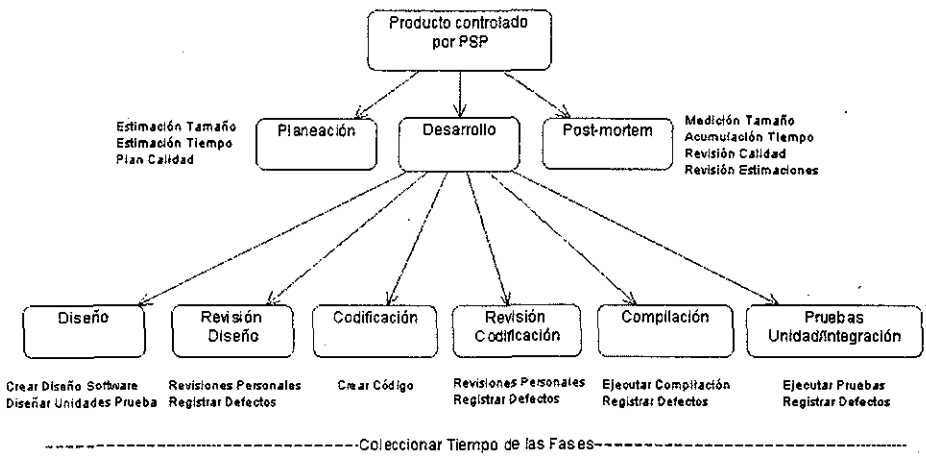


Figura 3.1. Estructura de Personal Software Process dentro del proceso de desarrollo de un producto

3.2.3 Métricas de Calidad en PSP

Existen tres métricas básicas en PSP: tiempo de desarrollo, defectos y tamaño. Todas las otras medidas son derivadas de éstas. La inconsistencia y la variación de calidad de los datos recolectados por diferentes individuos, son algunas de las dificultades que se

presentan dentro de una organización. Afortunadamente en PSP, la valoración y predicción del trabajo son basados en datos personales. Esto hace más fácil asegurar la consistencia de los datos y su relevancia, lo cual es importante para hacer estimaciones confiables y encontrar tendencias significantes a partir de los datos.

Es importante notar que los datos recolectados en PSP son dependientes del programador y lenguaje. Esto resulta bueno para valorar y estimar tendencias personales con un mismo lenguaje, pero pobre para estimar desarrollo en otro lenguaje o comparar por ejemplo productividad entre dos personas.

Defectos

Esta medida refleja la calidad del producto, proporciona al programador un espejo para encontrar áreas que necesitan mejorar. El registro de defectos provee información para numerosas métricas indirectas como: densidad de defectos, número de defectos inyectados y encontrados por fase, y estimaciones para las densidades de defectos en proyectos futuros.

Tiempo

El registro de tiempo proporciona los datos para: valorar el costo del producto, el costo de calidad, y estimar el esfuerzo. La estimación del esfuerzo es especialmente importante para que el programador planee su trabajo. PSP recomienda un registro riguroso del tiempo, identificando las interrupciones que se presenten durante la ejecución de una tarea en particular.

Tamaño

El tamaño del programa es la base de la estimación del esfuerzo en PSP. Algunas medidas derivadas son el nivel de reutilización, densidad de defectos y la productividad. El tamaño en PSP es medido en líneas de código (LOC). En lenguajes orientados a objetos, se registra el tamaño de cada clase, así como el número de métodos de la clase. En lenguajes no orientados a objetos, se registra el tamaño de la función o del procedimiento. Esta granularidad aumenta la exactitud de las estimaciones de tamaño.

3.2.4 Justificación para utilizar PSP

De un estudio efectuado por David F. Rico [Rico99], se desprende que PSP presenta algunas ventajas en comparación con algunas estrategias conocidas para mejorar el proceso de software. Las estrategias que forman parte del estudio se describen a continuación:

- Personal Software Process (PSP) – [Humphrey, 1995].
- Clean Room Methodology – [Pressman, 1997; Kaplan, Clark, y Tang, 1995].
- Software Reuse – [Poulin , 1997]
- Defect Prevention Process – [Jones, 1985]
- Software Inspection – [Fagan, 1976]
- Software Test Process – [Blackburn's, 1998]
- Capability Maturity Model (CMM) [Paulk, Weber, Curtis, y Chrissis, 1995], e
- ISO 9000. [ISO 9000, 2000]

La Tabla 3.1, resume alguna de las ventajas de PSP con respecto al incremento de la calidad, productividad, y reducción del ciclo de tiempo para desarrollar un producto. Así por ejemplo, PSP tiene una ventaja de 20 veces superior a ISO 9000, en lo referente a la calidad de los productos que se obtienen. Las ventajas observadas con este estudio, fueron el sustento para tomar PSP como base para el diseño a escala general del modelo cuantitativo propuesto en el capítulo II.

VENTAJAS DE PSP			
SPI	Incremento calidad (ventaja promedio de 38x)	Incremento productividad (ventaja de productividad de 42x)	Reducción en el ciclo de tiempo para desarrollar un producto (ventaja de 63x)
Clean Room	6x	26x	46x
Software Reuse	59x	41x	45x
Prevention	53x	58x	97x
Inspections	28x	20x	30x
Testing	44x	18x	27x
Software CMM	56x	37x	55x
ISO 9000	20x	97x	144x

Tabla 3.1. Ventajas de PSP con respecto a otras estrategias de mejora de proceso del software (SPI)

3.3 Diseño del Modelo de Medición a Escala General

El modelo propuesto en el Capítulo II, está conformado por 3 fases que detallan los distintos pasos desde el diseño del método de medición hasta su implementación y evaluación. El objetivo del modelo es que pueda ser utilizado a escala general por cualquier organización de software; la fase *Diseño del Método de Medición* (Figura 3.2), nos permitirá definir el alcance (objetivos) y las medidas que necesitan ser recolectadas para alcanzar las metas que se impone una organización. Las fases posteriores *Aplicación y Evaluación del Método de Medición*, es parte del trabajo que se debe tomar en cuenta para que el método de medición se incorpore como parte del proceso de software (Ver Capítulo V).

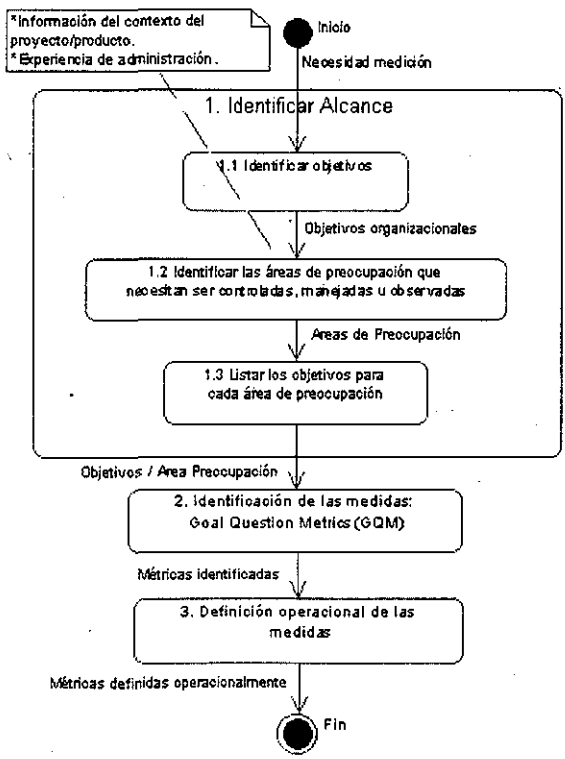


Figura 3.2. Diseño del Método de Medición del modelo propuesto

Como cualquier herramienta para administrar el desarrollo de un producto de software, la utilización de las métricas que se obtendrán como resultado del diseño del método de medición, no pueden garantizar que la elaboración del producto sea exitosa. Pero, sin embargo, ayuda al administrador a tomar decisiones más reales que ayuden a controlar los riesgos y problemas a los que se enfrentará durante el desarrollo. A continuación, se aplica los pasos requeridos para diseñar el modelo que cumpla el objetivo citado anteriormente.

1. Identificar el Alcance

1.1. Identificar los objetivos.

En función de la experiencia revelada en cientos de proyectos de diferentes características que utilizaron PSP como modelo de proceso de software, podemos identificar los siguientes objetivos que pueden ser aplicados dentro de una organización:

Estimar, dar seguimiento y:

- *Administrar* el esfuerzo software
- *Incrementar* la calidad de los productos de software
- *Reducir* el costo desarrollo de software
- *Reducir* la duración del ciclo desarrollo
- *Mejorar* la productividad
- *Eliminar* los defectos antes de las pruebas
- *Minimizar* el costo de las pruebas de software

El objetivo referente a la estimación no será cumplido sino hasta que el método de medición posea datos históricos.

1.2. Identificar las áreas de preocupación que necesitan ser controladas, manejadas u observadas.

El modelo PSM (Practical Software and Systems Measurement) [Jones00], propone agrupar las áreas de preocupación en un conjunto de categorías. Estas categorías son un

grupo de mediciones relacionadas que proveen tipos similares de información acerca de un área de preocupación especifica.

Tomando en cuenta los objetivos que se pueden alcanzar utilizando Personal Software Process (PSP), podemos identificar las siguientes áreas de preocupación que deberían ser controladas, manejadas, u observadas a fin de que el administrador pueda identificar y manejar riesgos que podrían impactar en la consecución de un proyecto, estas son: cronograma y progreso, recursos y costos, crecimiento y estabilidad, calidad del producto, y desempeño del proceso.

Cada área de preocupación posee un conjunto de categorías el cual se muestra en la Tabla 3.2. Las áreas propuestas, hacen referencia a las metas de negocio de cualquier organización y son independientes del método de desarrollo.

Áreas de Preocupación - Categorías	
Área de Preocupación	**Categoría**
Cronograma y Progreso	*Milestone performance*
	Progreso de las unidades de trabajo
Recursos y Costos	Personal
Crecimiento y estabilidad	Tamaño del producto y estabilidad
Calidad de producto	Corrección Funcional
Desempeño del Proceso	Eficiencia Proceso
	Efectividad Proceso

Tabla 3.2. Áreas de preocupación a controlar, manejar u observar

1.3. Listar los objetivos para cada área de preocupación. (Goals)

Una vez identificadas las áreas de preocupación y sus categorías, se propone una serie de objetivos que permitan controlarlas y mejorarlas. Cada uno de los objetivos se detallan en la Tabla 3.3.

Áreas de Preocupación – Categorías - Objetivos		
Área de Preocupación	**Categoría**	**Objetivos**
Cronograma y Progreso	***Milestone performance***	Controlar cambios en el cronograma que ayuden a evaluar el riesgo de alcanzar futuros *milestone*.
	Progreso de las unidades de trabajo	Evaluar el progreso en cualquier punto del desarrollo del producto
Recursos y Costos	***Personal***	Analizar la distribución real del trabajo
Crecimiento y estabilidad	***Tamaño del producto y estabilidad***	Conocer la cantidad y frecuencia de cambios de los productos
Calidad del producto	***Corrección Funcional***	Reducir el número de defectos de software Mejorar la calidad de los productos
Desempeño del proceso	***Eficiencia proceso***	Incrementar la eficiencia del proceso sin comprometer la calidad del producto.
	Efectividad proceso	Mejorar la efectividad del proceso

Tabla 3.3. Objetivos para cada área de preocupación

2. Identificación de las medidas

Una vez que los objetivos para cada área de preocupación se han definido, es necesario identificar las medidas que se deben recolectar dentro de la organización a fin de alcanzar los objetivos propuestos (*ver 1.1. Identificación de los objetivos*).

Lamentablemente la identificación de qué se debe medir es el aspecto más difícil de la medición. El método Goal Question Metrics, permite establecer este proceso, para lo cual se inicia con una serie de objetivos claramente definidos, luego de lo cual se especifica una serie de preguntas cuyas respuestas indicarán si la meta esta siendo alcanzada.

Partiendo de los objetivos establecidos en el punto anterior, detallamos las preguntas.

Preguntas (Questions)

Áreas de Preocupación – Categorías – Objetivos - Preguntas			
Área de Preocupación	**Categoría**	**Objetivos**	**Preguntas**
Cronograma y Progreso	***Milestone performance***	Controlar cambios en el cronograma que ayuden a evaluar el riesgo de alcanzar futuros milestone.	P1. ¿Cuál es la fecha proyectada para terminar el producto? P2. ¿Qué actividades están a tiempo, adelantadas o retrasadas con respecto al cronograma?
	Progreso de las unidades de trabajo	Evaluar el progreso en cualquier punto del desarrollo del producto	P3. ¿Están las solicitudes de cambio siendo implementadas a una proporción suficiente para cumplir el cronograma? P4. ¿Está disminuyendo la tendencia de solicitudes de cambio con la terminación del proyecto?
Recursos y Costos	***Personal***	Analizar la distribución real del trabajo	P5. ¿Están las tareas o actividades, tomando más o menos esfuerzo que el esperado?
Crecimiento y estabilidad	***Tamaño del producto y estabilidad***	Conocer la cantidad y frecuencia de cambio de los productos	P6. ¿Cuánto ha cambiado el tamaño del producto?
Calidad del producto	***Corrección Funcional***	Mejorar la calidad de los productos	P7. ¿Qué porcentaje de esfuerzo es consumido en revisiones y aprobaciones? P8. ¿Cuán efectivas son las revisiones y aprobaciones?
Desempeño del proceso	***Eficiencia proceso***	Incrementar la eficiencia del proceso sin comprometer la calidad del producto.	P9. ¿Cuál es la productividad de los productos? P10. ¿Cuánto tiempo tomará completar una actividad?
	Efectividad proceso	Mejorar la efectividad del proceso	P11. ¿Qué porcentaje de esfuerzo es consumido en trabajo vuelto a efectuar?

Tabla 3.4. Preguntas para alcanzar los objetivos de cada área de preocupación

Cada pregunta se numera de P1 a P11 para proporcionar un seguimiento a las medidas del producto individual y cada área de preocupación.

Métricas Básicas (Metrics)

Una vez que las áreas de preocupación con sus respectivas preguntas asociadas se han identificado, se debe proporcionar un conjunto de medidas que permitan responder a las inquietudes planteadas con el fin de alcanzar los objetivos propuestos. Las medidas que detallamos a continuación son fácilmente obtenibles en cualquier producto de software y como podemos observar, algunas son parte del conjunto de medidas básicas que propone Personal Software Process (PSP) para mejorar la calidad de los productos.

Métricas Básicas		
ID	**Nombre Métrica**	**Preguntas que se puede responder**
M1	Fechas *Milestones*	P1, P2
M2	Estado de Solicitud de Cambios	P3, P4
M3	Esfuerzo actual	P5, P7, P9, P11
M4	Esfuerzo consumido durante las revisiones o aprobaciones	P7
M5	Tamaño	P6, P9
M6	Defectos	P8
M7	Tiempo	P10
M8	Productividad	P9
M9	Retrabajo (esfuerzo consumido en la corrección de defectos)	P11

Tabla 3.5. Métricas que ayudaran a alcanzar los objetivos de cada área de preocupación

Las preguntas P1, P2, P3, P4, P5, P6, P8 y P10 pueden ser respondidas mediante el uso de indicadores (gráficas o tablas) que faciliten la interpretación de los datos. Por otra parte, las preguntas P7, P9, P11 pueden ofrecer respuestas numéricas, las cuales se describen en la Tabla 3.6.

Respuestas numéricas a las métricas recolectadas		
P7	% esfuerzo consumido en revisiones o aprobaciones	100 x Esfuerzo revisiones o aprobaciones M4 / Esfuerzo producto M3
P9	Productividad del Producto	Tamaño M5 / Esfuerzo producto M3
P11	% esfuerzo consumido en trabajo vuelto a efectuar	100 x Esfuerzo Trabajo vuelto a efectuar M9 / Esfuerzo Producto M3

Tabla 3.6. Respuestas numéricas a ciertas preguntas efectuadas

3. Definición operacional de las medidas

Una vez que se han identificado las medidas, se las debe definir; los nombres no son suficientes. El concepto a ser medido debe ser claramente definido. Además debemos poder decir exactamente a otras personas, cómo es obtenida cada medida, a fin de que se pueda coleccionar e interpretar los valores correctamente.

Una forma clara de explicar cómo utilizar las medidas identificadas, es la utilizada por el Departament of Defense and US Army en su método PSM (Practical Software and Systems Measurement) [Jones00]. PSM dispone de plantillas para definir cada una de las mediciones, tanto en sus características en función del proyecto y producto a desarrollar, como en los procedimientos para recopilación y análisis.

En el Anexo A se explica el significado de cada atributo de estas plantillas.

A continuación se definen en forma operacional, las medidas establecidas mediante el método GQM.

Fechas *Milestone*

Área de preocupación: Cronograma y progreso
Categoría: *Milestone performance*

Esta métrica mide las fechas iniciales y finales de una actividad, evento o producto. La medida provee un punto de vista sobre las actividades y eventos planeados.

Guía de Selección

Aplicación Proyecto
- Aplicable a todos los tamaños y tipos de proyectos.
- Incluida en la mayoría de prácticas de medición de la industria.

Proceso de Integración
- Los datos requeridos son obtenidos de la planeación del proyecto-producto y/o la documentación. Los datos deben estar enfocados en las actividades y eventos principales.
- Milestones detallados proveen una mejor indicación del progreso y permiten la identificación temprana de problemas.
- Si las actividades o eventos son replaneados, las fechas originales deben ser retenidas para observar los cambios en los valores planeados.

Usualmente Aplicado Durante

Planeación Proyecto (Estimación)
Análisis de Requerimientos (Estimación y Actual)
Diseño (Estimación y Actual)
Implementación (Estimación y Actual)
Integración y Prueba (Estimación y Actual)
Operación y Mantenimiento (Estimación y Actual)

Guía de Especificación

Ítems Típicos de Datos
- Fecha de inicio de la actividad o evento.
- Fecha final de la actividad o evento.

Atributos Típicos

Actividad o nombre del evento
Versión del plan

Típicamente Recolectado para cada
- Actividad

Cuentas Actuales Basadas en
- Al culminar exitosamente una tarea

Esta medida responde preguntas como:

¿Cuál es la fecha proyectada para terminar el producto?

¿Qué actividades y productos están a tiempo, adelantadas o retrasadas con respecto al cronograma?

¿Cuán a menudo cambia el cronograma?

Estado Solicitud de Cambios

Área de preocupación: Cronograma y progreso
Categor a: Progreso de las unidades de trabajo

Esta medida, cuenta el número total de solicitudes de cambio que afectan un producto. La medida provee una indicación de la cantidad de trabajo vuelto a efectuar que ha sido ejecutado o es requerido. Esta medida sólo identifica el número de cambios; no comunica el impacto funcional de los mismos o la cantidad de esfuerzo requerido para implementarlos.

Guía de Selección

Aplicación Proyecto
- Aplicable a todos los tamaños y tipos de productos.
- Usado a menudo en operaciones y programas de mantenimiento

Proceso de Integración
- Los datos deben ser disponibles de los productos que ponen las solicitudes de cambio bajo control de configuración.
- A menudo usado en los productos donde los requerimientos evolucionan a lo largo del ciclo de vida.

Usualmente Aplicado Durante

Análisis de Requerimientos (Actual)
Diseño (Actual)
Implementación (Actual)
Integración y Prueba (Actual)
Operación y Mantenimiento (Actual)

Guía de Especificación

Ítems Típicos de Datos
- Número de solicitudes de cambio generadas.
- Número de solicitudes de cambio resueltas.

Atributos Típicos

Clasificación cambios (corrección de defectos, mejora)
Presentado/Aprobado

Típicamente Recolectado para cada
- Especificación (sistema, requerimiento, diseño)

Alternativas en la Solicitud de Cambios incluye
- Nuevas características
- Reporte de acciones correctivas

Cuentas Actuales Basadas en
- Aprobación solicitud de cambio
- Solicitud de cambio implementado
- Solicitud de cambio aprobado

Esta medida responde preguntas como:

¿Cuántas solicitudes de cambio han impactado el producto?
¿Están las solicitudes de cambio siendo implementadas a una proporción suficiente para cumplir el cronograma?
¿Esta disminuyendo la tendencia de solicitudes de cambio con la terminación del proyecto?

Esfuerzo	**Área de preocupación:**	**Recursos y Costos**
	Categoría:	**Personal**

La medida del esfuerzo cuenta el número de horas de labor o número de personal aplicado a todas las tareas. Esta es una medida fácil de entender. Esta puede ser categorizada por la actividad, así como por producto. Esta medida usualmente se relaciona con el costo, pero además se puede direccionar a otras áreas incluyendo Planes y Progreso, y Desempeño Proceso.

Guía de Selección

Aplicación Proyecto
- Aplicable a todos los tamaños y tipos de productos.
- Incluida en la mayoría de prácticas de medición de la industria.

Proceso de Integración
- Los datos son usualmente derivados desde un sistema de reportes y/o un sistema separado para manejar el tiempo.
- Todas las horas de labor deben ser coleccionadas.
- Esta medida es más efectiva cuando la cuenta y sistema de reportes esta ligada a productos individuales
- La planeación de datos es usualmente basado en modelos de estimación, datos históricos, o juicio de los ingenieros.

Usualmente Aplicado Durante

Planeación Proyecto (Estimación)
Análisis de Requerimientos (Estimación y Actual)
Diseño (Estimación y Actual)
Implementación (Estimación y Actual)
Integración y Prueba (Estimación y Actual)
Operación y Mantenimiento (Estimación y Actual)

Guía de Especificación

Ítems Típicos de Datos
- Número de horas de labor (horas, días, meses, etc.)
- Número de personal

Atributos Típicos

Categoría de Labor

Típicamente Recolectado para cada
- Componente

Cuentas Actuales Basadas en
- Criterio de Reportes

Esta medida responde preguntas como:

¿Están los recursos de desarrollo siendo aplicados según el plan?

¿Están las tareas o actividades, tomando más o menos esfuerzo que el esperado?

Tamaño

Área de preocupación: Crecimiento y Estabilidad
Categoría: Tamaño del producto y estabilidad

El Tamaño puede ser medido en líneas de código (LOC), puntos de función, número de páginas, etc., dependiendo de las características del producto. El tamaño es una medida de software bien entendida que ayuda a la estimación del costo del proyecto, el esfuerzo requerido, cronograma, y productividad. Cambios en el tamaño pueden originar riesgos en el desarrollo, y posible trabajo adicional.

Guía de Selección

Aplicación Proyecto
- Usado en productos de cualquier dimensión. Menos importante en productos donde el código es generado utilizando herramientas de generación de código y ambientes de programación visual.
- Incluida en la mayoría de prácticas de medición de la industria.

Proceso de Integración
- Los datos requeridos son obtenidos a través de la utilización de herramientas de medición, u observación personal.
- Usualmente requiere un entrenamiento formal (Ej. Puntos de Función).
- Una metodología consistente debe ser *utilizada para efectuar esta medición.*

Usualmente Aplicado Durante

Planeación Proyecto (Estimación)
Análisis de Requerimientos (Estimación)
Diseño (Estimación)
Implementación (Estimación y Actual)
Integración y Prueba (Actual)
Operación y Mantenimiento (Actual)

Guía de Especificación

Ítems Típicos de Datos
- Número de líneas de código (LOC), número de puntos de función, número de paginas, etc.

Atributos Típicos

Lenguaje
Origen (nuevas, reusadas)
Uso (externo, parte principal del producto)

Típicamente Recolectado para cada
- Unidad o equivalente

Definición puede incluir
- Para las LOC:
  - Líneas Lógicas
  - *Líneas Físicas*
  - Comentarios, etc.

Cuentas Actuales Basadas en
- Pasando las pruebas unitarias
- Pasando las inspecciones

Esta medida responde preguntas como:

¿Cuán grande es el producto de software?

¿Cuánto ha cambiado el tamaño del producto?

Defectos

Área de preocupación: Calidad del producto
Categoría: Corrección Funcional

Esta medida proporciona la cantidad, y estado de los defectos reportados. El número de defectos indica la cantidad de trabajo vuelto a efectuar, y tiene un impacto directo en la calidad. Las tasas de apertura de defectos pueden indicar la madurez del producto (una disminución debe ocurrir al completar una prueba). Las tasas de cierre son una indicación de progreso, y puede ser usados para predecir la terminación de una prueba. Dar seguimiento de la duración de tiempo que los defectos permanecen abiertos, puede ser usado para determinar el progreso al corregir los defectos, o el trabajo que esta siendo aplazado. La medida Densidad de Defectos - una expresión del número de defectos en una cantidad del producto - pueda ser derivado de esta medida. La densidad de defectos puede identificar componentes con la concentración más alta de defectos.

Guía de Selección

Aplicación Proyecto
- Aplicable a todos los tamaños y tipos de productos.
- Incluida en la mayoría de prácticas de medición en la industria.

Proceso de Integración
- Requiere un proceso bien definido de pruebas e inspecciones y un proceso disciplinado de seguimiento de defectos.
- Esta medida esta generalmente disponible durante la integración y pruebas. Es beneficioso iniciar el seguimiento de defectos lo mas temprano.
- Fácil de recolectar datos actuales cuando se usa un sistema de seguimiento de defectos automatizado.
- La densidad de defectos requiere la recolección del numero de defectos y el tamaño de cada componente.
- Los defectos de software son a menudo defectos lógicos.

Usualmente Aplicado Durante

Análisis de Requerimientos (Estimación y Actual)
Diseño (Estimación y Actual)
Implementación (Estimación y Actual)
Integración y Prueba (Estimación y Actual)
Operación y Mantenimiento (Actual)

Guía de Especificación

Ítems Típicos de Datos
- Número de defectos
- Promedio de duración de los defectos

Atributos Típicos

Categoría de Defectos (requerimientos, diseño, producto, documentación)
Estado Defecto (abierto, cerrado)

Típicamente Recolectado para cada
- CI o equivalente

Cuentas Actuales Basadas en
- Defectos aceptados por el control de configuración
- Corrección de defectos exitosamente probados/inspeccionados

Esta medida responde preguntas como:

¿Los defectos reportados y la tasa de cierre permitirán cumplir las fechas de integración y pruebas programadas?

¿Qué porcentaje de esfuerzo es consumido en revisiones y aprobaciones?

Productividad

Área de preocupación: Desempeño del Proceso
Categoría: Eficiencia del Proceso

La medida de productividad compara la cantidad del producto completado con la cantidad de esfuerzo requerido. Esta medida es una entrada básica para planificar proyecto y evaluar si los niveles de rendimiento son suficientes para conseguir las estimaciones de costo y calendario.

Guía de Selección

Aplicación Proyecto
- Aplicable a todos los tamaños y tipos de productos.
- Aplicable a todos los dominios.

Proceso de Integración
- Para comparar la productividad de diferentes productos, las mismas definiciones de tamaño de producto y esfuerzo deben ser usados.
- El ambiente, lenguaje, herramientas, complejidad y experiencia del personal, pueden afectar la productividad.
- Para componentes de software, el tamaño del producto puede ser medido como líneas de código, número de componentes, o puntos de función.

Usualmente Aplicado Durante

Planeación Proyecto (Estimación)
Análisis de Requerimientos (Estimación y Actual)
Diseño (Estimación y Actual)
Implementación (Estimación y Actual)
Integración y Prueba (Estimación y Actual)
Operación y Mantenimiento (Estimación y Actual)

Guía de Especificación

Ítems Típicos de Datos
- Tamaño Producto (ej. Líneas de Código)
- Número de horas de labor

Atributos Típicos

Lenguaje (para software)

Típicamente Recolectado para cada
- Proveedor

Cuentas Actuales Basadas en
- Luego de completar exitosamente las pruebas

Esta medida responde preguntas como:

¿Se está produciendo a una tasa suficiente para alcanzar la fecha de terminación?

¿Cuan eficiente es el proceso?

Tiempo

Área de preocupación: Desempeño del Proceso
Categoría: Eficiencia del Proceso

Esta medida registra la longitud de tiempo que toma un proceso para completar todas las actividades asociadas. La acumulación de todos los procesos determina el tiempo total para completar un proyecto. Normalmente, un objetivo clave en la mejora de procesos es reducir el tiempo de desarrollo completo.

Guía de Selección

Aplicación Proyecto
- Aplicable a todos los tamaños y tipos de productos.

Proceso de Integración
- Se debe definir claramente los criterios de entrada y salida para cada proceso.

Usualmente Aplicado Durante
Planeación Proyecto (Estimación)
Análisis de Requerimientos (Estimación y Actual)
Diseño (Estimación y Actual)
Implementación (Estimación y Actual)
Integración y Prueba (Estimación y Actual)
Operación y Mantenimiento (Estimación y Actual)

Guía de Especificación

Ítems Típicos de Datos
- Tiempo de inicio del proceso
- Tiempo de culminación del proceso

Atributos Típicos
Identificador del proceso

Típicamente Recolectado para cada
- Proceso

Cuentas Actuales Basadas en
- Al iniciar y terminar un proceso

Esta medida responde preguntas como:

¿Cuánto tiempo tomará completar esta actividad?

Retrabajo

Área de preocupación: Desempeño del Proceso
Categoría: Efectividad del Proceso

Esta medida sigue la pista de la cantidad de esfuerzo de trabajo consumido en la corrección de defectos. Además, identifica la calidad del esfuerzo inicial del producto, los productos que necesitan retrabajo, y procesos que necesitan mejorar.

Guía de Selección

Aplicación Proyecto
- Aplicable a todos los tamaños y tipos de productos.

Proceso de Integración
- El esfuerzo del retrabajo sólo debe incluir el esfuerzo asociado para corregir defectos. El esfuerzo consumido para incorporar realces no es retrabajo.
- El costo del retrabajo y estimaciones del cronograma deben ser incluidas en los planes y comparados con los valores reales.
- Para los componentes de software, se debe enfocar en las línea del código afectadas por volver a efectuar un trabajo (añadidas, modificadas, borradas).

Usualmente Aplicado Durante
- Planeación (Estimación)
  Análisis de Requerimientos (Estimación y Actual)
  Diseño (Estimación y Actual)
  Implementación (Estimación y Actual)
  Integración y Prueba (Estimación y Actual)
  Operación y Mantenimiento (Estimación y Actual)

Guía de Especificación

Ítems Típicos de Datos
- Horas de labor
- Número de componentes afectados por retrabajo

Atributos Típicos
Categoría de labor

Típicamente Recolectado para cada
- Actividad

Cuentas Actuales Basadas en
- Al concluir la corrección de defectos

Esta medida responde preguntas como:

¿Cuánto esfuerzo es consumido en la corrección de defectos del producto?

¿Qué porcentaje de esfuerzo es consumido en trabajo vuelto a efectuar?

Resumen:

En este capítulo se describieron en forma operacional (mediante las plantillas que propone PSM), las medidas que se deben recolectar a fin de: *administrar* el esfuerzo del software, *incrementar* la calidad de los productos de software, *reducir* el costo de desarrollo, *reducir* la duración del ciclo de desarrollo, *mejorar* la productividad, e*liminar* los defectos antes de las pruebas, *minimizar* el costo de las pruebas de software; todos estos, aspectos a tener en cuenta dentro de cualquier organización.

Como base fundamental de estos objetivos se utilizó el modelo de proceso de software Personal Software Process, pues es uno de los modelos mas detallados en cuanto a como usar las mediciones como herramienta para evaluar, estimar y controlar el proceso. Para identificar las medidas que permitan alcanzar los objetivos trazados se utilizó el método Goal Question Metrics (GQM).

Capítulo

Modelo Cualitativo multinivel de calidad y control de cambios

En este capítulo, se presenta un modelo genérico multinivel de calidad y control de cambios [DeJesús01]. Se inicia identificando los roles que están presentes en el proceso de producción del software. Luego se presentan los diferentes grados de madurez en el *proceso de producción de software* relacionados con el control de calidad de los productos y el control de cambios, con estos modelos se obtienen diversas combinaciones llegando al final a presentar un Modelo Genérico para el Control de Calidad y Cambios en los Productos de Software que incluya todos los roles presentes en el proceso de producción.

4.1 Introducción

Desde que el producto de software es definido hasta que este queda fuera de servicio se producen cambios que resultan normalmente en una versión diferente del producto. Usualmente un cambio en un producto de software requiere cambios en otros productos relacionados. Por ejemplo un cambio en los requerimientos significa cambios en la especificación, diseño, código, y pruebas. Estos cambios que pudieran darse deberán ser evaluados y negociados con los grupos afectados para que ajusten sus planes de software, productos de trabajo y actividades.

Así, los grupos afectados deben llevar a cabo una adecuada administración de requisitos porque son la base para estimar, planear, realizar y darle seguimiento a las actividades del proyecto de software durante todo su ciclo de vida.

Un producto de software puede requerir diferentes controles de calidad y solicitudes de cambio, ya sea porque se necesite de una evaluación independiente de quien lo produce, que podría ser un revisor, quien verificaría el cumplimiento de la especificación de los requisitos. Otros productos podrían necesitar un doble control de calidad, en donde además

de un revisor, un aprobador evaluaría el producto para la minimización del impacto en otros productos, debido a su modificación.

Además, para mejorar la calidad de los productos, podrían realizarse revisiones tipo inspección y pruebas tipo "testing", pero siempre garantizando la coordinación de las actividades de comunicación y colaboración entre todos los miembros de los grupos de trabajo, para evitar conflictos y/o realizar tareas repetidas.

A continuación se presenta un modelo genérico para el control de calidad y cambios a los productos de software con tres niveles configurables de control que den la suficiente flexibilidad para ser utilizados de acuerdo a las necesidades de control de un producto y/o recursos de una organización en particular.

4.2. Roles en los grupos de producción de software.

En la literatura de la Ingeniería del Software [CMM®] podemos encontrar descripciones de diversos roles que intervienen en el proceso de producción de software.

Para fines de este modelo los roles que nos interesan son:

- Solicitante
- Productor
- Revisor
- Aprobador
- Bibliotecario

Estos roles están involucrados en diferentes etapas del "ciclo de vida" de un producto de software que abarca su creación a causa de una solicitud, su constante mejora mediante sus revisiones, su aprobación, su uso y hasta que se retire.

Estos roles pueden ser desempeñados por una o más personas, así será posible tener a un grupo de Productores, a un grupo de Revisores o a un grupo de Aprobadores, etc.

A continuación se describen brevemente las responsabilidades de cada rol con respecto a un producto de trabajo de software.

Solicitante

- El solicitante es el que realiza una solicitud de creación de un producto de software o la solicitud de cambio al mismo. Un solicitante puede ser ya sea el cliente o cualquier miembro del grupo del proyecto.

Productor

- El *Productor* será el responsable de producir un producto de software.
- Es responsable de corregir defectos y/o introducir cambios.

Revisor

- El *Revisor* verifica y/o valida el producto para asegurar la calidad de su contenido. En caso de encontrar defectos regresa el producto al productor con los defectos documentados. Las revisiones pueden realizarse con técnicas tipo inspecciones, revisiones entre colegas, etc., o pueden referirse a pruebas automatizadas tipo "testing".
- En caso de no encontrar defectos acepta el producto.

Aprobador

- El Aprobador determina si el producto cumple con los criterios de entrada a la biblioteca de las líneas base.
- En caso de encontrar defectos regresa el producto al productor con los defectos documentados.
- Si el Aprobador determina que el producto es aprobado éste podrá ingresar a la biblioteca de líneas base.

Bibliotecario

- El Bibliotecario organiza las diferentes bibliotecas del proyecto para cada línea base, guarda los componentes y los proporciona cuando se les hace un cambio y en su caso actualiza las bibliotecas.

4.3. Proceso iterativo genérico de producción de Software

La calidad de los productos de software y el control de cambios a éstos son elementos fundamentales que reflejan la madurez de un proceso de producción.

En la práctica de la industria del software y en los modelos de referencia tipo CMM® existen diferentes niveles de tratarlos.

A continuación se presentarán dos modelos con diferente nivel de control sobre:

- Control de producción y calidad del software
- Control de cambios a productos de software

Con estos modelos se pueden dar diversas combinaciones entre los diferentes niveles de control de producción y calidad con control de cambios, llegando al final a proponer un Modelo Genérico para el Control de Calidad y Cambios en los Productos de Software incluyendo a todos los roles descritos en la sección anterior.

Los niveles de procesos de producción de software que se muestran a continuación representan una forma gradual de cómo los grupos de producción de software pueden controlar la producción, la calidad y los cambios:

- **El Control de Producción** tiene que ver con la identificación de diferentes atributos del producto, tales como: el nombre, quien es el productor, la fecha de producción, su versión, la relación con otros productos y control de acceso del mismo que puede estar en un estado editable o no editable.

