Centro de Investigación y de Estudios Avanzados
del IPN

Departamento de Computación

Evaluación y Análisis de la Calidad en los Sistemas en Internet

Tesis Doctoral

Leticia Dávila Nicanor

Supervisada por: Dr. Pedro Mejía Álvarez

Junio 2008

Agradecimientos

A mis padres, mis hermanos y a mi hijo les dedico este trabajo que he podido sacar adelante gracias a su apoyo y cariño incondicional que siempre me han brindado y gracias a los cuales he llegado hasta este punto.

A mi asesor le agradezco el apoyo que siempre me dio en todos los sentidos, su guía fue determinante abriendo mi camino hacia senderos de luz y conocimiento para culminar este trabajo con entereza.

A Dios por haber iluminado mi camino y mi estancia cuando me trajo hasta el Cinvestav donde he conocido personas muy valiosas en todos los sentidos que me han ayudado a superarme profesional y personalmente.

M. en C. Leticia Dávila Nicanor.

Abstract

Software reliability is a necessity for any organization that develops and maintains software systems. During the development process, the test phase is the phase that occupies a greater period of time for its execution. Normally this phase can use from 40 to 60development process and in spite of occupying this great amount of time, usually never produces fault free systems.

So far in the area of Software Reliability hundred of models have been proposed and used to estimate the reliability and the probability of failure in the system components. However the vast majority of these are not applicable in real operational contexts. Some of those, who currently have demonstrated their applicability, still present problems such as, difficulties in obtaining inputs, slow in its execution, require an extensive mathematical knowledge for its solution, and in general present great difficulties to locate faults and their causes, which greatly complicates their interface with the testing phase.

The contribution of this work focuses on the development of processes and methodologies for the evaluation and analysis of reliability in software systems for the Internet. In this thesis work, we use formal techniques such as statistical modeling and graphs for the analysis and evaluation of reliability. One of the most important objectives of our work is that the proposed methodologies and processes can serve as a model in the checking and verification of quality for organizations that develop software products under a reliable approach. One of our main objectives in developing the model was its applicability in real operational contexts.

The methodologies we have developed are applicable in practice through the use of a CASE tool we developed, and its results were compared with other current reliability modeling proposals with satisfactory results.

Resumen

La Confiabilidad del software es una necesidad para cualquier organización que desarrolla sistemas de software. Durante el proceso de desarrollo, la fase de pruebas es la fase que ocupa un mayor periodo de tiempo para su ejecución. Normalmente esta fase puede durar del 40 al 60 % del tiempo total empleado para el proceso del desarrollo y a pesar de ocupar esta gran cantidad de tiempo, normalmente los sistemas numnca estan libres de fallos.

Hasta ahora en el área de la Confiabilidad del Software se han propuesto más de un ciento de modelos que estiman el nivel de la Confiabilidad y que determinan la probabilidad de fallo en los componentes del sistema. Sin embargo la gran mayoría de estos no son aplicables en contextos operativos reales. Algunos de los que actualmente si han demostrado su aplicabilidad presentan dificultades tales como, entradas difíciles de obtener, lentas en su ejecucion, requiere de un conocimiento matemático extenso para su solución y en general con una gran dificultad para localizar fallos y sus causas, lo cual complica mucho su interfaz con la etapa de pruebas.

La contribución del presente trabajo se enfoca en el desarrollo de procesos y metodologías para la evaluación y el análisis de la Confiabilidad en productos de software para Internet. En este trabajo, se emplean técnicas formales como la modelación estadística y la técnica de grafos en el análisis de fallos para la evaluación de la Confiabilidad. Uno de los objetivos más importantes de nuestro trabajo es que las metodologías y procesos propuestos puedan servir como modelo en la validación y verificación de la calidad para las organizaciones que desarrollan productos de software bajo un enfoque confiable. Uno de nuestros principales objetivos en el desarrollo de los modelos fue su aplicabilidad real en contextos operativos.

Las metodologias desarrolladas las hemos llevado a la práctica mediante herramientas y sus resultados los comparamos con otras propuestas actuales de modelación de la confiabilidad obteniendo resultados satisfactorios.

Índice general

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

La *Confiabilidad* es una disciplina de la Ingeniería de Software. En la actualidad es una de las áreas de mayor interés para todos los usuarios y desarrolladores de sistemas de software. Debido a este interés se ha convertido en una área de recientes inovaciones científicas y tecnólogicas. Algunos de los aspectos más importantes que se han desarrollado son:

- El desarollo de técnicas para la modelación de la Confiabilidad tomando al sistema como un todo.
- El desarrollo de nuevos enfoques del proceso de desarrollo con un enfoque confiable.
- El desarrollo de técnicas de modelado de la Confiabilidad para ver al sistemas como un conjunto de elementos ó componentes que interactúan con un objetivo común.
- El desarrollo de técnicas para predecir fallos durante la etapa de pruebas.
- El desarrollo de herramientas de software para apoyar la etapa de pruebas.

La calidad de los sistemas de software se estima en función de los atributos de calidad a los que aspira, dependiendo de los objetivos para los cuáles fue desarrollado y de su contexto de operación. Un atributo de calidad que se exige en cualquier sistema de software es la Confiabilidad, la cuál refleja la confianza del usuario en la operación libre de errores del sistema.

Debido a la importancia de la Confiabilidad para los sistemas de software en general, se han desarrollado procesos, prácticas y metodologías para el desarrollo de software confiable, lo cual ha

generando un área de estudio denominada Ingeniería de la Confiabilidad del Software (ICS). Sus autores (Musa, Iannino y Okumoto (1987)), han basado este enfoque en teorías relacionadas con el análisis de los perfiles operacionales, la aplicación de los modelos aleatorios durante el proceso de evaluación, estimaciones estadísticas y teoría de muestreo secuencial.

Los beneficios de seguir un proceso de desarrollo, basado en la Ingeniería de la Confiabilidad permiten mejorar, estandarizar y extender las prácticas del desarrollo en los productos de software. En la actualidad, grandes empresas e instituciones que desarrollan sistemas de software con calidad, tales como AT&T, Alcatel, Bellcore, CNES, ENEA, Ericsson Telecom, Telecom Francia, Hewlett-Packard, Hitachi, IBM, Lockheed-Martin, Lucent Technologies, Microsoft, Mitre, Motorola, los Laboratorios de Propulsión de Jets de la NASA, Nortel, la Universidad de Carolina del Norte, Raytheon, Aviones Militares de Saab, Tandem Computers, la Fuerza Aérea y la Marina de los Estados Unidos, por mencionar algunos, actualmente desarrollan software con un enfoque confiable.

El estudio de la Confiabilidad bajo el enfoque ICS se ha abordado desde tres líneas de investigación: la Evasión de Fallos, la Tolerancia a Fallos y la Detección de Fallos.

La Detección de Fallos son los aspectos centrales de nuestra investigación. En torno de ésta línea de investigación se han realizado múltiples trabajos, cuyo objetivos principales ha sido detectar los fallos antes de la operación productiva de los sistemas de software. Para este efecto se han propuesto técnicas para mejorar la efectividad en la localización de los fallos de forma tal que ayude al desarrollo de la aplicación de software durante la fase de pruebas.

Algunas de las principales limitaciones de los estudios actuales son las siguientes:

- La validéz en la predicción de las hipótesis de los modelos de evaluación no concuerda con los resultados obtenidos en la práctica.
- La entrada para estimar la mayoría de los modelos es costosa y difícil de obtener.
- Para estimar el nivel de confiabilidad los actuales estudios se basan en modelos universales. Este enfoque limita la evaluación del sistema en y dificulta la localización de fallos particulares.
- Los actuales estudios para determinar la procedencia de fallos sólo se limitan a determinar la funcionalidad sin tomar ningun otro atributo del sistema.
- En los actuales estudios requiere de técnicas matemáticas complicadas y de sistemas de software muy complejos para el desarrollo de modelos de evaluación estadística.

- Los modelos propuestos son dificiles de aplicar a sistemas grandes y complejos.
- Son pocas las propuestas de evaluación y modelación que se han llevado a la práctica.
- Las herramientas del proceso de automatización de pruebas que existen en el mercado son muy limitadas, dependen del modelo para el cual fueron dise *n*adas y ofrecen poca funcionalidad.

La contribución del presente trabajo se enfoca al desarrollo de procesos y metodologías para la evaluación y el análisis de la Confiabilidad de los sistema de software desde un enfoque ICS. Tomamos como caso de estudio un sistema en Internet. En este trabajo nos basamos en el desarrollo de modelos predictivos del comportamiento del sistema de software evaluado para establecer nuestros análisis.

En la evaluación de los sistemas de software proponemos utilizar modelos estadísticos que puedan tomarse como modelos de referencia para estimar el nivel de Confiabilidad. Esta propuesta sigue la política de la Confiabilidad a la Medida del enfoque ICS, en donde sea posible evaluar la confiabilidad considerando las caracteristicas y los atributos del sistema de software.

En nuestro estudio de la confiabilidad seguimos dos enfoques.

El primer enfoque obtiene la confiabilidad del sistema considerándolo como una caja negra. Las entradas y las salidas están basadas en la funcionalidad del sistema. EL objetivo de este enfoque consiste en comparar el funcionamiento ideal contra el funcionamiento real. De la diferencia resultante se lleva a cabo un análisis para determinar que tantos fallos tiene el sistema real. En este enfoque no se localizan fallos particulares, sino que solo se obtiene la confiabilidad total del sistema en base a la diferencia entre la operación real y la ideal.

En el segundo enfoque, se lleva a cabo un análisis de la funcionalidad y complejidad del sistema y se proponen modelos estadísticos predictivos para detectar las zonas del software que presentan una mayor probabilidad de presentar fallos. Se lleva a cabo un análisis del código de la aplicación para encontrar su complejidad, la cual se relaciona con la probabilidad de fallos y la relacion entre los componentes. El sistema se estudia como una caja blanca donde el objetivo es estudiar la relacion que guardan los componentes para estimar las probabilidades de fallos, las rutas de fallo críticas y la cobertura de las evaluciones con la finalidad de mejorar la efectividad en la localización y corrección de fallos.

1.1. Objetivos generales y específicos de la tesis

General

El objetivo de estudio en nuestra tesis es mejorar el proceso de evaluación y análisis de la confiabilidad en sistemas de software. Este objetivo lo pretendemos abordar mediante el estudio de la funcionalidad del sistema tomando como marco de referencia el enfoque ICS. Así mismo proponemos basar nuestros estudios en modelos formales utilizando métricas de calidad de software.

Las aportaciones específicas de ésta tesis son:

1. La propuesta de una metodología que nos proprcione un patrón de referencia a nivel práctico de la confiabilidad del sistema de software evaludado.

   Esta propuesta se llevara a cabo mediante los siguientes desarrollos:

   - El desarrollo de modelos con una validéz predictiva objetiva basada en la práctica.
   - El desarrollo de modelos con entradas claras y sencillas de obtener que se pueden automatizar mediante algoritmos de baja complejidad.
   - El desarrollo de modelos formales que no requieran de complicados procesos matemáticos.
   - El desarrollo de modelos para sistemas de gran escala.

2. La propuesta de un modelo para encontrar la confiabilidad global del sistema, la cual permita localizar fallos en los componentes.

   Este modelo contará con los siguientes desarrollos:

   - Desarrollo de un proceso de evaluación de la confiabilidad que incluya métricas de calidad durante la fase de pruebas.
   - El desarrollo de un análisis estadśtico que permita localizar la procedencia de los fallos. En este análizis el sistema será evaluado como una caja blanca en donde sea posible estudiar la relación de los componentes, las rutas criticas y la cobertura de las evaluaciones del sistema de software.
   - El desarrollo de herramientas para automatizar el proceso de evaluación.
   - La comparación del modelo propuesto contra un modelo estocástico actual que halla sido llevado a cabo en la práctica.

1.2. Infraestructura

Para lograr llevar a cabo nuestros objetivos, se utilizará un caso de estudio para aplicar la metodología, en el proceso de desarrollo y en el producto de software. El producto de software que utilizaremos es un sistema de Inscripciones en Internet, tomamos el modelo de inscripción del Departamento de Ingeniería Eléctrica (*DIE*). El (SIV) se desarrolló tomando en cuenta teorías, procesos y técnicas de la Ingeniería de Software.

Los elementos que soportan el sistema SIV son:

- *Un servidor de Base de Datos.* El manejador de Bases de Datos será *Mysql* [104].
- *Un servidor de páginas Web.* En este caso es *Apache* [105].
- *El lenguaje para desarrollar las interfaces: PHP* [106].
- *El sistema operativo.* En nuestro caso es *Linux* [20].
- *Conectividad con Internet.*

1.3. Organización del documento

La presente tesis está estructurada en cinco capítulos.

- En el capítulo 2 se introduce el concepto de la Confiabilidad.
- En el capítulo 3 se presenta la metodología y el proceso para el proceso de evaluación. Estos son aplicados al caso de estudio.
- En el capítulo 4 se presenta la metodología y el proceso para el análisis de la procedencia de los fallos. Estos son aplicados también al caso de estudio.
- En el capítulo 5 presentamos un estudio comparativo de nuestro modelo contra el modelo propuesto por Whittaker [99]. En este estudio se ven claras las ventajas de la técnicas que utilizamos contra una de las técnicas clásicas de la modelación estocástica de pruebas para sistemas de software. Este estudio ademas de ser teórico lo llevamos al ámbito práctico.

- Finalmente el capítulo 6 contiene las conclusiones del trabajo realizado y se da un bosquejo de las posibles líneas de trabajo a futuro.
- Adicionalmente se incluyen también apéndices que permiten aclarar conceptos, técnicas y especificaciones del sistema de evaluación.

Capítulo 2

Desarrollo de Productos de Software Confiables

En este capítulo abordaremos el estudio de la *Confiabilidad* y expondremos los conceptos básicos utilizados en el desarrollo de sistemas de software confiables. En este sentido describiremos el proceso de desarrollo desde el punto de vista de la Ingeniería de Confiabilidad del Software. Así mismo, describiremos la relación de la Ingeniería de Confiabilidad de Software con el proceso de evaluación, los elementos que intervienen y finalmente discutiremos las características que hay que tomar en cuenta para evaluar los sistemas en Internet.

2.1. Introducción

La calidad del software se plantea en función a los atributos de calidad que demanda el sistema de software. Esto depende de los objetivos para los cuáles fue desarrollado el sistema de software y también de su contexto de operación. Un atributo de calidad que es necesario en cualquier sistema de software es la *Confiabilidad*. Los estudios para la estimación de la Confiabilidad de los sistemas de software son muy importantes ya que su objetivo es localizar fallos y encontrar sus fuentes.

La *Confiabilidad* actualmente es una de las prioridades en el desarrollo de los sistemas de software con calidad. La Confiabilidad se ha vuelto una disciplina de estudio denominada: *Ingeniería de Confiabilidad de Software (ICS)* . Sus autores Musa, Iannino y Okumoto (1987), han basado sus

estudios en teorías como el análisis de los perfiles operacionales, la aplicación de los modelos aleatorios durante el proceso de evaluación, estimaciones estadísticas y la teoría de muestreo secuencial. Los beneficios de seguir un proceso de desarrollo desde el punto de vista de la *ICS* permiten mejorar, estandarizar y extender las prácticas del desarrollo en los productos de software.

En la actualidad las grandes empresas e instituciones que desarrollan sistemas de software con calidad se han beneficiado de seguir el enfoque de ICS.

2.2. Ingeniería de la Confiabilidad del Software (ICS)

Cuando se desarrolla software con un enfoque ICS [93] es necesario seguir las fases del proceso de Ingeniería de la Confiabilidad del Software en paralelo con el proceso de desarrollo. Las fases del proceso para la ICS son:

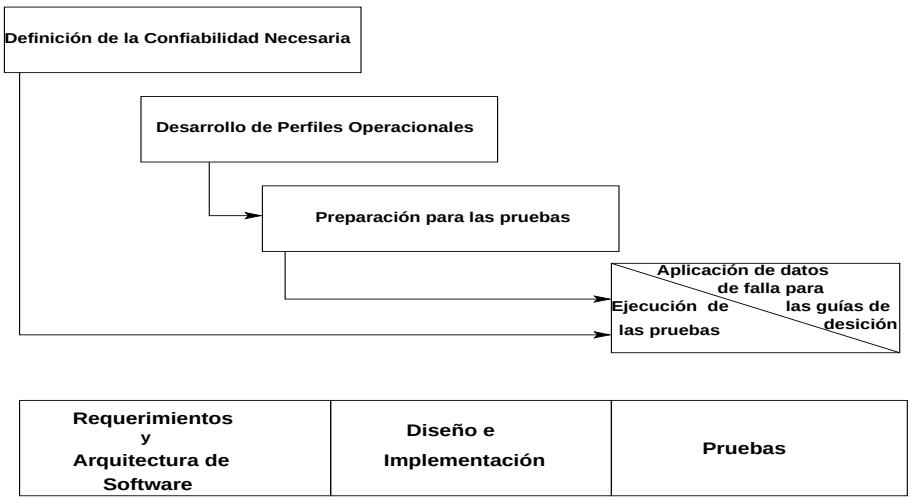

Figura 2.1: Diagrama del Proceso de la Ingeniería de la Confiabilidad del Software.

1. *Definición del Producto de Software.* Cuando se lleva a cabo la definición del producto de software, es necesario establecer quiénes son los proveedores y clientes e incluir una lista de los sistemas asociados. Los sistemas asociados incluyen el producto base a sus variaciones y a otros sistemas relacionados al producto.

2. *Implementación de los perfiles operacionales.* Un perfil operacional en un sistema de software se caracteriza por un conjunto de operaciones, las cuáles tienen una probabilidad de ocurrencia

que garantiza la utilización de esta operación en el sistema por parte de los usuarios. Los perfiles operacionales pueden obtenerse desde los requerimientos.

3. *Ingeniería de Confiabilidad a la Medida*. Las actividades de este proceso son:

   - Definición de *fallo* en términos del proyecto que se desarrolla.
   - Selección de una unidad de referencia para cuantificar los fallos.
   - Establecimiento del objetivo de la intensidad del fallo del sistema (FIO) para el sistema asociado.
   - Para cualquier software:
     - Encontrar el desarrollo del objetivo de la intensidad del fallo del sistema.
     - Seleccionar estrategias de Confiabilidad de software (tolerancia a fallos y revisión de requerimientos) para el desarrollo del objetivo de la intensidad del fallo del sistema.

4. *Preparación de las pruebas*. Durante la planeación de las pruebas se preparan casos de prueba basados en los perfiles operacionales. Esta es una de las etapas que concentra el trabajo que se hace en las fases anteriores.

5. *Ejecución de las pruebas*. La efectividad en la ejecución de los casos de prueba depende en gran medida de la herramienta con la que se realiza la fase de pruebas. Otro punto importante es el tipo de pruebas que se van a realizar en esta fase. Si se desea evaluar solo la funcionalidad del sistema de software, las pruebas de caja negra son suficientes, pero sí lo que se desea es evaluar los componentes o las estructuras de los algoritmos del sistema entonces es necesario aplicar las pruebas de caja blanca.

2.2.1. Determinación de los perfiles operacionales

Los perfiles operacionales [45] reflejan los objetivos y servicios para los cuales fue desarrollado el producto de software. En el establecimiento de los perfiles operacionales es necesario entender el significado de una operación y su implicación. Una operación se define como una tarea lógica *principal* para el sistema, la cual es invocada por uno ó varios inicializadores; el control es devuelto al sistema cuando dicha tarea finaliza. Las tareas *principales* están relacionadas con uno ó varios requerimientos funcionales y no se derivan de las subtareas que se plantean en la arquitectura del diseño.

En la ejecución de una operación pueden intervenir un conjunto de elementos de software y hardware. De esta forma un perfil operacional en un sistema de software se caracteriza por un conjunto de operaciones, las cuales tienen una probabilidad de ocurrencia que garantiza la utilización de cada operación en el sistema por parte de los usuarios [68]. La probabilidad de ocurrencia se refiere a la estimación de la utilización de cada operación en relación con todas las operaciones que ejecuta el sistema.

Los cinco puntos principales para determinar adecuadamente un perfil operacional son:

1. *Identificación de los inicializadores de las operaciones.* Los inicializadores de operación incluyen tipos de clientes (vistas), sistemas externos y controladores del sistema. Para identificar las vistas de los usuarios primero se estiman las expectativas de los clientes y los servicios esperados, basados en el caso de negocio que fue preparado previamente para justificar el desarrollo del producto de software.

2. *Creación de listas de operación.* Para la creación de la lista de operaciones asignadas a cada perfil es necesario partir de los requerimientos funcionales. Cuando tenemos esta lista es recomendable complementar la información con otras fuentes como los diagramas de flujo, los casos de uso, las versiones anteriores, etc.

3. *Revisión de la lista de operación.* En este proceso se revisa y cuestiona una serie de factores prácticos acerca de las operaciones como son el tiempo de operación, la congruencia, la consistencia, la probabilidad de ejecución y el tamaño del sistema. Cada operación debe poseer sustanciales diferencias de procesamiento comparado con las otras. Las operaciones deben estar bien formadas y en el caso de que envíen mensajes y desplieguen datos, estos procesos deben formar parte de la operación y no constituir la operación misma.

   Otro aspecto importante es que la probabilidad de ocurrencia de una operación debe ser considerablemente alta en la ejecución de la mayoría de las operaciones determinadas. Las operaciones con baja probabilidad de ocurrencia pueden ser descartadas. Por último, el número total de operaciones debe ser igual ó menor en comparación al tamaño del sistema.

4. *Determinación de la taza de ocurrencia.* Una taza de ocurrencia de una operación es el número de ocurrencias de la operación dividida entre el tiempo del conjunto total de operaciones que se ejecutan. La taza de ocurrencia debe ser expresada con respecto a un mismo periodo de tiempo; es decir a un marco de tiempo constante.

Para muchos analistas, les resulta difícil la primera vez que calculan estas tasas, debido a que normalmente estos datos quedan sujetos a estimaciones indirectas. Normalmente en las primeras ocasiones es recomendable basarse en antiguas versiones del sistema ó en sistemas similares.

5. *Determinación de la probabilidad de ocurrencia.* La probabilidad de ocurrencia es un parámetro que permite establecer si es necesario incluir una operación como parte de la funcionalidad del sistema. Si la probabilidad es muy baja implica que la operación definida no es prioritaria para el sistema. Con lo cual, es posible que esta operación sea parte de la funcionalidad de otra operación con probabilidad de ocurrencia mayor.

   La probabilidad de ocurrencia se puede definir como la razón de la taza de ocurrencia de cada operación sobre el total de las tazas de ocurrencia.

El diseño de los perfiles operacionales comienza desde la etapa de requerimientos. Esta lista se refina iterativamente en las siguientes fases del proceso de desarrollo.

2.2.2. Planeación de pruebas en base a los perfiles operacionales

Cuando el software falla es necesario localizar e identificar las fallas que causan los defectos en el sistema. Esto es posible mediante un *proceso de evaluación.* El proceso de evaluación se realiza mediante las pruebas del software y utilizando métricas. Desde el punto de vista de la producción, cuando no hay métricas de producto, no existen criterios consistentes para decidir cuándo el producto es estable. Uno de los objetivos principales del proceso de evaluación es detectar los defectos.

El proceso de evaluación puede realizarse mediante los siguientes enfoques:

- *Pruebas de caja negra.* Estas pruebas evalúan la funcionalidad general del sistema. En estas pruebas los casos de prueba son derivados de la especificación del software y no se consideran detalles de implementación.

- *Pruebas de caja blanca.* Estas son pruebas estructurales que permiten evaluar los componentes, los algoritmos y las estructuras de datos del sistema. Estas pruebas son complementarias a las pruebas de caja negra. Es decir, que aportan datos adicionales a la evaluación.

De acuerdo a la ICS el proceso de evaluación siempre debe estar basado en los perfiles operacionales. Los perfiles operacionales nos darán información acerca de cómo los usuarios podrán utilizar el producto de software desarrollado.

El proceso de pruebas inicia con una planeación en la etapa de requerimientos; tomando como punto de partida los requerimientos funcionales. Todos los requerimientos funcionales deberán ser probados al menos una vez. Con base en los requerimientos funcionales se establecen los perfiles operacionales y la funcionalidad de los servicios que se van a evaluar del sistema de software.

2.3. 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 ó 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 ó de los procesos de software.

Existe un rechazo en algunas compañias que desarrollan software para introducir medidas debido a que el costo de implementación es alto y los beneficios no son claros. Una razón de esto es que, en muchas compañías los procesos de 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 mtricas sino hasta que existan disponibles estándares y herramientas que les faciliten su implementación.

A menudo es imposible medir los atributos de calidad del software de forma directa. Los atributos como la confiabilidad, la mantenibilidad o la complejidad, por citar algunos ejemplos, se ven afectados por diversos factores y no existen métricas directas para ellos. Más bien, es necesario 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 espera obtener de la prueba. El proceso de medición es de las actividades más importantes ya que de éste 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.

2.4. Métricas de Software

Una métrica de software es cualquier tipo de medida relacionada con un sistema [26, 6, 94], 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. 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 a la medida, adecuadas a su marco de producción [94],.

2.4.1. 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 cómo 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 y 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.

2.4.2. Tipos de métricas de software

Las métricas de software pueden clasificarse en tres categorías [26],:

- 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 ésta tesis son aquellas que nos pueden ayudar a evaluar la calidad [94]. Dichas métricas son un subconjunto de las del producto y el proceso. Las métricas de calidad están divididas 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 intrínseca del producto y la satisfacción del cliente con respecto al producto. La tercera está en función del mantenimiento durante el ciclo de vida esperado para el producto de software.

2.4.3. Métricas de producto final

Las métricas de producto final más importantes para la Confiabilidad 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 intervalo de tiempo, durante la operación del producto de software.

3. *Problemas detectados por el usuario.* Esta métrica se utiliza con mayor frecuencia dentro de las empresas. Consiste en que el usuario pruebe por un tiempo determinado el producto de software y que reporte que problemas encuentra durante la evaluación del sistema. 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.

De acuerdo a la IEEE/ANSI (Instituto Nacional de Estándares de los E.E.U.U.) [35] 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 límites especificados para la unidad.

2.4.4. Métricas de Complejidad del Software

La teoría de la complejidad computacional es la parte de la teoría de la computación que estudia los recursos requeridos durante el cálculo para resolver un problema [94]. Los recursos comúnmente estudiados son el tiempo (número de pasos de ejecución de un algoritmo) y el espacio (cantidad de memoria utilizada).

Para probar la complejidad de un producto de software podemos utilizar su cobertura. Esta cobertura puede calcularse en función del porcentaje de instrucciones de requerimientos funcionales probados y se puede representar con un grafo de control. Cuando se utilizan grafos de control, los nodos denotan acciones, y los vértices conectan estas acciones con otras generando las rutas. Una ruta es una secuencia de nodos conectados por los vértices. El grafo puede contener ciclos que representan parte de la estructura del programa. En un caso ideal, las pruebas recorren todas las rutas posibles. Los nodos que contienen una condición, tal como una expresión lógica en una senten-

cia condicional, pueden ser una combinación de predicados elementales conectados por operadores lógicos. Estos casos requieren al menos dos pruebas para cubrir la condición.

En las condiciones múltiples se requiere que todas las posibles combinaciones de predicados elementales en las condiciones sean cubiertas por el conjunto de pruebas. Una métrica de complejidad que está de acuerdo a este contexto es la Complejidad Ciclomática de McCabe.

La Complejidad Ciclomática de McCabe es una métrica de complejidad del producto de software que establece el mínimo de rutas que es necesario para evaluar una sección específica del sistema [71]. Esta ruta es determinada mediante grafos de donde se obtienen las rutas de evaluación. El número de McCabe está asociado a cada módulo de la arquitectura del producto de software.

La complejidad ciclomática de McCabe (CV) nos da el número de rutas que son linealmente independientes:

$CV(G) = V(G) + 1$

Donde: $V(G) = E - N + p$

E = número de vértices.

N = número de nodos.

p = número de componentes.

Asi sea el siguiente algoritmo:

1. Procedure insert (a,b,n,x);
2. Begin bool found := false;
3. for $i := 1$ to n do
4. if $a[i] = x$;
5. Then found := true; goto leave end if
6. end do;
7. Leave;
8. if found
9. then $b[i] := b[i] + 1$;

10. else $n := n + 1$; $a[n] := x$; $b[n]i := 1$ end if

11. end insert

El grafo de la Figura 2.2 muestra el flujo de control de un algoritmo estructurado en el cuál se tienen sentencias condicionales. Cada nodo esta asociado con un arco. El número de nodos corresponde a predicados elementales del algoritmo, donde: $E = 13$,$N = 11$, y $p = 1$. Por lo tanto $V(G) = 3$ y $CV(G) = 4$.

Para este grafo, la métrica de evaluación deriva un conjunto de al menos 4 rutas que podrán ser probados y que corresponden al número mínimo de pruebas para el algoritmo.

