Centro de Investigación y de Estudios Avanzados del IPN

Departamento de Ingeniería Eléctrica

Sección de Computación

Evaluación de la Calidad en Sistemas de Información en Internet

Tesis que presenta

Leticia Dávila Nicanor

para obtener el Grado de

Maestra en Ciencias

en la Especialidad de

Ingeniería Eléctrica opción Computación

Director de la Tesis: Dr. Pedro Mejía Alvarez

México D.F. Diciembre del 2003

Resumen

A pesar del gran número de artículos de investigación y normas existentes sobre el tema de validación de calidad del producto de software, existen hoy en dia muy pocas Industrias de Software que utilicen procesos de evaluación y análisis para este efecto. Actualmente, la gran mayoría de estudios están enfocados a las actividades de administración de los proyectos de desarrollo de software. En diversos entornos industriales y académicos, la calidad del software ha sido evaluada (validada) mediante distintos estudios analíticos. De dichos entornos, la evaluación de la calidad del software ha evolucionado hacia modelos formales estadísticos que se basan en métricas como fundamento para el aseguramiento, control y evaluación de la calidad de un producto o proceso de software. Grandes compañías como IBM, Hewlett Packard, Motorola y Siemens, entre otras, fundamentan su marco de producción de software con éste enfoque estadístico, lo cual las ha convertido en pioneras de este campo.

La contribución del presente trabajo está dirigida al desarrollo de una metodología para la evaluación y el análisis de los atributos de calidad en los productos de software para Internet. En este trabajo, empleamos la teoría de modelación estadística para el análisis y evaluación de los atributos de calidad. Nuestro principal objetivo es lograr que esta metodología sirva de modelo para cualquier organización que requiera llevar a cabo la validación de los atributos de calidad en sus productos de software.

Índice General

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

Debido a su creciente importancia, los Sistemas de Información en Internet demandan de una alta calidad en su desarrollo y operación. Sin embargo, lograr que un sistema de software o producto de software opere con calidad es complejo debido a que distintos factores que son parte del proceso de desarrollo y producto de software, afectan de manera directa a su operación.

Esta problemática se ha tratado de enmarcar y resolver con múltiples investigaciones y congresos dedicados a este tema. Es posible apreciar la evolución de la problemática a través de los congresos de calidad de software, organizados por el Comité de Software de la Organización Europea para la Calidad. En el primero congreso celebrado en Bruselas en 1988 (ECSQ 88), los temas centrales fueron los elementos de control de calidad; las pruebas sus métodos, las herramientas y técnicas aplicables a las pruebas, las pruebas de integración de sistemas y de aceptación. En el congreso celebrado en Oslo en el año de 1990, las pruebas y las métricas de producto fueron el tema principal. En 1992, en Madrid (ECSQ 92), se resaltó la reciente publicación de la directiva comunitaria relativa a la certificación ISO 9001. En Basilea en 1994 (ECSQ 94), se discutió la etapa de transición del sistema ISO 9001 a los modelos de madurez y mejora continua. En la conferencia celebrada en Dublín en 1996 (ECSQ 96), se resaltó la consolidación de los modelos de mejora continua.

En la Conferencia Internacional en Ingeniería de Software celebrada en Orlando Florida en el año del 2002 (ICSE 2002) se establecieron los atributos necesarios en un software con calidad para aplicaciones que operan en Internet, intranet y de comercio electrónico. Los atributos de calidad que requiere un sistema de esta naturaleza son: *confiabilidad*, *seguridad*, *usabilidad*, *disponibilidad*,

escalabilidad y *mantenibilidad*.

Garantizar que un sistema software posee ciertos atributos de calidad, es una tarea ardua. Esto es debido a que no existe algún estudio que pueda servir de guía para estimar los aspectos mínimos que se deben asegurar. El aseguramiento de la calidad de un sistema de software implica la realización de evaluaciones del sistema. Sin embargo, no existen estudios que hablen en forma específica de cómo se debe realizar el proceso de evaluación para un cierto atributo de calidad.

Cada uno de los atributos de calidad de un producto de software es un tema de gran amplitud, por lo cual en esta tesis nos enfocaremos al estudio de la evaluación de la *confiabilidad* en los sistemas en Internet. Un camino obvio para estimar la *confiabilidad* de un sistema es simular su contexto operacional y verificar el tiempo en el que se presentan fallos. Diversos estudios estadísticos han sido aplicados en la solución de este problema, sin embargo, existen distintas limitaciones con este enfoque. Una de estas limitaciones es la dificultad de generar un contexto operacional desde un punto de vista ideal que esté cercano a un contexto real.

Es evidente la importancia de la evaluación de un producto de software en un contexto real. Algunas industrias que desarrollan software, en la actualidad se limitan sólo a realizar pruebas del producto de software y a corregir los defectos que puedan ser localizados en dichas pruebas. La falta de un análisis formal de los resultados, fomenta que en muchos de los casos este proceso resulte poco fiable y carente de aportaciones significativas. Además, no se aplican métricas de software durante el proceso de evaluación, lo cual produce una gran incertidumbre acerca de los atributos que se desean asegurar. Este proceso resulta ser tan inmaduro que en la mayoría de las ocasiones lejos de beneficiar al producto de software, este resulta afectado.

De acuerdo a este enfoque es obvio que la calidad del producto se ve afectada por el proceso de desarrollo. Existe una gran dependencia de la calidad del producto de software con el cumplimiento de los requerimientos. En este aspecto, *los procesos de mejora continua* se restringen a frecuentes validaciones durante cada una de las fases. El Instituto de Ingeniería de Software de la Universidad de Carnegie Mellon, ha desarrollado un modelo de madurez de la capacidad (CMM), y el proceso de mejora continua (PSP - Proceso de Software Personal). Estas aportaciones desde su creación han sido implementadas y valoradas en organizaciones como IBM, Hewlett Packard y Motorola. CMM y PSP están diseñados de acuerdo a la filosofía de su creador Watts Humprey, para quién la solución a los problemas de calidad consiste en una mejora de los procesos con los que la organización construye, mantiene y gestiona el software.

De acuerdo a lo expuesto anteriormente se ve la necesidad de formalizar los aspectos y los pasos que se deben de seguir en la valoración de atributos de calidad como lo son la *confiabilidad* de los sistemas de software. La contribución del presente trabajo de investigación está enfocada a la validación de la propuesta de una metodología que permita valorar la *confiabilidad* de los sistemas en internet. En esta metodología se integrarán, técnicas de simulación y modelación estadísticas, sobre procesos de madurez y mejora continua. Estos elementos integrados son la base para la evaluación, análisis y mejora de la calidad en la operación de los productos de software.

1.1 Objetivos generales y específicos de la tesis

General

La presente tesis tiene el objetivo de desarrollar una metodología para la validación de los atributos de calidad en sistemas de información en Internet. En esta metodología se integrarán métricas de software y técnicas de modelación estadística para su análisis. El objetivo de este proceso es evaluar, predecir y controlar la calidad de los productos de software mediante la utilización de modelos estadísticos.

En la actualidad, existen una gran cantidad de textos y normas de calidad. Sin embargo no existe alguno que contenga la formalización del proceso de validación y evaluación de los atributos de calidad durante el desarrollo de software para los sistemas de información en internet. En ésta tesis proponemos el desarrollo de una metodología para el mejoramiento de la calidad en los sistemas de internet. Las hipótesis de ésta metodología se valorarán mediante su aplicación a un caso de estudio práctico.

Los objetivos específicos son los siguientes:

- Se utilizarán técnicas de simulación estadística para predecir el comportamiento de los atributos de calidad en contextos ideales.

- Se propondrá un proceso de evaluación mediante la selección de métricas de software de calidad que sean la base para realizar el análisis de la operación de los atributos de calidad en el sistema de software.

- Se utilizarán modelos estadísticos para realizar el análisis de las evaluaciones.

- Se compararán los modelos obtenidos para realizar la valoración de la evolución del comportamientos de los atributos de calidad.
- Se introducirá al proyecto el modelo PSP (Proceso de Software Individual) como marco de trabajo con el fin de aplicar mejoras al proceso y con ello beneficiar la calidad de operación del producto.

1.2 Metodología propuesta

La metodología que se propone se conforma de cinco proceso:

1. *Evaluación del atributo de calidad en el sistema ideal.* En este proceso la se simula la operación del sistema en un contexto ideal. El análisis se realizará mediante la obtención de modelos estadísticos. De aquí en adelante la metodología se enfocará a buscar que el comportamiento de la confiabilidad en un contexto de operación real tienda a aproximarse al observado en la operación ideal.
2. *Evaluación del atributo de calidad en el sistema real (inicial).* El modelo del comportamiento de la confiabilidad buscara comparar el comportamiento del sistema en un contexto ideal contra el comportamiento del sistema en un contexto real.
3. *Implementación del proceso de mejora continua PSP.* La implementación de PSP se enfoca a mejorar la manera en la que se gestiona el software durante su proceso de desarrollo.
4. *Evaluación del sistema final.* Esta etapa es necesaria para cuantificar la mejoras obtenidas durante la implementación de PSP. El modelo resultante, se espera que tenga un comportamiento que tienda a ser como el ideal.
5. *Análisis de los resultados y conclusiones.* La metodología se propone como parte de un proceso de mejora continua, donde los puntos que son clave durante la mejora de la calidad del producto de software sean la base de conocimiento para el desarrollo de proyectos de software futuros.

La tercera y cuarta etapa de la metodología propuesta, es establecida en forma iterativa. Se espera que con la implementación de PSP, los modelos resultantes serán cada vez más parecidos al modelo ideal. Esta iteración mejora - evaluación depende del tiempo y alcance que cada producto de software requiera.

El caso de estudio donde se pretende aplicar la metodología es un sistema de software (SIV) para apoyar el proceso de inscripciones v

'ıa internet a una universidad. Tomamos el modelo de inscripciones que sigue el Departamento de Ingeniería Eléctrica (*DIE*). El SIV tiene la finalidad de mejorar los procesos para la inscripción de los alumnos del centro, a los respectivos cursos que se imparten en cada sección del *DIE*. El (SIV) se realizó tomando en cuenta teorías, procesos y técnicas de la Ingeniería de Software.

1.3 Infraestructura

La arquitectura del sistema que utilizaremos como caso de estudio, cuenta con los siguientes elementos:

- *Un servidor de Base de Datos.* El manejador de Bases de Datos será *Mysql* [23].
- *Un servidor de páginas Web.* El servidor utilizado es *Ápache* [24].
- *El lenguaje para desarrollar las interfaces: PHP* [25].
- *El sistema operativo.* En nuestro caso es *Linux* [22].
- *Conectividad con Internet.*

1.4 Contribuciones o resultados esperados

En este trabajo se pretende presentar un estudio teórico y práctico que permita implementar una metodologías para la mejora de la calidad en sistemas de información en internet. En esta metodología se desarrollarán los procesos para la validación y el control de los atributos de calidad, resaltando los resultados obtenidos al mejorar el proceso de desarrollo de los productos de software mediante la implementación de procesos de mejora continua. Debido a la magnitud de cada uno de los atributos, el presente trabajo sólo abordará la *confiabilidad*. Se entregará un sistema de inscripciones a través de Internet (SIV), optimizado de acuerdo a la metodología que se plantea.

1.5 Organización del documento

La presente tesis está estructurada en seis capítulos.

- En el capítulo 2 se introducen los conceptos de administración de calidad, calidad de software y los modelos que la definen. También se discuten las métricas de software más empleadas para medir aspectos de calidad y del proceso de evaluación.
- En el capítulo 3 se hace una revisión general de lo que son los procesos de madurez y mejora continua, en este caso se habla especialmente del Proceso de Software Personal (PSP).
- En el capítulo 4 se presenta el detalle de la metodología, y se implementan sus dos primeras fases.
- En el capítulo 5 se desarrolla la fase que corresponde a la implementación del proceso de mejora y la fase de la metodología que corresponde a la evaluación de las mejoras.
- Finalmente el capítulo 6 contiene las conclusiones del trabajo realizado y se presentan las posibles líneas de trabajo a futuro.

Capítulo 2

Evaluación de la Calidad en el Desarrollo de Productos de Software

El objetivo de este capítulo es dar un panorama del concepto de la calidad en un producto de software, así como explicar la importancia de su proceso de evaluación y sus implicaciones. En este capítulo se abordar'an temas importantes para el aseguramiento de la calidad como son la administración de la calidad, las métricas y los atributos de calidad del software.

2.1 Introducción

Dentro del campo de la evaluación de la calidad del software, se han realizado múltiples estudios, an'alisis y metodologías. En su mayoría, estos estudios tienden hacia enfoques formales, en donde los modelos estadísticos basados en métricas de software son la base para el aseguramiento, control y evaluación de la calidad de un producto o proceso de software. Grandes compañías tales como IBM, Hewlett Packard, Motorola y Siemens entre otras, han adoptado este enfoque en su marco de producción, para implementar atributos de calidad como lo son la mantenibilidad una vez que el producto de software se ha completado. Esto las convierte en pioneras de este campo.

El objetivo principal del proceso de evaluación es lograr el control del proceso de desarrollo y del producto de software. Esto se logra mediante el monitoreo y la medición de los atributos de las actividades que intervienen en el coste, calidad y todas aquellas características que afectan la

producción de software. Para mejorar la calidad necesitamos verificar el software y sus defectos, para disminuirlos a medida que avanza el desarrollo del proyecto. Este proceso de evaluación es posible mediante la medición del software. Sin embargo, el software no es fácil de medir. Es posible medir algunos atributos del software, y estos sólo pueden ser medidos de manera indirecta. Las métricas pueden utilizarse para medir tanto el proceso de desarrollo como el producto de software. Su aplicación puede variar dependiendo del número de líneas de código o de la complejidad del software en medición. Las métricas no son simples valores, sino que dependen de la magnitud del sistema y del tiempo estimado que se tiene para su obtención. Una vez que se ha obtenido el conjunto de valores de las evaluaciones, se procede a modelar los resultados para obtener un estimado del comportamiento del sistema de acuerdo a la tendencia que manifiestan los resultados. En el modelado se utilizan por lo general métodos formales.

Todo este estudio tiene el objetivo de mejorar los siguientes aspectos del proceso de desarrollo y del producto de software.

- *Planeación.* Estimación de costos, entrenamiento, recursos, tiempos y presupuestos.
- *Organización.* La asignación y orden de los recursos y las actividades.
- *Control.* El nivel y el seguimiento de las actividades del desarrollo del software para dar cumplimiento a lo planeado.
- *Procesos.* Todas las actividades que intervienen dentro del proceso de desarrollo de software.
- *Cliente.* Este punto concentra el objetivo principal de medir y evaluar el software, porque lo principal es la satisfacción del cliente.

2.2 Administración de la Calidad

Lograr un alto nivel de calidad de un producto o servicio es el objetivo de la mayoría de las organizaciones. La calidad del software es un concepto complejo que no se puede definir de una manera sencilla ya que intervienen diversos elementos. En principio, la administración de la calidad comprende simplemente definir procedimientos y estándares a utilizar durante el desarrollo de software y comprobar que todo el personal los siga. En la práctica la administración de la calidad es más que esto.

La administración de la calidad del software se estructura en tres actividades principales:

- *Aseguramiento de la calidad.* El establecimiento de un marco de trabajo de procedimientos y estándares organizacionales que conduce a software de alta calidad.

- *Planeación de la calidad.* La selección de procedimientos y estándares adecuados a partir de este marco de trabajo y la adaptación de éstos para un proyecto de software específico.

- *Control de la calidad.* La definición y promulgación de los procesos que aseguran que los procedimientos y estándares para la calidad del proyecto son seguidos por el equipo de desarrollo de software.

Un estándar internacional que se puede utilizar en el desarrollo de un sistema de administración de la calidad en todas las industrias es el *ISO 9000*. Este es un conjunto de estándares que se aplican a una gran variedad de organizaciones que van desde las Industrias de Manufactura hasta las Industrias de servicios. ISO 9001 es el más general de los estándares y se aplica a las organizaciones interesadas en el proceso de calidad del diseño, desarrollo y mantenimiento de productos de software.

El control de la calidad implica vigilar el proceso de desarrollo de software para asegurar que se sigan los procedimientos de consolidación de software para asegurar que se sigan los procedimientos de consolidación y estándares de calidad. La calidad de software aplica a todas las etapas del desarrollo del software. Sin embargo es de particular importancia tomar en cuenta los siguientes puntos antes de plantearse metas y objetivos de calidad.

- Identificación del alcance y de los objetivos del proyecto.

- Identificación de la infraestructura del proyecto.

- Análisis las características del proyecto.

- Identificación de las actividades del proyecto.

- Revisión y publicación del plan.

La calidad del proceso de desarrollo afecta directamente a la calidad de los productos a entregar. De lo cual podemos concluir que la calidad del producto está íntimamente ligada a los procesos de producción. El término de calidad también está ligado a la *cultura de calidad* que practican los individuos que integran una organización.

2.3 Evolución del Concepto de Administración de Calidad

Desde un punto de vista tradicional la calidad es una idea que no es posible medir y cuantificar. Este concepto se ha transformado en función de las necesidades económicas de la humanidad al paso del tiempo. Actualmente la calidad puede ser medida y definida en términos de satisfacción de requerimientos del cliente o usuario final, mediante la implementación de métricas y el análisis de estas. Partiendo de este enfoque existen dos aspectos fundamentales que marcan la implementación de Calidad en los productos de Software, estos son: *la calidad intrínseca del producto de software y la satisfacción del cliente.*

En la evolución del enfoque de la *Calidad* han influido los modelos, conceptos e investigaciones que se han realizado para entender el concepto de calidad. Otro factor que ha influido en el camino a seguir es la repercusión de hechos históricos como la Segunda Guerra Mundial. Después de la Segunda Guerra Mundial, Japón se encontraba frente a la difícil tarea de reconstruir su economía. En aquel momento, las fuerzas de ocupación de los EEUU, decidieron apoyar en la reconstrucción de la economía y la infraestructura de manera directa, con el objetivo de evitar que el Japón recuperara su capacidad bélica. Para ello crearon la CCS (Civil Communication Section), que debería difundir mensajes pro-EEUU en la población, a través de programas de radio. Lamentablemente, la población no contaba con radios.

Antes de la guerra, Japón construyó establecimientos industriales orientados a la fabricación de radios, pero luego de la guerra, los administradores experimentados del Japón fueron alejados de puestos de esta naturaleza por su labor durante la guerra y el personal con el que se contaba carecía de formación y experiencia, por lo que el resultado fueron productos de bajísima calidad. Se crearon instituciones como el NETL (National Electric Testing Laboratory), con la responsabilidad de controlar la calidad. Sin embargo, poco tiempo después se reconoció que esta estrategia nunca podría alcanzar buenos resultados. Así que se reorientaron los esfuerzos hacia la capacitación de nuevas generaciones de administradores. El programa que se realizó conjuntamente por la CCS y la JUSE (Unión de Científicos e Ingenieros del Japón) incluía el Control Estadístico de la Calidad.

2.3.1 El control estadístico de la calidad

Entre los temas de capacitación, se incluyó el Control Estadístico de la Calidad (CEC) y especialmente los aportes en este campo de Walter Shewhart. La JUSE atribuyó a este enfoque una razón, tal vez la principal, de la victoria de los EEUU en la guerra y orientó su interés hacia este

campo, solicitando a la CCS que les recomendara a expertos que pudieran profundizar y reforzar el tema. Shewhart, uno de los expertos no estaba disponible, así que recomendaron a un profesor de la Universidad de Columbia, que había estudiado y aplicado los métodos de Shewhart, W. Edwards Deming. Ya en 1947 Deming había estado en el Japón como parte de una misión de observación económica, por lo que los japoneses ya lo conocían, facilitando su incorporación como instructor.

En 1950, durante dos meses, Deming entrenó a cientos de Ingenieros y Administradores, así como a ejecutivos de primer nivel, enfocándose principalmente en tres aspectos claves:

- El ciclo PDCA.
- Las causas de las variaciones.
- El control de procesos con Cuadros de Control.

Al inicio los resultados fueron bastante buenos, pero poco a poco se regresaba a la situación inicial, la información recolectada no era exacta y los ejecutivos no mostraban interés en continuar con el CEC. Para tratar de solucionar este dilema, la JUSE invitó a Joseph M. Juran para realizar conferencias y charlas respecto del Rol de la Gerencia en la Promoción de las Actividades de Control de Calidad. Esta visita marcó el salto en Japón de los primeros pasos en Calidad hacia la Calidad Total al introducir aspectos como la definición de las políticas de calidad y la planificación de la calidad. Lo cual se reforzó con el lanzamiento del libro "The Practice of Management"de Peter Drucker, en el que se plantea la *Administración por Objetivos*. Los Japoneses fusionaron las enseñanzas de Deming y Juran con la Administración por Objetivos y dieron los primeros pasos hacia la Planeación Estratégica de la Calidad y hacia la Administración de la Calidad Total ACT.

2.4 Administración de Calidad Total (ACT)

El término *Administración de Calidad Total* fue creado en 1985 por el Comando de Sistemas Aéreos Navales de la USAF, para describir el estilo que utiliza la administración japonesa para mejorar la calidad de sus productos de software. Actualmente el término ha tomado numerosos sentidos, de acuerdo a quién lo interpreta. En terminos generales éste representa un estilo de dirección exitoso en cuanto a la implementación de la calidad en los productos de software y en la satisfacción del cliente. La base para este tópico es la creación de una cultura de calidad que integra a todos los miembros de la organización.

Desde 1980 un gran número de compañías de los Estados Unidos adoptaron el ACT como marco de trabajo. Buscando mejorar la calidad en sus productos de software. La adopción de ISO 9000 por la Comunidad Europea como estandard administrativo y la aceptación de estos estándares por la iniciativa privada, han fomentado que cada vez mas organizaciones implementen marcos de trabajo con base en el ACT. Compañías en el marco industrial de la computación y la electrónica como Hewlett Packard, IBM, Motorola y Siemens (entre otras), han implementado satisfactoriamente ACT. Un claro ejemplo es IBM que con su división de AS/400, ven recompensado su esfuerzo en sus productos de software con reconocimientos como el Malcom Baldrige National Quality Award.

Los elementos principales del sistema ACT [15] (mostrados en la figura 2.1) se concentran en los siguientes aspectos:

- *Enfoque del cliente.* Si tenemos como objetivo principal la satisfacción total del cliente, nos enfrentamos a un problema que involucra varios aspectos técnicos y sociales. En este punto de acuerdo a la ACT se debe incluir estudios acerca de lo quiere y necesita el cliente de acuerdo a la especificación de sus requerimientos, medir y evaluar la satisfacción de ellos.

- *Proceso.* El objetivo principal es tomar como base las experiencias para enriquecer los procesos de mejora. Con este enfoque es posible obtener una línea de producción cada vez más estable en el proceso de negocio y en el proceso de desarrollo. Los procesos de mejora son muy importantes ya que a través de ellos se podrán obtener productos de software de mayor calidad.

- *Lado humano de la calidad.* El objetivo es crear una compañía con una cultura de calidad. Aquí intervienen factores psicológicos y sociales que motiven y convenzan al personal de las ventajas que trae laborar con Calidad.

- *Medidas y análisis.* El objetivo de evaluar y analizar es cuantificar los resultados que se van generando en cada fase del proceso de desarrollo. Con ello se permite establecer predicciones más claras y objetivas. Otro aspecto que se beneficia, es el control de la calidad del proceso y del producto de software.

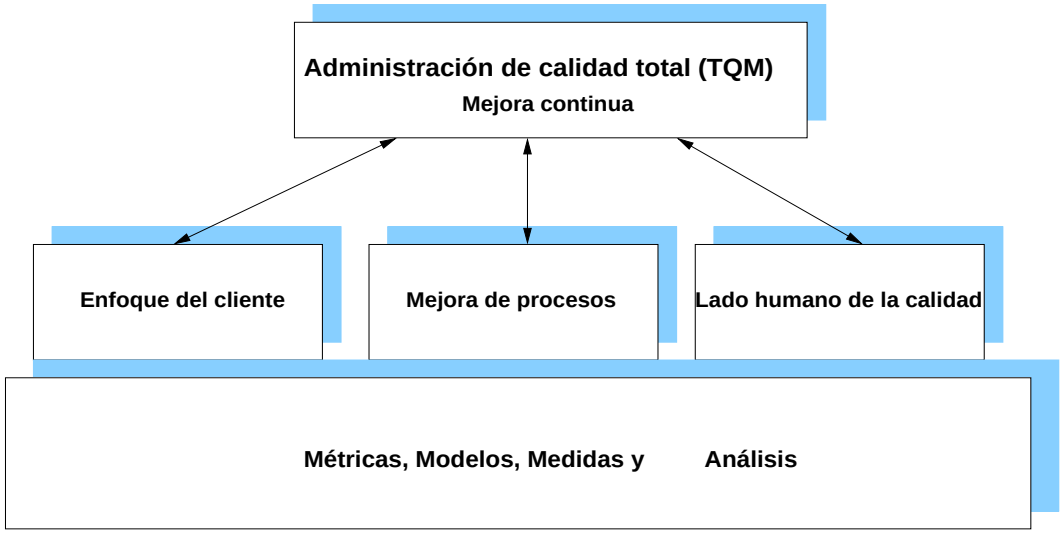

Figura 2.1: Ideas principales de la administración global de calidad (ACT).

2.5 Concepto de Calidad de Software

Mientras la calidad en general puede considerarse como un concepto entendible, en la práctica, la definición de calidad de un sistema puede ser vaga. Para saber cuando un sistema encuentra sus requerimientos de calidad necesitamos realizar un juicio objetivo. El concepto de calidad de software puede tener diferentes enfoques. El enfoque de un usuario final ó cliente y el del diseñador del software pueden ser distintos. El objetivo principal es que se logre llegar a un punto en común. En un esfuerzo por definir el concepto de calidad, algunos autores argumentan que una especificación o atributo de calidad afecta el cómo se obtienen mejoras en el funcionamiento de la operación de un producto de software.

De acuerdo a la terminología de la IEEE, la calidad de un sistema, componente o proceso de desarrollo de software, se obtiene en función del cumplimiento de los requerimiento iniciales especificados por el cliente o usuario final.

2.6 El Modelo de McCall

Las especificaciones de la calidad de un producto de software han sido objeto de trabajo de diferentes grupos, de los cuales uno de los más destacados es el de McCall [1]. En este trabajo se establece un modelo de Calidad para los productos de software. Este modelo establece tres áreas principales

que intervienen en la Calidad del Software.

1. *Calidad en la operación del producto.* En general se requiere que este pueda ser entendido fácil, que opere eficientemente y que los resultados sean los requeridos inicialmente por el usuario.

2. *Revisión de calidad del producto.* Tiene como objetivo realizar revisiones durante el proceso de desarrollo para detectar los errores que afecten a la operación del producto de software. Las revisiones involucran grupos de personas que examinan parte o todo el proceso del software, los sistemas o su documentación asociada para descubrir problemas potenciales. Las conclusiones de la revisión se registran formalmente y se pasan al autor o a quien sea responsable de corregir los problemas descubiertos.

3. *Calidad en el proceso.* Desde este enfoque se recomienda definir o seleccionar estándares y procedimiento que sirvan como marco de trabajo durante el desarrollo de software. En la figura 2.2 se pueden apreciar la relación que guardan estos tres aspectos.

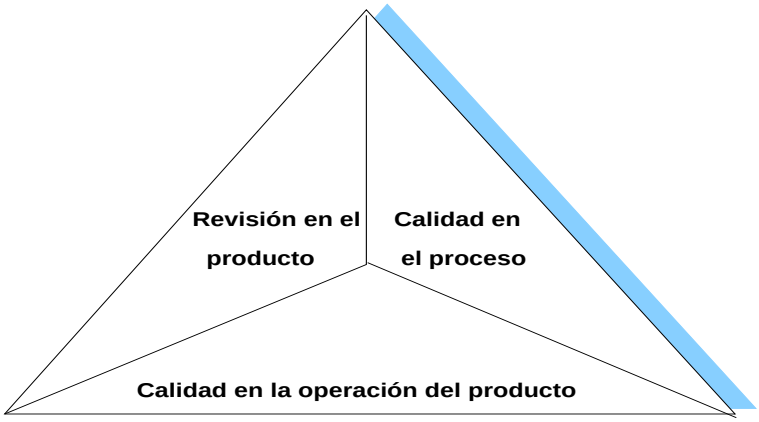

Figura 2.2: Modelo de McCall de calidad de software

2.7 El Modelo de Boehm

Otro modelo que es importante resaltar es el de Boehm [1]. Este modelo, descrito en la figura 2.3, destaca por ser uno de los mejor definidos. El modelo es de naturaleza jerárquica y los criterios de calidad se presentan en tres grandes subdivisiones. La primera división es hecha acorde a los

servicios que el sistema va a ofrecer (*portabilidad*). La segunda se hace de acuerdo a la operación del producto (*usabilidad*) y la tercera gran subdivisión se hace de acuerdo a la *mantenibilidad* del producto de software.