- **El Control de Calidad** está relacionado a las actividades de revisión y aprobación del producto con el fin de cumplir con los requisitos del solicitante y evitar los cambios no autorizados. Esta actividad se enfoca a básicamente en las actividades de encontrar y

eliminar defectos. Tiene que ver con los atributos del producto como su estado, sus defectos y control de acceso.

- **El Control de Cambios** a los requisitos del producto de software se controlan mediante solicitudes, donde se detalla el porqué y quién desea el cambio, lo que permitirá apoyar el análisis de impacto y su seguimiento así como determinar si el cambio fue solicitado, aceptado, rechazado, implantado o verificado.

4.3.1. Niveles de control de producción y calidad del software

En las prácticas de la industria del software podemos encontrar comúnmente tres niveles de control de producción y calidad de los productos de software.

- El nivel 1: Control personal de producción y calidad es el que ofrece menos control y deja toda la responsabilidad en manos del Productor.
- El nivel 2: Control de producción con revisión de calidad, incluye revisiones que ayudan a asegurar mayor calidad del producto.
- El nivel 3: Control de producción con doble revisión de calidad, incluye las prácticas que apoyan la administración de configuración de productos de software adecuada [CMM®].

La tabla 4.1. muestra el grado de participación de los roles en cada uno de los niveles:

Roles	**Nivel 1**	**Nivel 2**	**Nivel 3**
Productor	√	√	√
Revisor		√	√
Aprobador			√
Bibliotecario			√

Tabla 4.1 Participación de los roles en los niveles de producción y calidad del software

4.3.2. Niveles de control de cambios a productos de software

Los requisitos de un producto de software pueden evolucionar y cambiar en cualquier etapa del proyecto.

Los cambios a los requisitos obligan a realizar cambios a los productos de software. Estos cambios deberán ser evaluados y negociados con los grupos afectados con el fin de predecir los impactos al desarrollo del proyecto. Esta evaluación permite mantener la integridad de los productos de software y también un entendimiento común entre el solicitante y el grupo del proyecto de software. Podemos identificar los siguientes niveles:

- Nivel 1: Control verbal de cambios a productos de software, la solicitud se realiza en forma verbal, y en muchos casos improvisada, sin que medie una solicitud de cambio formal para ello, por lo tanto el productor no siempre se compromete con realizar el cambio

- Nivel 2: Control documentado de cambios a productos de software, aquí se introduce la formalidad mediante una solicitud documentada del cambio a un producto de software que ya cumplió con un ciclo de desarrollo. El objetivo es "no dejar nada en el aire" y que posteriormente implique diversas confusiones provocando que el producto no funcione adecuadamente o que impacte a otros productos.

- Nivel 3: Control de cambios evaluados a productos de software, se efectúa un análisis de impacto del cambio solicitado al sistema, teniendo un fuerte control del producto sobre todo considerando que el producto ya ha sido liberado y entregado al cliente o usuario final.

La tabla 4.2 muestra el grado de participación de los roles en cada uno de los niveles de control de cambios:

Roles	Nivel 1	Nivel 2	Nivel 3
Productor	√	√	√
Revisor			√
Aprobador			√
Bibliotecario			√

Tabla 4.2 Participación de los roles en cada uno de los niveles de control de cambios

4.3.3. Integración de control de calidad y cambios

Se expone un análisis del modelo de control de producción y calidad de software, con sus diferentes niveles, mediante su combinación con los niveles de control de cambios.

1. **Control personal de producción y calidad (nivel 1) y control verbal de cambios a productos de software (nivel 1).**

Este es el caso más común donde el control de la calidad depende del Productor y no existe control de cambios sobre el producto de software (Figura 4.1) todo se reduce a la informalidad en donde los posibles defectos que el Productor encuentre al producto no serían documentados, como tampoco los cambios que le sean solicitados.

Aunque a simple vista pudiera parecer no muy recomendable, si lo es en casos donde una persona es dueña de todo y ese "todo" es de tamaño "controlable" y las solicitudes de cambio y los productos son simples.

Característica general

- La calidad del producto depende totalmente del juicio del Productor
- La solicitud del cambio se realiza mediante una comunicación verbal

Roles involucrados

- El *Solicitante es* quien solicita un producto (informalmente) y expresa verbalmente sus necesidades de cambio del producto de software.
- El *Productor* es el responsable de producir un producto de software, realizar los cambios e introducirlo a un repositorio compartido cuando termine.

La Calidad depende de:

- La habilidad del productor para:
  - Comprender correctamente la solicitud del producto y la solicitud del cambio.
  - Cumplir con la solicitud durante el desarrollo del producto.
  - Del buen juicio para la finalización del producto.
- Ausencia de modificaciones controladas del producto terminado.

El control de los cambios depende de:

- La buena comprensión de la solicitud.
- Una buena memoria del productor.

No hay control de modificaciones a causa de defectos.

Recomendaciones:

- Este nivel de control es recomendado para solicitudes y productos simples y de bajo riesgo.
- En cuanto al control de cambios, es recomendado para pequeños cambios que no son importantes para el contenido general del producto.

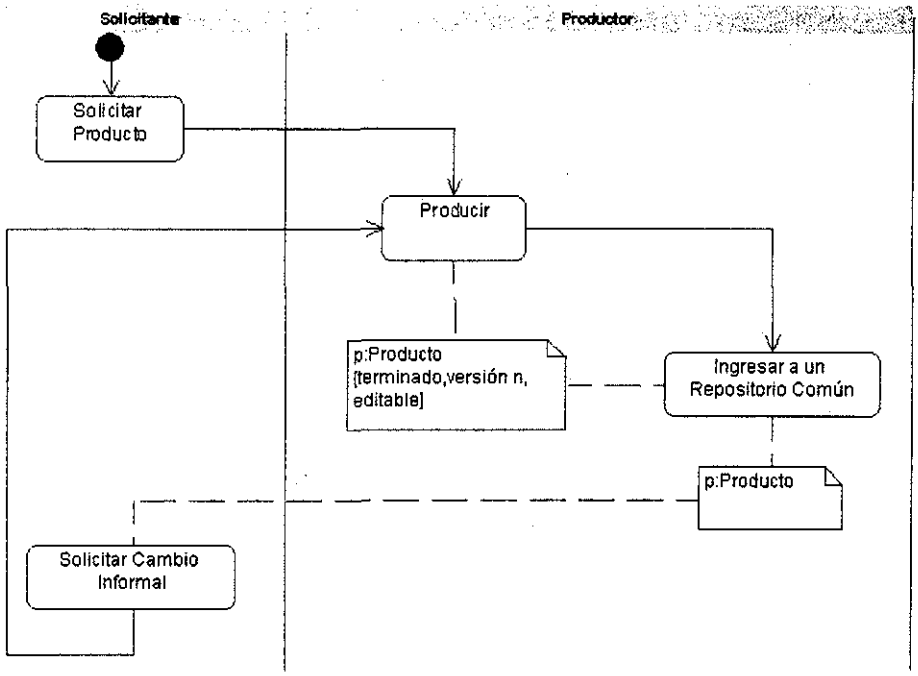


Figura 4.1 Control personal de producción y calidad (nivel 1) y control verbal de cambios a productos de software (nivel 1).

2. Control de producción con revisión de calidad (nivel 2) y control verbal de cambios a productos de software (nivel 1).

En el caso del control de producción y calidad nivel 2 con nivel 1 de control de cambios, (Figura 4.2), se incluye el rol del Revisor, que proporciona un control formal de calidad mucho mejor del que pudiera ofrecer el Productor.

Debido a la naturaleza informal de las solicitudes de cambio el producto terminado podría no corresponder con las expectativas del solicitante a pesar que existe un Revisor. Al Productor le solicitan directamente los cambios y el papel del Revisor podría quedar relegado o confuso ya que no sabría cuales son las expectativas del solicitante, "no sabría contra que revisar".

Por otro lado, dadas las características del rol del Productor, el análisis del impacto del producto sobre otros estaría limitado, salvo en caso de proyectos pequeños.

El control de cambios sobre el producto no existe, lo que puede dificultar la consistencia de versiones del producto.

Este esquema aunque factible, estaría limitado.

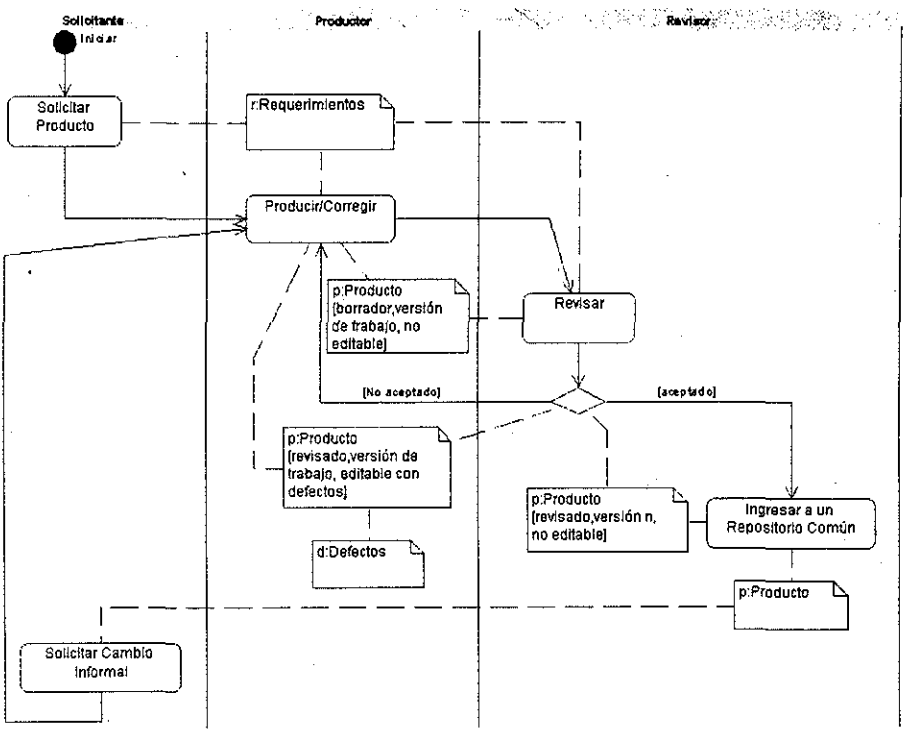


Figura 4.2 Control de producción con revisión de calidad (nivel 2) y control verbal de cambios a productos de software (nivel 1).

3. Control de producción con doble control de calidad (nivel 3) y control verbal de cambios a productos de software (nivel 1).

En este esquema sería muy riesgoso que un solicitante realice una solicitud de cambio informal, si en el modelo de producción y calidad existen los roles de revisor, aprobador y bibliotecario que controlan la producción del software.

4. Control personal de producción y calidad (nivel 1) y control documentado de cambios a productos de software (nivel 2).

Similar a 1, solo que existe una solicitud formal del cambio, (Figura 4.3).

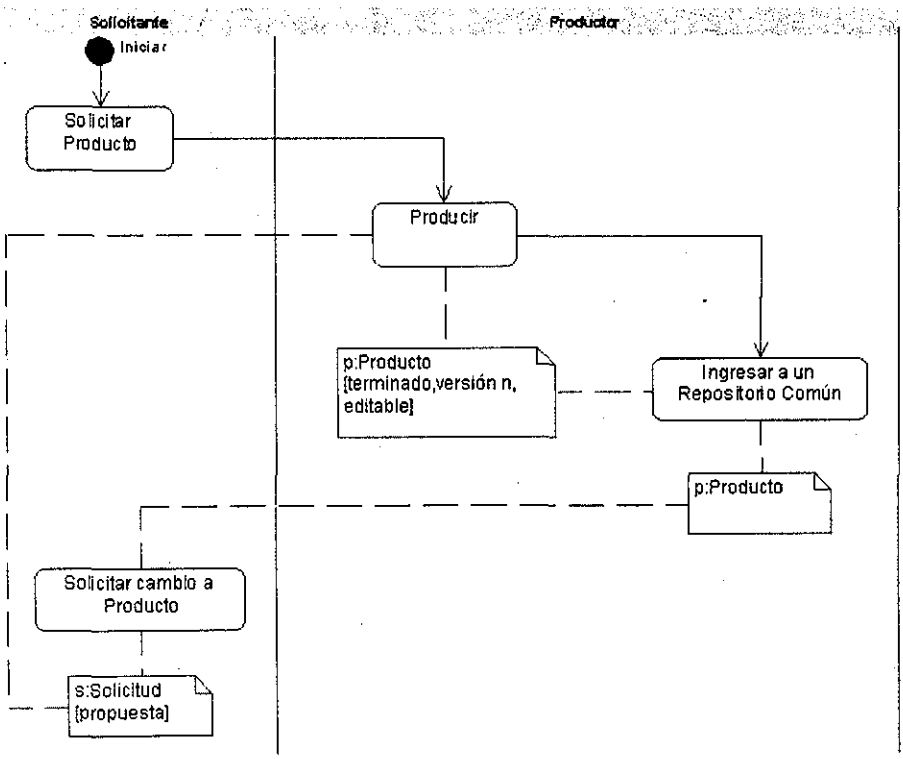


Figura 4.3 Control personal de producción y calidad (nivel 1) y control documentado de cambios a productos de software (nivel 2).

5. Control de producción con revisión de calidad (nivel 2) y control documentado de cambios a productos de software (nivel 2).

Similar a 2, solo que existe una solicitud formal del cambio, (Figura 4.4).

Característica General:

- La calidad del producto depende de una revisión independiente al productor, el cual utiliza una especificación documentada de los requisitos.
- La solicitud del cambio es documentada.

Roles involucrados:

- El *Solicitante* quien solicita el producto por medio de la especificación de requisitos documentados y expresa sus necesidades de cambio al producto de una forma documentada.
- El *Productor* es el responsable de producir un producto de software de acuerdo con los requisitos especificados y documentados. También, evalúa el impacto y el costo de la solicitud de cambio y decide si es o no aprobada. Corregirá los defectos encontrados por el Revisor.
- *Revisor* quien revisa utilizando técnicas de validación/verificación. Documenta los defectos y acepta el producto correcto.

La calidad depende de:

- La especificación documentada de los requisitos y su comprensión por parte del Productor y del Revisor(es).
- Las técnicas de revisión (verificación/validación).
- La claridad de la documentación de los defectos para el productor.

El control del cambio depende de:

- La buena comprensión de la solicitud documentada.

Recomendaciones:

- Este nivel de control es recomendado para toda clase de productos, especialmente para los entregados al cliente.
- En cuanto a los cambios, es recomendado cuando no consumen tiempo y no impactan a otros productos.

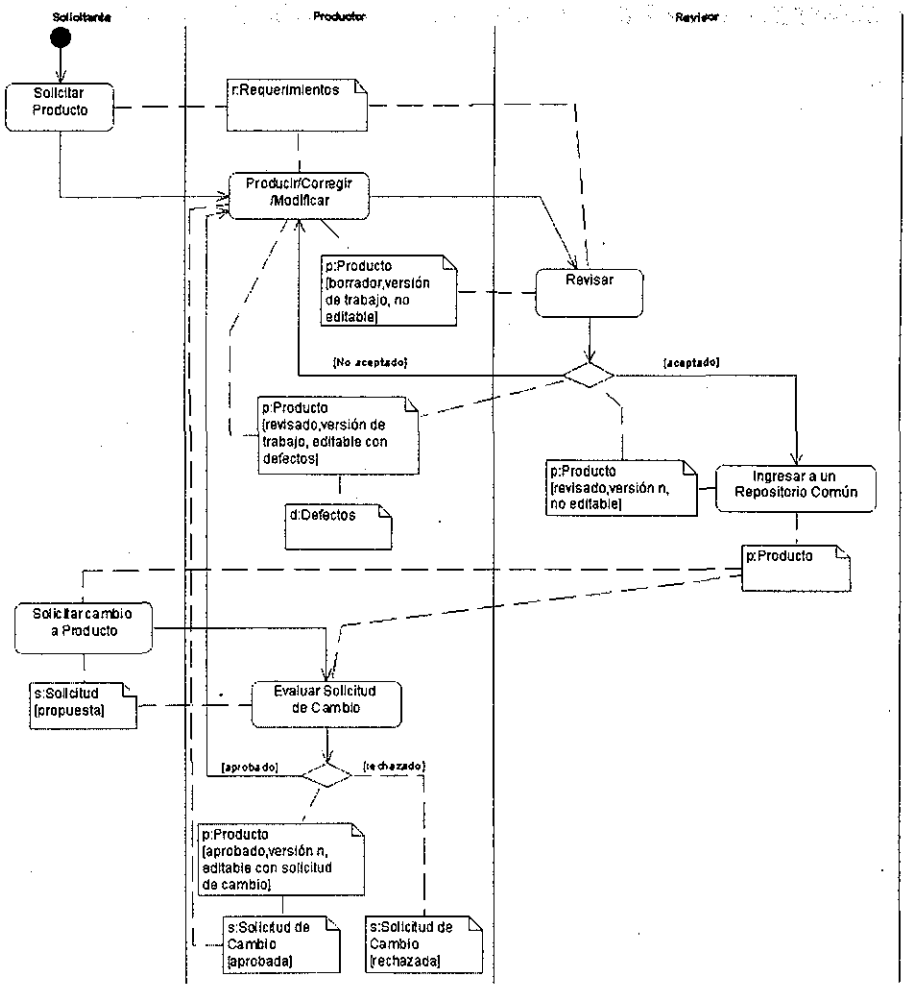


Figura 4.4 Control de producción con revisión de calidad (nivel 2) y control documentado de cambios a productos de software (nivel 2).

6. Control de producción con doble control de calidad (nivel 3) y control documentado de cambios a productos de software (nivel 2).

Este esquema es riesgoso, porque un solicitante realiza la solicitud de cambio directamente al productor, ignorando el rol del aprobador.

7. Control personal de producción y calidad (nivel 1) con control de cambios evaluados a productos de software (nivel 3).

Este esquema no tiene sentido porque mientras que en el nivel 1 de control de producción y calidad se tiene un repositorio compartido, el nivel 3 de control de cambios exige una biblioteca para la línea base.

8. Control de producción con revisión de calidad (nivel 2) y control de cambios evaluados a productos de software (nivel 3).

Este esquema no tiene sentido porque mientras que en el nivel 2 de control de producción y calidad se tiene un repositorio compartido, el nivel 3 de control de cambios exige una biblioteca para la línea base, aunque exista una solicitud documentada del cambio.

9. Control de producción con doble control de calidad (nivel 3) y control de cambios evaluados a productos de software (nivel 3).

La figura 4.5, que incluye un modelo ideal de control de producción y calidad de software con control de cambios.

Característica general:

- La calidad del producto depende no solo del cumplimiento de la especificación de los requisitos sino también de la minimización del impacto en otros productos de la línea base.
- Son evaluados el impacto y el costo de la solicitud de cambio.

Roles Involucrados:

- El *Solicitante,* quien solicita el producto por medio de la especificación de requisitos documentados y expresa sus necesidades del cambio al producto en una forma documentada.
- El *Productor,* quien produce un producto de software de acuerdo con los requisitos especificados y documentados. Corrige los defectos encontrados por el Revisor y el Aprobador.
- *Revisor,* quien revisa el producto utilizando técnicas de validación/verificación. Documenta los defectos y acepta el producto corregido.
- *Aprobador,* quien analiza el producto revisado para asegurarse que puede introducirse a las línea base sin impactar inesperadamente a otros productos. También, evalúa el impacto y el costo de la solicitud de cambio y decide si es o no aprobada.
- El *Bibliotecario,* quien administra la biblioteca de los productos en la línea base.

La calidad del producto depende de:

- Lo mismo que en el nivel 2 de control de producción con revisión de calidad.
- Listas de verificación de elementos para evaluar antes la inserción del producto a la biblioteca de la línea base. Estos elementos deberán expresar la dependencia de puntos de riesgo del producto con respecto a otros productos del proyecto que será revisado.

El control del cambio depende de:

- La buena comprensión de la solicitud documentada.
- La evaluación del impacto sobre el producto y otros productos y la estimación de los costos del cambio.

Recomendaciones:

- Este esquema es recomendado para productos de gran importancia, con tiempo de vida largo y necesidades de mantenimiento y en la que las solicitudes de cambio pueden consumir tiempo/costo y tener impacto sobre otras líneas base.

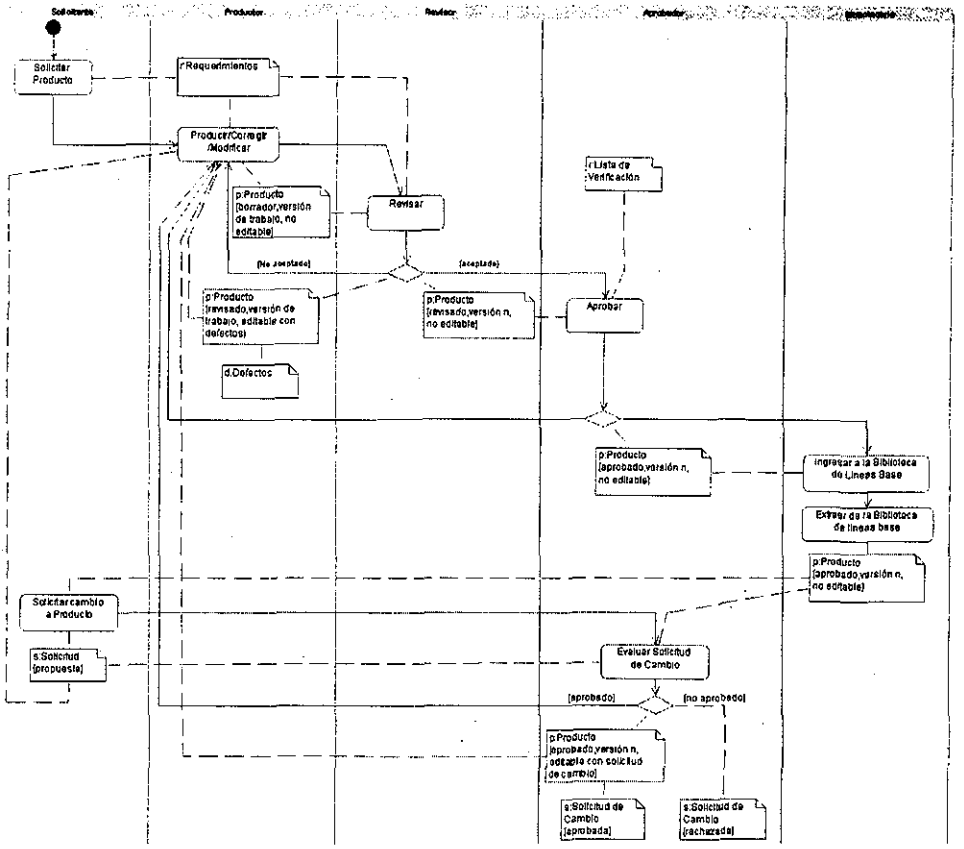


Figura 4.5 Control de producción con doble control de calidad (nivel 3) y control de cambios evaluados a productos de software (nivel 3).

4.3.4 Resumen de las posibles combinaciones de los diferentes niveles de control.

La tabla 4.3 resume las posibles combinaciones de los diferentes niveles de control de calidad y cambios a los productos de software:

√: indica el esquema cuya combinación es factible de implementar, determinado por la cantidad de recursos disponibles o por las necesidades propias de los proyectos y/o de la empresa.

X: indica el esquema muy riesgoso para implementar.

Control de cambios	**Control de producción y calidad**		
	Nivel 1 Personal	***Nivel 2 Con revisión de calidad***	***Nivel 3 Con doble control de calidad***
Nivel 1, Verbal	√	√	X
Nivel 2, Documentado	√	√	X
Nivel 3, Evaluado	X	X	√

Tabla 4.3 Resumen de las posibles combinaciones de los diferentes niveles de control de calidad y cambios a los productos de software

4.3.5 Modelo Genérico de Control de Calidad y Cambios a los Productos de Software.

Del análisis realizado al integrar los modelos, podemos concluir que las opciones más factibles para ser utilizadas en la práctica son:

1. Control personal de producción y calidad (nivel 1) y control verbal de cambios a productos de software (nivel 1).
2. Control de producción con revisión de calidad (nivel 2) y control documentado de cambios a productos de software (nivel 2).
3. Control de producción con doble control de calidad (nivel 3) y control de cambios evaluados a productos de software (nivel 3).

Las características de estos niveles se detallaron en la sección anterior

Estas tres combinaciones lo llamaremos Modelo Genérico de tres niveles de Control de Calidad y Cambios en los Productos de Software.

- El primer nivel corresponde al caso clásico del *heroísmo*, donde todo depende de la visión limitada de una sola persona.
- El segundo nivel refleja el reconocimiento del *valor de colaborar en equipo y de la documentación escrita.*
- El tercer nivel puede considerarse para *el control ideal* de proyectos a largo plazo y de mayor complejidad.

El aspecto genérico del modelo está cubierto por múltiples significados de los elementos asociados al modelo.

La tabla 4.4 ejemplifica las posibles substituciones por cada rol, actividad y objeto.

4.3.6 Aplicaciones del Modelo Genérico de Control de Calidad y Cambios a los Productos de Software.

El modelo genérico puede ser aplicado al menos para los tres siguientes propósitos:

1. *Para definir el estándar de los procesos de control de calidad y cambios para una organización o proyecto.*

   La organización o proyecto puede clasificar los productos de trabajo de software en tres categorías: simple, término medio o a largo plazo, basado en las recomendaciones de cada nivel de control.

   Por cada categoría, una instancia del proceso de control de calidad y cambios substituye los elementos genéricos por uno en particular, utilizando la guía de la tabla 4.4.

Cada producto de trabajo de software puede ser asignado al mismo nivel de control durante todo su ciclo de vida.

Otra posibilidad es la de cambiar dinámicamente el nivel de control de acuerdo a la madurez del producto.

2. *Para estimar el costo de producción y cambios , el cual incluye el costo del control de calidad.*

   Cada nivel de control del modelo, especifica las actividades necesarias que deben ser llevadas a cabo.

   Si podemos estimar para un producto su tamaño específico, el tiempo dedicado a cada actividad (producción, corrección, revisión, aprobación), y si podemos estimar el número de correcciones y cambios solicitados para este producto, entonces podemos estimar el costo total de la producción que incluye el control de la calidad y las modificaciones.

3. *Para crear herramientas que administren diferentes niveles de control de calidad y cambios.*

   El modelo puede utilizarse para diseñar herramientas de producción de software que pueden administrar, en el mismo ambiente, diferentes niveles de producción, calidad y control de cambios.

   Las herramientas pueden permitir también cambios dinámicos en el nivel de control durante el ciclo de vida del producto.

La tabla 4.4 ejemplifica posibles substituciones de conceptos genéricos del modelo de control de calidad y cambios a los productos de software.

Conceptos genéricos		Posibles substituciones
Roles	Solicitante	Cliente, Usuario Final, Administrador, Ingeniero de Software
	Productor	Analista, Diseñador, Administrador, Ingeniero de Software
	Revisor	Cliente, Usuario Final, Par de Ingeniero de Software, Probador, Administrador
	Aprobador	Cliente, Administrador, Equipo de Aseguramiento de la Calidad, Ingeniero de Software, Experto, Mesa de Control de Configuración del Software
	Bibliotecario	Ingeniero de Software, Administración de la Configuración del Software
Actividades	Solicitar un Producto	Mediante: Solicitud verbal, Solicitud documentada informalmente, Solicitud documentada formalmente
	Producir/Corregir	Para producir: Un plan, Especificación de Requisitos, Documentación en Análisis y Diseño, Planes de Prueba, Código, Manuales de Usuario
	Revisar	Revisión por el Administrador, Revisión con Cliente/Usuario Final, Revisión entre Pares, Inspecciones, Pruebas de Integración, Prueba del Sistema, Prueba de aceptación
	Aprobar	Revisión por el Administrador, Revisión por el Equipo de Aseguramiento de la Calidad, Revisión y Aprobación por parte de la Mesa de Control de Configuración del Software
	Insertar/Extraer de un repositorio común	Uso de directorio de archivos, Uso de herramientas simples de Administración de la Configuración, Uso de bibliotecas de líneas base.
Objetos	Producto	Plan, Especificación de Requisitos, Modelo de Análisis, Modelo de Diseño, Código, Casos de Prueba
	Requisitos	Requisitos Cliente/Usuario Final, Requisitos del Producto
	Defectos	Bitácora de defectos, bitácora de Pruebas
	Solicitud de Cambio	Solicitud de cambio verbal, Solicitud de cambio documentada informalmente, Solicitud de cambio documentada formalmente.
	Lista de Verificación	Lista implícita o explícita de elementos para revisar antes de aprobar.

Tabla 4.4 Ejemplos de posibles substituciones de conceptos genéricos del modelo de control de calidad y cambios a los productos de software.

El modelo genérico permite:

- La integración de la producción, la verificación corrección y el control de cambios de los productos de software en un solo proceso y que frecuentemente son referidos como prácticas o áreas separadas.
- El control de calidad y el control de los cambios en cada nivel puede verse como un patrón y especializarse de acuerdo a las necesidades o recursos particulares de la organización o proyecto
- Tres niveles de clasificación de las prácticas de calidad y control de cambios. El primer nivel es personal y aparentemente tiene un bajo costo en la producción pero probablemente un alto riesgo en la calidad. El segundo nivel tiene un mayor costo de producción pero la responsabilidad de la calidad no es más que de una persona. En el tercer nivel, los costos de producción parecen ser los más altos, pero el riesgo de la calidad puede ser la más baja. En la práctica, el tiempo que se pierde en las revisiones y aprobaciones puede ser recuperado para realizar las correcciones y especificaciones.

Resumen:

El modelo presentado en este capítulo es parte de un trabajo previo [DeJesús01], el cual hemos denominado *Modelo Cualitativo Multinivel de Calidad y Control de Cambios*. Se inició identificando los roles que están presentes en el proceso de producción del software. Luego se presentaron los diferentes grados de madurez en el *proceso de producción de software* relacionados con el control de calidad de los productos y el control de cambios, con estos modelos se obtuvieron diversas combinaciones llegando al final a presentar un modelo genérico que permita controlar la calidad y cambios en los productos de software que incluya todos los roles presentes en el proceso de producción. Finalmente se expusieron las diferentes aplicaciones del modelo descrito

Capítulo

Integración de los Modelos Cuantitativo y Cualitativo

En este capítulo se integra el modelo que permite medir el proceso de software (*modelo cuantitativo*) al modelo genérico multinivel de control de calidad y cambios (*modelo cualitativo*). Se indican las consideraciones que se tomaron en cuenta para la integración y se efectúa la misma.

5.1 Introducción

Uno de los objetivos fundamentales de un proceso de software maduro es el de producir productos de calidad que cumplan los requisitos del usuario. La satisfacción del cliente ha llegado a ser el lema de muchas organizaciones que intentan sobrevivir y prosperar en un mundo en que la competitividad aumenta día con día.

Durante el proceso de producción, el software llega a cambiar constantemente, debido a necesidades para repararlo (eliminar errores que no fueron detectados antes de liberar la aplicación), a necesidades de soporte por la evolución de la aplicación, por nuevos requisitos o a causa de cambios de los requisitos iniciales. Por lo anterior podemos decir que los miembros del grupo de ingeniería de software y otros grupos relacionados deben realizar sus actividades de administración de requisitos con el fin de que sean documentados y controlados, debido a que son la base para estimar, planear, realizar y dar seguimiento a las actividades del proyecto de software durante todo su ciclo de vida.

Además se debe contar con procedimientos que controlen los cambios para establecer y mantener la integridad de los productos.

Estos procesos de control, unidos a un conjunto de actividades necesarias para asegurar y garantizar que una organización obtenga un determinado nivel de calidad en el producto o

que cumpla con los requisitos establecidos, forman parte del *modelo genérico multinivel* detallado en el Capitulo IV.

Lamentablemente la utilización individual de este modelo no permite evaluar y mejorar el proceso de desarrollo del software. Dentro de una organización de software usualmente a mas de obtener un producto de calidad, se desea al menos estimar y controlar los planes y costos de desarrollo [Humphrey95]. La medición juega un rol primordial en el proceso de mejoramiento. Sin ésta, es difícil valorar la eficiencia del proceso y descubrir áreas que necesitan ser trabajadas.

A pesar de las ventajas que se obtienen al incluir las métricas en el proceso de producción de un producto de software, la situación real en la industria de software revela un uso limitado de la medición de procesos. El modelo del proceso de medición propuesto en el Capitulo II, proveerá la información necesaria para identificar y controlar muchas de las *áreas de preocupación* a los que se enfrenta un administrador de proyecto. Esto facilita la identificación de riesgos, el seguimiento efectivo de problemas específicos, la evaluación del impacto de los riesgos y problemas en el cumplimiento de los objetivos en lo que se refiere a costos, cronogramas, productividad y calidad del producto final.

A continuación se exponen los lineamientos que permiten integrar los modelos descritos dentro de un solo ambiente de trabajo, de tal manera que la integración y la interpretación de los resultados que proyecta el uso de éstos pueda provocar un cambio cultural en los desarrolladores de software, una mejora en las organizaciones que apliquen la integración de los modelos; contribuyendo al alcance de una disciplina de ingeniería madura.

5.2 Consideraciones

Debido a diferencias entre los modelos, es necesario tomar en cuenta algunas consideraciones a fin de poder efectuar la integración. El *modelo del proceso de medición (modelo cuantitativo)* requiere ser integrado dentro de un proceso de medición que involucre todo el ciclo de vida de un producto de software, a fin de que pueda ser usado

consistentemente por todas las personas encargadas de tomar decisiones a través de la organización.

Efectivamente, las fases implementación y evaluación del método de medición descritas en el Capitulo II (puntos 2.3.2 y 2.3.3), necesitan ser integradas al ciclo de vida de un producto de software, para lo cual se requiere:

- Asignar roles y responsabilidades.
- Definir los procedimientos de recolección; y
- Evaluar los mecanismos de evolución (ejemplo: revisiones del progreso mensual).

Por otra parte el *modelo genérico multinivel para controlar la calidad y cambios (modelo cualitativo)*, tiene implícito un proceso de ingeniería del producto de software bien definido que integra todas las actividades de ingeniería de software para producir en forma efectiva y eficiente productos de software correctos y consistentes.

Las tareas de la ingeniería del producto de software incluyen analizar los requisitos del software, desarrollar los requisitos del software, desarrollar la arquitectura del software, diseñar el software, implementar el software en el código, integrar los componentes de software y probar el software para verificar que satisface los requisitos especificados.

La documentación necesaria para realizar las tareas de ingeniería de software (documentos de requisitos de software, documentos de diseño de software, plan de pruebas y procedimientos de prueba) son desarrollados y revisados para asegurar que cada tarea obtiene los resultados de las tareas anteriores y los resultados producidos sean los correctos para las tareas subsecuentes (incluyendo las tareas de operar y mantener el software).

Cuando los cambios son aprobados, los productos de trabajo de software, los planes, los compromisos, los procesos y las actividades son revisados para reflejar que los cambios hayan sido efectuados. Todo este proceso de producción del software tiene definido una serie de roles, entre los que destacamos:

- Solicitante
- Productor
- Revisor
- Aprobador
- Bibliotecario

Estos roles están involucrados en diferentes etapas del "ciclo de vida" de un producto de software que abarca su creación a causa de una solicitud, su constante mejora mediante sus revisiones, su aprobación, su uso y hasta que se retire.

De las observaciones expuestas, se pretende aprovechar las tareas de la ingeniería del producto de software expuestas en el *modelo cualitativo* para anexar las métricas diseñadas mediante el *modelo cuantitativo*. Esta integración dará como resultado inmediato que los roles involucrados durante la producción del software, adopten nuevas responsabilidades a fin de responder a las inquietudes presentes en las *fases implementación y evaluación del método de medición*, como son:

- ¿Quién y cuándo se recolectará los datos de las métricas?
- ¿Quién(es) será el encargado de la explotación de los resultados?
- ¿Quién(es) auditará las métricas recolectadas?
- ¿Quién(es) evaluará las medidas y el proceso de medición?

5.3 Integración entre los Modelos Cualitativo y Cuantitativo

Del *modelo genérico multinivel para controlar la calidad y cambios (modelo cualitativo)*, se pueden identificar la factibilidad de utilización en la práctica de tres niveles de control de calidad y cambios en los productos de software (nivel personal, de equipo y de negocio). Valiéndonos de las consideraciones expuestas en el punto anterior integraremos las métricas identificadas en el *modelo de proceso de medición (modelo cuantitativo)* y asignaremos nuevas responsabilidades a los roles de cada nivel.