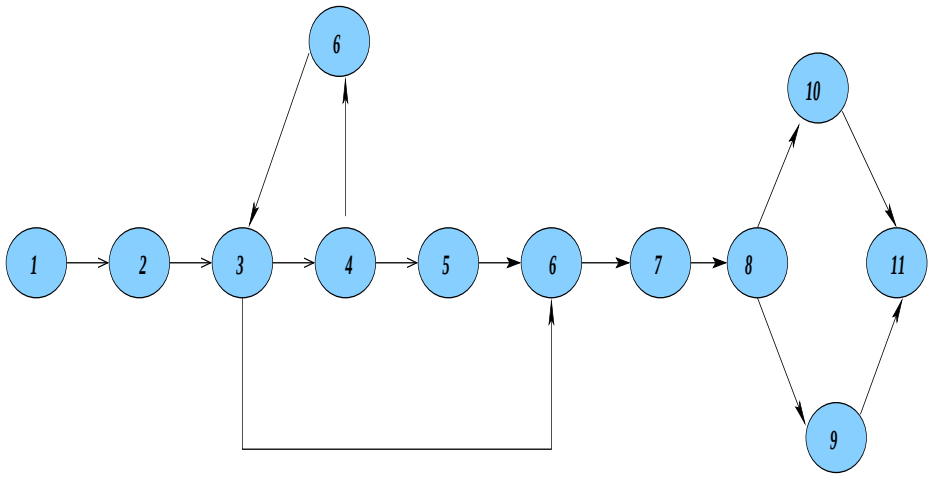

Figura 2.2: Grafo derivado de la estructura del algoritm presentado en esta sección

2.5. Técnicas de análisis para la Confiabilidad

La mejora de la confiabilidad incluye técnicas rigurosas de diseño y revisión del código, generación automática de casos de prueba, asignación estratégica de personal clave y la reingeniería de áreas de alto riesgo en un sistema de software. De ésta manera las técnicas empleadas para el estudio de la *Confiabilidad* del software resultan fundamentales en su estudio.

El estudio de la Confiabilidad permite determinar si hay fallos en un sistema. En el estudio de la confiabilidad existen tres enfoques:

1. *La Evación de Fallos.* Este es uno de los más importantes enfoques el cual implica detectar y eliminar fallos antes de la operación del sistema.

2. *La Tolerancia a Fallos.* Este enfoque asume que aunque se aplicaron pruebas de evación para remover fallos durante la etapa del dise o, muchos de los fallos permanecen durante la operación del sistema. La tolerancia a fallos detecta y tolera estos fallos de forma tal que el sistema continue en operacion, aunque su funcionalidad sea degradada.

3. *La Detección de Fallos.* El objetivo de este enfoque es detectar los fallos antes y despues de que el software sea puesto en operación. Para validar y verificar si existen fallos en el sistema de software se utilizan métodos y técnicas estáticos y dinámicos.

La Evasión, Detección y la Tolerancia a Fallos permiten alcanzar el nivel de Confiabilidad requerido en el desarrollo de los sistemas de software.

2.6. Aspectos de Calidad en los Sistemas en Internet

La *Confiabilidad* es una exigencia cada vez mayor en los Sistemas de Software en general. Esta exigencia se extiende hacia los sistemas en Internet en donde algunos de los aspectos que influyen en esta demanda son:

- Actualmente las aplicaciones Web manejan servicios con alto impacto como los procesos de negocio, transacciones comerciales e intercambio de información.
- El intercambio confiable de la información está en función del volumen y a la relevancia de los datos que las instituciones manejan.
- La globalización de la difusión de los servicios que los sistemas en Internet ofrecen.

La Internet ha generado cambios dramáticos en la forma de llevar a cabo el comercio y las comunicaciones. Con la Internet se han creado nuevas preferencias, expectativas y cambios en los negocios, la educación, la industria y el público en general. En la actualidad el mal funcionamiento de sitios estratégicos como los portales bancarios pueden comprometer oportunidades de negocio y la reputación de las compañias responsables de estos sitios. En Julio y Junio de 1999 el sitio de ebay.com estuvo fuera de operación por 21 horas. Este problema causó pérdidas estimadas en 5 millones de dólares. Como este caso existen muchos ejemplos más. La confiabilidad es una exigencia cada vez mayor en los Sistemas en Internet.

Debido a la creciente necesidad de contar con sitios en Internet que operen con calidad de la Conferencia Internacional de la Ingeniería de Software del año 2002 (ICSE 2002) se centró en los aspectos de Calidad que demandan los sistemas en Internet [90], [85]. En ésta conferencia se concluyó que los criterios de calidad más importantes son:(a) Confiabilidad, (b) Seguridad y (c) Usabilidad. En este trabajo de tesis nos centraremos en la *Confiabilidad*.

Las aplicaciones Web poseen características únicas que hacen la diferencia en la evaluación y el aseguramiento de la calidad del software tradicional 2.3. En un sistema en Internet el estudio de la Confiabilidad con respecto a la Detección de Fallos bajo un esquema dinámico y concurrente de pruebas es lo más apropiado porque tiene una mayor efectividad en el descubrimiento de fallos. Al respecto se han desarrollado diversos trabajos como los de Jeff Tian [96],[97].

Las aplicaciones Web pueden ser caracterizadas por los siguientes aspectos:

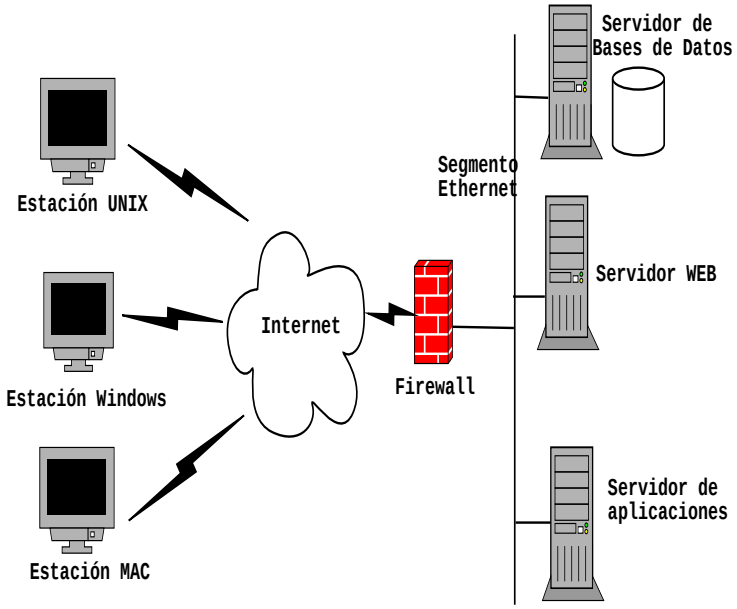

Figura 2.3: Arquitectura común para sistemas en Internet.

1. *Contexto de operación: Acceso masivo de usuarios para los sistemas en Internet.* En los sistemas en Internet el acceso simultáneo de un gran número de usuarios es parte de su naturaleza, lo cual debe resultar transparente para los usuarios. Un ejemplo es la operación de los portales bancarios. Normalmente estas aplicaciones tienen que prepararse para recibir a millones de usuarios de manera simultánea, los cuales manejan grandes volúmenes de información. Esto impacta en el personal involucrado en su desarrollo, operación, mantenimiento y en su in-

fraestructura. Para resolver el problema de la confiabilidad, en este tipo de sistemas se han desarrollado arquitecturas con esquemas de redundancia y recuperación.

2. *Dependencia en la arquitectura del software y en el proceso de desarrollo.* En las aplicaciones en Internet se tiene una mayor dificultad para establecer las causas de los errores, aunque es posible apreciar en estas un mayor impacto cuando ocurre un error. Por ejemplo cuando no se ejecuta adecuadamente una transacción en un portal bancario el impacto para el usuario y la institución responsable del sitio es muy alto. Sin embargo, encontrar la procedencia del error es dificil y no resulta inmediato, debido al gran número de elementos de software que intervienen durante su operación.

3. *La diversidad de arquitecturas para los sistemas en Internet.* Existen hoy en día una gran diversidad de arquitecturas h eterogéneas y distribuidas donde el *backend* se ejecuta para prestar los servicios de la aplicación. Aunque la mayoría de los desarrollos para aplicaciones en Internet opera en ambientes UNIX como son (Linux, Solaris, HP-UX, AIX), en menor escala también hay desarrollos sobre Windows y Mac. En el *frontend*, los usuarios de estos sistemas tienen la opción de elegir distinto software de navegación bajo sistemas operativos Windows, Linux o UNIX.

4. *Diversidad de perfiles para desarrollar, operar y mantener un sitio Web.* El desarrollo de un sitio Web requiere de un gran equipo de personas con diferentes perfiles. 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 parte de la dificultad que se tiene cuando se desarrolla una aplicación en Internet confiable.

Debido a las características antes mencionadas los sistemas Web experimentan modificaciones frecuentes debidas al avance tecnológico, al surgimiento de nuevas necesidades comerciales, así como al arribo de un número mayor de usuarios. En consecuencia los sistemas Web son muy sensibles de experimentar fallos. La mayoría de los desarrollos de sistemas Web tienen como i prioridad proporcionar una amplia funcionalidad en beneficio de los usuarios, sin embargo actualmente en muchos de estos no se tiene en cuenta el aseguramiento de la calidad.

Capítulo 3

Metodología para la evaluación de la Confiabilidad

El objetivo del presente capítulo es presentar nuestra metodología para evaluar la Confiabilidad global en los sistemas en Internet. El término *global* se refiere a la cobertura de la funcionalidad en el sistema de software. La metodología se basa en la estimación de niveles de Confiabilidad mediante pruebas de software en donde los resultados son analizados con la técnica de modelación estadística. La metodología propone un proceso en el cual se evaluará al sistema bajo condiciones ideales y bajo condiciones reales. Este proceso tiene como objetivo determinar la cercanía de la operación real contra el ideal estimado.

3.1. Introducción

En actualidad un gran número de empresas que desarrollan software (en especial software crítico) son obligadas a utilizar estándares y modelos de mejora como el D0-178B, la gama de ISO9000, CMM-SEI y el enfoque ICS. Estos estándares y modelos proveen un conjunto de buenas prácticas para mejorar y evaluar los productos software.

Dentro del enfoque ICS una de las primeras fases es la determinación de la Ingeniería de Confiabilidad a la Medida, en donde se lleva un estudio de las características funcionales del sistema.

La contribución del presente trabajo se enfoca en el desarrollo de una metodología para evaluar la *Confiabilidad global*. La metodología propuesta opera bajo un enfoque de Confiabilidad a la Me-

dida en donde la técnica de análisis utilizada es la teoría de modelación estadística. Inicialmente evaluamos al sistema bajo condiciones ideales para establecer el modelo de referencia y bajo condiciones reales determinamos el modelo de operación real. Cuando se tienen ambos modelos es posible determinar la cercanía que presenta la operación real de la ideal.

3.2. Trabajo Relacionado

Los modelos de confiabilidad son usados para asentar la confiabilidad de un producto de software ó para estimar el número de defectos latentes.

Los modelos de confiabilidad son importantes por las siguientes razones:

- Por que representan una estimación de la calidad en el producto.
- Por que permite estimar los recursos de planeación en la fase de corrección de errores y en la fase de mantenimiento.

En términos generales los modelos de Confiabilidad pueden ser clasificados en dos categorías: *Modelos Estáticos y Modelos Dinámicos*. Un modelo dinámico, usualmente se basa en distribuciones estadísticas, en donde se utiliza el actual patrón de defectos para estimar la confiabilidad del producto final. Los modelos estáticos ayudan a los ingenieros de software a mejorar la calidad de su diseño e implementación. El modelo que proponemos en esta primera propuesta es un Modelo Dinámico.

De acuerdo al grupo de expertos que establecen las bases de confiabilidad (Iannino en 1984), en general un buen modelo de evaluación debe de cumplir con los siguientes criterios:

- *Validación en la Predecibilidad*. La cual es la capacidad del modelo para predecir un ambiente de fallos ó el numero de defectos para un periodo especifico de tiempo, basados en datos de entrada actualizados.
- *Capacidad*. La habilidad del modelo para cuantificar con exactitud las necesidades de administradores, ingenieros y usuarios en la administración y control de proyectos de desarrollo de software.
- *Clausulas de Calidad*. Las clausulas que el modelo describe deben ser consistentes y logicas para los Ingenieros de Software.

- *Aplicabilidad.* El modelo debe ser capáz de operar en cualquier sistema de software sin importar el tamaño, su estructura ó su número de funciones.

- *Simplicidad.* Un modelo sebe ser simple en tres aspectos:

  1. La colección de datos de entrada debe ser simple y de bajo costo.
  2. Debe ser simple en su concepción matemática, sin que sea necesario un amplio conocimiento matemático.
  3. Debe ser facil de implementar en cualquier ambiente computacional.

La modelación global de la Confiabilidad del Software puede ser clasificada en dos clases, dependiendo de las variables dependientes del modelo. Para *Modelos de Tiempo Entre Fallos*, la variable de estudio es el tiempo entre fallos. Para *Modelos de Conteo de Fallos* la variable que se estudia es el número de fallos en un marco de tiempo especifico. El tiempo puede ser tiempo de ejecución de CPU ó tiempo de calendario como horas, semanas ó meses. El número de defectos es tratado como una variable aleatoria.

Entre los Modelos mas importantes que se orientan al tiempo de arribo entre fallos están los siguientes

- **El modelo de Jelinski Moranda (1972)**. Este modelo [38] asume que ocuuren N fallos en el sistema de software al comienzo de las pruebas. Estos fallos ocurren aleatoriamente y todos los fallos contribuyen de manera equitativa para generar fallos subsecuentes. El modelo también asume que el tiempo de corrección es despreciable y que la precisión con la que se corrigen los fallos es perfecta. Estas condiciones en un ambiente operativo son difíciles de encontrar, por lo que sus resultados se alejan de la realidad.

- **El modelo de Littlewood**. Este modelo [39]asume condiciones similares al de Jelinski - Moranda, excepto que los fallos influyen forma inequitativa para generar fallos subsecuentes. En este modelo de acuerdo al progreso de las pruebas el arribo de los fallos sigue un función decreciente, sin embargo el tamaño del error es siempre el mismo.

- **El modelos de Goel-Okumoto**. En el modelo de Jelinski Moranda se asume que el tiempo de corrección es despreciable y que las correccciones son perfectas. En la práctica estas clausula no son reales. Cuando corregimos defectos en un sistema de software existe una alta probabilidad de insertar nuevos defectos durante el proceso de corrección. Por ello Goel

Okumoto proponen un modelo [40] que toma como base un revisión imperfecta. Este modelo estima un funcion de riesgo durante el intervalo entre el anterior $(i-1)$ y el actual fallo i.

$$Z(t_i) = [N - p(i-1)]\lambda \tag{3.1}$$

Donde:

$N =$ El número de Fallos cuando inician las pruebas.

$p =$ Es la probabilidad de una revisión imperfecta.

$\lambda =$ Es la taza de error por fallo.

De los Modelos más importantes que se orientan al de fallos que ocurren en un marco de tiempo determinado tenemos:

- **EL modelo NHPP (Proceso de Poisson No Homogeneo) de Goel y Okumoto**. Este modelo [41] se deriva de los fallos que son detectados durante un intervalo dado durante la etapa de pruebas. Goel y Okumoto proponen que el número de fallos observados en un tiempo t, $N(t)$ pueda se modelado como un proceso de Poisson no homogeneo con una taza de fallos dependiente del tiempo. Ellos proponen que la taza de fallos siga un distribución exponencial. El modelo es definido por la siguiente expresión:

$$P\{N(t) = y\} = \frac{[m(t)]^y}{y!}e^{-m(t)}, y = 0, 1, 2, \ldots \tag{3.2}$$

Donde:

$m(t) = a(1 - e^{-bt})$

$\lambda(t) \equiv m'(t) = abc^{-bt}$

En el modelo, m(t) es el numero esperado de fallos observados durante el tiempo t; $\lambda(t)$ es la densidad de fallos: a es el número esperado de fallos para ser observados eventualmente; y b es la taza de error para cada fallo. Así $m(t)$ y $\lambda(t)$ son distribuciones de probabilidad acumulativa.

Para detectar el número de fallos en este modelo, a, es tratada como una variable aleatoria que depende de las pruebas y de otras variables de ambiente. Esta es la diferencia fundamental con los modelos que tratan el número de fallos como un valor desconocido y constante.

La distribución exponencial asume un patrón decreciente de la taza de defectos ó fallos. Se ha obeservado que en varios casos la taza de fallo incrementa y luego decrementa. Goel en 1982 propone un modelo generalizado que aloja mas de un parametro para determinar una media y una densidad para estimar la taza de fallos. El modelo que propone es el siguiente:

$$m(t) = a(1 - e^{-bt^c})\lambda(t) \equiv m'(t) = abce^{-bt^c}t^{c-1} \tag{3.3}$$

Donde a es el número esperado de fallos que pueden ser eventualmente detectados, y b y c son constantes donde se refleja la calidad de las pruebas. La media y la densidad de defectos representan actualmente la distribucion de Weibull.

- **Modelo de Ejecución Logaritmica de Poisson (Musa Okumoto)**. Este modelo [42] tambien determina el numero de fallos en un cierto tiempo,τ,aunque este asume un Proceso de Poisson Logaritmico. En este caso, los autores argumentan que el Proceso de Poisson Logaritmico es mejor para modelar una operacion no uniforme de los perfiles operacionales, cuando algunas funciones son ejecutadas mucho mas frecuentemente que otras. Este modelo es ejecutado en intervalos de tiempo (simulando tiempo de calendario:horas, días, meses). También se propone un enfoque sistematico para convertir los resultados a datos de tiempo de calendario. El modelo se forma de dos componentes, el componente de tiempo de ejecución y el componente de tiempo de calendario.

$$u(\tau) = \frac{1}{\theta}(\lambda_0\theta_\tau + 1) \tag{3.4}$$

Donde λ es la intensidad inicial de fallos, y θ es la taza de recuccion en la intensidad de fallos normalizada para cada fallo.

- **Modelos S con retrazo e inflexión**. Yamada en 1983 [43] argumenta que un proceso de pruebas consiste en un buen proceso de deteccion de defectos y en la captura de este defecto. Sin embargo el tiempo necesario para la captura del defecto implica un cierto retrazo con respecto del momento en el que se produjo. Para el tratamiento de esta situacion, los autores ofrecen un modelo que implica un cierto retrazo en la funcion de crecimiento del modelo. Este modelo tambien esta basado en un Proceso de Poisson no Homogeneo. El modelo es el siguiente:

$$m(t) = k[1 - (1 + \lambda t)e^{-\lambda t}] \tag{3.5}$$

Donde t es el tiempo, λ es la taza de detección de errores y K es el número total de defectos o taza acumulativa total de defectos.

En 1984 Ohba [44] propone otro modelo que se denomina con retraso e inflexión S. El modelo describe un dependencia de fenomenos de fallos con una dependencia mutua en la detección de defectos. Específicamente, según este modelo la mayoría de los fallos no detectados llegan a ser detectables. Este modelo tambíen se basa en un Proceso de Poisson no Homogeneo. El valor de la media esta de acuerdo a la función:

$$I(t) = K\frac{1 - e^{-\lambda t}}{1 + ie^{-\lambda t}} \tag{3.6}$$

donde t es el tiempo, λ es la taza de detección de errores, i es el factor de inflexión y K es el número total de defectos ó la distribución acumulativa.

El retardo S y la inflexión se toman en cuenta para el periodo de entrenamiento en el cual los evaluadores se familiarizan con el software al comienzo del periodo de pruebas. El periodo de pruebas es asociado con el retardo ó patrón de inflexiones como lo describen la distribucion acumulativa. El modelo exponencial asume que la mayoría de los defectos arriban al comienzo de la etapa de pruebas y despues declina. El retardo asume que las pruebas comienzan y que los defectos tiene un retardo en aparecer y en el modelo de inflexión el retardo es mas nítido. El comportamiento de estos modelos se pueden observar en la siguientes Figuras 3.1,3.2 y 3.3)

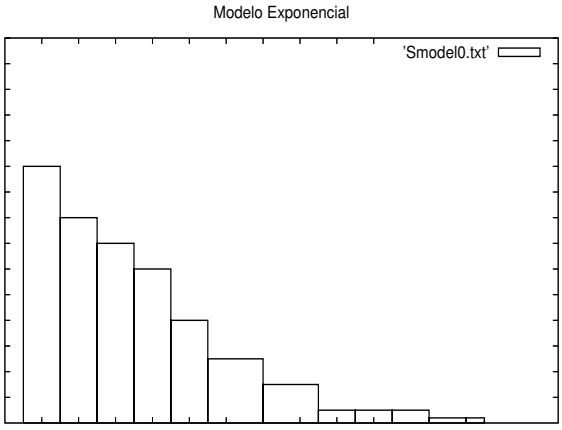

Figura 3.1: Modelo Exponencial

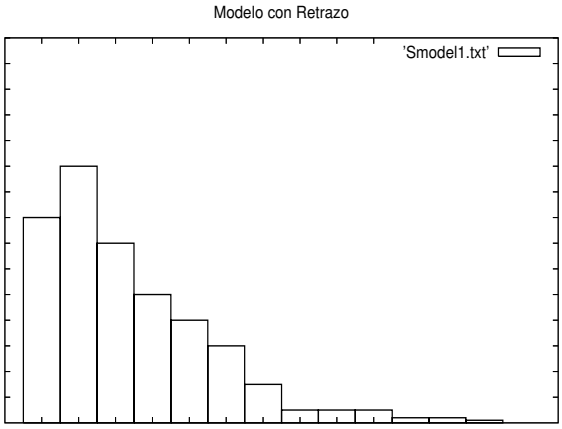

Figura 3.2: Modelo con Retrazo S

Además de los modelos descritos anteriormente, en las dos decadas pasadas han sido propuestos mas de un ciento de otros modelo. Cada uno de estos modelos con sus propias hipótesis, aplicaciones y limitaciones. Desafortunadamente, de acuerdo a estudios recientes, tambien se ha visto que la gran mayoría de estos modelos no han sido evaluados con datos ni ambientes reales, por lo cual pocos realmente se encuentran hoy dia en uso. Desde un punto de vista práctico, para algunos modelos el costo de la obtención de los datos de entrada es muy alto, y en otros casos los modelos mismos no son muy claros o simplemente no operan en ambientes reales, sino que solo bajo ambientes simulados.

Estas conclusiones, se confirman en el estudio de Elbert [37, 94] en donde se realizan un estudio con los siete modelos mas representativos del área, y en donde las entradas provienen de un sistema

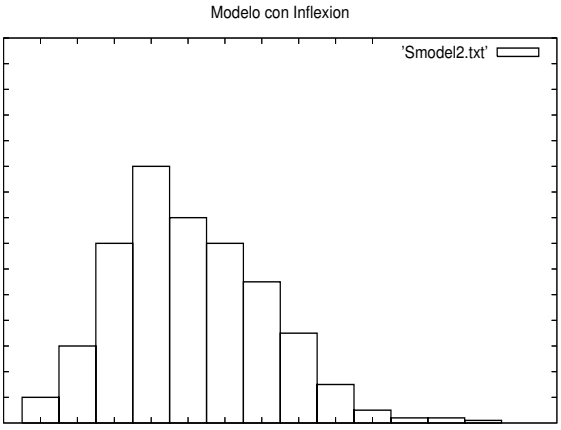

Figura 3.3: Modelo con Inflexión

muy grande y complejo conformado por varios millones de lineas de código. La conclusión de Elbert es de que en la mayoría de los modelos, estos son dificiles de desarrollar, sus entradas son dificiles de obtener y en general los resultados que brinden son poco realistas y con poca conección con la etapa de pruebas.

3.3. Establecimiento de la Metodología

Las fases utilizadas en nuestra metodología son las siguientes:

1. **Evaluación del atributo de calidad en el sistema ideal.**

   La meta en esta fase es obtener un modelo que describa el comportamiento de los atributos de calidad para un sistema ideal. Para el funcionamiento del sistema ideal se utilizarán técnicas de simulación. Durante la simulación aplicaremos el proceso de evaluación y el análisis que describiremos en la sección 3.3. 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. Esto es 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 del atributo de calidad en el sistema real (inicial).**

   En esta fase nuestra meta es obtener un modelo que evalúe cualitativamente el 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 3.3.1.

3. **Análisis de los resultados y conclusiones.** En esta fase, se llevará a cabo un análisis de las evaluaciones obtenidas del producto de software. Así mismo, se llevará a cabo una comparación de los objetivos planteados contra los objetivos alcanzados. Este análisis se registra en bitácoras, ya que servirá para mejorar la calidad de productos de software futuros.

   Teniendo el análisis es posible entonces determinar si es necesario mejorar la calidad del sistema. En tal caso es importante analizar y localizar las fuentes de los fallos haciendo un estudio particular del sistema.

   Para el funcionamiento del sistema ideal se utilizarán técnicas de simulación. Durante la simulación aplicaremos el proceso de evaluación y el análisis que describiremos en la sección 3.3. 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. Esto es 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.

3.3.1. 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, del tipo de sistema, de las condiciones de desarrollo y operación del sistema. 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, ó éstas 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 que se adecuan a su propio marco de producción (como la desarrollada por Motorola *Six Sigma* [94]).

   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 utilizadas 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 éste 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 con las métricas de software.

   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 la realizamos mediante el estudio de su distribución estadística. Los resultados de esta evaluación permiten analizar el conjunto de datos obtenidos mediante gráficas conocidas como histogramas (los cuáles pueden obtenerse mediante herramientas como *Arena*). Los histogramas permiten representar los valores de la métrica contra su frecuencia. Estos histogramas nos darán una aproximación al tipo de modelo que mejor se ajusta al comportamiento de los resultados obtenidos. El paso siguiente consiste en reemplazar el histograma por una ley de probabilidad que mejor represente éste comportamiento. Esta ley de probabilidad será el reflejo del comportamiento de la población de métricas estudiadas. La ley escogida nos podrá indicar si nuestra evaluación fue correctamente realizada.

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

   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 que métodos los son más adecuados.

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

   Una vez obtenidos los parámetros resultantes, éstos 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 estudiadas. Por lo tanto, en este proceso se realiza una comparación de los valores que se obtienen en el histograma contra los resultados que se obtienen en 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 (ó 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.

3.4. Caso de Estudio

En esta sección describiremos mediante un caso de estudio la metodología propuesta.

En la metodología se considera la evaluación del sistema de software bajo condiciones ideales y bajo condiciones reales.

En las condiciones ideales se toma en cuenta que no existe sobre carga de usuarios, no hay fallos en la red, y no hay fallos en la plataforma de cómputo. Solo existen fallos en la funcionalidad del sistema. Los resultados obtenidos por la fase de evalución del sistema bajo condiciones ideales, nos permite encontrar el comportamiento de referencia al cual el sistema debería de llegar en el caso óptimo.

Por otro lado, en la fase de evalución bajo condiciones reales, las anteriores restricciones se elimiman y se considera la operación del sistema bajo condiciones de operación real. En esta fase, se repiten las mismas pruebas realizadas bajo la fase de evaluacion ideal del sistema, pero bajo condiciones reales.

De la comparación del comportamiento del sistema ideal contra el sistema real, será posible predecir que tan lejos esta la operación del sistema del caso ideal, y que tantos fallos serán necesarios eliminar en la etapa de pruebas.

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 [104]*), un servidor Web (*Apache [105]*). El lenguaje utilizado para las interfaces de usuario fue (*PHP [106]*). El tiempo empleado para el desarrollo del sistema y de su documentación ha sido de 6 meses aproximadamente. El diagrama a bloques del sistema se puede observar en la Figura 4.3.

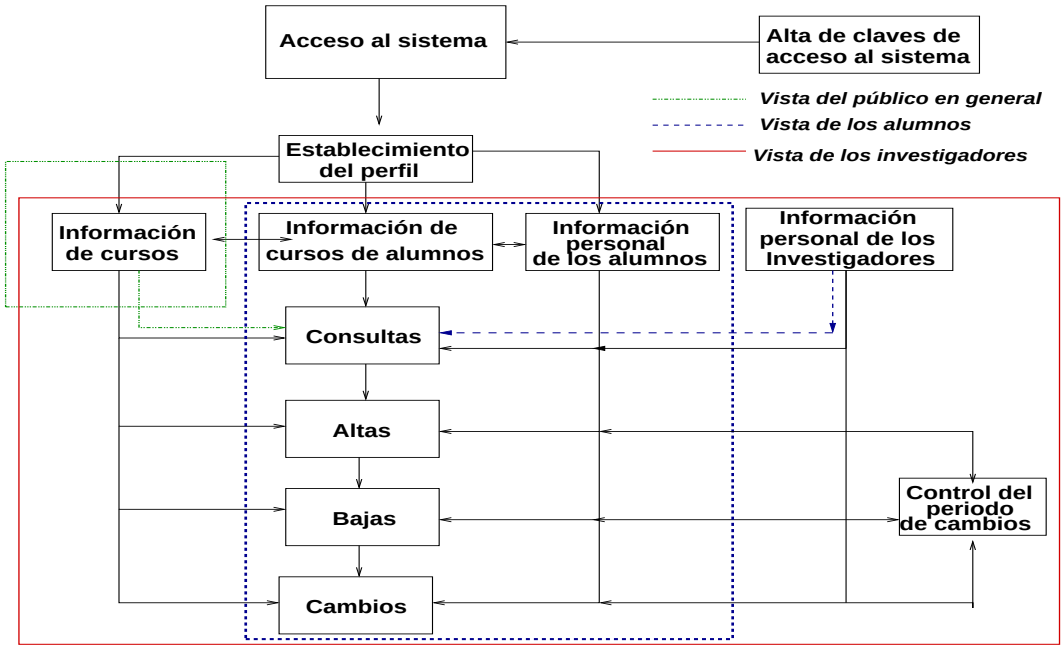

Figura 3.4: Módulos que conforman el Sistema de Información vía Internet (SIV)

3.4.1. Planteamiento del Problema

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 mediante claves 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 inscripción, consulta y administración de la información de cursos, calificaciones, profesores y alumnos.

3.4.2. Fase 1: Determinación del comportamiento ideal de la *Confiabilidad*

En esta fase, se evaluará y modelará el comportamiento del sistema bajo condiciones ideales, aplicando el proceso definido en la Sección 3.3.

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 confiabilidad, 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 determinado.

   El lenguaje de programación empleado para la simulación fue Java.