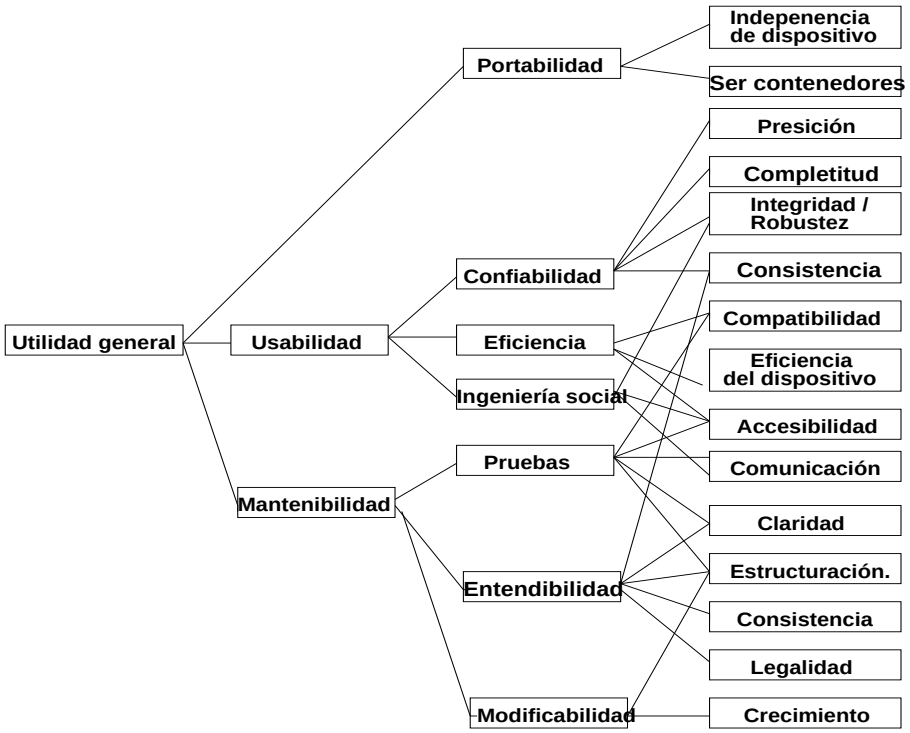

Figura 2.3: Modelo de Boehm para clasificar los criterios de calidad

Entre los criterios básicos del modelo podemos mencionar:

- *Usabilidad.* Este atributo de calidad de software se enfoca a mejorar la simplicidad, entendibilidad y facilidad de uso de un sistema de software para un cliente o usuario final.
- *Mantenibilidad.* Tradicionalmente de define como el esfuerzo requerido para localizar y especificar un error en la operación de un módulo, función o sistema de software.
- *Portabilidad.* Se define como el esfuerzo requerido para transportar la configuración de hardware y/o software un módulo, función o sistema de software en el ambiente de una plataforma a otra.
- *Confiabilidad.* La *confiabilidad* de un sistema de cómputo es una propiedad que implica el grado de confianza esperado por parte del usuario en la operación adecuada del sistema al

utilizarlo. La *confiabilidad* es afectada por cuatro aspectos fundamentales:

- *Disponibilidad*. Define la probabilidad de que el sistema esté funcionando en un tiempo determinado.
- *Fiabilidad*. Es la probabilidad de que el sistema funcione correctamente durante un intervalo de tiempo específico.
- *Seguridad*. Representa la capacidad de que el sistema no afecte su entorno y el de quien lo utiliza.
- *Protección*. Representa la capacidad del sistema para protegerse de intrusiones accidentales o programadas.

Sin embargo, la *disponibilidad*, la *seguridad* y la *protección* se ven afectadas por la *fiabilidad*. Para evaluar la *fiabilidad* debemos tomar en cuenta que los sistemas de software no son entidades estáticas. La *fiabilidad* de un sistema se complica a medida que este crece. Es posible caracterizar el comportamiento de la *fiabilidad* estudiando el comportamiento de las fallas, para lo cual, podemos considerar por ejemplo el tiempo de arribo de fallos, el número de fallos ocurrido en un tiempo determinado, o la media del tiempo de ocurrencia de fallas. En todos estos casos estamos afectados por variables aleatorias. Podemos realizar modelos subsecuentes del comportamiento de las fallas tratándolas como parte de un proceso estocástico. Cuando el software falla es necesario localizar y reparar las fallas que causan los errores del sistema.

- *Eficiencia*. Se refiere al esfuerzo en base al costo de los recursos de la computadora que emplea un sistema, módulo ó función de software para su ejecución.
- *Ingeniería social*. Este atributo se refiere al entendimiento y aceptación del sistema de software del grupo social al cual se enfocaron los requerimientos del sistema.
- *Pruebas*. Es el conjunto de exámenes que se realizan a un sistema, modulo ó función una vez finalizado su desarrollo. Estos exámenes se realizan con el objetivo de asegurar que en sistema, modulo ó función, cumple con la especificación de los requerimientos del usuario o cliente final.
- *Entendibilidad*. Es la claridad en cuanto a la lógica del código fuente de un módulo, función o sistema de software.

- *Modificabilidad.* Este atributo está muy ligado a la Entendibildad, debido a que si un código es entendible, como consecuencia podrá ser cambiado en su contenido sin que esto afecte negativamente al sistema en el cual se integra.

Criterios adicionales que han complementado al modelo de Boehm y que surgido en años recientes son:

- *Flexibilidad.* Representa el esfuerzo requerido para modificar un el código de un sistema, módulo ó función de software en operación.
- *Reusabilidad.* El alcance para el cual un programa puede ser usado en otras aplicaciones.
- *Interoperatividad.* Se define como el esfuerzo requerido para acoplar un sistema a otro.

2.8 Interrelación de los Criterios de Calidad

De acuerdo al estudio de Perry [1], descrito en la figura 2.4, las marcas obscuras implican una relación inversa, mientras que las claras expresan una relación directa.

Se puede observar que la relación entre los atributos de calidad varía. Cuando los atributos de calidad mantienen una *relación inversa*, se asume que algunos atributos transgreden la operación de los otros. Por otro lado, cuando los atributos mantienen una *relación directa* se asume que es posible que estos se puedan beneficiar entre sí. Por ejemplo, si observamos el aspecto de *confiabilidad*, veremos que la *reusabilidad* le afecta; por el contrario, la *mantenibilidad* le favorece. Esto es debido a que la *confiabilidad* se refiere a la seguridad de la operación sin errores de un sistema, módulo ó función. Mientras que el objetivo de un *código reusable* es que un mismo código pueda operar en distintos ambientes de desarrollo; lo que aumenta la probabilidad de tener errores difíciles de percibir dentro del código. Por el contrario la *mantenibilidad* se enfoca a reducir el número de errores de un sistema, módulo ó función. La conclusión del estudio es que los atributos de calidad que debe tener un producto de software dependen en gran medida del objetivo del desarrollo del producto de software, de su proceso de desarrollo y de su contexto de operación.

La interrelación que existe entre algunos atributos de calidad se discute a continuación.

- *Integridad vs. Eficiencia (relación inversa).* Un sistema de software posee la propiedad de *integridad* sí, los recursos manipulados por éste no son alterados o destruidos por usuarios,

entidades o procesos no autorizados. Al implementar el atributo de *Integridad* es necesario contemplar un número mayor de condiciones para cumplir con las reglas de integridad. Este punto incrementa el tamaño del código afectando la eficiencia. El objetivo de la *Eficiencia* es lograr que un sistema de software requiera un costo menor en cuanto a los recursos de la computadora. Sin embargo, a medida que aumenta el tamaño del código aumentan los recursos empleados para su ejecución. Por lo anterior la *integridad* afecta la operación de la *eficiencia*.

- *Usabilidad vs. Eficiencia (relación inversa).* El objetivo del atributo de *usabilidad*es lograr que al usuario le sea cómoda y fácil la operación del sistema de software. Desafortunadamente para lograr implementar este atributo, en algunas ocasiones se sacrifican atributos importantes como lo es la *seguridad*. Otro inconveniente es que el código crece significativamente, lo cual impacta en el número de recursos empleados para la ejecución del producto de software. Un claro ejemplo de ineficiencia y aplicaciones usables son los productos de software de Microsoft.

- *Mantenibilidad vs Flexibilidad (relación directa).* El objetivo del atributo de *flexibilidad* es generar códigos bien estructurados que sean entendibles. Uno de los objetivos de la *mantenibilidad* es corregir los defectos que se puedan presentar en la operación de un sistema de software. Si un sistema de software es flexible en su código, este será mucho mas fácil de mantener y las modificaciones que se requieran realizar pueden localizarse con mayor facilidad. Por lo cual la *flexibilidad* beneficia a la *mantenibilidad*.

- *Portabilidad vs. Reusabilidad (relación directa).* El *código portable* tiene por objetivo operar libre de especificaciones de ambiente. Esta es una de las características de la *reusabilidad*. La *portabilidad* de un modulo ó sistema de software le beneficia a la *reusabilidad*.

- *Correcciones vs Eficiencia (relación neutral).* El objetivo de *corregir* un sistema de software el liberarlo de errores de operación. Sin embargo, eliminar los errores del sistema no siempre asegura que el tiempo de ejecución del sistema vaya a disminuir. La *corrección* de errores no asegura que la *eficiencia* se vea favorecida, pero tampoco le afecta.

En la figura 2.4 se muestra el análisis de Perry, el cual define las relaciones entre los criterios de calidad.

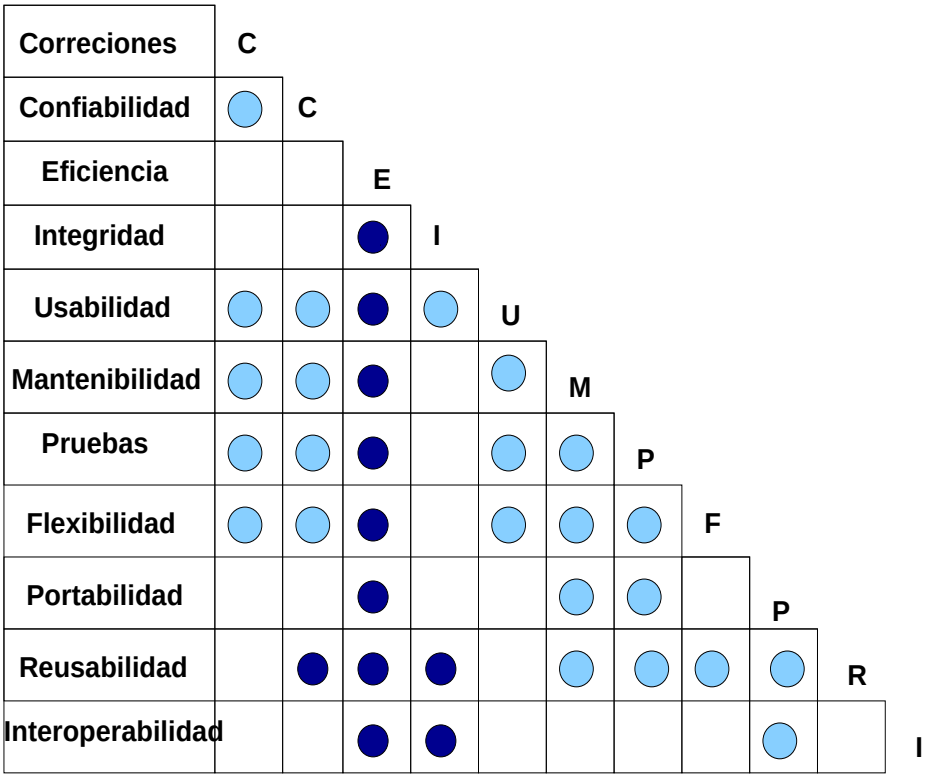

Figura 2.4: Relación entre los atributos de calidad para un producto de software.

2.9 Aspectos de Calidad en los Sistemas en Internet

El Word Wide Web (WWW) fue originalmente diseñado para presentar información mediante una apariencia sencilla, con sitios sin complicaciones significativas ya que su consistencia era primordialmente de texto e hipervínculos. Las aplicaciones que existen hoy en día cuentan con una gran variedad de lenguajes para crear aplicaciones de comercio electrónico, información distribuida, entretenimiento, trabajo en grupo y encuestas, entre otras actividades. Existen hoy en día una gran diversidad de arquitecturas heterogéneas y distribuidas donde pueden ejecutarse y prestar estos servicios. El software de aplicaciones Web puede ser distribuido, con una implementación en múltiples lenguajes y estilos, múltiples interfaces de usuario, integración de múltiples sitios y de Bases de Datos. Podemos apreciar en la figura 2.5 la arquitectura de un sitio Web moderno.

Los componentes que integran este tipo de aplicaciones incluyen software tradicional (navegadores como Explorer, Netscape), software contemporáneo (servidores de Web como Apache), lenguajes que son interpretes de guiones(Java Script), archivos HTML, programas que son compilados(programas en Lenguaje Java), Bases de Datos (Oracle, MySQL, Informix, DB2), imágenes

gráficas (jpg, gif) y complejas interfaces de usuario (por ejemplo, presentaciones en Flash de Macromedia). El desarrollo de un sitio Web requiere de un gran equipo de personas con diferentes perfiles y backgrounds. Estos equipos incluyen programadores, diseñadores gráficos, Ingenieros en usabilidad, especialistas en integración de la información, expertos en redes y administradores de Bases de Datos. Esta diversidad de perfiles en cuanto al personal requerido es necesaria para iniciar un sitio Web en la actualidad.

Cuando desarrollamos una aplicación para Internet se pueden encontrar diversos problemas. Una de las preocupaciones fundamentales, es la integración y adaptación de los elementos que las conforman. Lo cual implica mucha de la complejidad que interviene en su desarrollo. En general el tamaño en cuanto a líneas de código para este tipo de software, normalmente se estima en varios miles o incluso millones de líneas de código. Esto es debido a que su enfoque primordial es proveer servicios de información. El manejo de una gran cantidad de registros de datos está comprometido. Lo cual puede implicar desde un pequeño manejador de bases de datos hasta una infraestructura de DatawareHouse. Por lo expuesto con anterioridad las organizaciones demandan una mayor calidad por parte de este tipo de productos de software.

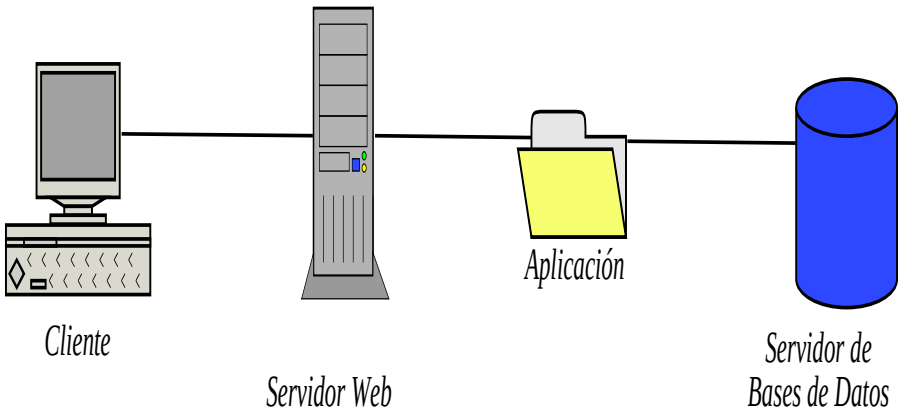

Figura 2.5: Arquitectura de un sitio Web en la actualidad.

Entre una de las ventajas que ofrece el software basado en la Internet es la reducción de los costos. Esto es debido a que los procedimiento de la organización pueden ser automatizados y administrados por la red. Sin embargo la información almacenada depende de la capacidad de los equipos de cómputo. Algunas veces se construyen complicada aplicaciones de negocio que generan grandes flujos de datos a través de la red (por ejemplo las aplicaciones bancarias).

Debido a la importancia que la calidad de software que la Internet ha despertado en los ultimos

años; la Conferencia Internacional de la Ingeniería de Software del año 2002 (ICSE 2002) se centró en los aspectos de Calidad para los Sistemas en Internet [13]. En esta conferencia se concluyó que los criterios de calidad más importantes a los que puede aspirar un producto de software de esta naturaleza son los siguientes:

- *Confiabilidad.*
- *Usabilidad.*
- *Seguridad.*

Así mismo, en la conferencia ICSE 2002 se definieron otros criterios de calidad para los sistemas en Internet.

- *Disponibilidad.*
- *Escalabilidad.*
- *Mantenibilidad.*

2.9.1 Fiabilidad

Es la probabilidad de que el sistema funcione correctamente durante un intervalo de tiempo. El número de usuarios de sitio Web es muy grande y las expectativas en cuanto a la *fiabilidad* acerca de las aplicaciones Web son muy grandes porque a través de estas se manejan pedidos, compras, ventas ó simplemente despliegue de información importante para los usuarios. Un inadecuado funcionamiento en estos productos de software no solo elimina la confianza de los usuarios sino también puede generar grandes pérdidas económicas.

2.9.2 Usabilidad

Tomas Powell describe que la *usabilidad* es una de las características más importantes para hacer un sitio atractivo para los usuarios y el cual puedan ser aceptado e integrado a sus actividades cotidianas. El objetivo de la *usabilidad* es mejorar la simplicidad, entendibilidad y facilidad de uso del sitio web. La percepción de la *usabilidad* es influenciada por las características de los usuarios, como son la edad, el nivel educativo, el perfil tecnológico, etc. La percepción de esta característica se ve afectada por las diferencias culturales asociadas, como por ejemplo el diseño gráfico, el uso

de los colores y el contenido de la información [4]. La relación de algunos de los elementos que intervienen se presentan en la figura 2.6.

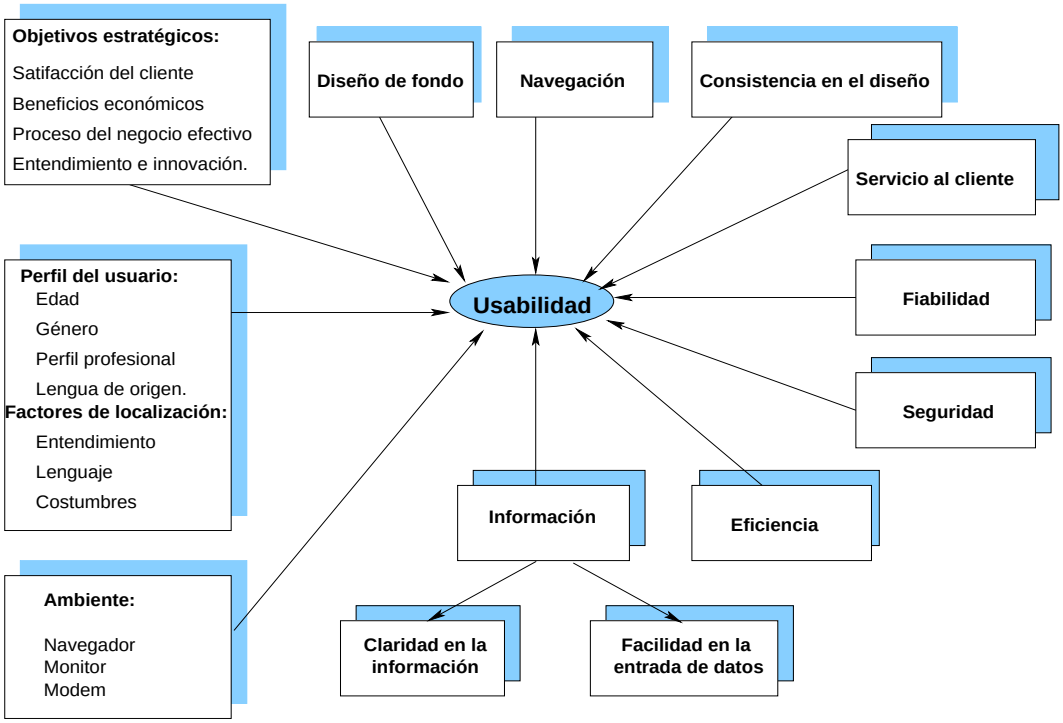

Figura 2.6: Elementos que intervienen en la usabilidad.

2.9.3 Seguridad

La Seguridad representa la capacidad de que el sistema no afecte su entorno y el de quién lo utiliza. Existen muchos sistemas de software en Internet que trabajan con usuarios mediante el acceso personalizado, por ejemplo las aplicaciones bancarias. Esto conlleva la introducción de una serie de medidas de seguridad implementadas dentro de las aplicaciones y de los servicios. En caso de no aplicar medidas de seguridad adecuadas, no sólo existe el riesgo de perjudicar la imagen de las organizaciones, también esta latente el riesgo de afectar la economía de los usuarios del sistema de software. En la actualidad existen compañías que tienen grandes pérdidas económicas, debido principalmente la falta o a la implementación inadecuada del aspecto de calidad *seguridad*.

2.9.4 Disponibilidad

Este atributo, define la probabilidad de que el sistema esté funcionando por un tiempo determinado. Sobre un sitio Web, los clientes no únicamente esperan disponibilidad 24 horas al día, los siete días

en la semana durante todo el año. Sólo una pérdida de tiempo de 10 minutos puede ocasionar grandes pérdidas, por ejemplo en las aplicaciones de la bolsa de valores. En otros casos las pérdidas pueden ser de ventas de diferentes artículos, como libros (es el caso de Amazon), revistas, etc. La disponibilidad también implica que todos los usuarios puedan accesar a la información no importando si utilizan un browser de Netscape ó Explorer, o las plataformas Windows o Unix. En el caso *Shockwave Flash* el cual es únicamente compatible con Microsoft Internet Explorer y Netscape sobre Windows, los usuarios que trabajan sobre Unix y Netscape no podrán accesar a la información que es manejada por este software. En este caso, también está latente la poca disponibilidad del sistema. Es claro que el hecho de hacer disponible un sitio desde todos los puntos de vista implica adoptar lenguajes más robustos y estables, aunque la facilidad para desarrollar estas aplicaciones se vea reducida.

2.9.5 Escalabilidad

Las necesidades para la escalabilidad han derivado en una gama muy amplia con la que se cuenta en la actualidad de tecnología de innovación en cuanto a software y hardware. La industria ha desarrollado nuevos lenguajes de software, estrategias de diseño, tecnologías de comunicación de datos y transferencia de protocolos. La necesidad surge cuando el número de clientes ó usuarios que utilizan los servicios de una aplicación en Web crece. Con el objetivo de evitar el riesgo de insuficiencia de recursos en el sistema, normalmente se programa crecimientos en la infraestructura. Cuando un sistema de esta naturaleza esta preparado para crecer, lo hace sin complicaciones. Este aspecto debe ser contemplado desde la planeación, análisis y el diseño, dentro del proceso de desarrollo de software.

2.9.6 Mantenibilidad

Un aspecto novedoso que deriva del software basado en la Internet, son las nuevas versiones (releases), o las actualizaciones. El software tradicional involucra mercadeo, precios, y ventas, personal de instalación para los clientes. Sin embargo, este proceso es caro y el desarrollo de este software usualmente requiere modificaciones de mantenimiento, con sus estimados de tiempo. Es el caso, por ejemplo, de las múltiples versiones que manejan, las aplicaciones de Microsoft. El mantenimiento que se requiere para aplicaciones basadas en el web muchas veces debe ser inmediato. El acceso de los clientes es algo que implica una actualización constante en cuanto a información y el servicio a los módulos. Los cambios que se realizan en mucho de los casos deben ser inmediatos y transparentes

al usuario. En ocasiones extremas las aplicaciones deben ser actualizadas sin dejar de proporcionar el servicio. Por citar algunos ejemplos tenemos las aplicaciones que manejan ventas por Internet, las aplicaciones que manejan acciones de alguna casa de bolsa, ó las aplicaciones bancarias donde se realizan transacciones en línea.

2.10 Proceso de Medición de Productos de Software

La medición del software se refiere a derivar un valor numérico para algún atributo de un producto de software o un proceso del software. Comparando estos valores entre ellos y con los estándares aplicados en la organización, es posible sacar conclusiones de la calidad del software o de los procesos de software.

Existe una reticencia en las compañias que desarrollan software para introducir medidas debido a que los beneficios no son claros. Una razón de esto es que, en muchas compañías los procesos del software utilizados aún están pobremente organizados y no son suficientemente maduros como para utilizar dichas medidas. Otra razón es que no existen estándares para las métricas y por lo tanto, es difícil llevar a cabo la recolección y el análisis de datos. En la actualidad, muchas compañías no están preparadas para introducir mediciones sino hasta que existan disponibles tales estándares y herramientas.

A menudo es imposible medir los atributos de calidad del software de forma directa. Los atributos como la *fiabilidad*, *mantenibilidad* o la complejidad, por citar algunos ejemplos, se ven afectados por diversos factores y no existen métricas directas para ellos. Los esfuerzos se orientan a medir algún atributo interno del software (como el número de defectos) y suponer que existe una relación entre lo que se puede medir y lo que se quiere saber. El proceso de medición es de las actividades más importantes ya que de este depende que los resultados puedan ser fiables para poder emprender posteriormente un análisis objetivo.

Los pasos que normalmente se siguen en el proceso de medición son:

- Seleccionar las medidas a realizar.
- Seleccionar los componentes a evaluar.
- Medir las características de los componentes.
- Identificar las mediciones anómalas.

- Analizar los componentes anómalos.

2.11 Métricas de Software

Una métrica de software es cualquier tipo de medida relacionada con un sistema, proceso ó documento de software. Las métricas a emplear dependen del atributo de calidad a evaluar y de las condiciones de desarrollo y operación del sistema. Los atributos de software más comunes son los mostrados en las figuras 2.3 y 2.4. Para cada atributo es posible que existan varios tipos de métricas de software que se pueden aplicar. En algunos casos las métricas de software existentes no son aplicables al ambiente de operación del proyecto, o estas son difíciles de obtener. En estos casos es posible implementar nuevas métricas que estén de acuerdo a las normas de calidad de la organización. Compañías como Motorola, IBM y Hewlett Packard han desarrollado métricas adecuadas a su marco de producción .

2.11.1 Características de las métricas de software

Las métricas deben cumplir con ciertos puntos tales como:

- *Nombre.* El identificador de la métrica que pueda ser conocido.
- *Definición.* La descripción de los atributos de las entidades que pueden medirse utilizando la métrica. Debe describirse como se calcula y cuál es su valor por defecto.
- *Objetivo.* Enumera los objetivos que pueden ser alcanzables y las respuestas que se pueden obtener mediante dicha métrica. Así como la justificación de la importancia de la métrica.
- *Procedimiento de análisis.* Aquí se describe cómo se entiende el uso de la métrica. Las precondiciones bajo las cuáles actúa para obtener una interpretación adecuada de los valores de estas. Es necesario contar con técnicas de análisis y herramientas para el modelado.
- *Responsabilidades.* Este punto se refiere a que siempre debe existir un responsable de colectar, registrar los datos de las medidas, preparar los reportes y analizar los datos.

2.11.2 Tipos de métricas de software

Las métricas de software pueden clasificarse en tres categorías:

- Métricas de producto.
- Métricas de proyecto.
- Métricas de proceso.

Las *métricas del producto* describen las características del producto como son el tamaño, la complejidad, las características de diseño y los atributos de calidad. Las *métricas del proceso* pueden ser utilizadas para mejorar el desarrollo y el mantenimiento del software. Las *métricas del proyecto* describen características administrativas y su ejecución, como son costo, planeación, productividad del personal, etc.

Las métricas que nos interesan en esta tesis son aquellas que nos pueden ayudar a evaluar la calidad [15]. Dichas métricas son un subconjunto de las del producto y el proceso. Las métricas de calidad están subdivididas en:

1. *Métricas de Producto final*
2. *Métricas del proceso de desarrollo*
3. *Métricas del mantenimiento del software.*

Las primeras dos contemplan niveles que se relacionan con la calidad instrínseca del producto y la satisfacción del cliente con respecto al producto. La tercera división esta en función del mantenimiento durante el ciclo de vida esperado para el producto de software.

2.11.3 Métricas de producto final

Las Métricas de Producto Final más importantes para nuestro estudio son las siguientes.

1. *La media del tiempo de ocurrencia de fallos.* Se refiere al promedio del tiempo que tarda en producirse un error durante la operación de un producto de software.
2. *Densidad de defectos.* El concepto se refiere al número de errores que se ejecutan en un marco de tiempo, durante la operación del producto de software.
3. *Problemas detectados por el usuario.* Esta métricas se utiliza con mayor frecuencia dentro de las empresas. Consiste en que el usuario pruebe por un tiempo determinado el producto de software. La cantidad de tiempo para realizar las pruebas varía de seis meses a sólo un mes,

dependiendo del criterio de cada organización. Los problemas una vez detectados se reportan. Esta métrica es un tanto subjetiva, ya que depende de la experiencia, habilidad y enfoque personal de los usuarios.

4. *Niveles de satisfacción del cliente.* Esta métrica se refiere a la satisfacción que el cliente encuentra con el producto de software. Los niveles que se manejan son los siguientes.