Nivel 1. Control personal de producción y calidad y control verbal de cambios a productos de software con manejo de métricas que permiten evaluar el proceso de producción. (***Nivel Personal***)

Este es el caso más común donde el control de la calidad depende del Productor y no existe control de cambios sobre el producto de software (Figura 5.1) todo se reduce a la informalidad en donde los posibles defectos que el Productor encuentre al producto no serían documentados, como tampoco los cambios que le sean solicitados.

Aunque a simple vista pudiera parecer no muy recomendable, si lo es en casos donde una persona es dueña de todo y ese "todo" es de tamaño "controlable" y las solicitudes de cambio y los productos son simples.

Característica general

- La calidad del producto depende totalmente del juicio del Productor
- La solicitud del cambio se realiza mediante una comunicación verbal
- Las mediciones tal como: tamaño, tiempo de desarrollo y milestones, dependen de la disciplina del personal.

Roles involucrados

- *El Solicitante* es quien solicita un producto (informalmente) y expresa verbalmente sus necesidades de cambio del producto de software.
- *El Productor* es quien desarrolla el producto de software, realiza los cambios e introduce el producto terminado a un repositorio compartido. Además es el responsable de registrar el tiempo dedicado a desarrollo, las fechas milestone, y el tamaño del producto. Puede en cualquier momento determinar la productividad y el cumplimiento del cronograma de trabajo.

La calidad depende de:

- La habilidad del productor para:
  - Comprender correctamente la solicitud del producto y la solicitud del cambio.

- Cumplir con la solicitud durante el desarrollo del producto.
- Del buen juicio para la finalización del producto.

- Ausencia de modificaciones controladas del producto terminado

El control de los cambios depende de:

- La buena comprensión de la solicitud.
- Una buena memoria del productor.

Nota: No hay control de modificaciones a causa de defectos.

Las mediciones dependen de:

- La disciplina del productor para registrar las mediciones y de la exactitud para medir el tamaño del producto, tiempo de desarrollo y fechas milestones.

Indicadores

- Estabilidad Producto – es posible evaluar la estabilidad del tamaño para cada versión.
- Esfuerzo acumulado – es posible analizar el esfuerzo de desarrollo acumulado para cada versión del producto.
- Productividad – es posible conocer la productividad (tamaño/esfuerzo) para cada versión.
- Cronograma – es posible conocer el estado del producto durante el desarrollo, y con respecto a las fechas planificadas en el cronograma.

Recomendaciones:

- Solicitudes y productos simples y de bajo riesgo.
- En cuanto al control de cambios, es recomendado para pequeños cambios que no son importantes para el contenido general del producto.
- Debido a que este nivel es utilizado para productos de software pequeños y de poca importancia, no deben servir como base de estimación para productos futuros en cuanto a tiempo de desarrollo, tamaño, cronograma o costos de producción.

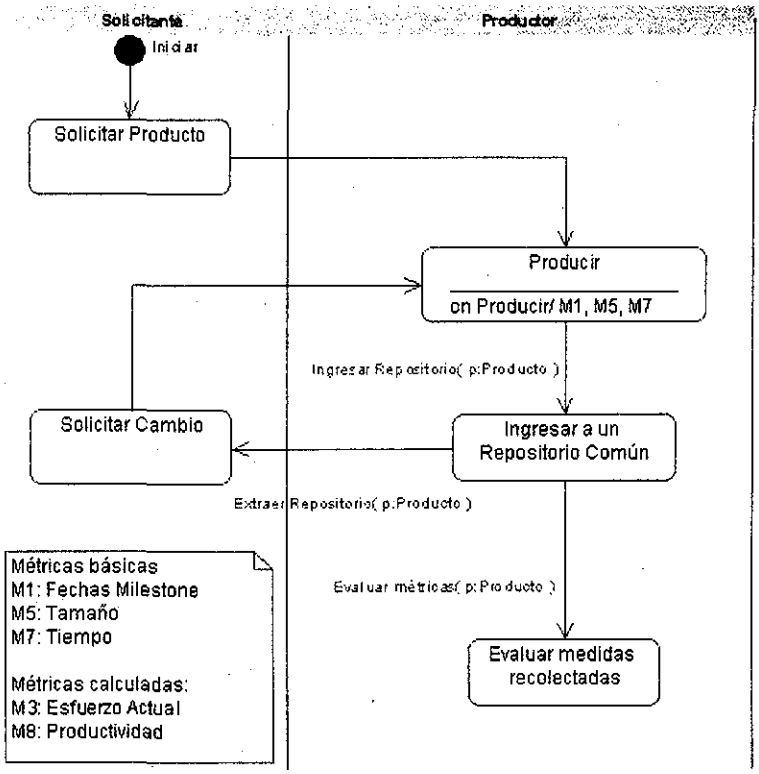


Figura 5.1. Diagrama de actividades UML para el Nivel 1: Nivel Personal

Nivel 2. Control de equipo de producción con revisión de calidad y control documentado de cambios a productos de software mas manejo de métricas que permiten evaluar la calidad del producto y el proceso de producción. ***(Nivel de Equipo)***

En éste nivel se cuenta con unos requisitos documentados para el desarrollo del producto. Los requisitos pueden ser cualquier producto de trabajo, tales como: requisitos del cliente, documentos de análisis, diseño, el código, etc., que se usan para construir el producto en cuestión, y con respecto a ellos dicho producto tiene que estar correcto y consistente.

Se introduce el rol del Revisor cuya razón podría originarse desde la incipiente necesidad del Productor cuando acepta al decir: "terminé mi producto pero necesito que alguien me lo revise" o hasta una inspección formal.

Es de gran utilidad revisar si el producto cumple con los requisitos del solicitante por alguien diferente al que produjo el software.

Las revisiones pueden abarcar diferentes técnicas de verificación y validación desde las revisiones entre colegas, las inspecciones formales o las pruebas unitarias, de integración, de sistema o de aceptación. Un punto importante es el registro de defectos encontrados para facilitar el trabajo de corrección.

Característica general:

- La calidad del producto depende de una revisión independiente al productor, el cual utiliza una especificación documentada de los requisitos.
- La solicitud del cambio es documentada.
- Las mediciones no dependen exclusivamente de la disciplina del productor, sino además del personal del equipo de producción.

Roles involucrados:

- El *Solicitante* quien solicita el producto por medio de la especificación de requisitos documentados y expresa sus necesidades de cambio al producto de una forma documentada. Puede además evaluar la calidad del producto en cualquier etapa de producción, así como el cumplimiento del cronograma de trabajo.
- El *Productor* es quien desarrolla el producto de software de acuerdo con los requisitos especificados y documentados. También, recibe y evalúa el impacto de una solicitud de cambio y decide si es o no aprobada. Corregirá los defectos encontrados por el Revisor. Es responsable de registrar el tiempo dedicado a desarrollo, corrección de defectos o modificación a causa de solicitudes de cambio; las fechas milestone, y el tamaño del producto. Además puede determinar la productividad, calidad del producto, esfuerzo

acumulado en desarrollo, corrección o modificación y el cumplimiento del cronograma de trabajo.

- *Revisor* quien revisa utilizando técnicas de validación/verificación. Documenta los defectos y acepta el producto correcto. Es responsable de registrar el tiempo dedicado a revisión y registrar los defectos localizados si fuese el caso. Además puede determinar el trabajo vuelto a efectuar a fin de valorar la efectividad de las revisiones y pruebas.

La calidad depende de:

- La especificación documentada de los requisitos y su comprensión por parte del Productor y del Revisor(es).
- Las técnicas de revisión (verificación/validación).
- La claridad de la documentación de los defectos para el productor.

El control del cambio depende de:

- La buena comprensión de la solicitud documentada.

Las mediciones dependen de:

- De la disciplina del productor para registrar las medidas y de la exactitud para medir el tamaño del producto, tiempo de desarrollo, fechas milestone, tiempo dedicado a corrección de defectos y tiempo de modificación (a causa de una solicitud de cambio). Además, las medidas dependen de la disciplina del revisor para registrar el tiempo dedicado a revisión y los defectos encontrados.

Indicadores:

Los indicadores para el nivel 2, incluyen los indicadores del nivel 1, así como:

- Esfuerzo acumulado – es posible evaluar el esfuerzo consumido en desarrollo, corrección de defectos, modificación y revisión.
- Calidad del producto – es posible establecer la calidad del producto de acuerdo al número de defectos encontrados y corregidos. Esto permite planear nuevas pruebas de calidad y revisión.

- Esfuerzo de producción – es posible conocer el porcentaje de tiempo invertido en desarrollo, corrección de defectos, revisión y modificación a causa de una solicitud de cambio.
- Estabilidad del producto – es posible evaluar la estabilidad del producto en términos del número de solicitudes de cambio.

Recomendaciones:

- Este nivel de control es recomendado para toda clase de productos, especialmente para los entregados al cliente.
- En cuanto a los cambios, es recomendado cuando no consumen tiempo y no impactan a otros productos.
- La recolección apropiada de las métricas especificadas permitirá establecer una base de datos histórica que ayude a entender la capacidad del proceso de software actual.
- Los desarrolladores pueden usar las descripciones del producto para entender la calidad de su trabajo e identificar las debilidades durante el proceso.

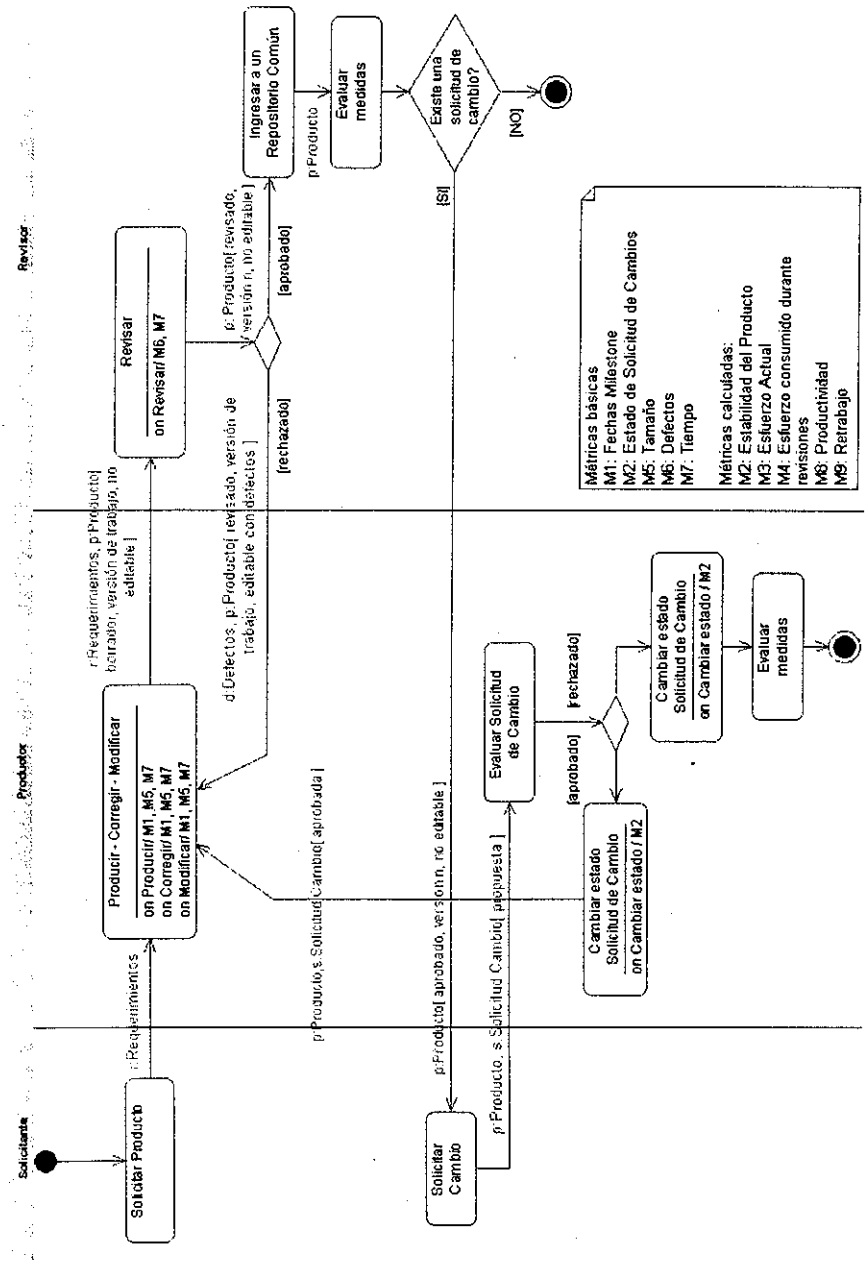


Figura 5.2. Diagrama de actividades UML para el Nivel 2: Nivel de Equipo

Nivel 3. Control de negocio de producción con doble control de calidad y control documentado de cambios a productos de software mas manejo de métricas que permiten evaluar la calidad del producto y el proceso de producción. ***(Nivel de Negocio)***

Bajo este nivel se introducen tres roles más, el del Aprobador, Bibliotecario y Administrador de Calidad, por obvias razones esto implica un mayor grado de especialización y de división de tareas que podría considerarse hasta cierto punto burocrático ya sea porque no se reconoce su utilidad o no se cuentan con los recursos necesarios. En este nivel, un Aprobador revisa las características deseadas del producto, apoyándose en una lista de verificación y un Bibliotecario mantiene un esquema de administración que permite un almacenamiento y recuperación ágil de los diferentes productos de software. El Administrador de calidad es el encargado de explotar los resultados de la medición, auditar y evaluar las medidas y el proceso de medición

Característica general:

- La calidad del producto depende no solo del cumplimiento de la especificación de los requisitos sino también de la minimización del impacto en otros productos de la línea base.
- Son evaluados el impacto y el costo de la solicitud de cambio.
- Las mediciones no dependen solamente de la disciplina del equipo (como en el nivel 2) sino además de la disciplina individual del personal de la organización.

Roles Involucrados:

Los roles de Solicitante, Productor y Revisor descritos en el nivel 2, son complementados con los siguientes roles:

- *Aprobador,* quien analiza el producto revisado para asegurarse que puede introducirse a las línea base sin impactar inesperadamente a otros productos. También, evalúa el impacto y el costo de la solicitud de cambio y decide si es o no aprobada. Es responsable de registrar el tiempo dedicado a revisión y registrar los defectos

localizados si fuese el caso. Determina el porcentaje de trabajo vuelto a efectuar a fin de valorar la efectividad de las aprobaciones.

- *Bibliotecario,* quien administra la biblioteca de los productos en la línea base.
- *Administrador de Calidad,* es el responsable de evaluar las medidas e indicadores con el fin de conocer si satisfacen las necesidades de información de los administradores del proyecto. Evalúa el proceso de medición y toma las medidas adecuadas para mejorar la calidad del producto.

La calidad del producto depende de:

- Lo mismo que en el nivel 2 de control de producción con revisión de calidad.
- Listas de verificación de elementos para evaluar antes la inserción del producto a la biblioteca de la línea base. Estos elementos deberán expresar la dependencia de puntos de riesgo del producto con respecto a otros productos del proyecto que será revisado.

El control del cambio depende de:

- La buena comprensión de la solicitud documentada.
- La evaluación del impacto sobre el producto y otros productos y la estimación de los costos del cambio.

Las mediciones dependen de:

- De las dependencias descritas en el nivel 2, así como la disciplina del aprobador para registrar la lista de defectos y el tiempo invertido en aprobación.

Recomendaciones:

- Este esquema es recomendado para productos de gran importancia, con tiempo de vida largo y necesidades de mantenimiento y en la que las solicitudes de cambio pueden consumir tiempo/costo y tener impacto sobre otras líneas base.
- Los administradores pueden entender el nivel de calidad de sus productos, así como la calidad que los procesos existentes son capaces de alcanzar.
- Efectuar estimaciones reales y planear el tamaño, esfuerzo, y tiempo para proyectos futuros, basados en los datos históricos.

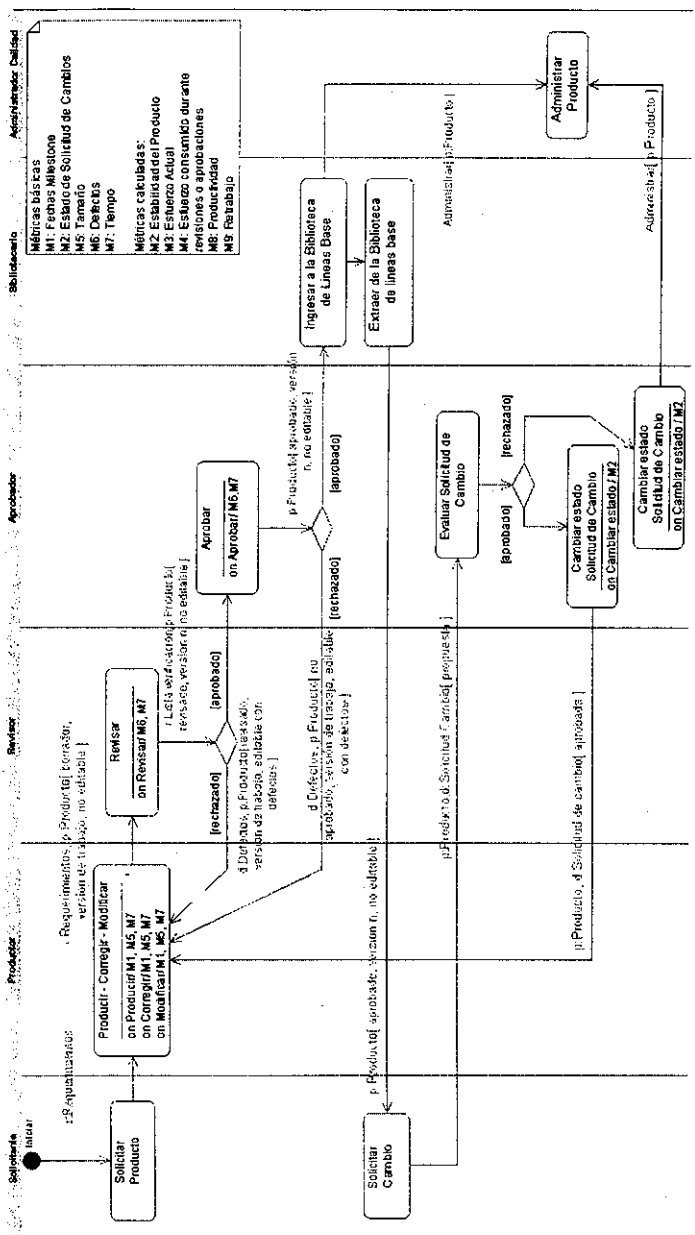


Figura 5.3. Diagrama de actividades UML para el Nivel 3: Nivel de Negocio

Resumen:

En este capítulo se integraron en un solo ambiente de trabajo los modelos descritos en los capítulos II y IV. Debido a diferencias entre los modelos se propusieron ciertas consideraciones a fin de efectuar la integración.

Entre los objetivos que se pretenden alcanzar con el modelo integrado es que la interpretación de los resultados que proyecta el uso de éste pueda provocar un cambio cultural en los desarrolladores de software, una mejora en las organizaciones que apliquen el modelo; contribuyendo al alcance de una disciplina de ingeniería madura.

Además el modelo integrado servirá como base para la construcción de una herramienta que permita controlar el proceso de un producto de software.

Capítulo

Marco Contextual

A continuación se describen los elementos tecnológicos que se tomaron en cuenta para desarrollar una herramienta que permita controlar el proceso de un producto de software. A lo largo de los capítulos anteriores se ha descrito el marco teórico correspondiente a este trabajo. El capítulo inicia presentando la tecnología Java Server Pages (JSP) como medio para implementar la aplicación. Luego se propone el medio para integrar la base de datos dentro de la Web. Enseguida se detallan las características de la herramienta que permite obtener dinámicamente graficas en líneas, barras o sectores a fin de presentar los resultados del registro de métricas. Finalmente se presenta la opción seleccionada para efectuar el intercambio de archivos entre repositorios (local y centralizado).

6.1 Sistemas de Información basados en la Red

Hoy en día, el segmento más activo en el mercado de las nuevas tecnologías es la aplicación de la Tecnología WEB de INTERNET al diseño de Sistemas de Información. La aplicación de la Tecnología WEB está permitiendo que muchas empresas transformen -con un mínimo impacto- su organización del trabajo, incrementando su productividad de forma tangible. La WEB está aportando a la empresa un factor competitivo básico, al optimizar procesos de circulación y distribución de información asociados al trabajo en grupo.

6.2 Tecnología JavaServer Pages (JSP)

Usualmente al construir aplicaciones WEB, estamos acostumbrados a la idea de escribir un programa que genere la pagina entera (las partes estáticas y dinámicas) usando el mismo programa. Una solución en la cual podemos separar las dos partes, es mediante la tecnología JavaServer Pages. Justamente, una de las grandes ventajas que ofrecen los JSP con respecto a otras tecnologías WEB como CGI y Servlets, es que permite separar la

presentación final de la lógica de negocio. La tecnología JSP, es un gran "Rapid Application Development" (RAD) de aplicaciones Web.

A continuación se explican los conceptos y las ventajas de la tecnología JSP que llevaron a escogerla como medio de implementación.

La Web Dinámica

El Web se ha desarrollado desde un sistema de información distribuido hypermedia basado en red que ofrecía información estática hasta un mercado para vender y comprar mercancías y servicios. Las aplicaciones cada vez más sofisticadas para permitir este mercado requieren una tecnología para presentar la información dinámica.

Las soluciones de primera generación incluyeron CGI, que es un mecanismo para ejecutar programas externos en un servidor web. El problema con los scripts CGI es la escalabilidad; se crea un nuevo proceso para cada petición.

Las soluciones de segunda generación incluyeron vendedores de servidores Web que proporcionaban plug-ins y APIs para sus servidores. El problema es que sus soluciones eran específicas a sus productos. Por ejemplo, Microsoft proporcionó las páginas activas del servidor (ASP) que hicieron más fácil crear el contenido dinámico. Sin embargo, su solución sólo trabajaba con Microsoft IIS o Personal Web Server. Por lo tanto, si deseábamos utilizar ASP teníamos que confiarnos a los productos de Microsoft y no estaríamos gozando de la libertad de seleccionar nuestro servidor web y sistema operativo preferidos.

Otra tecnología de segunda generación que es absolutamente popular en las empresas son los Servlets. Los Servlets hacen más fácil escribir aplicaciones del lado del servidor usando la tecnología Java. El problema con los CGI o los Servlets, sin embargo, es que tenemos que seguir el ciclo de vida de escribir, compilar y desplegar.

Las páginas JSP son una solución de tercera generación que se pueden combinar fácilmente con algunas soluciones de la segunda generación, creando el contenido dinámico, y

haciendo más fácil y más rápido construir las aplicaciones basadas en Web que trabajan con una variedad de otras tecnologías: servidores Web, navegadores Web, servidores de aplicación y otras herramientas de desarrollo.

JavaServer Pages (JSP)

La tecnología JSP es una especificación abierta (y gratis) disponible y desarrollada por Sun Microsystems como un alternativa a Active Server Pages (ASP) de Microsoft, y son un componente dominante de la especificación de Java 2 Enterprise Edition (J2EE). Muchos de los servidores de aplicaciones comercialmente disponibles (como BEA WebLogic, IBM WebSphere, Live JRun, Orion, Tomcat, etcétera) ya utilizan tecnología JSP.

JSP contra ASP

JSP y ASP ofrecen funciones similares. Ambos utilizan etiquetas para permitir código embebido en una página HTML, seguimiento de sesión, y conexión a bases de datos. Algunas de las diferencias triviales son:

- Las páginas ASP están escritas en VBScript y las páginas JSP están escritas en lenguaje Java. Por lo tanto, las páginas JSP son independientes de la plataforma y las páginas ASP no lo son.
- Las páginas JSP usan tecnología JavaBeans como arquitectura de componentes y las páginas ASP usan componentes ActiveX.

Más allá de estas diferencias triviales, hay varias diferencias importantes, que nos ayudo a elegir esta tecnología como medio para desarrollar la aplicación:

- *Velocidad y Escalabilidad*: Aunque las páginas ASP son cacheadas, siempre son interpretadas, las páginas JSP son compiladas en Servlets Java y cargadas en memoria la primera vez que se las llama, y son ejecutadas para todas las llamadas siguientes: Esto le da a las páginas JSP la ventaja de la velocidad y escalabilidad sobre las páginas ASP.

- *Etiquetas Extensibles*: Las páginas JSP tiene una característica avanzada conocida como etiquetas extensibles. Esto mecanismo permite a los desarrolladores crear etiquetas personalizadas. En otras palabras, las etiquetas extensibles nos permiten extender la sintaxis de las etiquetas de las páginas JSP. No podemos hacer esto en ASP.
- *Libertad de Elección*: A menos que instalemos Chili!Soft ASP, las páginas ASP sólo trabajan con Microsoft IIS y Personal Web Server. El uso de páginas ASP requiere un compromiso con los productos de Microsoft, mientras que las páginas JSP no nos imponen ningún servidor web ni sistema operativo. Las páginas JSP se están convirtiendo en un estándar ampliamente soportado.

Tomando en consideración las ventajas que ofrece la tecnología JavaServer Pages, se decidió optar por la misma para desarrollar la aplicación. La Figura 6.1, muestra la arquitectura de esta tecnología. En el Anexo B se detalla el método de instalación para cada componente de la figura.

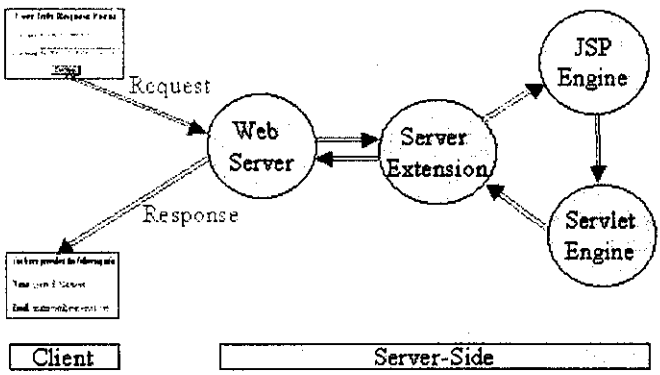


Figura 6.1. Modelo de la arquitectura JSP

6.3 Integración de Bases de Datos en la Web

Como la Web es un medio para localizar, enviar, recibir información de diversos tipos, la integración de las bases de datos en la Web ayudarían a reducir costos y a almacenar información además de aumentar la rapidez de difusión de la misma.

Para la integración de bases de datos con la Web es necesario contar con una interfaz que realice las conexiones, extraiga la información de la base de datos, le dé un formato adecuado de tal manera que puede ser visualizada desde un navegador de la Web (Microsoft-Internet Explorer y Netscape-Navegator) y permita lograr sesiones interactivas entre el usuario y la base de datos.

Un tipo de interfaz con cierto nivel de programación que permite comunicarse con las bases de datos lo constituye el elemento tecnológico Java y su Conectividad de Bases de Datos (JDBC) como se muestra en la Figura 6.2 [Paul99].

JDBC es una especificación de un conjunto de clases y métodos de operación que permiten a cualquier programa Java acceder a sistemas de bases de datos de forma homogénea. Lógicamente, al igual que ODBC, la aplicación de Java debe tener acceso a un driver JDBC adecuado. Este driver es el que implementa la funcionalidad de todas las clases de acceso a datos y proporciona la comunicación entre el API JDBC y la base de datos real [WEB1].

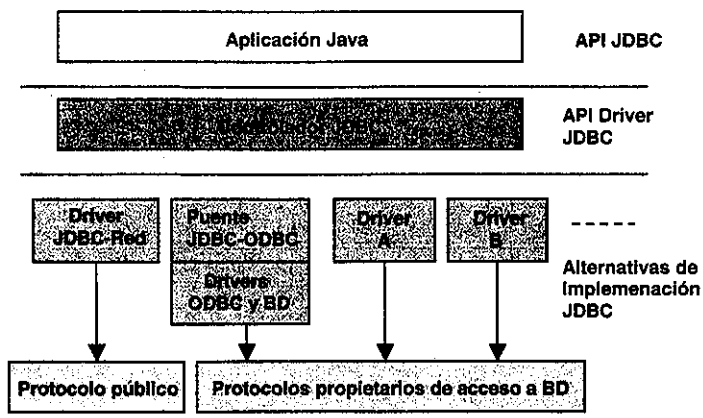


Figura 6.2. Conectividad JDBC.

Para el desarrollo de la herramienta se utilizará el Manejador de Base de Datos Access u Oracle8i y el componente Java (JDBC) ofrecerá la conectividad de bases de datos que administrará los servicios a proporcionar.

6.4 Gráficas dinámicas en la Web

A fin de presentar los resultados del registro de métricas de manera que éstos puedan ser interpretados fácilmente, se hace necesario recurrir a una herramienta que permita obtener en forma dinámica gráficas en líneas, barras o sectores. De las herramientas disponibles en el mercado se eligió EasyCharts [WEB2], pues cumple con los requerimientos de esta aplicación. Algunas de sus características son:

EasyCharts es un conjunto flexible y fácil de usar de componentes gráficos basados en Java que pueden ser usados para visualizar gráficas en páginas web, crear gráficas en aplicaciones java, o generar gráficas como imágenes en un servidor de aplicaciones (ejemplo Tomcat) usando servlets. EasyCharts soporta además gráficas de barras, líneas, sectores o una combinación de cualquiera de estos tipos en una sola gráfica.

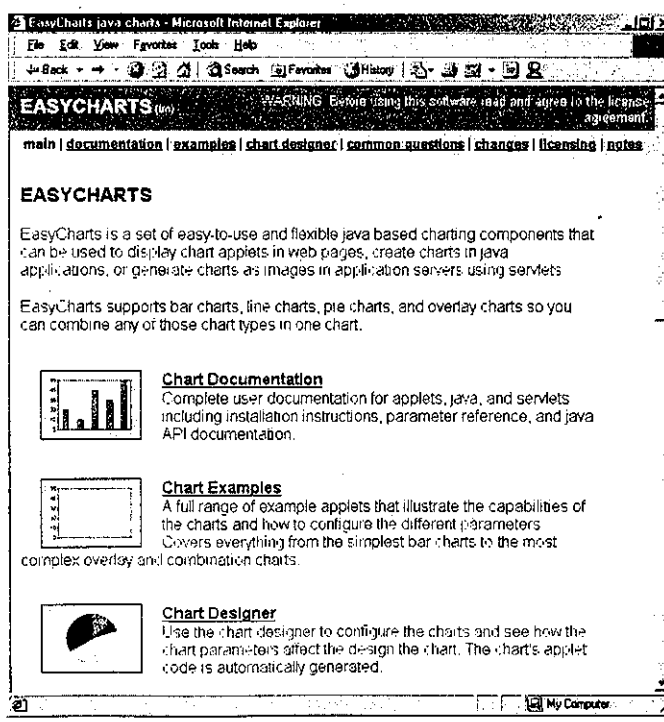


Figura 6.3. EasyCharts -Herramienta para crear gráficas en una pagina Web.

6.5 Intercambio de archivos mediante el protocolo File Transfers Protocol (FTP)

Uno de los requisitos que debe proporcionar la herramienta a desarrollar, es la de permitir transferir archivos desde un repositorio local hacia un repositorio central (compartido), una vez que el producto quede terminado o listo para revisión o aprobación.

La idea de transferir un archivo a un repositorio central, es con el fin de que el producto quede a disposición de otros usuarios; en el caso por ejemplo de que este forme parte de los requerimientos de otro producto de software.

Con este fin se requiere de un servidor FTP que controle el acceso y transferencia de los productos (archivos).

De las herramientas analizadas en el mercado como: Serv-U, WarFTPd y GuildFTPd [WEB3], se escogió la ultima pues posee algunas ventajas que resumimos a continuación:

- Creación de usuarios avanzado
- Los archivos eliminados pueden ser enviados a la Papelera de Reciclaje
- Permite múltiples instancias de ejecución del programa
- Limites de velocidad mínimas y máximas
- Generación aleatoria de contraseñas, con opción de copiar la contraseña en el portapapeles
- Ocultar archivos y carpetas
- Múltiples puertos de conexión
- Control de velocidad de descarga

En el anexo B se detallan los pasos requeridos para instalar esta herramienta. En la figura 6.4 se presenta la pantalla de inicio, una vez instalada la misma.

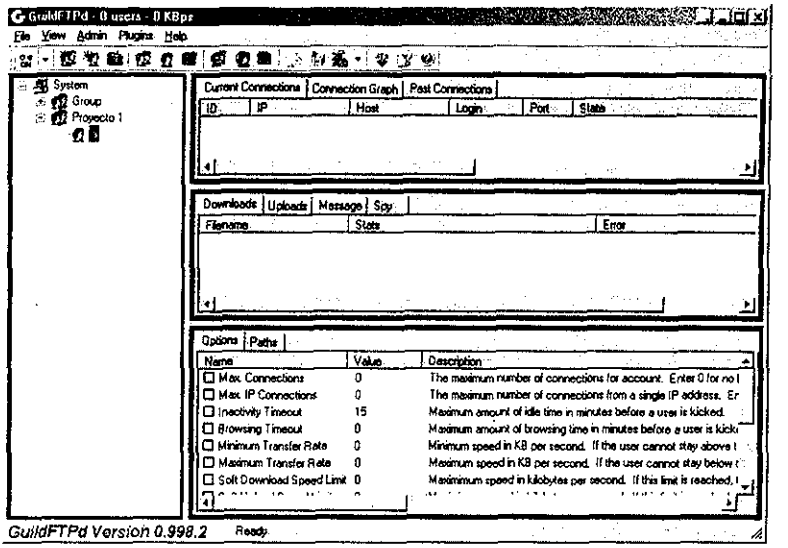


Figura 6.4. GuildFTPd -Herramienta para transferencia de archivos.

Resumen:

En este capítulo se analizó las características de los elementos tecnológicos utilizados para implementar una herramienta que permita controlar el proceso de un producto de software. Además se presentó EasyCharts como el medio seleccionado para obtener dinámicamente gráficas en una página Web.

Finalmente se detallaron las características de GuildFTPd, herramienta que cumple la función de servidor FTP, la cual permitirá efectuar el intercambio de archivos entre un repositorio central y uno local.

Capítulo

Especificaciones del análisis y diseño

En este capítulo se exponen los resultados obtenidos en las fases de análisis y diseño de la herramienta que permite controlar el proceso de un producto de software.

7.1. Consideraciones generales

Con base en lo expuesto en el capítulo V, donde se expone un proceso de producción de software con control de calidad, cambios y métricas, se propone una herramienta con distintos niveles de control, utilizando para el diseño los elementos tecnológicos que se expusieron en el capítulo VI.

UML (Unified Modeling Language) se escogió como base para el modelado de escenarios, diagramas de clase y de estado, principalmente, ya que permite integrar varios modelos que reflejan las mejores prácticas en la industria, además que facilita el proceso de modelado de sistemas de software. La herramienta Rational Rose[2] resultó de gran ayuda para elaborar todos los diagramas.

7.2 Requisitos de la herramienta

La primera actividad que se debe realizar para el desarrollo de un sistema es definir el problema que se quiere resolver. En este caso consiste en crear una herramienta que permita llevar el control del proceso de un producto de software. La herramienta, permitirá configurar el nivel de control de calidad y cambios de los productos de software, recolectar y explotar las medidas básicas apropiadas dependiendo del nivel de control seleccionado (*personal, equipo o negocio*). La tabla 7.1 muestra los niveles de control y su descripción.

[2] Rational Rose es desarrollado por Rational Software Corporation, para mayor información sobre este producto se puede consultar la página http://www.rational.com/

Nivel de Control	Descripción
Nivel Personal (NP)	Control personal de producción y calidad y control verbal de cambios a productos de software con manejo de métricas que permiten evaluar el proceso de producción.
Nivel de Equipo (NE)	Control de producción de equipo con revisión de calidad y control documentado de cambios a productos de software mas manejo de métricas que permiten evaluar la calidad del producto y el proceso de producción.
Nivel de Negocio (NN)	Control de producción de negocio con doble control de calidad y control documentado de cambios a productos de software mas manejo de métricas que permiten evaluar la calidad del producto y el proceso de producción.

Tabla 7.1 Niveles de Control para controlar el proceso de un producto de software

Se utilizará la siguiente nomenclatura para referirnos a cada uno de los niveles de control:

NP. Nivel de Control Personal

NE. Nivel de Control de Equipo

NN. Nivel de Control de Negocio

Además tenemos un nivel de administración(disponible solo para participantes con nivel de administrador).