   En la simulación se emplea un modelo *productor−consumidor*, en donde la información se maneja mediante una cola de recursos compartidos. La sincronización del *productor-consumidor* se realizó mediante la sincronización de hilos. El productor tiene la función de generar los datos de salida para el consumidor, de acuerdo al funcionamiento del modelo lógico descrito en la Figura 3.5. La función del consumidor es actualizar y graficar el comportamiento de la simulación en cada unidad de tiempo, de acuerdo a los datos que el productor le envía a través de la cola de recursos compartidos.

   El diagrama a bloques de la operación del productor se muestra en la Figura 3.5. En este

[1] De acuerdo a la Notación de Kendal [82], 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).

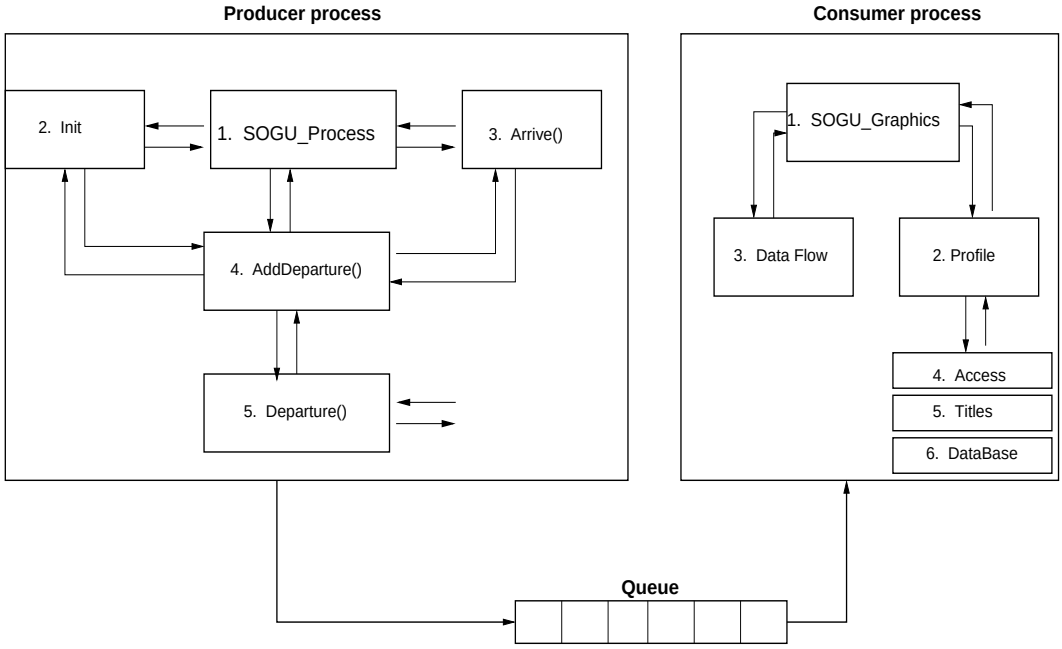

Figura 3.5: Diagrama a bloques del funcionamiento del Productor - Consumidor

diagrama se pueden apreciar 5 módulos. En el módulo principal, *Proceso del SIV*, se ejecuta el ciclo principal de la simulación. El segundo módulo *Rutina de Inicialización* tiene la función de inicializar las estructuras de datos, 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 una llamada a la función *departure()* y se verifica la generación de errores durante la operación. En el caso de que hayan existido errores, se aumenta el valor de la métrica de software empleada y se le programa al usuario su salida del sistema. El quinto módulo, *departure()*, se encarga de programar los tiempos de salida de cada usuario para la sección o parte del sistema que está utilizando.

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

   El atributo que nos interesa medir es la *Confiabilidad*. Como se menciona anteriormente, la *Confiabilidad* nos permite evaluar que tan libre de fallos se encuentra un sistema.

   Tipos de métricas.

En nuestro caso nos interesa medir el atributo de *Confiabilidad* en la operación del producto de software. Las métricas que son posibles utilizar para la Confiabilidad son [94]:

- **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 de 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*.

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 (determinación de las condiciones iniciales para el caso ideal).

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 500 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.**

      La simulación se realizó dentro de un lapso de tiempo de 100 unidades en el cual se contabilizaron el número de defectos que aparecieron en el sistema.

   c. **Identificar las mediciones anómalas.** Por el hecho de tratarse de una simulación que trabaja con condiciones ideales no se detectaron 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. Los elementos

D. Defec	Frec	x	Den. Prob.	Dist. Acum.
0	107	0.000	0.214	0.214
1	159	1.00	0.318	0.532
2	110	2.00	0.220	0.752
3	77	3.00	0.154	0.906
4	35	4.00	0.0700	0.976
5	6	5.00	0.0120	0.988
6	3	6.00	0.00600	0.994
7	2	7.00	0.00400	0.998
8	1	8.00	0.00200	1.00

Tabla 3.1: Resumen de los resultados obtenidos durante las evaluaciones del caso ideal

que integran el resumen son los siguientes: Densidad 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 3.6.

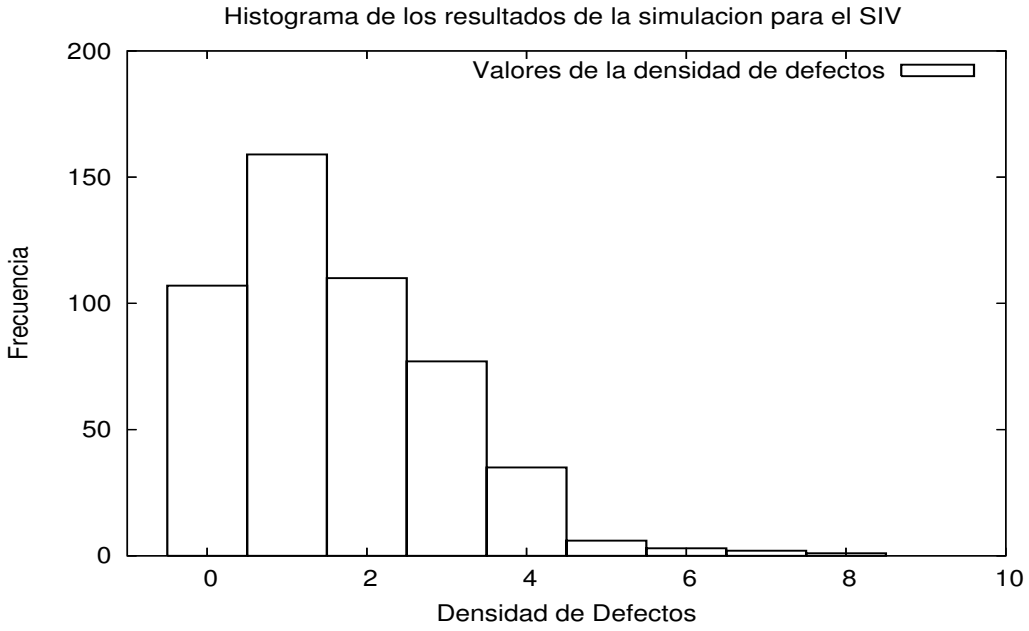

Figura 3.6: Histograma para 500 simulaciones.

Mediante el histograma apreciamos que la tendencia de los resultados se aproxima a una distribución (ó curva) de Weibull. La curva de Weibull es un metamodelo [82], 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 3.3, de la siguiente forma.

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

$$f(x) = \alpha\beta^{-\alpha}x^{\alpha-1}e^{-(x/\beta)^{\alpha}} \tag{3.7}$$

   Evaluar los parámetros contenidos en la ley escogida.

   Derivado del análisis, obtenemos las siguientes ecuaciones que permiten calcular los valores de α y β.

$$\frac{\sum_{i=1}^{n} X^{\alpha} lnX_i}{\sum_{i=1}^{n} X^{\alpha}} - \frac{1}{\alpha} = \frac{\sum_{i=1}^{n} lnX_i}{n} \tag{3.8}$$

$$\beta = \left(\frac{\sum_{i=1}^{n} X_i^{\alpha}}{n}\right)^{1/\alpha} \tag{3.9}$$

   Donde:

   - n = 500 (número de evaluaciones)
   - X_i hace referencia a cada uno de los valores de la densidad de defectos determinada en cada evaluación.

   Para estimar los parámetros de la función de densidad utilizamos la técnica de estimadores de máxima verosimilitud (MLEs). Estos parámetros se obtienen mediante la resolución de ecuaciones simultáneas que la técnica de MLEs nos provee para el modelo de Weibull. La resolución la determinamos mediante técnicas de métodos numéricos que se aplican con los datos que se han obtenido [82]. Para nuestro caso, estos datos son los valores obtenidos de la densidad de defectos en cada evaluación. La Ecuación 3.8 se resuelve mediante el método de *Newton-Raphson*, mientras que la Ecuación 3.9 se obtiene de manera directa conociendo el valor de α. Los valores obtenidos de los parámetros a partir de las Ecuaciones 3.8 y 3.9 son:

$$\alpha = 1{,}63 \text{ y } \beta = 2{,}4.$$

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

   Tomando en cuenta los valores de α y β antes descritos, la función de probabilidad se calcula de la siguiente forma.

$$f_i(x) = \begin{cases} 0{,}391x^{0{,}63}e^{-(x/2{,}4)^{1{,}63}} & \text{Si } x \geq 0 \\ 0 & \text{En otro caso} \end{cases}$$

   La gráfica del Comportamiento de la *confiabilidad* de acuerdo al modelo ideal f_i se presenta en la Figura 3.7.

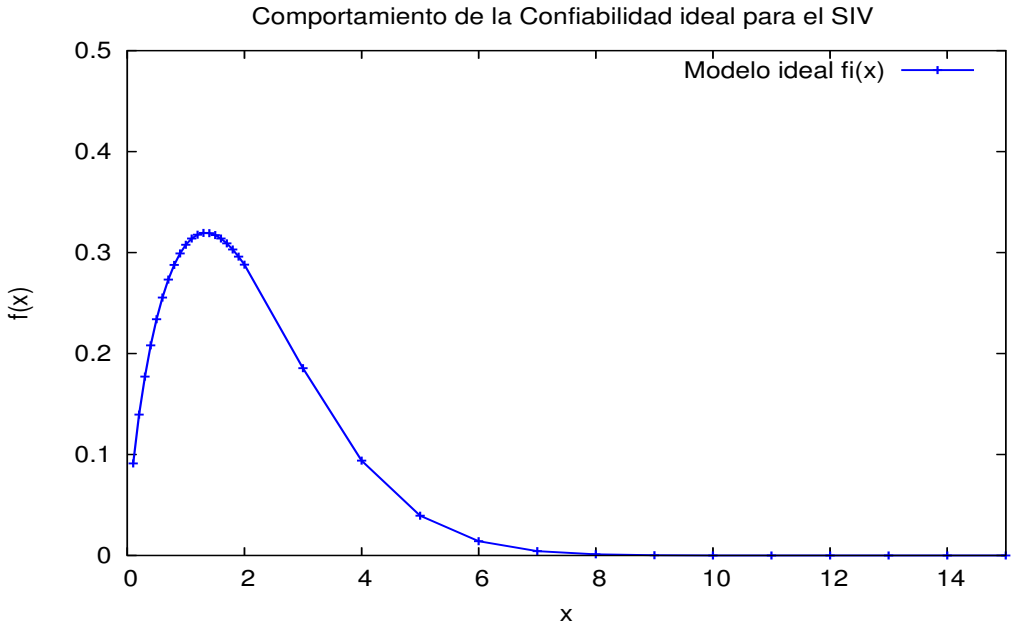

Figura 3.7: Comportamiento de la densidad de defectos (*confiabilidad*) en el SIV

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 permite validar 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 confiabilidad.**

De la información proporcionada por el Histograma de la Figura 3.6, es posible observar que la densidad de defectos presenta valores bajos con un alta frecuencia. Esta conclusión se obtiene del hecho de que en la mayoría de las evaluaciones se obtuvo un bajo número de errores.

Observando la información proporcionada por el modelo de la Figura 3.7, es posible observar que la tendencia de la curva que describe el comportamiento de la densidad de defectos, presenta sus máximos entre 0 y 2. La densidad de defectos que esperamos tener en la operación del caso real oscila entre 0 a 2 defectos. Con un rango de 0 errores como mínimo y 8 como máximo.

3.4.3. Fase 2:Evaluación de la confiabilidad en el sistema real

1. **Determinación de las condiciones iniciales para el caso real.** Las condiciones del proceso fueron las siguientes.

   a) El proceso de evaluación fue realizado por la herramienta que se describirá en la sección 3.5.

   b) Las evaluaciones se realizaron de manera concurrente.

   c) Las vistas que tomaron los evaluadores fueron: Investigadores, Coordinador, Alumnos y Público en General.

   d) Los módulos evaluados en el sistema fueron: Cursos (cursos x alumno), Alumnos, Investigadores.

   e) Las transacciones que realizó cada evaluador de acuerdo a su perfil fueron: consultas, altas, bajas y cambios.

   f) El arribo de los evaluadores estuvo determinado de acuerdo a una distribución exponencial.

   g) El tiempo de la evaluación general duró 100 unidades de tiempo.

   h) Los datos de la matriz se tomaron de manera aleatoria.

   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 *Confiabilidad* y la métrica para dicho atributo es la *densidad de defectos*.

   Condiciones de desarrollo. En el Sistema de Inscripciones por Internet seguimos el modelo de desarrollo en cascada. En el 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, el cuál fue el equivalente a un lapso de 100 minutos en un proceso manual. El número de experimentos fue de 500, los cuáles se llevaron a cabo en 3 horas. Si este proceso se hubiera realizado de forma manual se hubieran requerido de varios meses para este fin.

3. **Proceso de medición.**

   a. **Selección de los componentes a medir.** Durante la operación del sistema real se midieron los módulos: Cursos (cursos x alumno), Alumnos, Investigadores.

   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 por cada evaluador.

   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:

Periodo de Inscripciones:

- No se actualiza automáticamente la fecha.

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 mas de 4 cursos por alumno.

Consulta de cursos por alumno:

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

Alta de alumnos:

- 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 500 evaluaciones se presenta en la Tabla 4. Al igual que en el caso ideal en la tabla se registran los siguientes valores: Densidad de defectos (D.Defec), frecuencia de ocurrencia (Frec), valores de 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 3.8.

   Mediante el histograma apreciamos 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.

$$f(x) = \alpha\beta^{-\alpha}x^{\alpha-1}e^{-(x/\beta)^{\alpha}} \tag{3.10}$$

D. Defec	Frec	x	Den. Prob.	Dist. Acum.
0	4	0.000	0.00800	0.00800
1	2	1.00	0.00400	0.0120
2	16	2.00	0.0320	0.0440
3	6	3.00	0.0120	0.0560
4	43	4.00	0.0860	0.142
5	8	5.00	0.0160	0.158
6	51	6.00	0.102	0.260
7	17	7.00	0.0340	0.294
8	48	8.00	0.0960	0.390
9	14	9.00	0.0280	0.418
10	63	10.0	0.126	0.544
11	15	11.0	0.0300	0.574
12	51	12.0	0.102	0.676
13	8	13.0	0.0160	0.692
14	33	14.0	0.0660	0.758
15	12	15.0	0.0240	0.782
16	36	16.0	0.0720	0.854
17	6	17.0	0.0120	0.866
18	20	18.0	0.0400	0.906
19	6	19.0	0.0120	0.918
20	13	20.0	0.0260	0.944
21	6	21.0	0.0120	0.956
22	8	22.0	0.0160	0.972
23	3	23.0	0.00600	0.978
24	7	24.0	0.0140	0.992
25	2	25.0	0.00400	0.996
26	1	26.0	0.00200	0.998
32	1	32.0	0.00200	1.00

Tabla 3.2: Resumen de los resultados obtenidos durante las evaluaciones del caso real

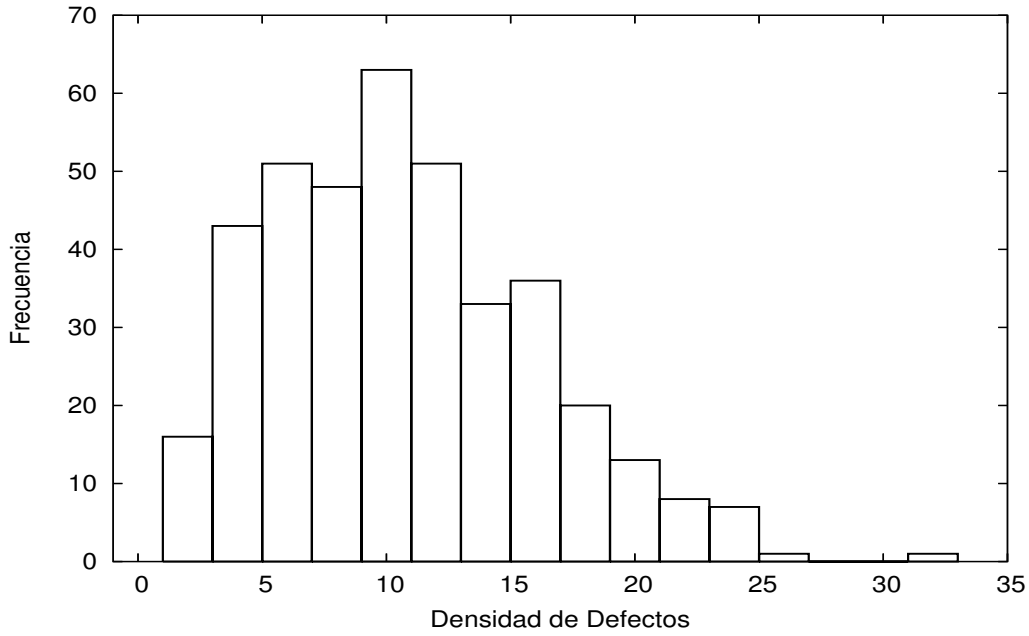

Figura 3.8: Histograma para 500 mediciones de errores

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

$$\alpha = 2{,}16 \text{ y } \beta = 12{,}8$$

Al igual que en caso ideal, los valores de α y β se resuelven de acuerdo a las Ecuaciones 3.8 y 3.9.

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 nos da la siguiente expresión.

$$f_r(x) = \begin{cases} 0{,}00876x^{1,16}e^{-(x/12,8)^{2,16}} & \text{Si } x \geq 0 \\ 0 & \text{En otro caso} \end{cases}$$

   La gráfica del comportamiento de la *confiabilidad* del modelo real f_r se presenta en la Figura 3.9.

7. **Comparación del modelo obtenido.**

   Es posible observar que el modelo obtenido para el caso real sigue una distribución de Weibull. El modelo obtenido sigue en contexto con el concepto de *Densidad de Defectos*, por lo que no se opone a representar el comportamiento de la operación real del sistema.

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

   Los parámetros α y β representan el grado de aproximación que existe entre los modelos ideal y real. Con los resultados obtenidos en el modelo real podemos establecer que la diferencia entre estos parámetros, determinan la diferencia entre los modelos, como se ve en la siguiente expresión:

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

   En la Figura 3.9 se observa la comparación de los modelos resultantes (ideal y real) de las evaluaciones del caso de estudio. Es posible observar que el comportamiento obtenido del modelo real está alejado del comportamiento del modelo ideal. Si tomamos los valores máximos de las curvas de los modelos veremos que para el modelo real estos valores oscilan entre 9 y 11 errores, los cuáles se encuentran muy alejados del modelo ideal que presenta sus valores máximos entre 0 y 2.

   También se puede observar de la Figura 3.9 que la curva que describe al modelo real esta desplazada hacia valores mayores de error cuyo rango es de 0 a 32; el cual es cuatro veces mayor que en del modelo ideal que es de 0 a 8. Esto nos expresa que el producto de software presenta un nivel de calidad muy por debajo de lo que esperamos (tomando como referencia el modelo ideal). Por lo tanto, concluimos que son necesarias mejoras en el producto de software. y para mejorarlo es necesario localizar y reparar los defectos encontrados.

3.5. Automatización del Proceso de Evaluación

Para realizar el proceso de evaluación en la fase 2 de la metodología se contemplarón las siguientes condiciones:

1. El arribo de los evaluadores bebería ser de acuerdo a una distribución exponencial.
2. El perfil de los evaluadores debería ser asignado de manera aleatoria
3. Los evaluadores deberían operar en forma concurrente, con una matríz de pruebas común.
4. La evaluación del sistema debería realizarse en función del cumplimiento de los requerimientos del sistema.

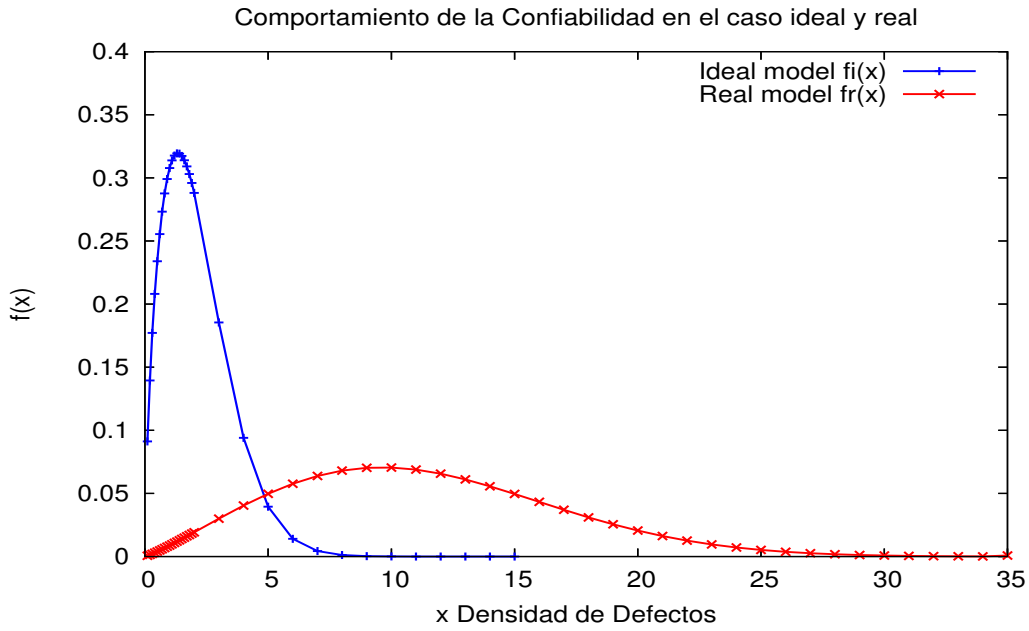

Figura 3.9: Gráfica del modelo ideal y real para el caso de estudio

Debido a que mediante un proceso manual realizado con un sólo evaluador no lograríamos cumplir con las especificaciones anteriores y la duración del tiempo empleado para este fín sería de varios días, decidimos desarrollar una herramienta para automatizar el proceso de evaluación.

Previamente realizamos un estudio acerca de las herramientas existentes en el ámbito academico e industral utilizadas para evaluar aplicaciones en Internet [6], [8]. De este estudio concluimos que no existia ninguna herramienta (gratuita) que nos permitiera evaluar la funcionalidad de un sistema de información en Internet considerando las condiciones enumeradas con anterioridad y que además se adaptara al contexto de nuestras aplicaciones. Por esta razón, nos surgió la necesidad de desarrollar una herramienta que funcionara de acuerdo al contexto deseado para el proceso de pruebas del caso de estudio.

Los requerimientos de esta herramienta de evaluación fuerón los siguientes:

1. La evaluación de los clientes debía de ser de manera concurrente.
2. Cada cliente debía actuar de acuerdo a las vistas del sistema.
3. La evaluación del sistema debía de realizarse por módulos.
4. Las transacciones que cada cliente realizaría serían de acuerdo a su perfil.
5. El arribo de los evaluadores debía ser de tipo exponencial.

6. Los evaluadores deberían de tener una matriz de pruebas común.

7. Después de la evaluación se necesita realizar un análisis de los errores obtenidos durante el proceso de pruebas.

8. El tiempo de la evaluación general debía debe ser expresado en unidades de tiempo.

El contexto de la herramienta se muestra en la Figura 3.10.

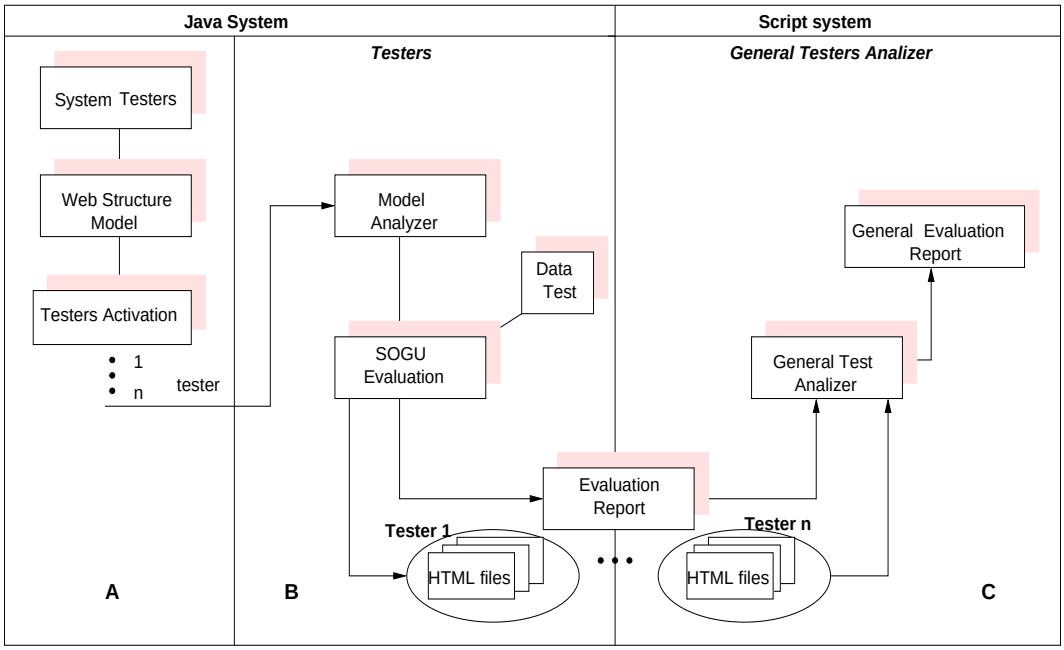

Figura 3.10: Contexto de evaluaciones para el SIV

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

1. Diseño. De acuerdo a los requerimientos para la herramienta y de acuerdo a las necesidades del sistema determinamos que éste debería de ser un autómata que operara de acuerdo a las acciones que pueden realizar las vistas en el sistema. Así la matriz de estados en sus columnas tiene la relación de los secciones del sistema y en sus filas 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 módulo examinado.

2. Codificación. Nuestro sistema de pruebas esta conformado por 3 subsistemas, los cuales representamos en la Figura 3.10. Dos de los subsistemas están implementados en lenguaje Java y el tercero se desarrolló mediante Scripts. Todo el sistema de evaluación se ejecuta bajo la plataforma Linux.

El primer subsistema determina la estructura del sitio Web que vamos a evaluar. También tiene la función de despertar a los threads de los evaluadores virtuales y activa los arribos de estos evaluadores tomando en cuenta una distribución exponencial.

El segundo subsistema se encarga de analizar y establecer la navegación del evaluador en el sitio de acuerdo a su perfil y a la estructura del sitio Web. Una vez determinada esta estructura de navegación el evaluador procede a realizar las pruebas. Los datos de las transacciones durante las evaluaciones los toma de manera aleatoria de una matriz de pruebas (data test).

El tercer subsistema realiza el proceso de análisis de los resultados y analiza el conjunto de bitácoras de respuesta de las transacciones que realizó cada uno de los evaluadores. Con dicho análisis se obtienen los resultados del reporte general de pruebas determinando la *Densidad de Defectos* durante la evaluación.

El tiempo de ejecución del sistema es el mismo que el de la simulación.

Descripción del diagrama de clases:

- La clase *Test_SIV* tiene la función de despertar a los hilos(threads) que controlan a los evaluadores virtuales mediante los métodos:
- *Init()* inicializa 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 con una distribución exponencial y llama al método *Acceso*.
- *Acceso(int skill, int test_time)*verifica si el usuario es válido dentro del sistema.
- *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* realiza este proceso
- *Request* opera con la clase URL. Esta clase tiene la función de establecer las peticiones de los clientes hacia el servidor. En la petición establecemos el protocolo (http), la dirección (xxx.xxx.xxx.xxx) y el *path* de la aplicación. El signo de interrogación (?) separa la página que se desea consultar, y el signo (&) las variables y sus valores. Las respuestas del servidor se descargan en archivos html, que posteriormente se analizarán

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

- *Data_test*. Va a proveer los datos de la matriz de pruebas de manera aleatoria.

El proceso de análisis se realizó mediante *scripts* (programación del shell). En general tienen la función de revisar el conjunto de archivos de respuesta de las transacciones que realiza cada uno de los evaluadores durante el proceso de pruebas. Con este análisis se obtiene el número de errores y con ello la *Densidad de Defectos*.
3. Implementación El sistema está operando en una plataforma Linux. La herramienta fué desarrollada con el Lenguaje Java.

3.6. 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 *confiabilidad*.

La metodología se implemento en dos fases, las cuales permitieron evaluar un sistema bajo condiciones de operación real e ideal.

En la metodología se utilizaron técnicas de modeliación estadística para estimar la confiabilidad del sistema bajo estas dos condiciones de operacion.

De esta comparación fue posible deducir que la confiabilidad del sistema bajo condiciones reales estuvo alejada de la confiabilidad del sistema bajo condiciones ideales. El resultado de esta metodología es este análisis que consiste en determinar si los niveles de confiabilidad estan cerca o lejos de los nivels óptimos.