   - Muy Satisfecho - 100 %.
   - Satisfecho - 75 %.
   - Neutral - 50 %.
   - Insatisfecho - 25 %.
   - Muy Insatisfecho - 0 %.

La calidad instrínseca de un producto puede medirse por el número de "bugs" (o defectos del producto durante su operación) ó por el tiempo que tarda en presentarse un error. Las métricas citadas con anterioridad estan enfocadas a los errores que ocurren durante la operación de un software, por lo cual, es necesario describir el concepto de error.

De acuerdo a la IEEE/ANSI (Instituto Nacional de Estándares de los E.E.U.U.) se manejan las siguientes definiciones de error.

- Un error es un olvido humano que se visualiza en un resultado incorrecto de un software.
- Un fallo es el resultado de una condición accidental que causa una unidad del sistema, cuando es requerida para el funcionamiento del sistema.
- Un defecto es una anomalía en un producto.
- Una falla ocurre cuando una unidad funcional de un software relacionado con un sistema no tiene la eficiencia necesaria que requiere el funcionamiento de este, debido a los limites especificados para la unidad.

2.11.4 Métricas del proceso de desarrollo

Las Métricas del proceso de desarrollo están orientadas a medir la calidad en el ciclo del proceso de desarrollo. La idea principal es entender que dentro del proceso de desarrollo las métricas juegan un

papel muy importante debido a que a través de ellas es posible cuantificar y evaluar el acercamiento hacia el comportamiento esperado, sin que sea necesario esperar hasta que el producto finalice para realizar esta evaluación. Estas métricas son menos formales que las del producto. Mientras que algunas organizaciones sólo toman en cuenta un tipo de métricas, otras contemplan diversos tipos de métricas de acuerdo a la fase del proyecto.

Las principales métricas del proceso de desarrollo son las siguientes:

- *Densidad de defectos durante el mecanismo de pruebas.* La velocidad del arribo de defectos es el indicador de que el software está experimentando una alta incidencia de errores durante el proceso de desarrollo. Una densidad de defectos nunca sigue una distribución uniforme.

- *Patrón de arribo de defectos durante el mecanismo de pruebas.* El conjunto de pruebas para la densidad de defectos son un indicador global de la *confiabilidad* del sistema. El patrón está referido a semanas o ocasionalmente a meses.

- *Patrón basado en la fase de remoción de defectos.* Este patrón de métricas es una extensión de la métrica de densidad de defectos. El agregado de este patrón consiste en el rastreo de los defectos para todas las faces del ciclo de desarrollo, incluyendo la revisión del diseño, inspección del código y verificaciones formales previas a las pruebas.

- *Efectividad en la eliminación de defectos.* Esta métrica esta definida mediante la siguiente expresión:

$$DRE = \frac{DRFD}{DLP} * 100\%$$

  Donde, DRFD= Defectos removidos durante la fase de desarrollo, DLP=Defectos latentes en el producto y DRE = Efectividad en la remoción de defectos.

2.11.5 Métricas del mantenimiento de software

Cuando el desarrollo de un producto de software se termina y entra en ciclos de revisión se puede decir que la fase de mantenimiento comienza un ciclo de vida. Durante esta fase, los defectos que tiene el producto de software son reportados por el cliente y estos son corregidos durante las revisiones.

Algunas métricas de mantenimiento del software son las siguientes.

- *Indice de bitácoras administrativas (BMI).*

$$BMI = \frac{NPCM}{NPADM} x100\%$$

Donde, NPCM = Número de problemas cerrados durante el mes, y NPAM = Número de problemas que llegan durante es mes.

- *Correcciones y tiempos de corrección.* Es el tiempo promedio en que se realiza la corrección de un defecto de software desde su detección hasta su saneamiento.

- *Porcentaje de correcciones defectuosas (PDF).*

$$PDF = \frac{NFERTCSN}{NFDST} x100\%$$

Donde, NFERTCSN = Número de correcciones que exceden el criterio del tiempo de acuerdo a nivel especificado por el proceso y NFDST = Número de correcciones liberadas en un tiempo determinado.

- *Correcciones de calidad.* Esta métrica está enfocada a la satisfacción del cliente con respecto a los productos entregables.

Capítulo 3

Procesos de Mejora Continua

El objetivo de este capítulo es analizar y comprender el funcionamiento de los procesos de desarrollo y la forma en que los procesos de mejora continua mejoran la calidad del producto. Con el objetivo de entender los procesos de mejora y el Proceso de Software Personal, en este capítulo primero explicaremos los modelos para el desarrollo de software, la administración de proyectos de software y el modelo de capacidad de madurez de los procesos de software CMM.

3.1 Introducción

La mejora de la calidad de un producto de software es compleja debido a que implica revisar los procesos de desarrollo de software, las actividades que intervienen en ellos y como se están realizando. Modelos como el Modelo de Maduración de la Capacidad CMM - SEI, proporcionan un enfoque para evaluar las capacidades de los elementos que integran una organización que desarrolla software. Para mejorar las cualidades de calidad de un producto de software está implícito el factor humano. El enfoque de los procesos de mejora continua permite elevar la calidad de los productos mediante la mejora de las prácticas personales de los ingenieros que desarrollan software dentro de una organización. PSP es un ejemplo de ello.

El Proceso de Software Personal (PSP) se enfoca a la mejora de las prácticas individuales mediante la valoración de las capacidades de los Ingenieros de Software y a través de la evaluación de sus tiempos de su productividad y eficiencia. De esto deriva un análisis que los propios Ingenieros

realizan. De este análisis, aprenden a medir sus capacidades. PSP proporciona una secuencia de actividades que les ayudan a mejorar la forma en como realizar su trabajo. El proceso de evaluación-análisis se realiza durante todo el Proceso, con lo cual es posible que los ingenieros valoren sus avances. Sin embargo aplicar Procesos de Mejora Continua no es fácil, ya que podría existir mucha resistencia al cambio de actitudes y de enfoques para desarrollar software. Sobre todo en aquellas organizaciones donde no se tiene bien claro el objetivo que se persigue, donde no se tiene un proceso de desarrollo de software maduro y en donde no le dan la importancia al hecho de laborar con eficiencia generando productos de calidad.

3.2 Modelos para el Proceso de Desarrollo de Software

El modelo del proceso de desarrollo elegido por parte del administrador es muy importante debido a que marcará la pauta para la planeación y organización del proyecto de software. La elección dependerá de las características del modelo. Así por ejemplo, el modelo de cascada es oportuno si se conocen las fases del proyecto, debido a experiencias anteriores. De lo contrario, podría resultar inconveniente si no se tiene experiencia en este tipo de desarrollo. Bajo este ambiente puede convenir el modelo del prototipado.

Los modelos de mayor uso para este fin son los siguientes:

- *Modelo de cascada.* Este modelo, descrito en la figura 3.1, toma las actividades fundamentales del proceso: especificación, desarrollo, validación y evolución y los representa como fases separadas del proceso, como la especificación de requerimientos, el diseño de software, la implementación, las pruebas, y el mantenimiento.

- *Desarrollo evolutivo ó incremental.* Este enfoque, descrito en la figura 3.2, se entrelazan las actividades de especificación, desarrollo y validación. Un primer sistema se desarrolla rápidamente a partir de especificaciones abstractas. Entonces se ajusta con ayuda del cliente para producir un sistema que satisfaga sus necesidades.

- *Desarrollo formal de sistemas.* Este enfoque se basa en la producción de una especificación matemática formal y en la transformación de esta especificación utilizando métodos matemáticos, para construir los programas.

- *Desarrollo basado en la reutilización.* Este enfoque se basa en la existencia de un número

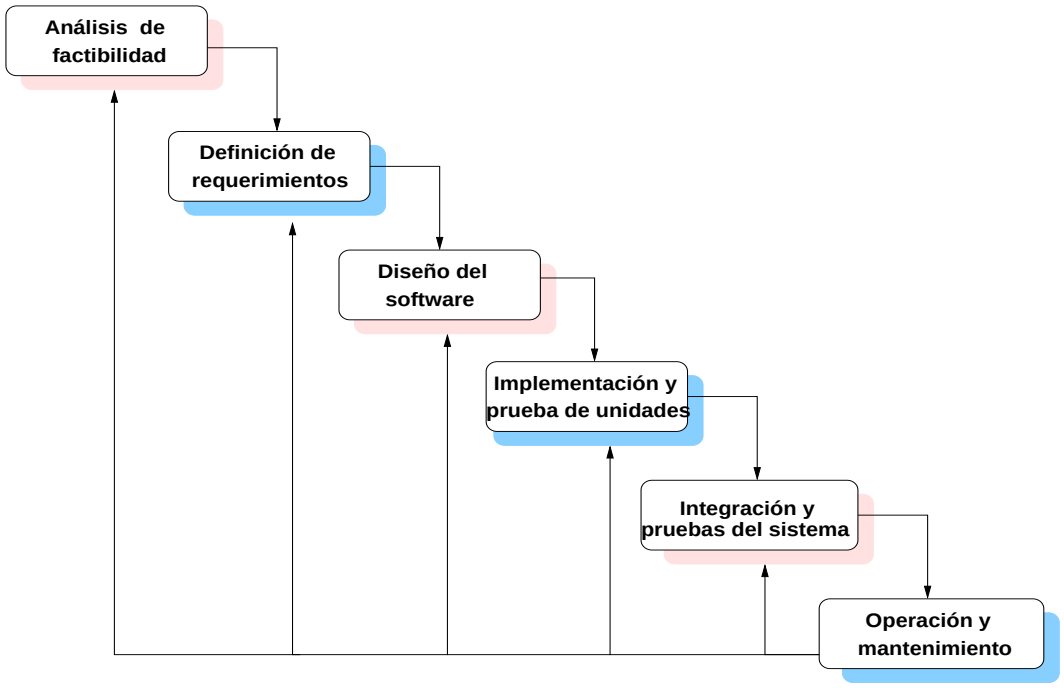

Figura 3.1: Modelo de cascada

significativo de componentes reutilizables. El proceso de desarrollo del sistema se enfoca a integrar estos componentes en el sistema más que en desarrollarlos desde cero.

- *El Modelo de espiral.* El modelo en espiral del proceso del software, que originalmente fue propuesto por Boehm (1988), hoy se conoce ampliamente. Mas que representar el proceso de software como una sucesión de actividades con retrospectiva de una actividad a otra, se representa como espiral. Así, el ciclo más interno podría referirse a la *factibilidad* del sistema, el siguiente ciclo a la definición de requerimientos del sistema, el siguiente ciclo al diseño del sistema y así sucesivamente.

3.3 El Modelo de Madurez de la Capacidad del Proceso (CMM - SEI)

El CMM (Capability Maturity Model) ó Modelo de Madurez de la Capacidad del Proceso (CMM) fue creado por el Instituto de Ingeniería de Software de la Universidad de Carnegie Mellon. Este modelo está compuesto de varios niveles que permite evaluar la madurez y capacidad de las empresas

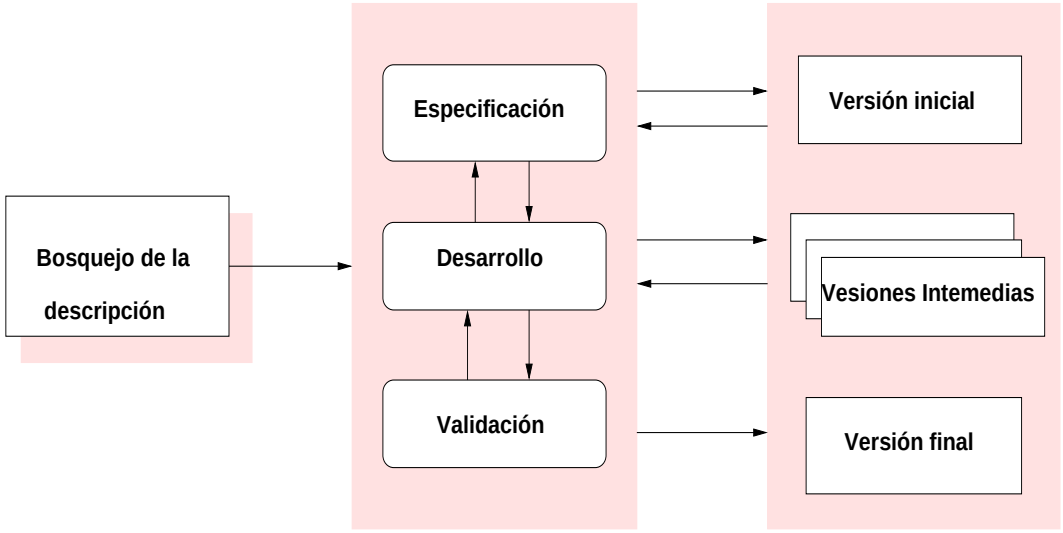

Figura 3.2: Modelo de desarrollo evolutivo.

que desarrollan software. El producto de este trabajo de valoración de la capacidad fue el Modelo de Madurez de la Capacidad de Software del SEI [11],[10]. Debido a las características que tiene implícitas el modelo es posible utilizarlo para la mejora de los procesos de desarrollo de software. El CMM está compuesto de cinco niveles, que intervienen en las diferentes facetas del desarrollo de procesos.

Los Niveles del CMM, que se muestran en la figura 3.3, son los siguientes:

1. Nivel inicial. En este nivel la organización no tiene procedimientos de administración efectiva o planes de desarrollo de sus proyectos de software. Aunque en la organización podrían existir los elementos formales para el control del proyecto, no existen mecanismos organizacionales para asegurar que se utilizarán en forma consistente. El desarrollo del software puede ser exitoso, pero las características como la calidad y los procesos (costos, duración, desempeño) son impredecibles en la mayoría de los proyectos. En este nivel, la eficiencia en el desarrollo de proyectos por lo general se mejora mediante la utilización de métodos efectivos para el control y administración del proyecto.

2. Nivel repetible. En este nivel, una organización tiene procedimientos formales de administración del proyecto, de aseguramiento de la seguridad y de control de la configuración. Se denomina repetible debido a que la organización puede repetir proyectos del mismo tipo de forma exitosa y por que en cada repetición los proyectos podrán ser mejorados y desarrollados con un menor coste. Sin embargo, hay una falta de un modelo de procesos formal. El éxito del proyecto depende de los

administradores individuales que motivan al equipo y del ambiente y procesos de la organización. La principal diferencia entre el nivel inicial y el nivel repetible es que el nivel repetible provee un control de la planeación del proyecto y los compromisos establecidos. La experiencia en proyectos similares es prioritaria para el éxito de los proyectos, porque se logra un mejor control sobre los costos, planes y cambios en los requerimientos.

En este nivel, además de lograr dominar las actividades definidas en el nivel inicial, se consiguen los siguientes aspectos.

- *Administración de la configuración del software.* En esta actividad se establece y mantiene la integridad en todos los puntos clave del ciclo completo del proyecto. Así mismo, en esta actividad se consigue la identificación de los puntos de configuración, las líneas de producción y los procedimientos para el ciclo de control de cambios.

- *Aseguramiento de la calidad del software.* Esta actividad involucra revisiones y auditorías de los procesos y productos para validar que están cumpliendo con niveles aceptables (de acuerdo a los estándares adoptados).

- *Administración de la sub-contratación del software.* En los proyectos de software muy grandes, con frecuencia es necesario sub-contratar grupos de trabajo para desarrollar software que no es posible desarrollar dentro del proyecto. La planeación para las actividades de los sub-contratistas es necesaria y el progreso de estas siempre debe ser planeada.

- *Seguimiento y vigilancia del proyecto de software.* Esta actividad está relacionada con la visibilidad del progreso actual. Los resultados intermedios tienen que ser revisados y vigilados con respecto al plan de actividades. Cuando sea necesario, este plan de actividades tiene que ser re-evaluado y re-diseñado buscado la *fiabilidad* del proceso y del proyecto.

- *Planeación de proyectos de software.* Esta actividad toma en cuenta la creación de los planes para la ejecución y administración del proyecto. En la planeación, un grupo de personas con la responsabilidad de entregar el proyecto de forma exitosa genera estos planes y su documentación. La planeación de proyectos se refiere a la identificación de actividades, etapas, su calendarización y las entregas producidas por un proyecto. Esto implica hacer el bosquejo de un plan para guiar el desarrollo hacia las metas del proyecto. La estimación del costo es una actividad que se refiere al estimado de los recursos requeridos para llevar a cabo el plan

del proyecto. La administración efectiva de un proyecto de software depende de la planeación completa de todo el progreso del proyecto. El administrador debe tener la capacidad de anticiparse a los problemas que puedan surgir y sugerir alternativas de solución. Un plan que se prepara al inicio de un proyecto debe utilizarse como guía para la realización del mismo. La calendarización del proyecto implica separar todo el trabajo de un proyecto en actividades complementarias y considerar el tiempo requerido para completar dichas actividades. Se puede dar el caso en el cual algunas de estas actividades puedan realizarse en paralelo. En la estimación de las actividades, los administradores deben tomar en cuenta que pueden surgir problemas, por los cuál se debe considerar el tiempo en el cual pueda existir algún imprevisto. Por lo general, el calendario de proyecto se representa como un conjunto de gráficos que muestran la división del trabajo, las dependencias de las actividades y la asignación del personal.

- *Administración de requerimientos.* Esta actividad involucra la estabilidad y mantenimiento de los acuerdos con el cliente con respecto a los requerimientos del sistema de software. El software sufrirá de cambios de manera inevitable, pero el control y la documentación en estos casos es de vital importancia.

3. Nivel definido. En este nivel, una organización tiene definidos sus procesos por lo que tienen una base para la mejora cualitativa de procesos. Existen procedimientos formales que aseguran que el proceso se sigue en todos los proyectos de software.

En este nivel, además de lograr dominar las actividades definidas en los niveles 1 y 2, se consiguen los siguientes aspectos.

- *Revisión de puntos.* En esta actividad se busca identificar errores en el desarrollo mediante inspecciones y revisiones del proceso de desarrollo.

- *Coordinación interna de grupos.* En esta actividad lo mas importante es la integración entre el grupo de Ingenieros de software y otros grupos, como son el grupo de aseguramiento de calidad, los usuarios finales o representantes y los administradores del proyecto. Esta coordinación es importante ya que permitirá poner de acuerdo a distintos grupos y llegar a acuerdos.

- *Ingeniería de productos de software.* En esta actividad se proyecta el proceso definido en la actividad anterior para mejorar y adecuar todas las actividades del desarrollo, como son: man-

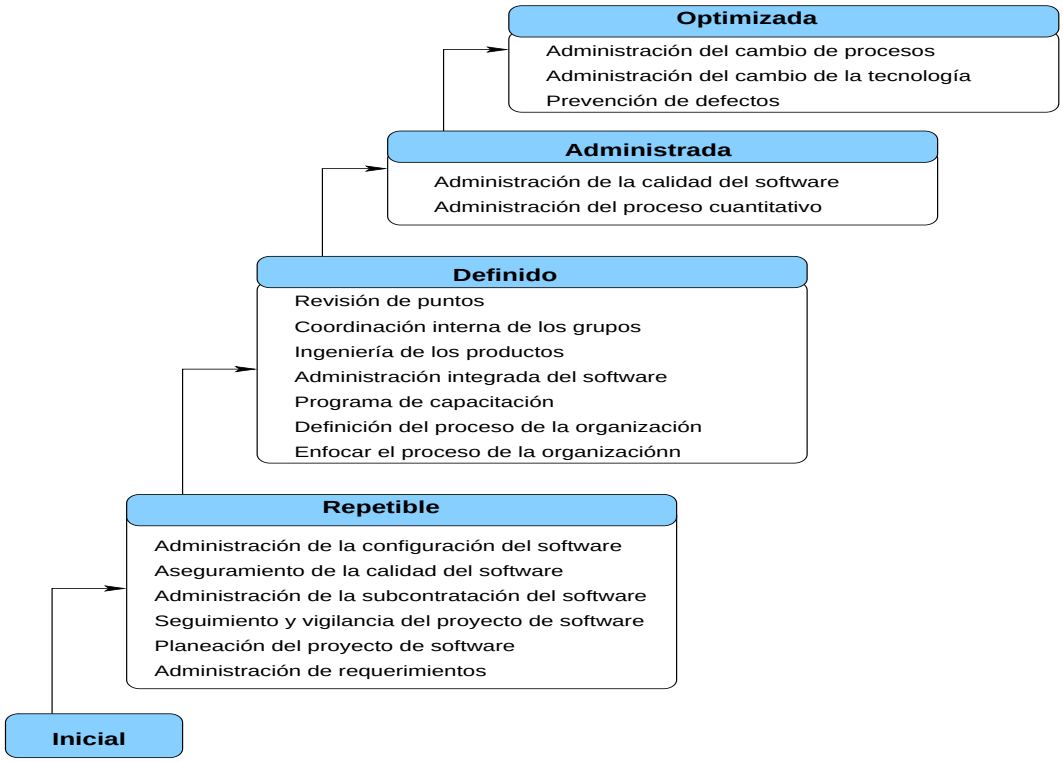

Figura 3.3: Areas del CMM - SEI.

tenimiento y documentación de los requerimientos, diseño y codificación, planeación, ejecución de pruebas y en general la consistencia del trabajo.

- *Administración integrada del software.* Esta actividad involucra el desarrollo de un proyecto específico en cuanto a la línea de proceso que proviene del proceso estándar. Dado que en muchas ocasiones los procesos de software son comunes para distintos proyectos desarrollados, la experiencia que se adquiere puede ser valiosa en futuros desarrollos.

- *Programa de capacitación.* El propósito del programa de capacitación es desarrollar los perfiles y habilidades necesarias para los integrantes del grupo de trabajo con el propósito de hacer mas eficientes sus roles. La capacitación es a nivel organizacional, a nivel de los proyectos y de manera individual.

- *Definición del proceso de la organización.* En esta actividad la organización desarrolla y mantienen un estándar de proceso para el desarrollo del software.

- *Enfocar el proceso de la organización.* Esta actividad involucra el establecimiento y consolida-

ción de las responsabilidades por separado del grupo, para las actividades que intervienen en el proceso de la organización. El grupo tiene la responsabilidad de asentar en forma regular, la información del mantenimiento y de los resultados del proceso de software en una base de datos y la introducción de nuevas herramientas y tecnología.

4. Nivel administrativo. En este nivel, se tiene un proceso definido y un programa formal de colección de datos cualitativos. Recolecta las métricas del proceso y del producto para alimentar la actividad de mejora de procesos.

En este nivel, además de lograr dominar las actividades definidas en los niveles anteriores, se consiguen los siguientes aspectos.

- *Administración de la calidad del software.* En este punto la idea principal es aumentar la calidad del producto. Las normas de calidad son monitoreadas y revisadas desde el inicio hasta el término del producto.

- *Administración del proceso cuantitativo.* En este punto se hace una medición del proceso, se hace un análisis de estas medidas y finalmente se hace los ajustes necesarios para implementar la mejora de los procesos.

5. Nivel optimizado. En este nivel, una organización está comprometida con la mejora continua de procesos. La mejora se calcula y planea como parte integral de los procesos de la organización. En esta etapa el énfasis para la optimización ha cambiado del producto al proceso.

En este nivel, además de lograr dominar las actividades definidas en los niveles anteriores, se consiguen los siguientes aspectos.

- *Prevención de defectos.* Envuelve la identificación de las causas comunes de defectos, con lo que también se cubre la prevención. Esta actividad se realiza de manera periódica.

- *Administración del cambio de la tecnología.* Esta actividad ve específicamente la identificación de nuevas tecnologías y su transición ordenada dentro de la organización.

- *Administración de cambios en el proceso.* La mejora continua de procesos de forma ordenada se aplica para la mejora de la calidad de los productos, la productividad de la organización y la reducción del tiempo necesario para el desarrollo de los productos.

En la actualidad muchas de las organizaciones tienen la iniciativa de introducir programas para la mejora de procesos de software buscando lograr un alto nivel de maduración en sus procesos. Para corregir la eficiencia de trabajo de los ingenieros se ha desarrollado PSP. PSP está enfocado a la forma de trabajar de manera individual de los ingenieros que forman parte de los procesos de desarrollos de software de las compañías. Es un diseño personalizado de los procesos de mejora continua. Los principios básicos del CMM y del PSP son complementarios y de hecho hay puntos dentro del modelo de CMM en los que PSP apoya.

3.4 Proceso de Software Personal

El proceso de software personal (PSP) es un proceso que define fases para su aplicación midiendo los avances y las mejoras. Este proceso tiene un enfoque personal lo cual ayuda a los ingenieros o personal que desarrollan software, mejorando su eficiencia a nivel personal. El impacto de uso de PSP se ve reflejado en el incremento de la eficiencia en la organización. Este proceso provee una solución para medir y analizar el desarrollo y la productividad en los proyectos paso a paso. El objetivo es disminuir los defectos en el código, mejorando la planeación y la productividad. Los procesos de mejora son difíciles porque la mayoría de las personas no aceptan fácilmente cambiar sus hábitos que a veces son ya naturales y parte de su personalidad.

El enfoque de un proceso de mejora se basa en los siguientes puntos:

- *Definir el proceso de Calidad.*
- *Medir la calidad del producto.*
- *Entender el proceso.*
- *Ajustar el proceso.*
- *Utilizar el proceso ajustado.*
- *Medir los resultados.*
- *Comparar los resultados con los objetivos planteados.*
- *Retro-alimentar y mejorar continuamente.*

PSP es un enfoque que muestra como aplicar métodos de Ingeniería de Software para el personal que desarrolla software en sus tareas diarias, con el objetivo de mejorar su trabajo [18], [19] y [14]. Proporciona métodos detallados de planificación de tiempos y estimación de actividades. Muestra a los Ingenieros como controlar su rendimiento frente a estos planes y explica cómo los procesos definidos guían su trabajo. PSP ha sido introducido en muchas organizaciones. Los datos reunidos en su aplicación muestran que PSP es efectivo en la mejora de la eficiencia en la planificación de las actividades del proceso de desarrollo de software y en la calidad de los productos que se obtienen. PSP puede considerarse como una disciplina que proporciona un marco de trabajo estructurado para desarrollar habilidades personales. PSP se basa en la definición de medidas y análisis de datos. De acuerdo a este análisis se verifica en que parte del proceso hay que ejecutar los cambios. Este proceso de evaluación y análisis se realiza en todas las facetas que marca PSP.

Uno de los puntos mas importantes que se aplica con PSP es la reducción de defectos. Otro punto que es parte de este procesos es mejorar la planeación de los tiempos para la ejecución de cada una de las actividades.

3.5 Etapas de PSP

Un proceso de software es una secuencia de pasos requeridas para desarrollar o mantener un producto de software. Las etapas de PSP se muestran en la figura 3.4.

Evaluación del personal PSP0. En este punto se inicia PSP. El primer paso es que los Ingenieros entiendan como aplicar PSP mediante la utilización de reportes escritos de su labor personal. Este reporte se realiza mediante la toma de tiempos de las actividades involucradas en el proceso de desarrollo. Otro punto importante es el registro de los defectos que se realizan durante las fases así como de sus correcciones. Estos son datos de entrada que dan la posibilidad de llevar a cabo una evaluación objetiva. También sirven de patrón de referencia a futuro durante el progreso de la aplicación de PSP. PSP0 está compuesto de tres fases:

- Planeación.
- Proceso de desarrollo.
- Postmortem.

PSP0 ha evolucionado en la versión PSP0.1, la cual se enfoca a la evaluación del desarrollo, mediante la implementación de métricas y estándares en el código. El objetivo de este enfoque es evaluar y cuantificar las mejoras. Una de las formas que PSP propone es PIP la cual permite a quien aplica PSP registrar sus problemas, tipos e ideas. Este formato ayudara a enfocarse posteriormente a la mejora en la manera de realizar su proceso personal de desarrollo. Utilizando los formatos es posible constatar la ayuda que proporcionan para reunir y utilizar los datos de entrada.

Planeación personal (PSP1). El objetivo de esta fase en introducirse al método PROBE. La utilización de PROBE tiene como objetivo estimar tamaños y tiempos en el desarrollo del software, tomando como referencia los datos recopilados en la fase PSP0. En este punto normalmente se utiliza regresión lineal y se estiman los parámetros. De acuerdo a la gráfica obtenida se realizan las predicciones de tamaño y tiempo estimados para el desarrollo del software. La planeación de actividades son la entrada de PSP1.1.