NA. Nivel de Administración

La herramienta además controlará los proyectos de software, participantes, unidades de medida, productos de software, solicitudes de cambio, los requisitos y los defectos encontrados al producto. Las características con las que cumplen estos elementos son:

Proyectos de Software

Num.	Elemento a controlar	Nivel de Administración
1.	Identificación del proyecto	√
2.	Nombre	√
3.	Descripción	√
4.	Fecha de creación	√
5.	Creador del proyecto	√
6.	Participantes en el proyecto	√

√: Indica si el nivel controla este elemento

Participantes

Num.	Elemento a controlar	Nivel de Administración
1.	Identificación del participante	√
2.	Nombre	√
3.	Login	√
4.	Contraseña	√
5.	Correo electrónico	√
6.	Nivel de acceso (administrador, desarrollador)	√

√: Indica si el nivel controla este elemento

Unidades de medida

Num.	Elemento a controlar	Nivel de Administración
1.	Identificación de la unidad de medida	√
2.	Nombre	√
3.	Descripción	√

√: Indica si el nivel controla este elemento

Productos de software.

Num.	Elemento a controlar	Nivel de control		
		NP	NE	NN
1.	Identificación del producto	√	√	√
2.	Identificación del proyecto	√	√	√
3.	Título	√	√	√
4.	Versión	√	√	√
5.	Nivel de control	√	√	√
6.	Estado del producto (borrador, en revisión o aprobado)	√	√	√
7.	Identificación del responsable de su producción	√	√	√
8.	Fecha de creación de producto	√	√	√
9.	Fecha planeada de terminación del producto	√	√	√
10.	Fecha real de terminación del producto	√	√	√
11.	Descripción general del contenido del producto	√	√	√
12.	Identificación de la unidad de medida	√	√	√
13.	Archivo asociado al producto	√	√	√
14.	Tamaño del producto	√	√	√
15	Tiempo dedicado a:			
	Desarrollo	√	√	√
	Revisión		√	√

	Modificación		√	√
	Corrección de Defectos		√	√
	Aprobación			√
16.	Identificación del responsable de la revisión		√	√
17.	Identificación del responsable de aprobar el producto			√

√: Indica si el nivel controla este elemento

Requisitos de un producto.

Los requisitos serán controlados en el Nivel de Control de Equipo (NE) y de Negocio (NN):

Num.	Elemento a controlar	Nivel de control	
		NE	NN
1.	Identificación del documento de requisitos	√	√
2.	Título del documento de requisitos	√	√
3.	Identificación del producto que debe cumplir el requisito	√	√
4.	Identificación del responsable de la definición del requisito	√	√
5.	Fecha de creación de producto	√	√
6.	Listado de requisitos	√	√

√: Indica si el nivel controla este elemento

Defectos de un producto.

Los defectos, también serán controlados en el Nivel de Control de Equipo (NE) y en el Nivel de Control de Negocio (NN): Se documentan los defectos encontrados a un producto de software, cuando es evaluado durante una revisión o aprobación donde se determina si el producto cumple con los requisitos solicitados.

Num.	Elemento a controlar	Nivel de control	
		NE	NN
1.	Identificación del documento de defectos	√	√
2.	Título del documento de defectos	√	√
3.	Estado del Documento de Defectos (en creación, reportado, corregido)	√	√
4.	Identificación del producto al que se le encontraron los defectos	√	√
5.	Identificación del responsable que encontró los defectos	√	√
6.	Fecha de creación del documento de defectos	√	√
7.	Listado de defectos	√	√

√: Indica si el nivel controla este elemento

Solicitudes de cambios a los productos de software

Las solicitudes de cambio serán controladas en el Nivel de Control de Equipo (NE) y de Negocio (NN):

Num.	Elemento a controlar	Nivel de control	
		NE	NN
1.	Identificación de la solicitud de cambio	√	√
2.	Título de la solicitud de cambio	√	√
3.	Identificación del producto al que se le solicita el cambio	√	√
4.	Estado de la solicitud de cambio (en creación, solicitado, aprobado, rechazado)	√	√
5.	Identificación del responsable de la solicitud de cambio	√	√
6.	Fecha de la solicitud de cambio	√	√
7.	Listado de cambios	√	√

√: Indica si el nivel controla este elemento

7.2.1 Requisitos no funcionales

1. La herramienta podrá implantarse fácilmente, sin depender de la plataforma en la que fue desarrollado.
2. Permitir el acceso a los productos de software en forma distribuida, de manera que se compartan con diferentes miembros del proyecto en un ambiente de red.
3. Desvincular el formato de los productos de un procesador de textos en específico.
4. Minimizar el uso de pantallas y utilizar términos conocidos en español para facilitar el aprendizaje y permitir una rápida aplicación de la herramienta.
5. Contar con un esquema que garantice al personal autorizado tener acceso controlado a los productos de software.

Con el fin de cumplir con estos requisitos, la opción escogida es el lenguaje de programación Java a través de Java Server Pages (JSP), emplear un Servidor de JSP's, y una herramienta para graficar los indicadores del producto.

7.2.2 Requisitos funcionales

En cuanto a las funciones que debe desempeñar la herramienta es necesario considerar principalmente los siguientes requisitos:

1. Niveles Configurables de control de productos y solicitudes de cambio, dependiendo de las necesidades del proyecto.
2. Dependiendo del nivel de control se podrá crear, editar y consultar el contenido de los productos, solicitudes de cambio, requisitos, reporte de defectos del producto, tamaño, y tiempo dedicado a una tarea en particular .
3. Notificar a los respectivos responsables mediante el uso de mensajes, el estado de la producción, revisión o aprobación de los productos de software, solicitudes de cambio y defectos encontrados, sin tener que utilizar otra herramienta para este fin.
4. Cambiar el estado de los productos y solicitudes de cambio.
5. Permitir que los revisores o aprobadores realicen la evaluación de los productos de software y las solicitudes de cambio. Y determinar si los requisitos fueron cumplidos o los defectos corregidos.
6. Generar nuevas versiones de los productos aprobados.
7. Hacer disponible los productos de software cuando se solicita un cambio.
8. Garantizar el acceso a mas más de un usuario a la vez a la lista de productos, solicitudes de cambio, requisitos y defectos.
9. Permitir que cualquier participante en un proyecto tenga acceso a los indicadores actuales del producto.
10. Obtener gráficamente los resultados de las mediciones para cada nivel de configuración del producto.
11. Permitir la transferencia de archivos entre repositorios locales y centralizados.
12. Establecer una base de datos con la información recolectada de las medidas básicas, a fin de ayudar a entender la capacidad del proceso de software actual y ofrecer una base de estimación para proyectos futuros.

7.3 Actores.

Las funciones definidas por los requisitos de la herramienta serán llevadas a cabo por el usuario o grupo de usuarios que interactúan con el mismo y que se denominan "actores". Un usuario puede pertenecer a varios grupos dependiendo de la función que realice. A continuación se describen brevemente las responsabilidades de cada rol con respecto a un producto de trabajo de software y su nivel de control.

- **Administrador,** es el único que puede crear y editar proyectos, participantes (administradores, desarrolladores), y unidades de medida para los productos de software. Es el autorizado para cambiar los roles asignados a un producto de software (desarrollador, revisor o aprobador). En principio es el responsable de todos los productos.
- **Solicitante**, realiza una solicitud de creación de un producto o la solicitud de cambio al mismo. Un solicitante puede ser el cliente o cualquier miembro del grupo del proyecto.
- **Productor**, dependiendo del nivel de control es el que crea y edita los productos o solicitudes de cambio, requisitos y defectos. Envía a evaluación los documentos estableciendo revisores y aprobadores para los productos y los modifica cuando existen solicitudes de cambio. Puede evaluar su nivel de producción y el cumplimiento en el cronograma de trabajo.
- **Revisor**, verifica y/o valida el producto para asegurar la calidad de su contenido y determinar si los requisitos fueron cumplidos. Cambia el estado del producto a revisado o lo rechaza, en cuyo caso regresa el producto al productor con los defectos documentados.

  El revisor ingresa el producto a un repositorio compartido en el Nivel de Control de Equipo o lo envía a un Aprobador para su evaluación en el Nivel de Control de Negocio. Además puede evaluar la efectividad de las revisiones o pruebas en base a las métricas recolectadas.

- **Aprobador,** es el que con la lista de verificación evalúa el producto y cambia el estado del producto a aprobado o lo rechaza, en cuyo caso regresa el producto al productor con los defectos documentados.

  Si el aprobador determina que el producto es aprobado y cumple con los criterios de entrada a la Biblioteca de las línea base, lo envía al Bibliotecario.

  Puede evaluar la efectividad de las aprobaciones en base a las métricas recolectadas.
- **Bibliotecario,** organiza las diferentes bibliotecas del proyecto para cada línea base, guarda los productos y los proporciona cuando se solicita un cambio y en su caso actualiza las bibliotecas cuando se aprueba y se realiza el cambio.
- **Administrador de Calidad,** es el responsable de evaluar las medidas e indicadores con el fin de conocer si satisfacen las necesidades de información de los administradores del proyecto.

  Además evalúa el proceso de medición y toma las medidas adecuadas para mejorar la calidad del producto.

Los actores involucrados en el desarrollo de un producto dependerán de su nivel de control:

Actor	Nivel		
	Personal	**Equipo**	**Negocio**
Administrador Herramienta	√	√	√
Solicitante	√	√	√
Productor	√	√	√
Revisor		√	√
Aprobador			√
Bibliotecario			√
Administrador Calidad			√

Tabla 7.2 Actores participantes en cada nivel de control de producción

Una vez establecidos los requisitos e identificados los actores se puede continuar con las siguientes actividades de la fase de análisis.

7.4 Análisis

Utilizando el modelado orientado a objetos y dadas las características del modelo intregado aplicado dentro de la herramienta se planteó un análisis incremental, iniciando con el nivel de administración de la aplicación, nivel de control personal, nivel de control de equipo y nivel de control de negocio.

Durante el análisis se agruparon las clases siguiendo el modelo de tres capas:

- **Dominio del problema (DP).** Esta formado por el grupo de clases que son significativas a la herramienta.
- **Interfaz De Usuario (IU).** Formado a partir de las clases del dominio del problema y son con las que el usuario interactúa con la herramienta por medio de ventanas.
- **Manejo de Datos (MD).** Se obtiene de cada una de las clases del dominio del problema y que se desea sean persistentes

7.4.1. Nivel de Administración (NA)

Clases Principales del Nivel de Administración (NA)

Clases principales que cumplen con los requisitos para controlar los procesos generales para toda la herramienta:

Componente	Clases
Interfaz De Usuario	index, principal1, principal2 projects, introduceDatosPRO,editProject,deleteProject units, introduceDatosUNT, editUnit, deleteUnit users, introduceDatosUSR, editUser, deleteUser ***(las clases a continuación son usadas en todos los niveles)*** errorPage about avisos
Dominio del Problema	suscribe6, godeleteProject suscribe8, godeleteUnit suscribe7, godeleteUser
Manejo de Datos	DataSupport PoolConnection

Tabla 7.3 Clases principales del Nivel de Administración

En el Anexo C, se proporciona el prototipo de la interfaz de usuario, que ejecuta la funcionalidad del nivel de administración de la herramienta.

Casos de Uso del Nivel de Administración (NA)

La parte dinámica del análisis es definida mediante los diferentes casos de uso que se dan en el Nivel de Administración y son:

Num.	Caso de Uso	Descripción
1.	Entrar al sistema	Validar la clave de acceso y la contraseña del usuario.
2.	Crear Proyecto	El Administrador introducirá los datos de control del proyecto.
3.	Editar Proyecto	El Administrador puede editar los datos del proyecto, siempre y cuando sea el creador del proyecto.
4.	Trabajar Proyecto	El participante podrá trabajar con los productos pertenecientes al proyecto, siempre y cuando forme parte de los integrantes del proyecto seleccionado.
5.	Asignar Nuevos Roles	El Administrador podrá asignar nuevos roles al producto seleccionado (desarrollador, revisor o aprobador).
6.	Crear Participantes	El Administrador introducirá los datos de control del nuevo participante.
7.	Editar Participantes	El Administrador puede editar los datos del participante seleccionado.
8.	Crear Unidades Medida	El Administrador introducirá los datos de control de la unidad de medida.
9.	Editar Unidades de Medida	El Administrador puede editar los datos de la unidad de medida seleccionada.

Tabla 7.4 Casos de Uso del Nivel de Administración

7.4.2. Nivel de Control Personal (NP)

Clases Principales del Nivel de Control Personal (NP)

Clases principales que cumplen con los requisitos para controlar un producto de software con Nivel de Control Personal:

Componente	Clases
Interfaz De Usuario	*introduceDatosPNS* editProduct deleteProduct oneProductNS ***(las clase a continuación es usada en los niveles posteriores)*** products, metrics
Dominio del Problema	suscribe ***(las clases a continuación son usadas en los niveles posteriores)*** godeleteProduct saveMetrics
Manejo de Datos	dataSupport poolConnection

Tabla 7.5 Clases principales del Nivel de Control Personal

En el Anexo D, se proporciona el prototipo de interfaz de usuario, que ejecuta la funcionalidad de este nivel de control.

Casos de Uso del Nivel de Control Personal (NP)

La parte dinámica del análisis es definida mediante los diferentes casos de uso que se dan en el Nivel de Control Personal y son:

Num.	Caso de Uso	Descripción
1.	Crear Producto Nivel Personal	El usuario seleccionará el nivel de control e introducirá datos de control.
2.	Editar Producto	El Productor podrá editar el contenido de un producto seleccionado, si es editable.
3.	Mostrar Producto	El participante podrá leer el contenido de un producto seleccionado y evaluar el resultado de las métricas recolectadas.
4.	Registrar métricas Producto	El Productor podrá introducir las métricas básicas para este nivel de control.
5.	Ingresar Repositorio	El Productor podrá ingresar el producto al repositorio central (si el estado del producto es terminado) o copiarlo a un repositorio local (si es editable).

Tabla 7.6 Casos de Uso del Nivel de Control Personal

Diagrama de Estado del Nivel de Control Personal (NP)

Un diagrama de estado que indica los diferentes estados que pasa un producto debido a su comportamiento dinámico provocado por un estímulo.

Diagrama de estado del producto.

En el Nivel de Control Personal (NP) un producto al ser creado está en estado de borrador y editable en una versión inicial.

Se podrá editar un producto mientras el Productor no determine que el producto esta terminado. Durante el desarrollo se debe recolectar el tamaño y tiempo de desarrollo.

Una vez que el Productor termine de editar el producto, lo ingresará en el repositorio compartido y cambiará su estado a terminado. Si el productor atiende una solicitud de cambio verbal a un producto, creará una nueva versión para editarlo y cambia su estado a borrador.

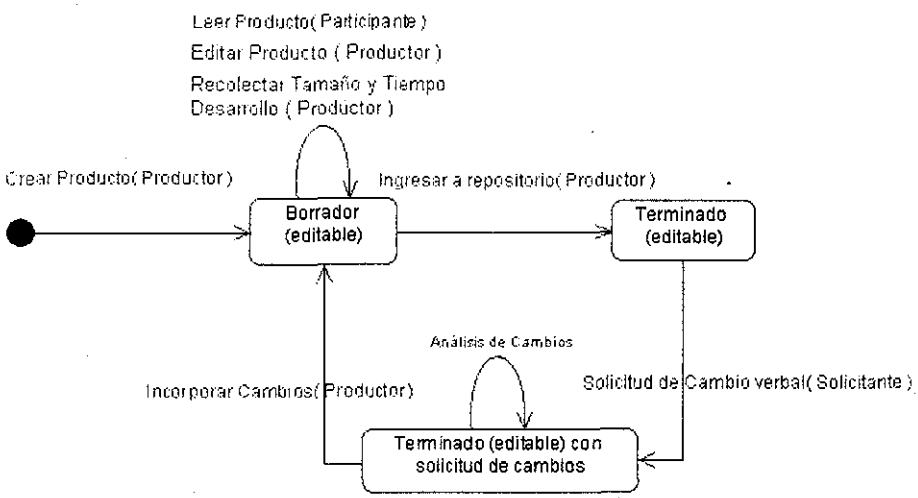


Figura 7.1 Diagrama de Estado del producto: Nivel de Control Personal

7.4.3. Nivel de Control de Equipo (NE)

Clases Principales del Nivel de Control de Equipo (NE).

En este nivel se incluyen clases del Nivel de Control Personal y se definen nuevas clases debido a que en este nivel de control se adicionan los casos de uso que implican la revisión del producto por parte de un Revisor.

Además, se agregan otras clases debido a que en este nivel de control se pretende controlar los requisitos del producto, los defectos encontrados al mismo debido a una revisión y además los cambios solicitados a un producto.

Componente	Clases
Interfaz De Usuario	introduceDatosPNI editProduct1 deleteProduct oneProductNE ***(las clase a continuación es usada en los niveles posteriores)*** requirement, introduceDatosREQ, editRequirement deleteRequirement, oneRequirement defects, introduceDatosDEF, editDefect, deleteDefect oneDefect changes, introduceDatosCAM, editChange, deleteChange, oneChange message, onemessage, deleteMessage
Dominio del Problema	suscribe1 ***(las clase a continuación es usada en los niveles posteriores)*** suscribe3, godeleteRequirement suscribe4, godeleteDefect suscribe5, godeleteChange
Manejo de Datos	dataSupport poolConnection

Tabla 7.7 Clases principales del Nivel de Control de Equipo

En el Anexo E, se proporciona el prototipo de interfaz de usuario, que ejecuta la funcionalidad de este nivel de control.

Casos de Uso del Nivel de Control de Equipo (NE)

La parte dinámica del análisis es definida mediante los diferentes casos de uso que se dan en el Nivel de Control de Equipo y son:

Num.	Caso de Uso	Descripción
1.	Crear Producto Nivel de Equipo	El usuario seleccionará el nivel de control e introducirá datos de control, antes debe haber sido creado los requisitos del producto.
2.	Editar Producto	El Productor podrá editar el contenido de un producto seleccionado, si es editable o El Revisor podrá editar el contenido de un producto seleccionado, si es no editable.
3.	Mostrar Producto	El participante podrá leer el contenido de un producto seleccionado y evaluar el resultado de las métricas recolectadas.
4.	Registrar métricas Producto	El Productor o Revisor podrá introducir las métricas básicas para este nivel de control.
5.	Ingresar Repositorio	El Productor podrá ingresar el producto al repositorio central (si el estado del producto es terminado) o copiarlo a un repositorio local (si es editable). El Revisor podrá recuperar el archivo del repositorio central para revisarlo.
6.	Enviar a evaluación un producto	El productor selecciona un producto, cambia el estado del producto a no editable.
7.	Revisar producto	El Revisor, cambia el estado del producto a revisado y no editable
8.	Rechazar producto	El Revisor modifica los datos de control del producto, y reporta los defectos encontrados si los hubiera.
9.	Crear requisito	Un participante podrá introducir los datos de control del requisito.
10.	Editar requisito	El Productor del requisito puede editar los datos de control del requisito.
11.	Mostrar requisito	Un participante puede observar las características del requisito
12.	Crear reporte de defectos	El Revisor introduce los datos de control y descripción de los defectos encontrados a un producto cuando ha sido evaluado
13.	Editar defecto	El Productor del reporte de defectos puede editar los datos de control.
14.	Mostrar Reporte de Defectos	Un participante puede mostrar las características de un reporte de defectos
15.	Crear solicitud de cambio	Un participante puede introducir los datos de control y descripción de la solicitud de cambio a un producto
16.	Editar solicitud de cambio	El Productor de la solicitud de cambio puede editar los

		datos de control de la solicitud de cambio
17.	Mostrar solicitud de cambio	Un participante puede mostrar las características de una solicitud de cambio
18.	Enviar a evaluación un producto	El Productor de la solicitud de cambio envía al Productor la solicitud de cambio.
19.	Rechazar solicitud de cambio	El Productor analiza la solicitud de cambio y la rechaza, se envía un mensaje al Productor de la solicitud de cambio.
20	Aceptar solicitud de cambio	El Productor analiza la solicitud de cambio y la acepta, el producto cambia a estado editable, se envía un mensaje al Productor de la solicitud de cambio.
21.	Leer mensaje	El Productor, Revisor o Participante puede leer un mensaje sobre el estado de un producto, revisión o solicitud de cambio

Tabla 7.8 Casos de Uso del Nivel de Control de Equipo

Diagrama de Estado del Nivel de Control de Equipo (NE)

En este nivel de control los objetos con un comportamiento dinámico se obtiene de las clases de *productos, requisitos, defectos y cambios* que corresponden al producto, al requisito, el defecto y solicitud de cambio.

Diagrama de estado del producto.

En el Nivel de Control de Equipo (NE) un producto al ser creado está en estado de borrador y editable en una versión inicial. Se podrá editar un producto mientras el Productor no lo envíe a revisión. Durante el desarrollo se debe recolectar el tamaño y tiempo de desarrollo.

Si el productor envía a un Revisor el producto para su evaluación y si es aceptado se genera una nueva versión aprobada y el Revisor lo ingresa al repositorio compartido.

Durante la revisión se debe recolectar el tiempo dedicado a revisión. Si el producto que se está revisando es rechazado se envía al Productor y pasa nuevamente al estado de borrador con una nueva versión. Ahora el productor recolectará el tamaño y tiempo de corrección.

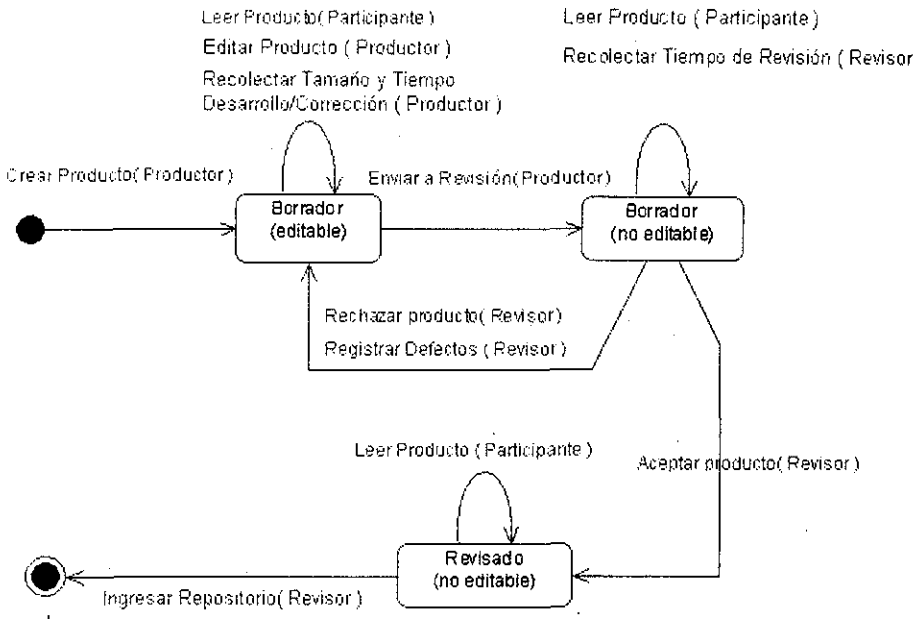


Figura 7.2 Diagrama de Estado del producto: Nivel de Control de Equipo

Diagrama de estado de la solicitud de cambio.

Un solicitante puede crear una solicitud de cambio en estado de borrador y editable y enviársela al Productor.

El Productor podrá rechazar la solicitud de cambio y en este caso cambia su estado a Rechazado.

El Productor podrá aprobar la solicitud de cambio y en este caso cambia su estado a Aprobado.

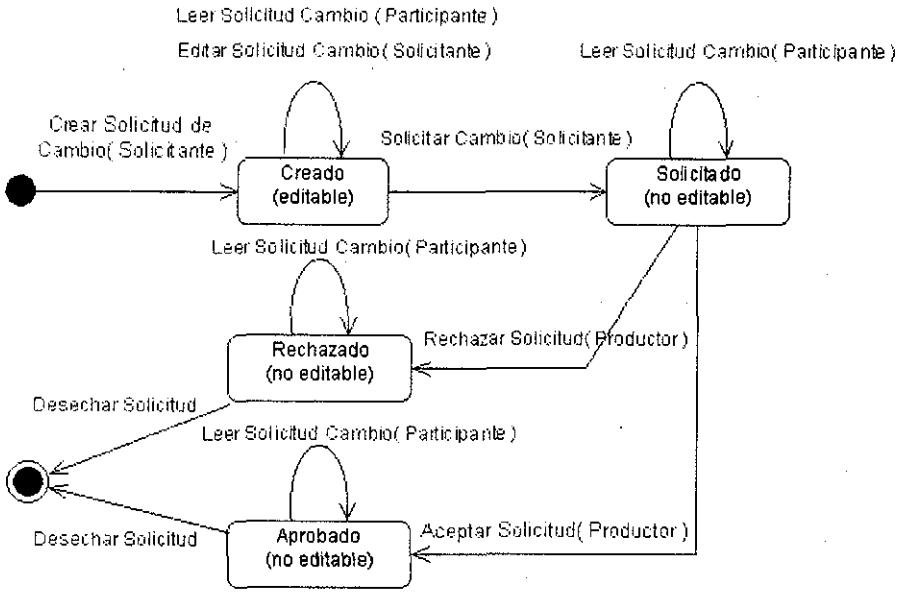


Figura 7.3 Diagrama de Estado de una solicitud de cambio: Nivel de Control de Equipo

7.4.4. Nivel de Control de Negocio (NN)

Clases Principales del Nivel de Control de Negocio (NN).

Incluye las clases del Nivel de Control Intermedio, excepto las que se modifican porque en este nivel existe un Aprobador y un Bibliotecario y se necesitan datos de control del Aprobador.

También se modifican algunas clases que se relacionan con la solicitud de cambio debido a que un Aprobador la evalúa y reporta defectos encontrados al producto de software.

En la siguiente tabla se muestran las clases que intervendrían en este nivel de control.

Componente	Clases
Interfaz De Usuario	introduceDatosPNC editProduct2 deleteProduct oneProductNN ***(mas todas las clases definidas en el nivel anterior referente a requisitos, defectos, cambios, mensajes)***
Dominio del Problema	suscribe2 ***(mas todas las clases definidas en los niveles anteriores)***
Manejo de Datos	dataSupport poolConnection

Tabla 7.9 Clases principales del Nivel de Control de Negocio

En el Anexo F, se proporciona el prototipo de interfaz de usuario, que ejecuta la funcionalidad de este nivel de control.

Casos de Uso del Nivel de Control de Negocio (NN).

Incluye los casos de uso del Nivel de Control de Equipo, excepto los que se modifican por el hecho de existir un Aprobador y porque existe la necesidad de identificar datos de control del mismo.

Se modifican algunos casos de uso que se relacionan con la solicitud de cambio debido a que en este nivel el Aprobador es el que la evalúa.

Asumiremos que se utilizan los mismos casos de uso del Nivel de Control de Equipo (NE) y solo mencionaremos los que se modifican y los que se agregan:

Num.	Caso de Uso	Descripción
1.	Crear Producto Nivel de Negocio	El usuario seleccionará el nivel de control e introducirá datos de control, antes debe haber sido creado los requisitos del producto.
2.	Editar Producto	El Productor podrá editar el contenido de un producto seleccionado, si es editable o El Revisor podrá editar el contenido de un producto seleccionado, si es no editable o El Aprobador podrá editar el contenido de un producto seleccionado, si es no editable.

3.	Mostrar Producto	El Administrador de Calidad podrá evaluar el resultado de las métricas recolectadas. El participante podrá leer el contenido de un producto seleccionado y evaluar el resultado de las métricas recolectadas.
4.	Métricas Producto	El Productor, Revisor o Aprobador podrá introducir las métricas básicas para este nivel de control.
5.	Ingresar Repositorio	El Productor podrá ingresar el producto al repositorio central (si el estado del producto es terminado) o copiarlo a un repositorio local (si es editable). El Revisor podrá recuperar el archivo del repositorio central para revisarlo. El Aprobador podrá recuperar el archivo del repositorio central para aprobarlo.
6.	Enviar a evaluación un producto	El Productor de la solicitud de cambio envía al Aprobador la solicitud de cambio.
7.	Rechazar solicitud de cambio	El Aprobador analiza la solicitud de cambio y la rechaza, se envía un mensaje al Productor de la solicitud de cambio.
8.	Aceptar solicitud de cambio	El Aprobador analiza la solicitud de cambio y la acepta, el producto cambia a estado editable, se envía un mensaje al Productor de la solicitud de cambio.

Tabla 7.10 Casos de Uso del Nivel de Control de Negocio

Diagramas de Estado del Nivel de Control de Negocio (NN).

Los diagramas de estado para el producto y la solicitud de cambio se modifican porque tienen que ser evaluados por un Aprobador.

Diagrama de estado del producto.

En el Nivel de Control de Negocio (NN) un producto al ser creado está en estado de borrador y editable en una versión inicial.

Se podrá editar un producto mientras el Productor no lo envíe a revisión. Si el Productor envía a un Revisor el producto para su evaluación y si es aceptado se genera una nueva versión aprobada, el Revisor cambia el estado del producto a revisado y lo envía a un Aprobador para su evaluación.

Si el Revisor rechaza el producto lo envía al Productor y se mantiene en estado de borrador con una nueva versión. Si el Aprobador al evaluar el producto lo aprueba lo enviará a un Bibliotecario para su ingreso a la Biblioteca y cambia su estado a aprobado.

Si el Aprobador rechaza el producto lo envía al Productor y pasa nuevamente al estado de borrador con una nueva versión.

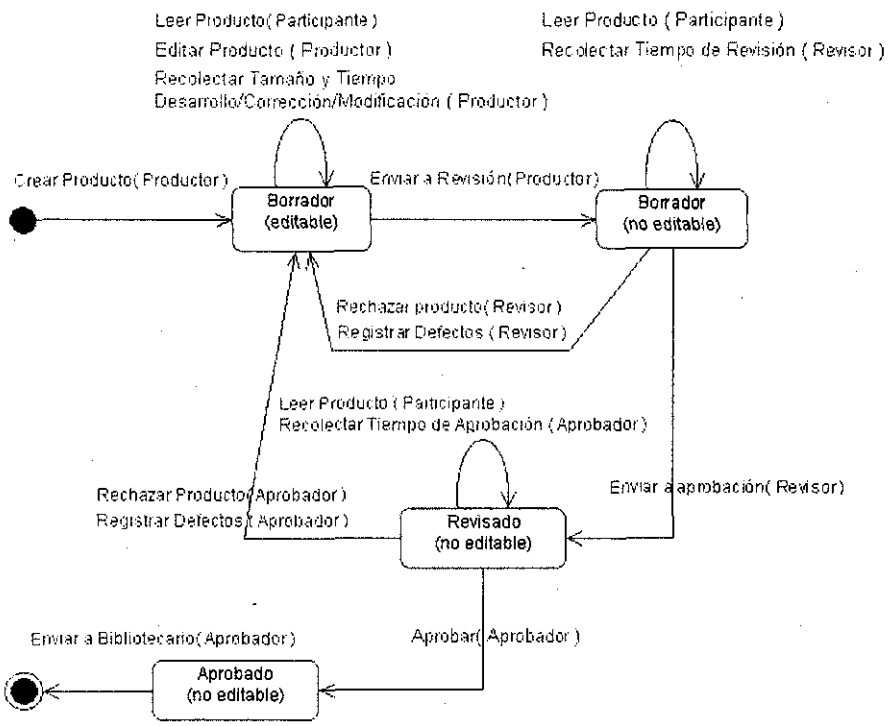


Figura 7.4 Diagrama de Estado del producto: Nivel de Control de Negocio

Diagrama de estado de la solicitud de cambio.

Un solicitante puede crear una solicitud de cambio en estado de borrador y editable y enviársela al Aprobador.

El Aprobador podrá rechazar la solicitud de cambio y en este caso cambia su estado a Rechazado.

Si el Productor aprueba la solicitud de cambio y en este caso cambia su estado a Aprobado y una vez atendida cambiá su estado a Procesado.

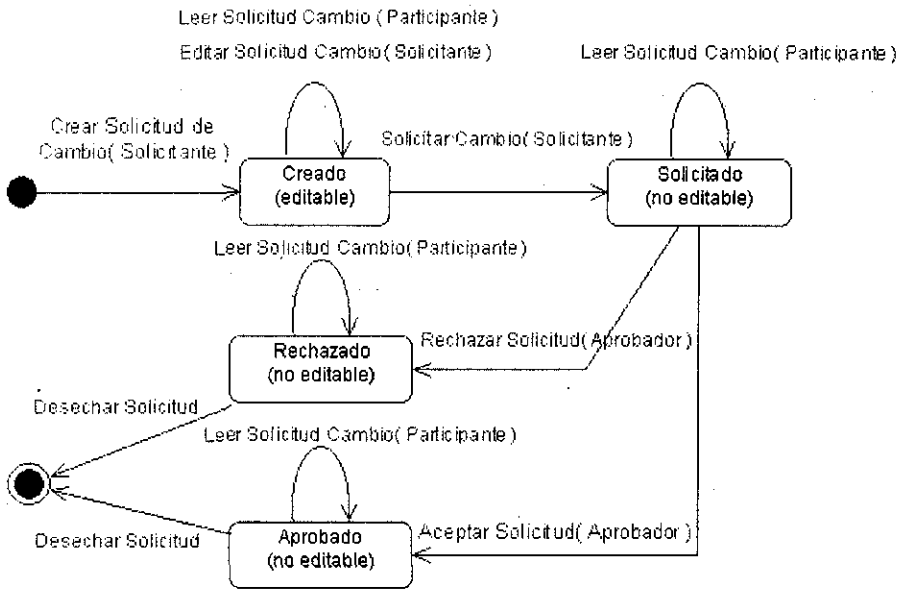


Figura 7.5 Diagrama de Estado de una solicitud de cambio: Nivel de Control de Negocio

7.5 Diseño

Durante esta fase se refinan los modelos que se obtuvieron durante la fase de análisis y se adaptan los modelos obtenidos de los niveles de control al ambiente de implementación.

Características del ambiente de implementación:

- *Hardware.* El servidor deberá tener la capacidad suficiente para procesar las peticiones que se hagan desde múltiples servidores que se conectarán con el primero vía Internet

por medio de un browser. Por esta razón se requiere una computadora con procesador mayor a 200 MHz, con conexión a la red y 128 Mb de memoria RAM como mínimo.

Los clientes, por otra parte, no requieren de mayor capacidad de proceso que la requerida para mantener un browser de Internet funcionando.

- *Software*. En el servidor se requiere un manejador de bases de datos Oracle o Access, servidor de páginas http y de jsp's (se sugieren Apache para el primero y TomCat para el segundo).

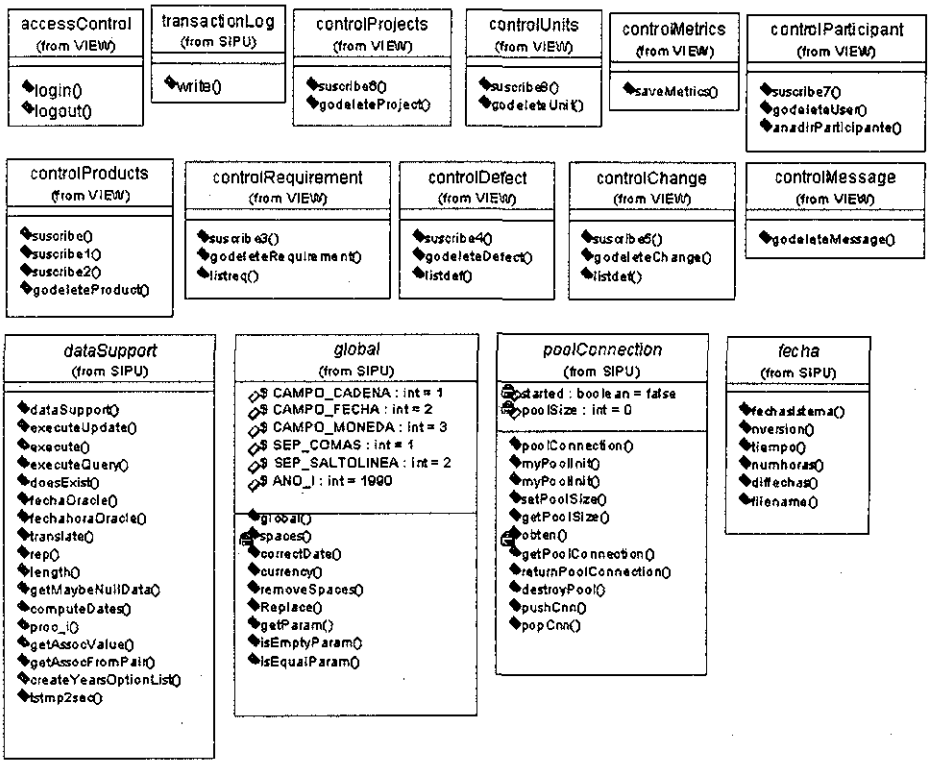
Los clientes únicamente requieren el browser de Internet, ya que únicamente recibirán páginas html. Sin embargo, se requiere que dicho browser tenga la capacidad de soportar código JavaScript a fin de ejecutar las posibles validaciones de los datos que se enviarán al servidor.