Nuestro principal argumento a favor de la metodología es de que es clara, escalable y facil de adaptar a caulquier sistema, con entradas faciles de obtener y con resultados que resultan también faciles de interpretar y aplicar a la fase de pruebas.

Con el fin de llevar a la práctica esta metodología desarrollamos una herramienta muy util e importante para reducir tiempos de evaluación y obtención de datos y resultados de la evaluación. Sin esta herramienta, el tiempo de evaluación nos hubiera llevado varios meses, haciendo en consecuencia impractica la aplicación de la metodología.

Capítulo 4

Escenarios de Riesgo

4.1. Resumen

El objetivo de los estudios de la Confiabilidad es localizar y corregir problemas potenciales en el sistema de software. En la realización de estos estudios se han propuesto diversos modelos predictivos de los cuales pocos de ellos se han llevado a la práctica, como es el caso del modelo estocástico que propone Whittaker [99], [101].

Nuestra propuesta se enfoca al desarrollo de un modelo predictivo para el proceso de evaluación. Este modelo es una guía para planear las pruebas y ampliar la cobertura en los componentes donde se tiene una mayor probabilidad de fallos. El modelo propuesto además de estimar la cobertura del sistema durante el proceso de evaluación, nos proporcionará información acerca de la probabilidad de fallos para cada componente del sistema, las rutas (paths) de evaluación más sensibles a fallos y en general información importante para la fase de evaluación y corrección de errores. En nuestro caso, este estudio lo hemos implementado en la práctica, para lo cual hemos desarrollado una herramienta que obtiene el modelo de evaluación de manera automática y lo aplica durante la fase de pruebas. La cobertura de las evaluaciones y las rutas de evaluación se basan en el modelo desarrollado. Los errores resultantes de las pruebas son analizados por nuestra herramienta la cual genera bitácoras de información con las que podemos obtener la secuencias de evaluación que presentan los fallos, el componente anómalo, el error en cuestión y las métricas de software utilizadas (densidad de defectos

(DD) y media de ocurrencia de fallos (MTTF) a nivel global y en cada uno de los componentes del sistema).

4.2. Introducción

En el desarrollo de los sistemas de software es muy importante localizar e identificar los fallos. La etapa de pruebas en este caso tiene un papel determinante. El desarrollo de las pruebas ocupa del 40 % al 60 % del desarrollo del sistema. Sin embargo, a pesar de este esfuerzo no existe una garantía de que el sistema de software quede libre de fallos críticos. Por citar algunos casos importantes tenemos el caso del Ariane vuelo 501[79], Therac-25[80] y el caso de las caídas de portales de Internet como ebay.com, bancos importantes en el mundo y otros [81]. En este sentido se han desarrollado trabajos que estiman modelos predictivos de los cuales se obtiene información que sirve como guía para optimizar la etapa de pruebas. Esta situación permite optimizar el tiempo, el esfuerzo y los recursos que son empleados en la etapa de pruebas.

Los modelos predictivos son usados para obtener el nivel de Confiabilidad de un producto de software. Existen modelos que estiman este nivel de Confiabilidad en cada una de las etapas del proceso de desarrollo. Los modelos que son usados con mayor frecuencia son los que estiman la cobertura de las evaluaciones durante la fase de pruebas. La efectividad del modelo es directamente proporcional al número de defectos localizados bajo el mismo marco de tiempo y con los mismos recursos.

Los modelos predictivos de confiabilidad pueden ser clasificados en dos categorías:*Modelos Predictivos Dinámicos y Modelos Predictivos Estáticos*. Un *Modelo Dinámico* usualmente se determina utilizando distribuciones estadísticas, para lo cual, se toma el patrón actualizado de defectos que resulta de la etapa de pruebas. Estos modelos nos sirven de patrón para analizar el comportamiento de la Confiabilidad en el producto final en operación. Este enfoque toma al sistema de software como una sola unidad. Un *Modelo Estático* es un estudio del sistema visto como un conjunto de unidades que operan de acuerdo a cierta funcionalidad y que es sensible a fallos. Para analizar la posibilidad de la incidencia de los fallos se toman en cuenta atributos del sistema que permiten estimar la interacción de las unidades (módulos ó componentes). Esta interacción en un contexto de evaluación genera un ambiente propicio para la incidencia de los fallos en el sistema.

Los modelos estáticos ayudan al Ingenieros de Software a mejorar la calidad de su diseño e implementación. En el desarrollo de los modelos estáticos, la mayoría de los estudios llevados a

cabo toman en cuenta la funcionalidad del sistema [52, 86, 91]. Estos trabajos simulan ambientes de fallos para identificar los componentes críticos, lo cual excluye propiedades de operación que pueden afectar las predicciones de una manera severa. Por otro lado, pocos estudios toman en cuenta la complejidad del sistema [Sanyal1997]. En estos estudios se toma en cuenta la propagación de los fallos tomando en cuenta la complexidad del código del sistema. Sin embargo la mayoría de estos estudios no han sido hasta la fecha llevados a la práctica ademas de que se conoce que la costo de la obtención de las entradas de estos modelos en muchos de los casos es costosa y difícil de obtener.

El trabajo propuesto nos introduce al desarrollo de un modelo para la Confiabilidad y su análisis bajo un contexto operativo. Este trabajo lo basamos en una técnica que denominamos *Escenarios de Riesgos de Confiabilidad ERC*. Esta técnica asocia la funcionalidad y la complejidad del sistema mediante un estudio de la interacción de los componentes del sistema de software. En la primera fase de esta técnica construimos un grafo que denominamos *Grafo de Dependencias Funcionales (GDF)*. A este grafo, le asignamos pesos de acuerdo a la complejidad ciclomática de los componentes que intervienen en cada relación funcional. En una segunda fase, se toma el *GDF* y se le aplica el proceso de los *ERC* para determinar las trayectorias críticas que tienen probabilidades de presentar un mayor número de incidencia de fallos. Esta técnica permite identificar las rutas de ejecución (paths) y los componentes donde los fallos tienen una mayor probabilidad de ocurrencia.

El objetivo principal de nuestra propuesta es proporcionar los elementos necesarios para que las pruebas sean diseñadas para encontrar el mayor numero de fallos posibles con el menor esfuerzo.

4.3. Trabajo relacionado.

Distintos modelos predictivos y técnicas han sido propuestas para estimar los fallos de un sistema de software. Una de las técnicas mas utilizadas es la modelación de procesos estocásticos con Cadenas de Markov. Sin embargo este enfoque tiene los siguientes problemas:

- Dependiendo de la funcionalidad y del entorno del sistema, el espacio de estados puede crecer demasiado, lo que dificulta su estudio y en un extremo puede volverlo intratable.
- Al ser muy grande el espacio de estados, el costo computacional de los algoritmos para resolver los sistemas de ecuaciones lineales puede ser muy costoso.
- La confiabilidad de cada componente y su interrelación no puede ser estudiada de forma conjunta.

- La combinación de diversos enfoques o escenarios no pueden ser tratados en un mismo estudio.

La simulación orientada a eventos discretos es una alternativa para resolver algunos de estos problemas. En este caso, también los parámetros de la infraestructura pueden ser tomados en cuenta y es posible capturar el detalle del comportamiento del sistema. Al respecto Gokhale y Trivedi [86] nos muestran la flexibilidad para analizar aplicaciones basadas en componentes mediante simulación basada en eventos discretos. Su enfoque se basa en la generación aleatoria de fallos en los componentes del sistema. Ellos proponen un proceso sistemático el cual retorna el tiempo de arribo de fallos en un componente dado. La confiabilidad del sistema se estima obteniendo el número total de fallos a través de la simulación. Este enfoque asume la existencia de un grafo del flujo de control de un programa. En la simulación, el arribo de los fallos se genera de manera aleatoria y determina tazas de reparación. Con estos elementos se generan fallos para la aplicación en tiempo de ejecución. Ellos asumen que el sistema de software está ya terminado y que está conformado de m componentes que tienen una interrelación. La ejecución del sistema comienza con el componente 1 y finaliza en el componente m. La relación entre los componentes es especificada mediante las probabilidades de transición entre los componentes, denotadas por ω_{ij}. Estas ω_{ij} representan las probabilidades de que el componente j sea ejecutado después de la ejecución del componente i.

En este proceso, cada componente es descrito mediante un nivel de confiabilidad que el modelo proporciona. En su estudio, Gokhale y Trivedi [86] utilizan una aplicación de software la cual esta formada por 10 componentes los cuales comienzan su ejecución con el componente 1 y la terminan con el componente 10.

El grafo del flujo de control está representado mediante la Figura 4.2 y las probabilidades de transición entre los componentes son mostradas en la Tabla 4.1.

Inicialmente asumen que el ambiente de fallos de los componentes es especificado por sus confiabilidades R_n. El objetivo del experimento es simular la distribución de fallos de la aplicación, dadas las tazas de fallos de los componentes y las probabilidades de transición en un escenario típico de pruebas, donde la cobertura y el número de evaluaciones están bien determinadas para una duración especifica de t unidades de tiempo.

Los resultados obtenidos son mostrados en la gráfica siguiente:

Una de las principales desventajas de este modelo es que le asigna un mismo tiempo de operación para cada componente e ignora los fallos que ocurren en las transiciones e interfaces de los componentes. Si bien esta propuesta es adecuada para estimar parámetros de confiabilidad, no es

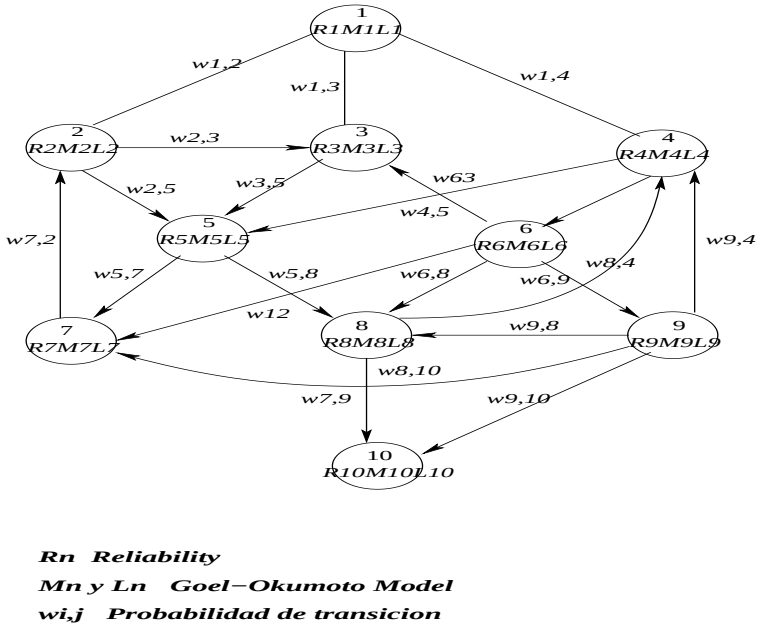

Figura 4.1: Grafo del flujo de control de una aplicación terminal

lo suficientemente clara como para llevarla a la práctica, debido a que carece de elementos (como es el caso de las rutas de ejecución) para la etapa de pruebas.

Sanyal [95] nos introduce al análisis de la propagación de fallos mediante el estudio de las dependencias del sistema utilizando la técnica de grafos [95]. El enfoque se basa en la Ingeniería de Reversa, en donde a partir del código fuente se generan grafos dirigidos que representan las dependencias del software. Las dependencias se obtienen mediante la unión de la probabilidad de dependencias de los eventos con la propagación de los fallos. Este enfoque toma en cuenta la arquitectura del sistema evaluado, aunque no estudia la validez de estas relaciones desde un punto

$\omega_{1,2} = 0{,}60$	$\omega_{1,3} = 0{,}20$	$\omega_{1,4} = 0{,}20$	
$\omega_{2,7} = 0{,}70$	$\omega_{2,5} = 0{,}30$		
$\omega_{3,5} = 1{,}00$			
$\omega_{4,5} = 0{,}40$	$\omega_{4,6} = 0{,}30$		
$\omega_{5,7} = 0{,}40$	$\omega_{5,8} = 0{,}60$		
$\omega_{6,3} = 0{,}30$	$\omega_{6,7} = 0{,}30$	$\omega_{6,8} = 0{,}10$	$\omega_{6,9} = 0{,}30$
$\omega_{7,2} = 0{,}50$	$\omega_{7,9} = 0{,}50$		
$\omega_{8,4} = 0{,}25$	$\omega_{8,10} = 0{,}75$		
$\omega_{9,8} = 0{,}10$	$\omega_{9,10} = 0{,}90$		

Tabla 4.1: Tabla de probabilidades de transición de los componentes de la aplicación de software

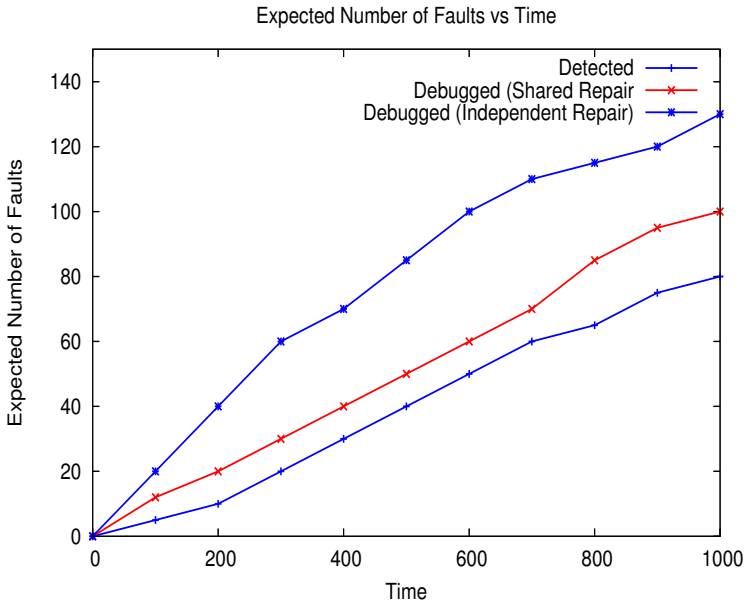

Figura 4.2: Reparaciones del sistema: compartida e independiente

de vista funcional del sistema.

También existen trabajos que se enfocan al estudio de la funcionalidad. Cortesella y Postojanova [52] proponen una metodología para la valoración de un factor de riesgo basado en el funcionamiento que origina el tratamiento inadecuado de los requerimientos funcionales. La metodología elabora diagramas UML para estimar la probabilidad de la falla del funcionamiento y la combina con la estimación de la severidad de la falla que se obtiene mediante el análisis funcional, utilizando Cadenas de Markov.

James A. Whittaker y Michael G. Thomason, proponen un modelo basado en cadenas de Markov [99]. Este estudio genera un modelo estocástico que determina la cobertura de evaluación del sistema en cuestión. El modelo que ofrece esta propuesta se ha llevado a la práctica en sistemas de software de industrias como France Telecom R&D, Israel Aircraft Industries, etc.

La importancia del desarrollo de un marco de evaluación capaz de llevar a la práctica los resultados del modelo dió origen a la herramienta MaTeLo (Markov Chain Test Logic) la cual fue construida en un proyecto europeo entre los años 2001-2003, en colaboración con cuatro empresas desarrolladoras de software y dos universidades europeas [101]. MaTeLo basa el proceso de evaluación en la propuesta de Whittaker y Thomason con la cual se determina la cobertura de evaluación del sistema y las pruebas se hacen mediante la técnica de pruebas de caja negra (la métrica que

reporta es la MTTF).

Se han desarrollado distintos enfoques utilizando Cadenas de Markov para desarrollar modelos estocásticos de confiabilidad. Los Modelos Estocásticos utilizan como entradas los estados del sistema y las transiciones de un estado a otro. Las transiciones reflejan los fallos de los componentes. Los modelos estocásticos tienen limitaciones en su aplicación para sistemas muy grandes ó complejos debido a que el número de estados puede llegar a ser muy grande y computacionalmente muy costoso.

El modelo que desarrollamos en esta tesis se basa en la representación del sistema como una composición de componentes de software y no como estados del sistema. Mientras que la modelación de estados se enfoca en el comportamiento del sistema, nosotros nos enfocamos en capturar la interacción entre los componentes. Esta interacción la expresamos mediante la arquitectura del sistema de software.

Goseva-Popstojanova y Trivedi [88] presentan una clasificación de enfoques basados en la arquitectura de software para estimar la confiabilidad de sistemas basados en componentes. Ellos identifican tres clases basadas en los métodos usados para describir la arquitectura agregando un ambiente de fallos de los componentes y sus enlaces.

- Modelos basado en estados, en donde la arquitectura del software y el ambiente de fallos son representados mediante cadenas de Markov, o procesos de semi-Markov.
- Modelos basado en trayectorias, donde la Confiabilidad es estimada mediante un conjunto de escenarios.
- Modelos aditivos, los cuales se enfocan en la estimación de la intensidad del tiempo de dependencia de los fallos usando componentes con datos para la introducción de fallos.

Otro trabajo importante es el de Yacoub [98], el cual realiza un estudio para determinar escenarios de confiabilidad en sistemas basados en componentes utilizando el enfoque basado en trayectorias. En este estudio el autor asigna un nivel de confiabilidad a cada componente y retoma la idea de Gokhale y Trivedi. Mediante la técnica de simulación establece un estudio donde se estima el comportamiento y la interacción entre los componentes de sistema de software. En la simulación también se integran algunos factores del contexto de operación para estimar su influencia en la confiabilidad. Este trabajo es un aporte muy importante debido a que estima el comportamiento de la confiabilidad mediante la integración de los componentes y toma en cuenta los fallos que se

pueden generar en la interacción de los componentes. Sin embargo todo el trabajo se limita a la simulación de este para su estudio, sin que se lleve el estudio a un contexto de operación.

4.4. Modelo de Evaluación de la Confiabilidad mediante Escenarios de Riesgo

En este capítulo presentamos nuestra propuesta para la evaluación de la confiabilidad mediante los escenarios de riesgo.

En nuestra propuesta tomamos en cuenta el enfoque basado en trayectorias donde estimamos rutas de evaluación que presentan un mayor riesgo de generar fallos en la fase de pruebas. Estas trayectorias se determinan mediante un GDF donde se integran la funcionalidad y la complejidad del sistema. La funcionalidad se determina mediante un análisis de los requerimientos funcionales. Por otro lado, la estimación de la prioridad de las dependencias funcionales se obtiene mediante una simulación basada en eventos discretos, en donde la probabilidad de ocurrencia en cada operación es calculada en un ambiente en donde el arribo de los fallos en cada componente es aleatorio y sigue una distribución de Weibull. Este proceso sirve para establecer el GDF.

La complejidad se obtiene mediante la evaluación de los componentes en la arquitectura del sistema. Al utilizar la complejidad ciclomática asociamos características de la arquitectura al peso de cada arista en las relaciones del *GDF*. Así el modelo del grafo queda listo para su análisis y la generación de los escenarios de riesgos *ER*.

Los escenarios de riesgo permiten obtener información acerca de las rutas funcionales del sistema en donde hay una mayor probabilidad de localizar fallos.

4.4.1. Grafo de Dependencia Funcionales (GDF)

El modelo que estamos proponiendo se basa en el desarrollo y análisis de un grafo al que denominamos *Grafo de Dependencias funcionales (GDF)*. La construcción y análisis de este grafo es el camino que nos permite llegar a establecer las rutas críticas del sistema y al que denominamos *Escenarios de Riesgo*. En estos *Escenarios de Riesgo* se representa la ruta funcional en donde esperamos una mayor incidencia de fallos. Con el análisis de este grafo podemos determinar la probabilidad de fallo en cada ruta y de cada componente, así como información que es importante en el proceso de corrección de fallos.

Generación de las Relaciones Funcionales

Para generar el GDF es necesario conocer y entender los requerimientos funcionales del sistema, para poder establecer las relaciones funcionales. Cuando se han identificado las relaciones funcionales, es necesario enfocar este conocimiento acorde al enfoque de Ingeniería de la Confiabilidad (SRE). Este enfoque permite estimar la prioridad y la necesidad de estas relaciones funcionales en los sistemas examinando los valores de las *tazas de ocurrencia*. Las tazas de ocurrencia se determinan estimando el tiempo en que los perfiles operacionales utilizan cada una de las secciones del sistema y de sus operaciones. Los perfiles operacionales representan las operaciones y el conjunto de servicios de un sistema de software particular en función de un usuario especificado.

Algunos autores proponen asentar estos valores en base a las versiones anteriores del sistema ó de sistemas parecidos, sin embargo cada sistema y versión difiere en su enfoque operacional, ya que dependen de sus requerimientos. En consecuencia el ambiente en que se generan los fallos varía entre las aplicaciones ya que estos pueden presentar distintas funcionalidades del sitio y distintos contextos de operación.

En la estimación de las tazas de ocurrencia, en nuestro modelo proponemos utilizar la simulación basada en eventos discretos, a partir de la cual sea posible establecer la importancia de estas relaciones funcionales en un ambiente de fallos.

Generación del GDF

De acuerdo a los conceptos definidos en la sección anterior definimos al Grafo de Dependencias Funcionales de la siguiente forma.

Definición. Sea un grafo G definido como un conjunto no vacío de vértices $V = v1, v2, ...vn$ y un conjunto de aristas $A = a1, a2, ..., an$. Los subconjuntos de vértices están formados de al menos dos nodos enlazados mediante una arista que tiene un peso ω. Un vértice v representa un subconjunto a, b con $a, b \in V, a \neq b$ y una arista A con un peso ω.

Donde:

1. Los subconjuntos de vértices $\{a, b\}$ representan las relaciones funcionales: *accesos, perfiles operacionales y operaciones (transacciones) del sistema.*

    - El primer vértice v_s representa el acceso del sistema.

- Los vértices que derivan de v_s son denominados de la siguiente forma $v_{op1}, v_{op2}, .., v_{opn}$ y representan los perfiles operacionales del sistema.
- Los vértices que derivan de cada perfil representan $v_{t1}, v_{t2}, .., v_{tn}$ las operaciones ó transacciones asociadas a los perfiles.
- Los vértices que se relacionan con las operaciones que derivan del diseño del sistema son representados con la nomenclatura $v_{dt1}, v_{dt2}, ..., v_{dtn}$.
- Las aristas que enlazan los vértices nos indican la relación de las dependencias funcionales del sistema.

2. Cada relación funcionales del sistema sera cuantificada con la probabilidad de ocurrencia.
3. Cada arista tiene un peso ω que la pondera.
4. El peso ω_a de cada una de las aristas se asigna de acuerdo al valor de la Complejidad Ciclomática del software del (los) módulo(s) implícito(s) en la relación funcional $\{a, b\}$.

A partir de este punto es necesario estimar la complejidad ciclomática de cada uno de los componentes del sistema que intervienen en las relaciones funcionales estimadas, para lo cual se necesita seleccionar una métrica de software. Una de las métricas más utilizadas es la *Complejidad Ciclomática de McCabe*. La complejidad ciclomática de McCabe, *establece el mínimo de rutas que es necesario para evaluar una sección específica del sistema*. Esta métrica ha demostrado dar resultados objetivos cuando se desea evaluar cualquier algoritmo [71].

Con los elementos definidos anteriormente es posible desarrollar y analizar el GDF.

4.4.2. Proceso para diseñar Escenarios de Riesgo

Nuestra técnica tiene el objetivo de obtener información acerca de las rutas funcionales del sistema donde hay una mayor probabilidad de localizar fallos.

Para obtener los escenarios de riesgo es necesario utilizar como entrada la información proporcionada por el GDF.

Definición. *Un Escenario de Riesgo es un conjunto de relaciones funcionales que representan la trayectoria de navegación más compleja entre las relaciones funcionales de origen y las operaciones funcionales del sistema.*

Definición. *La probabilidad de fallo de un escenario de riesgo es igual al promedio del peso del escenario* P_{ak} *desde* $1, .., m$ *elementos y el peso total de los escenarios* $\sum_{k=1}^{k=n} P_{ak}$ *desde* $(1, .., Nescenarios)$ *en relación con la probabilidad de ocurrencia que ha sido calculada en un ambiente de fallos:*

$$PF_{ER} = \frac{1}{2} * \frac{\sum_{k=1}^{k=m} P_{ak} + P_ocurrence}{\sum_{k=1}^{k=n} P_{ak}} \tag{4.1}$$

Donde:

$PF_{ER} =$ *Probabilidad de Fallo del Escenario de Riesgo*

$P_{ak} =$ *Peso de la arista*

$P_{ocurrence} =$ *Peso de ocurrencia*

Las actividades para el proceso de desarrollo de los Escenarios de Riesgo son las siguientes:

1. *Aplicación del algoritmo de caminos mínimos.* El primer paso para obtener los escenarios de riesgo es la aplicación del algoritmo de caminos mínimos sobre el GDF. Un algoritmo eficiente para estimar la ruta más corta, es el algoritmo de Dijkstra. Así teniendo un grafo dirigido $GDF = (V, E)$, con un peso asignado $\omega : E \leftarrow R+$ y los vértices bien establecidos s en V, es posible encontrar la ruta mas corta desde s para cada vértice v en V.

2. *Estimación de la probabilidad de Fallo del Escenario de Riesgo* PF_{ER}. La PF_{ER} es un valor numérico que representa las probabilidades de la existencia de fallos en el escenario que se estudia. La probabilidad de fallo de los Escenarios de Riesgo se determina mediante el promedio del peso de escenario y la probabilidad de ocurrencia del escenario en un ambiente de fallos.

3. *Determinación de los elementos en los casos de prueba críticos.* Tomando los escenarios de riesgo con probabilidades de fallo mayores se determinan los perfiles operacionales, las operaciones y los módulos de software implícitos en estos.

4. *Estimación de la probabilidad de Fallo de los Componentes.* Cuando se ha determinado la probabilidad de Fallo para cada escenario, se calcula la probabilidad de fallo de cada uno de los componentes que intervienen en los escenarios mediante el siguiente cálculo:

   El cálculo para estimar la probabilidad de fallo del componente c_i de lleva a cabo de la siguiente forma:

$$PFC_i = \sum_{k=1}^{k=n} PF_{ER} * \frac{N}{N_{ET}} \tag{4.2}$$

Donde:

PFC_i = Probabilidad de Fallo del Componente i.

PF_{ER} = Probabilidad de Fallo del Escenario de Riesgo ER.

N = El número total de Escenarios de Riesgo donde intervieve el componente.

N_{ET} = El número total de Escenarios de Riesgo.

5. *Análisis de los resultados.* Cuando se han determinado los escenarios de riesgo es posible hacer la validación y la verificación de la funcionalidad del sistema en relación a la lógica de los valores estimados de acuerdo a la complejidad. Debido a que los resultados de riesgo proporcionan información sobre las rutas de ejecución con mayor probabilidad de fallos, esta información es utilizada para preparar los casos de prueba con lo cual podremos localizar un mayor numero de fallos durante la fase de pruebas.

6. *Preparación de los casos de prueba.* Al tener las rutas de los escenarios de riesgos bien establecidas y las probabilidades de fallo determinadas, podemos planear la fase de pruebas haciendo un especial énfasis en los escenarios de riesgos que tienen los componentes con mayor probabilidad de fallo. Asi podemos ampliar el número de pruebas y los valores de entrada en donde hay una mayor probabilidad de encontrar una alta incidencia de fallos.

7. *Resultados y análisis.* Las predicciones que el modelo estima, acerca de los componentes de software que tienen una mayor Probabilidad de Fallo y la rutas funcionales que tienen una mayor Probabilidad de Fallo son las estradas para estimar la cobertura de evaluación durante la fase de Pruebas.

4.5. Caso de Estudio

En esta sección describiremos mediante el caso de estudio la aplicación de la metodología propuesta. 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 [104]), un servidor Web (Apache [105]). El lenguaje utilizado para las interfaces de usuario fue (PHP [106]). El número de líneas de código utilizadas en la implementación del sistema fue de 1200. 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 4.3.

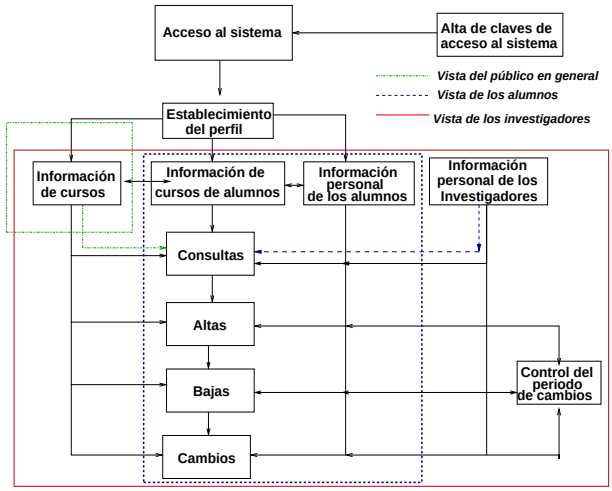

Figura 4.3: Módulos que conforman el Sistema de Información vía Internet (SIV)

4.5.1. Descripción del sistema

Se trata de un Sistema de Inscripciones vía Internet para una Universidad, en donde existen tres vistas principales: (a). Investigadores/Profesores (1 Coordinador), (b) Alumnos, (c) Publico en general. El acceso al sistema es con 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.

4.5.2. Determinación del Grafo de Dependencias Funcionales (GDF).

De acuerdo a la sección 4.4.1, para el desarrollo de nuestro modelo, el primer paso es la determinación del GDF. Esto es estudiando la funcionalidad del sistema de acuerdo a la especificación sus requerimientos. El documento de especificación de requerimientos para el SIV se encuentra el apéndice A.1.