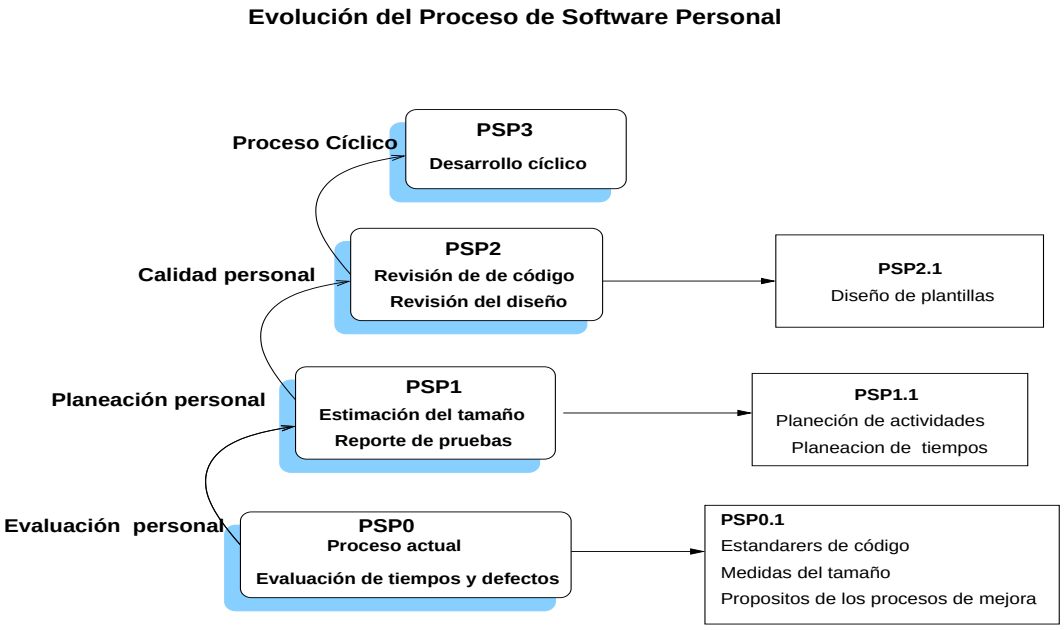

Figura 3.4: Fases de PSP.

Calidad personal (PSP2). Introduce a la administración de defectos. En este punto los datos de entrada son los defectos con los cuales se construye y verifica un estudio para realizar el diseño y la revisión del código. Las *listas de verificación* son utilizadas para realizar las revisiones y mejorar su codificación. Las listas de verificación son hechas de acuerdo a sus perfiles y prácticas personales. PSP2 ha evolucionado en PSP2.1. En este punto se introducen especificaciones de diseño y técnicas

de análisis a lo largo de las fases con el objetivo de prevenir la implementación de defectos. Los procesos de análisis son nuevamente los puntos de referencia. El análisis deriva de la evaluación de los tiempos que duran sus actividades y del número de defectos que introducen y remueven en cada una de las fases del proceso. Quién aplica PSP aprende a evaluar y mejorar su eficiencia personal.

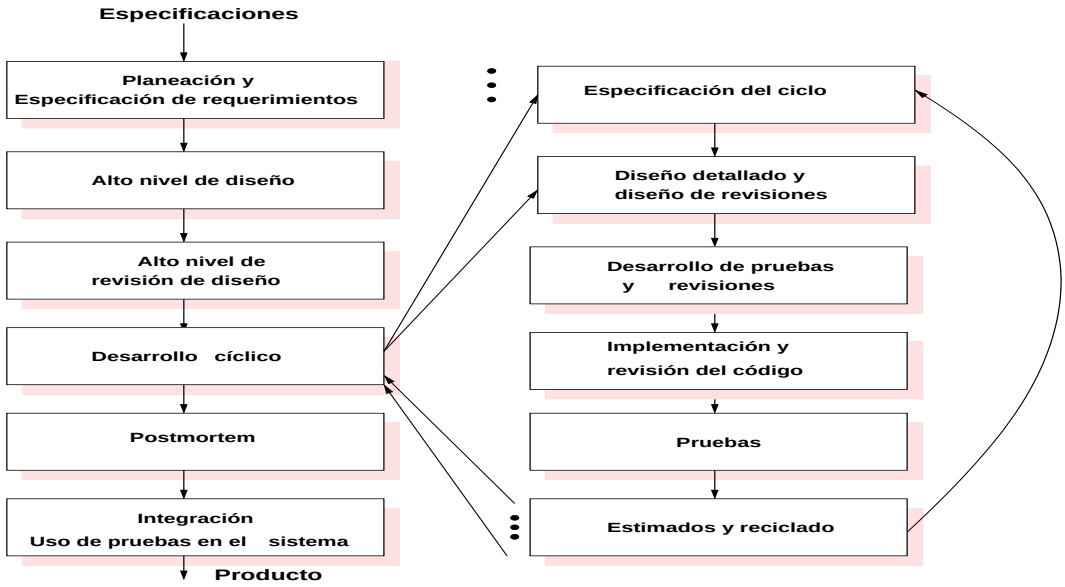

Figura 3.5: Proceso PSP3

Mejora cíclica (PSP3). La figura 3.5 muestra como cada una de las fases son base para la siguiente fase en un modo escalar hacia arriba. Se dice que PSP3 es cíclico porque cada vez que se aplica los resultados son mejores. Como se aprecia en la figura 3.6, la relación que guarda CMM y PSP es directa. CMM se enfoca a la cuestión de qué entorno a las actividades en un proceso de desarrollo de software que tiende a mejorar. Mientras que PSP se enfoca al cómo mejorar los procesos. En términos generales se mejora el desempeño de los Ingenieros que son parte del proceso de desarrollo del software. En la figura 3.6 es posible observar los elementos de PSP en CMM (en negritas).

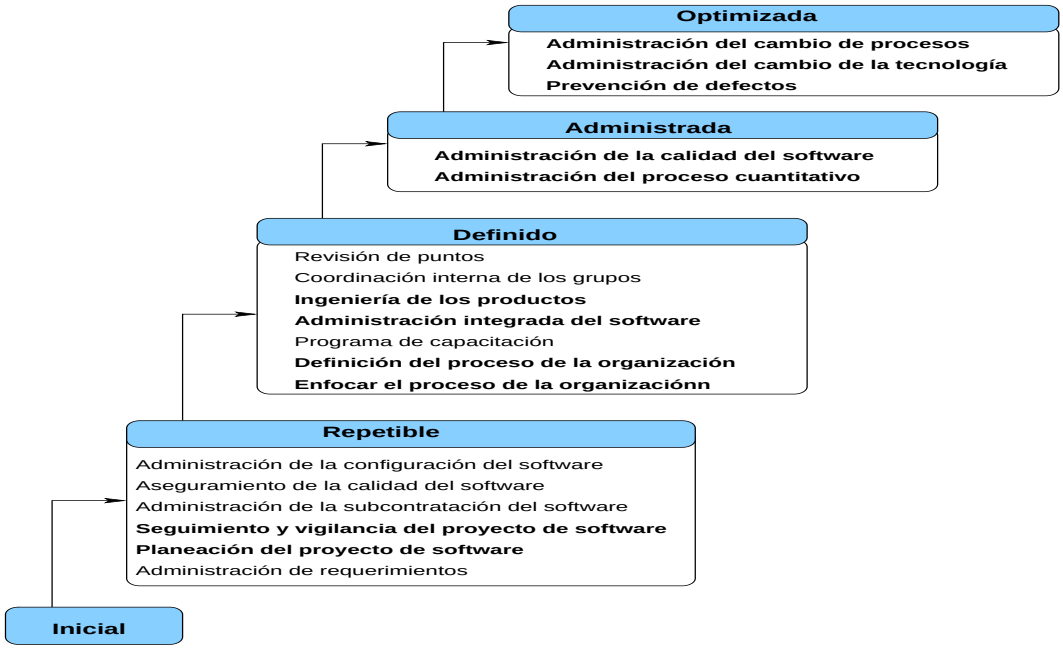

Figura 3.6: Elementos de PSP en CMM - SEI

Capítulo 4

Metodología para la Evaluación de la Calidad en Sistemas en Internet

4.1 Introducción

A pesar del gran número de artículos de investigación y normas existentes sobre el tema de validación de calidad del producto de software, existen hoy en día muy pocas Industrias de Software que utilicen procesos de evaluación y análisis para este efecto. Actualmente, la gran mayoría de estudios están enfocados a las actividades de administración de los proyectos de desarrollo de software. En diversos entornos industriales y académicos, la calidad del software ha sido evaluada (validada) mediante distintos estudios analíticos. De dichos entornos, la evaluación de la calidad del software ha evolucionado hacia modelos formales estadísticos que se basan en métricas como fundamento para el aseguramiento, control y evaluación de la calidad de un producto o proceso de software. Grandes compañías como IBM, Hewlett Packard, Motorola y Siemens, entre otras, fundamentan su marco de producción de software con este enfoque estadístico, lo cual las ha convertido en pioneras de este campo.

La contribución del presente trabajo está dirigida al desarrollo de una metodología para la evaluación y el análisis de los atributos de calidad en los productos de software para Internet. En este trabajo, empleamos la teoría de modelación estadística para el análisis y evaluación de los

atributos de calidad. Nuestro principal objetivo es lograr que esta metodología sirva de modelo para cualquier organización que requiera llevar a cabo la validación de los atributos de calidad en sus productos de software. En la Sección 4.2 se describe la metodología que proponemos para evaluar el sistema de información [21]. Así mismo, en esta sección se describe el proceso de modelado estocástico y el proceso de evaluación que seguiremos en la metodología propuesta. En la Sección 4.3 se describe un caso de estudio que nos permitirá evaluar nuestra metodología en sus dos primeras fases.

4.2 Establecimiento de la Metodología

El objetivo de este trabajo es el desarrollo de una metodología para la evaluación del comportamiento del atributo de calidad *fiabilidad* en los sistemas de información en Internet.

Los objetivos particulares de la metodología son los siguientes:

1. **Evaluación y Predicción de la Calidad.**

   En este objetivo se pretende evaluar y predecir la calidad del producto de software mediante la aplicación de métricas de software y modelos estadísticos. Con el apoyo de las técnicas de modelación estadística [3] realizaremos un análisis del comportamiento del sistema y de sus atributos de software. Los resultados de nuestro análisis serán evaluados mediante un caso de estudio, en donde evaluaremos el comportamiento del sistema bajo condiciones ideales y las compararemos contra la operación del sistema en un contexto real. Una vez hecha esta comparación, seremos capaces de evaluar que tan lejos está el comportamiento de nuestro sistema del funcionamiento ideal.

2. **Mejora de los Procesos.**

   En este objetivo se busca introducir el modelo PSP (Proceso de Software Personal) [7] al producto de software, en busca de mejoras del proceso y producto. La introducción del Proceso de Software Personal en nuestra metodología, nos permitirá mejorar el funcionamiento del producto de software y con ello mejorar el comportamiento de los atributos de calidad.

3. **Administración de la Calidad.**

   En este objetivo se implementará un proceso de administración de calidad y las actividades clave del proceso para el aseguramiento, la planeación y el control de la calidad del producto.

Los primeros 2 pasos de esta metodología se realizan en forma iterativa hasta que los atributos de calidad estén tan cercanos al ideal como se requiera. En este trabajo, nos centraremos en la primera parte de esta metodología, que consiste en la evaluación y predicción de la calidad del producto de software. Es importante mencionar que los 3 pasos de esta metodología se utilizan actualmente en muchas industrias de software. Sin embargo, hasta donde hemos podido investigar, el primer paso no está formalizado en la literatura actual y por otro lado, ha habido poca divulgación por parte de las industrias de software de los procesos que siguen para conseguir calidad. Por esta razón, la contribución de este trabajo consiste en la formalización de la evaluación de la calidad para el atributo de *fiabilidad* en los productos de software en Internet.

4.2.1 Fases de la metodología

Las fases utilizadas en nuestra metodología las cuales se muestran en la figura 4.1, son las siguientes:

1. **Evaluación de la *fiabilidad* en el sistema ideal.**

   La meta en esta fase, es obtener un modelo que describa el comportamiento de los atributos de calidad del sistema en un contexto ideal. Para el funcionamiento del sistema ideal se utilizarán técnicas de simulación. Con la simulación operando aplicaremos el proceso de evaluación y el análisis que describiremos en la Sección 4.2.4. El modelo obtenido servirá de guía durante el desarrollo del producto (sistema real) de software.

   Es importante destacar, que los productos de software en Internet tienen características y necesidades diversas, aunque el contexto de operación sea el mismo (la Internet). Utilizar un modelo ya establecido que sirva de guía para todos los productos resulta poco fiable debido a que los requerimientos son distintos en cada producto. Por lo tanto, es necesario obtener el modelo que mejor se ajuste a las necesidades particulares de cada producto de software. Además, el modelo obtenido no será el mismo para todos los atributos de calidad de software a evaluar, debido a que cada atributo evalúa diferentes propiedades del sistema.

2. **Evaluación de la *fiabilidad* en el sistema real (inicial).**

   En esta fase nuestra meta es obtener el modelo que exprese la situación cualitativa del producto de software. El sistema real, es un sistema de información en Internet para inscripciones en una Universidad. Mediante la evaluación del comportamiento del sistema real y del análisis de los resultados conoceremos la situación actual del atributo de calidad de interés.

El proceso de evaluación y análisis de los resultados para el sistema ideal y real se describe en la Sección 4.3.

3. **Implementación del Proceso de Mejora.**

   El objetivo de esta fase es aplicar PSP (Proceso de Software Personal) al desarrollo del producto de software, con el fin de mejorar el desempeño de los atributos de la calidad en el producto de software. Las condiciones del ambiente de desarrollo y el tipo de producto de software influyen en la selección del proceso de mejora. Además, es importante tomar en cuenta como entradas las necesidades de calidad de la organización.

4. **Evaluación del sistema final.**

   Para cuantificar las mejoras obtenidas con la implementación del proceso de mejora (PSP) evaluaremos nuevamente el atributo de calidad del producto de software. Esperamos que el modelo resultante se acerque al comportamiento del modelo ideal. De cualquier forma, dependiendo de los requerimientos de calidad esperados se decidirá el número de iteraciones *Evaluación-Mejora* necesarias.

5. **Análisis de los resultados y conclusiones.**

   Después de aplicar el proceso de mejora un número determinado de veces, esperamos obtener la calidad deseada en el producto de software. En esta fase, se llevará a cabo un análisis de las evaluaciones obtenidas del producto de software, así como la identificación y la cuantificación de la mejoras obtenidas durante el proceso. Así mismo, se llevará a cabo una comparación de los objetivos planteados contra los objetivos alcanzados. Este análisis deberá registrarse en bitácoras, ya que servirá para mejorar la calidad de productos de software futuros.

4.2.2 Técnicas de modelado

Existen diversas técnicas de modelado como son las Cadenas de Markov, Redes de Petri, Verificación de modelos (Model Checking) y los Modelos Estadísticos .

- *Cadenas de Markov.* Es una serie de eventos en la cual la probabilidad de que ocurra un evento depende del evento inmediato anterior. Sin embargo no se recomiendan para modelar *fiabilidad*, debido a que no distinguen entre los fallos seguros y los no seguros, por otro lado no toman en cuenta el proceso de restauración o reparación en un sistema [5].

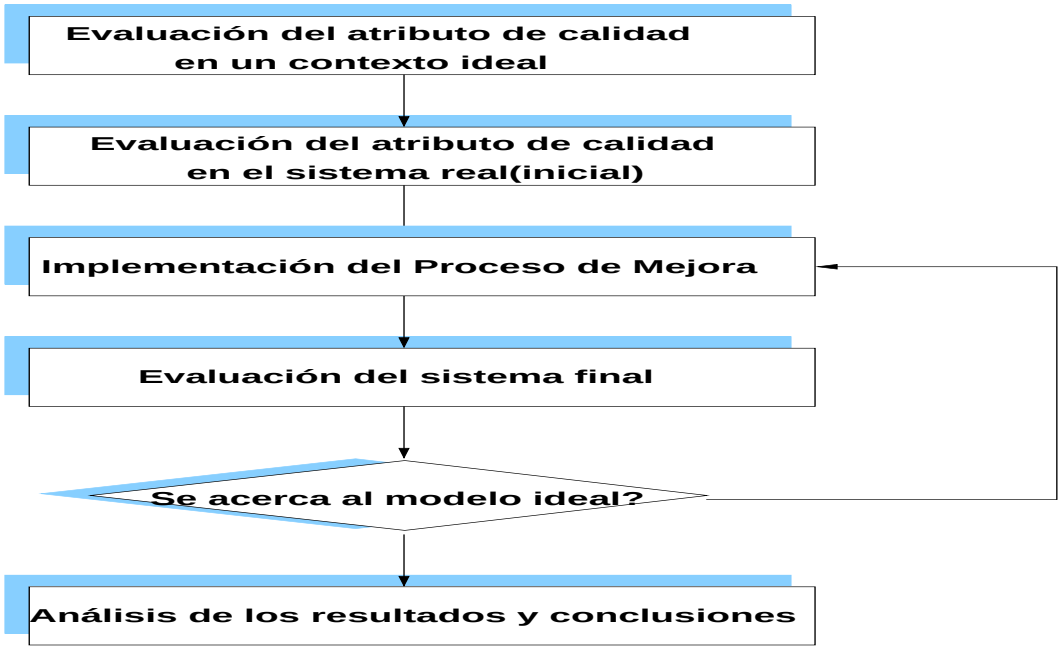

Figura 4.1: Etapas de la metodología

- *Redes de Petri.* Es una red dirigida bipartita que incluye un conjunto de lugares P (places), un conjunto de transiciones T (transitions) y un conjunto de arcos dirigidos A. Un lugar p es una entrada para una transición t si existe un arco incidente a la transición*(p,t)*. Un lugar p es una salida de una transición t si existe un arco incidente de transición al lugar *(t,p)*. el conjunto de arcos puede ser particionado en el conjunto de arcos de entrada a transiciones A_i y el conjunto de arcos de salida de transiciones A_o.

  Las Redes de Petri pueden contener tokens, usualmente se denota como Red de Petri con marcado. Su estado está definido por el número m_i de tokens contenidos en cada lugar t_i. El estado de la red de petri se llama marcado y su denotación es la siguiente: $M = m_1, m_2, ..., m_n$

  $PN = (P, T, A, M_o)$

  $P = p_1, p_2...p_n$

  $T = t_1, t_2...t_n$

  $A_i \subset P \times T$

  $A_o \subset T \times P$

  $A = A_1 \cup A_o$

  $M = m_o1, m_02, ..., m_on$

- *Verificación de modelos (Model Checking).* Es un modelo que se basa en la especificación formal de los módulos o programas que son representados por sistemas de transición con estados. Este modelado puede aplicarse a sistemas grandes, sin embargo a medida que el sistema crece, el espacio de estados se vuelve demasiado grande [9].

- *Modelos Estadísticos.* Es un modelo orientado al estudio de poblaciones (por ejemplo métricas de software). El cual se puede expresar como una relación entre entidades a las que se les representa mediante funciones de distribución. Lo que lleva a la obtención de una ley de probabilidad que representa el comportamiento de los atributos de dicha población [2].

Nuestro problema radica en estudiar el comportamiento de la *fiabilidad* en los sistemas de Información en Internet. En la gran mayoría de los casos este tipo de sistemas están conformados por un gran número de módulos. La tendencia al crecimiento en cuanto al tamaño del sistema se ve influenciada por el volumen de información que normalmente manejan. En la mayoría de los tipos de modelos, el tamaño del sistema a analizar viene a ser una restricción. Por otro lado es importante contemplar que la metodología se enfoca a organizaciones, por lo cual el tipo de modelo seleccionado no debe ser demasiado complicado. De acuerdo al anterior análisis de técnicas de modelado concluimos que la opción que mejor se adapta a nuestro objetivo es la obtención de Modelos Estadísticos.

4.2.3 Empleo de modelos estadísticos

Los modelos estadísticos tienen su base en el estudio de poblaciones. Una población es un conjunto limitado de individuos o elementos de la misma especie sometidos a un estudio estocástico. Estos elementos tienen propiedades fundamentales en común, que son la base para clasificar a la población. En nuestro caso, el objeto de estudio (o la población) son los atributos de calidad de software. Las métricas de software permiten evaluar a la población. Nuestro interés consiste en estudiar la tendencia del comportamiento de los atributos de calidad (en particular la *confiabilidad*) del producto de software. Para lo cual nos proponemos evaluar y analizar el comportamiento de los atributos de calidad asociados a un producto de software, mediante el estudio de su distribución estadística. La evaluación del comportamiento de estos atributos la realizaremos con métricas de software y sus resultados los describiremos mediante histogramas. El problema fundamental consiste en reemplazar el histograma por una curva teórica ó modelo que represente una ley de

probabilidad. Esta ley de probabilidad será la imagen definitiva de la población de las métricas de software estudiadas y permitirán describir el comportamiento del atributo de calidad.

El procedimiento para obtener el modelo estadístico consiste de los siguientes pasos.

1. *Escoger el tipo de ley de probabilidad que corresponde mejor asociar a la población.* Esta ley podría tener un origen puramente empírico; por ejemplo, el histograma de los errores que ocurren durante un lapso de tiempo en la operación de un producto de software. Ejemplos de estas leyes podrían ser, la distribución de Weibull, la Normal, la de Poisson, o la Gamma.

2. *Determinar los parámetros contenidos en la ley escogida.* Una ley contiene parámetros que dependen de la población estudiada. La modelación proporciona las expresiones para obtener el valor de estos parámetros para todos los modelos.

3. *Comparar la ley escogida.* Una vez escogido el tipo de ley y valorados los parámetros numéricos, debemos verificar que la ley esté de acuerdo con la población estudiada. El resultado de la comparación puede ser favorable o desfavorable. Si es favorable, expresaría que la ley escogida representa fielmente a la población estudiada. De lo contrario, sería necesario buscar otra ley que represente mejor el comportamiento de la población.

4.2.4 Proceso de evaluación y modelación

El proceso para realizar la evaluación del producto de software, el análisis de los resultados y la modelación se compone de los siguiente pasos.

1. **Determinación de las condiciones iniciales.**

   Cada fase tiene su enfoque para valorar las condiciones del ambiente de operación del producto de software. Tomar en cuenta estas condiciones es básico para el proceso de evaluación. Por ejemplo, algunas de estas condiciones podrían ser: (a) las entradas del sistema, (b) sus restricciones y (c) los servicios que proporciona el sistema.

2. **Selección del atributo de calidad y de sus métricas de software.**

   Es fundamental definir el atributo de calidad de software que pretendemos estudiar. En la selección del atributo influyen las necesidades prioritarias de cada organización en sus productos de software. Una vez seleccionado el atributo de calidad es necesario verificar su correspondiente métrica. En la selección de las métricas influyen los siguientes factores:

Tipos de métricas. Las métricas que se van a emplear dependen del atributo de calidad a evaluar y de las condiciones de desarrollo y operación del sistema. Los atributos de software más comunes son los mostrados en las figuras 2.3 y 2.4. Para cada atributo es posible que existan varios tipos de métricas de software que se pueden aplicar. En algunos casos las métricas de software existentes no son aplicables al ambiente de operación del proyecto, o estas son difíciles de obtener. En estos casos es posible implementar nuevas métricas que están de acuerdo a las normas de calidad de la organización. Compañías como Motorola, IBM y Hewlett Packard han desarrollado métricas que se adecuan a su marco de producción [15].

Condiciones de desarrollo. Las condiciones de desarrollo del sistema permiten describir:

- *El Proceso de desarrollo de software utilizado.* Existen distintos procesos de desarrollo como son: (a) Proceso de desarrollo en cascada, (b) Proceso evolutivo, (c) Proceso en espiral, (d) Prototipado ó (e) Reutilización de componentes.
- *El personal involucrado en el desarrollo.* Dependiendo del dominio de la aplicación es necesario verificar que el personal sea especialista en el área de dominio.
- *El presupuesto asignado al desarrollo del producto de software.* El presupuesto tiene un impacto directo en la selección del personal, de los componentes utilizados y del tiempo de desarrollo del producto de software.
- *Las condiciones de calidad impuestas por la organización.*

Tiempo de evaluación. En la selección de las métricas también influyen las unidades de tiempo necesarias para realizar las mediciones. Estas pueden unidades pueden ser unidades de tiempo de CPU, días, semanas, meses o incluso años.

3. **Proceso de Medición.**

   La medición del software se refiere a derivar un valor numérico para algún atributo de un producto de software. El proceso de medición es una actividad clave, ya que de este depende que los resultados puedan ser fiables y fáciles de evaluar.

   Los pasos a seguir en este proceso son los siguientes.

   a. Seleccionar los componentes a evaluar.
   b. Medir las características de los componentes que forman parte del sistema o el sistema de forma global, con las métricas de software establecidas.

c. Identificar las mediciones anómalas.

d. Analizar los componentes anómalos.

4. **Evaluación de los resultados y selección del modelo.**

   La evaluación del proceso de medición se realiza mediante el estudio de su distribución estadística. Los resultados de ésta evaluación permiten analizar el conjunto de datos obtenidos mediante gráficas conocidas como histogramas (los cuales pueden obtenerse mediante herramientas como *Arena*). Los histogramas permiten representar los valores de la métrica contra su frecuencia. Estos histogramas representan una aproximación al tipo de modelo que mejor se ajusta al comportamiento de los resultados obtenidos. El histograma puede se reemplazado por una ley de probabilidad que mejor represente este comportamiento. Esta ley de probabilidad será el reflejo del comportamiento de la población de métricas estudiadas. La ley escogida indicaría si la evaluación fue correctamente realizada.

5. **Estimación de los parámetros del modelo.**

   De acuerdo a la Sección 4.2.3, esta actividad es parte fundamental en la obtención del modelo. Para la obtención del modelo se usan diversas técnicas de métodos numéricos, como el método de los MLEs (maximun - likelihood estimators), el método de los mínimos cuadrados, regresión polinomial, entre otras. En ocasiones, es necesario aplicar varios métodos para llegar a resultados confiables y dependiendo del parámetro a estimar algunos métodos son mas adecuados que otros.

6. **Sustitución de los parámetros obtenidos y graficación del modelo resultante.**

   Una vez obtenidos los parámetros resultantes, estos son graficados a fin de observar su tendencia. La gráfica obtenida representa el comportamiento del atributo de calidad.

7. **Validación del modelo escogido.**

   Con los parámetros obtenidos debemos verificar que el modelo, tal y como fue construido está de acuerdo con la población de métricas estudiada. Por lo tanto en este proceso, se realiza una comparación de los valores que se obtienen con el histograma contra los resultados que se obtienen con el modelo.

8. **Realización de las predicciones del atributo de calidad.**

En este proceso, las predicciones del atributo de calidad se realizan mediante la observación del modelo (o ley de probabilidad) obtenido. Con este modelo, además de representar el comportamiento de los atributos de calidad, es posible también predecir el comportamiento a futuro de estos atributos.

4.3 Caso de Estudio

En esta sección describiremos mediante un caso de estudio las dos primeras fases de la metodología. En nuestro caso de estudio, evaluaremos un sistema de inscripciones vía Internet *(SIV)* para una Universidad. La arquitectura del sistema consiste en una plataforma Linux (*Redhat v.8*), un manejador de bases de datos (*MySQL [23]*) y un servidor Web (*Apache [24]*). El lenguaje utilizado para las interfaces de usuario fue (*PHP [25]*). El número de líneas de código utilizadas en la implementación del sistema fue de 3240. El tiempo empleado para el desarrollo del sistema y de su documentación ha sido de 6 meses aproximadamente. El diagrama a bloques de sistema se puede observar el la figura 4.2.

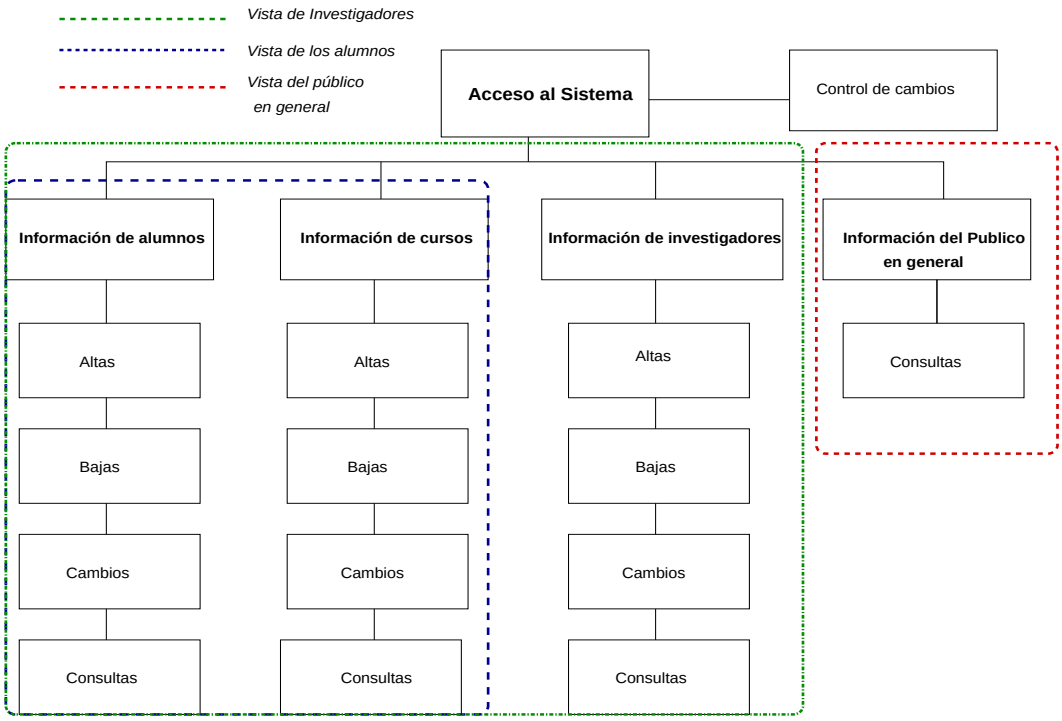

Figura 4.2: Módulos que conforman el SIV

4.3.1 Descripción del sistema

Se trata de un Sistema de Inscripciones vía Internet para una Universidad, en donde existen tres vistas principales: (a). Investigadores/Profesores, (b) Alumnos, (c) Público en general. El acceso al sistema es con nombre de usuario y password. Cada usuario del sistema tendrá acceso a distintos servicios que proporciona el sistema. En general se pretende que el sistema provea los servicios de altas de cursos calificaciones y alumnos, con sus respectivas bajas y modificaciones.