7.5.1 Clases resultantes del diseño

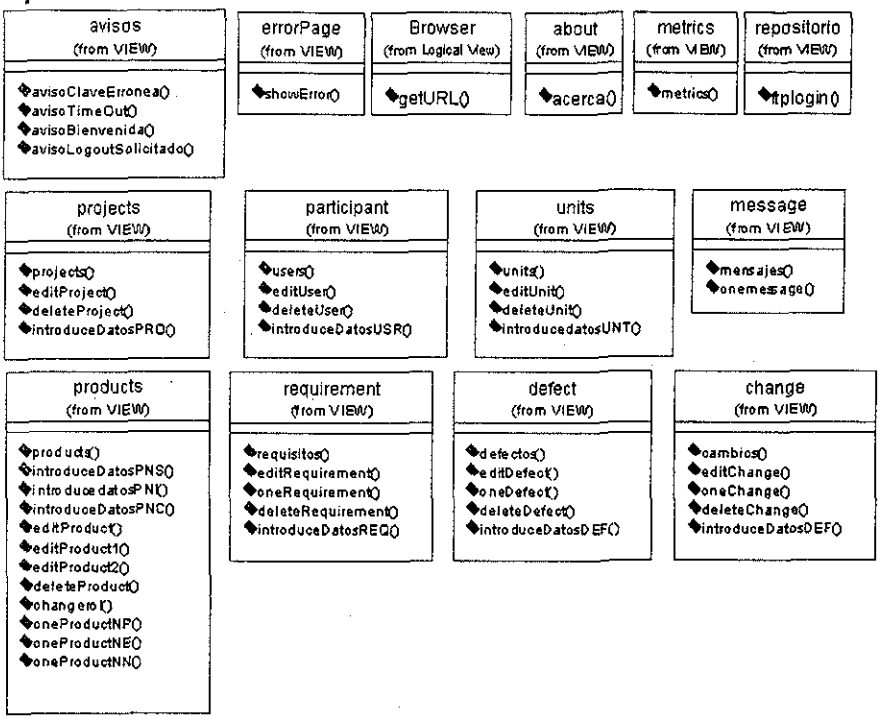
Al realizar el refinamiento de las clases obtenidas en la fase de análisis y mediante el uso de la arquitectura Modelo Vista-Controlador se agruparon las clases en tres grandes bloques:

- El *modelo* el cual contiene los datos y la funcionalidad de la aplicación. Es independiente de la representación de los datos.
- Las *vistas* muestran la información al usuario de una cierta forma. Existen todas las que se necesite definir.
- Cada *vista* tiene un *controlador* asociado. Los controladores reciben entradas en forma de eventos que responden a mandos realizados por el usuario a través del ratón o del teclado. El control traduce estos eventos a peticiones a la *vista* o al *modelo*.

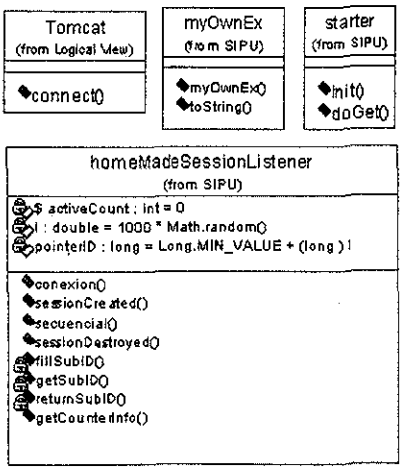
Paquete Modelo



Paquete Vista



Paquete Controlador



7.5.2 Arquitectura del sistema

Para la descripción de la arquitectura del sistema se establece la arquitectura lógica y física. La arquitectura lógica del sistema muestra la agrupación de las clases en categorías, mientras que la arquitectura física muestra el agrupamiento de los módulos en subsistemas.

Arquitectura Lógica

Las categorías que integran el sistema están compuestas por:

- Paquete Modelo – que contiene la funcionalidad de la herramienta.
- Paquete Vistas – contiene la información que se muestra al usuario.
- Paquete Controlador –acepta los eventos de entrada, traduce los eventos de entrada a peticiones al modelo o a las vistas.

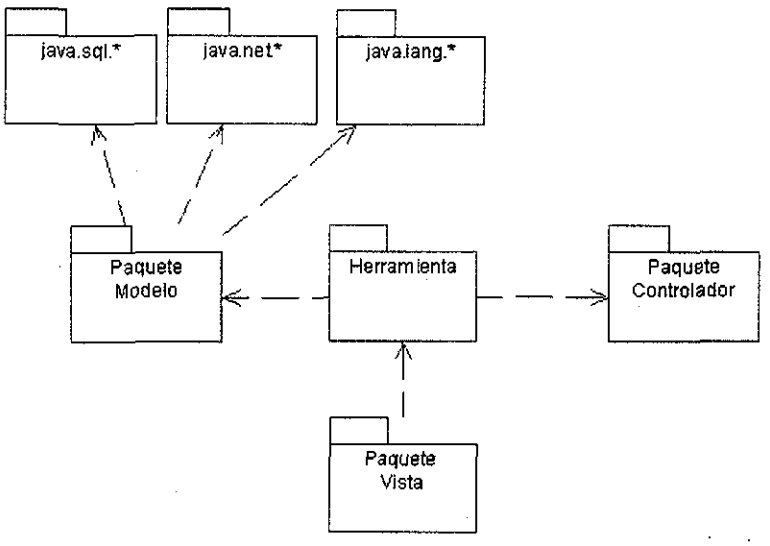


Figura 7.6 Arquitectura Lógica de la herramienta

Arquitectura Física

La arquitectura física comprende el agrupamiento de los módulos en subsistemas y esta compuesto por:

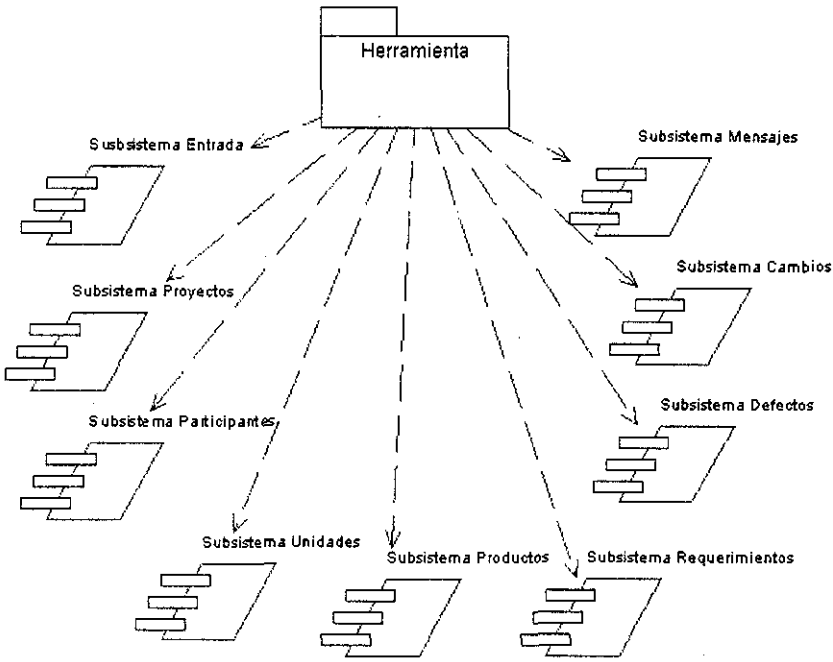


Figura 7.7 Arquitectura Física de la herramienta

Todos los subsistemas se describen a continuación:

Subsistema de Entrada, donde se verifica que el usuario que intenta entrar al sistema tiene derecho de acceso.

Subsistema Proyectos, control de un proyecto, se crean y editan los datos de un proyecto.

Subsistema Participantes, control de participantes, se crean y editan los datos de un participante de un proyecto. Es posible crear usuarios que posean un nivel de administrador.

Subsistema Unidades, control de unidades de medida, se crean y editan las unidades de medida que servirá como dato de medida para la creación de un producto.

Subsistema Productos, control del producto, donde se crea y editan, se muestra su contenido y se cambia de estado.

Subsistema Requerimientos, control de requisitos, donde se crea y editan los requisitos con los que debe cumplir un producto, además se muestra su contenido.

Subsistema Defectos, control de defectos, donde se crea y editan los defectos encontrados a un producto después de un proceso de evaluación, se muestra su contenido y se cambia su estado.

Subsistema Cambios, control de cambios, donde se crea y editan las solicitudes de cambio a los productos, se muestra su contenido y se cambia su estado.

Subsistema Mensajes, permite visualizar los mensajes enviados entre los roles de producción.

7.5.3 Diseño de la Base de Datos

Para la creación de la base de datos se consideran las clases que son persistentes, estas pertenecen al grupo de Manejo de Datos (ver análisis 7.4).

Se utilizó un esquema de base de datos relacional. Todas las relaciones se crearon con el apoyo de Microsoft Access. La figura 7.8 muestra el diagrama entidad-relación para todo el sistema.

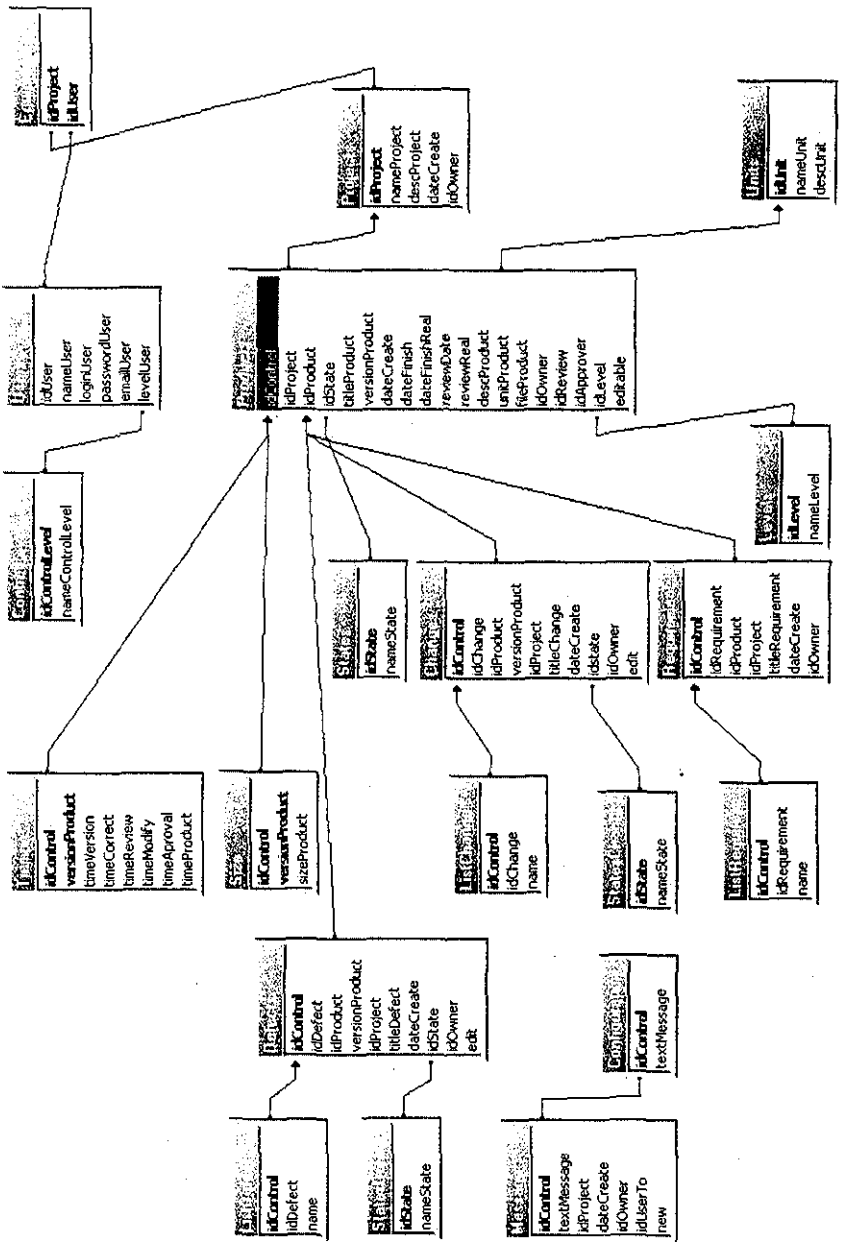


Figura 7.8 Diagrama Entidad-Relación de la herramienta

7.6 Aportaciones

A continuación se efectúa una comparación de la evolución que ha sufrido esta herramienta iniciando con los prototipos A y B, estudios previos a esta tesis:

A. Prototipo ConDor. Del M. en C. Carlos Pérez Escobar.

B. Prototipo C3PoS. Del M. en C. Martín de Jesús Jiménez

C. Herramienta para controlar el proceso de un producto de software

No.	Característica	Productos		
		A	B	C
1	Control de versiones	√	√	√
2	Soporte de la mesa de Control de la Configuración del Software	√	√	√
3	Control de cambios	√	√	√
4	Definición de los roles asociados a un producto	√	√	√
5	Control de seguridad, mediante el acceso autorizado a la herramienta y a los productos	√	√	√
6	Reporte de los defectos encontrados al producto durante las verificaciones y validaciones	X	√	√
7	Niveles configurables de control para un producto de software	X	√	√
8	Trabajo en un ambiente en red	X	√	√
9	Registro histórico de las versiones de un producto	X	X	√
10	Niveles configurables de control para un proyecto	X	X	√
11	Soporte de métricas con el fin de determinar tendencias de algún producto o proyecto	X	X	√
12	Permitir la transferencia de productos entre repositorios ubicados localmente y repositorios compartidos en forma central	X	X	√
13	Soportar la tolerancia a fallas debido a su trabajo bajo un ambiente de red	X	X	√
14	Permitir la integración de nuevas unidades de medida	X	X	√

√ : Cumple con la característica
X : No cumple con la característica

Resumen:

En este capítulo se efectuó el análisis y diseño orientado a objetos de la herramienta a desarrollar. Se utilizó como lenguaje de modelado UML (Unified Modeling Language), debido a las facilidades que ofrece para visualizar, especificar, construir y documentar sistemas orientados a objetos. Para efectuar un análisis y diseño adecuados, se propuso dividir la herramienta en distintos niveles de control:

- Nivel de Control de Administración
- Nivel de Control Personal
- Nivel de Control de Equipo
- Nivel de Control de Negocio

Para cada nivel se definieron las clases principales, casos de uso y diagramas de estado necesarios. Finalmente se especificó la arquitectura física y lógica, el diagrama entidad-relación y las aportaciones de la herramienta con respecto a trabajos anteriores.

Conclusiones

Uno de los objetivos de este trabajo fue la integración de un modelo de control de cambios y calidad (*modelo cualitativo*) a un modelo de manejo de métricas (*modelo cuantitativo*) en un solo ambiente de trabajo. Esta integración permitió la construcción de una herramienta que permite controlar el proceso de un producto de software.

Durante el transcurso de la elaboración, se encontraron una serie de premisas que es necesario puntualizarlas:

1. A pesar de que el modelo de control de cambios y calidad integra en un solo proceso la producción, corrección, revisión, aprobación y el control de cambios de los productos de software, éste no permite asegurar calidad en el proceso mediante el cual el producto fue desarrollado.

2. Del estudio efectuado pudimos reconocer que la mejor forma de evaluar y mejorar el proceso de desarrollo del software es la medición; es decir medir el esfuerzo, costos, duración, defectos y cambios. A pesar de esto se pudo observar que la situación real en la industria de software revela un uso limitado de la medición de procesos.

3. Luego de observar las características y ventajas de las métricas de software, se decidió incorporar éstas dentro de un proceso de medición claro y fácil de entender, basado fundamentalmente en las propuestas identificadas de la literatura actual, con el objetivo de valorar la eficiencia del proceso y descubrir áreas que necesitan ser trabajadas.

4. A pesar de las ventajas de cada modelo, se pudo observar que la utilización individual no permitía evaluar, mejorar y controlar el proceso de desarrollo del software, con el propósito de mejorar la calidad de los productos. Así pues, se integró los modelos

dentro de un solo ambiente de trabajo a fin de no tener que recurrir a procesos separados para alcanzar el objetivo propuesto.

5. Con la integración de los modelos a través de una sola herramienta se obtuvo una mayor integración y coordinación de las actividades de los miembros que intervienen en el desarrollo y evaluación de los productos de software, mejorando la comunicación y sobre todo la colaboración, que representa la piedra angular de un grupo de trabajo.

6. Tanto en la integración de los modelos cuanto en la elaboración de la herramienta, se vio reflejado la importancia de contar con un lenguaje de modelado de procesos, pues de esta forma todas las entidades son definidas en un lenguaje adecuado de representación sin demasiada complejidad; así el riesgo de diferentes interpretaciones se reduce ampliamente. Se eligió como lenguaje de modelado a UML (Unified Modeling Language, aceptado por Object Management Group, 1997), fundamentalmente porque a mas de ofrecer una descripción sobre la información del proceso, describe el aspecto dinámico del mismo. Además es ampliamente usado y sustentado por muchas herramientas comerciales disponibles.

7. Un factor fundamental para la creación de la herramienta fue decidir la tecnología de desarrollo. Luego de un análisis se decidió crear una aplicación que este disponible bajo la WEB, para lo cual se escogió la tecnología Java Server Pages debido a las ventajas que ofrece: habilidad de separar la lógica del programa de su presentación, portabilidad, debido a que las páginas JSP son compiladas al *bytecode* de Java, facilidad para el reuso, y extensibilidad a través de librerías de *tags* personalizados.

8. El modelo integrado propone un proceso de medición bien definido, con el cual es posible establecer una colección de medidas sumamente sólida. Sin embargo como las medidas básicas evolucionan, sus definiciones y usos también puede expandirse, por ejemplo:

- Los reportes de problemas pueden expanderse a estado de seguimiento, tipo, severidad, y prioridad;
- Los atributos de tamaño pueden ser rastreados por el lenguaje, plataforma, estado de desarrollo, origen, y método de producción;
- Los atributos de esfuerzo pueden agregarse para rastrear la clase laboral, fase, y actividades realizadas; y
- El tiempo puede ser rastreado por fechas y criterios de terminación.

9. La utilización de esta herramienta, al igual que cualquier otra que sirva para administrar el desarrollo de un producto de software, no puede garantizar que la elaboración del producto sea exitosa. Pero, sin embargo, ayuda al administrador a tomar decisiones más reales que ayuden a controlar los riesgos y problemas a los que se enfrentará durante el desarrollo.

Trabajos a futuro

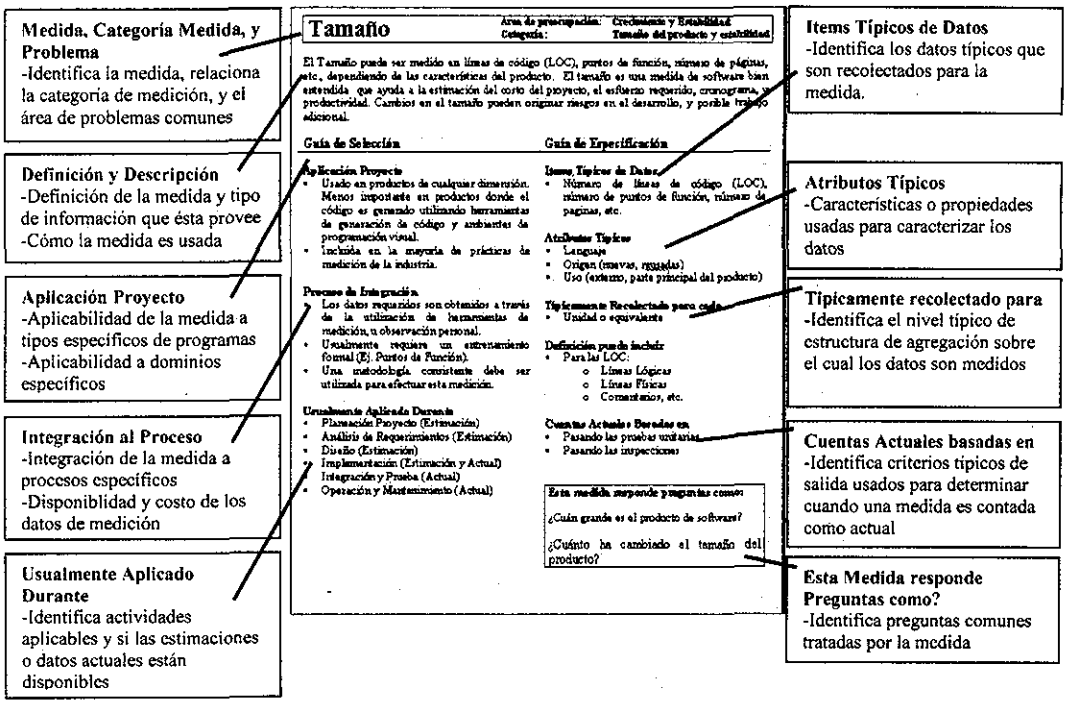
Este trabajo constituye la base para la implementación de una herramienta con mayor funcionalidad, para lo cual se hacen las siguientes recomendaciones:

- La posibilidad de incluir un control más completo a nivel de proyecto, con administración de roles, requisitos y administración de configuración.
- Permitir la integración con otras herramientas que existen en el mercado para permitir crear productos con diferentes formatos.
- Permitir la transferencia de productos entre repositorios ubicados localmente y repositorios compartidos en forma central, sin tener que recurrir a otras herramientas.

Anexos

Anexo A: Tabla de Descripción de Medidas

Las *tablas de descripción de medidas* proveen dos tipos de información. La **guía de selección,** usada para determinar si la medida es aplicable al proyecto, su gestión asociada y sus procesos técnicos. El segundo tipo de información es la **guía de especificación,** usada para definir los datos específicos y requisitos de implementación para cada medida.



Anexo B: Instalación de las Tecnologías Implementadas

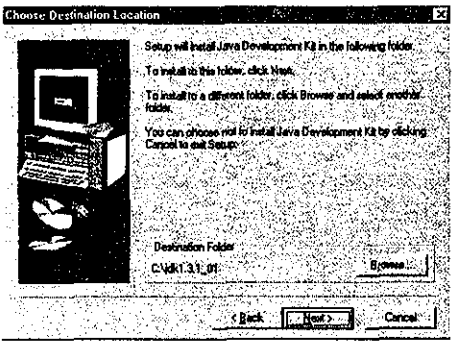
A continuación se resumen algunos elementos de tecnología que se utilizaron para crear la herramienta para controlar el proceso de un producto de software.

Preliminares

Para poder utilizar la herramienta en cuestión es necesario instalar tanto el JSDK de Java como el servidor de JSP's TomCat [WEB4], así como las herramientas EasyCharts y GuildFTPd.

Instalación del JSDK

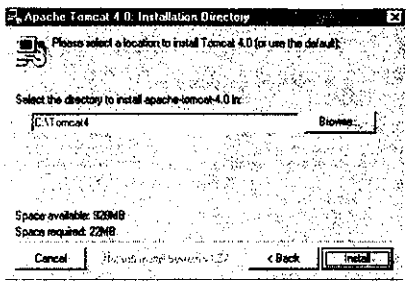
Este sistema utiliza la versión 1.3.0.1_01 del JSDK de Sun. Es necesario que, después de su instalación, se actualice la variable de ambiente PATH a fin de que las compilaciones de las clases puedan tener efecto.



Pantalla de instalación del JSDK 1.3.1_01

Instalación del servidor TomCat

Posterior a la instalación del JSDK se requiere la instalación del servidor de JSP's. Para esta herramienta se empleó el servidor TomCat versión 4.0.



Pantalla de instalación del servidor TomCat 4.0

Es necesario que esta instalación se haga posteriormente a la instalación del JSDK, puesto que el instalador del servidor TomCat requiere la ubicación en donde se encuentra el JSDK. TomCat hace uso de las clases del JSDK en el momento en que realiza la compilación de los JSP's.

Además, se decidió emplear la versión 4.0 puesto que se hace uso de los eventos asociados con la clase que implementa la interfase `HttpSessionListener`. Ésta, como se verá más adelante, controla, entre otros, los eventos asociados con el inicio y finalización de la sesión (invocación de JSP's dentro de la herramienta, por parte de un usuario remoto, vía un browser).

A diferencia de las versiones anteriores, esta versión no requiere de configuraciones adicionales para su funcionamiento.

Por omisión, durante la instalación, se establece el puerto 8080 para realizar la conexión con este servidor. Sin embargo, en el archivo `server.xml`, que se encuentra en la ruta `D:\Tomcat40\conf` (donde `D:\Tomcat40` es el directorio donde se instaló el servidor) puede modificarse este puerto por el puerto 80 con el objeto de no tener que referirse al puerto de conexión en el momento de escribir el URL de conexión desde el cliente. La zona del archivo `server.xml` donde se puede realizar esta modificación es la siguiente:

```
<Connector
className="org.apache.catalina.connector.http.HttpConnector"
               port="8080" minProcessors="5" maxProcessors="75"
               enableLookups="true" redirectPort="8443"
               acceptCount="10" debug="0"
connectionTimeout="60000"/>
```

El archivo web.xml

Es posible ejecutar Servlets inmediatamente después de que se inicia la ejecución del servidor TomCat, y antes de invocarse la ejecución de JSP's. El archivo `web.xml` presente en todo proyecto que administre TomCat sirve para este propósito. Para el proyecto TESIS, este archivo se encuentra en la siguiente ubicación `D:\Tomcat40\webapps\Tesis\WEB-INF`, donde `Tesis` es el nombre de la carpeta donde se aloja toda la herramienta. La apariencia de este archivo, para la herramienta en particular, es la siguiente:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/j2ee/dtds/web-app_2_3.dtd">

<web-app>
  <listener>
    <listener-class>TESIS.homeMadeSessionListener</listener-class>
  </listener>
  <servlet>
    <servlet-name>starter</servlet-name>
    <servlet-class>TESIS.starter</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <session-config>
    <session-timeout>20</session-timeout>
  </session-config>
</web-app>
```

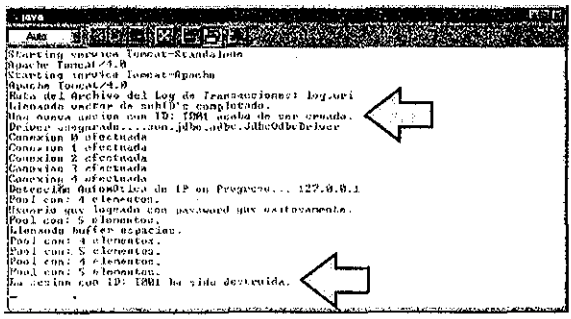
El Servlet starter. En este caso el Servlet que se ejecuta al iniciar TomCat es `starter.class`, el cual se encuentra en la dirección `D:\Tomcat40\webapps\Tesis\WEB-INF\classes\TESIS`, razón por la cual se coloca el prefijo `"TESIS."`

En particular, este Servlet se utilizó para establecer las conexiones con la base de datos antes de que el primer cliente haga uso de la aplicación. De esta forma, al hacer uso de los servicios de la aplicación, ya estarían listas las conexiones para ser usadas.

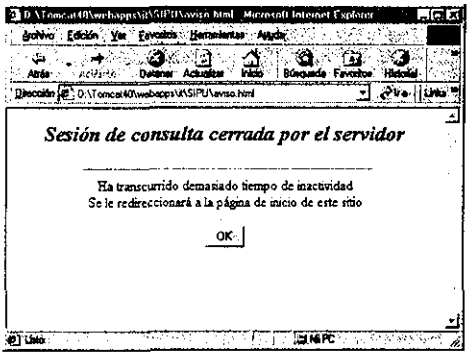
Timeout. Además, en este mismo archivo puede establecerse el tiempo que el servidor tiene que esperar después de un período de inactividad por parte del cliente para dar por terminada la sesión.

El listener. Por último, en este archivo se indica cuál será la clase que funcionará como "listener", es decir, cuál será la clase que implementará los eventos asociados con el inicio y finalización de una sesión (métodos `sessionCreated` y `sessionDestroyed` respectivamente).

En este caso la clase `homeMadeSessionListener.class` es la clase que al implementar la interfase `HttpSessionListener`, Controla los eventos mencionados.



Ventana de ejecución de TomCat con indicaciones de la creación y destrucción de una sesión



Ventana con el mensaje asociado con la finalización de la sesión debido a la inactividad del cliente

Iniciando la herramienta

Pool de conexiones

Debido a la cantidad indeterminada de usuarios que pueden hacer uso simultáneo de una aplicación en Internet, es necesario implementar una estrategia para no exceder el número de conexiones establecidas con la base de datos. Un pool de conexiones resuelve este problema. Básicamente un pool de conexiones es un repositorio de objetos de tipo conexión que administra dichas conexiones para responder a las peticiones que haga una aplicación determinada. De esta forma, dicha aplicación sólo podrá pedir las conexiones disponibles en el pool, regresándolas a éste sin destruirlas.

En particular, el proyecto TESIS emplea un pool que establece 5 conexiones con la base de datos. La clase que implementa el pool es `D:\Tomcat40\webapps\Tesis\WEB-INF\classes\Tesis\poolConnection.class`

En la primera figura de esta página se observa la indicación de las conexiones establecidas por el pool al inicio de la aplicación. Y posteriormente, se observa un reporte con el número de conexiones disponibles cada vez que se hace uso de la base de datos por parte los clientes que lo requieran.

Archivos `*.properties` y la clase `ResourceBundle`

Los parámetros necesarios para establecer la conexión con la base de datos vía JDBC fueron leídos de un archivo `drivers.properties` alojado en `D:\Tomcat40\webapps\Tesis\WEB-INF\classes` por medio de una clase `ResourceBundle`.

Esto permite que únicamente sea necesario cambiar el archivo `drivers.properties` para establecer una conexión con una base de datos diferente o para cambiar cualquier parámetro que se desee, todo sin la necesidad de realizar ninguna recompilación. Sin embargo, será necesario finalizar la ejecución de TomCat para que tenga efecto la actualización de los archivos `*.properties`, ya que éstos son cargados y retenidos en memoria durante la ejecución del servidor (esto mismo es válido para las clases, no así para los archivos `*.html` o `*.jsp`).

El contenido del archivo `drivers.properties` es el siguiente:

```
DBMS=ORACLE
URL=jdbc:oracle:thin:@uxmcc1.iimas.unam.mx:1521:mcic
DRV=oracle.jdbc.driver.OracleDriver
USR=jorgem
PSW=mau99
nConn=5
```

Archivos `*.inc`

Es posible incrustar bloques de código dentro de archivos JSP. Sin embargo, si se realiza cualquier modificación a un archivo *.inc, deberá hacerse alguna modificación al archivo JSP para que el servidor TomCat recompile el JSP.

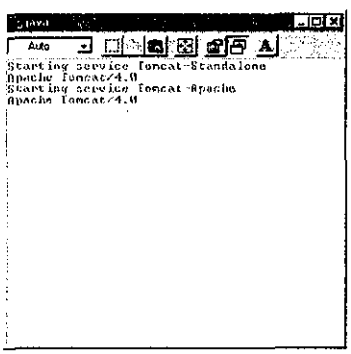
La forma de incrustar este tipo de archivos dentro de JSP's es con una sentencia del siguiente tipo:

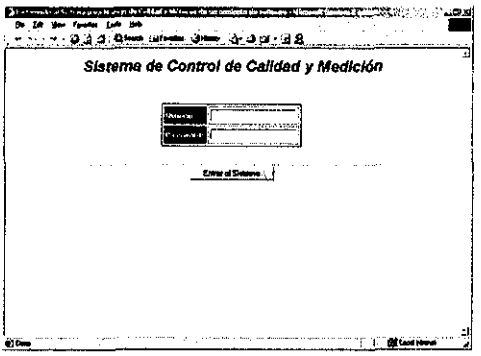
```
<%@ include file="archivo.inc" %>
```

Archivos HTML VS JSP

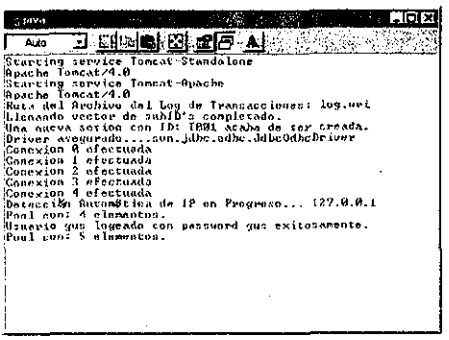
Aunque TomCat puede administrar tanto archivos HTML como JSP, únicamente a través de la ejecución del primer JSP's por parte de un nuevo cliente, se crea una nueva sesión (correspondiente a ese cliente).

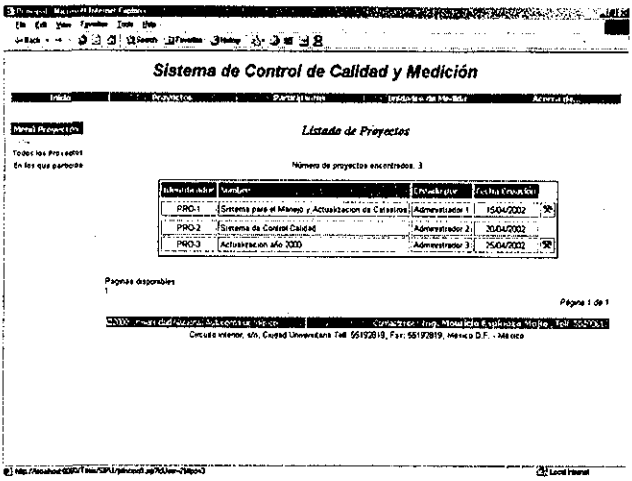
En el proyecto, la primera interfaz mostrada es un HTML (`index.html`), y no es sino hasta la invocación de la siguiente interfaz (`login.jsp`) que se crea la sesión respectiva.





Invocación de la primera interfaz (index.HTML). Se observa que TomCat no ha creado aún una sesión.



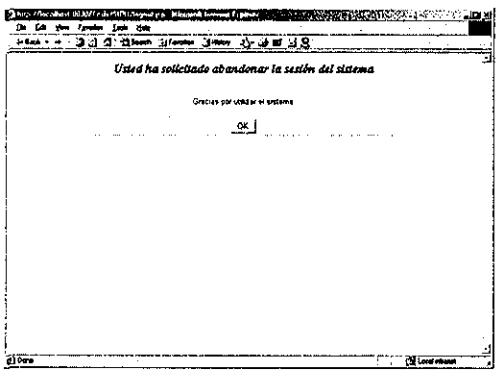


Invocación de la siguiente interfaz (login.JSP). Se observa que TomCat crea una sesión.

El objeto session (clase HttpSession)

El objeto session existe, para cada cliente, desde la primera invocación de un JSP o Servlet, y persiste mientras no transcurra el tiempo de inactividad establecido en el timeout del archivo `web.xml`. Sin embargo, se puede solicitar explícitamente su destrucción (ver liga "logout") mediante el método `invalidate()`.

Este objeto es útil para conservar un contexto relacionado exclusivamente con el usuario en cuestión.

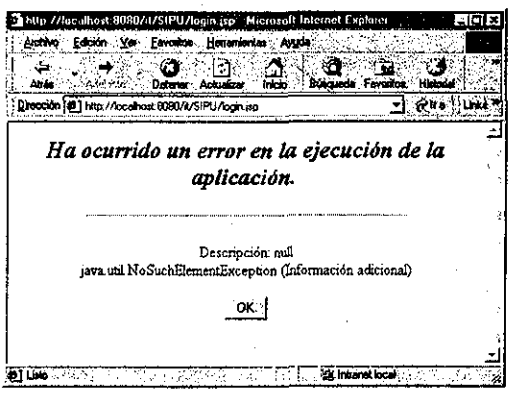


Mensaje mostrado una vez que se eligió terminar la sesión (logout)

Presentación de errores

Existe un JSP que se invoca al ocurrir errores durante la ejecución de la aplicación. De esta forma, el error es mostrado como HTML al cliente que provocó su aparición. Debido a su naturaleza, este archivo JSP puede controlar el flujo del programa.

En el proyecto TESIS, el JSP `errorPage.jsp` manda el flujo de la aplicación a la interfaz inicial (`index.html`).



Apariencia del archivo errorPage.jsp durante la ocurrencia de un error "no existe la base de datos"

El JSP que provoca el error debe tener el siguiente sentencia al inicio de su código:

```
<%@page language="java" import="java.sql.*"
errorPage="errorPage.jsp" %>
```

Asimismo, el jsp indicado como errorPage.jsp debe tener la siguiente sentencia al inicio de su código:

```
<%@ page isErrorPage="true" %>
```

Nota: Sólo puede haber un JSP de este tipo por proyecto.

Cambio del flujo lógico de la aplicación

Se cuenta con distintas opciones para direccionar el flujo de una aplicación:

Código HTML. El flujo puede redireccionarse mediante invocaciones del cliente hechas mediante links o SUBMIT's dentro de FORM's.

Código JavaScript. Aún desde el cliente, puede recurrirse a código JavaScript para redireccionar de una manera dinámica el flujo lógico de la aplicación. En particular pueden emplearse las sentencias como `location URL` o bien `history.go(n)`

La introducción de código JavaScript dentro de un JSP puede hacerse como si se tratara de código HTML normal. Consecuentemente, su ejecución tendrá lugar hasta su llegada al cliente.

Es posible, por tanto, complementar la ejecución de la lógica de la aplicación tanto en el servidor como en el cliente, haciendo que procesos como validaciones, mensajes, o bien, cambios de flujo del programa, tengan lugar en el cliente, reduciendo cargas de proceso innecesarias en el servidor.

Código Java. Puede decidirse el flujo lógico de la aplicación desde el servidor. En particular la sentencia `<jsp:forward page="archivo.html" />`, dentro de un JSP, redirecciona el flujo hacia el archivo indicado.

Cabe mencionar que al encontrarse el flujo con esta sentencia, el código que sigue dentro del JSP ya no se ejecuta.

Por último, si se desea insertar variables de tipo `String` en la cadena que conforma el atributo `page` de esta sentencia, se aconseja alojar TODA LA DIRECCIÓN URL en dicha variable.

Forzando en todo momento la recarga de ciertos archivos JSP

En el proyecto TESIS se requiere que la interfaz asociada con el archivo `login.jsp` sólo pueda ser visible una sola vez por sesión. Es decir, no se desea que el mismo usuario se registre más de una vez por sesión. Este objetivo puede lograrse si se garantizara que siempre que se intente mostrar esta interfaz (via back's, o bien, escribiendo la dirección del archivo login.jsp) en el browser del cliente, se realizara la petición respectiva al servidor en vez de mostrar el archivo html contenido en la historia del explorador.

Una vez hecha la petición al servidor, se pueden tomar las previsiones necesarias para no reenviar el archivo html que corresponde a la ventana de login.

Para conseguir que un determinado archivo JSP, asociado con una interfaz HTML en el browser del cliente, siempre se solicite del servidor, se deben agregar las siguientes sentencias a este archivo:

```
<%
  response.setHeader("Cache-Control","no-cache");
  response.setHeader("Pragma","no-cache");
  response.setDateHeader("Expires",0);
%>
<HEAD><META HTTP-EQUIV="Expires" CONTENT="Mon, 06 Jan 1990 00:00:01
GMT"></HEAD>
```

Las primeras instrucciones (Java) le indican al browser que no deberá mostrar el archivo HTML asociado, sino solicitarlo al servidor de donde provino.

La última instrucción (HTML) le indica al browser cuál es la fecha de expiración del archivo.

Log de transacciones

Debido a que en una aplicación para Internet no se puede tener control sobre la frecuencia con que se hace uso de ésta, ni el momento en que se emplea, resulta conveniente contar con un medio para registrar la historia de las operaciones realizadas.

Esto permite registrar los errores que se están produciendo así como llevar estadísticas sobre el desempeño de la aplicación.

En el proyecto TESIS se creó un archivo de texto (`log.wri`) donde se guarda este tipo de información (conexiones libres, mensajes de error, IP's de clientes, creación y destrucción de sesiones).

Instalación de la herramienta EASYCHARTS

Para evitar algunos de los problemas que ocurren al descargar y ejecutar applets en un explorador web, tales como falta de la maquina virtual de java o compatibilidad, se pueden generar gráficas como imágenes .jpg o .gif en el servidor y enviarlas al explorador web.

Para hacer esto se necesita instalar un servidor de aplicaciones que soporte Java Servlets o Java Server Pages(tm) (JSP). Un ejemplo de servidor de aplicaciones puede ser Tomcat.

Entonces se debe añadir el archivo chartServer.jar al CLASSPATH o servidor de aplicaciones de acuerdo a las instrucciones de la aplicación. Una vez ejecutados los pasos anteriores, se puede iniciar el servidor de aplicaciones.

Para acceder al servlet se debe usar un tag HTML parecido al tag *image*:

```
<img
src="http://server_name:8080/servlet/com.objectplanet.chart.ChartServlet?
chart=bar&
width=200&height=100&
rangeStep=10&
sampleValues=32,87,46,29,36,45,92,76,34,59,27,63,45">

<img
src="http://server_name:8080/servlet/com.objectplanet.chart.ChartServlet?
chart=line&
width=200&height=100&
sampleValues=32,87,46,29,36,45,92,76,34,59,27,63,45">

<img
src="http://server_name:8080/servlet/com.objectplanet.chart.ChartServlet?
chart=pie&
width=200&height=100&
3dModeOn=true&
sampleValues=32,87,46,29,36,45,92,76,34,59,27,63,45">
```

Estas líneas generan una grafica en el servidor, y retornan al cliente una imagen .JPG estándar. Todos los parámetros para las graficas servlets tienen la sintaxis: `parametror=valor` con un signo & usado como delimitador de parámetros.

```
parameterName=value&anotherParameterName=another value
```

Si desea generar otro formato de imágenes se puede usar otro decodificador de imágenes, extendiendo la clase ChartServlet y sobrescribiendo el método encodeChartImage()

Para utilizar el GifEncoder de ACME, siga los siguientes pasos:

1. Descargar el ACME GifEncoder de la dirección:
   http://www.acme.com/java/software/Acme.JPM.Encoders.GifEncoder.html
2. Agregue los paquetes ACME en su CLASSPATH o ambiente de ejecución java
3. Extender la clase ChartServlet y sobreescribir el metodo encodeChartImage()

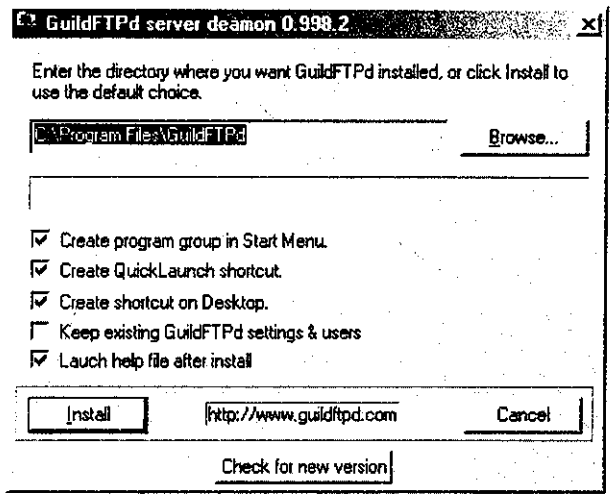
```
import com.objectplanet.chart.*;
import Acme.JPM.Encoders.*;

public class GifChartServlet extends ChartServlet {
   public String encodeChartImage(Image image, OutputStream out)
throws IOException {
      // create a gif image and send it to the client
      GifEncoder encoder = new GifEncoder(image, out);
      encoder.encode();
      out.flush();
      return "image/gif";
   }
}
```

4. Compilar y asegurarse que la clase esta disponible para su java runtime
5. Invocar al servlet:

```
<img
src="http://server:8080/servlet/GifChartServlet?sampleValues=10,20,30,
40,50">
```

Instalación de la herramienta GuildFTPd

Para instalar esta herramienta se requiere haber descargado previamente el archivo GuildFTPd.exe de la dirección www.guildftpd.com.



Pantalla de instalación de GuildFTPd

Anexo C: Interfaz de Usuario del Nivel de Administración

Entrada al Sistema

Para ingresar a la herramienta es necesario digitar la dirección URL en donde se encuentra instalada la aplicación. (http://localhost:8080/Tesis/TESIS/index.html)

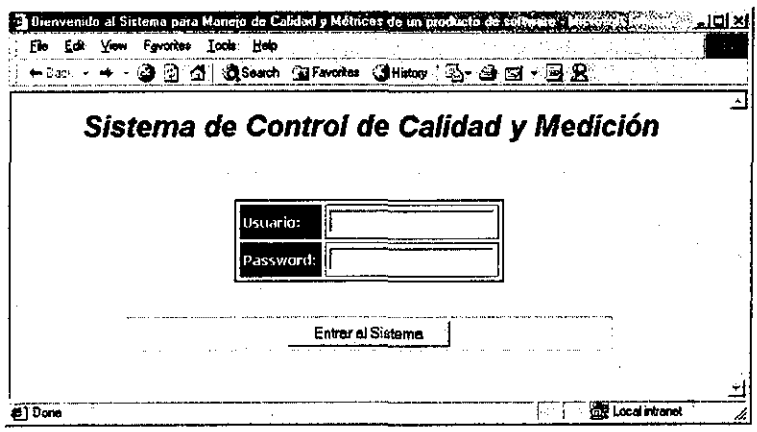


Figura 1. Página de inicio de la herramienta

La página requiere que el participante ingrese el nombre de usuario y su contraseña a fin de acceder a las utilidades de la herramienta; en caso que los datos ingresados no sean validos se mostrara una pagina de aviso (figura 2).

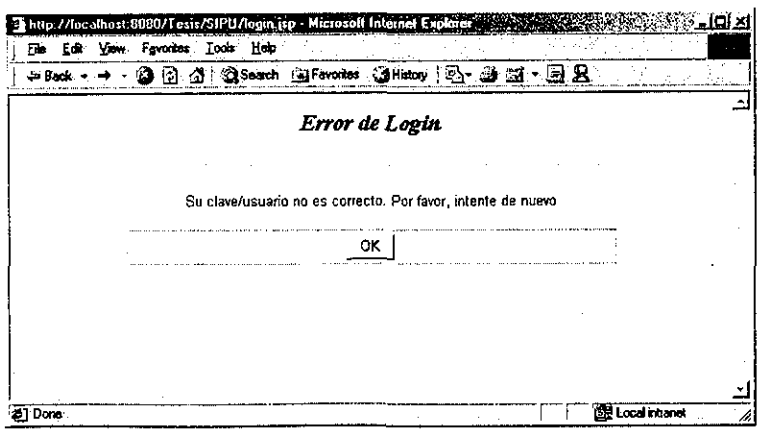


Figura 2. Página de aviso de datos incorrectos

Administración de Proyectos

Una vez que el nombre de usuario y contraseña han sido comprobados, se muestra la información general de los proyectos creados. En la pagina de la figura 3 se muestran además todas las opciones disponibles que pueden ser editadas si el participante tiene un nivel de administrador o visualizadas si el participante es desarrollador.

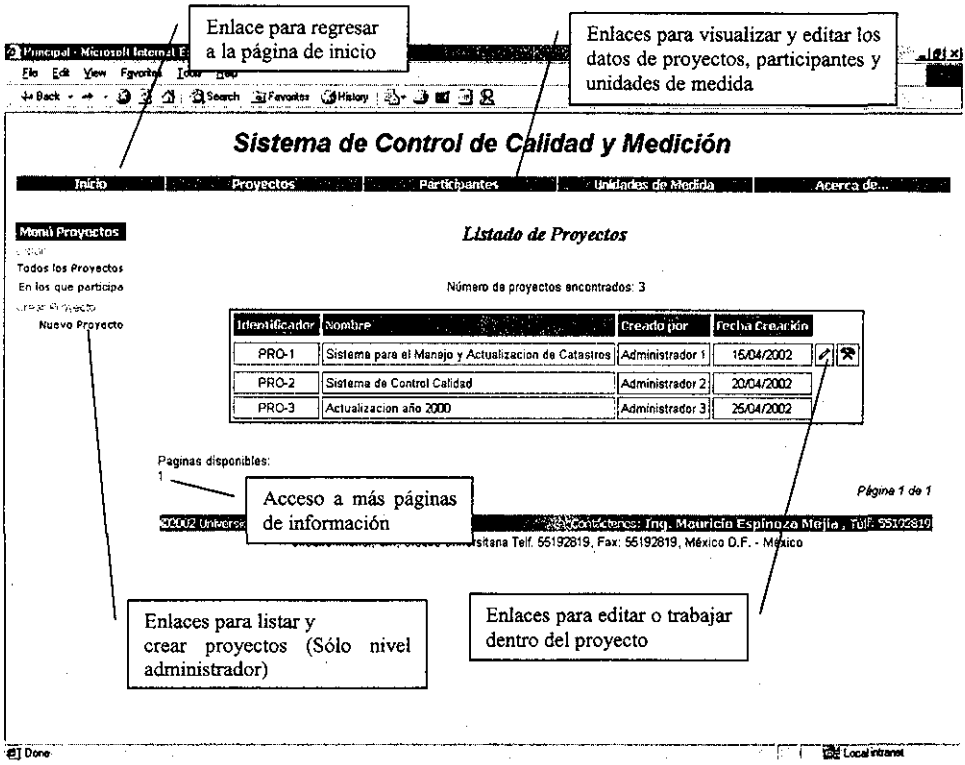


Figura 3. Administración de Proyectos

✎: Permite editar los datos de un proyecto, esta opción esta disponible únicamente si el participante Administrador fue el creador del proyecto.

⚒: Permite ingresar a trabajar con los productos pertenecientes al proyecto. La opción es disponible si el participante forma parte de los integrantes del proyecto.

Crear Proyectos

En la pagina de la figura 4 se puede ingresar la información necesaria para crear proyectos. Esta opción es disponible únicamente si el participante tiene nivel Administrador. Hay que notar que los campos de texto marcados con (*) son obligatorios y en caso de no ser ingresados se mostrara un mensaje de aviso como se observa en la figura 5.

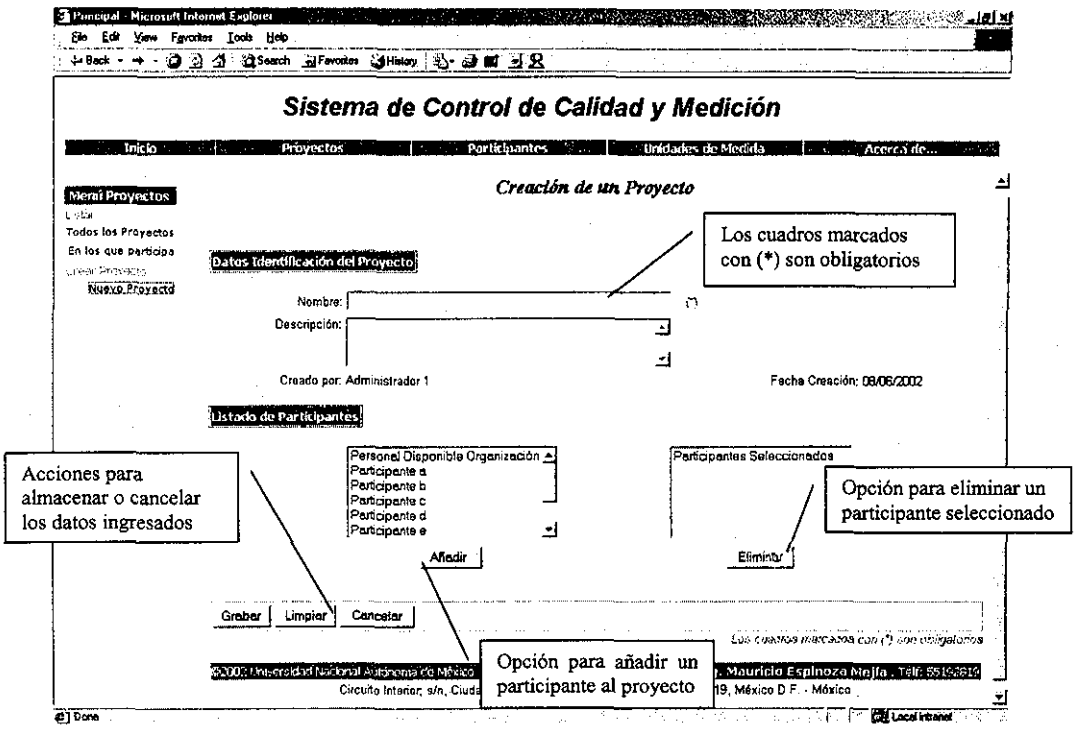


Figura 4. Pagina para crear proyectos

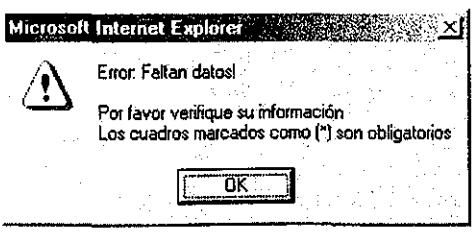


Figura 5. Aviso en caso de no ingresar la información solicitada

Edición de los Datos de un Proyecto

Para editar los datos de un proyecto es necesario ser Administrador y ser el creador del proyecto. Para acceder a esta opción se debe dar un clic en ✎ (ver figura 3). Si esta opción no esta disponible quiere decir que el nivel de control del participante actual no es de Administrador.

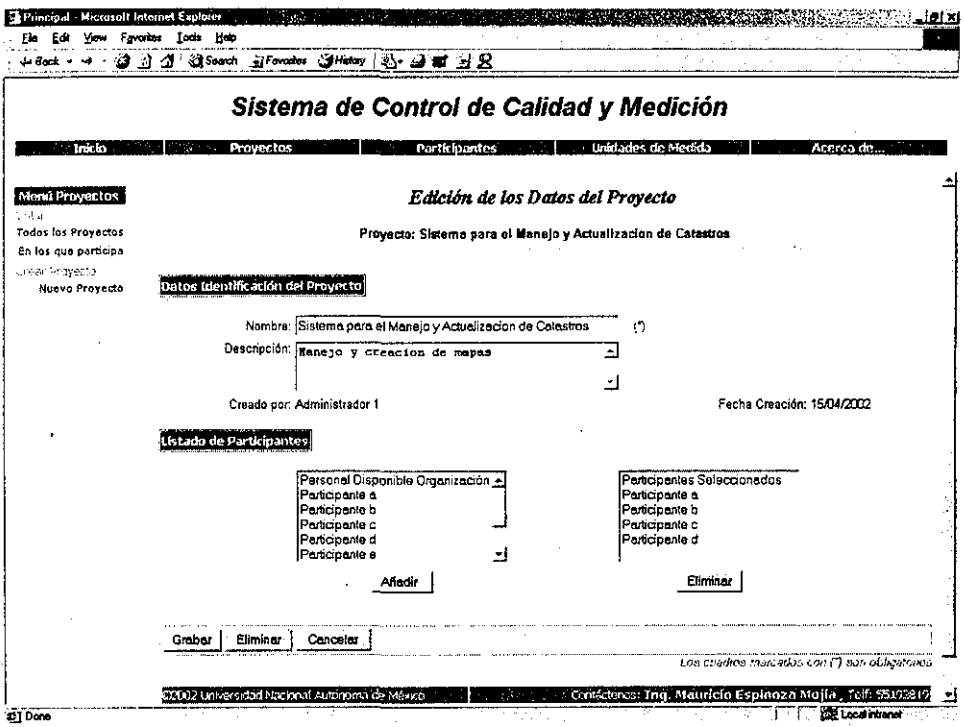


Figura 6. Edición de los datos de un proyecto

Administración de Participantes

En la pagina de la figura 7 se pueden observar los datos generales de los participantes. Los datos de un participante pueden ser editados si es Administrador o visualizadas si el participante es desarrollador.

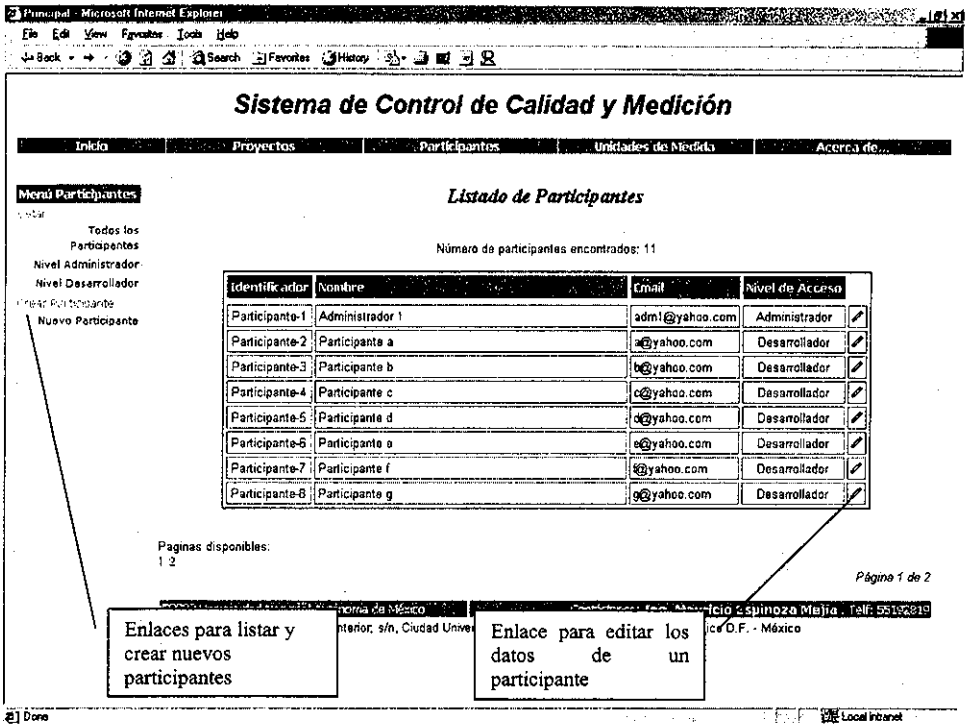


Figura 7. Administración de participantes

Crear Participantes

En la pagina de la figura 8 se puede ingresar la información necesaria para crear participantes. Esta opción es disponible únicamente si es Administrador. Hay que notar que los campos de texto marcados con (*) son obligatorios y en caso de no ser ingresados se mostrara un mensaje de aviso como se observa en la figura 5.

Con esta opción es posible crear participantes que tengan nivel de Administrador, además se debe proporcionar el nombre de usuario y la contraseña que permitirán acceder a la herramienta.

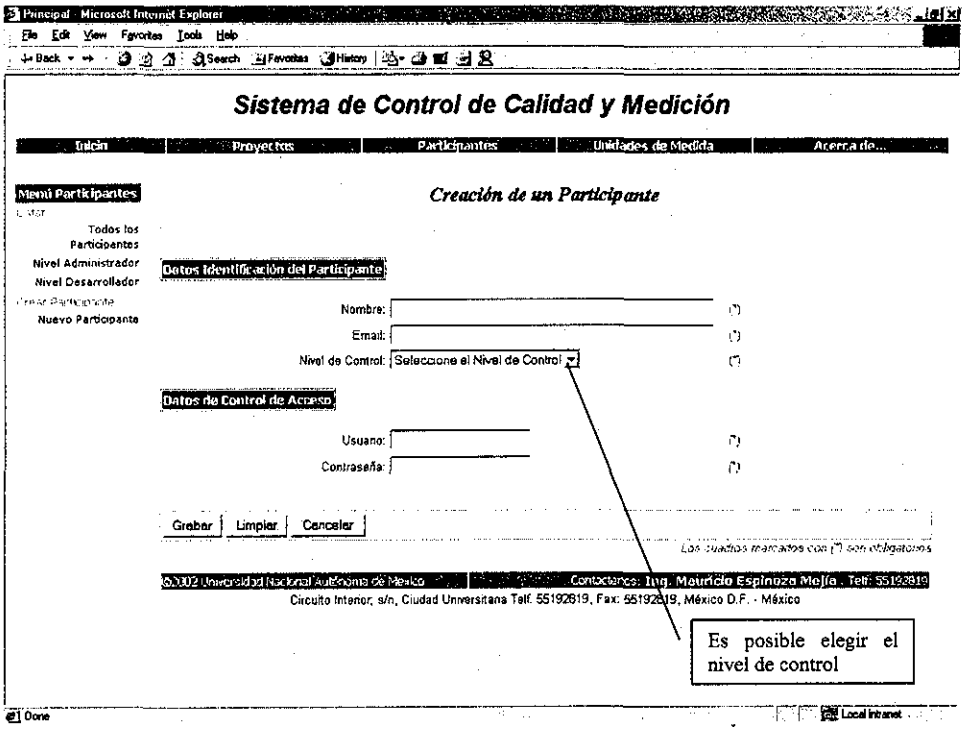


Figura 8. Creación de participantes

Edición de los Datos de un Participante

Para editar los datos de un participante es necesario ser Administrador. Para acceder a esta opción se debe dar un clic en ✎ (ver figura 7). Si esta opción no esta disponible quiere decir que el nivel de control del participante actual no es de Administrador.

Al igual que cuando se crea un participante, todos los cuadros de texto marcados con (*) son obligatorios, en caso de no ingresar la información suficiente se obtendrá un cuadro de mensaje como el de la figura 5.

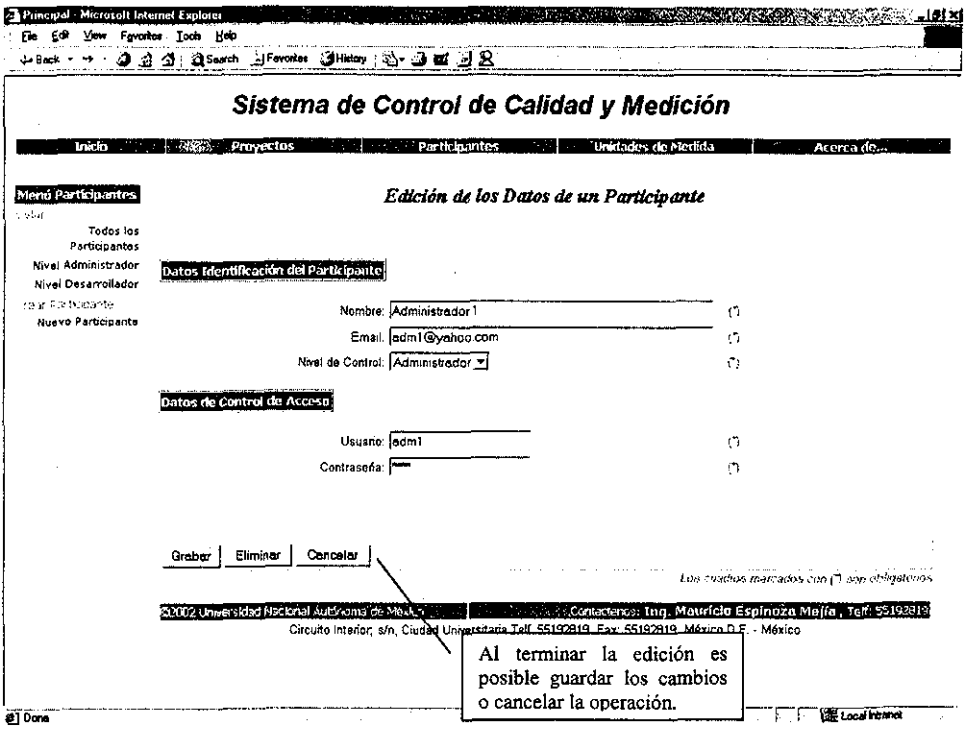


Figura 9. Edición de los datos de un participante

Administración de Unidades de Medida

En la pagina de la figura 10 se pueden observar los datos generales de las unidades de medida. Los datos de una unidad de medida pueden ser editados si es Administrador o visualizadas si el participante es desarrollador.

Las unidades de medida creadas mediante esta opción, servirán como requisito de entrada al momento de crear un producto.

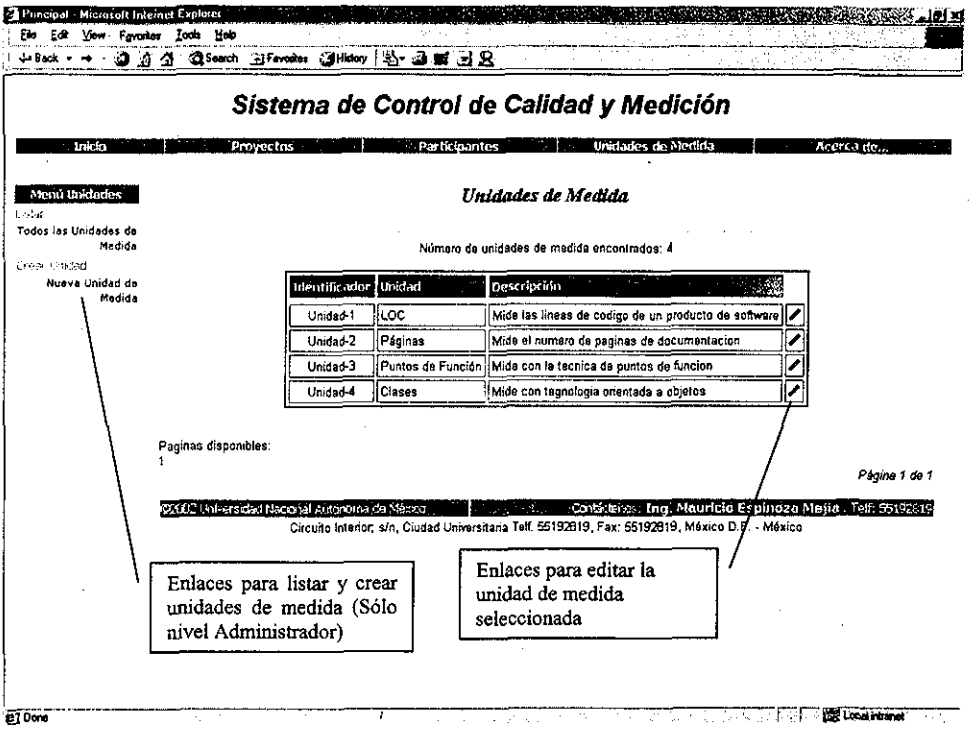


Figura 10. Administración de unidades de medida

Crear Unidades de Medida

En la pagina de la figura 11 se puede ingresar la información necesaria para crear unidades de medida. Esta opción es disponible únicamente si es Administrador. Hay que notar que los campos de texto marcados con (*) son obligatorios y en caso de no ser ingresados se mostrara un mensaje de aviso como se observa en la figura 5.

Las unidades de medida creadas con esta opción estarán disponibles al momento de crear un producto.

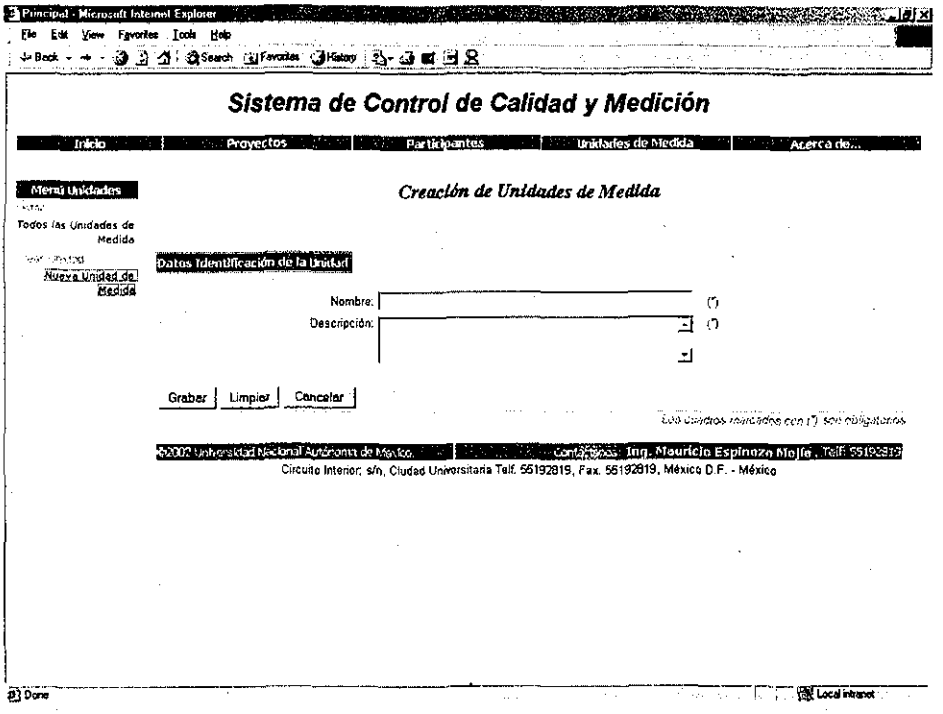


Figura 11. Creación de una unidad de medida

Edición de los Datos de una Unidad de Medida

Para editar los datos de una unidad de medida es necesario ser Administrador. Para acceder a esta opción se debe dar un clic en ✎ (ver figura 10). Si esta opción no esta disponible quiere decir que el nivel de control del participante actual no es de Administrador.

Al igual que cuando se crea un participante, todos los cuadros de texto marcados con (*) son obligatorios, en caso de no ingresar la información suficiente se obtendrá un cuadro de mensaje como el de la figura 5.

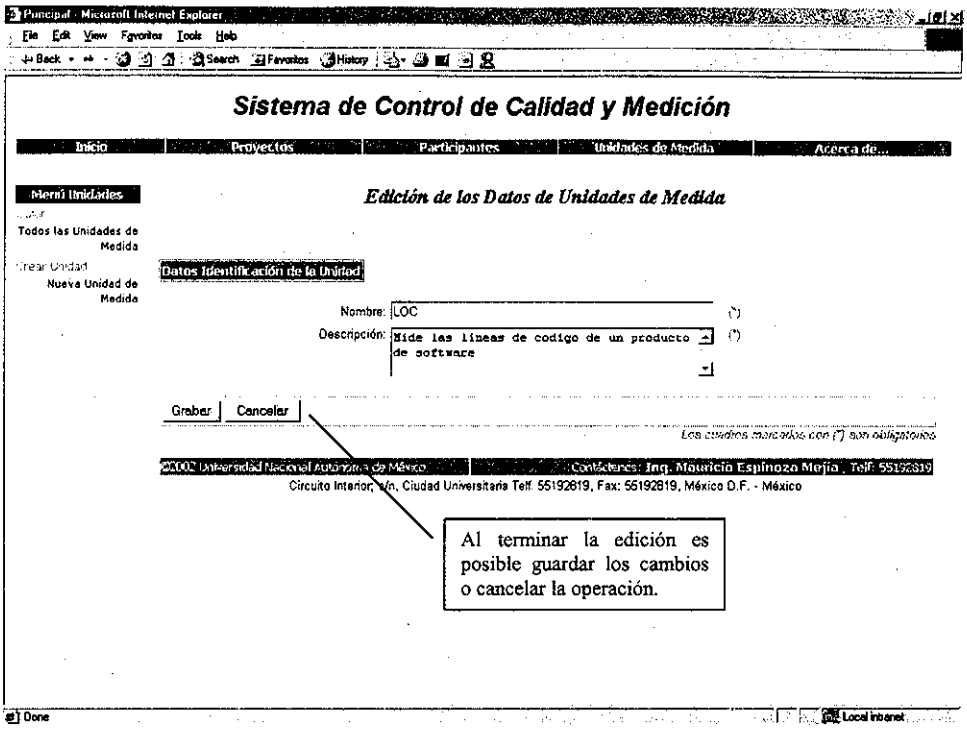


Figura 12. Edición de los datos de una unidad de medida

Asignar nuevos roles

Mediante esta opción es posible asignar nuevos roles para un producto seleccionado (productor, revisor o aprobador). Esta opción es disponible únicamente si es Administrador. Para efectuar esta operación es necesario dar un clic en ⚒ (ver figura 3) del proyecto que contiene al producto que se desea alterar.

Una vez dentro del proyecto se debe elegir el producto requerido y presionar la imagen (ver figura 13) localizada a la derecha de los datos del producto. Finalmente se debe ingresar la información requerida como se observa en la pagina de la figura 14.

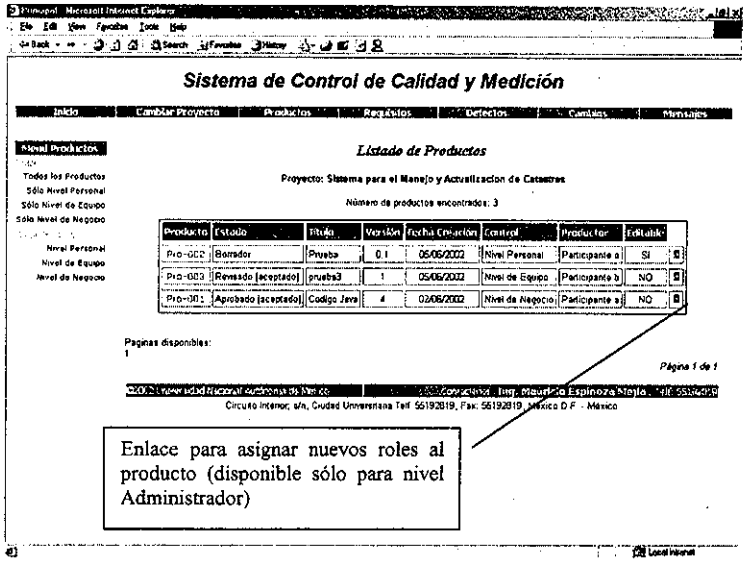


Figura 13. Elección del producto a asignar nuevos roles

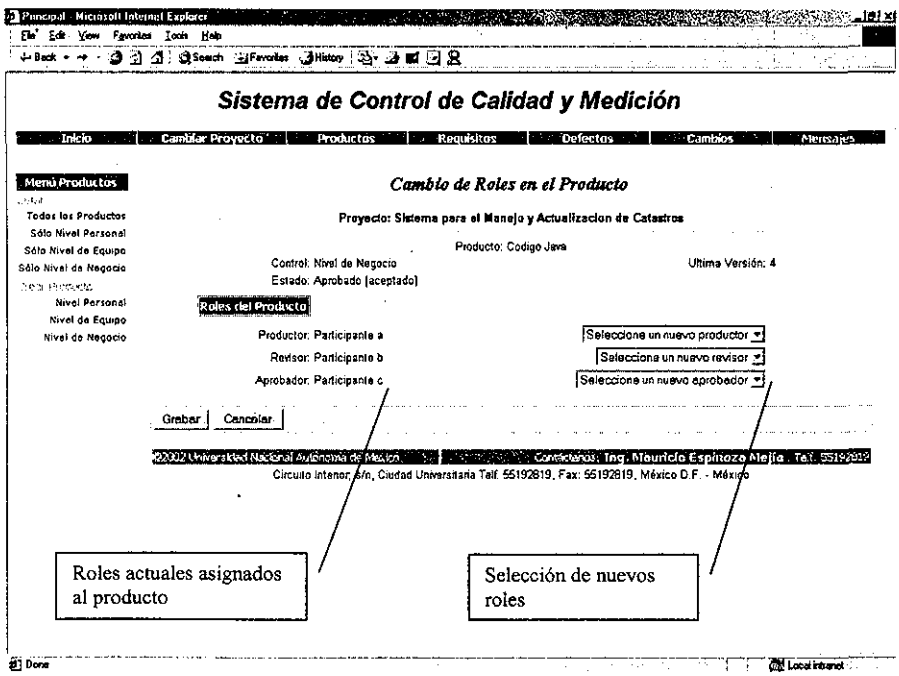


Figura 14. Asignación de nuevos roles al producto

Anexo D: Interfaz de Usuario del Nivel de Control Personal

Información de Productos

Para acceder al listado de productos es necesario seleccionar el proyecto que contiene los productos con los que se desea trabajar (ver figura 3, Anexo C).

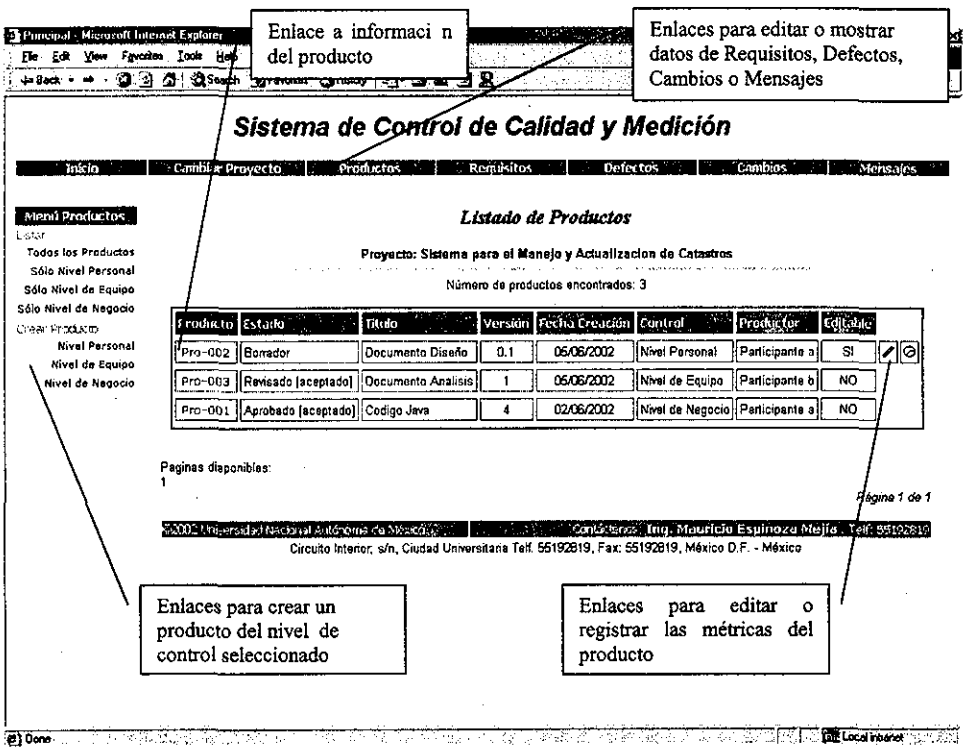


Figura 1. Listado de Productos

✎: Permite editar los datos de un producto, esta opción esta disponible únicamente si:

el participante es el Productor y el producto es editable,

el participante es el Revisor y el producto es no editable en estado de borrador o,

si el participante es el Aprobador y el producto es no editable y ha sido revisado.

⊘: Permite ingresar los datos de las métricas como lo son el tamaño y tiempo dedicado a una tarea en particular.

Crear Producto Nivel Personal

En la pagina de la figura 2 se puede ingresar la información necesaria para crear un producto de nivel personal. Hay que notar que los campos de texto marcados con (*) son obligatorios y en caso de no ser ingresados se mostrara un mensaje de aviso como se observa en la figura 5, Anexo C.

Para acceder a esta opción se debe dar un clic en el enlace respectivo, situado en la parte inferior izquierda de la pantalla de la figura 1.

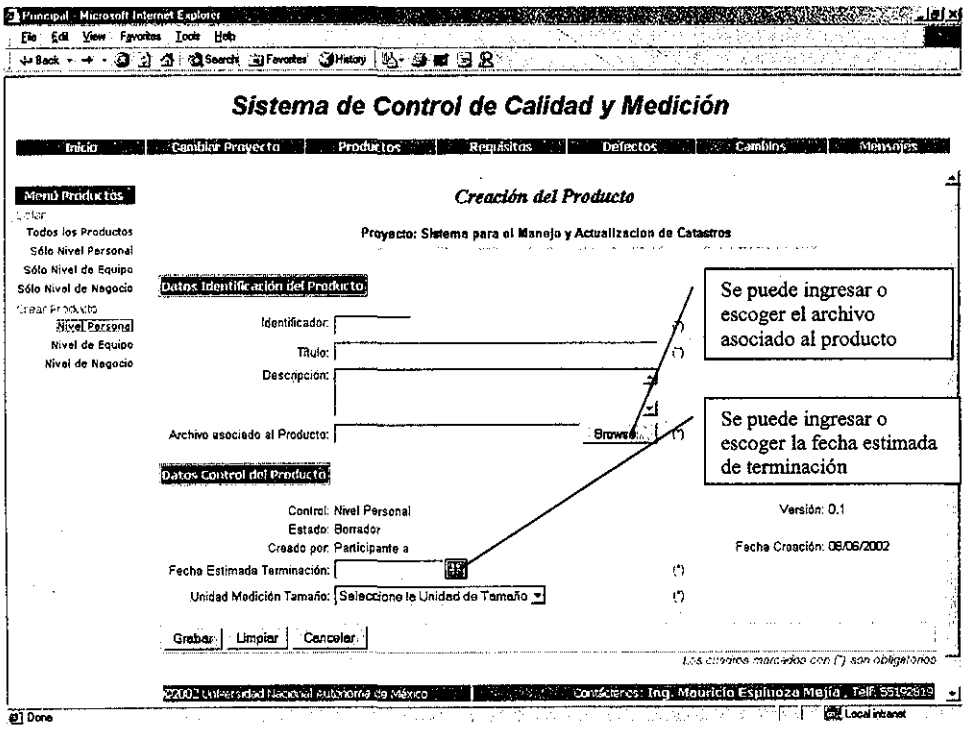


Figura 2. Creación de un producto de nivel personal

Edición de los Datos de un Producto de Nivel Personal

Para editar los datos de un producto de Nivel Personal es necesario ser el productor. Para acceder a esta opción se debe dar un clic en ✎ (ver figura 1). Si esta opción no esta disponible quiere decir que el participante actual no es el Productor.

Al igual que cuando se crea el producto, todos los cuadros de texto marcados con (*) son obligatorios, en caso de no ingresar la información suficiente se obtendrá un cuadro de mensaje como el de la figura 5, Anexo C.

Cuando el productor así lo decida es posible cambiar el estado del producto a terminado o crear una nueva versión del producto. Si se desea cambiar el producto a terminado se debe enviar el producto al repositorio dando clic en FTP.

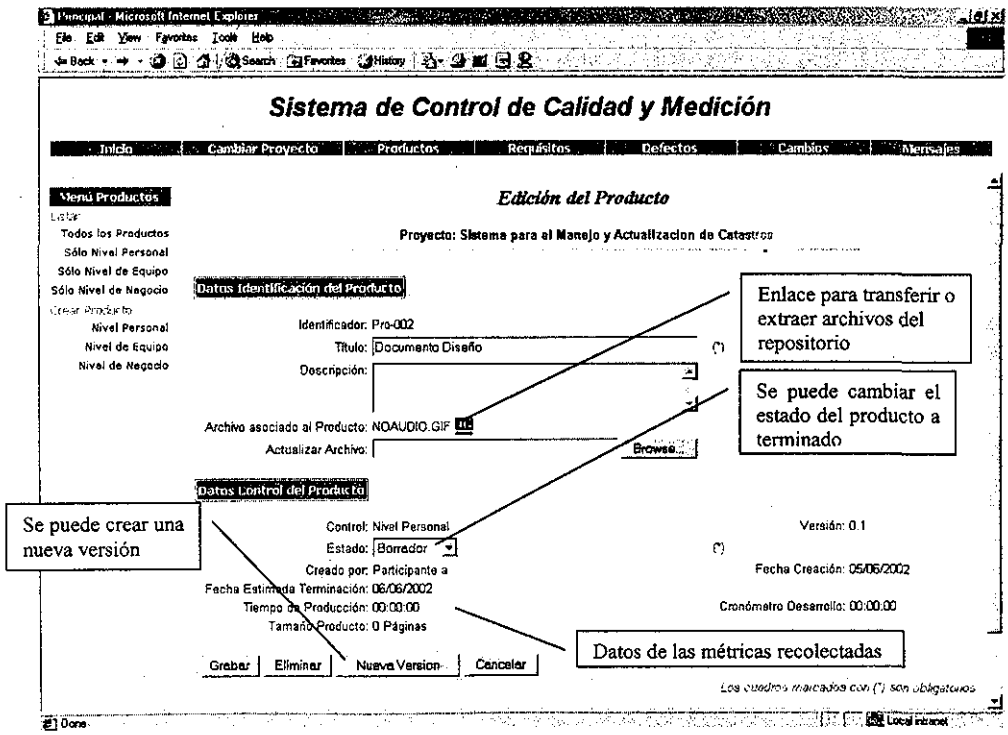


Figura 3. Edición de un producto de nivel personal

Mostrar Información de un Producto de Nivel Personal

Para visualizar los datos de un producto de Nivel Personal se debe dar un clic en el identificador del producto (ver figura 1). Esta opción esta disponible para todos los participantes.

Con las métricas recolectadas se puede obtener información de la estabilidad de tamaño, esfuerzo acumulado, productividad o el cumplimiento del cronograma de trabajo.

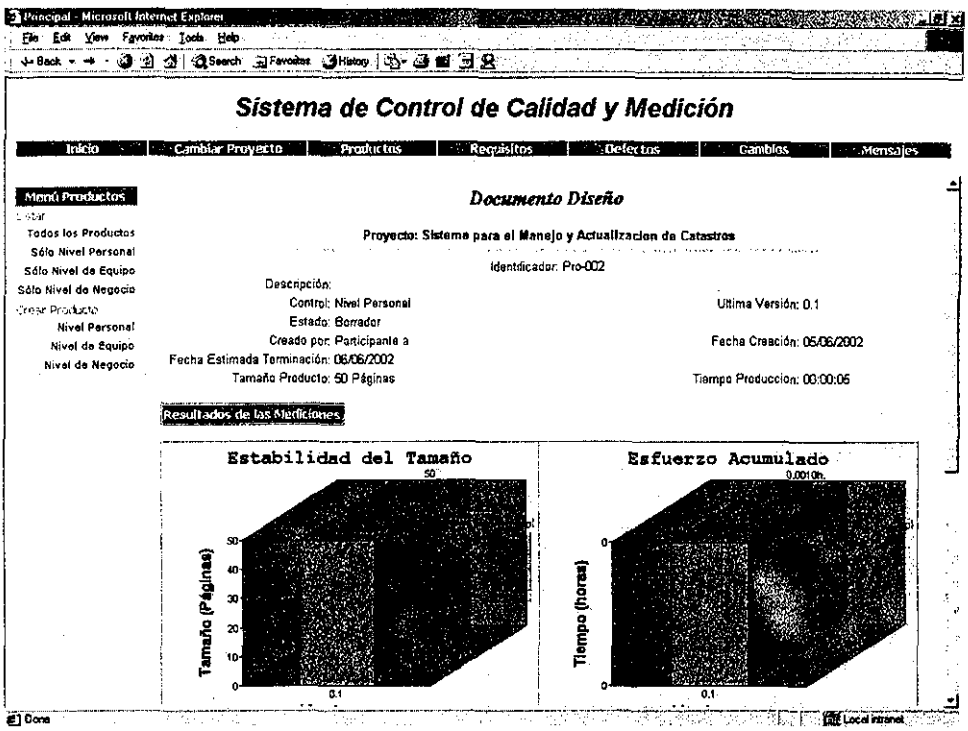


Figura 4. Información de un producto de nivel personal

Registrar métricas de un Producto de Nivel Personal

Para registrar las métricas de un producto de Nivel Personal se debe dar un clic en ⊘ (ver figura 1). Esta opción esta disponible unicamente si es el Productor.

Con las métricas recolectadas se puede obtener información como se observa en la figura 4.

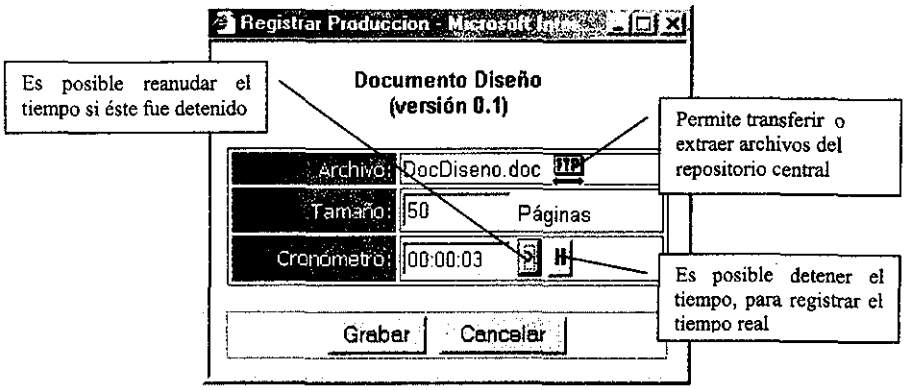


Figura 5. Registro de las métricas de un producto de nivel personal

Transferir archivos o extraerlos del repositorio central

Cuando el producto cambia de estado a terminado o cuando el revisor o aprobador necesita el archivo asociado al producto para revisarlo o aprobarlo, se hace necesario enviarlo o recuperarlo del repositorio central, para lo cual es necesario dar un clic en FTP (ver figura 3). Enseguida aparecerá una pantalla en donde se debe registrar el nombre de usuario y la contraseña para acceder al repositorio central.

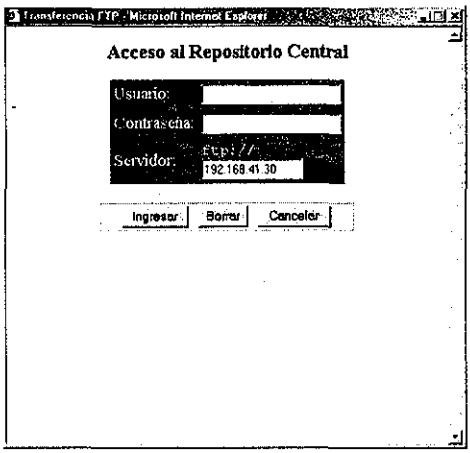


Figura 6. Conexión vía FTP al repositorio central

Copiar archivos o extraerlos del repositorio central

Una vez especificados el nombre de usuario y contraseña validos se puede copiar al repositorio o extraer del mismo el archivo necesario. Se puede efectuar esta operación dando un clic con el botón derecho del mouse y seleccionando la operación que se requiere del submenú que aparece.

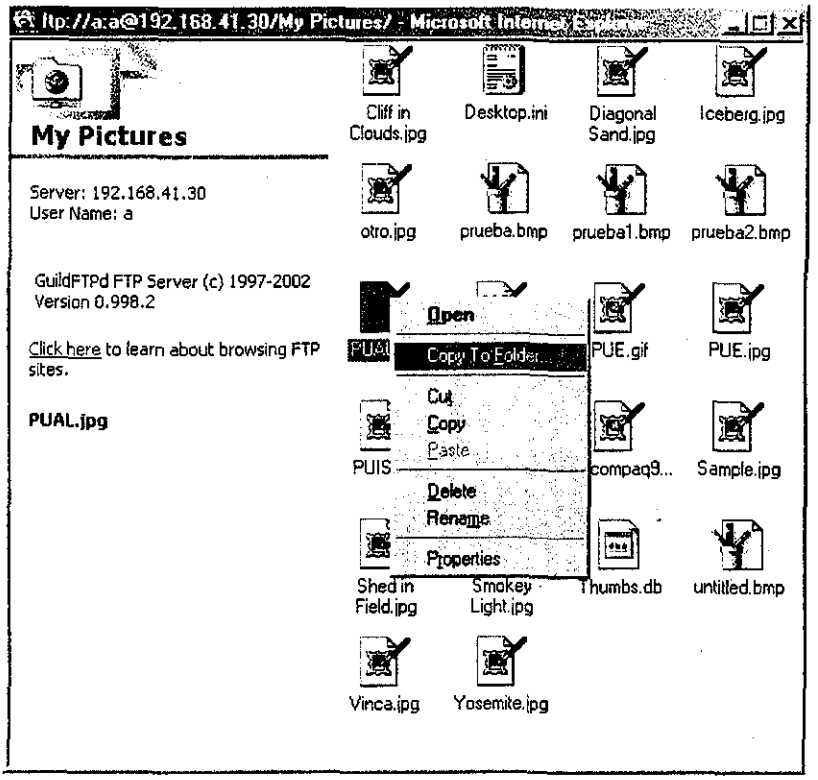


Figura 7. Copia de archivos al repositorio central

Anexo E: Interfaz de Usuario del Nivel de Control de Equipo

En este nivel se incluyen las interfaces que difieren del Nivel de Control Personal, pues se adicionan los casos de uso que implican la revisión del producto por parte de un Revisor.

Además, se agregan otras pantallas debido a que en este nivel de control se pretende controlar los requisitos del producto, los defectos encontrados al mismo debido a una revisión y además los cambios solicitados a un producto.

Crear Producto Nivel de Equipo

Para crear un producto con este nivel de control se siguen los mismos pasos que el nivel anterior. (ver figura1, Anexo D).

En la pagina de la figura 1 se puede ingresar la información necesaria para crear un producto de nivel de equipo. Hay que notar que los campos de texto marcados con (*) son obligatorios y en caso de no ser ingresados se mostrara un mensaje de aviso como se observa en la figura 5, Anexo C. Antes de la creación de un producto con este nivel de control deben haberse creado los requisitos (ver figura 5).

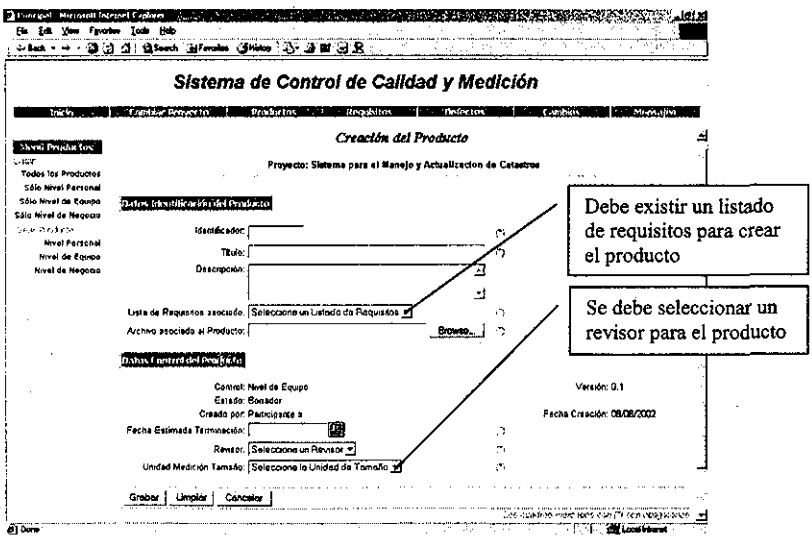


Figura 1. Creación de un producto de nivel de equipo

Edición de los Datos de un Producto de Nivel de Equipo

Para editar los datos de un producto de Nivel de Equipo es necesario ser el Productor o Revisor asignado. Para acceder a esta opción se debe dar un clic en el Listado de Productos ✎ (ver figura 1, Anexo D). Si esta opción no esta disponible quiere decir que el participante actual no es el Productor ni Revisor.

Al igual que cuando se crea el producto, todos los cuadros de texto marcados con (*) son obligatorios, en caso de no ingresar la información suficiente se obtendrá un cuadro de mensaje como el de la figura 5, Anexo C.

Cuando el productor así lo decida es posible enviar el producto a evaluación o crear una nueva versión del producto. En cualquier momento es posible transferir archivos o extraerlos del repositorio dando clic en FTP.

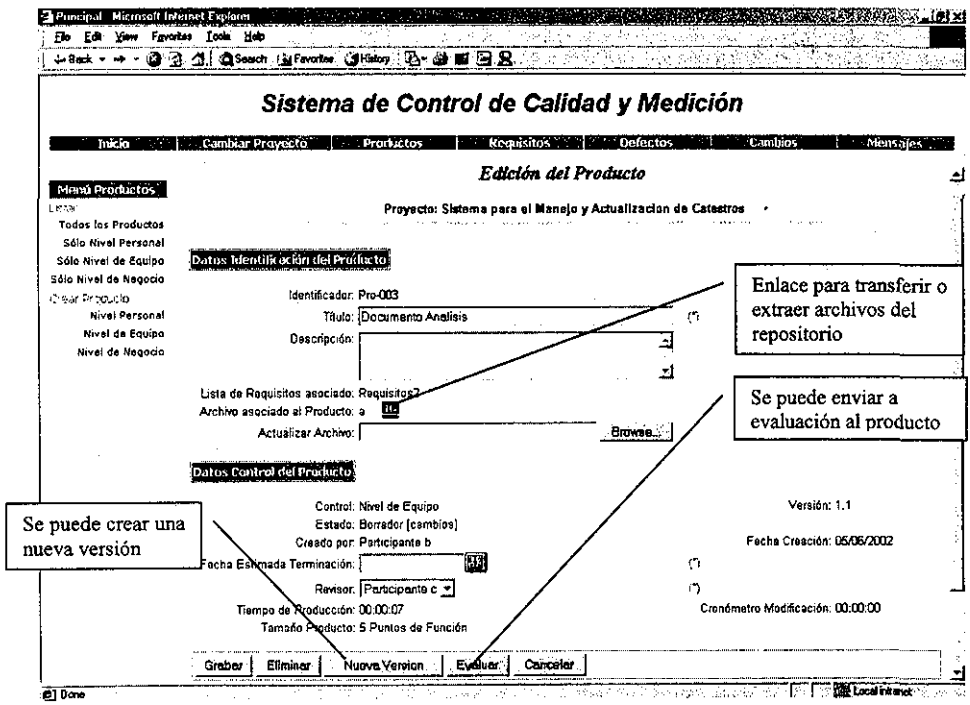


Figura 2. Edición de un producto de nivel de equipo

Mostrar Información de un Producto de Nivel de Equipo

Para visualizar los datos de un producto de Nivel de Equipo se debe dar un clic en el identificador del producto (ver figura 1, Anexo D). Esta opción esta disponible para todos los participantes.

Con las métricas recolectadas se puede obtener información de la estabilidad de tamaño, esfuerzo acumulado, productividad, calidad del producto, porcentaje del esfuerzo total de producción, estabilidad del producto o el cumplimiento del cronograma de trabajo.

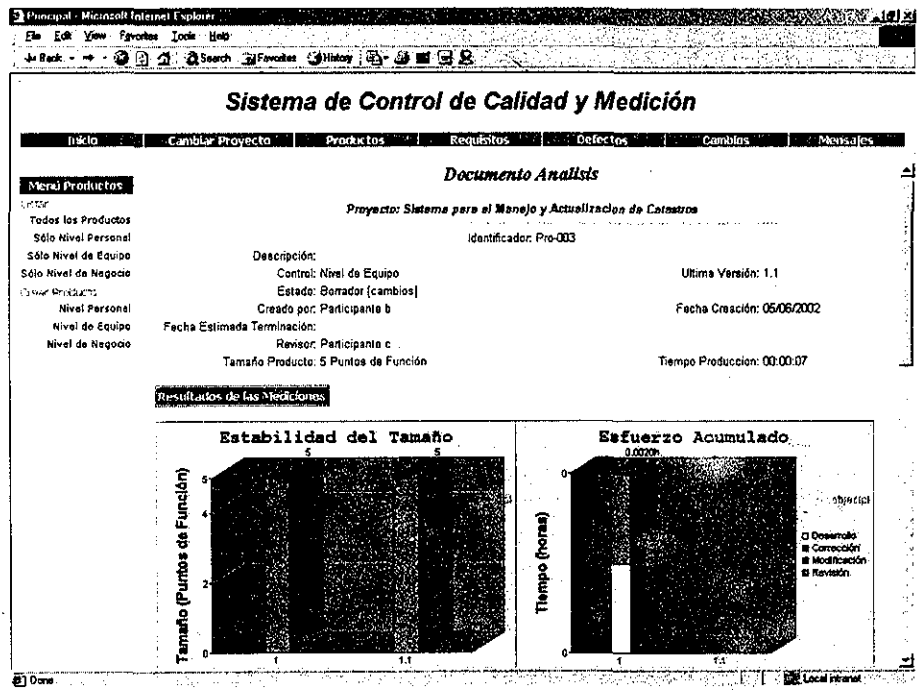


Figura 3. Información de un producto de nivel de equipo

Registrar métricas de un Producto de Nivel de Equipo