De acuerdo al requerimiento **RB0**, debe existir una entidad funcional que permita el acceso al sistema sólo mediante clave de usuario y password. En este caso los nodos $v_{s0} y v_{s1}$ representan esta entidad, y esta relación la podemos encontrar en la Tabla 4.2. Los nodos $v_{op3}, v_{op4}, v_{op5}$ y v_{op6}

establecen los perfiles operacionales del sistema de acuerdo a los requerimientos **RB9**. Los nodos v_{o7}, v_{o9} representan operaciones asociadas a las consultas (requerimientos **RB4, RB8, RB11, RB13, RB15 y RB17**).

Los nodos v_{d8} representan una relación funcional que deriva del diseño y que apoya la operación v_{o9} asociada a la inscripción de los alumnos a los cursos (requerimientos **RB1,RB4 y RB5**). En este caso la inscripción está restringida al número de materias que el alumno tiene inscritas, al cupo del curso y al marco de tiempo que tiene como margen para este proceso. De ahí la necesidad de integrar la entidad v_{d8} que apoya esta parte. El nodo v_{d10} apoya el cambio de curso, es decir la operación de reinscripción que representa el nodo v_{o11} de acuerdo al requerimiento **RB16**.

De esta manera hemos generado las entidades funcionales. La relación de las entidades funcionales y los vértices aparece en la Tabla 4.2. En la Tabla 4.2 tenemos en la primera columna la nomenglatura asignada a los vértices y en la segunda columna tenemos la entidad funcional que representa. El GDF resultante lo podemos apreciar en la Figura 4.4.

4.5.3. Determinación de la probabilidad de ocurrencia.

El siguiente paso en el desarrollo de nuestro GDF es la estimación de la probabilidad de ocurrencia de cada una de las operaciones estimadas de acuerdo a los requerimientos se realizó la simulación de la funcionalidad del sistema de software.

Las condiciones para la simulación son las siguientes:

- El tiempo de arribo para los usuarios al sistema esta determinado por una distribución exponencial $\mu = 5$.
- Se utilizó un servidor de colas del tipo MM1 en el pool del servidor Web.
- Para determinar si el usuario espera en la cola a que el servidor lo atienda se decidió en base a una probabilidad de $1/(n+1)$, 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 la operación del sistema, éste puede trabajar adecuadamente hasta con k usuarios (donde $k = 100$).

Vértice	Relación Funcional
Vs1,vs2	Acceso al SIV
Vop3	Perfil Operacional del Público en General
Vop4	Perfil Operacional de los Alumnos
Vop5	Perfil Operacional de los Investigadores
Vop6	Perfil Operacional del Coordinador Académico
Vo7	Consulta de Cursos P.G.
Vo9	Consulta de Cursos para los alumnos
Vo11	Reinscripción a otro curso
Vo12	Baja del curso
Vo14	Inscripción en un nuevo curso
Vo16	Consulta de la Información General de un curso o de un investigador
Vo18	Procesamiento de los cambios de la información general de un curso o investigador
Vo19	Procesamiento de la baja de la información de un curso o de un investigador
Vo21	Procesamiento de la alta de un nuevo curso o investigador
Vo23	Asignación de usuario y password
Vo25	Validación en los módulos del sistema
Vd8	Consulta del catalogo de cursos
Vd10	Cambio de curso
Vd13	Selección de un nuevo curso
Vd15	Consulta del catalogo de cursos e investigadores
Vd17	Cambios en la información general de un curso ó investigador
Vd20	Alta de la información general de un nuevo curso o de un investigador
Vd22	Alta de nuevos usuarios en el sistema
Vd24	Asignación del periodo de inscripciones
Vop4b	Acceso del Perfil Investigador ó Perfil Coordinador al Perfil Alumnos con la restricción de acceso solo a los alumnos que asesora
Vop5b	Acceso del Perfil Coordinador al Perfil Investigador

Tabla 4.2: Relación de vértices y entidades funcionales

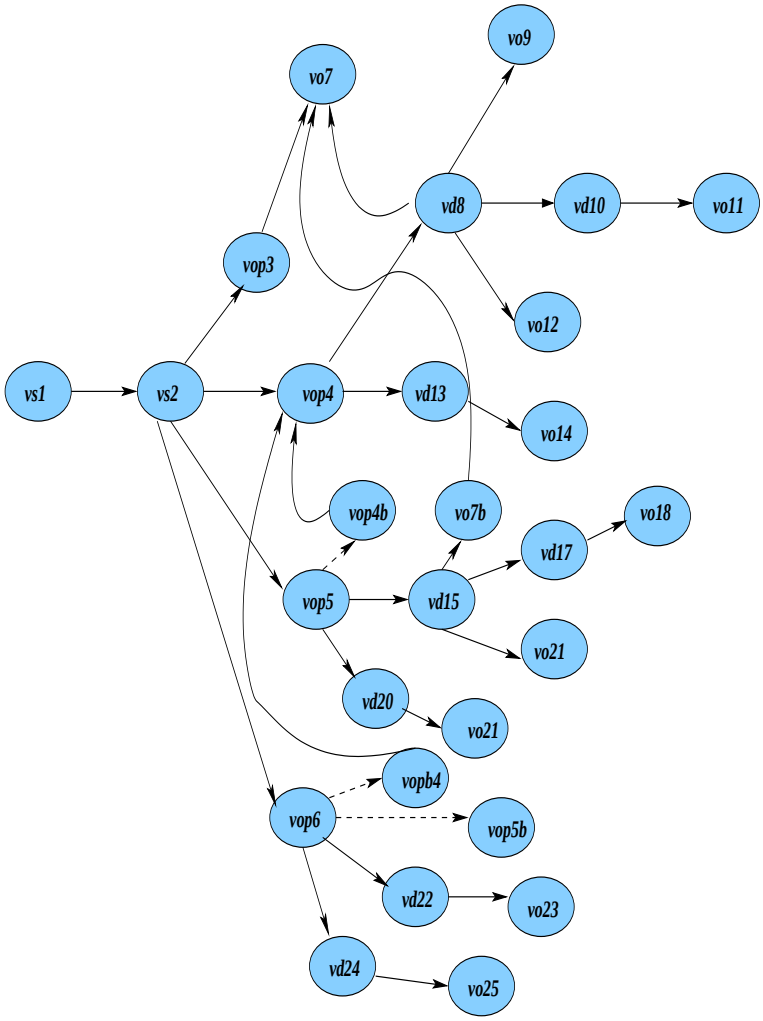

Figura 4.4: Grafo de Dependencias Funcionales para el SIV

- Dependiendo de su vista, los usuarios pueden realizar solo x tipos de transacciones, con t unidades de tiempo.

- El tiempo de operación de los usuarios en el sistema esta determinado por una distribución normal ($\mu = 5$ y $\sigma = 3$).

- La distribución de las vistas esta en un proporción de 70 % para el perfil de alumnos, 20 % en el perfil de investigadores y 10 % en el perfil del público en general.

- La distribución para determinar el tiempo en cada transacción esta de acuerdo a una distribución de Weibull donde $\alpha = 2{,}16$ y $\beta = 12{,}8$.

$$f(x) = \alpha\beta^{-\alpha}x^{\alpha-1}e^{-(x/\beta)^{\alpha}} \tag{4.3}$$

La simulación nos sirve para determinar la probabilidad de ocurrencia de cada una de las operaciones del los perfiles operacionales en un marco de tiempo que es de 1000 unidades de tiempo.

El lenguaje de programación empleado para la simulación fue Java. El diagrama a bloques de la operación del productor se muestra en la Figura 4.5.

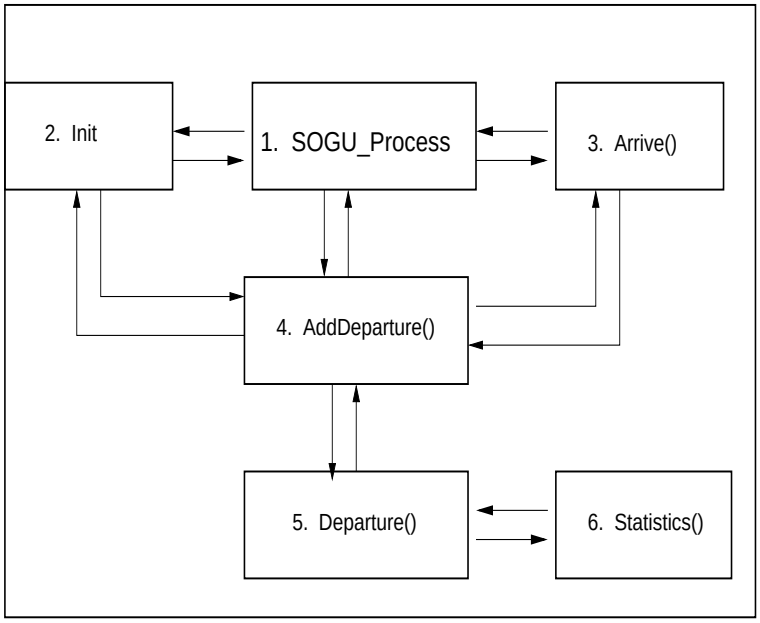

Figura 4.5: Diagrama a bloques de la simulación

En este diagrama se pueden apreciar 6 módulos. En el módulo principal, Proceso del SIV, se ejecuta el ciclo de la simulación. El segundo módulo Rutina de Inicialización tiene 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()*, establecemos el tiempo que va a estar el usuario realizando la operación apoyados con una distribución normal, se realiza un llamada al la función *departure()* y se verifica la generación de errores durante la operación simulada. En el caso de que hayan existido errores, se aumenta la densidad de fallos 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 ó parte del sistema que está utilizando. El sexto módulo, obtiene las estadísticas de las ocurrencias de los usuarios en las operaciones del sistema. Para el procesamiento se colocaron estructuras de almacenamiento en AddDepartures, y el módulo de inicialización.

El sexto módulo *statistics*, obtiene las estadísticas de las frecuencias de los usuarios en las operaciones del sistema. Para el procesamiento se colocaron estructuras de almacenamiento en AddDepartures y Departure.

Después de haber realizado 5000 simulaciones, realizamos las estadísticas de la información y se determinó el promedio de la frecuencia de incidencia de cada uno de los perfiles operacionales. En cada una de las operaciones, al dividirla entre el tiempo de simulación (que es de 1000 ejecuciones por cada simulación) estimamos la probabilidad de ocurrencia de los fallos. Para este proceso desarrollamos programas con guiones en shell. La plataforma que se utilizó para realizar este proceso fue Solaris 2.9 [107]. Los resultados pueden observarse en la Tabla 4.3. En la Tabla 4.3 en la primera columna representamos el (los) vértice(s) de la ruta funcional hemos evaluado. En la segunda columna tenemos los componentes implícitos en la relación funcional. En la tercera columna tenemos la entidad funcional, en la cuarta columna tenemos la frecuencia con la que se utiliza de acuerdo a los promedios de las 5000 simulaciones, y en la quinta columna tenemos la probabilidad de ocurrencia.

4.5.4. Estimación de la Complejidad Ciclomática de las Relaciones Funcionales.

Finálmente y de acuerdo a la sección 4.4.1, para que nuestro GDF quede listo para los Escenarios de Riesgo, es necesario estimar la Complejidad Ciclomática de cada una de las entidades funcionales y asociarlas a sus aristas.

En la Tabla 4.3 tenemos la relación entre las entidades funcionales y los componentes que están presentes en esta entidad

La métrica seleccionada es el número de McCabe. Los pesos asignados a las relaciones funcionales se pueden observar en la Figura 4.6

4.5.5. Determinar los Escenarios de Riesgo en el sistema de software

De acuerdo a la sección 4.4.2 el proceso para determinar los escenarios de riesgo para el sistema de software estudiado es el siguiente:

Entidad funcional	Vértice	Componente Implícito	Frecuency	Ocurrence_rate
Acceso	Vs1	C1	340	0.34
Establecer-Perfil	vs2	C2	340	0.34
Perfil Operacional PG	vop3	C3,C4	29	0.029
Consulta-Cur	Vd8, Vo7, Vd15, Vo16	C5,C6	343	0.658
Consulta-Inv	Vd8, Vo7 Vd15, Vo16	C5,C6	342	0.658
Perfil Operacional Alumno	vop4	C3,C4	316	0.316
Consulta-Alum	Vd8,Vd15, vo9	C5,C6	318	0.637
Consulta CursoXAlumno	vo9	C6,C7	319	0.637
Alta-CursosxAlumno	vo14	C8,C9	255	0.255
Baja-CursosxAlumno	vo11	C8,C10	189	0.189
Cambios-CursosxAlumno	vo12	C8,C11	124	0.124
Perfil Operacional Investigador	vop5	C3,C4	28	0.028
Alta-Inf	vo21	C12,C13	50	0.05
Baja-Inf	vo19	C5,C14	42	0.042
Cambios-Inf	vo18	C5,C15,C16	35	0.035
Perfil Operacional Coordinador	vop6	C3,C4	29	0.029
Alta-usuarios	vo23	C17,C18	29	0.029
Periodo-Insc	vo25	C19,C20	22	0.066
Alta-Periodo	vo25	C19,C20	22	0.066
Baja-Period	vo25	C19,C20	15	0.066
Cambios-Period	vo25	C19,C20	7	0.066

Tabla 4.3: Relación de vértices y entidades funcionales con los valores de la taza de ocurrencia

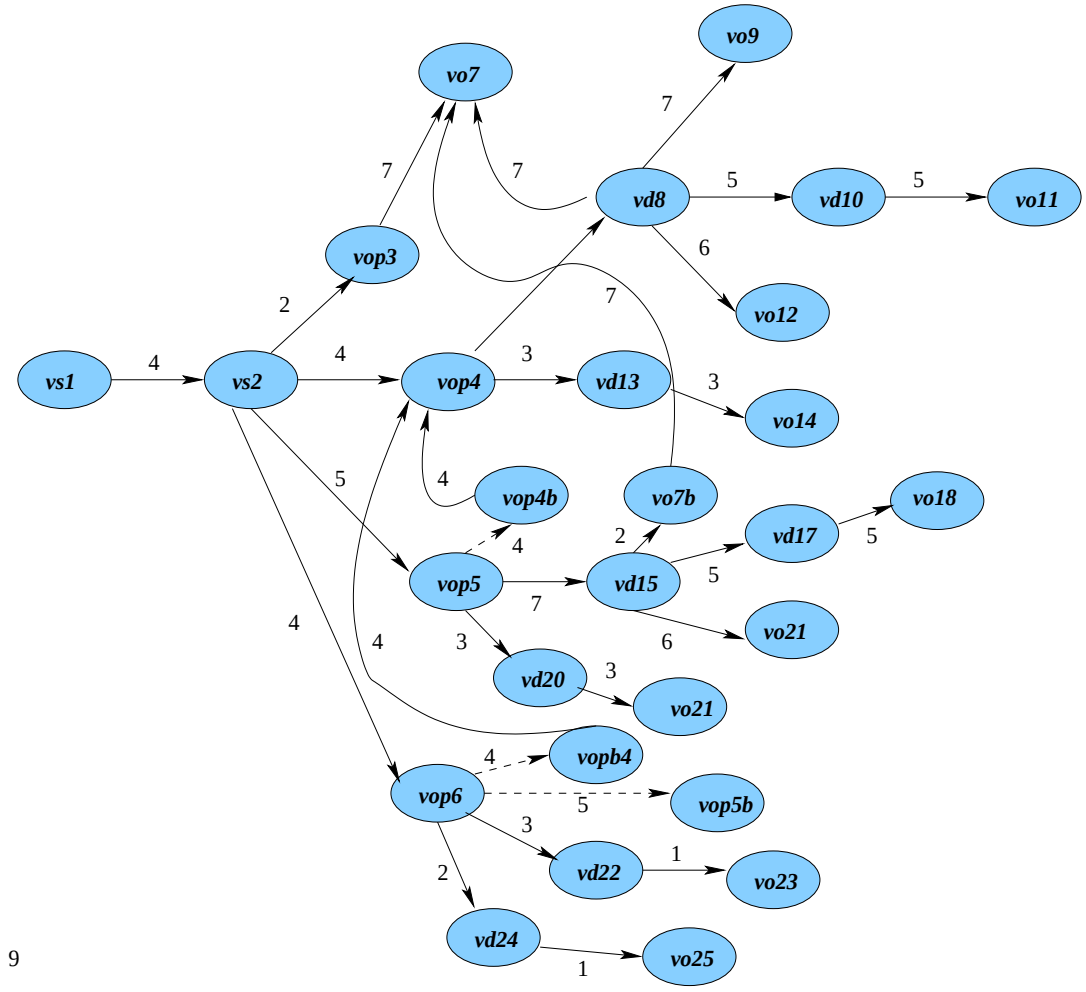

Figura 4.6: Asignación de Peso de acuerdo a la Complejidad Ciclomatica de McCabe

1. **Aplicación el algoritmo de caminos mínimos**.

   Nosotros aplicamos el algoritmo de Dijkstra. El algoritmo descubre en primera instancia las rutas de menor peso. En nuestro caso implica que son las rutas menos complejas. Posteriormente se buscan las rutas con mayor complejidad hasta que descubre la ruta más compleja. Los resultados los podemos observar en la Tabla 4.4. En la Tabla 4.4 las columnas describen el identificador de los Escenarios de Riesgo, la ruta de vértices que conforman al Escenario, el peso de acuerdo a la complejidad ciclomática (este peso se determina con la suma de las aristas por donde pasa el Escenario), la relación de los componentes que integran la ruta del Escenario (en esta columna podemos observar que existen componentes que están presentes en la mayoría de los escenarios) y la probabilidad de ocurrencia (debido al proceso que hemos seguido tenemos que cada escenario esta relacionado con un servicio u operación funcional del

ID	PATH	Peso	Componentes	Prob-ocurr
E1	vs1, vs2, vop5 ,vd15, vd17, vo18	260	C1,C2,C3,C4,C5,C15,C16	0.035
E2	vs1, vs2, vop4, vd8, vd10, vo11	250	C1,C2,C3,C4,C5,C8,C10	0.189
E3	Vs1, vs2, vop4, vd8 ,vo9	220	C1,C2,C3,C4,C5,C8,C10	0.189
E4	Vs1, vs2, vop5, vd15, vo16	220	C1,C2,C3,C4,C5,C6	0.658
E5	Vs1, vs2, vop5, v15 ,vo19	220	C1,C2,C3,C4,C5,C14	0.042
E6	Vs1, vs2, vop4, vd8, vo12	210	C1,C2,C3,C4,C5,C8,C11	0.124
E7	Vs1, vs2, vop5, vd20, vo21	150	C1,C2,C3,C4,C12,C13	0.05
E8	Vs1, vs2, vop4, vd13 ,vo14	140	C1,C2,C3,C4,C8,C9	0.255
E9	Vs1, vs2, vop3, vo7	130	C1,C2,C3,C4,C5,C6	0.658
E10	Vs1, vs2, vop6 vd22 ,vo23	120	C1,C2,C3,C4,C17,C18	0.029
E11	Vs1, vs2, vop6 vd24, vo25	110	C1,C2,C3,C4,C19,C20	0.066

Tabla 4.4: Resultados para los escenarios de riesgos

sistema, de esta forma hemos podido establecer la relación del escenario con la probabilidad de ocurrencia).

2. **Estimación de la Probabilidades de Fallo de los Escenario de Riesgo**.

   En base de la ecuación 4.1 se obtuvo la Probabilidad de Fallo cuyos resultados se muestran de la Tabla 4.5.

   Los escenarios están ordenados de acuerdo a su complejidad. Al estimar la Probabilidad de Fallo que relaciona la complejidad y la probabilidad de ocurrencia, se puede observar que los valores de criticidad de los escenarios cambian porque en este caso los escenarios E4, y E9 (Tabla 4.5) adquieren la mayor criticidad, por el contrario el escenarios E1 (Tabla 4.4) va a la séptima posición. De la relaciones propuestas en la sección 4.4.2, determinamos la Probabilidad de Fallo de acuerdo a la ecuación 4.1 para cada Escenario y los resultados se muestran en la Tabla 4.5

   En la Tabla 4.5 las columnas indican el identificador de los Escenarios de Riesgo, la ruta de vértices que conforman al Escenario, el peso de acuerdo a la complejidad ciclomática, la relación de los componentes que integran la ruta del Escenarios. En la quintan la probabilidad de ocurrencia y finálmente el valor de la Probabilidad de Fallo.

3. **Determinación de los elementos en los casos de prueba críticos**.

   Adicional a la PFER, los escenarios también nos proporcionan información relevante para

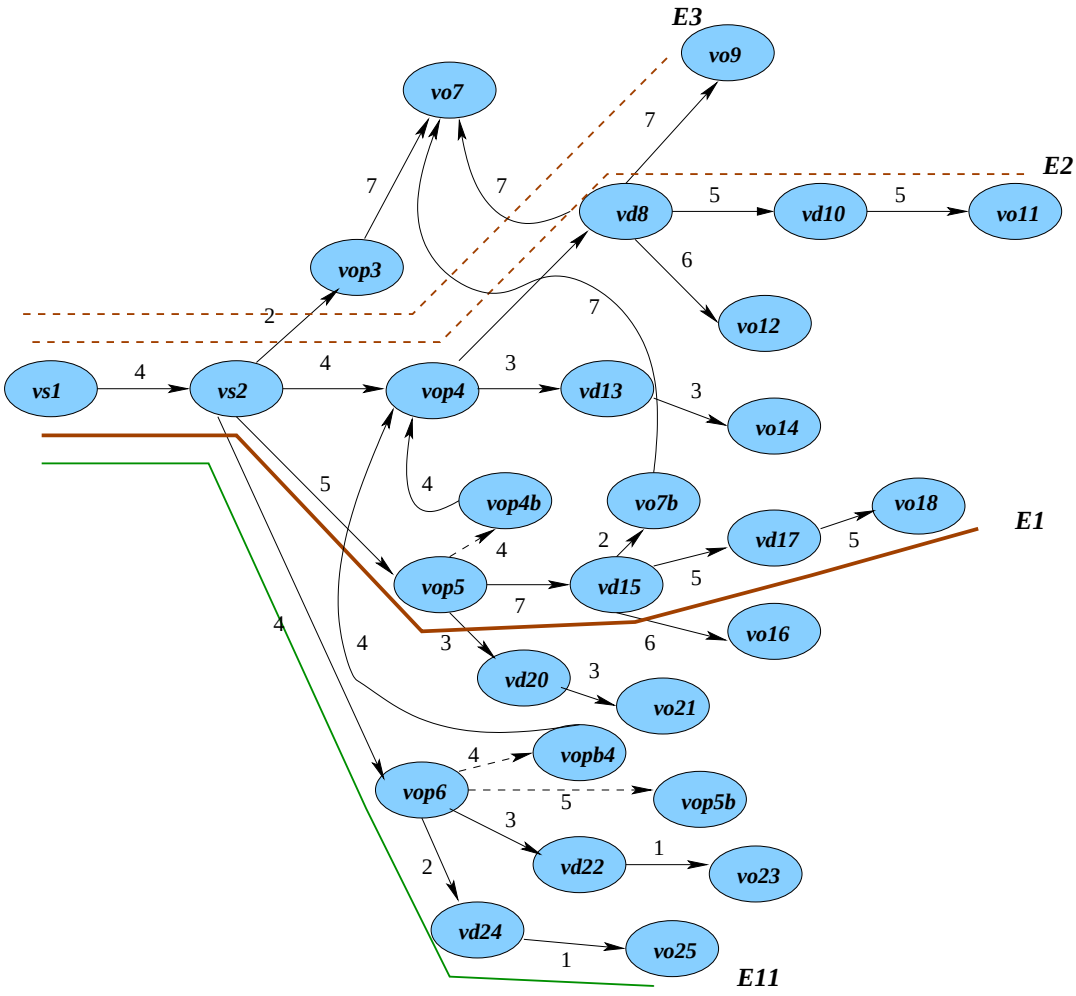

Figura 4.7: Escenarios de Riesgo

la etapa de pruebas. Por lo cual podemos concluir que es conveniente extender el rango de pruebas en el perfil del investigador cuando realiza las operaciones de cambio en la información del sistema. Otra operación critica es la consulta cuando esta se sobrecarga.

4. **Determinación de las Probabilidades de Fallo de los Componentes del Sistema**.
   De la relaciones propuestas en la secciíon 4.4.2, determinamos la Probabilidad de Fallo para cada componente del sistema de acuerdo a la ecuación 4.2. Los resultados los podemos observar en la Tabla 4.6. En la Tabla 4.6 se la columnas indican el componente que se esta evaluando, el número de escenarios donde interviene, el valor del promedio de la Probabilidad de Fallo de los escenarios donde actúa, el valor del factor (este factor se estima de acuerdo al número de escenarios donde actúa) de acuerdo a la relación 4.2, finálmente tenemos el valor de la

ID	PATH	Peso	Componentes	Prob-ocurr	Prob Fallo
E4	vs1, vs2, vop5, vd15, vo16	220	C1,C2,C3,C4,C5,C6	0.658	0.383
E9	V1, v2, v3, v7	130	C1,C2,C3,C4,C5,C6	0.658	0.361
E8	V1, v2, v4, v13, v14	140	C1,C2,C3,C4,C8,C9	0.255	0.1619
E2	vs1, vs2, vop4, vd8, vd10, vo11	250	C1,C2,C3,C4,C5,C8,C10	0.189	0.156
E3	vs1, vs2, vop4, vd8 ,vo9	220	C1,C2,C3,C4,C5,C8,C10	0.189	0.148
E6	V1, v2, v4, v8, v12	210	C1,C2,C3,C4,C5,C8,C11	0.124	0.1137
E1	vs1, vs2, vop5 ,vd15, vd17, vo18	260	C1,C2,C3,C4,C5,C15,C16	0.035	0.0815
E5	vs1, vs2, vop5, v15 ,v19	220	C1,C2,C3,C4,C5,C14	0.042	0.075
E7	V1, v2, v5, v20, v21	150	C1,C2,C3,C4,C12,C13	0.05	0.0619
E11	V1, v2, v6, v24, v25	110	C1,C2,C3,C4,C19,C20	0.066	0.06
E10	V1, v2, v6, v22, v23	120	C1,C2,C3,C4,C17,C18	0.029	0.044

Tabla 4.5: Ordenamiento de los escenarios de riesgo en función de la probabilidad de Fallo

Probabilidad de Fallo del componente estudiado.

Después de haber determinado la Probabilidad de Fallo para cada componente, tenemos la relación del nombre de los componentes y su identificador en la Tabla4.7. Las columnas describen el identificador del componente hasta este momento, el nombre del componente en el SIV y el valor de la Probabilidad de Fallo del componente. Hemos ordenado los componentes de acuerdo a su Probabilidad de Fallo en orden decreciente, de esta forma el componente más crítico se encuentra en la primera posición.

5. **Análisis de los resultados**.

   De acuerdo a los resultados, podemos observar que los escenarios que tienen una Probabilidad de Fallo mayor son E4 y E9. El escenario E4 está relacionado con el perfil del investigador en la operación de bajas en la información de sus alumnos, y de la información en general. Es interesante observar que esta ruta tiene un valor de complejidad medio en relación con otros escenarios, pero con respecto del valor que denota la probabilidad de ocurrencia, es de los más utilizados por los usuarios del sistema. Otro escenario crítico es el de la operación de consultas en la información de cursos e investigadores. En este escenario la complejidad ciclomática no tiene un valor muy alto, pero la probabilidad de ocurrencia si es la más alta, lo que implica que esta es una operación que será utilizada con una frecuencia muy alta por los usuarios de sistema.

Componente	Escenarios donde Interviene	Probabilidad Promedio	Factor de Escenario	Probabilidad Fallo
C1	11	0.1496	1	0.1496
C2	11	0.1496	1	0.1496
C3	11	0.1496	1	0.1496
C4	11	0.1496	1	0.1496
C5	7	0.1883	0.6363	0.11982
C6	2	0.372	0.1818	0.06763
C7	1	0.148	0.0909	0.01345
C8	3	0.143	0.2727	0.039
C9	1	0.1619	0.0909	0.01471
C10	1	0.156	0.0909	0.01418
C11	1	0.1137	0.0909	0.0103
C12	1	0.0619	0.0909	0.0056
C13	1	0.0619	0.0909	0.0056
C14	1	0.075	0.0909	0.0068
C15	1	0.0815	0.0909	0.0074
C16	1	0.0815	0.0909	0.0074
C17	1	0.044	0.0909	0.004
C18	1	0.044	0.0909	0.004
C19	1	0.06	0.0909	0.0054
C20	1	0.06	0.0909	0.0054

Tabla 4.6: Parámetros para Estimar la Probabilidad de Fallo de los Componentes

Componente	Nombre	Probabilidad Fallo
C1	password.php	0.1496
C2	conecta.php	0.1496
C3	menu.php	0.1496
C4	menu_sig.php	0.1496
C5	catalogo.php	0.1198
C6	consultas.php	0.067
C8	cat_cur_alumn.php	0.039
C9	procesa_alal.php	0.0147
C10	procesa_alba.php	0.01418
C7	cursos_alumn.php	0.013
C11	procesa_alca2.php	0.01
C15	cambios.php	0.0074
C16	procesa_cambios.php	0.0074
C14	bajas.php	0.0068
C12	altas.php	0.0056
C13	procesa_altas.php	0.0056
C19	ambiente.php	0.0054
C20	procesa_ambiente.php	0.0054
C17	insert_user.php	0.004
C18	procesa_user.php	0.004

Tabla 4.7: Estimación de la Probabilidad de Fallo de los Componentes

Feature	Risk Scenarios
Density Defects (Number of Errors	8635
MTTF (seg.)	4.33
Component Failure	**DD**
Menú.php	2272
Menú_sig.php	1631
Conecta.php	1603
Consultas.php	1133
catalogo.php	455
cat_cur_alum.php	453
cursos_alumn.php	226
Bajas_alumn.php	225
Procesa_altas.php	224
Procesa_ambiente.php	224
Procesa_alca2.php	223
Procesa_alba.php	124
cambios.php	3

Tabla 4.8: Resultados de las pruebas en el SIV

Al estudiar la Probabilidad de Fallo de los componentes, observamos que los componentes que tienen un valor mayor son los que siempre se utilizan, porque establecen el alcance en la información para cada vista del sistema, que el caso de los componentes C1,C2,C3 y C4. Otros componentes críticos son los relacionados con las consultas que es caso de los componentes C5 y C6, esta operación es la más utilizada por parte de los usuarios.