4.3.2 Fase 1: Determinación del comportamiento ideal de *fiabilidad*

En esta fase, se evaluará y modelará el comportamiento del sistema bajo condiciones ideales, aplicando el proceso definido en la Sección 4.2.

1. **Determinación de las condiciones iniciales para el caso ideal.** Las condiciones para la simulación son las siguientes:

   El tiempo entre arribos de los usuarios al sistema se presenta siguiendo una distribución exponencial con una media $\mu = 5$ unidades de tiempo. El tiempo que permanece cada usuario utilizando el sistema posee una distribución normal con una media $\mu = 5$ unidades de tiempo y una varianza $\sigma = 3$. La modelación del arribo de usuarios se hace mediante colas, para lo cual se utilizó un servidor de colas del tipo M/M/1 en el pool del servidor web [1] con una probabilidad de $(n+1)^{-1}$, donde n representa el tamaño actual de la cola (número de usuarios en el sistema). La condición de salida del sistema para cualquier usuario se presenta, (a) cuando se genera un error del usuario durante la operación del sistema y (b) cuando el usuario solicita su salida del sistema.

   Por razones de *fiabilidad*, en cuanto a la capacidad del servidor Web y la del manejador de la Base de Datos, el sistema puede trabajar adecuadamente hasta con k usuarios (donde $k = 100$). Dependiendo de su vista, los usuarios pueden realizar solo x tipos de transacciones, con t unidades de tiempo para realizar cada transacción. Donde x depende de la vista y t sigue una distribución normal con una media $\mu = 5$ y una varianza $\sigma = 3$. El problema consiste en determinar el número de errores que se generan en el sistema en un lapso de tiempo

[1] De acuerdo a la Notación de Kendal [3], en la notación A/B/s, A se refiere a la distribución de los tiempos entre arribos, B es la distribución de los tiempos de servicio y s representa el número de servidores. Las distribuciones más comunes son M (Markov, o exponencial), G (general) y D (determinista, o constante).

determinado. El lenguaje de programación empleado para la simulación fue Java.

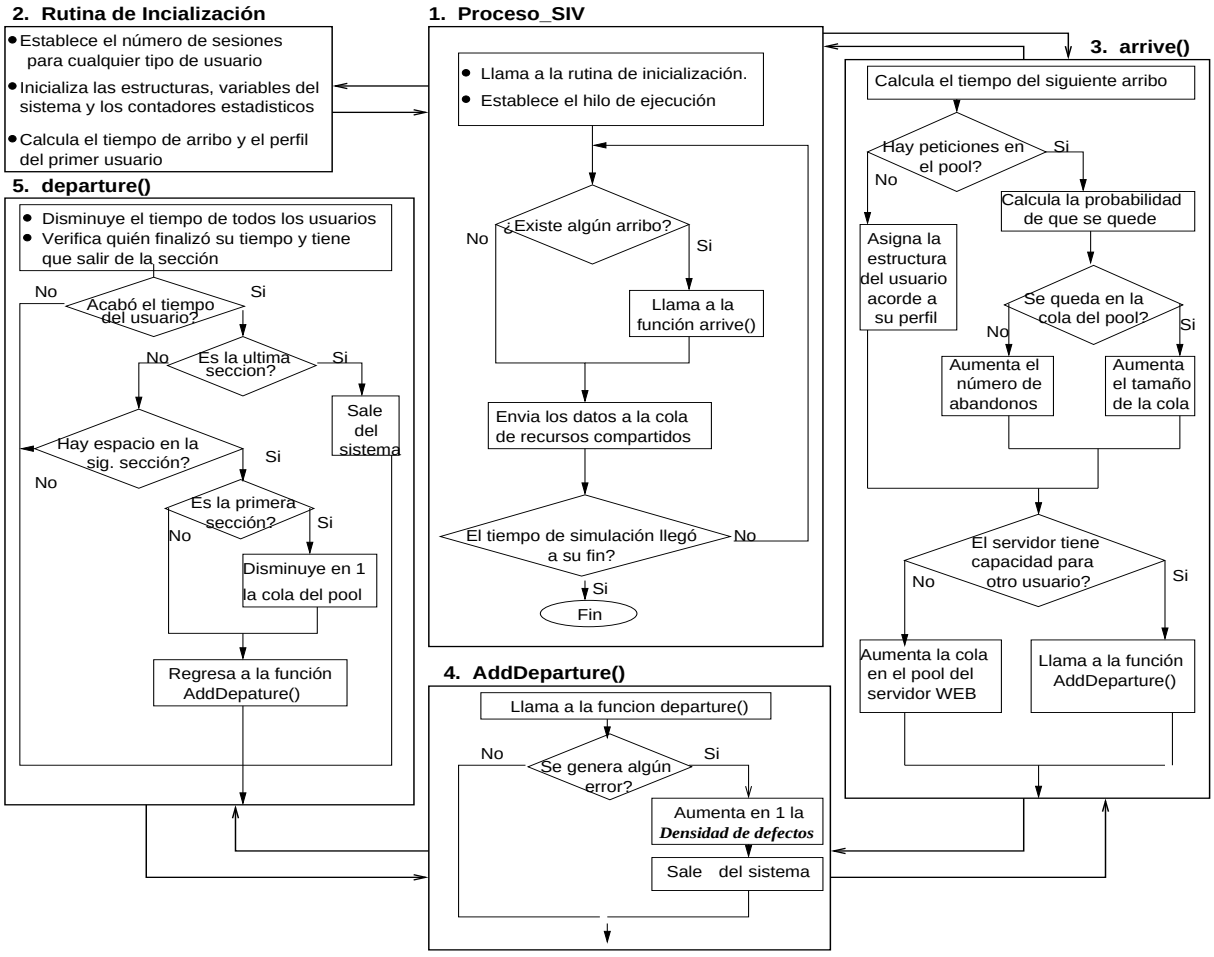

Figura 4.3: Diagrama a bloques del funcionamiento del Productor

En la simulación se emplea un modelo *productor − consumidor*, en donde la información se maneja mediante una cola de recursos compartidos. El productor tiene la función de generar los datos de salida para el consumidor. La función del consumidor es actualizar y graficar el comportamiento de la simulación cada unidad de tiempo, de acuerdo a los datos que el productor le envía a través de la cola de recursos compartidos. La interfaz gráfica que se muestra en la figura 4.4 se desarrolló con el fin de visualizar el comportamiento del productor-consumidor.

El diagrama a bloques de la operación del productor se muestra en la figura 4.3. En este diagrama se pueden apreciar 5 módulos. En el módulo principal, *Proceso del SIV*, realiza el ciclo de ejecuciones de la simulación. El segundo módulo, *Rutina de Inicialización*, tiene

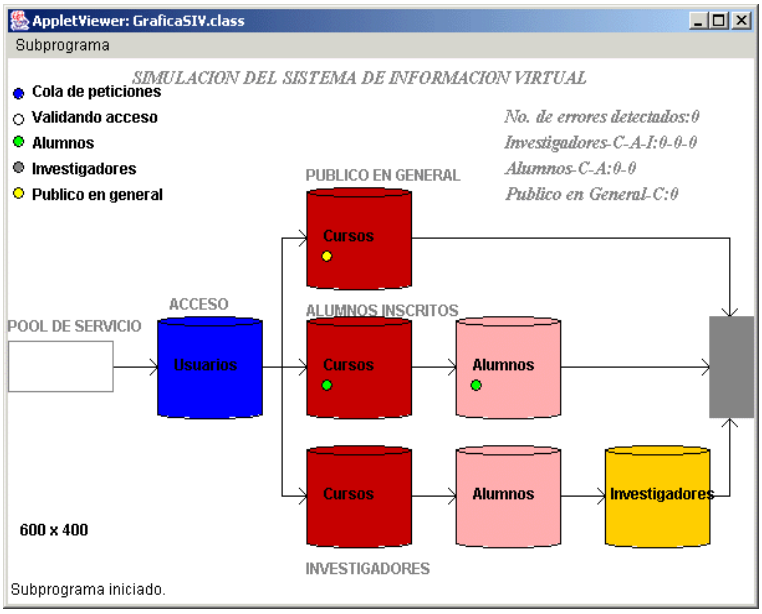

Figura 4.4: Interfase gráfica de la simulación del SIV

la función de inicializar las estructuras, los contadores estadísticos y de programar el primer arribo. El tercer módulo, *arrive()*, calcula el tiempo de arribo de los usuarios, verifica si los usuarios se quedan en la cola del pool del servidor Web y verifica si el sistema tiene capacidad para más usuarios. En el caso de que el sistema tenga suficiente capacidad, se le permite el acceso al usuario y se realiza una llamada a la función *Addeparture()*. En el cuarto módulo, *AddDeparture()*, se realiza un llamada a la función *departure()* y se verifica la generación de errores durante la operación simulada. En el caso de que hallan existido errores, se aumenta la *densidad de defectos* y se le programa al usuario su salida del sistema. El quinto módulo, *departure()*, se encarga de programar los tiempos de salida para cada usuario para la sección o parte del sistema que está utilizando.

2. **Selección del atributo de calidad y de sus métricas de software.** Como se ha determinado con anterioridad es la *fiabilidad*.

   Tipos de métricas. En nuestro caso nos interesa medir el atributo de *fiabilidad* en la operación del producto de software. Las métricas que son posibles utilizar para la *fiabilidad* son[15]:

   - **densidad de defectos**. La *densidad de defectos* es una métrica de calidad que se define como el número de errores que ocurren durante un lapso de tiempo determinado.

- **media de ocurrencia de fallos.** Se refiere al promedio del tiempo que tarda en producirse un fallo durante la operación de un producto de software.

En nuestro caso utilizaremos la métrica de *densidad de defectos*. En este punto el tipo de defectos que estamos contemplando son los de funcionamiento del sistema. Es decir aquellos que no cumplen con la especificación de los requerimientos del sistema. La lista de defectos que se localizaron esta en la tabla **??**.

Condiciones de desarrollo. Para el caso ideal, como se comenta anteriormente, se desarrolló un simulador del sistema de Inscripciones por Internet en lenguaje Java. Las condiciones de operación del simulador se describen en la primera parte de esta fase.

Tiempo de evaluación. El tiempo para cada evaluación es el tiempo que tarda cada simulación. Cada evaluación se llevo a cabo en 100 unidades de tiempo (lo cual se llevó a cabo en aproximadamente 30 segundos de tiempo de ejecución). Se llevaron a cabo 70 evaluaciones.

3. **Proceso de medición.**

   a. **Selección de los componentes a evaluar.** La simulación se diseñó para medir todos los componentes que forman al sistema.

   b. **Medir las características de los componentes con las métricas de software.** Dado que la métrica fue la *densidad de defectos*, la simulación se programó para que en un marco de tiempo de 100 unidades se contabilizara en número de defectos que aparecieron en la ejecución de la simulación del sistema. Este número de defectos se contempla para cada una de los perfiles y secciones del sistema.

   c. **Identificar las mediciones anómalas.** Por el hecho de tratarse de una simulación que trabaja con condiciones ideales se determinó que todos los resultados que la simulación reportó eran correctos. En este caso no se obtuvieron mediciones anómalas.

   d. **Identificar los componentes anómalos.** Para el caso ideal no se presentaron componentes anómalos.

4. **Evaluación de los resultados y selección del modelo.**

   El resumen de los resultados para nuestro proceso de evaluación y selección del modelo se muestra en la tabla 4.1. Los elementos que integran el resumen son los siguientes: Densidad

D. Defec	Frec	x	Den. Prob.	Dist. Acum.
0	16	0.00	0.222	0.222
1	23	1.00	0.319	0.542
2	16	2.00	0.222	0.764
3	12	3.00	0.167	0.931
4	3	4.00	0.0417	0.972
5	0	5.00	0.000	0.972
6	1	6.00	0.0139	0.986
7	0	7.00	0.000	0.986
8	1	8.00	0.0139	1.00

Tabla 4.1: Resumen de los resultados obtenidos durante las evaluaciones del caso ideal

de defectos (D.Defec), frecuencia (Frec), valor asignado en el eje x (x),la densidad de la probabilidad (Den. Prob) y los valores de la distribución acumulativa (Dist. Acum). El histograma que corresponde a estos datos se muestra en la figura 4.5. Mediante el histograma apreciamos que la tendencia de los resultados se aproxima a una distribución (o curva) de Weibull. La curva de Weibull es un meta-modelo [3], en donde α y β son sus parámetros.

5. **Estimación de los parámetros del modelo.**

   Los parámetros del modelo se obtienen siguiendo el procedimiento descrito en la Sección 4.2), de la siguiente forma.

   Escoger el tipo de ley de probabilidad que nos proponemos asociar al histograma.

$$f(x) = \alpha\beta^{-\alpha}x^{\alpha-1}e^{-(x/\beta)^{\alpha}} \tag{4.1}$$

   Evaluar los parámetros contenidos en la ley escogida. Del análisis de la ecuación 4.1, se obtuvo en [3] mediante estimadores de máxima verosimilitud (maximum-likelihood estimators MLEs), las siguientes ecuaciones que permiten calcular los valores de α y β.

$$\frac{\sum_{i=1}^{n} X^{\alpha} lnX_i}{\sum_{i=1}^{n} X^{\alpha}} - \frac{1}{\alpha} = \frac{\sum_{i=1}^{n} lnX_i}{n} \tag{4.2}$$

$$\beta = \left(\frac{\sum_{i=1}^{n} X_i^{\alpha}}{n}\right)^{1/\alpha} \tag{4.3}$$

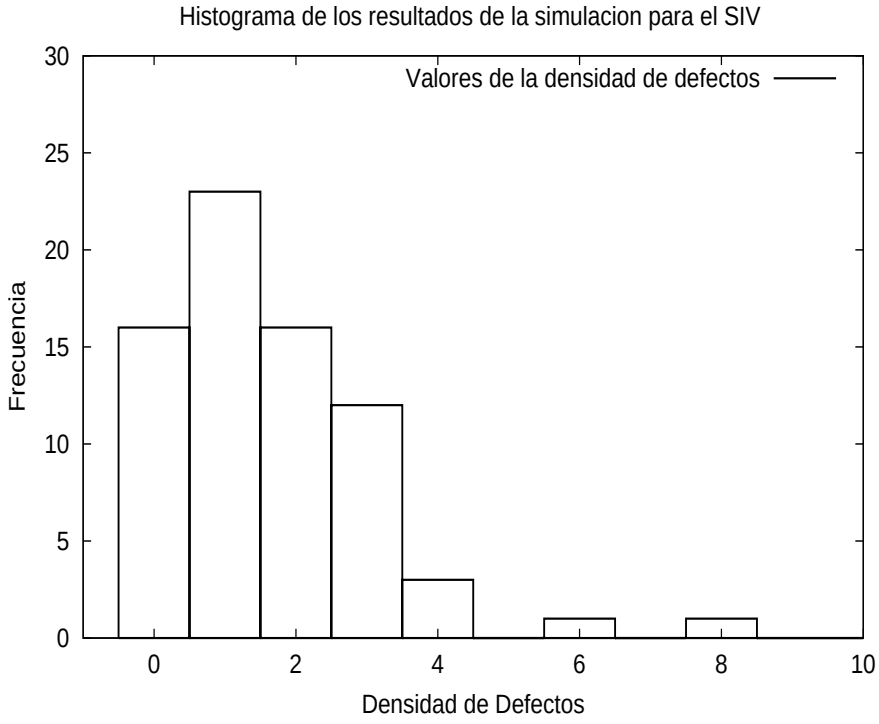

Figura 4.5: Histograma para 70 simulaciones.

La ecuación 4.2 se resuelve mediante el método de *Newton-Rapson*, mientras que la ecuación 4.3 se obtiene de manera directa conociendo el valor de α. En ambos casos $\boldsymbol{n}$ se refiere al numero de evaluaciones, en nuestro caso experimentos con la simulación que estamos contemplando. Los valores obtenidos de los parámetros a partir de las ecuaciones 4.2 y 4.3 son:

$$\alpha = 1.54 \text{ y } \beta = 2.37.$$

6. **Sustitución de los parámetros obtenidos graficación del modelo ideal f_i.**

   De acuerdo a los valores de α y β antes descritos la función de probabilidad se calcula de la siguiente forma.

$$f_i(x) = \begin{cases} 0.26478x^{0.54}e^{-(x/2.37)^{1.54}} & \text{Si } x \geq 0 \\ 0 & \text{En otro caso} \end{cases}$$

   La gráfica del Comportamiento de la *fiabilidad* de acuerdo al modelo ideal f_i se presenta en la figura 4.6.

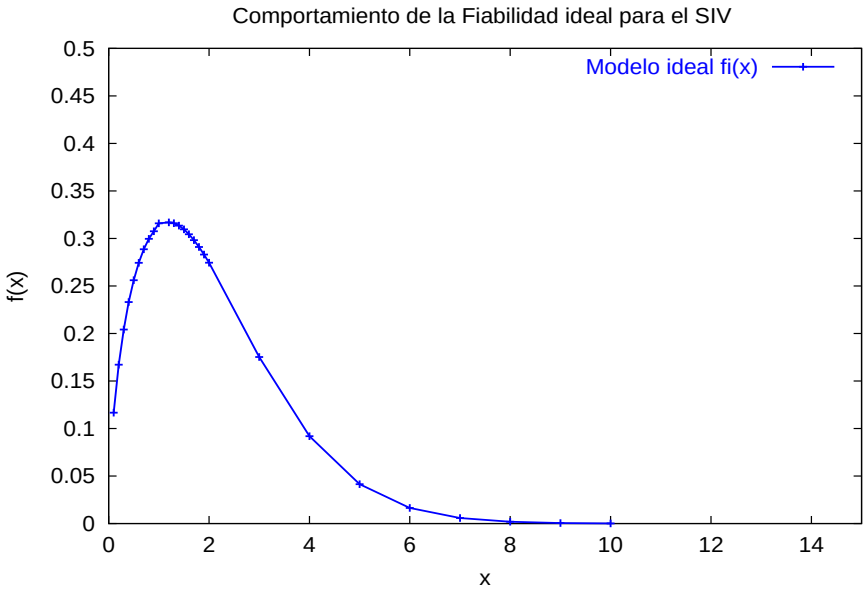

Figura 4.6: Comportamiento de la densidad de defectos (*fiabilidad*) en el SIV

7. **Validación del modelo obtenido.**

   De acuerdo a la modelación estadística, la curva de Weibull permite representar el tiempo de ocurrencia de fallos de alguna pieza de un sistema ó equipo. Por lo cual, el hecho de que el modelo obtenido siga una curva de Weibull nos encontrar un modelo para nuestro proceso. En otras palabras, podemos decir que la métrica *Densidad de Defectos* es la adecuada para representar el comportamiento de la *fiabilidad* del producto de software.

8. **Realización de las predicciones de la *fiabilidad.***

   De acuerdo a la información proporcionada por el Modelo de la figura 4.6, es posible observar que la tendencia de la curva que describe características de la densidad de defectos, como por ejemplo que presenta sus máximos en 1 y 2. Podemos establecer que la densidad de defectos a la que aspiramos en el caso real oscila entre 1 y 2. Con un rango de 0 errores como mínimo y 8 como máximo. De acuerdo a la información obtenida en la tabla 4.2, es posible observar que el perfil que acumula un mayor número de errores es el que corresponde al de *Alumnos*. El cual tiene el promedio mayor de ocurrencia de errores. Para esta vista la sección donde se presenta la incidencia mas alta de acuerdo a sus valores promedio y máximos (tabla 4.2) es la que corresponde a la información de los cursos. Para la vista de los *Investigadores* es posible observar que la sección con un mayor promedio y un número máximo es la de los alumnos

Identificador	Perfil	Seccion	No. de errores	Promedio	Máximo	Mínimo
1.1	Investigadores	Cursos	9	0.125	1	0
1.2		Alumnos	13	0.180	2	0
1.3		Investigadores	8	0.111	2	0
2.1	Alumnos	Cursos	46	1.260	4	0
2.2		Alumno	32	0.876	3	0
3	Público en General	Cursos	1	0.013	1	0

Tabla 4.2: Resumen de los resultados para el caso ideal

(tabla 4.2). La sección que presenta un promedio menor en esta vista es la que maneja la información de los investigadores. También es posible observar que el comportamiento de la vista del *Público en General* tiene una incidencia de errores baja. Esto se ve reflejado en su promedio y valor máximo. Tiene los valores menores de todas las vistas del SIV. De acuerdo a lo anterior podemos estimar la vista que puede generar una mayor número de errores es la correspondiente a los *Alumnos*. Con respecto a las secciones , es posible determinar que los promedios mas altos en incidencia de errores se dan en las secciones de cursos y alumnos. De acuerdo a los requerimientos del sistema, la intersección de estas dos secciones es la de Cursos x Alumno (inscripción de los alumnos a los cursos que se imparten en el colegio).

4.3.3 Fase 2: Evaluación de la *fiabilidad* en el sistema real siguiendo proceso de evaluación y modelación

1. **Determinación de las condiciones iniciales para el caso real.**

   Las condiciones del proceso fueron las siguientes. Las evaluaciones fueron hechas por un sólo usuario, el cual tomó el rol de tres tipos de vistas: alumnos, investigadores y público en general. La forma de realizar las transacciones sobre el SIV fue de manera aleatoria para cada rol, mediante una matriz de pruebas. La matriz de pruebas contenía los distintos servicios y los datos requeridos a accesar del sistema. Esta matriz estuvo compuesta de opciones válidas y opciones no-válidas. Los datos de prueba fueron tomados de manera aleatoria de la matriz de pruebas, para cada evaluación.

   Las condiciones de operación en el sistema fueron las siguientes:

   - El tiempo entre arribos (de cada usuario) fue de 100 minutos.

- El servidor Web operó con un solo procesador de 500 Mhz.
- La capacidad del servidor Web y la del manejador de la Base de Datos con la cual pueden operar adecuadamente el sistema (de acuerdo a los estándares del proveedor *redhat*) es de hasta 100 usuarios.
- Dependiendo de su vista, los usuarios sólo pueden realizar un número limitado de transacciones.

Los resultados obtenidos en cada evaluación se almacenaron en bitácoras y se promediaron con el fin de graficar los histogramas.

2. **Selección de los atributos a medir y de sus métricas.**

   Tipos de métricas. Al igual que en caso ideal, el atributo a evaluar es la *fiabilidad* y la métrica para dicho atributo será la *densidad de defectos*.

   Condiciones de desarrollo. En el desarrollo del SIV seguimos el modelo de cascada. En el proceso desarrollo de este sistema intervinieron dos personas especialistas en el desarrollo de sistemas en Internet y el tiempo de desarrollo fue de 6 meses.

   Tiempo de evaluación. El tiempo para la evaluación de cada experimento fue de 100 unidades de tiempo (lo cual se llevó a cabo en un lapso de 100 minutos). El número de experimentos para el caso real fue de 70. Los experimentos fueron llevados a cabo en 3 semanas. El número de evaluaciones en el caso ideal y en el caso real es de 70 debido a que el tiempo destinado para estas evaluaciones no rebasa las 3 semanas.

3. **Proceso de medición.**

   a. **Selección de los componentes a medir.**

      Durante la operación del sistema real se midieron los componentes que corresponden a las secciones de Cursos, Alumnos, Investigadores y la sección de Cursos por alumnos que integran el sistema.

   b. **Medir las características de los componentes con las métricas de software.**

      En este punto la métrica utilizada fue la *densidad de defectos*. Durante cada experimento se contabilizó el número de defectos ocurridos.

c. **Identificar las mediciones anómalas.** Durante la ejecución del sistema no ocurrieron mediciones anómalas.

d. **Identificar los componentes anómalos.**

Los errores que encontramos con mayor frecuencia en los componentes seleccionados durante la operación del sistema son los siguientes:

Alta de CURSOS:

- No reconoce automáticamente la sección a la que pertenece la persona que da de alta el curso.
- No valida totalmente la información antes del proceso de inserción en la Base de Datos.

Alta de cursos por alumno:

- Permite registrar más de 4 cursos por alumno.

Consulta de cursos por alumno:

- No se presenta la lista completa de los alumnos que asesora un investigador/profesor.
- No se presenta la información completa de los datos del alumno.

Alta de alumnos,cursos e investigadores:

- No valida totalmente la información antes del proceso de inserción en la Base de Datos.

4. **Evaluación de los resultados y selección del modelo.**

El resumen de los datos obtenidos durante las 70 evaluaciones se presenta en la tabla 4.3. Al igual que en el caso ideal en la tabla se registran los siguientes valores: Densidad de defectos (D.Defec), frecuencia de ocurrencia (Frec), la representación en x (x), la densidad de probabilidad (Den. Prob.) y la distribución acumulativa (Dist. Acum.). El histograma mostrado en la figura 4.7, nos muestra que la tendencia que presentan los datos describe una curva de Weibull.

5. **Estimación de los parámetros del modelo.**

La función de densidad utilizada para modelar el caso real es la siguiente.

$$f(x) = \alpha\beta^{-\alpha}x^{\alpha-1}e^{-(x/\beta)^{\alpha}} \tag{4.4}$$

D. Defec	Frec	x	Den. Prob.	Dist. Acum.
0	5	1.00	0.714	0.0714
1	0	2.00	0.000	0.0714
2	20	3.00	0.286	0.357
3	15	4.00	0.214	0.571
4	8	5.00	0.114	0.686
5	10	6.00	0.143	0.829
6	1	7.00	0.143	0.843
7	2	8.00	0.0286	0.871
8	1	9.00	0.0143	0.886
9	1	10.00	0.0143	0.900
10	2	11.00	0.0286	0.929
11	0	12.00	0.000	0.929
12	0	13.00	0.000	0.929
13	5	14.00	0.0714	1.00

Tabla 4.3: Resumen de los resultados obtenidos durante las evaluaciones del caso real

en donde, los parámetros α y β obtenidos son los siguientes.

$$\alpha = 1.54 \text{ y } \beta = 5.17$$

Al igual que en caso ideal, los valores de α y β se resuelven de acuerdo a las ecuaciones 4.2 y 4.3.

6. **Sustitución de los parámetros obtenidos f_r y graficación del modelo real.**

   Sustituyendo los valores de α y β en la función del modelo real f_r da la siguiente expresión.

$$f_r(x) = \begin{cases} 0.0796x^{0.54}e^{-(x/5.17)^{1.54}} & \text{Si } x \geq 0 \\ 0 & \text{En otro caso} \end{cases}$$

   La gráfica del comportamiento de la *fiabilidad* de acuerdo al modelo real f_r se presenta en la figura 4.8.

7. **Comparación del modelo obtenido.**

   Es posible observar que el modelo obtenido para el caso real sigue una distribución de Weibull, la cual permite representar el comportamiento de la *fiabilidad* real.