Para registrar las métricas de un producto de Nivel de Equipo se debe seguir los mismos pasos que en el nivel anterior. A diferencia de los productos con Nivel Personal, esta opción esta disponible al Productor y Revisor asignado. Con las métricas recolectadas se puede obtener información como se observa parcialmente en la figura 3.

Información de Requisitos

Para acceder al listado de requisitos se debe escoger el enlace Requisitos como se observa en la figura 4.

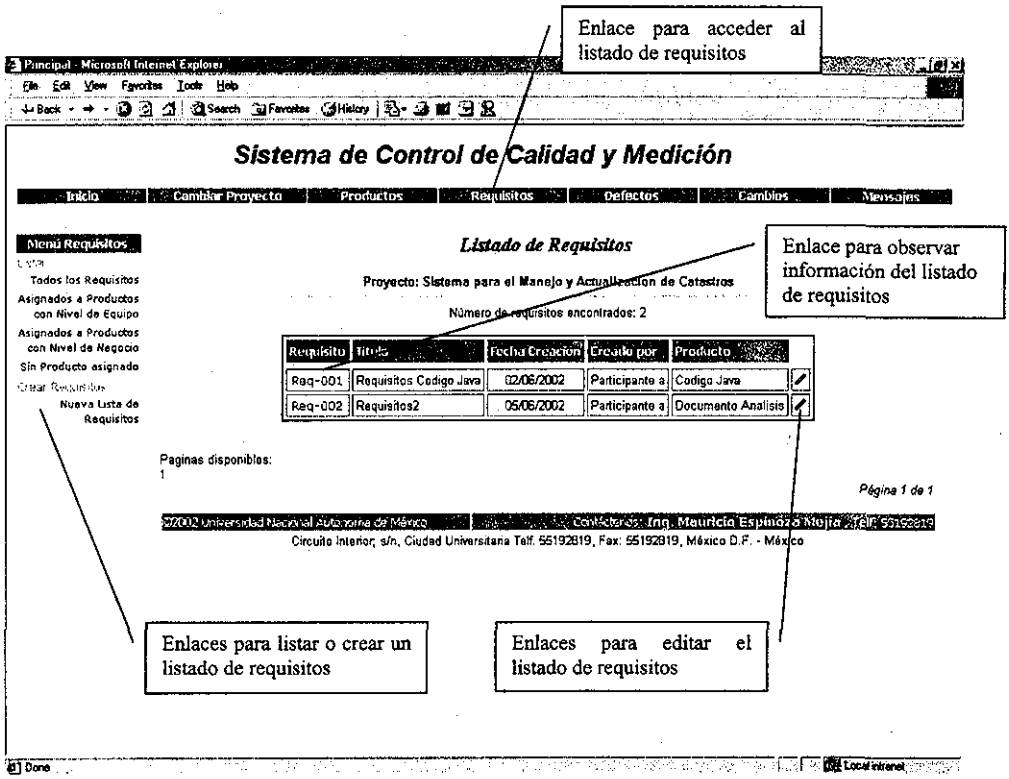


Figura 4. Listado de Requisitos

Crear Listado de Requisitos

En la pagina de la figura 5 se puede ingresar la información necesaria para crear un documento de requisitos. Hay que notar que los campos de texto marcados con (*) son obligatorios y en caso de no ser ingresados se mostrara un mensaje de aviso como se observa en la figura 5, Anexo C.

Para acceder a esta opción se debe dar un clic en el enlace respectivo, situado en la parte inferior izquierda del menú Requisitos (ver figura 4).

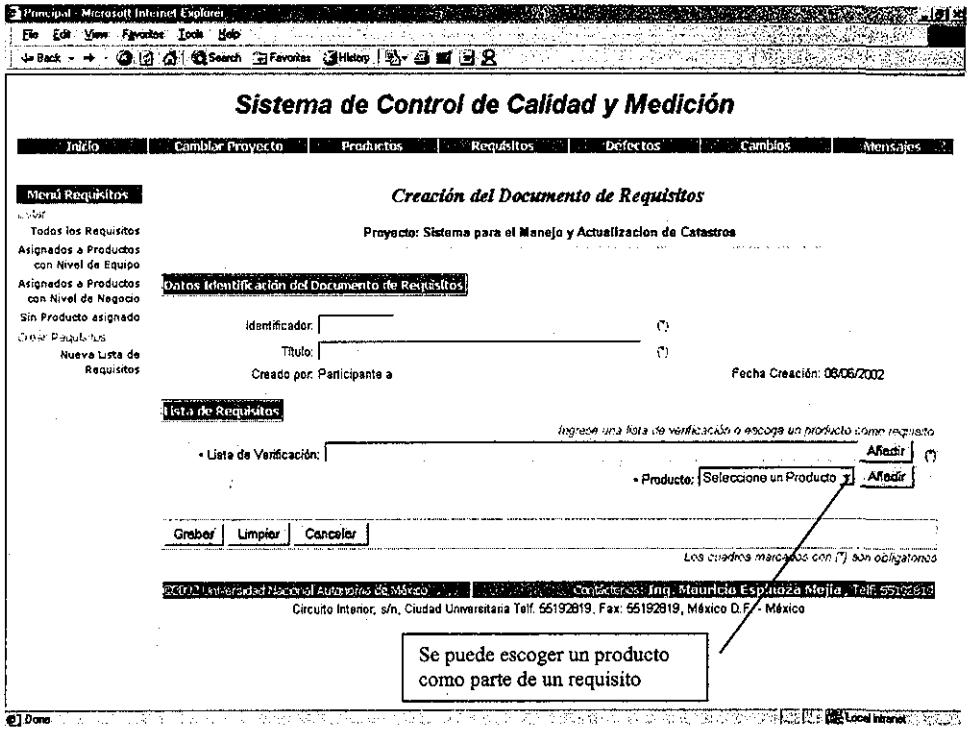


Figura 5. Creación de un Listado de Requisitos

Edición de un Listado de Requisitos

Para editar los datos de un Listado de Requisitos es necesario ser el Productor del documento. Para acceder a esta opción se debe dar un clic en ✎ (ver figura 4). Si esta opción no esta disponible quiere decir que el participante actual no es el Productor del documento.

Al igual que cuando se crea el listado de requisitos, todos los cuadros de texto marcados con (*) son obligatorios, en caso de no ingresar la información suficiente se obtendrá un cuadro de mensaje como el de la figura 5, Anexo C.

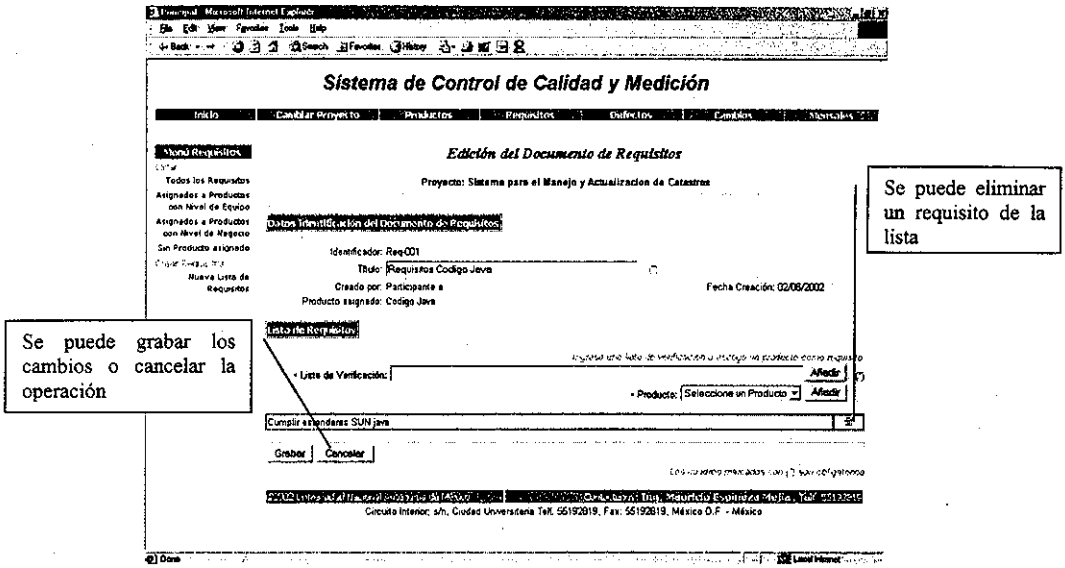


Figura 6. Edición de un Listado de Requisitos

Mostrar Información de un Listado de Requisitos

Para visualizar los datos de un Listado de Requisitos se debe dar un clic en el identificador del requisito (ver figura 4). Esta opción esta disponible para todos los participantes.

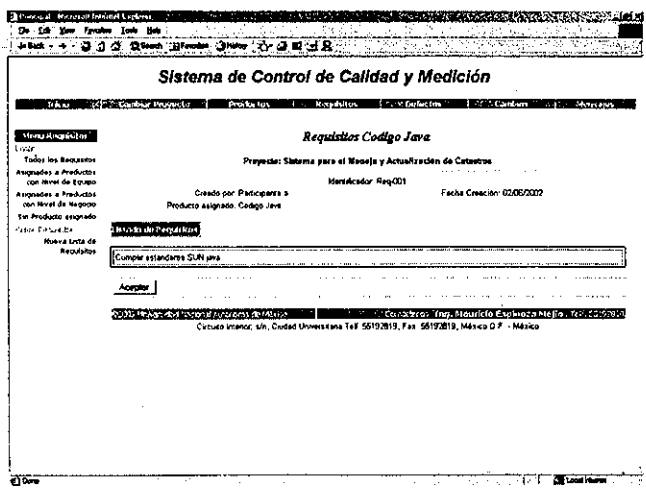


Figura 7. Información de un Listado de Requisitos

Información de Defectos

Para acceder al listado de defectos se debe escoger el enlace Defectos como se observa en la figura 8.

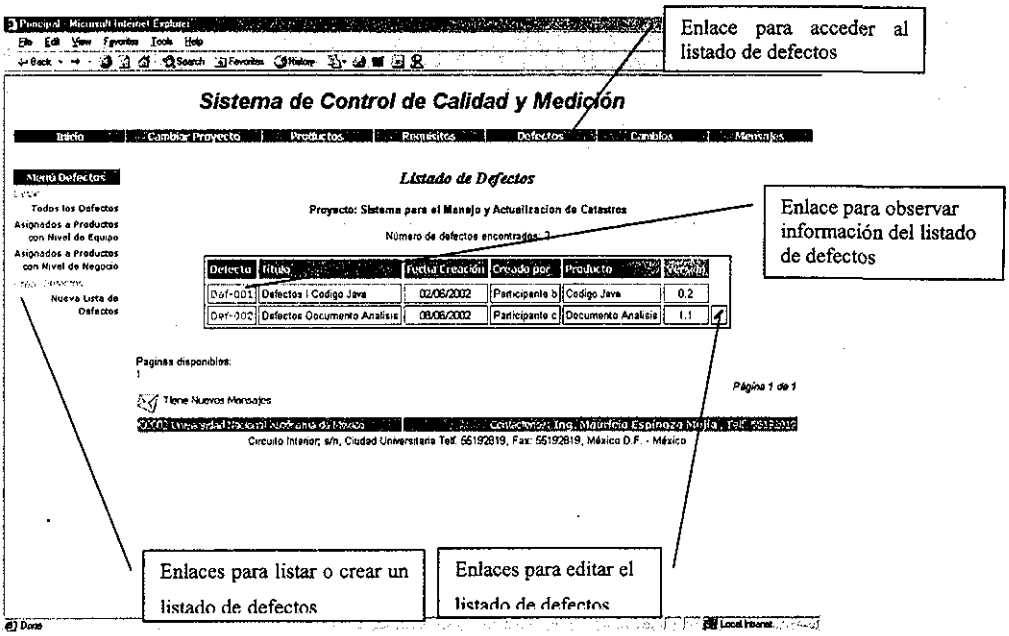


Figura 8. Listado de Defectos

Crear Listado de Defectos

En la pagina de la figura 9 se puede ingresar la información necesaria para crear un documento de defectos. Hay que notar que los campos de texto marcados con (*) son obligatorios y en caso de no ser ingresados se mostrara un mensaje de aviso como se observa en la figura 5, Anexo C.

Para acceder a esta opción se debe dar un clic en el enlace respectivo, situado en la parte inferior izquierda del menú Defectos (ver figura 8).

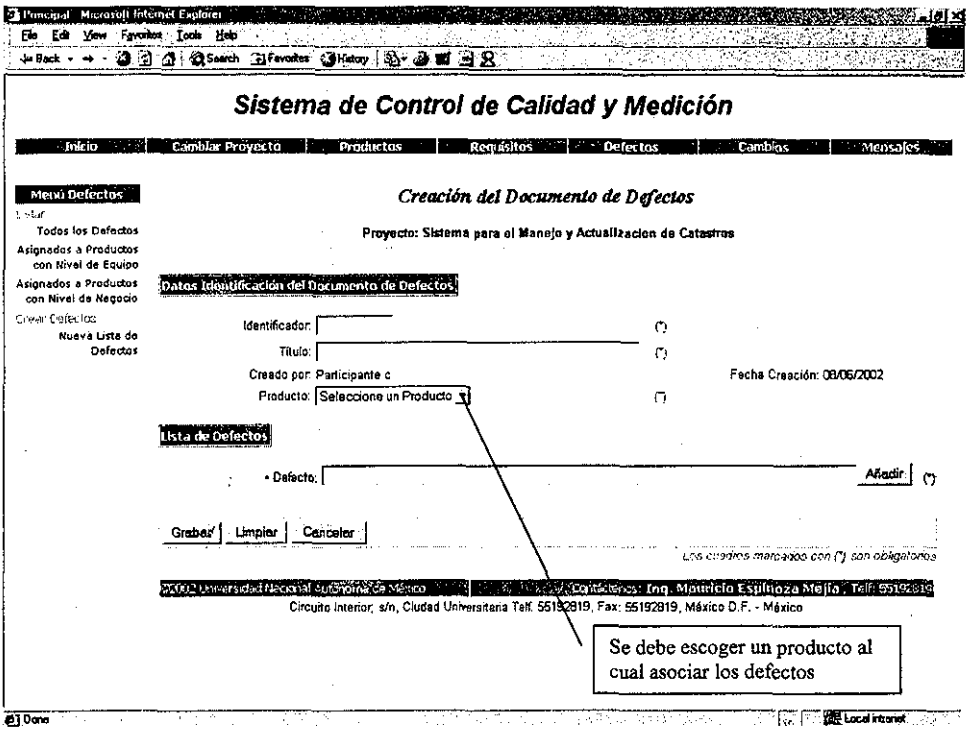


Figura 9. Creación de un Listado de Defectos

Edición de un Listado de Defectos

Para editar los datos de un Listado de Defectos es necesario ser el Productor del documento. Para acceder a esta opción se debe dar un clic en ✎ (ver figura 8). Si esta opción no esta disponible quiere decir que el participante actual no es el Productor del documento.

Al igual que cuando se crea el listado de defectos, todos los cuadros de texto marcados con (*) son obligatorios, en caso de no ingresar la información suficiente se obtendrá un cuadro de mensaje como el de la figura 5, Anexo C.

Cuando el productor del documento termine de ingresar todos los defectos localizados puede rechazar el producto.

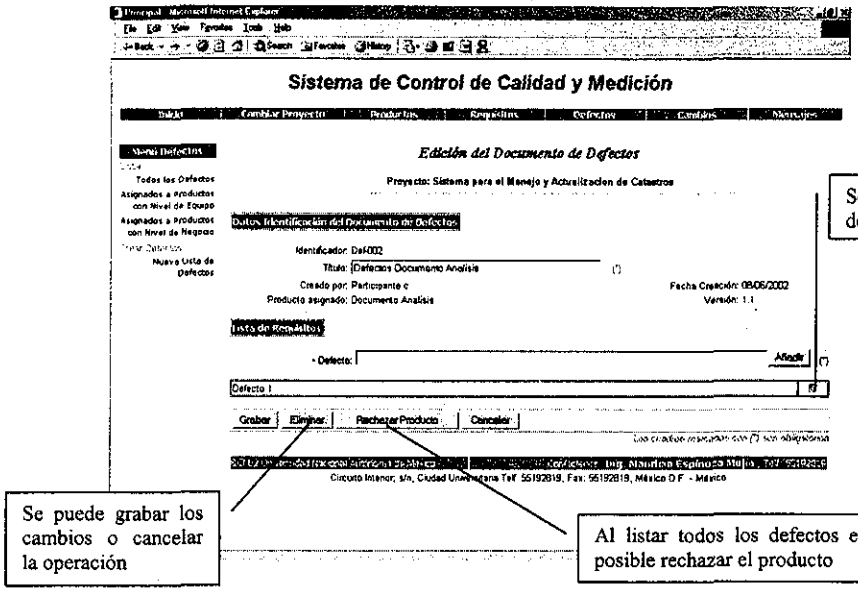


Figura 10. Edición de un Listado de Defectos

Mostrar Información de un Listado de Defectos

Para visualizar los datos de un Listado de Defectos se debe dar un clic en el identificador del defecto (ver figura 8). Esta opción esta disponible para todos los participantes.

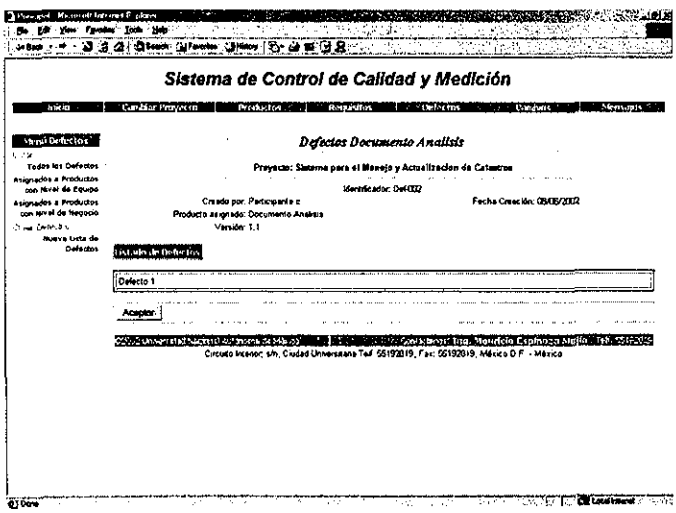


Figura 11. Información de un Listado de Defectos

Información de Solicitudes de Cambio

Para acceder al listado de solicitudes de cambios se debe escoger el enlace Cambios como se observa en la figura 12.

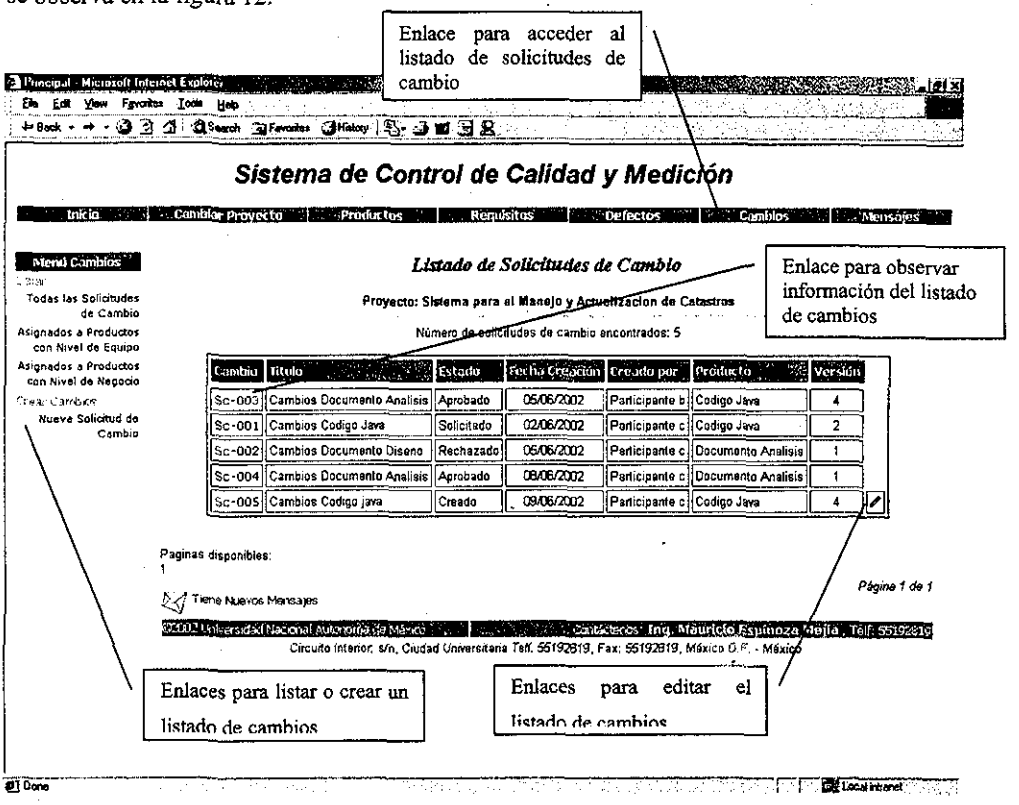


Figura 12. Listado de Solicitudes de Cambio

Crear Listado de Solicitud de Cambio

En la pagina de la figura 13 se puede ingresar la información necesaria para crear un documento de solicitud de cambio. Hay que notar que los campos de texto marcados con (*) son obligatorios y en caso de no ser ingresados se mostrara un mensaje de aviso como se observa en la figura 5, Anexo C.

Para acceder a esta opción se debe dar un clic en el enlace respectivo, situado en la parte inferior izquierda del menú Cambios (ver figura 12).

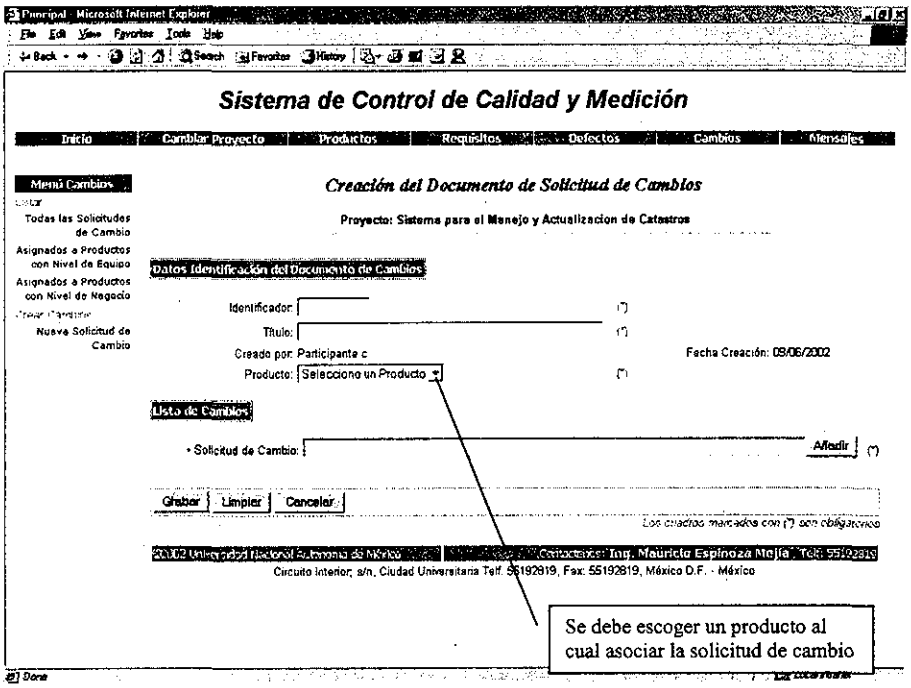


Figura 13. Creación de un Documento de Solicitud de Cambio

Edición de un Listado de Cambios

Para editar los datos de un Listado de Cambios es necesario ser el Productor del documento. Para acceder a esta opción se debe dar un clic en ✎ (ver figura 12). Si esta opción no esta disponible quiere decir que el participante actual no es el Productor del documento.

Al igual que cuando se crea el listado de cambios, todos los cuadros de texto marcados con (*) son obligatorios, en caso de no ingresar la información suficiente se obtendrá un cuadro de mensaje como el de la figura 5, Anexo C.

Cuando el productor del documento termine de ingresar todos los cambios que se requieren efectuar, se puede solicitar el cambio al producto.

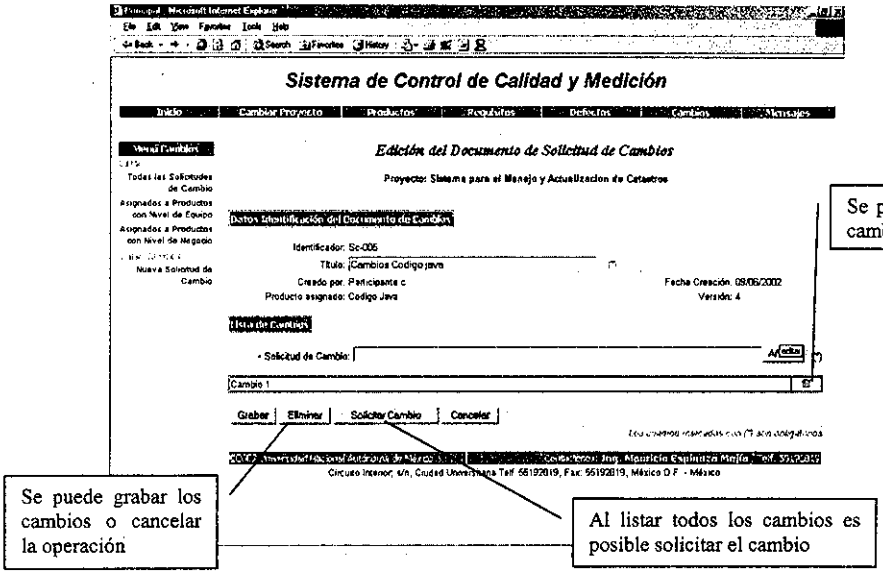


Figura 14. Edición de un Listado de Defectos

Mostrar Información de un Listado de Cambios

Para visualizar los datos de un Listado de Cambios se debe dar un clic en el identificador del cambio (ver figura 12). Esta opción esta disponible para todos los participantes.

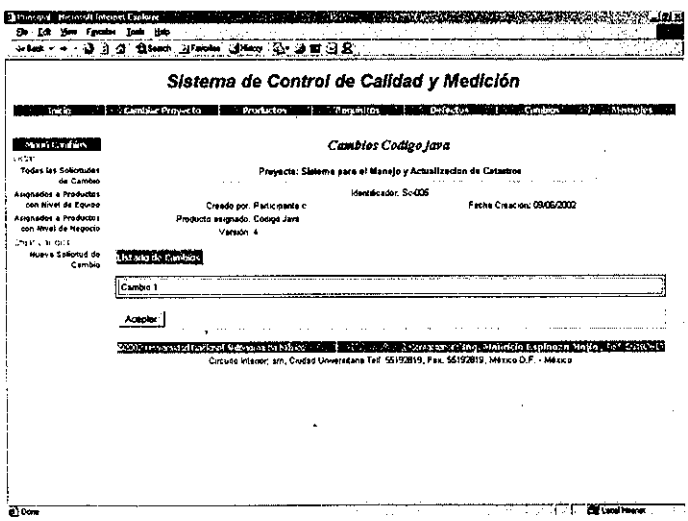


Figura 15. Información de un Listado de Cambios

Listado de Mensajes

Para acceder al listado de mensajes se debe escoger el enlace Mensajes como se observa en la figura 16.

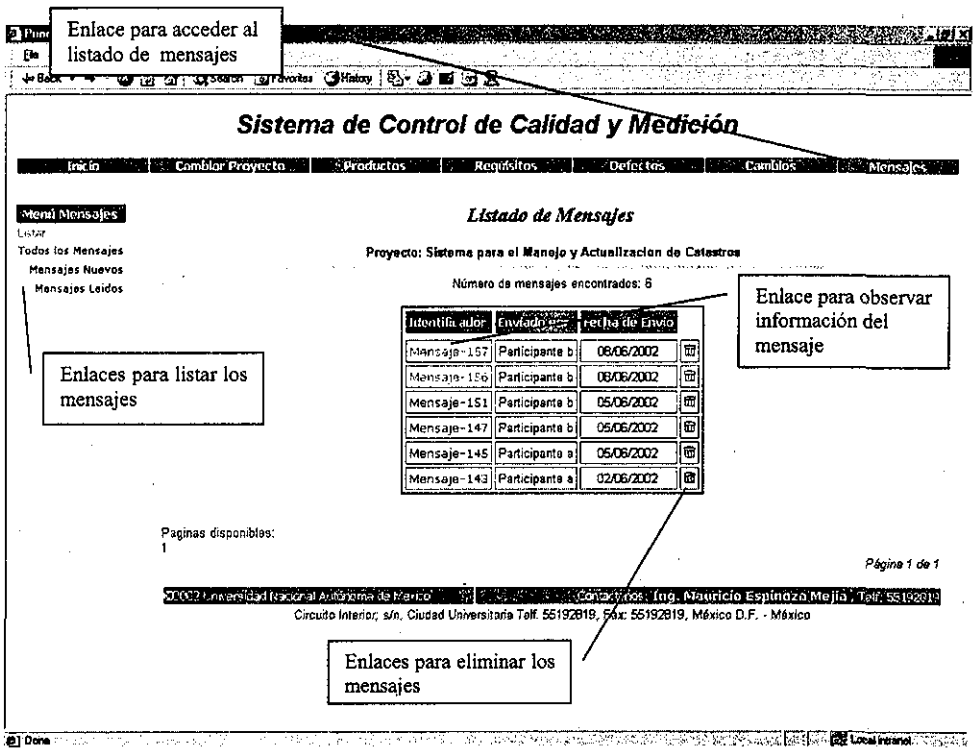


Figura 16. Listado de Mensajes

Mostrar Información de un Mensaje

Para visualizar los datos de un Mensaje se debe dar un clic en el identificador del mensaje (ver figura 16). Esta opción esta disponible para todos los participantes.

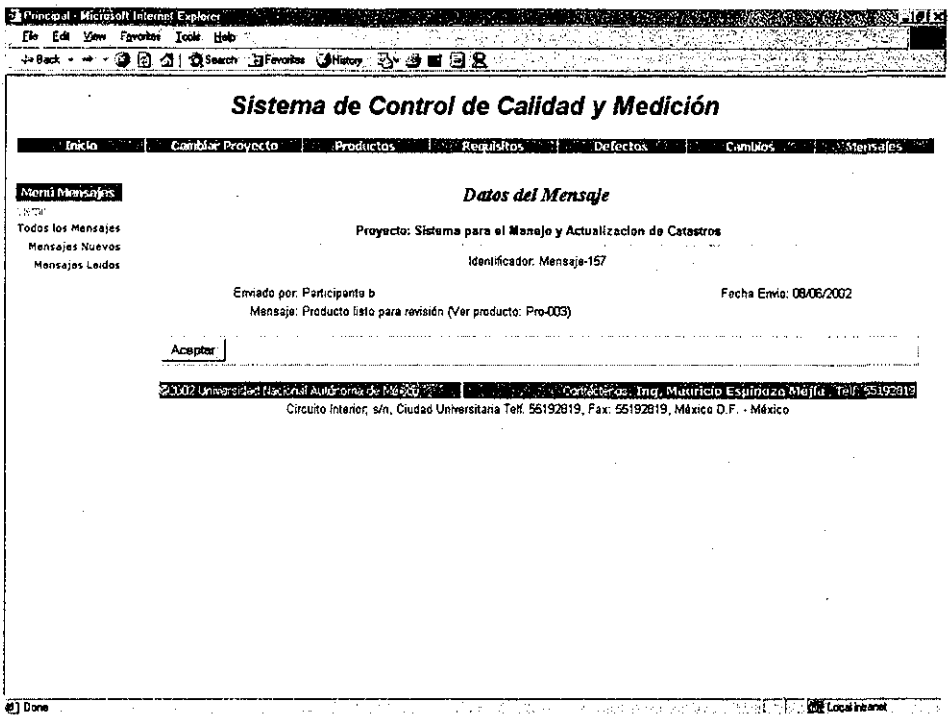


Figura 17. Información de un Mensaje

Anexo F: Interfaz De Usuario del Nivel de Control de Negocio

En este nivel se incluyen las interfaces que difieren del Nivel de Control Personal y de Equipo pues en este nivel existe un Aprobador y un Bibliotecario y se necesitan datos de control del Aprobador. En lo referente a las solicitudes de cambio, la evaluación es efectuada por el Aprobador, pero los datos son registrados en las mismas interfaces del nivel anterior.

Crear Producto Nivel de Negocio

Para crear un producto con este nivel de control se siguen los mismos pasos que el nivel anterior. (ver figura1, Anexo E).

En la pagina de la figura 1 se puede ingresar la información necesaria para crear un producto de nivel de negocio. Hay que notar que los campos de texto marcados con (*) son obligatorios y en caso de no ser ingresados se mostrara un mensaje de aviso como se observa en la figura 5, Anexo C. Antes de la creación de un producto con este nivel de control deben haberse creado los requisitos (ver figura 5, Anexo E).

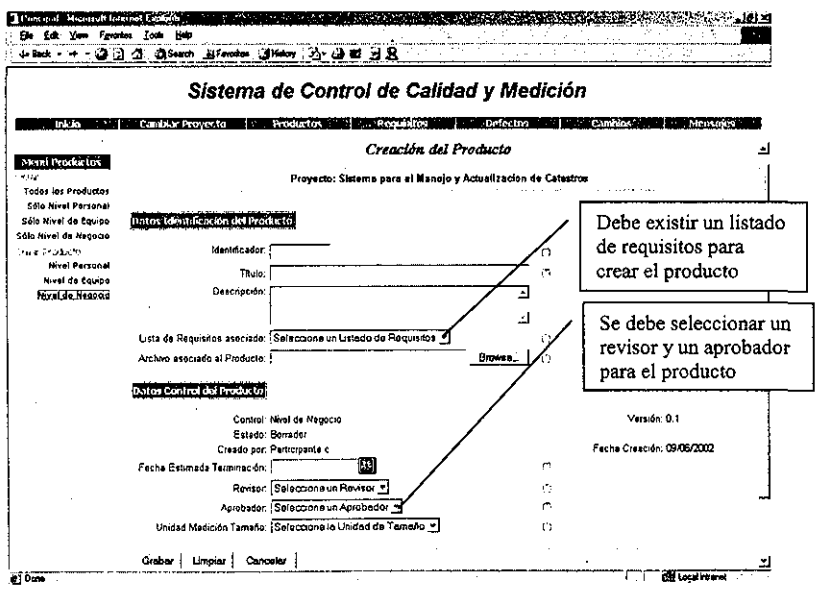


Figura 1. Creación de un producto de nivel de negocio

Edición de los Datos de un Producto de Nivel de Negocio

Para editar los datos de un producto de Nivel de Negocio es necesario ser el Productor, Revisor o Aprobador. Para acceder a esta opción se debe dar un clic en el Listado de Productos ✎ (ver figura 1, Anexo D). Si esta opción no esta disponible quiere decir que el participante actual no es Productor, ni Revisor, ni Aprobador.

Al igual que cuando se crea el producto, todos los cuadros de texto marcados con (*) son obligatorios, en caso de no ingresar la información suficiente se obtendrá un cuadro de mensaje como el de la figura 5, Anexo C.

Cuando el productor así lo decida es posible enviar el producto a evaluación o crear una nueva versión del producto.

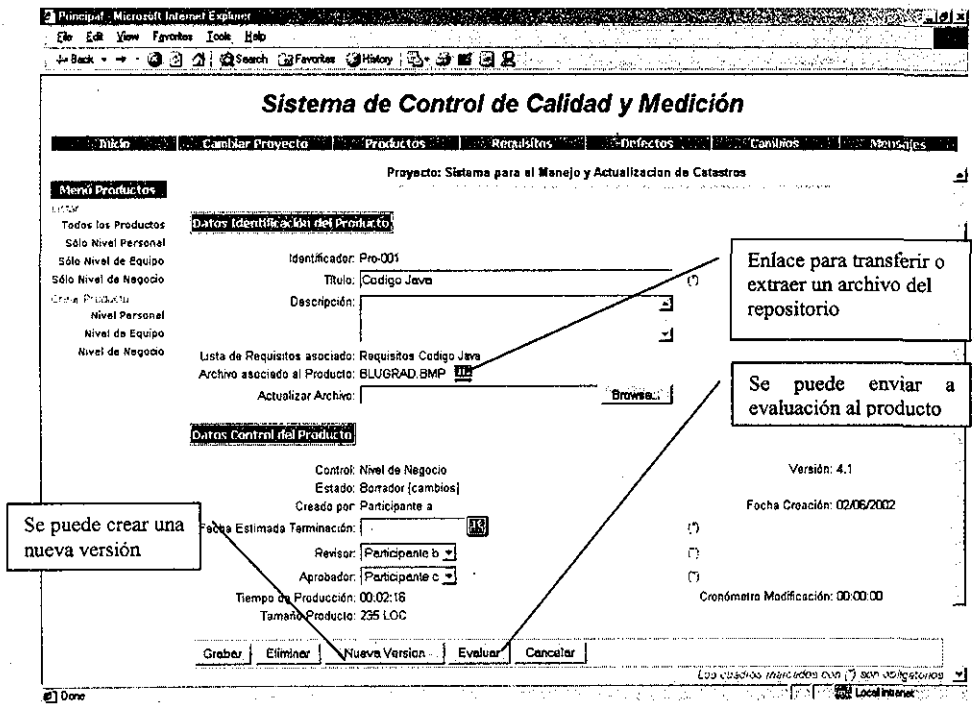


Figura 2. Edición de un producto de nivel de negocio

Mostrar Información de un Producto de Nivel de Negocio

Para visualizar los datos de un producto de Nivel de Negocio se debe dar un clic en el identificador del producto (ver figura 1, Anexo D). Esta opción esta disponible para todos los participantes.

Con las métricas recolectadas se puede obtener información de la estabilidad de tamaño, esfuerzo acumulado, productividad, calidad del producto, porcentaje del esfuerzo total de producción, estabilidad del producto o el cumplimiento del cronograma de trabajo. Además se dispone del tiempo dedicado exclusivamente a aprobación.

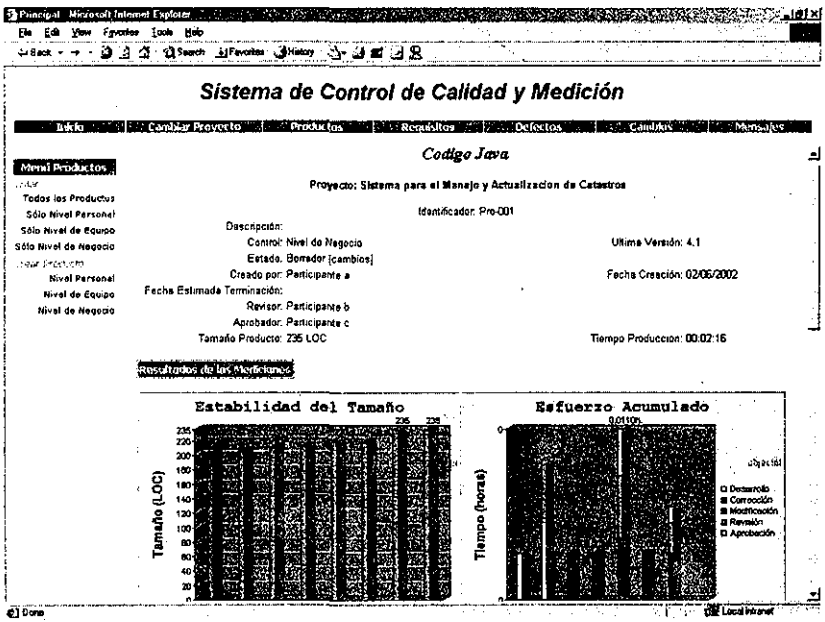


Figura 3. Información de un producto de nivel de negocio

Registrar métricas de un Producto de Nivel de Negocio

Para registrar las métricas de un producto de Nivel de Negocio se debe seguir los mismos pasos que en el nivel anterior. A diferencia de los productos con Nivel Personal y de Negocio, esta opción esta disponible al Productor, Revisor y Aprobador asignado. Con las métricas recolectadas se puede obtener información como se observa parcialmente en la figura 3.

Bibliografía

[Augustine97] Augustine, Thomas Capt.; Schroeder, Schroeder; An Effective Metrics Process Model; Colorado Technology University, Colorado Springs, USA; Technical Report; 1997.

[Basili84] Basili, V.R.; D.M. Weiss: A methodology for collecting valid software engineering data; IEEE Transactions on Software Engineering; Vol. SE-10, N.6; 1984.

[Boehm81] Boehm, B. W.: Software Engineering Economics; Englewood Cliffs, N. J.; Prentice-Hall; 1981.

[Boehm84] Boehm, B. W.: Software Engineering Economics; *IEEE Trans. Software Eng. SE-10*, 1; 1984.

[Booch99] Booch, G.; Rumbaugh, J.; Jacobson I.: The Unified Modeling Language User Guide; Addison-Wesley; 1999.

[CMM®] Paul, Mark C.; et Al.: The Capability Maturity Model, Guidelines for Improving the Software Process; SEI Series in Software Engineering; Adison-Wesley, EUA; 1995.

[CMMI] CMMI Model Components Derived from CMMI – SE/SW, V1.0 Process Areas; Software Engineering Institute, Carnegie Mellon; 2000.

[DeJesús01] De Jesús Jiménez, Martín; Oktaba, Hanna: Un prototipo para el control de calidad y cambios en los productos de software; Tesis de Maestría; Postgrado en Ciencias e Ingeniería de la Computación; UNAM, México 2001

[Fenton91] Fenton, E. Norman: Software Metrics A rigorous approach; Chapman & Hall, Primera Edición; 1991.

[Florac97] Florac, William A.; Park, Robert E.; Carleton, Anita D.; Practical Software Measurement: Measuring for Process Management and Improvement; Software Engineering Institute, Carnegie-Mellon University, USA; GUIDEBOOK CMU/SEI-97-HB-003; 1997.

[Ghezzi91] Ghezzi, C.; Jazayeri, M.; Mandrioli, D.: Fundamentals of Software Engineering; Prentice Hall; 1991.

[Grady87] Grady, Robert B.; Caswell, Deborah L.: Software Metrics: Establishing a Company-Wide Program; Englewood Cliffs, NJ: Prentice-Hall; 1987.

[Hamlet01] Hamlet, D., Maybee, J.: The Engineering of Software Technical Foundations for the Individual; Addison-Wesley; 2001.

[Humphrey89] Humphrey, Watts, S.: *Managing the Software Process*; Reading, MA; Addison-Wesley Publishing Company, Inc.; 1989.

[Humphrey95] Humphrey, W. S.: A Discipline for Software Engineering; Addison-Wesley; 790 p. ISBN 0-201-54610-8; 1995.

[ISO12207] International Standard ISO/IEC 12207: Information technology - Software life cycle processes; 1995.

[ISO15504] ISO/IEC 15504 Software Process Assessment, Technical Report; 1999

[ISO9126] ISO. Information Technology - Software Product Evaluation - Quality Characteristics and Guide Lines for their Use; Geneva, Switzerland. International Standards Organization, ISO/IEC IS 9126; 1991.

[Jacquet97] Jacquet, Jean-Philippe; Abran, Alain; From Software Metrics to Software Measurement Methods: A Process Model; Université du Québec à Montréal, Department of Computer Science, 1997.

[Jones00] Jones, Cheryl: Practical Software and Systems Measurements; Department of Defense of US Army; A foundation for Objective Project Management; 2000.

[McCabe76] McCabe, T. J.: A Complexity Measure; *Trans. Software Eng. SE*-2; 1976.

[McAndrews93] McAndrews, Donald R.: Establishing a Software Measurement Process; Software Engineering Institute, Carnegie-Mellon University, USA; Technical Report CMU/SEI-93-TR-16 ESC-TR-93-193; 1993.

[Morisio95] Morisio, Mauricio: A methodology to measure the software process, Torino, Italy; Dipartimento di Automatica e Informatica, Politecnico di Torino, 1995.

[Park92] Park, Robert E.: Software Size Measurement: An Architecture for Counting Source Statements (CMU/SEI-92-TR-20, ADA258304); Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University; September 1992.

[Paul99] Paul, S. Wang: Java With Object-Oriented and Word Wide Web Applications; EUA; 1999.

[Paulk95] Paulk, M., et al: The Capability Maturity Model, Guidelines for Improving the Software Proces; SEI Series in Software Engineering; Addison-Wesley; 1995.

[Rico99] Rico, David F.: Using Cost Benefit Analyses to Develop a Pluralistic Methodology for Selecting from Multiple Prescriptive Software Process Improvement (SPI) Strategies; MSA 685 Project Report; 1999.

[Sommerville96] Sommerville, I.:Software Engineering. 5th edn.; Addison Wesley; 1996.

Sitios en Internet

[WEB1] http://www.dis.um.es/~bmoros/Tutorial/parte21/cap21-3.html; *Tutorial de Java - Conectividad JDBC*, 2001.

[WEB2] http://www.objectplanet.com/EasyCharts/; *EasyCharts*.

[WEB3] http://www.nitrolic.com/; *GuildFTPd*.

[WEB4] http://jakarta.apache.org/; *Apache Tomcat*.