Nuestra propuesta cuantifica las rutas críticas haciendo un promedio entre la complejidad y la ocurrencia. Es importante observar que la combinación de los componentes implícitos en las relaciones funcionales pueden hacer muy sencible a fallos una ruta, como es el caso del escenario E4. Sin embargo cuando se determinan las Probabilidades de Fallo de los componentes podemos estimar que los componentes que intervienen en estas rutas no necesariamente tienen que ser los mas críticos. El valor de la ocurrencia puede hacer muy crítica una sección del sistema no tan compleja.

Con los resultados de las pruebas reflejados en la Tabla 4.9 podemos confirmar que nuestro modelo tiene una validéz predictiva. De acuerdo a las pruebas los componentes que tiene una mayor probabilidad de fallo son los componentes: C1,C2,C3 y C4. En nuestro estudio

previo a las pruebas, determinamos que estos 4 componentes son los que tienen una mayor Probabilidad de Fallo, es decir donde esperamos una ocurrencia de fallos mayor.

Con la información que nuestro modelo nos proporciona podemos predecir las secciones del sistema donde es necesario ampliar el número de pruebas, las rutas donde debemos ampliar el número de pruebas y al analizar la relaciones funcionales críticas podemos inferir el rango de valores en las variables de entrada que es necesario cubrir. Al determinar estos valores optimizamos la cobertura de las evaluaciones, así el esfuerzo invertido en esta etapa se canaliza hacia los componentes y secuencias que tienden a fallar con más frecuencia.

4.5.6. Automatización de las pruebas.

La herramienta que hemos utilizado para esta etapa es una extensión de la presentada en el capítulo 3. En este caso la herramienta tiene una complejidad mayor que la anterior, pero las mejoras son notables. Algunas ventajas adicionales son:

1. Las evaluaciones de la pruebas se hacen en tiempo de ejecución.
2. El análisis de las respuestas se realiza mediante un parser y un escaner siguiendo la teoría de los compiladores.
3. Es posible obtener diversas métricas de software: la Densidad de Defectos **DD** y la Media de Ocurrencia de Fallos **MTTF**.
4. La bitácora de errores es mucho más precisa, reporta fallos, secuencias de fallo y componentes anómalos, así como el tiempo en el que ocurrió el fallo.
5. Es posible evaluar cualquier sistema de software.
6. Es posible aplicar cualquier modelo de evaluación en la cobertura.

La herramienta que hemos desarrollado es un framework de evaluación que toma un conjunto de escenarios (rutas ó secuencias) específicos como entrada. Esta herramienta prepara los datos de prueba para la ejecución de las pruebas, ejecuta las pruebas y evalúa los resultados de la pruebas en línear. De esta manera puede determinar métricas como la densidad de defectos **DD** y sobretodo la media de ocurrencia de fallos **MTTF**. Los resultados los guarda en una bitácora. Estas bitácoras son una importante fuente de información, porque cada log contiene el fallo, la secuencia donde se

presenta el fallo y el componente, así como el tiempo en el que ocurre dicho fallo. La herramienta se divide en 5 subsistemas:

1. *Risk Escenaries Subsystem* proporciona la entrada, que son los escenarios en forma de listas.

2. *Testing Tool System* es el sistema central, este subsistema controla el ambiente de evaluación para las pruebas. El subsistema genera evaluadores virtuales siguiendo una distribución exponencial. Las rutas de evaluación de estos evaluadores virtuales se hacen de acuerdo a la prioridad de los escenarios. Estas rutas se manejan en forma de listas. Cada nodo de la lista tiene relación con los valores de las matrices de prueba que están en el subsistema *Test Matrix Subsystem.*

3. *Test Matrix Subsystem.* En este subsistema se guardan los valores de los nodos que se están evaluando, como son el (los) componente(s) que lo integran, las variables de entrada y los resultados de salida esperados. En este sentido el sistema tiene cierta inteligencia porque es capáz de determinar mediante el análisis de la respuesta el siguiente nodo a evaluar y prepara los datos de la evaluación, lo cual se confima con la lista de los nodos del escenario. Esto es posible debido a que el subsistema de análisis *Analizer System.*

4. *Analizer System* está conformado por un escaner y un parser, lo cual le da ventajas a la herramienta para determinar el siguiente paso de la prueba. Cuando una secuencia ha concluido el subsistema *Analizer System* se encarga de realizar la evaluación de la prueba, en este punto se sigue la técnica de caja negra, los valores de entrada y de salida son determinados con la información que el subsistema de Matriz de pruebas reporta. En el caso de que exista algún fallo este se registra en la bitacora y se contabiliza.

   A esta herramienta de evaluación también le hemos agregado subsistemas que son capaces de automatizar nuestra propuesta. Para esto desarrollamos los subsistemas

5. *Functional Graph Subsystem y Risk Escenaries Subsystem*, los cuales se encargan de formar el GDF de acuerdo a los datos de entrada del usuario y aplican el algoritmo de caminos mínimos, en este caso es el algoritmo de Dikjistra. Estos subsistemas preparan la entrada para la ejecución de subsistema de evaluación. Estos módulos están a consideración del usuario pueden ser o no ejecutados, porque si se desea evaluar otro modelo, basta con sólo proporcionar

la rutas ó secuencias de evaluación y llenar la matriz de pruebas, para que el sistema realice la cobertura de cualquier modelo en cualquier sistema.

El diagrama a bloques de la herramienta de evaluación se presenta en la siguiente Figura 4.8.

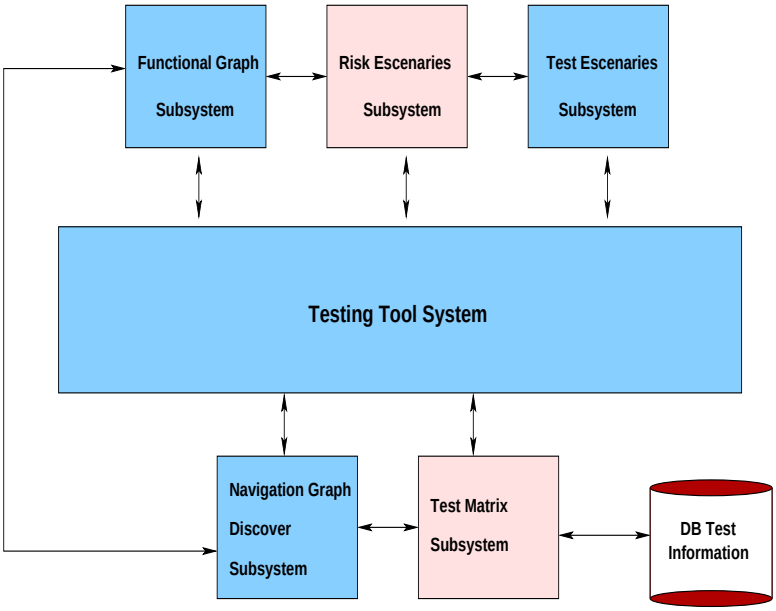

Figura 4.8: Arquitectura de la Herramienta de Evaluación

En la Tabla 4.9 tenemos en los primeros renglones los valores de las métricas generales que obtuvimos en la etapa de pruebas. En los siguientes renglones deglosamos la Densidad de Defectos estimadas para cada uno de los componentes que presentaron fallos durante las pruebas.

De acuerdo a los resultados reflejados en la Tabla 4.9 podemos confirmar que nuestro modelo tiene una validéz predictiva. De acuerdo a las pruebas los componentes que tiene una mayor probabilidad de fallo son los componentes: C1,C2,C3 y C4. En nuestro estudio previo a las pruebas, determinamos que estos 4 componentes son los que tienen una mayor Probabilidad de Fallo, es decir donde esperamos una ocurrencia de fallos mayor.

Con la información que nuestro modelo nos proporciona podemos predecir las secciones del sistema donde es necesario ampliar el número de pruebas, las rutas donde debemos ampliar el número de pruebas y al analizar la relaciones funcionales críticas podemos inferir el rango de valores en las variables de entrada que es necesario cubrir. Al determinar estos valores optimizamos la cobertura de las evaluaciones, así el esfuerzo invertido en esta etapa se canaliza hacia los componentes y secuencias que tienden a fallar con más frecuencia.

Feature	Risk Scenarios
Density Defects (Number of Errors	8635
MTTF (seg.)	4.33
Component Failure	**DD**
Menú.php	2272
Menú_sig.php	1631
Conecta.php	1603
Consultas.php	1133
catalogo.php	455
cat_cur_alum.php	453
cursos_alumn.php	226
Bajas_alumn.php	225
Procesa_altas.php	224
Procesa_ambiente.php	224
Procesa_alca2.php	223
Procesa_alba.php	124
cambios.php	3

Tabla 4.9: Resultados de las pruebas en el SIV

La etapa de pruebas en nuestra propuesta es muy importante porque aunque nuestro enfoque de predicciones se acerca mucho a los resultados obtenidos,la etapa de pruebas detalla el nivel de confiabilidad de cada componente. En nuestro caso tenemos la misma probabilidad de fallo para los 4 componentes y con la etapa de pruebas detallamos los valores de fallo asignados a los componentes.

4.6. Conclusiones

Después de aplicar nuestro modelo, podemos observar que cumple con criterios de calidad que los modelos eficientes deben tener, como la *validéz predictiva*. En base de los resultados podemos estimar que la *validéz predictiva* de nuestro modelo es competente porque los resultados teóricos coinciden con los resultados que se obtienen en un contexto práctico y real. Las prediciones de confialidad que arroja nuestro modelo no sólo están basadas en la funcionalidad, como en los anteriores modelos, sino que también analizamos la complejidad ciclomática de la arquitectura del sistema. El análisis integrado de estas dos variables nos permite generar mejores predicciones con un mayor rango de cobertura y en un menor tiempo.

Otro criterio de calidad con el que cumple nuestro modelo es la *simplicidad en su aplicación*. Al desarrollar el modelo podemos observar que para determinar el GDF y los ER no se necesita de un complejo conocimiento matemático y además los valores de entrada y en general del proceso pueden automatizarce con sistemas de computo, esta situación reduce a horas el trabajo que de forma manual necesitaría de meses para su desarrollo.

Estos dos criterios de acuerdo a los especialistas del área [94] (Iannino, Okumoto, y otros) son los más importantes. Esta situación se hace muy clara cuando el modelo se trata de aplicar en contextos productivos de la industria.

En nuestro caso también tenemos criterios de calidad adicionales como la *aplicabilidad*, esto es que el enfoque de generar trayectorias mediante grafos nos permite hacer estudios de sistemas de gran escala y con un costo computacional relativamente pequeño.

Otro criterio con el que contamos es la *calidad en las hipótesis* que se derivan de nuestro modelo ya que nos muestran de una manera plausible que la combinación de los componentes relacionados en una operación funcional, puede determinar un camino crítico. Otro punto es que la frecuencia de uso por parte de los usuarios en un servicio del sistema también interviene en la criticidad de los servicios. De estos puntos podemos concluir que la complejidad ciclomática en una ruta si afecta la criticidad de esta ruta, elevando su Probabilidad de Fallo. Otro punto es que la sobrecarga en algún servicio del sistema afecta de una manera directa la criticidad de los componentes que intervienen en esta operación, elevando su Probabilidad de Fallo.

En general la metodología para obtener nuestro modelo proporciona un marco eficiente e innovador para la localización efectiva de fallos y análisis de la confiabilidad en los sistemas de software en ambientes operativos. Estas cualidades finálmente benefician a la etapa de pruebas debido a que con estos resultados es posible localizar y eliminar fallos con mayor certeza.

Capítulo 5

Estudio comparativo.

5.1. Resumen

En este capitulo se presenta un estudio comparativo entre el modelo de evaluación de la confiabilidad propuesto en el capitulo 4, y el estudio propuesto por Whittaker [99].

Nuestra propuesta se enfoca en el desarrollo de un modelo predictivo para el proceso de evaluación de la confiabilidad. Este modelo es una guía para planear las pruebas y ampliar la cobertura en los componentes donde se tiene una mayor probabilidad de fallo. El modelo propuesto además de estimar la cobertura del sistema durante el proceso de evaluación, nos proporciona información acerca de la probabilidad de fallo para cada componente del sistema, las secuencias (paths) de evaluación más sensibles a fallos y en general información importante para la fase de evaluación y corrección de errores.

Este estudio comparativo lo hemos implementado con simulaciones de manera automática y se aplica durante la fase de pruebas. La evaluación de nuestro modelo y del modelo de Whittaker [100] la llevamos a cabo utilizando un caso de estudio de un sistema de inscripciones vía Internet (SIV) para una Universidad. La evaluación de nuestro modelo la llevamos a cabo mediante una herramienta de evaluación de la confiabilidad.

La cobertura de las evaluaciones y las secuencias (paths) de evaluación se basan en cada modelo. Los errores resultantes de las pruebas son analizados por nuestra herramienta, la cual genera

bitácoras de información con las que podemos obtener la ruta de evaluación que resulta con fallos, el componente anómalo y el error obtenido. Las métricas de software que utilizamos para evaluar la confiabilidad son la densidad de defectos (DD) y la media de ocurrencia de fallos (MTTF) a nivel global y en cada uno de los componentes del sistema.

El objetivo del estudio comparativo es demostrar dentro de un ambiente real las ventajas que ofrece nuestro modelo.

De los trabajos relacionados que revisamos, hemos seleccionado el estudio de James A. Whittaker y Michael G. Thomason, A Markov Chain Model for Statistical Software Testing [99], para compararlo contra nuestro modelo. Este estudio genera un modelo estocástico que determina la cobertura de evaluación del sistema. La base de este estudio es mediante la técnica de las cadenas de Markov. De todos los modelos estudiados y descritos en el capitulo 4, el modelo de Wittaker [100] obtiene parámetros similares a los de nuestro modelo, además de que es un modelo no solo teórico que implementado en la practica, por lo cual lo consideramos el mas apropiado para su comparación.

Dada la importacia practica del modelo de Wittaker [100], este fue implementado dentro de un framework y herramienta de evaluación denominado MaTeLo (Markov Chain Test Logic).

Como se explica en el capitulo anterior, en nuestro modelo predictivo el análisis de la confiabilidad la llevamos a cabo mediante la técnica de grafos. Los *nodos* del grafo son relacionados con la funcionalidad de los componentes y el peso de las aristas que unen estos nodos está de acuerdo a la complejidad ciclomática de los componentes que intervienen en la relación funcional. Las relaciones de funcionalidad las obtenemos del documento de especificaciones y la complejidad ciclomática la estimamos mediante métricas de software. En la propuesta del estudio comparativo, el modelo predictivo se basa en un estudio estocástico. El modelo se representa mediante cadenas de Markov y los componentes del sistema son tratados como estados de la cadena.

Durante la fase de pruebas en nuestra propuesta, las secuencias ó paths de navegación de pruebas se determinan automáticamente mediante nuestra herramienta, la cual utiliza nuestro modelo predictivo. Estas secuencias son denominadas *Escenarios de Riesgo* los cuales son evaluados por nuestro framework. En el estudio de Wittaker estas secuencias son rutas de pruebas estimadas de acuerdo al criterio del analista que se basa en el modelo de la Cadena de Markov.

5.2. Introducción

Las pruebas en los sistemas de software llevadas a cabo mediante evaluación estadística son experimentos aleatorios que requieren de una caracterización del espacio de muestras y de la distribución de probabilidad asociada. También se requiere de la definición de un espacio de eventos adecuado y de un método computacional de comparación descriptiva de las variables aleatorias.

Una Cadena de Markov se define como un proceso estocástico discreto que cumple con la propiedad de Markov, es decir, si se conoce la historia del sistema hasta su instante actual, su estado presente resume toda la información relevante para describir en probabilidad su estado futuro. Para determinar los parámetros de una cadena, debemos primero *resolver* la cadena, esto es, determinar la probabilidad que tiene cualquier estado en cualquier instante de tiempo. En algunas ocasiones, en la práctica esta tarea es ardua, debido a que involucra la solución de complicadas ecuaciones diferenciales. En otras ocasiones se pueden obtener soluciones transitorias. Sin embargo en la mayoría de los casos es necesario determinar los valores de los estados en equilibrio o *soluciones estacionarias*. Estos valores se refieren a las probabilidades que toman los estados después de que el sistema ha estado en operación por un período suficientemente largo de tiempo. En el apéndice B1 se da una introducción a las Cadenas de Markov.

Del estudio que realizan Whittaker y Thomason, en donde proponen un modelo para evaluar estadísticamente sistemas de software mediante Cadenas de Markov (su detalle viene en el apéndice C) podemos resumirlo en los siguientes puntos:

1. **Diseño de la Cadena de Markov**. Para el desarrollo de este punto los autores proponen un proceso que es descrito en el apéndice C. En este punto es principal obetivo es establecer el modelo con sus entradas. Para esto es necesario realizar las siguientes actividades.

   - *Estimación de las probabilidades de entrada de la cadena de Markov.*
   - *Generación de las secuencias de la cadena de Markov.*

2. **Análisis de los estados estables**. En este punto se buscan los valores de las soluciones estacionarias, para lo cual es necesario utilizar tiécnicas matemáticas y sistemas de computo complejos. En esta parte es posible apoyarse con la descripcion de la teoría y conceptos del apéndice B. Las actividades para este proceso son las siguientes:

   - *Representacion de la cadena de Markov como una matrix de estados U.*

- *Solución de los sistemas de ecuaciones lineales de la matrix de estados U.*
- *Determinacion del vector de soluciones π.*
- *Desarrollo de las funciones de cobertura del caso de estudio , de acuerdo al vector de soluciones π_i.*

3. **Desarrollo de las evaluaciones del caso de estudio SIV de acuerdo al modelo**. En esta actividad solo es ncesario contar con una herramienta que pueda aplicar la cobertura de las pruebas de acuerdo al modelo. Otro punto importante es que esta herramienta pueda determinar los fallos cuando se presentan.

4. **Análisis de resultados**. De acuerdo a los resultados que se obtienen es posible determinar la media de ocurrencia de Fallos. Tambien es posible determinar el número de fallos y un estimados del componente donde es probable que incidan. Esto depende de la herramienta con la que sea evaluado el sistema de software.

5. **Conclusiones**. Finálmente el registro del comportamiento del sistema a lo largo del tiempo, puede darnos una idea del comportamiento general, lo cual nos ayuda a tomar desiciones en la parte operativa.

5.3. Caso de Estudio

En nuestro estudio comparativo, evaluaremos un sistema de inscripciones vía Internet (SIV) para una Universidad. Se trata de un Sistema de Inscripciones vía Internet para una Universidad, en donde existen tres vistas principales: (a) InvestigadoresProfesores, (b) Alumnos, (c) Público en general. Dentro de la vista de Investigadores tenemos un usuario especial que es el Coordinador Académico, el cual también podría considerarse como una vista. El acceso al sistema es con clave de entrada 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 inscripciones a cursos, información de alumnos, cursos e investigadores que imparten los cursos, así como módulos para hacer cambios dentro de la información general.

La información que se muestra en el sistema depende del perfil de usuario que lo utilizautiliza. De esta forma, el publico en general solo puede consultar los cursos que se imparten en la universidad, mientras que los Alumnos además pueden inscribirse y actualizar sus datos personales. En el caso

del perfil de Investigadores, estos pueden dar de alta los cursos que van a impartir, actualizarlos, ver la lista de alumnos inscritos a su curso, los cursos de los alumnos que asesoran y también pueden actualizar su información personal. El caso del Coordinador Académico es general, ya que puede acceder a toda la información del sistema además de contar con la capacidad de establecer el periodo de inscripciones y dar de alta nuevos usuarios en el sistema (alumnos, profesores, etcétera).

5.4. Diseño de la Cadena de Markov.

5.4.1. Estimación de las probabilidades de entrada de la cadena de Markov

La Cadena General que corresponde al SIV la hemos desglosado en 4 cadenas, cada una de estas cadenas corresponde a cada uno de los perfiles operacionales del sistema y sus operaciones. Los perfiles operacionales y las operaciones del sistema son las variables que se van a analizar:

- Los perfiles operacionales son: *Público en General (PG), Alumnos (Al), Investigadores (Inv) y el Coordinador Académico (Coord).*
- Las operaciones que el sistema realiza son:*Acceso, Consulta de cursos (ConsultaCur), Consulta de Investigadores (ConsultaInv), Consulta de alumnos (ConsultaAlum), consulta de los cursos donde están inscritos los alumnos (Consulta CursosXAlumno), las inscripciones (Alta CursosXAlumno), las bajas (Bajas CursosXAlumno), los cambios (Cambios CursosXAlumno), el alta de la información general de los cursos, investigadores y alumnos (Alta Inf), los cambios (Cambios Inf), alta de los usuarios en el sistema (Altausuarios), el establecimiento del periodo de Inscripción (Periodo Inscripción, el alta del periodo (Alta Periodo), los cambios (Cambios Periodo) y la Baja (Baja Periodo)* y la Salida del sistema (Salida)}.

Para establecer estas relaciones hemos verificado la especificación de los requerimientos descritos en el apéndice A. En la Figura 5.1, se describe el perfil operacional que corresponde al Público en General (Requerimiento RB8 de acuerdo al Apendice A2). Esta vista sólo puede ver la información de los cursos y de los investigadores que imparten estos cursos. El espacio de estados de esta vista es el siguiente:$\{(Perfil = PG, Operacion = ConsultaInv)$,$(Perfil = PG, Operacion = ConsultaCur)\}$.

En la Figura 5.2, se describe el perfil operacional que corresponde al los alumnos (Requerimientos RB1,RB2,RB3,RB4, RB5,RB6,RB7 y RB9 de acuerdo al apendice A2). En esta vista se

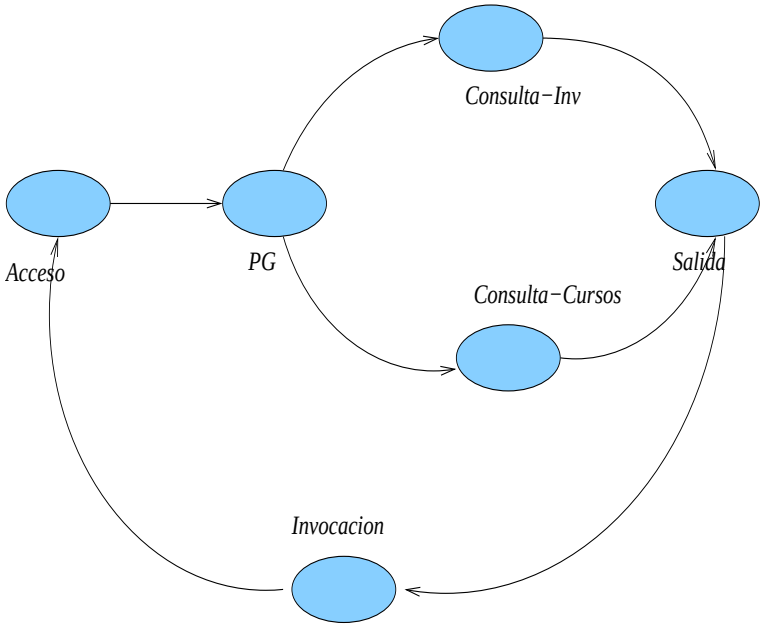

Figura 5.1: Cadena de Markov para el perfil PG.

puede ver la información de los cursos y de los investigadores que imparten estos cursos. También puede verse su información personal y permite inscribirse a los cursos. El espacio de estados de esta vista es el siguiente: $\{(Perfil = Alumno, Operacion = ConsultaInv)$,$(Perfil = PG, Operacion = ConsultaCur)$, $(Perfil = Alumno, Operacion = Consultaalumno)$,$(Perfil = Alumno, Operacion = ConsultaCursosXAlumnos)\}$.

Las operaciones asociadas a la Consulta de CursosXAlumnos son:$\{(Operacion = ConsultaCursosXAlumnos, Operacion = AltaCursosXAlumno)$, $(Operacion = ConsultaCursosXAlumnos, Operacion$ $BajaCursosXAlumno)$,$(Operacion = ConsultaCursosXAlumnos, Operacion = CambiosCursosXAlumno)$,$(O$ $ConsultaCursosXAlumnos, Operacion = ConsultaCursosXAlumno)\}$.

En la Figura 5.3, se representa el perfil operacional que corresponde al los investigadores (Requerimientos RB5, RB9, RB11, y RB15 del apendice A2). En esta vista se puede ver la información de los cursos, la información de los investigadores, hacer modificaciones en la informacion, asi como ver la informacion de los alumnos que asesora y hacer cambios en sus inscripciones. El espacio de estados de esta vista es el siguiente: $\{(Perfil = Inv, Operacion = ConsultaInv)$, $(Perfil = Inv, Operacion = ConsultaCur)$, $(Perfil = Inv, Operacion = ConsultaAlumno)\}$.

Las operaciones asociadas a la Consulta de CursosXAlumnos son:$\{(Operacion = ConsultaCursosXAlumnos, Operacion = AltaCursosXAlumno)$, $(Operacion = ConsultaCursosXAlumnos, Operacion$

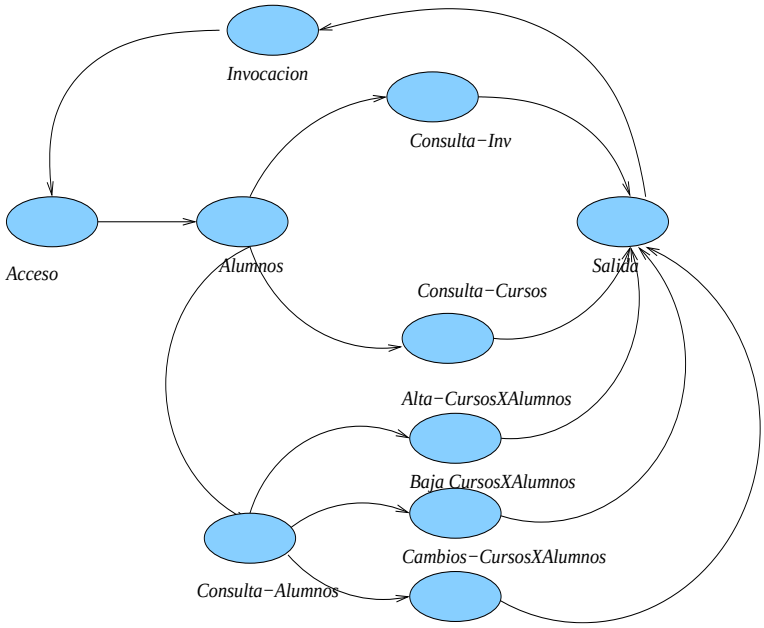

Figura 5.2: Cadena de Markov para el perfil Alumnos.

$BajaCursosXAlumno$),($Operacion = ConsultaCursosXAlumnos, Operacion = CambiosCursosXAlumno$), $ConsultaCursosXAlumnos, Operacion = ConsultaCursosXAlumno$)}.

Las operaciones asociadas a la Consulta de Cursos son:{($Operacion = ConsultaCursos(catalogo), Operacion$ $AltaInf$), ($Operacion = ConsultaCursos, Operacion = BajaInf$),($Operacion = ConsultaCursos, Operacion$ $CambiosInf$),($Operacion = ConsultaCursos, Operacion = ConsultaInf$)}.

Las operaciones asociadas a la Consulta de Investigadores son:{($Operacion = ConsultaInv(catalogo), Opera$ $AltaInf$), ($Operacion = ConsultaInv, Operacion = BajaInf$),($Operacion = ConsultaInv, Operacion =$ $CambiosInf$),($Operacion = ConsultaInv, Operacion = ConsultaInf$)}.

En la Figura 5.4, se describe el perfil operacional que corresponde al coordinador académico (Requerimientos RB4,RB5,RB6,RB7,RB9,RB12 y RB13 del apendice A2). En esta vista se puede ver la información de los cursos, la información de los investigadores, hacer modificaciones en la informaciń, puede ver la informacion de los alumnos, hacer cambios en sus inscripciones y además establece periodo de inscripciones (PeriodoInsc) y el alta de usuarios en el sistema. Así el espacio de estados asociado a esta vista es el siguiente: {($Perfil = Coord, Operacion = ConsultaInv$),($Perfil =$ $Coord, Operacion = ConsultaCur$), ($Perfil = Coor, Operacion = ConsultaAlumno$),($Perfil =$ $Coor, Operacion = PeriodoInsc$), ($Perfil = Coor, Operacion = Altausuarios$)}.