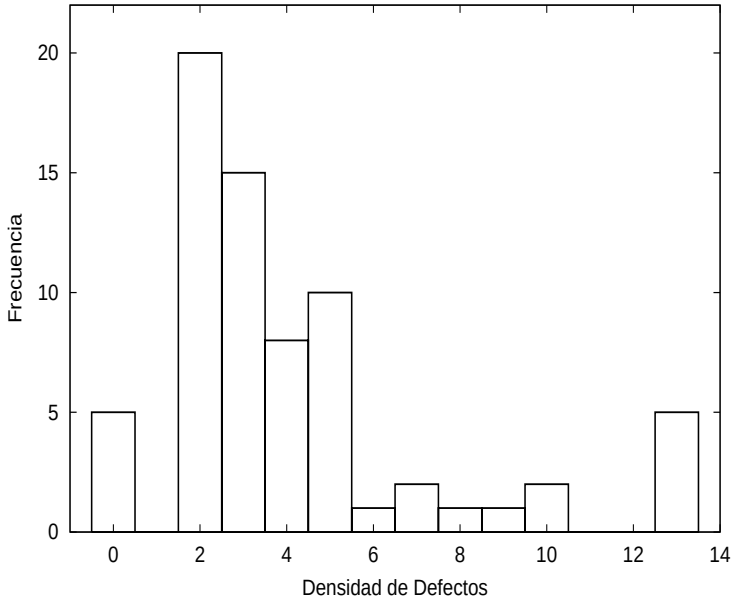

Figura 4.7: Histograma para 70 mediciones de errores

8. **Realización de las predicciones de la *fiabilidad*.**

   En la figura 4.8 se observa la comparación de los modelos resultantes (ideal y real) de las evaluaciones del caso de estudio. Los parámetros α y β representan el grado de aproximación que existe entre los modelos ideal y real. Para obtener un producto de software con un buen nivel de *fiabilidad* se debe cumplir lo siguiente.

   $$\alpha_r \rightarrow \alpha_i \text{ y } \beta_r \rightarrow \beta_i$$

   Es posible observar en la figura 4.8 que el comportamiento obtenido del modelo real está alejado del comportamiento del modelo ideal. Esto se puede derivar de la observación de que:

   $$\alpha_r \neq \alpha_i \text{ y } \beta_r \neq \beta_i \Rightarrow f_r(x) \neq f_i(x).$$

   Por lo tanto, concluimos que son necesarias mejoras en el producto de software, para lo cual será necesaria la ejecución de la segunda parte de la metodología, que consiste en el proceso de mejora PSP.

De acuerdo a la información obtenida en la tabla 4.4, es posible observar que el perfil que acumula un mayor número de errores es el correspondiente a los *Alumnos*. Este comportamiento se esperaba,

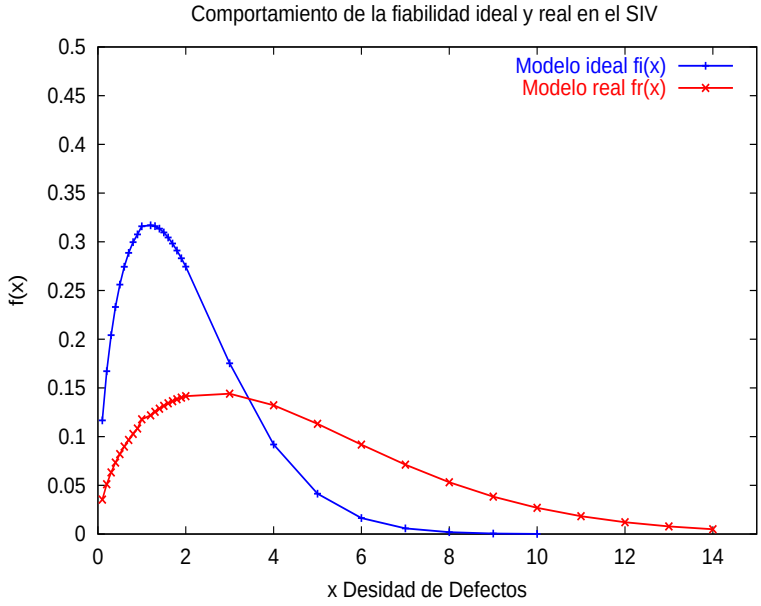

Figura 4.8: Gráfica del modelo ideal y real para el caso de estudio

de acuerdo a los resultados obtenidos en la evaluación del caso ideal. Sin embargo el promedio de ocurrencia de errores es mucho mayor a lo esperado en el caso ideal. La sección donde se presenta la incidencia más alta de acuerdo a sus valores promedio y máximos es la que corresponde a la información de los cursos. De manera similar que en el caso ideal, en la evaluación real no sólo resalta para el perfil de *Alumnos*, sino que también lo hace en el perfil de los *Investigadores* y en *Público en General*. En la vista de *Investigadores* la distribución de errores esperados es diferente del caso ideal. La sección de mayor promedio y número máximo es la de los cursos. La sección que presenta un promedio menor en esta vista es la que maneja la información de los investigadores. Este último punto corresponde a lo esperado en el caso ideal. Es posible observar que el comportamiento del público en general en incidencia de errores es baja . Sin embargo al igual que en las otras vistas el número de incidencia de errores es mucho mayor en el caso real.

De acuerdo a lo anterior confirmamos que la vista que genera un mayor número de errores es la correspondiente a los *Alumnos* y la sección mas problemática es la de Cursos x alumnos (inscripción de los alumnos a los cursos que se imparten). La localización de errores reales para el SIV se realiza en la tabla 4.5. Donde se relaciona el identificador del caso ideal, el perfil donde se presenta el error, la sección, la transacción que activa el error, el tipo de error y en la ultima columna la comparación de los valores esperados en el caso ideal y los reales NR-NI (Número de Errores en

Identificador	Perfil	Sección	No. de errores	Promedio	Máximo	Mínimo
1.1	Investigadores	Cursos	69	1.9436	5	0
1.2		Alumnos	35	0.9859	2	0
1.3		Investigadores	19	0.5352	2	0
2.1	Alumnos	Cursos	102	2.8732	6	0
2.2		Alumno	58	1.6571	2	0
3	Público en General	Cursos	6	0.2181	2	0

Tabla 4.4: Resumen de los resultados obtenidos para el caso real

el caso Ideal-Número de Errores en el caso Real). Observando los resultados de la relación de los errores localizados en el caso real con los identificadores del caso ideal, es posible concluir que el comportamiento de la incidencia de errores en los perfiles en el caso ideal, comparadas con los resultados del caso real, en los comportamientos son parecidos en la mayoría de los casos. Sin embargo los valores obtenidos en cuanto a la ocurrencia de errores no son como lo predice el caso ideal, de hecho todos los resultados son mucho mayores en el caso real.

4.4 Conclusiones

En este trabajo se formuló una metodología para la validación de sistemas de información en Internet. El atributo de calidad que se analizó fue la *fiabilidad*. Es claro que un buen análisis de calidad del producto de software es una de las bases más importantes para la toma decisiones dentro de una organización que desarrolla software de calidad. Por tal razón, la evaluación de un producto de software permitirá predecir su comportamiento en cuestión del atributo de calidad.

En nuestro trabajo hemos aplicado modelación estadística para predecir del comportamiento de los atributos de calidad de un sistema de software. Este proceso de análisis ha resultado satisfactorio porque nos permite predecir el comportamiento de nuestro sistema a nivel del atributo estudiado. También nos permite estimar el comportamiento esperado en las vistas y secciones del sistema. En las dos primeras fases aplicadas de la metodología, observamos que la "Evaluación del sistema en un contexto ideal", nos permite establecer predicciones del comportamiento del sistema real. También nos proporciona un patrón de referencia y las secciones o modulos del sistema que conviene evaluar con mayor cuidado durante la aplicación de la segunda fase de la metodología propuesta "Evaluación del sistema real".

Mediante la evaluación de un caso de estudio, hemos concluido que la metodología desarrollada

es una herramienta eficiente dentro del proceso de desarrollo de software. Esta herramienta permite controlar la calidad producto de software resultante. El uso de esta metodología permitirá también lograr una reducción de costos en el sistema de software y también de su proceso de su desarrollo. Esto es porque cuando se tienen productos de baja calidad, normalmente se inyectan en el proyecto mas recursos económicos y mas tiempo en la búsqueda de un mejor nivel de calidad.

En este capítulo, la metodología propuesta fue implementada en su primer fase de evaluación y predicción de la calidad del producto de software. En el siguiente capitulo terminaremos de ejecutar completa nuestra metodología incluyendo el proceso de mejora PSP y su evaluación, con el fin de mejorar los objetivos de calidad. Se pretende extender este trabajo para que esta metodología pueda ser aplicada a la mayoría de los atributos que pueden lograr los sistemas de información en Internet que operan bajo un contexto cliente - servidor.

Id	Perfil	Sección	Transacción	Tipo de errores	EI-ER
1.1.1	*Investigadores*	cursos	consultas	no se ve la sección a la que pertenece el curso	9-69
1.1.2			altas	No valida la información en el formulario No reconoce automáticamente la sección a la que pertenece la persona que da de alta el curso.	
1.1.3			bajas	No valida que el investigador que dio de alta el curso sea el mismo que lo da de baja	
1.1.4			cambios	No verifica que los cambios sean realizados por el mismo investigador que lo dio de alta	
				No verifica que los cambios sean del tipo de campo esperado en la BD	
1.2.1		alumnos	consultas	No se presentan algunos campos que corresponden a la información de los alumnos	
				No se presenta la lista completa de los alumnos que asesora un investigador	
1.2.2		cursos x alumno	altas	Permite inscribirse a mas de 4 cursos por cuatrimestre	13-35
1.2.3			cambios	Cuando se hacen los cambios se toman cursos que el alumno ya tiene programados	
1.3.1		investigadores	altas	No valida la información en el formulario	8-19
			bajas	No verifica que la baja de información sea realizada por el mismo investigador que lo dio de alta	
1.3.3			cambios	No verifica que los cambios sean del tipo de campo esperado en la B.D.	
				No verifica que los cambios sean realizados por el mismo investigador que lo dio de alta	
2.1.1	*Alumnos*	cursos	consultas	No se ve la lista completa de cursos no se ve la sección a la que pertenece el curso	46-102
2.1.2			altas	Permite inscribirse a mas de 4 cursos por cuatrimestre	
2.1.3			cambios	Cuando se hacen los cambios se toman cursos que el alumno ya tiene programados	
2.2.1		alumnos	consultas	No se presentan algunos campos que corresponde a los alumnos	32-58
				A veces no presenta todos los datos del catalogo	
				Faltan datos para identificar al asesor	
2.2.2			altas	No valida la información en el formulario	
2.2.3			bajas	No realiza bien la baja	
2.2.4			cambios	No verifica que los cambios sean del tipo de campo esperado	
				no procesa adecuadamente la información	
2.1.1	*Público en general*	cursos	consultas	No se ve la lista completa de cursos	1-6

Tabla 4.5: Localización de errores reales para el SIV.

Capítulo 5

Implementación del Proceso de Mejora Continua

5.1 Introducción

En el capítulo anterior se describió la metodología para evaluar la calidad en sistemas de información en Internet, desarrollada en esta tesis y se detallaron las primeras dos fases de esta metodología. En estas dos primeras fases se evaluó la calidad del sistema bajo condiciones ideales y bajo condiciones reales. En este capítulo se describen las fases 3 y 4 de la metodología que consisten en la aplicación del Proceso de Mejora Continua (PSP) al caso de estudio (el SIV), y la Evaluación del producto de software después de la implementación del proceso de mejora. El objetivo de la fase 4 es cuantificar las mejoras obtenidas en la calidad (fiabilidad) de sistema.

Durante el desarrollo de la tesis hemos abordado el tema de los procesos de mejora continua, mediante la implementación del Proceso de Software Personal (PSP). En este proceso se aplican correcciones y revisiones del sistema mediante resúmenes y guiones de PSP. El objetivo principal de PSP es mejorar la calidad de los procesos de desarrollo del software y con ello mejorar la calidad del producto. Es de hacer notar, que en la fase 2 de la metodología, la calidad del sistema real fue evaluada de forma manual. El tiempo en que se realizó dicha evaluación fue de varias semanas, por lo que detectamos la necesidad de automatizar este proceso. Esta necesidad se cubrió con el desarrollo e implementación de una herramienta para evaluar la confiabilidad del sistema. Mediante

esta herramienta, se logró bajar los tiempos de evaluación del sistema a pocos minutos.

5.2 Implementación del Proceso de Mejora

5.2.1 Evaluación del personal

PSP es un proceso definido. Los procesos definidos están compuestos normalmente de guiones, tablas, plantillas y estándares (como se puede apreciar en la figura 5.1). Un guión del proceso es un conjunto de pasos escritos, que los usuarios o agentes del proceso siguen cuando utilizan el proceso (en este caso PSP). El guión inicial del PSP se presenta en la tabla 5.1.

La primer etapa en la implementación de PSP se enfoca a obtener el rendimiento y la manera en la que el personal realiza su trabajo. En este punto es necesario que quién implementa PSP evalue sus funciones, los tiempos que emplea, las pérdidas de tiempo y los resultados logrados. Este aspecto está orientado a comprender cómo se utiliza el tiempo, lo que implica la revisión de los siguientes puntos:

1. Clasificación de las actividades principales.

2. Registro del tiempo de cada una de las actividades principales.

3. Registro de tiempos en forma normalizada.

4. Guardar los registros de tiempos en Bitácoras.

El formato del registro de actividades y tiempos puede ser como el de la tabla 5.1. El campo Fecha es para anotar la fecha en la que inicia nuestra actividad y la siguiente casilla es para registrar el día en que se termina la actividad. Otro factor que hay que tomar en cuenta es el tiempo en que hubo interrupciones en el trabajo (TI).

Fecha	Inicio	Fin	TI	δ T	Actividad	Comentario	C	U
23-Junio-2003	9:00	9:50	10 min	40 min	Investigación de PSP0	En este punto solo había	No	2
				2 horas	Investigación de PSP0	que leer bibliografía y papers	No	1
23-Junio-2003	10:00	10:50	15 min	45 min	Investigación de PSP0	Leer bibliografía y papers	No	1
23-Junio-2003	11:00	12:50	20 min	40 min	Investigación de PSP0	Leer bibliografía y papers	No	1
23-Junio-2003	12:00	12:30	0 min	30 min	Investigación de PSP0	Leer bibliografía y papers	No	1

Tabla 5.1: Ejemplo de la utilización del guión del PSP para apoyar la evaluación personal

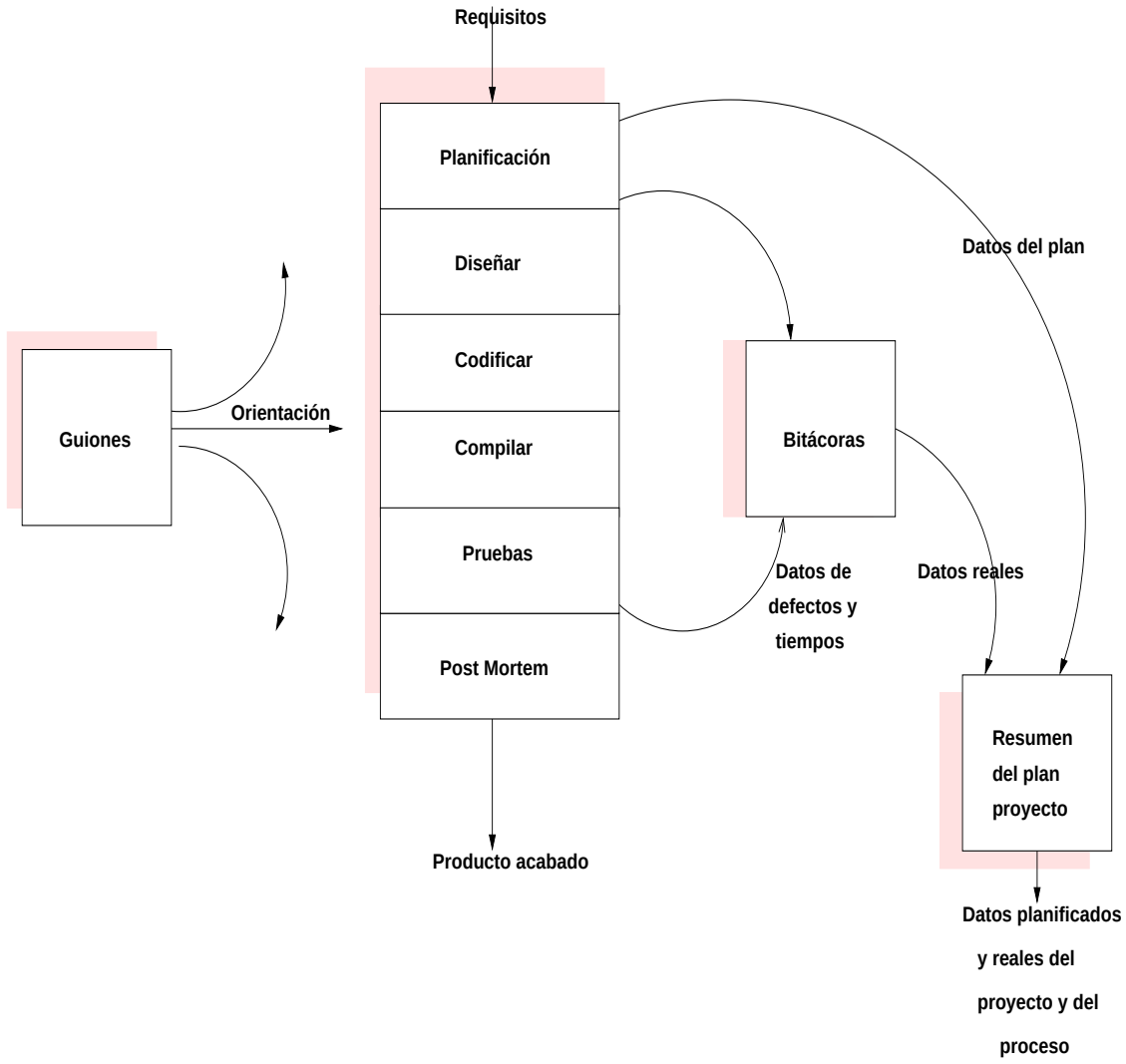

Figura 5.1: Flujo de PSP

La variación del tiempo (δT) en la que se culminan las actividades es importante. Otro punto a considerar consiste en examinar si se trabaja por unidades (U) o por actividades completas (C).

Cuando se aplica este formato y se empieza a llevar un registro de las actividades y los factores que intervienen para su finalización. Quién lo implementa debe percatarse de su eficiencia y de sus vicios. Uno de los factores que se presentan con mas frecuencia son las interrupciones de la labor, con pequeñas actividades como las charlas "cortas", tomar el café, la hora de la comida, hablar por teléfono, etc. Es frecuente emplear tiempos excesivos para actividades de menor importancia. Resulta impresionante a nivel personal que si se estima realizar una actividad en un tiempo determinado, casi siempre el tiempo que se ocupa supera al tiempo planeado. Sin embargo en un

porcentaje alto, las interrupciones hacen acto de presencia con una alta frecuencia y consumen un tiempo largo.

La segunda etapa de PSP es aprender a realizar estimaciones de tiempo para la programación de las actividades de acuerdo al guión 5.2. Realizar la planeación de las actividades y los tiempos del proceso de desarrollo de software, no siempre es sencillo de realizar. Sin embargo el paso anterior es muy importante ya puede ser tomado como un patrón de referencia para la eliminación de las interrupciones más frecuentes.

PSP proporciona un guión para planear el proceso de desarrollo del producto de software. La primera columna de la tabla 5.2 hace referencia a las fases del proceso. En la segunda columna se indican las actividades que intervienen en cada etapa. De acuerdo a la estimación del rendimiento del personal es como se planea el tiempo para cada actividad que se marca en el guión.

La última etapa de PSP se orienta a la obtención y corrección de los defectos. El término *defecto* en este caso indica el incumplimiento de las especificaciones de requerimientos. Estos incumplimientos pueden ser originados por una sentencia incorrecta dentro del código. Es posible asegurar que un producto de software que opera con calidad es aquel que más se acerca a lo especificado por el usuario y por lo tanto opera con el menor número de defectos. Para este este trabajo, todo lo que no funciona de acuerdo a la especificación de los requerimientos del sistema, se considera como *defecto*. En este punto, PSP se basa en la tabla 5.2 para proporcionar un guión aumentado, descrito en la tabla 5.3). En este guión se resalta la localización y corrección de los defectos en cada una de las actividades que intervienen en las fases del proceso de desarrollo del producto de software.

El Proceso de Software Personal aplica el análisis de la eficiencia, la planeación de las actividades y la localización de los errores. Conforme se va aplicando PSP en las actividades del proceso y en general en los proyectos de software, se obtienen mejores resultados, y es entonces cuando el personal aprende de sus errores, mejora y se motiva.

De los resultados obtenidos durante la implementación de PSP en el desarrollo de la metodología, se presenta el resumen mostrado en la tabla 5.4, el cual es el Plan del Proyecto de PSP y el registro de resultados obtenidos en nuestro caso de estudio. De este Plan de Proyecto se pueden observar varios puntos:

- El tiempo por fase (min) que se planea para la codificación de los programas del sistema es menor al real. Esto puede ser posible porque la estimación de las lineas de código por hora que se plantea es de 60. Sin embargo la realidad es que la capacidad del programador es de

47 lineas.

- Una de las actividades que tienen menos tiempo asignado en su estimación de tiempos es la fase de *Diseño*. En base a los resultados reales también su resultado es bajo en comparación de las otras fases.

- Es posible apreciar que la sección de defectos introducidos es la que mas defectos tiene. De acuerdo a este resultado podemos concluir que las fases en las que el programador tiene que poner un cuidado especial, asignando tiempos adecuados son en el *Diseño* y en la *Codificación*. Otro punto que se puede concluir es que si no hay un diseño adecuado, la Codificación también se ve afectada.

El calculo de los datos del plan del proyecto se puede apreciar en la tabla 5.5.

De la aplicación de PSP se localizaron y corrigieron los defectos de cada etapa. Algunos de estos defectos ya se habían detectado en las fases 1 y 2 de la metodología. El resumen de los errores localizados y corregidos, se muestra en las tablas 5.6, 5.8 y 5.7.

Finalmente, el diagrama a bloques del SIV después de aplicar PSP quedó como se muestra en la figura 5.2 y el diagrama entidad-relación de la Base de datos se muestra en la figura 5.3.

Propósito	**Guiar en el desarrollo de pequeños programas**
Criterios de entrada	La descripción del problema tabla Resumen del Plan del Proyecto del PSP Datos de tamaños y tiempos reales de programas anteriores Bitácora de Registro de Tiempos
Planificación	Se obtienen descripciones de las funciones del programa Estimación los LOC Max, Min y total requeridos Determinación de los Minutos/LOC Cálculo de los tiempos de desarrollo Max, Min y Total Llenado de la tabla Resumen del Plan del Proyecto Anotación del tiempo de planificación en la bitácora de Registro de Tiempos
Diseño	Diseño del programa Anotación del diseño en el formato especificado Anotación del tiempo de diseño en el cuaderno Registro de Tiempos
Codificación	Implementación del diseño Utilizar un formato estándar para introducir el código Anotación del tiempo de codificación en la bitácora de Registro de Tiempos
Compilación	Compilación del programa Corrección de los errores encontrados Anotación del tiempo de corrección en la bitácora de Registro de Tiempos
Pruebas	Pruebas del programa Corrección de los errores encontrados Anotación del tiempo de pruebas en el cuaderno Registro de Tiempos
Postmortem	Completación de la tabla de Resumen del Plan del Proyecto con los datos de tiempo y tamaño reales. Anotación del tiempo postmortem en el cuaderno Registro de Tiempos
Criterios de salida	Programa probado a fondo Diseño adecuadamente documentado Listado completo del programa Resumen del plan del proyecto completado Cuaderno de Registro de Tiempos completado

Tabla 5.2: Guión del proceso de PSP para la planeación de actividades

Propósito	**Guiar en el desarrollo de pequeños programas**
Criterios de entrada	La descripción del problema tabla Resumen del Plan del Proyecto del PSP Datos de tamaños y tiempos reales de programas anteriores Cuaderno de Registro de Tiempos **Cuaderno de Registro de Defectos**
Planificación	Se obtienen descripciones de las funciones del programa Estimación los LOC Max, Min y total requeridos Determinación de los Minutos/LOC Cálculo de los tiempos de desarrollo Max, Min y Total Llenado de la tabla Resumen del Plan del Proyecto Anotación del tiempo de planificación en el Cuaderno de Registro de Tiempos
Diseño	Diseño del programa Anotación del diseño en el formato especificado Anotación del tiempo de diseño en el Cuaderno Registro de Tiempos
Codificación	Implementación del Diseño Se utiliza un formato estándar para introducir el código Anotación del tiempo de codificación en el Cuaderno Registro de Tiempos
Compilación	Compilación del programa Corrección y **anotación** de los defectos encontrados Anotación del tiempo de corrección en el Cuaderno Registro de Tiempos
Pruebas	Pruebas del programa Corrección y **anotación** de los defectos encontrados Anotación del tiempo de pruebas en el Cuaderno Registro de Tiempos
Postmortem	Terminación de la tabla de Resumen del Plan del Proyecto con los datos de tiempo y tamaño de **defectos** reales. Anotación del tiempo postmortem en el Cuaderno Registro de Tiempos
Criterios de salida	Programa probado a fondo Diseño adecuadamente documentado Listado completo del programa Resumen del plan del proyecto completado Cuaderno de Registro de Tiempos y **defectos** completado

Tabla 5.3: Guión del proceso de PSP3

Resumen	**Plan**	**Real**		**Hasta la fecha**	
Minutos/LOC	2	0.78		2.78	
LOC/Hora	60	47		107	
Defectos KLOC	0.005	0.01		0.010	
Tamaño Programa (LOC):	3300	3076		6379	
Total Nuevo y Cambiado	300	164		464	
Tamaño máximo	3300	3240		6540	
Tamaño mínimo	3000	3076		6076	
Tiempo por fase(min):	**Plan**	**Real**	**Hasta la Fecha**	**% Hasta la Fecha**	
Planificación	480	240	900	8.2	
Diseño	960	480	1440	13.13	
Codificación	1920	1980	3900	35.58	
Revisión del código:	960	460	1420	12.95	
Compilación	480	420	900	23.07	
Pruebas	960	480	1440	13.13	
Postmortem	480	480	960	24.61	
Total	6240	4720	10960	100	
Tiempo máximo	1920	1980			
Tiempo mínimo	480	240			
Defectos introducidos	**Plan**	**Real**	**Hasta la Fecha**	**% Hasta la Fecha**	**Def/Hora**
Planificación	1	5	6	18.18	0.4
Diseño	1	15	16	48.48	0.66
Codificación	0	4	4	12.12	0.0615
Revisión de código	0	0	0	0	
Compilación	0	0	0	0	
Pruebas	0	1	1	3.03	0.041
Total	2	31	33	100	
Defectos eliminados					
Planificación	1	5	6	18.18	0.4
Diseño	1	15	16	48.48	0.66
Codificación	0	4	4	12.12	0.0615
Revisión de código	0	0	0	0	0
Compilación	0	0	0	0	
Pruebas	0	1	1	3.03	0.041
Total	2	31	33	100	

Tabla 5.4: Resumen del plan del proyecto de PSP3

Propósito	Definición
Minutos/LOC	Se redactan los minutos/LOC planificados en este proyecto Minutos/LOC reales=Tiempo total de desarrollo / Tamaño en LOC del sistema
LOC/Hora	LOC/Hora=Minutos/LOC / 60 LOC/Hora real=60/ Minutos/LOC reales
Tiempo por fase	Tiempo total de desarrollo = (LOC total nuevas & Cambiadas) * Minutos/LOC Tiempo Máximo=Tamaño máximo * Minutos/LOC. Tiempo Mínimo = Tamaño mínimo * Minutos/LOC.
Hasta la fecha	Para cada fase, es la suma del tiempo real y el tiempo hasta la fecha de los programas más recientes
% Hasta la fecha	Para cada fase, (100 * Tiempo de la fase hasta la fecha) / tiempo total hasta la fecha de los programas mas recientes
Defectos reales introducidos	Después del desarrollo, se localiza y anota el número real de defectos introducidos en cada fase
Hasta la fecha	Defectos reales + Defectos hasta la fecha
% Hasta la fecha	(100 * Defectos reales de la fase hasta la fecha) / Total de defectos hasta la fecha
Defectos reales eliminados	Después del desarrollo, se localiza y anota el número real de los defectos eliminados en cada fase
Hasta la Fecha	Defectos reales + Defectos hasta la fecha
% Hasta la fecha	(100 * Defectos hasta la fecha para la fase) / Total de defectos hasta la fecha
Defectos/hora	(60 * Defectos de la fase hasta la Fecha)/ Tiempo de la fase hasta la fecha

Tabla 5.5: Instrucciones del resumen del plan del proyecto del PSP

Id	Sección	Módulo	Errores localizados	Acción correctiva
			Requerimientos	
1.1	Acceso	Acceso	Los datos de acceso como login y password deberán ser encriptados	Especificación en el documento de requerimientos Integración de funciones de encriptación en la B.D.
1.2	Vistas	Investigador	En las vistas se debe de considerar al coordinador como otra vista	Especificación en el documento de requerimientos Integración en el diseño de la vista de los investigador la vista del coordinador
1.3	Cursos Alumnos Inv	Cambios Bajas	La información que se modifica y elimina debe estar restringida a los usuarios que dan de alta	Especificación en el documento de requerimientos Integración en el diseño de los módulos de bajas y cambios para las secciones de cursos,alumnos e investigadores
1.4	Todas	Todos	El SIV deberá tener un mecanismo de respaldo de información periódico	Especificación en el documento de requerimientos Diseño de un guión de shell que efectúe un respaldo del sistema en computacion.cs.cinvestav.mx
1.5	Cursos	Altas Bajas Cambios Consultas	La información de los cursos también debe ser clasificada por cuatrimestre	Especificación en el documento de requerimientos Implementaron en el diseño de la B. D. del campo cuatrimestre en todas las tablas de los cursos

Tabla 5.6: Resumen de errores de acuerdo al plan de PSP (requerimientos)

Id	Sección	Módulo	Errores localizados	Acción correctiva
			Codificación	
3.1	Cursos	Altas	No reconoce la sección a la que pertenece la persona que da de alta el curso	Corrección de empleo del campo la sección de la B.D. en el modulo de altas
3.2		Cambios	No verifica que los cambios sean del tipo de campo esperado en la BD	Corrección en el modulo de cambios la manera de verificar el tipo de campo esperado
3.3		Consultas	No se ve la sección a la que pertenece el curso	Corrección en el empleo del campo que corresponde a la sección de la B.D.
3.4	Alumnos	Consultas	No se presentan algunos campos que corresponden a la información de los alumnos	Utilización adecuada de los nombres de los campos que corresponden a B.D. de los campos que corresponden a B.D.

Tabla 5.7: Resumen de errores de acuerdo al plan de PSP (codificación)

Id	Sección	Módulo	Errores localizados	Acción correctiva
			Diseño	
2.1	Cursos	altas	No se validan los campos en los formatos de alta	Verificar en la BD los formatos Integrarlo al modulo de altas
2.2		bajas	No se valida que la información sea eliminada por quién la inserta	Integrar en la B.D. Un campo que identifique a quien dio de alta la información y verificar este campo en el proceso del módulos de bajas
2.3		cambios	No valida que la información sea modificada por quien la inserta	Tomar de la B.D. el campo que identifica la clave del que hizo la alta y con ello verificar la clave que pretende hacer cambio
2.4		consultas	No se ve el campo cuatrimestre	Integrar el campo correspondiente al cuatrimestre
2.5	Inv	altas	No se validan los campos en los formatos de alta	Verificar en la BD los formatos Integrarlo al modulo de altas
2.6		bajas	No se valida que la clave que inserta la información sea la misma clave en el proceso de bajas	Integrar en la B.D. Un campo que que identifique la clave que hizo la alta de datos
2.7		cambios	No se valida que la clave que inserta sea quien está realizando el cambio	Tomar de la B.D. el campo que identifica la clave d y que sea la misma en el modulo de cambios
2.8		consultas	No se presenta la lista completa de los cursos por sección	Verificar que el campo que corresponde a la sección del sistema este adecuadamente implementado
2.9	Alumnos	altas	No se validan los campos en los formatos de alta	Verificar en la BD los formatos Integrarlo al modulo de altas
2.10		bajas	No valida que la clave que realiza la inserción sea la misma de la baja	Verificar en la B.D. la clave de la alta y verificar que sea la misma que está operando
2.11		cambios	No valida que la clave que realiza la inserción sea la misma del cambio	Verificar de la B.D.la clave de la alta y verificar que sea la misma que está operando
2.12		consultas		
2.13	Cur x Alum	altas	Permite la inscripción de los alumnos a mas de 4 cursos por cuatrimestre	Implementar en el modulo de altas la condición de que solo sean 4 cursos por cuatrimestre
2.14		consultas	No se presenta la lista completa de los alumnos que asesora un investigador	Se verifico el diseño en el modulo de consultas Se verifico en la B.D. que el campo asesor este adecuadamente implementado
		bajas		
2.15		cambios	Cuando se hacen los cambios se toman cursos que el alumno ya tiene programados	Implementar en el de cambios la condición de que los cursos que pueden elegirse sean distintos de los que ya están asignados al alumno

Tabla 5.8: Resumen de errores de acuerdo al plan de PSP (diseño)

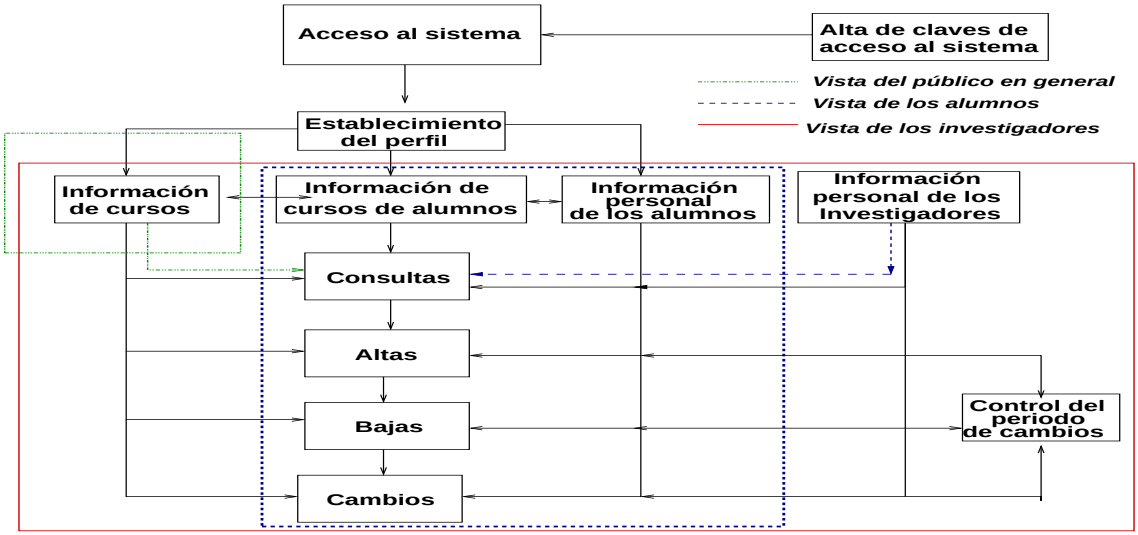

Figura 5.2: Diagrama a bloques del SIV después de aplicar PSP.

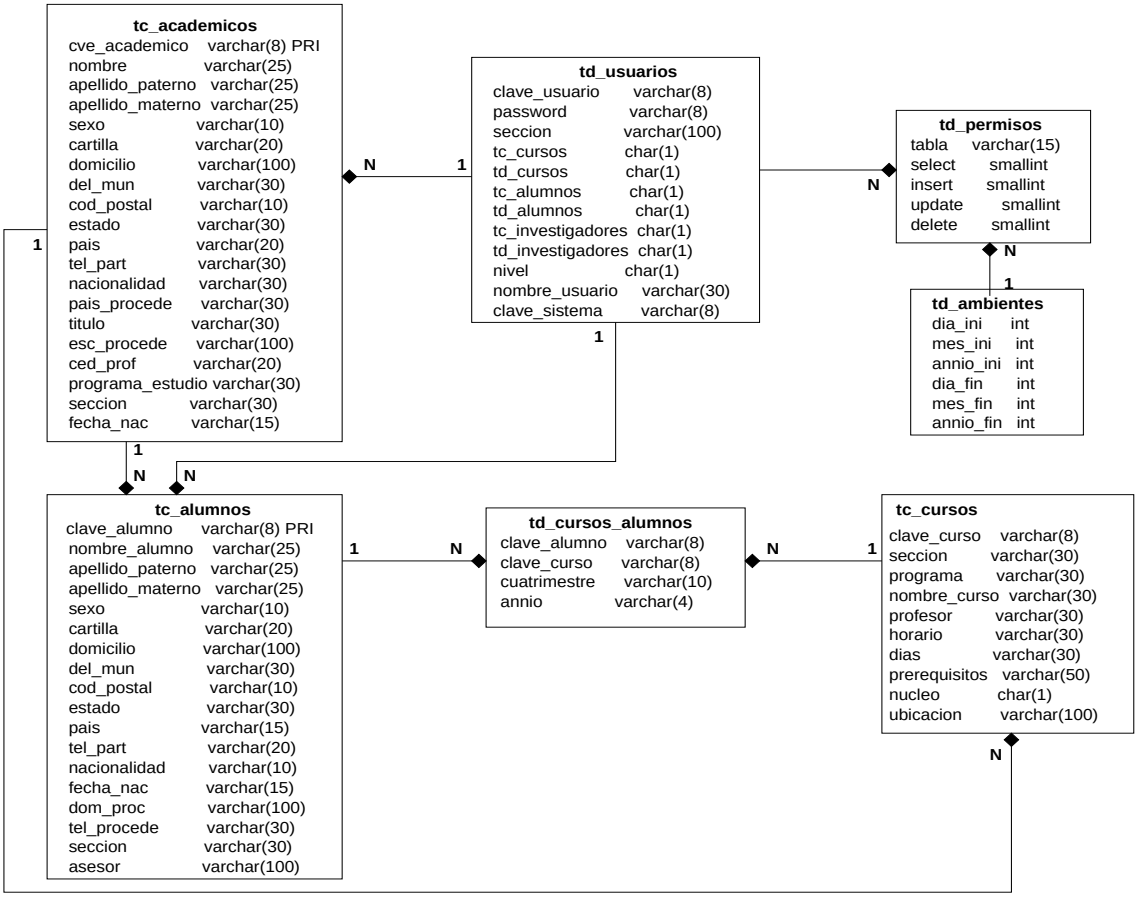

Figura 5.3: Diagram entidad-relación después de aplicar PSP.

5.3 Automatización del Proceso de Evaluación

En las pruebas realizadas en la Fase 2 de la metodología, se contempló que el proceso de evaluación operara considerando arribos que siguieran una distribución exponencial. Otro punto es que los usuarios operaran de manera concurrente, con una matriz de pruebas común. Sin embargo las evaluaciones realizadas durante la aplicación de la fase 2, fueron hechas por un sólo usuario y los arribos fueron programados en periodos de 100 minutos, por lo cual el sistema se evaluó de forma secuencial. Esto implicó que este usuario fuera el único dentro del sistema durante el proceso de evaluación. La secuencialidad elimina muchas fuentes de errores.

Se realizó un estudio acerca de las herramientas que existen para evaluar aplicaciones en Internet [6], [8]. Sin embargo no encontramos ninguna herramienta (que fuera gratuita) que nos permitiera evaluar la funcionalidad de un sistema de información en Internet y que considerara los requerimientos del sistema. Por esta razón, surgió la necesidad de programar una herramienta que funcionara de acuerdo al contexto deseado para las evaluaciones de SIV. Se planteó el problema de realizar una herramienta para evaluar el sistema SIV con las siguientes características:

- La evaluación de los clientes es de manera concurrente.
- Cada cliente actúa de acuerdo a su vistas: Investigadores, Coordinador, Alumnos y Público en General.
- Los módulos a evaluar en el sistema son: Cursos (cursos x alumno), Alumnos, Investigadores.
- Las transacciones que cada cliente realizaría (de acuerdo a su perfil) en los modulos del sistema son: consultas, altas, bajas y cambios.
- El arribo debía ser de tipo exponencial.
- Los clientes tienen una matriz de pruebas en común.
- Después de la evaluación se realiza un análisis de los errores obtenidos durante las evaluaciones de los clientes.
- El tiempo de la evaluación general debía durar 100 unidades de tiempo.

El contexto de la herramienta se muestra en la figura 5.4.

Durante las fases del desarrollo del sistema se contemplaron los siguientes puntos:

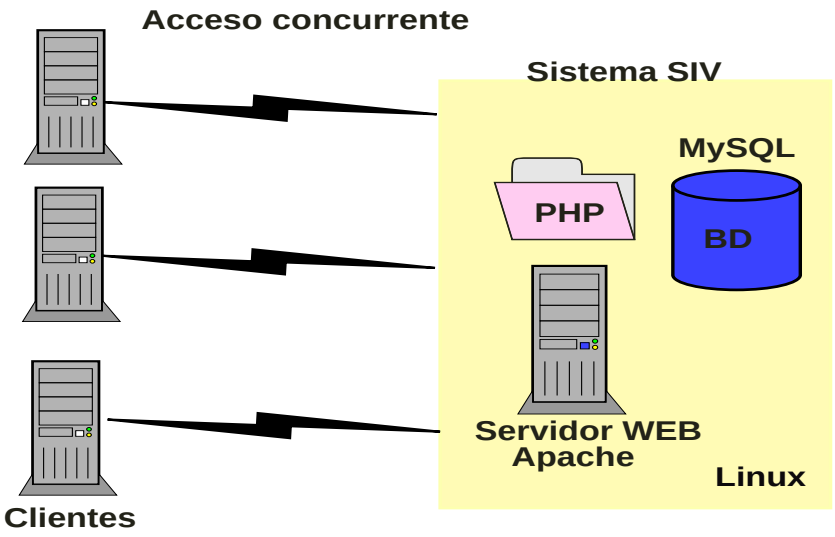

Figura 5.4: Contexto de evaluaciones para el SIV

1. Diseño. De acuerdo a los requerimientos para la herramienta, se determinó que ésta debería de ser un autómata que operara en base de las transacciones que pueden realizar las vistas en los modulos del sistema. Así la matriz de estados, en sus columnas tiene la relación de los modulos del sistema y en sus filas tiene la relación de las vistas que pueden operar en el sistema. El contenido de esta matriz determina las transacciones que puede ejecutar la vista en el modulo a examinar.

2. Codificación. La clases que la integran a la herramienta son las siguientes:

- La clase **Test_SIV**. Tiene la función de despertar a los threads que controlan a los usuarios virtuales mediante los métodos:
- *Init()*. Tiene la función de inicializar los contadores estadísticos, las estructuras de datos y programa el arribo y el perfil del primer usuario.
- *Arrive()* se encarga de programar los siguientes arribos en base de una distribución exponencial y de llamar a *Acceso*.
- *Acceso(int skill, int test_time)* verifica si el usuario es válido dentro del sistema. En caso de ser válido genera una instancia de *Proceso_user*.
- *Proceso_user* es un autómata que tiene determinadas las acciones que cada uno de los perfiles de usuario puede realizar en cada una de las secciones mediante *action_upon_state*, *set_action_section* y *transactions*

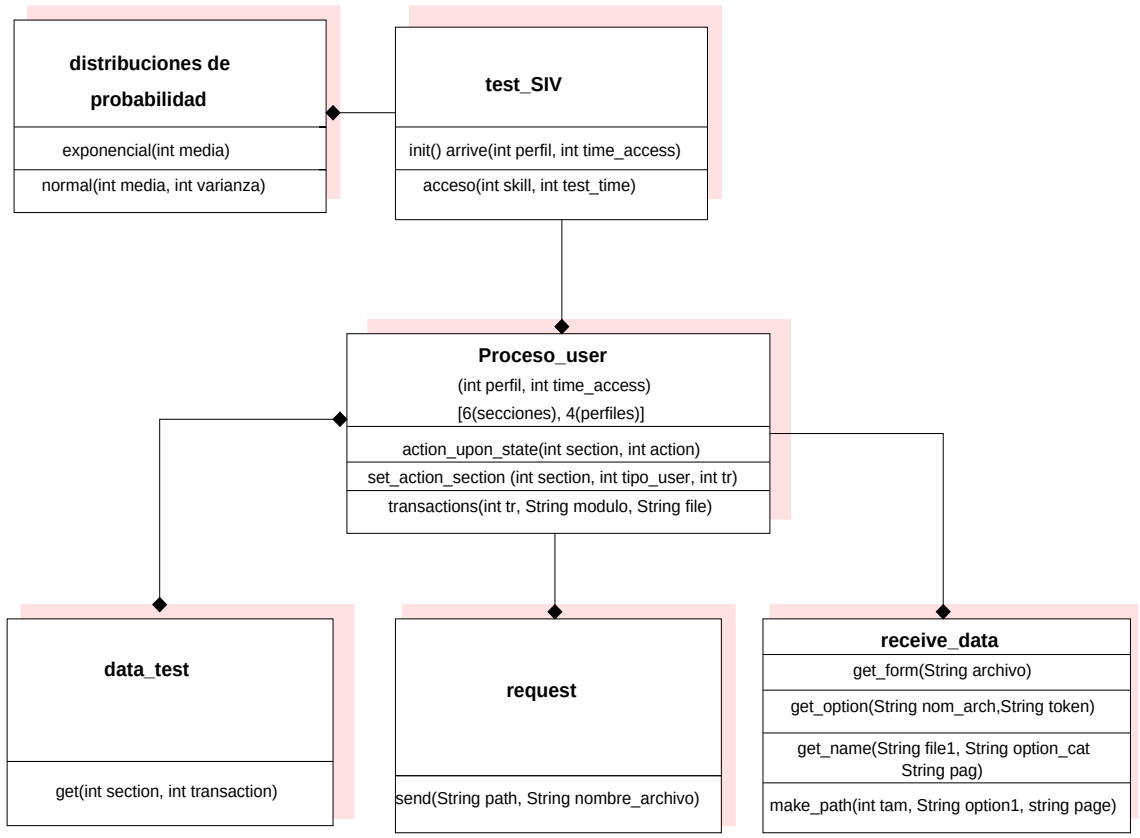

Figura 5.5: Diagrama de clases de la herramienta para realizar evaluaciones al SIV

- *Request* opera mediante la clase URL. Tiene la función de establecer las peticiones de los clientes hacia el servidor. En el tipo de petición se establece el protocolo (http), la dirección (xxx.xxx.xxx.xxx) y el *path*, separado con signo de interrogación (?), de la pagina que se desea consultar y con el signo (&) de las variables y sus valores.

- *Receive* revisa las respuestas del servidor y responde de acuerdo al análisis de la respuesta dependiendo del caso.

El proceso de análisis se realizó mediante *guiones en shell* (o programación del shell). En general tiene la función de revisar el conjunto de archivos de respuesta de las transacciones que cada uno de los clientes realizó durante la evaluación en el SIV. Con este análisis se obtiene el número de errores y con ello la *Densidad de Defectos*. La operación del script se muestra en la figura 5.6.

3. Implementación El sistema está operando en una plataforma Linux. La herramienta fue

desarrollada con el Lenguaje Java.

5.4 Fase 4: Evaluación de la *fiabilidad* en el Sistema Real después de PSP

1. Determinación de las condiciones iniciales para el caso real.

Las condiciones del proceso fueron las siguientes. Las evaluaciones fueron hechas por la herramienta que se describe en la sección 5.3, la cual operó tomando el rol de tres tipos de vistas: alumnos, investigadores (incluye la del coordinador) y público en general. La distribución del los perfiles se hizo de acuerdo al análisis en donde el 70 % de los usuarios fueron alumnos, el 20 % Investigadores (incluyendo al Coordinador) y el 10 % restante fueron el público en general. Las transacciones sobre el SIV fue realizada de manera concurrente. Todos los clientes trabajaron con una matriz de pruebas común. La matriz de pruebas contenía los distintos servicios y los datos requeridos a accesar del sistema. Esta matriz estuvo compuesta de opciones válidas y opciones no-válidas. Los datos de prueba fueron tomados de manera aleatoria de la matriz de pruebas, para cada evaluación.

Las condiciones de operación en el sistema fueron las siguientes:

- El arribo de los clientes se dio en base de una distribución exponencial.
- La evaluación de los usuarios fue hecha de manera concurrente.
- La evaluación se realizó sobre una plataforma Linux.
- El servidor Web operó con un solo procesador de 500 Mhz.
- La capacidad del servidor Web y la del manejador de la Base de Datos con la cual pueden operar adecuadamente el sistema (de acuerdo a los estándares del proveedor *redhat*) es de hasta 100 usuarios.

Los resultados obtenidos en cada evaluación se analizaron mediante un sistema realizado con guiones en shell, dedicado a examinar cada una de las respuestas en el proceso de evaluación. El análisis de ese programa se guardó en bitácoras.

2. Selección de los Atributos a Medir y sus Métricas.

Tipos de métricas. Al igual que en las fases anteriores, el atributo a evaluar fue la *fiabilidad* y la métrica para dicho atributo fue la *densidad de defectos*.

Condiciones de desarrollo. En el SIV se aplicó PSP para la corrección de sus defectos. En el desarrollo de este proceso de mejora sólo intervino una persona en el desarrollo del sistema en Internet y el tiempo de implementación fue de 6 semanas.

Tiempo de evaluación. El tiempo para la evaluación de cada experimento fue de 100 unidades de tiempo. El número de experimentos fue de 70. Los experimentos fueron llevados a cabo en 3 horas. Como se puede observar, la herramienta de evaluación permitió reducir considerablemente el tiempo asignado a las evaluaciones, con respecto al tiempo que se empleo en la fase 2. Recordemos que el tiempo de las evaluaciones de la fase 2 (que se hicieron de forma manual) fue de tres semanas.

3. Proceso de Medición.

a. **Selección de los componentes a medir.**

   Durante la operación del sistema real se midieron los componentes que corresponden a las secciones de Cursos, Alumnos, Investigadores y a la sección de Cursos por alumnos que integran el sistema.

b. **Medir las características de los componentes con las métricas de software.**

   Al igual que en las fases anteriores de la metodología, la métrica utilizada fue la *densidad de defectos*. Durante cada experimento se contabilizó el número de defectos ocurridos.

c. **Identificar las mediciones anómalas.** Durante la ejecución del sistema no detectamos mediciones anómalas.

4. Evaluación de los resultados y selección del modelo.

El resumen de los datos obtenidos durante las 70 evaluaciones se presenta en la tabla 5.9. Al igual que en las fases anteriores de la metodología, en la tabla se registran los siguientes valores: Densidad de defectos (D.Defec), frecuencia de ocurrencia (Frec), la representación en x (x), la densidad de probabilidad (Den. Prob.) y la distribución acumulativa (Dist. Acum.). El histograma resultante se presenta en la figura 5.7. De los datos del histograma derivamos la gráfica del modelo real después de aplicar PSP. La gráfica del modelo real después de aplicar PSP se muestra en la figura 5.8 Como se puede apreciar, esta gráfica representa una curva de Weibull.

D. Defec	Frec	x	Den. Prob.	Dist. Acum.
0	24	0.00	0.300	0.0300
1	10	1.00	0.125	0.425
2	20	2.00	0.0250	0.450
3	8	3.00	0.100	0.550
4	8	4.00	0.100	0.650
5	2	5.00	0.250	0.675
6	14	6.00	0.175	0.850
7	12	8.00	0.150	1.00

Tabla 5.9: Resumen de los resultados del SIV después de aplicar PSP

5. Estimación de los parámetros del modelo.

La función de densidad utilizada para modelar el caso real es la siguiente.

$$f(x) = \alpha\beta^{-\alpha}x^{\alpha-1}e^{-(x/\beta)^{\alpha}} \tag{5.1}$$

Evaluar los parámetros contenidos en la ley escogida. Del análisis de la ecuación 5.1, se obtuvieron los estimadores de máxima verosimilitud (maximum-likelihood estimators MLEs) [3], que derivaron en las siguientes ecuaciones.

$$\frac{\sum_{i=1}^{n} X^{\alpha} lnX_i}{\sum_{i=1}^{n} X^{\alpha}} - \frac{1}{\alpha} = \frac{\sum_{i=1}^{n} lnX_i}{n} \tag{5.2}$$

$$\beta = \left(\frac{\sum_{i=1}^{n} X_i^{\alpha}}{n}\right)^{1/\alpha} \tag{5.3}$$

La ecuación 5.2 se resuelve mediante el método de *Newton-Rapson*, mientras que la ecuación 5.3 se obtiene de manera directa conociendo el valor de α. Los parámetros α y β obtenidos son los siguientes.

$$\alpha_r psp = 1.18 \text{ y } \beta_r psp = 3.8$$

6. Sustitución de los parámetros obtenidos f_r y graficación del modelo real después de aplicar PSP.

Sustituyendo los valores de α y β en la función del modelo real f_r nos dio la siguiente expresión.

$$f_r(x) = \begin{cases} 0.244x^{0.18}e^{-(x/3.8)^{1.18}} & \text{Si } x \geq 0 \\ 0 & \text{En otro caso} \end{cases}$$

La gráfica del comportamiento de la *fiabilidad* de acuerdo al modelo real con las mejoras se presenta en la figura 5.8.

7. Comparación del modelo obtenido.

Es posible observar que el modelo obtenido para el caso real sigue una distribución de Weibull, la cual permite representar el comportamiento de la *fiabilidad* real.

8. Realización de las predicciones de la fiabilidad.

En la figura 5.8 se observa la comparación de los modelos resultantes (ideal y real) de las evaluaciones del caso de estudio. Los parámetros α y β representan el grado de aproximación que existe entre los modelos ideal y real. Para obtener un producto de software con un buen nivel de fiabilidad se debe cumplir lo siguiente.

$$\alpha_r \rightarrow \alpha_i \text{ y } \beta_r \rightarrow \beta_i$$

Es posible observar en la figura 5.8 que el comportamiento obtenido del modelo real se acerca al comportamiento del modelo ideal. Su distribución alcanza sus máximos en el rango de los 0-2 errores al igual que en el modelo ideal. Sin embargo el rango de errores que se presenta para el modelo real después de aplicar PSP es menor que para el caso ideal y también menor que el del modelo real antes de aplicar PSP. El objetivo principal del proceso de mejora es que el sistema opere con un valores de *densidad de defectos* que tiendan a cero errores. Otro punto es que el rango de errores sea el menor posible ó al menos con el mismo rango que presenta la función ideal. El comportamiento del modelo real después de aplicar PSP es mas parecido al ideal. Esto se puede derivar de la observación de que:

$$\alpha_r psp \rightarrow \alpha_i \text{ y } \beta_r psp \rightarrow \beta_i \Rightarrow f_r psp(x) \rightarrow f_i(x).$$

Por lo tanto, concluimos que las mejoras en el producto de software se reflejaron en el comportamiento del producto de software, después de la implementación del proceso de mejora PSP.

5.5 Conclusiones

En este capitulo hay varios puntos importantes a resaltar. Durante la implementación de PSP fue posible observar que ayuda a presentar y con ello a analizar la forma en la que labora el personal encargado del proceso de desarrollo. Los métodos empleados a nuestro juicio son sencillos de aplicar, sin embargo presentaron información irrelevante en nuestro caso, como el tiempo que se emplea en cada fase del proceso de desarrollo. Durante la implementación de PSP se presentó la evaluación de las características de eficiencia con respecto a los tiempos y a los errores detectados. El factor de PSP, que a nuestro juicio mejora de manera directa la calidad del producto, es la detección y la corrección de los errores y los análisis que conlleva. La mejora de la calidad se puede apreciar mediante la evaluación que se realiza posteriormente. Cabe destacar que el objetivo de nuestro trabajo fue de que el sistema operara con los menores errores posibles, o dicho de otra forma, que se lograra una densidad de defectos que tendiera a cero. Otro punto es que el rango de errores sea el mínimo.

La herramienta desarrollada permitió reducir los tiempos de evaluación del SIV. Este desarrollo permitió observar la necesidad de contar a futuro con una herramienta de evaluación más general, capaz de evaluar cualquier sistema de información en Internet que opere en un contexto similar. La experiencia adquirida durante la implementación de la metodología nos permite concluir que la metodología es una herramienta capaz de mejorar la calidad de un producto de software basado en la Internet.