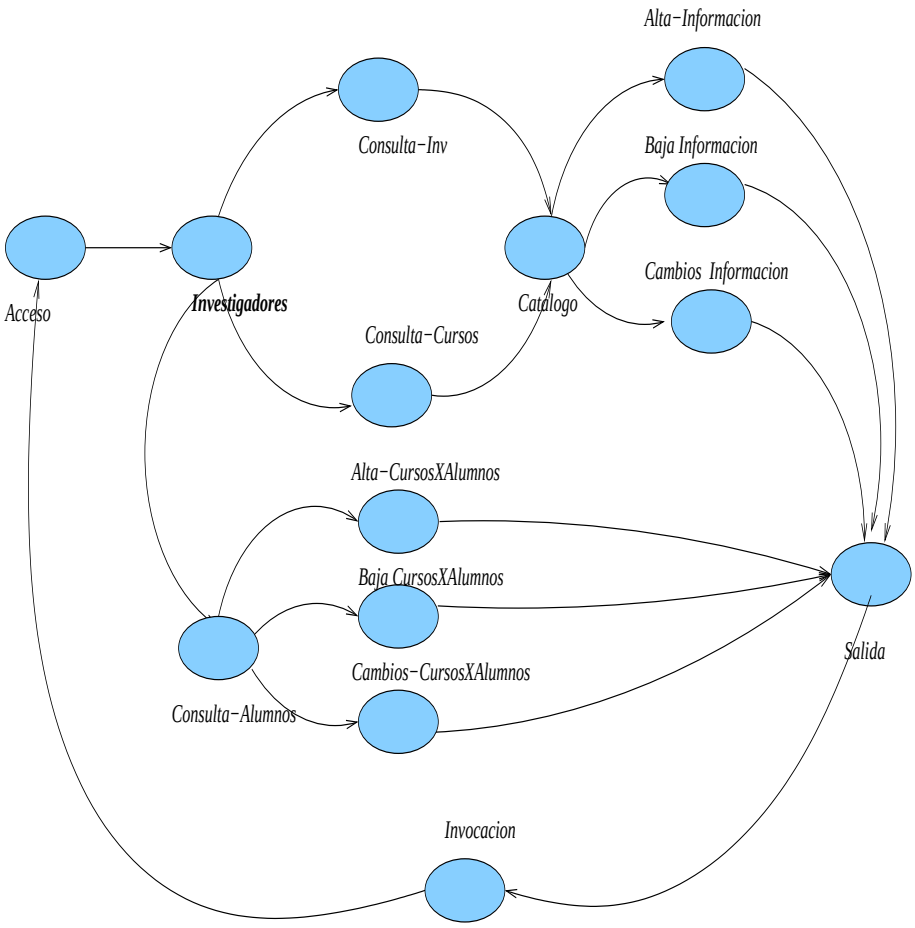

Figura 5.3: Cadena de Markov para el perfil Investigador.

Las operaciones asociadas son las mismas que las del perfil del investigador, además de las siguientes operaciones: $\{(Operacion = PeridoInsc, Operacion = AltaPeriodo)$,$(Operacion = PeriodoInsc, Operacion = BajaPeriodo)$, $(Operacion = PeriodoInsc, Operacion = CambiosPeriodo)$,$(Operacion = AltaUsuarios, Operacion = AltaUsuarios)$ $(Operacion = AltaPeriodo, Operacion = ConsultaCur)$, $(Operacion = CambiosPeriodo, Operacion = ConsultaCur)$, $(Operacion = BajaPeriodo, Operacion = ConsultaCur(catalogo)$ $\}$.

La matriz de probabilidades estacionarias para esta Cadena se encuentra representada en las Tablas 5.5, 5.6, 5.7,5.8 y 5.9. Los valores asentados en estas tablas se obtuvieron de los promedios que se hicieron con los resultados de 5000 simulaciones.

Para estimar las probabilidades de entrada de la Cadena de Markov realizamos 5000 simulaciones. En cada estado de la cadena se obtuvieron las estadísticas de la frecuencia con la cual cada uno de los perfiles operacionales ejecutó cada una de las secciones de código y cada uno de los

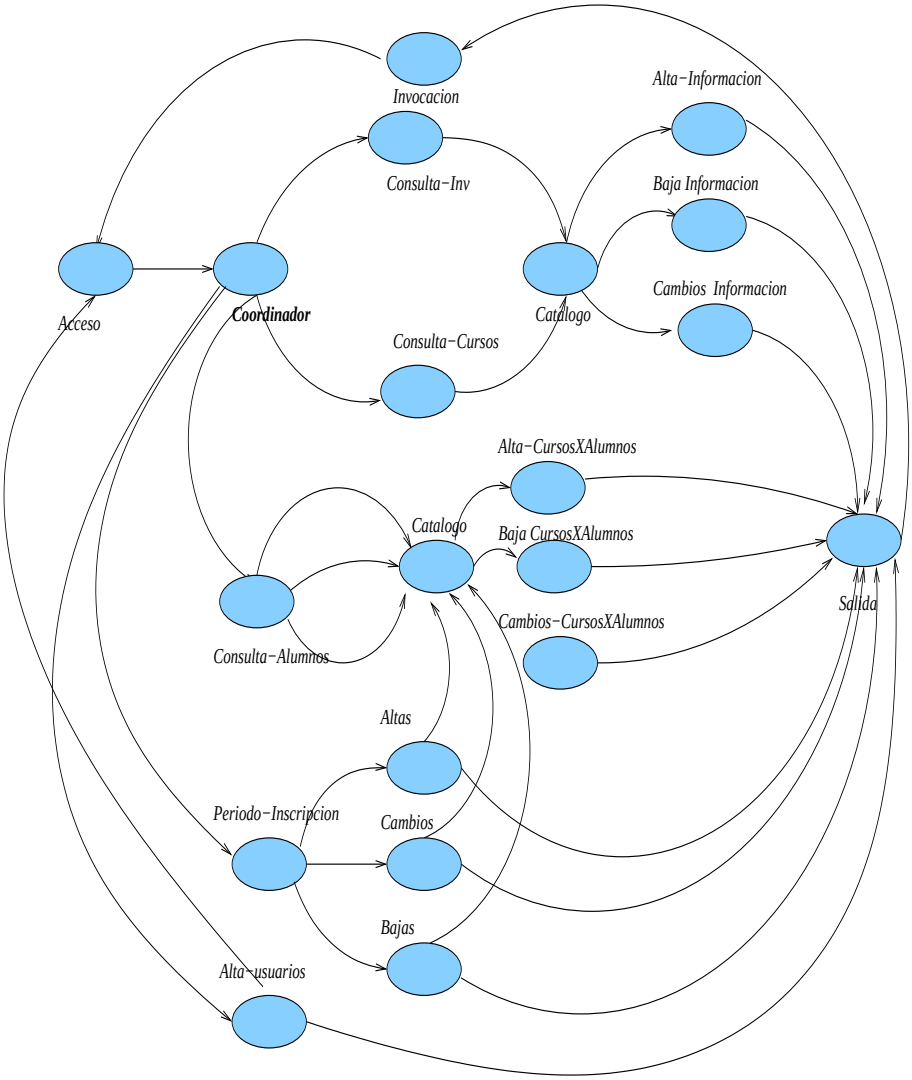

Figura 5.4: Cadena de Markov para el Perfil Coordinador.

componentes durante la simulación. El diagrama a bloques de la simulación se presenta en la Figura 5.5.

5.4.2. Generación de las secuencias de la cadena de Markov.

En las pruebas estadísticas del sistema de software los eventos de interés son secuencias de los estímulos (entradas) que representan una ejecución (comportamiento) del software. Estas secuencias constituyen el espacio de eventos del sistema de software y son obtenidas siguiendo el orden de estos puntos en la muestra (Cadena de Markov). Esto es, las secuencias que representan los atributos importantes del experimento aleatorio, son los casos de pruebas del sistema de software. De esta

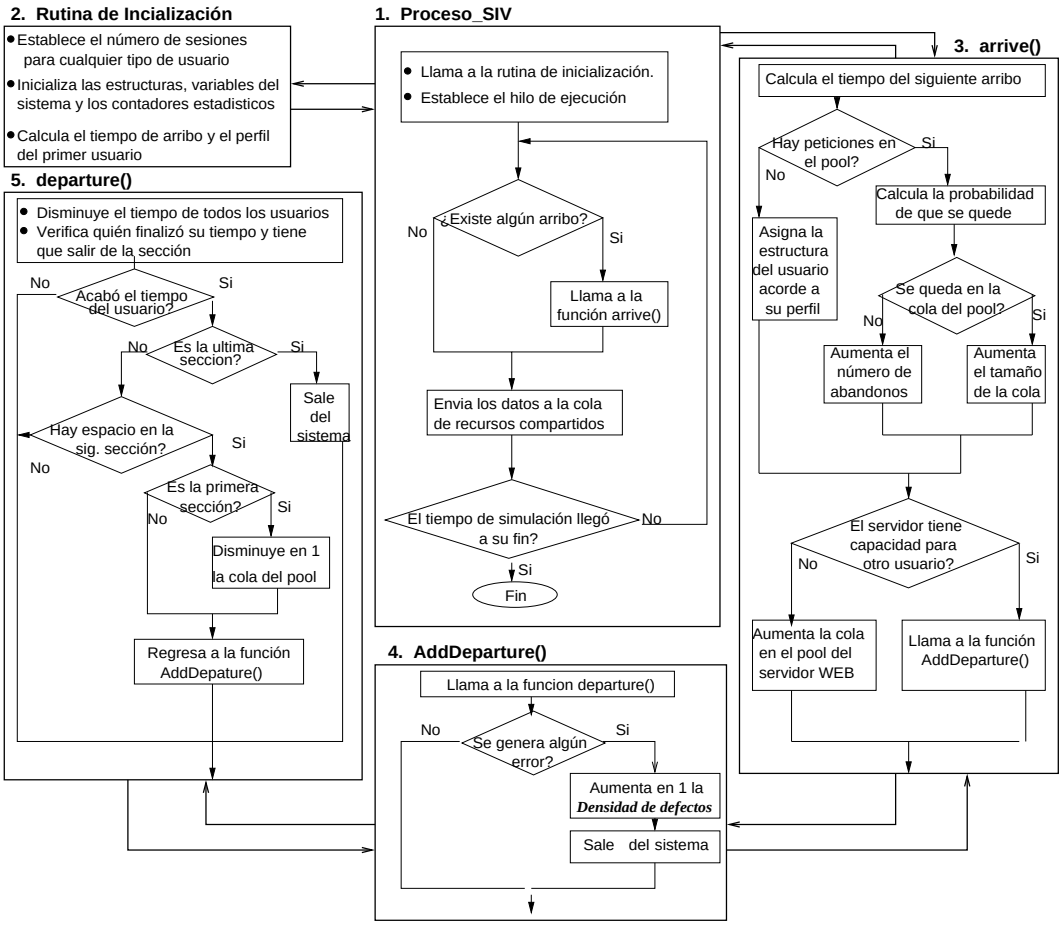

Figura 5.5: Diagrama a bloques de la simulación.

forma, cada secuencia representa una ruta representada por la cadena usada desde el estado inicial (sin invocar) hasta el estado terminal. Un conjunto de estructuras de secuencias completas establece la frecuencia en cada arco [100] .

En nuestro caso, de acuerdo al diseño de la Cadena de Markov y a los resultados de las probabilidades estacionarias que obtuvimos en las simulaciones, tenemos las siguientes secuencias:

1. $< Invocacion >< Acceso >< PG, Alumno, Investigador, Coordinador >< Consulta - Curso >$
2. $< Invocacion >< Acceso >< PG, Alumno, Investigador, Coordinador >< Consulta - Investigador >$
3. $< Invocacion < Acceso >< Alumno, Investigador, Coordinador >< Consulta - Alumno >< Consulta - CursoX Alumno >$
4. $< Invocacion >< Acceso >< Alumno, Investigador, Coordinador >< Consulta - Alumno >< Consulta - CursoX Alumno >< Alta - CursosxAlumno >$

5. $< Invocacion >< Acceso >< Alumno, Investigador, Coordinador >< Consulta - Alumno >< Consulta - CursoXAlumno >< Baja - CursosxAlumno >$

6. $< Invocacion >< Acceso >< Investigador, Coordinador >< Consulta - Curso >< Altas - Informacion >$

7. $< Invocacion >< Acceso >< Investigador, Coordinador >< Consulta - Curso >< Bajas - Informacion >$

8. $< Invocacion >< Acceso >< Investigador, Coordinador >< Consulta - Curso >< Cambios - Informacion >$

9. $< Invocacion >< Acceso >< Investigador, Coordinador >< Consulta - Investigador >< Altas - Informacion >$

10. $< Invocacion >< Acceso >< Investigador, Coordinador >< Consulta - Investigador >< Bajas - Informacion >$

11. $< Invocacion >< Acceso >< Investigador, Coordinador >< Consulta-Investigador >< Cambios-Informacion >$

12. $< Invocacion >< Acceso >< Coordinador >< Alta - usuarios >$

13. $< Invocacion >< Acceso >< Coordinador >< Periodo - Insc >< Alta - Periodo >$

14. $< Invocacion >< Acceso >< Coordinador >< Periodo - Insc >< Bajas - Periodo >$

15. $< Invocacion >< Acceso >< Coordinador >< Periodo - Insc >< Cambios - Periodo >$

5.5. Análisis de los estados estables.

En este punto es necesario determinar el vector de soluciones de probabilidades estables. Para resolver este problema, es necesario desarrollar el sistema de ecuaciones lineales que representa el vector de soluciones π, el cual en nuestro caso cuenta con 21 variables. El método de Gauss fue utilizado para resolver el sistema de ecuaciones.

5.5.1. Representacion de la cadena de Markov como una matrix de estados U

. Del apéndice B, es posible obtener la matriz de estados U, que deriva de la cadena de Markov del SIV.

La matriz U es la siguiente:

$x0$	$x1$	$x2$	$x3$	$x4$	$x5$	$x6$	$x7$	$x8$	$x9$	$x10$	$x11$	$x12$	$x13$	$x14$	$x15$	$x16$	$x17$	$x18$	$x19$	$x20$	b
0,5	0,5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0,5	0,04	0	0	0,38	0	0	0	0	0	0,04	0	0	0	0,04	0	0	0	0	0	0
0	0	0,40	0,4	0,2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0,04	0,5	0	0,38	0	0	0	0	0	0,04	0	0	0	0,04	0	0	0	0	0	0,1
0	0	0,02	0	0,5	0,4	0	0	0	0	0	0,04	0,1	0,1	0,1	0,04	0	0	0	0	0	0,1
0	0	0	0,2	0,2	0,20	0,2	0,2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0,24	0,24	0,24	0	0	0	0,03	0	0	0	0,03	0	0	0	0	0	0,24
0	0	0	0	0	0,26	0,26	0,26	0,06	0,06	0,06	0,03	0	0	0	0,03	0	0	0	0	0	0
0	0	0	0	0	0,23	0,23	0,06	0,23	0,06	0,06	0,03	0	0	0	0,03	0	0	0	0	0	0,06
0	0	0	0	0	0,18	0,18	0,25	0,07	0,18	0,07	0,04	0	0	0	0,04	0	0	0	0	0	0
0	0	0	0	0	0,12	0,12	0,22	0,1	0,1	0,12	0,05	0	0	0	0,05	0	0	0	0	0	0,12
0	0	0	0,11	0,11	0	0,11	0,11	0	0	0	0,11	0,11	0,11	0,11	0	0	0	0	0	0	0,11
0	0	0	0,11	0,09	0	0,11	0	0	0	0	0,11	0,2	0,11	0,11	0,09	0	0	0	0	0	0,09
0	0	0	0,12	0,06	0	0,12	0	0	0	0	0,12	0,12	0,18	0,12	0,06	0	0	0	0	0	0,12
0	0	0	0,04	0,04	0	0,14	0	0	0	0	0,04	0,14	0,14	0,18	0,14	0	0	0	0	0	0,14
0	0	0	0,14	0,14	0	0,15	0,15	0	0	0	0	0	0	0	0,15	0,15	0,12	0	0	0	0
0	0,16	0	0	0,17	0	0,16	0	0	0	0	0	0	0	0	0,17	0,17	0	0	0	0	0,17
0	0	0	0	0	0	0	0,24	0	0	0	0	0	0	0	0,19	0	0,22	0,19	0,13	0,06	0
0	0	0	0	0	0	0	0,23	0	0	0	0	0	0	0	0,18	0	0	0,18	0	0	0,18
0	0	0	0	0	0	0	0,33	0	0	0	0	0	0	0	0,22	0	0	0	0,22	0	0,22
0	0	0	0	0	0	0	0,57	0	0	0	0	0	0	0	0,14	0	0	0	0	0,14	0,14
0,5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,5

5.5.2. Solución de los sistemas de ecuaciones lineales de la matrix de estados U

. Para resolver el sistema de ecuaciones lineales que deriva de la matriz de estados U, y que tiene 21 variables empleamos un sistema de resolución de ecuaciones lineales.

La funcionalidad del sistema tiene las siguientes caraterísticas:

- El sistema debe resolver un conjunto de ecuaciones lineales (ecuaciones de primer grado) con n incógnitas.
- Es necesario conocer los valores de la matriz **H** y los valores de **B**.
- El método de solución para la resolución de este sistema es el de Gauss.
- La captura de información la realizamos mediante su interfáz gráfica.
- El sistema se realizó en lenguaje Java.

De la solución de este sistema se obtuvieron los valores del vector π_i.

En la Figuras 5.6 se muestra la matriz de entrada del sistema que representa el sistema de ecuaciones lineales que estamos evaluando. Los resultados de la evaluación de esta matriz se presentan en la Figura 5.7.

Figura 5.6: Interfaz de entrada de datos para el sistema de ecuaciones lineales.

5.5.3. Determinación del vector de soluciones estacionarias π

. En la Tabla 5.1 se representa el estado i que ha sido evaluado, su solución estacionaria π_i, el número de estados x_i necesarios para llegar al estado i y finálmente el número esperado de secuencias s de ejecución para el estado en cuestión. Estos valores han sido determinados de acuerdo al enfoque de Whittaker.

5.5.4. Desarrollo de las funciones de cobertura del caso de estudio SIV, de acuerdo al vector de soluciones π_i.

Para la cobertura de evaluación Wittaker utiliza las ecuaciones que se presentan en la Tabla C.3. Es importante señalar que estas funciones se resuelven mediante métodos númericos o mediante

State	π_i	$xi = frac1\pi$	$s = fracx_i/x_term$
1. Invocación	0.988	1.012145749	0.89
2. Acceso	0.011	90.90909091	79.55
3. PG	0.132	7.575757576	6.63
4. ConsultaCur	0.709	1.410437236	1.23
5. ConsultaInv	0.316	3.164556962	2.77
6. Alumno	0.146	6.849315068	5.99
7. ConsultaAlum	0.775	1.290322581	1.13
8. Consulta CursoXAlumno	0.047	21.27659574	18.62
9. AltaCursosxAlumno	0.102	9.803921569	8.58
10. BajaCursosxAlumno	0.545	1.834862385	1.61
11.CambiosCursosxAlumno	0.198	5.050505051	4.42
12. Investigador	0.729	1.371742112	1.20
13. AltaInf	0.29	3.448275862	3.02
14. BajaInf	0.25	4	3.50
15. CambiosInf	0.505	1.98019802	1.73
16. Coordinador	0.305	3.278688525	2.87
17. Altausuarios	0.432	2.314814815	2.03
18. PeriodoInsc	0.947	1.055966209	0.92
19. AltaPeriodo	0.743	1.34589502	1.18
20. BajaPeriod	0.753	1.328021248	1.16
21. CambiosPeriod	0.875	1.142857143	1.00
22. Salida	0.011	90.90909091	79.55

Tabla 5.1: Resultados para el vector de soluciones π

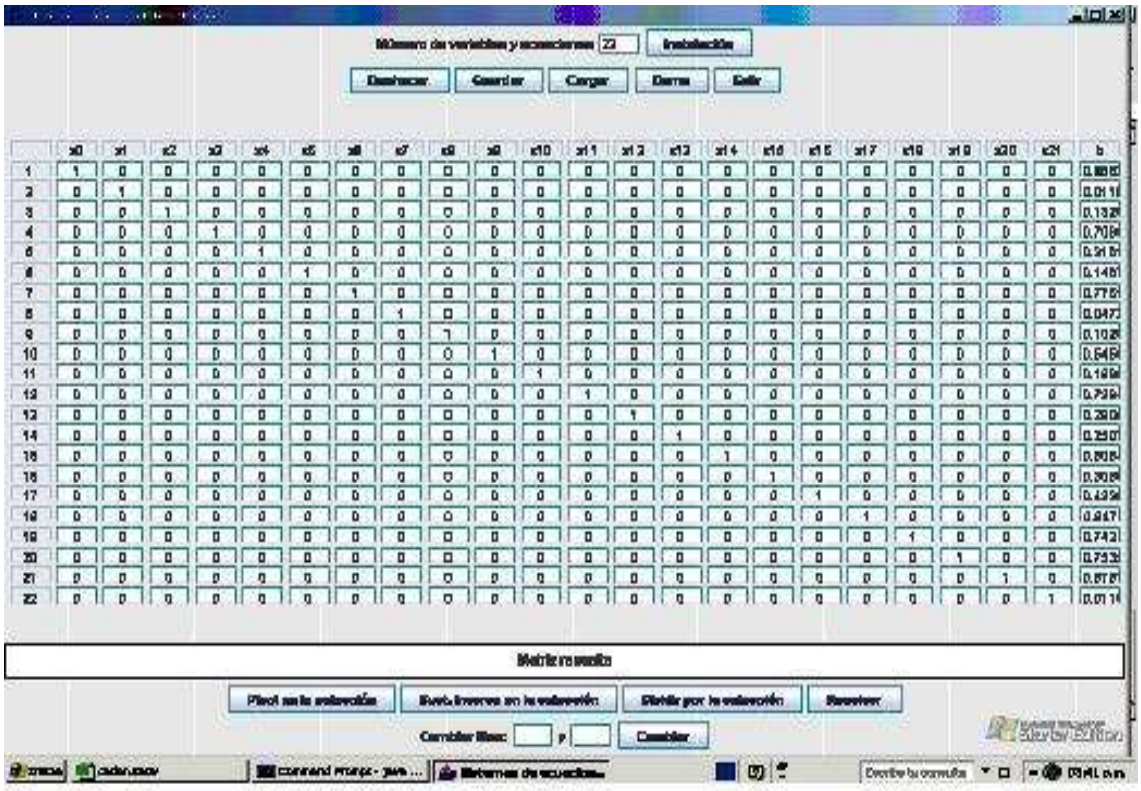

Figura 5.7: Resultados para el sistema de ecuaciones lineales evaluado.

la solución de sistemas de ecuaciones lineales (para los casos de determinación de los *tiempos de primera pasada*).

En la Tabla de resultados 5.2 se muestran los valores determinados para la *Distribución Estacionaria*, el *Tiempo de recurrencia* y el *Número de ocurrencias*. Estos valores han sido determinados mediante el desarrollo de algoritmos.

Los tiempos de primera pasada se determinaron resolviendo el sistema de ecuaciones que resultó de la expresión matemática que fue considerada en el trabajo de Whittaker y Thomason.

Tenemos que $m_{ij}(A)$ representa el tiempo esperado para alcanzar el subconjunto de estados A desde el estado i [99, 102]. La ecuacion que representa el comportamiento mas adecuado para determinar este tiempo es la siguiente de acuerdo a la tabla C.3.

Distribución Estacionaria	**Tiempo de recurrencia**	**Número de ocurrencias** del estado i al estado j
$\pi_j = \sum_i \pi_i U_{ij}$	$m_{jj} = \frac{1}{\pi_j}$	$m_{i,j}\pi_i = frac\pi_i\pi_j$
$\pi_0 = 1{,}0$	$m_{0,0} = 1{,}0$	$m_{0,1}\pi_0 = 2{,}0$
$\pi_1 = 0{,}5$	$m_{1,1} = 2{,}0$	$m_{1,2,}\pi_1 = 25{,}0$
$\pi_2 = 0{,}02$	$m_{2,2} = 50{,}0$	$m_{2,3,}\pi_2 = 2{,}5$
$\pi_3 = 0{,}0080$	$m_{3,3} = 125{,}0$	$m_{3,4,}\pi_3 = 2{,}0$
$\pi_4 = 0{,}0040$	$m_{4,4} = 250{,}0$	$m_{4,5}\pi_4 = 0{,}02$
$\pi_5 = 0{,}195$	$m_{5,5} = 5{,}14$	$m_{5,6}\pi_5 = 5{,}0$
$\pi_6 = 0{,}039$	$m_{6,6} = 25{,}69$	$m_{6,7}\pi_6 = 0{,}81$
$\pi_7 = 0{,}048$	$m_{7,7} = 20{,}72$	$m_{7,8}\pi_7 = 16{,}67$
$\pi_8 = 0{,}0030$	$m_{8,8} = 345{,}27$	$m_{8,9}\pi_8 = 0{,}94$
$\pi_9 = 0{,}0030$	$m_{9,9} = 325{,}73$	$m_{9,10}\pi_9 = 0{,}93$
$\pi_{10} = 0{,}0030$	$m_{10,10} = 304{,}42$	$m_{10,11}\pi_{10} = 0{,}14$
$\pi_{11} = 0{,}023$	$m_{11,11} = 42{,}61$	$m_{11,12}\pi_{11} = 7{,}87$
$\pi_{12} = 0{,}0030$	$m_{12,12} = 335{,}38$	$m_{12,13}\pi_{12} = 0{,}9$
$\pi_{13} = 0{,}0030$	$m_{13,13} = 302{,}14$	$m_{13,14}\pi_{13} = 0{,}89$
$\pi_{14} = 0{,}0040$	$m_{14,14} = 269{,}77$	$m_{14,15}\pi_{14} = 0{,}15$
$\pi_{15} = 0{,}024$	$m_{15,15} = 40{,}89$	$m_{15,16}\pi_{15} = 6{,}67$
$\pi_{16} = 0{,}0040$	$m_{16,16} = 272{,}6$	$m_{16,17}\pi_{16} = 1{,}25$
$\pi_{17} = 0{,}0030$	$m_{17,17} = 340{,}75$	$m_{17,18}\pi_{17} = 5{,}26$
$\pi_{18} = 0{,}0010$	$m_{18,18} = 1793{,}43$	$m_{18,19}\pi_{18} = 1{,}46$
$\pi_{19} = 0{,}0$	$m_{19,19} = 2621{,}16$	$m_{19,20}\pi_{19} = 2{,}17$
$\pi_{20} = 0{,}0$	$m_{20,20} = 5679{,}18$	$m_{20,21}\pi_{20} = 0{,}01$
$\pi_{21} = 0{,}016$	$m_{21,21} = 63{,}66$	$m_{20,21}\pi_{20} = 0{,}01$

Tabla 5.2: Resultados para determinar los parametros de cobertura de la Cadena de Markov del SIV

$$m_{ij}(A) = 1 + \sum U_{ik} m_{kj} \tag{5.1}$$

Los *estados transitorios* encontrados se representan mediante las siguientes ecuaciones:

$m_{12} = 1 + U_{11} m_{12}$

$m_{23} = 1 + U_{21} m_{12} + U_{22} m_{23}$

$m_{34} = 1 + U_{31} m_{12} + U_{32} m_{23} + U_{33} m_{34}$

$m_{45} = 1 + U_{41} m_{12} + U_{42} m_{23} + U_{43} m_{34} + U_{44} m_{45}$

$m_{56} = 1 + U_{51} m_{12} + U_{52} m_{23} + U_{53} m_{34} + U_{54} m_{45} + U_{55} m_{56}$

$m_{67} = 1 + U_{61} m_{12} + U_{62} m_{23} + U_{63} m_{34} + U_{64} m_{45} + U_{65} m_{56} + U_{66} m_{65}$

$m_{78} = 1 + U_{71} m_{12} + U_{72} m_{23} + U_{73} m_{34} + U_{74} m_{45} + U_{75} m_{56} + U_{76} m_{67} + U_{77} m_{78}$

$m_{89} = 1 + U_{81} m_{12} + U_{82} m_{23} + U_{83} m_{34} + U_{84} m_{45} + U_{85} m_{56} + U_{86} m_{67} + U_{87} m_{78} + U_{88} m_{89}$

$m_{9,10} = 1 + U_{91} m_{12} + U_{92} m_{23} + U_{93} m_{34} + U_{94} m_{45} + U_{95} m_{56} + U_{96} m_{67} + U_{97} m_{78} + U_{98} m_{89} + U_{99} m_{9,10}$

$m_{10,11} = 1 + U_{10,1} m_{12} + U_{10,2} m_{2,3} + U_{10,3} m_{34} + U_{10,4} m_{45} + U_{10,5} m_{56} + U_{10,6} m_{67} + U_{10,7} m_{78} + U_{10,8} m_{89} + U_{10,9} m_{910} + U_{10,10} m_{10,11}$

$m_{11,12} = 1 + U_{11,1} m_{12} + U_{11,2} m23 + U_{11,3} m34 + U_{11,4} m45 + U_{11,5} m56 + U_{11,6} m67 + U_{11,7} m78 + U_{11,8} m89 + U_{11,9} m910 + U_{11,10} m1011 + U_{11,11} m11, 12$

$m_{12,13} = 1 + U_{12,1} m_{12} + U_{12,2} m_{23} + U_{12,3} m_{34} + U_{12,4} m_{45} + U_{12,5} m_{56} + U_{12,6} m_{67} + U_{12,7} m_{78} + U_{12,8} m_{89} + U_{12,9} m_{910} + U_{12,10} m_{1011} + U_{12,11} m_{11,12} + U_{12,12} m_{12,13}$

$m_{13,14} = 1 + U_{13,1} m_{12} + U_{13,2} m_{23} + U_{13,3} m_{34} + U_{13,4} m_{45} + U_{13,5} m_{56} + U_{13,6} m_{67} + U_{13,7} m_{78} + U_{13,8} m_{89} + U_{13,9} m_{910} + U_{13,10} m_{10,11} + U_{13,11} m_{11,12} + U_{13,12} m_{12,13} + U_{13,13} m_{13,14}$