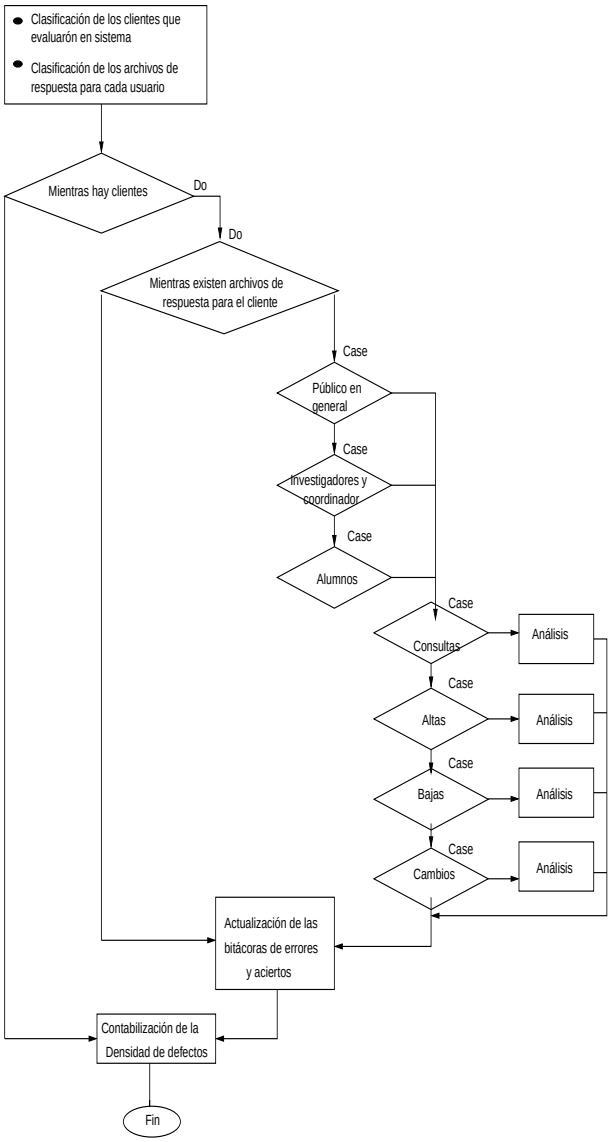

Figura 5.6: Diagrama del funcionamiento del analizador de resultados del proceso de evaluación

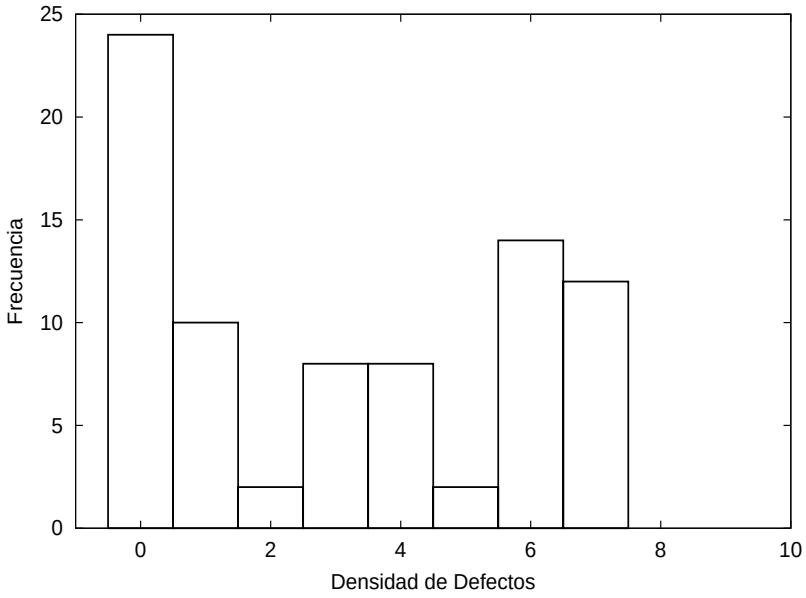

Figura 5.7: Histograma para 70 mediciones de errores

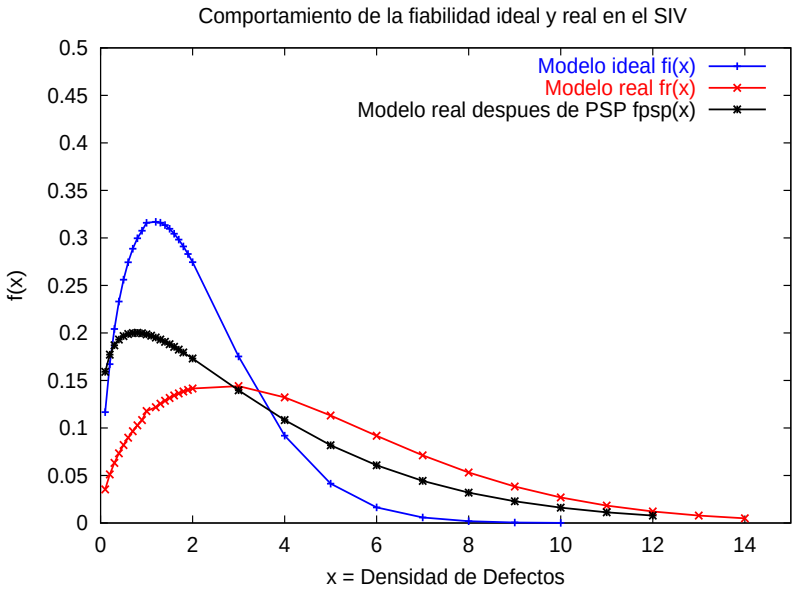

Figura 5.8: Gráfica del modelo ideal, real antes y después de implementar PSP

Capítulo 6

Conclusiones y Trabajo Futuro

6.1 Conclusiones

En este trabajo se formuló una metodología para la validación de sistemas de información en Internet. El atributo de calidad que se analizó fué la *fiabilidad*.

La implementación de técnicas de simulación en un contexto ideal, es fundamental para establecer modelos que sirvan como entradas fiables durante la implementación de la metodología. Esto es, debido a que el modelo ideal marca la dirección en la que se concentraran los esfuerzos para mejorar el comportamiento del atributo de calidad en cuestión.

El proceso de evaluación tiene como fundamento el establecimiento de las metricas de software, lo cual permite establecer la especificación del atributo de calidad de software desde un inicio. Como consecuencia el estudio es sistemático y consistente. Las actividades que incluye el proceso, ayudan a lograr un mejor control de los resultados y conservar el enfoque del objetivo de la evaluación.

La utilización de técnicas de modelación estadística demostró ser una herramienta eficáz para establecer las predicciones del comportamiento del atributos de calidad del producto de software. Este punto es particularmente visible cuando son superpuestos los modelos en la gráfica donde se observan el modelo ideal y las tendencias del modelo real antes y después de implementar PSP 5.8.

El proceso de mejora PSP fué aplicado a lo largo de todo el proceso de desarrollo que se implemento en un proceso de reingenieria del producto de software. El punto que consideramos fundamental en la mejora de la calidad del producto es el proceso de validación de la eficiencia

del personal. La manera en la que se gestiona y desarrolla el software influye en la calidad de su operación.

Mediante la evaluación del caso de estudio, hemos concluido que la metodología desarrollada es una herramienta eficiente dentro de los procesos de mejora continua. Esta herramienta permite cuantificar, mejorar y controlar la calidad producto de software resultante.

6.2 Trabajo Futuro

Se pretende extender el trabajo para que ésta metodología pueda ser aplicada a la mayoría de los atributos que pueden lograr los sistemas de Software en Internet. En este punto otro atributo que se considera fundamental para que un sistema opere adecuadamente en Internet es la *Seguridad*. Que es el primer candidato para aplicar la metodología.

De entre los aspectos de importancia que fuerón surgiendo durante el desarrollo del trabajo, también se observó la importancia que tienen las pruebas dentro de la evaluación de la calidad. Para este efecto existen muy pocas herramientas. En este momento la mayoría de las organizaciones que realizan pruebas al software que desarrollan, las llevan a cabo mediante procesos manuales ad-hoc y con grupos de trabajo o personas dedicada para este fín. Este proceso absorbe grandes cantidades de tiempo. Debido a esto, podemos vislumbrar como trabajo futuro el desarrollo de una herramienta genérica de pruebas para sistemas de información en Internet, capaz de sustituir a las pruebas manuales y cuyo objetivo sea el de reducir los tiempos de evaluación.

Apéndice

En este apéndice incluiremos la descripción del Sistema de Inscripciones Virtual (SIV) que sirvió de caso de estudio en la tesis.

A.1. Proyecto: Sistema de Inscripciones Virtual (SIV)

1. **El problema.**

   Se especificará, diseñará e implementará un sistema de inscripciones virtual que permita al Departamento de Ingeniería Eléctrica (*DIE*) realizar inscripciones a cursos de forma remota. El SIV permitirá agilizar la Inscripción a los cursos que imparte el DIE a través de la Internet. Asimismo, El SIV permitirá al alumno elegir los cursos que desea tomar dentro de la sección a la que pertenezca dentro del DIE. Se desea particularmente que se construya una interfaz gráfica amigable y eficiente.

2. **Requerimientos básicos**

   Un SIV con requerimientos mínimos debe proveer la siguiente funcionalidad.

   - El SIV permitirá a un alumno elegir los cursos que desea tomar en un cuatrimestre, en la Sección del DIE a la que pertenezca.
   - Solo podrán hacer uso del SIV aquellos alumnos que se encuentren registrados en el CINVESTAV-IPN y que estén como alumnos regulares (que no estén dados de baja) en alguna Sección del DIE.
   - Todos los accesos al SIV deberán hacerse desde una Interfaz gráfica accesible desde Internet.

- El alumno podrá consultar los contenidos de cada curso y los datos que le permitan elegir su bloque de materias.
- Así mismo, el sistema permitirá la creación y modificación de bases de datos conteniendo información de los cursos y de los alumnos inscritos en cada Sección del DIE del CINVESTAV-IPN.
- El acceso al SIV solo podrá hacerse mediante una clave que será inicializada por los coordinadores académicos de cada sección a cada alumno.
- Los coordinadores académicos de cada sección también contarán con su respectiva clave de acceso y podrán accesar la base de datos de alumnos y de materias con el fin de consultar, añadir o modificar estas bases de datos.
- Se pueden considerar las siguientes vistas al sistema: Público en general (los cuales solo pueden consultar datos de los cursos), Alumnos del CINVESTAV-IPN (los cuales pueden consultar e inscribirse a los distintos cursos a que les permite el reglamento, y Coordinadores Académicos quienes tienen todos los permisos para crear las bases de datos de alumnos y cursos y agregar/modificar su contenido.

3. **Bases de datos.**

   Las Bases de Datos podrán ser creadas y/o modificadas mediante un manejador de bases de datos convencional (por ejemplo Access).

   La información a incluir en las Bases de Datos es la siguiente:

   - **Alumnos:** Fecha actual, Fecha de Inscripción a la Sección, Datos Biográficos, Universidades o Colegios en donde estuvo inscrito antes el alumno, resultado del examen de admisión a la Sección, Beca del alumno, Asesor asignado.
   - **Cursos:** Nombre del curso, Profesor que la imparte, cuatrimestre en que se imparte, Contenido del curso, Cursos de pre-requisito, numero de alumnos registrados a este curso.

4. **Interfaz de usuario**

   Al inicio del sistema, se leerá la información en estas dos bases de datos. La interfaz de usuario será capaz de:

- Presentar un menú basado en ventanas y botones que permita desplegar los alumnos con sus datos respectivos y desplegar los cursos y la información que corresponde a cada curso.
- Permitirá a distinto tipo de usuarios (Coordinadores - alumnos - publico) introducir o leer información del sistema, presentado distintas vistas.
- Permitirá al coordinador crear las bases de datos, ver que alumnos están inscritos en cada curso, ver en que cursos se inscribió, y modificar e imprimir el contenido de las bases de datos.
- Permitirá al coordinador académico de alguna sección asignar una clave de acceso al sistema a cada alumno.
- Permitir a los alumnos seleccionar una Sección y un cuatrimestre del DIE en la cual desea inscribirse a llevar un conjunto de materias.

5. **Seguridad**

   El alumno podrá inscribirse a los cursos si cuenta con un password asignado por el coordinador académico. Solo podrá inscribirse a un numero máximo de cursos por cuatrimestre (de acuerdo a lo establecido por el reglamento). Además, solo podrá inscribirse dentro de las fechas "establecidas"previas al inicio del cuatrimestre correspondiente. El alumno podrá salvar el estado de SIV en cualquier momento. Si el alumno no termina de inscribirse a todas las materias que debe cursar un cuatrimestre, podrá hacerlo en una sesián futura, siempre y cuando sea antes de las fechas "establecidas"previas al inicio del cuatrimestre correspondiente. Además, si el alumno no salva el estado de su sesián al salirse, se le presentara un *Aviso*, advirtiéndole que no ha salvado y permitiéndole que salve o descarte los cambios hechos.

6. **Sugerencias**

   Se deberá consultar el reglamento del CINVESTAV-IPN, en lo que corresponde a inscripciones. Así mismo se debe consultar en todas las secciones del DIE sobre los cursos que se ofrecen en el próximo año escolar.

A.2. Descripción de los requerimientos

La descripción de los requerimientos se presenta en la tabla 6.2.

RF	Descripción
R.F.1	Los accesos a SIV deberán hacerse desde una Interfáz gráfica accesible desde Internet.
R.F.2	Las vistas a considerar son las siguientes: "Público en general, alumnos, coordinadores, e investigadores. "
R.F.3	El público en general sólo puede consultar los distintos cursos que les permite el reglamento.
R.F.4	Los alumnos pueden consultar e inscribirse a los distintos cursos que les permite el reglamento.
R.F.5	Los coordinadores investigadores tienen todos los permisos para administrar la información de su sección.
R.F.6	Los investigadores pueden consultar su información y la de de los alumnos que se inscriben a sus cursos
R.F.7	El acceso al SIV sólo podrá hacerse mediante una identificación de usuario ($Id_usuario$) y una contraseña para tener acceso a los servicios del alumno o del coordinador.
R.F.8	Cada usuario contará con un identificador ($Id_usuario$) y una contraseña que será cargada por los coordinadores investigadores de cada sección.
R.F.9	El coordinador tendrá la capacidad de modificar las contraseñas de los alumnos.
R.F.10	Permitir la inscripción de los alumnos a los cursos del periodo actual
R.F.11	Sólo podrán inscribirse mediante el SIV aquellos alumnos que se encuentren registrados en el SIV.
R.F.12	Sólo podrán inscribirse los alumnos que sean regulares según el reglamento.
R.F.13	Sólo podrán inscribirse los alumnos en el periodo establecido por la sección mediante el SIV.
R.F.14	Sólo podrán inscribirse como máximo a 4 cursos, a menos que el coordinador académico active otra opción. "
R.F.15	El sistema permitirá al alumno visualizar los cursos a los que se encuentra inscrito.
R.F.16	Los coordinadores deben tener la posibilidad de establecer el periodo de inscripciones
R.F.17	El sistema les permitirá a los coordinadores dar de baja a un alumno de un curso específico.
R.F.18	El sistema permitirá a los coordinadores visualizar los cursos que ha tomado un alumno en su estancia en el DIE.
R.F.19	Los coordinadores podrán visualizar los cursos a los que se encuentra actualmente inscrito un alumno.
R.F.20	El sistema permitirá a los coordinadores la administración de la información de los alumnos, cursos e investigadores de su sección."
R.F.21	El sistema debe permitir agregar información para alumnos
R.F.22	El sistema debe permitir agregar información para cursos
R.F.23	El sistema debe permitir agregar información para investigadores
R.F.24	El sistema debe permitir modificar información para alumnos
R.F.25	El sistema debe permitir modificar información para cursos
R.F.26	El sistema debe permitir modificar información para investigadores
R.F.27	El sistema debe permitir eliminar información para alumnos
R.F.28	El sistema debe permitir eliminar información para cursos
R.F.29	El sistema debe permitir eliminar información para investigadores
R.F.30	El sistema debe permitir cambiar la contraseña de los usuarios
R.F.31	El sistema debe notificar vía e-mail los cambios en la información al coordinador y a la persona que la modifica.

Tabla 6.1: Concentrado de requerimientos funcionales

A.3. Descripción del SIV

El SIV está compuesto por los módulos descritos en la figura 6.1. El funcionamiento del sistema se presenta mediante el diagrama de flujo de la figura 6.2. Así mismo, en la figura 6.4 se describen los servicios y funciones del SIV. El funcionamiento de la base de datos se representa mediante su diagrama de entidad relación mostrada en la figura 6.3.

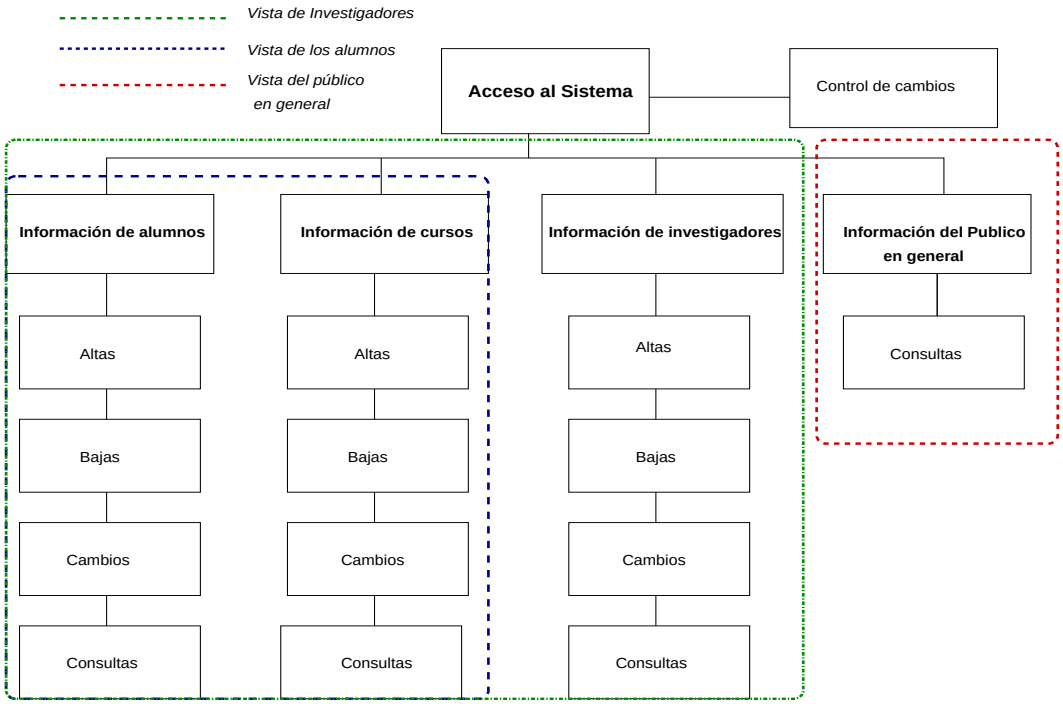

Figura 6.1: Diagrama a bloques del SIV

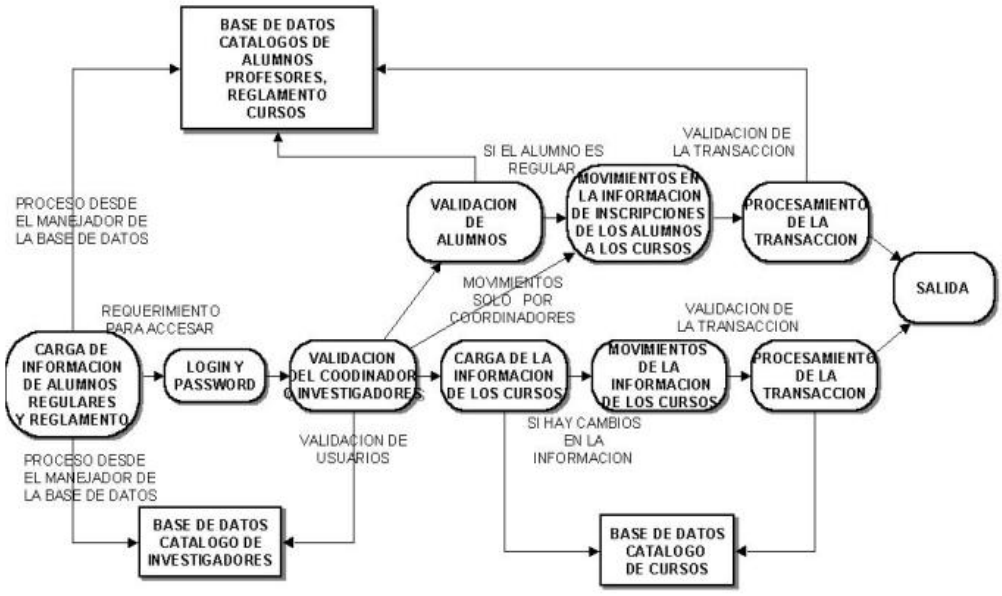

Figura 6.2: Diagrama de flujo de datos

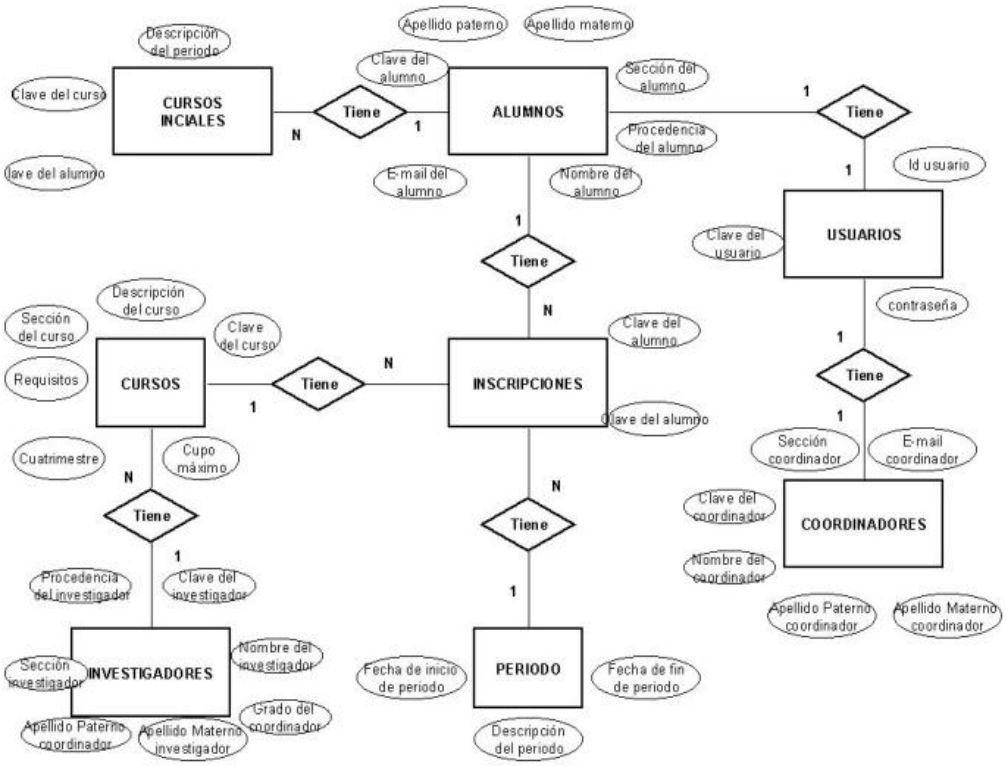

Figura 6.3: Diagrama entidad relación

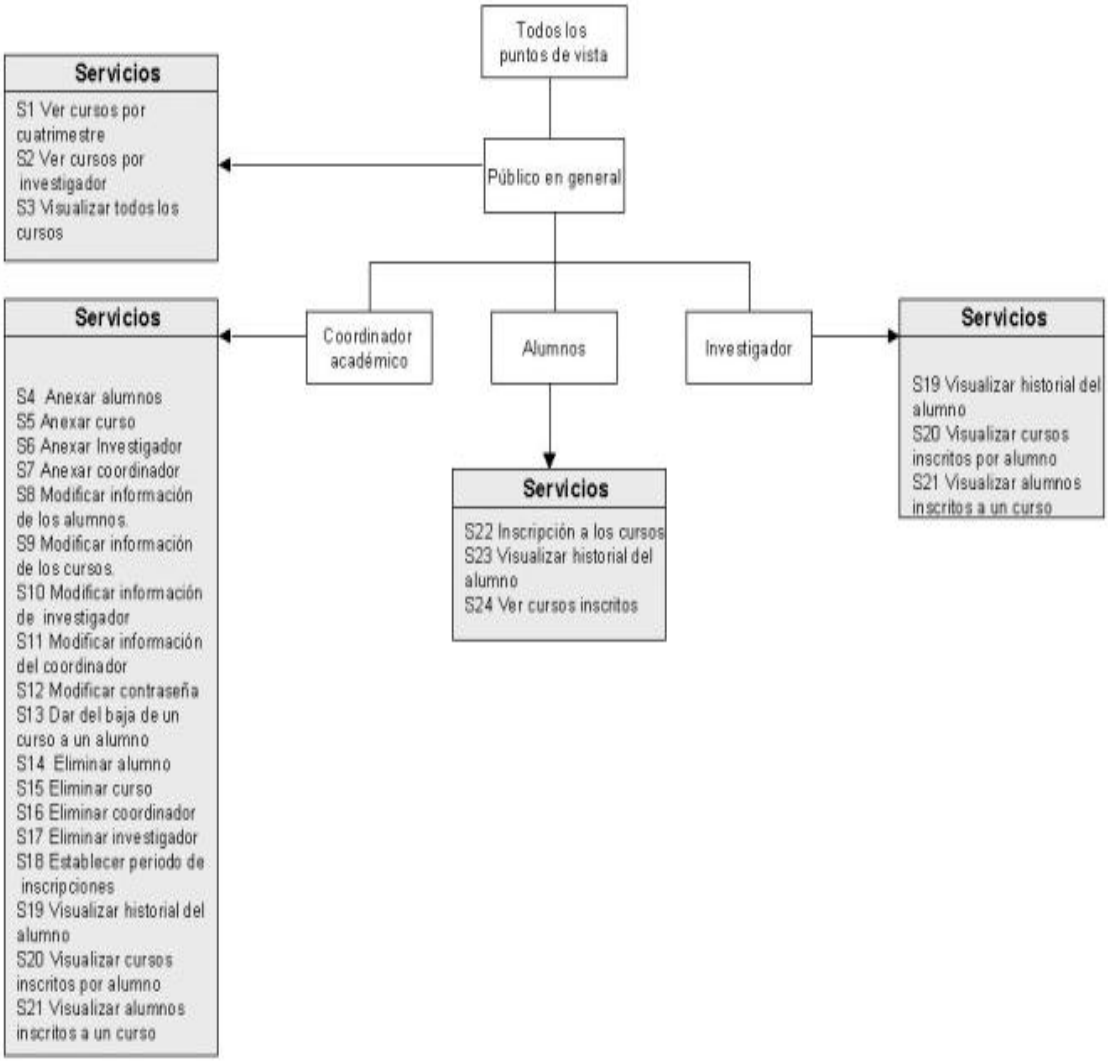

Figura 6.4: Servicios y funciones del SIV.

Bibliografía

[1] A.C. Gillies. Software Quality, Theory and Management. Thomson Computer Press. 1999.

[2] Alberto Moreno Bonett, Francisco Javier Jauffred M. 1997 Elementos de probabilidad y estadística. ISBN: 9701502574. McGraw - Hill , 1997.

[3] A. M. Law, W. David Kelton. Simulation Modeling and Analysis. Third Edition McGraw - Hill , 2000.

[4] Becker and F.E. Mottay. A Global Perspective on Web Site Usability. *IEEE Software, 0740-7459/00*, January/February 2001.

[5] B. Tombuyses, 1999. Reduction of the Markovian system by the influence graph method - error bound and reliability computation. *Reliability Engineering and System Safety*, vol. 63, No. 1, pp 1-11. 1999.

[6] Chaitanya Kallepalli, 2001. Jeff Tian, 2001 Usage Measurement for Statistical Web Testing and Reliability Analysis. Proceedings of the Seventh International Software Metrics Symposium (METRICS 01). *Computer IEEE 1530-1435/01.*

[7] D. J. Paulish, Siemens Corporation Research and Anita D. Carleton, Software Engineering Institute. Case Studies of Software Process-Improvement Measurement. *IEEE Computer 0018-9162/94*, pp.50 1994.

[8] Edward Hieatt and Robert Mee. Going Faster:Testing the Web Application. *IEEE Software,0740-7459/02,*March/April 2002, pp.60.

[9] E. Clarke, O. Grumberg, and D. Long, Software Engineering Institute. Model checking and abstraction. *ACM Transactions on Programing Languajes and Systems*, vol. 16, No. 5, pp. 1512-1542, January 1994.

[10] H. Van Vliet. Software Engineering, Principles and Practice , Second Edition. John Wiley and Sons, 2001.

[11] I. Somerville. Software Engineering, Sixth Edition. Pearson Education Limited, Harlow England, 2001.

[12] Jagadish Kamatar, Will Hayes. An Experience Report on the Personal Software Process. *IEEE Software, 0740-7459/00*, November/December 2000.

[13] J. Offutt. Quality Attributes of Web Software Applications. *IEEE Software, 0740-7459/02.* March/April 2002.

[14] Maurizio Morisio. Applying the PSP in Industry. *IEEE Software, 0740-7459/00*, November/December 2000.

[15] S. H. Kan. Metrics and Models in Software Quality Engineering, Second Edition. Addison - Wesley, 2003.

[16] S. Donatelli, 1994. Suporposed Generalized Stochastic Petri Nets:Definition and Efficient Solution. Aplication and Theory of Petri Nets , pp. 258-277, 1994.

[17] Watts S. Humphrey Introducción al Proceso de Software Personal. Addison - Wesley, 2001.

[18] Watts S. Humphrey, Software Engineering Institute. Using A Defined and Measured Personal Software Process. *IEEE Software, 0740-7459/96*, May 1996.

[19] Xiaoming Zhong, Nazim H. Madhavji, Khaled El Emam. Critical Factors Affecting Personal Software Processes. *IEEE Software, 0740-7459/00*, November/December 2000.

[20] Gina C. Green, Alan R. Hevner. The Successfull Diffusion of Innovations:Guidance for Software Development Organizations. *IEEE Software, 0740-7459/00*, November/December 2000.

[21] Leticia Dávila Nicanor, Pedro Mejía Álvarez. Evaluación de la Calidad de Software en Sistemas de Información en Internet. Congreso de Ingeniería Electrica, CINVESTAV - IPN, 2003.

[22] R. Stallman, *Linux and the GNU Project*, `http://www.gnu.org/gnu/linux-and-gnu.html`

[23] Kevin Yank, *Building a Database-Driven Web Site Using PHP and MySQL*, `http://www.mysql.com/articles/ddws/index.html`

[24] The Apache Software Foundation , *The Apache Software Foundation*, `http://www.apache.org`

[25] The PHP Group, *The PHP's site* , `http://www.php.net`

[26] J. Raj. The Art of Computer Systems Performance Analisys. ISBN 0-471-50336-3. John Wiley and Sons, 1991.