$m_{14,15} = 1 + U_{14,1} m_{12} + U_{14,2} m_{23} + U_{14,3} m_{34} + U_{14,4} m_{45} + U_{14,5} m_{56} + U_{14,6} m_{67} + U_{14,7} m_{78} + U_{14,8} m_{89} + U_{14,9} m_{9,10} + U_{14,10} m_{10,11} + U_{14,11} m_{11,12} + U_{14,12} m_{12,14} + U_{14,13} m_{13,14} + U_{14,14} m_{14,15}$

$m_{15,16} = 1 + U_{15,1} m_{12} + U_{15,2} m_{23} + U_{15,3} m_{34} + U_{15,4} m_{45} + U_{15,5} m_{56} + U_{15,6} m_{67} + U_{15,7} m_{78} + U_{15,8} m_{89} + U_{15,9} m_{910} + U_{15,10} m_{10,11} + U_{15,11} m_{11,12} + U_{15,12} m_{12,13} + U_{15,13} m_{13,14} + U_{15,14} m_{14,15} + U_{15,15} m_{15,16}$

$m_{16,17} = 1 + U_{16,1} m_{12} + U_{16,2} m_{23} + U_{16,3} m_{34} + U_{16,4} m_{45} + U_{16,5} m_{56} + U_{16,6} m_{67} + U_{16,7} m_{78} + U_{16,8} m_{89} + U_{16,9} m_{910} + U_{16,10} m_{10,11} + U_{16,11} m_{11,12} + U_{16,12} m_{12,14} + U_{16,13} m_{13,14} + U_{16,14} m_{14,15} + U_{16,15} m_{15,16} + U_{16,16} m_{16,17}$

$m_{17,18} = 1 + U_{17,1} m_{12} + U_{17,2} m_{23} + U_{17,3} m_{34} + U_{17,4} m_{45} + U_{17,5} m_{56} + U_{17,6} m_{67} + U_{17,7} m_{78} + U_{17,8} m_{89} + U_{17,9} m_{910} + U_{17,10} m_{10,11} + U_{17,11} m_{11,12} + U_{17,12} m_{12,14} + U_{17,13} m_{13,14} + U_{17,14} m_{14,15} + U_{17,15} m_{15,16} + U_{17,16} m_{16,17} + U_{17,17} m_{17,18}$

$m_{18,19} = 1 + U_{18,1}m_{12} + U_{18,2}m_{23} + U_{18,3}m_{34} + U_{18,4}m_{45} + U_{18,5}m_{56} + U_{18,6}m_{67} + U_{18,7}m_{78} + U_{18,8}m_{89} + U_{18,9}m_{910} + U_{18,10}m_{10,11} + U_{18,11}m_{11,12} + U_{18,12}m_{12,18} + U_{18,13}m_{13,14} + U_{18,14}m_{14,15} + U_{18,15}m_{15,16} + U_{18,16}m_{16,17} + U_{18,17}m_{17,18} + U_{18,18}m_{18,19}$

$m_{19,20} = 1 + U_{19,1}m_{12} + U_{19,2}m_{23} + U_{19,3}m_{34} + U_{19,4}m_{45} + U_{19,5}m_{56} + U_{19,6}m_{67} + U_{19,7}m_{78} + U_{19,8}m_{89} + U_{19,9}m_{910} + U_{19,10}m_{10,11} + U_{19,11}m_{11,12} + U_{19,12}m_{12,14} + U_{19,13}m_{13,14} + U_{19,14}m_{14,15} + U_{19,15}m_{15,16} + U_{19,16}m_{16,17} + U_{19,17}m_{17,18} + U_{19,18}m_{18,19} + U_{19,18}m_{18,19} + U_{19,19}m_{19,20}$

$m_{20,21} = 1 + U_{20,1}m_{12} + U_{20,2}m_{23} + U_{20,3}m_{34} + U_{20,4}m_{45} + U_{20,5}m_{56} + U_{20,6}m_{67} + U_{20,7}m_{78} + U_{20,8}m_{89} + U_{20,9}m_{910} + U_{20,10}m_{10,11} + U_{20,11}m_{11,12} + U_{20,12}m_{12,14} + U_{20,13}m_{13,14} + U_{20,14}m_{14,15} + U_{20,15}m_{15,16} + U_{20,16}m_{16,17} + U_{20,17}m_{17,18} + U_{20,18}m_{18,19} + U_{20,18}m_{18,19} + U_{20,19}m_{19,20} + U_{20,20}m_{20,21}$

$m_{21,22} = 1 + U_{21,1}m_{12} + U_{21,2}m_{23} + U_{21,3}m_{34} + U_{21,4}m_{45} + U_{21,5}m_{56} + U_{21,6}m_{67} + U_{21,7}m_{78} + U_{21,8}m_{89} + U_{21,9}m_{910} + U_{21,10}m_{10,11} + U_{21,11}m_{11,12} + U_{21,12}m_{12,14} + U_{21,13}m_{13,14} + U_{21,14}m_{14,15} + U_{21,15}m_{15,16} + U_{21,16}m_{16,17} + U_{21,17}m_{17,18} + U_{21,18}m_{18,19} + U_{21,18}m_{18,19} + U_{21,19}m_{19,20} + U_{21,20}m_{20,21} + U_{21,21}m_{21,22}$

$m_{22,1} = 1 + U_{22,1}m_{12} + U_{22,2}m_{23} + U_{22,3}m_{34} + U_{22,4}m_{45} + U_{22,5}m_{56} + U_{22,6}m_{67} + U_{22,7}m_{78} + U_{22,8}m_{89} + U_{22,9}m_{9,10} + U_{22,10}m_{10,11} + U_{22,11}m_{11,12} + U_{22,12}m_{12,14} + U_{22,13}m_{13,14} + U_{22,14}m_{14,15} + U_{22,15}m_{15,16} + U_{22,16}m_{16,17} + U_{22,17}m_{17,18} + U_{22,18}m_{18,19} + U_{22,18}m_{18,19} + U_{22,19}m_{19,20} + U_{22,20}m_{20,21} + U_{22,21}m_{21,22} + U_{22,22}m_{22,1}$

Resolviendo el sistema de ecuaciones, mediante la técnica de Gauss, tenemos los siguientes resultados:

$m_{12}(A) = 2$

$m_{23}(A) = 2$

$m_{34}(A) = 1{,}66$

$m_{45}(A) = 2{,}13$

$m_{56}(A) = 2{,}066$

$m_{67}(A) = 2{,}3$

$m_{78}(A) = 2{,}042$

$m_{89}(A) = 2{,}53$

$m_{9,10}(A) = 2{,}79$

$m_{10,11}(A) = 2{,}93$

$m_{11,12}(A) = 3{,}01$

$m_{12,13}(A) = 2{,}208$

$m_{13,14}(A) = 2{,}36$

$m_{14,15}(A) = 2{,}44$

$m_{15,16}(A) = 2{,}49$

$m_{16,17}(A) = 2{,}67$

$m_{17,18}(A) = 2{,}95$

$m_{18,19}(A) = 2{,}71$

$m_{19,20}(A) = 2{,}54$

$m_{20,21}(A) = 3{,}10$

$m_{21,22}(A) = 3{,}27$

$m_{22,1}(A) = 3$

5.6. Desarrollo de las evaluaciones del caso de estudio SIV de acuerdo al enfoque de Whittaker.

Para hacer la evaluaciones del sistema utilizamos la herramienta descrita en el Capítulo 4. Según Whittaker, el modelo que proponen es válido independientemente de cualquier herramienta ó ambiente de evaluación que sea utilizado. La única restricción que impone en el ambiente de evaluación es que la herramienta sea capáz de realizar las pruebas y de analizar los resultados de estas pruebas, para así establecer cuando y en donde se han producido fallos.

Nuestra herramienta fue desarrollada para cumplir con estos requisitos y además incluye un enfoque parecido al de Whittaker, ya que también evalua las rutas de navegación (secuencias).

Es importante mencionar que al realizar el proceso de pruebas, se utilizó el mismo framework bajo las mismas condiciones para ambos modelos. Sin embargo los parámetros de cobertura como: *Tiempo de Ocurrencia, el Número de ocurrencia y los Tiempos de Primera Pasada* cambian en el modelo estocástico, por lo cual desarrollamos un componente para controlar estos valores en el ambiente de las evaluaciones.

El componente que aplica la cobertura de evaluación del sistema de acuerdo al modelo estocástico tiene las siguientes características de funcionalidad:

- De acuerdo al modelo estocástico se determinan los arribos de los evaluadores virtuales en

base de los resultados de primera pasada (*first pasaje time*).

- Las *secuencias* de evaluación son determinadas en base de la Cadena de Markov.
- El tiempo de recurrencia de evaluación para cada estado (componente) se determina de acuerdo al *recurrente time*.
- El número mínimo de evaluaciones para cada uno de los estados se determina de acuerdo al numero de ocurrencia (*ocurrent number*).

De esta forma, la cobertura de evaluación se determina a partir de estos parámetros.

5.7. Análisis de resultados

Llevamos a cabo 2 experimentos para evaluar nuestro modelo y el modelo de Wittaker [99]. Los experimentos que llevamos a cabo permiten evaluar los modelos en base a dos parámetros que son el tiempo y el número de evaluadores virtuales. Ambos modelos prueban el mismo sistema (SIV), con el mismo framework de evaluación (TT). Sin embargo la cobertura, el tiempo, la ocurrencia y los paths ó secuencias de evaluación son determinados de acuerdo a cada modelo.

En esta evaluación, nuestra propuesta (*Escenarios de Riesgo*) y la propuesta de Whittaker (*MarkovChains*) se ejecutaron en 11865 ciclos de emulación. Este tiempo es el mínimo que la propuesta de Whittaker necesita para ejecutarse, de acuerdo a los resultados de cobertura del modelo estocástico. Esta es la primera diferencia que encontramos con nuestra propuesta, debido a que el tiempo que estimamos como mínimo para cubrir las pruebas de nuestro modelo es de 1000 ciclos de emulación. El tiempo para realizar la cobertura en el modelo estocástico es mayor porque cubre todas las secuencias del sistema de software y nuestro modelo cubre en primera instancia solo las rutas que estimamos como críticas, que en este caso, son aproximadamente la mitad de las rutas que cubre el modelo estocastico. Otro factor que influye es que el arribo de los evaluadores en nuestro modelo sigue una distribución exponencial y en el caso de modelo estocástico se determina de acuerdo a los parámetros de cobertura. Analizando esta situación es claro que los recursos y el esfuerzo empleados por el modelo de Wittaker para la fase de pruebas también se incrementan. Sin embargo, esto no garantiza que el número de fallos descubiertos sea mayor, ya que se puede dar el caso que las zonas críticas no se cubran adecuadamente porque halla que evaluar otras zonas del sistema de software.

Caracteristica	Escenarios de Riesgo	Markov Chains
Densidad de Defectos (No. Errores)	8635	191
MTTF (seg)	4.33	1
Num. Evaluador Virtual	2523	99
Componente anómalo	**DD**	**DD**
Bajas_alumn.php	225	0
Cambios.php	3	0
cat_cur_alum.php	453	0
Catalogo.php	455	0
Conecta.php	**1603**	**105**
Consultas.php	1133	**9**
cursos_alumn.php	226	0
Menú.php	**2272**	**36**
Menú_sig.php	**1631**	**36**
Procesa_alba.php	124	0
Procesa_alca2.php	223	0
Procesa_altas.php	**224**	**2**
Procesa_ambiente.php	224	0

Tabla 5.3: Tabla comparativa de resultados de los modelos de evaluación en función del tiempo.

El detalle de los fallos y las metricas de software que se han podido obtener son el resultado de la aplicacion de la herramienta de evaluación. En la Tabla 5.3 se presentan los resultados del primer estudio comparativo en función del tiempo. En la columna *Caracteristica* se describen las métricas de confiabilidad utilizadas, las cuales fueron la *Densidad de Defectos (DD) y la Media de Ocurrencia de Fallos (MTTF).* El número de evaluadores virtuales (*Num. Evaluador Virtual*) se obtiene de acuerdo a los resultados del modelo en cuestión. Finálmente para evaluar los componentes que tienen fallos determinamos la *Densidad de Defectos.*

De los resultados presentados en la Table 5.3 observamos que la cobertura de fallos que presenta nuestro modelo es mayor ya que obtiene un mayor numero de fallos. Es posible observar que en el mismo marco de tiempo, el número de evaluadores virtuales que se generan en nuestro modelo es mayor al de el modelo de Wittaker. Nuestra propuesta localiza un mayor número de fallos, y un mayor número de componentes anómalos, en comparación con los encontrados mediante el modelo estocástico de Whittaker [99]. Desde nuestro punto de vista estos resultados se presentan por las siguientes razones:

- La cobertura con el número de evaluadores no fué la adecuada para el modelo estocástico.
- En el modelo estocástico se evaluan todas las rutas aunque estas no sean críticas, lo cual hace al sistem,a muy lento y poco eficiente, mientras que nuestro modelo le da mas importancia a las rutas mas criticas en donde amplia la cobertura de las evaluaciones, por lo que localiza un mayor número de fallos.
- El modelo estocástico evalua al sistema siguiendo patrones en función del tiempo y no en base a las relaciones que guardan los distintos componentes del sistema, por lo cual le es dificil localizar componentes defectuosos.
- Nuestro modelo generó un mayor número de evaluadores virtuales, lo cual permite realizar un mayor numero de pruebas, lo cual a su vez podría ser la razón por la cual encontramos un mayor número de fallos.

Es importante observar que debido a que nuestro modelo generó un mayor número de observadores virtuales, esta pudiera ser la razón por la cual nuestro modelo obtuvo una ventaja respecto al modelo estocástico. A fin de no dar ventajas a nuestro modelo respecto del modelo estocástico, en nuestro segundo experimento consideramos el mismo número de **Evaluadores Virtuales** para ambos modelos. Para lograr esta misma proporción en ambos modelos fué necesario (de acuerdo a las restricciones del modelo) incrementar el tiempo de emulación del modelo estocástico 23 veces más ($11865 * 23$). De acuerdo a los resultados de la Tabla 5.4, podemos observar que el modelo estocástico localiza una densidad de defectos general **menor** que el enfoque de los *Escenarios de Riesgos*. Es importante resaltar que el número de componentes localizados en este experimento sigue siendo menor y de hecho el enfoque estocástico sólo localiza un componente más, a pesar de que el número de evaluadores virtuales es ligeramente mayor que nuestro modelo.

De acuerdo a estos resultados podemos concluir que nuestro modelo localiza un mayor número de fallos debido a que desde el principio se establecen las rutas críticas del sistema, lo que permite ampliar el rango de pruebas y encontrar los fallos que están latentes en estas rutas. Esta situación también mejora la efectividad de las pruebas y optimiza los recursos empleados para esta fase. Esta situación no sucede con el enfoque estocástico. El principal incoveniente con este modelo es que considera la cobertura de todo el sistema de software con variantes que no son muy significativas para ser evaluadas. Este proceso consume más tiempo y recursos de esta fase sin que exista la garantía de que los fallos se van a localizar con efectividad. Para mejorar la efectividad del modelo

Caracteristica	Escenarios de Riesgo	Cadenas de Markov
Densidad de Defectos (No. Errores)	8635	5790
MTTF (seg)	4.33	1
Num. Evaluador Virtual	2523	268
Componente anómalo	**DD**	**DD**
Bajas_alumn.php	225	0
cambios.php	3	0
cat_cur_alum.php	453	0
catalogo.php	**455**	**423**
Conecta.php	**1603**	**2539**
Consultas.php	**1133**	**423**
cursos_alumn.php	226	0
Menú.php	**2272**	**1130**
Menú_sig.php	**1631**	**1130**
Procesa_alba.php	124	0
Procesa_alca2.php	223	0
procesa_altas.php	**224**	**140**
procesa_ambiente.php	224	0

Tabla 5.4: Tabla comparativa de resultados de los modelos de evaluación en función del número de Evaluadores Virtuales.

estocástico se tendría que ampliar el rango de pruebas para todas las secuencias, lo cual haría que la complejidad y el tiempo de cómputo utilizado por el modelo crecieran de forma significativa.

Otra ventaja que encotramos con nuestro enfoque es que nuestro análisis se lleva a cabo mediante el estudio de la relación entre los componentes mientras que con el enfoque estocástico el enfoque está en función del tiempo. Sin embargo, es evidente que la relacion que guardan el tiempo y la localización de los fallos en los componentes del sistema no presenta una relacion directa y tampoco es muy clara.

5.8. Conclusiones

Después de aplicar nuestro modelo, podemos observar que sigue cumpliendo con criterios de calidad que los modelos eficientes deben tener [94]. En base a los resultados obtenidos en los experimentos podemos estimar que la *validéz predictiva* de nuestro modelo es adecuada ya que los resultados de nuestro modelo coinciden con los resultados que se obtienen en el modelo estocástico.

Además de esto, es claro que nuestro modelo localiza más fallos y componentes anómalos que el modelo estocástico.

Otro criterio de calidad con el que cumple nuestro modelo es la *simplicidad en su aplicación*. Al desarrollar el modelo modelo estocastico podemos observar que para determinar el los valores del vector de soluciones *pi* y los parámetros de cobertura del modelo estocástico se requiere de un conocimiento matemático amplio y además su solución demanda de una alta complejidad computacional la cual está en funciń al tamaño del espacio de estados que se analiza.

Otras ventajas que presenta nuestro son las siguientes:

- No se necesita de un gran conocimiento matemático para estimar el modelo basada en la teoría de grafos.
- Los valores de entrada y en general del proceso pueden automatizarce, sin que estos sistemas crezcan demasiado.
- La automatización conseguida mediante nuestra herramienta permite a nuestro modelo reducir a horas el trabajo que de forma manual necesita de meses para su desarrollo.

Estos dos criterios de acuerdo a los especialistas del área [94] (Iannino, Okumoto, y otros) son los más importantes. Esta situación se hace muy evidente cuando el modelo se trata de aplicar en sistemas mas complejos utilizados en la industria.

Otra ventaja que presenta nuestro modelo con respecto al modelo estocástico es la escalabilidad. Es claro que debido a que nuestro modelo genera trayectorias mediante grafos (y no mediante la generación de complejas cadenas de markov), esto nos permite hacer estudios de sistemas muy complejos y con un costo computacional relativamente bajo. Esta escalabilidad para el caso del modelo estocástico se dificulta enormemente debido a que al aumentar el espacio de estados del sistema, su estudio puede volverse demasiado complejo ó intratable.

En general nuestro modelo proporciona un marco eficiente e innovador para la localización efectiva de fallos y para análisis de la confiabilidad en los sistemas de software en ambientes operativos. Estas cualidades finálmente benefician a la etapa de pruebas debido a que con estos resultados es posible localizar y eliminar fallos con mayor certeza.

From State	Transition, Stimuli	To State	Frecuency	Probability
Invocación	Invocacion	Acceso	338.00	0.50
Invocación	Acceso	Acceso	338.00	0.50
Acceso	Acceso	Acceso	340	0.50
Acceso	Entrada	PG	29	0.04
Acceso	Entrada	Alumno	257	0.38
Acceso	Entrada	Investigador	27	0.04
Acceso	Entrada	Coordinador	27	0.04
			680.00	
PG	Consulta	PG	29.00	0.40
PG	Consulta	ConsultaCur	29.00	0.40
PG	Consulta	ConsultaInv	15.00	0.20
			73.00	
ConsultaCur	Consulta	PG	29.00	0.04
ConsultaCur	Consulta	ConsultaCur	343.00	0.50
ConsultaCur	Consulta	Alumno	259.00	0.38
ConsultaCur	Consulta	Investigador	28.00	0.04
ConsultaCur	Consulta	Coordinador	27.00	0.04
			686.00	
ConsultaInv	Consulta	PG	14.00	0.02
ConsultaInv	Alta	ConsultaCur	329.00	0.50
ConsultaInv	Baja	Alumno	260.00	0.40
ConsultaInv	Cambios	Investigador	27.00	0.04
ConsultaInv	Salida	Coordinador	27.00	0.04
			657.00	
Alumno	Consulta	ConsultaAlum	262.00	0.20
Alumno	Consulta	Consulta CursoXAlumno	263.00	0.20
Alumno	Consulta	Alumno	261.00	0.20
Alumno	Consulta	ConsultaCur	258.00	0.20
Alumno	Consulta	ConsultaInv	260.00	0.20
			1304.00	
ConsultaAlum	Consulta	Consulta CursoXAlumno	262.00	0.24
ConsultaAlum	Consulta	ConsultaAlum	262.00	0.24
ConsultaAlum	Consulta	Alumno	263.00	0.24

Tabla 5.5: Datos de entrada para la cadena de acuerdo a los promedios de las 5000 simulaciones (parte 1)

From State	Transition, Stimuli	To State	Frecuency	Probability
ConsultaAlum	Consulta	Investigador	28.00	0.03
ConsultaAlum	Consulta	Coordinador	28.00	0.03
ConsultaAlum	Consulta	Salida	262.00	0.24
			1105.00	
Consulta CursoXAlumno	Consulta	ConsultaAlum	263.00	0.26
Consulta CursoXAlumno	Consulta	Consulta CursoXAlumno	263.00	0.26
Consulta CursoXAlumno	Consulta	Alumno	263.00	0.26
Consulta CursoXAlumno	Consulta	Investigador	28.00	0.03
Consulta CursoXAlumno	Consulta	Coordinador	28.00	0.03
Consulta CursoXAlumno	Alta	AltaCursosxAlumno	56.00	0.06
Consulta CursoXAlumno	Baja	BajaCursosxAlumno	56.00	0.06
Consulta CursoXAlumno	Cambios	CambiosCursosxAlumno	56.00	0.06
			1013.00	
AltaCursosxAlumno	Alta	AltaCursosxAlumno	200.00	0.23
AltaCursosxAlumno	Consulta	Consulta CursoXAlumno	56.00	0.06
AltaCursosxAlumno	Consulta	Consulta Alumno	200.00	0.23
AltaCursosxAlumno	Acceso	Alumno	200.00	0.23
AltaCursosxAlumno	Bajas	BajaCursosxAlumno	56.00	0.06
AltaCursosxAlumno	Cambios	CambiosCursosxAlumno	56.00	0.06
AltaCursosxAlumno	Acceso	Investigador	28.00	0.03
AltaCursosxAlumno	Acceso	Coordinador	27.00	0.03
AltaCursosxAlumno	Salida	Salida	56.00	0.06
			879.00	
BajaCursosxAlumno	Baja	BajaCursosxAlumno	134.00	0.18
BajaCursosxAlumno	Cambios	CambiosCursosxAlumno	56.00	0.07
BajaCursosxAlumno	Alta	AltaCursosxAlumno	56.00	0.07
BajaCursosxAlumno	Consulta	Consulta CursoXAlumno	190.00	0.25
BajaCursosxAlumno	Acceso	Alumno	133.00	0.18
BajaCursosxAlumno	Acceso	Investigador	28.00	0.04
BajaCursosxAlumno	Acceso	Coordinador	28.00	0.04
BajaCursosxAlumno	Consulta	Consulta Alumno	134.00	0.18
			759.00	

Tabla 5.6: Datos de entrada para la cadena de acuerdo a los promedios de las 5000 simulaciones (parte 2)

From State	Transition, Stimuli	To State	Frecuency	Probability
CambiosCursosxAlum	Cambios	CambiosCursosxAlumno	67.00	0.12
CambiosCursosxAlum	Bajas	BajaCursosxAlumno	56.00	0.10
CambiosCursosxAlum	Altas	AltaCursosxAlumno	56.00	0.10
CambiosCursosxAlum	Consulta	Consulta CurXAlum	124.00	0.22
CambiosCurxAlum	Consulta	Consulta Alumno	67.00	0.12
CambiosCursosxAlum	Acceso	Alumno	67.00	0.12
CambiosCursosxAlum	Acceso	Investigador	29.00	0.05
CambiosCursosxAlum	Acceso	Coordinador	28.00	0.05
CambiosCursosxAlum	Salida	Salida	67.00	0.12
			561.00	
Investigador	Consulta	Consulta-Cur	28.00	0.11
Investigador	Consulta	ConsultaInv	28.00	0.11
Investigador	Consulta	ConsultaAlum	28.00	0.11
Investigador	Consulta	Consulta CursoXAlumno	28.00	0.11
Investigador	Acceso	Investigador	28.00	0.11
Investigador	Altas	AltaInf	28.00	0.11
Investigador	Bajas	BajaInf	28.00	0.11
Investigador	Cambios	CambiosInf	28.00	0.11
Investigador	Salida	Salida	28.00	0.11
			252.00	
AltaInf	Altas	Investigador	28.00	0.11
AltaInf	Altas	Coordinador	22.00	0.09
Alta-Inf	Consultas	Consulta-Cur	28.00	0.11
AltaInf	Consultas	Consulta-Inv	22.00	0.09
AltaInf	Consultas	Consulta-Alum	28.00	0.11
AltaInf	Altas	Alta-Inf	50.00	0.20
AltaInf	Bajas	Baja-Inf	28.00	0.11
AltaInf	Cambios	Cambios-Inf	28.00	0.11
AltaInf	Salida	Salida	22.00	0.09
			256.00	

Tabla 5.7: Datos de entrada para la cadena de acuerdo a los promedios de las 5000 simulaciones (parte 3)

From State	Transition, Stimuli	To State	Frecuency	Probability
BajaInf	Bajas	BajaInf	44.00	0.18
BajaInf	Acceso	Investigador	28.00	0.12
BajaInf	Acceso	Coordinador	14.00	0.06
BajaInf	Consulta	ConsultaInv	14.00	0.06
BajaInf	Consulta	ConsultaAlum	28.00	0.12
BajaInf	Consulta	ConsultaCur	28.00	0.12
BajaInf	Altas	AltaInf	28.00	0.12
BajaInf	Cambios	CambiosInf	28.00	0.12
BajaInf	Salida	Salida	28.00	0.12
			240.00	
CambiosInf	Cambios	CambiosInf	36.00	0.18
CambiosInf	Baja	BajaInf	28.00	0.14
CambiosInf	Altas	AltaInf	28.00	0.14
CambiosInf	Acceso	Investigador	7.00	0.04
CambiosInf	Acces	Coordinador	28.00	0.14
CambiosInf	Consulta	ConsultaInv	7.00	0.04
CambiosInf	Consulta	ConsultaAlum	28.00	0.14
CambiosInf	Consulta	ConsultaCur	7.00	0.04
CambiosInf	Salida	Salida	28.00	0.14
			197.00	
Coordinador	Acceso	Coordinador	29.00	0.15
Coordinador	Alta	Altausuarios	29.00	0.15
Coordinador	Consulta	PeriodoInsc	22.00	0.12
Coordinador	Consulta	ConsultaAlum	28.00	0.15
Coordinador	Consulta	Consulta CurXAlum	28.00	0.15
Coordinador	Consulta	ConsultaCur	27.00	0.14
Coordinador	Consulta	ConsultaInv	27.00	0.14
			190.00	

Tabla 5.8: Datos de entrada para la cadena de acuerdo a los promedios de las 5000 simulaciones (parte 4)

From State	Transition or Stimuli	To State	Frecuency	Probability
Altausuarios	Acceso	Acceso	27.00	0.16
Altausuarios	Consulta	ConsultaAlum	28.00	0.16
Altausuarios	Consulta	Coordinador	29.00	0.17
Altausuarios	Consulta	ConsultaInv	29.00	0.17
Altausuarios	Alta	Altausuarios	29.00	0.17
Altausuarios	Salida	Salida	29.00	0.17
			171.00	
PeriodoInsc	Consulta	PeriodoInsc	22.00	0.19
PeriodoInsc	Consulta	Consulta CursoXAlumno	28.00	0.24
PeriodoInsc	Alta	AltaPeriodo	22.00	0.19
PeriodoInsc	Baja	BajaPeriod	15.00	0.13
PeriodoInsc	Consulta	Coordinador	22.00	0.19
PeriodoInsc	Cambios	CambiosPeriod	7.00	0.06
			116.00	
AltaPeriodo	Consulta	ConsultaInv	27.00	0.22
AltaPeriodo	Consulta	Consulta CursoXAlumno	28.00	0.23
AltaPeriodo	Alta	AltaPeriodo	22.00	0.18
AltaPeriodo	Acceso	Coordinador	22.00	0.18
AltaPeriodo	Salida	Salida	22.00	0.18
			121.00	
BajaPeriod	Consulta	Consulta CursoXAlumno	22.00	0.33
BajaPeriod	Baja	BajaPeriod	15.00	0.22
BajaPeriod	Acceso	Coordinador	15.00	0.22
BajaPeriod	Salida	Salida	15.00	0.22
			67.00	
CambiosPeriod	Consulta	Consulta CursoXAlumno	28.00	0.57
CambiosPeriod	Cambios	CambiosPeriod	7.00	0.14
CambiosPeriod	Acceso	Coordinador	7.00	0.14
CambiosPeriod	Salida	Salida	7.00	0.14
			49.00	
Salida	Salida	Salida	338.00	0.50
Salida	Consulta	Invocación	338.00	0.50

Tabla 5.9: Datos de entrada para la cadena de acuerdo a los promedios de las 5000 simulaciones (parte 5)

Capítulo 6

Conclusiones y Trabajo Futuro

6.1. Conclusiones

En este trabajo de tesis se formularon metodologías y técnicas de modelado para la evaluación y análisis de la Confiabilidad para los sistemas de software en Internet.

En el presente trabajo se presentan dos metodologías para el modelado de la confiabilidad. La primera de estas metodologías evalua la Confiabilidad del sistema considerandolo como una caja negra. Este enfoque permite determinar un *modelo ideal* que nos sirve como parametro de referencia para asentar los niveles óptimos de confiabilidad a los cuáles el sistema puede aspirar. Por otro lado, en la metodología se obtiene un modelo que contempla una operación real del sistema. Ambos modelos se obtienen considerando al sistema como una caja negra, y se comparan para evaluar que tan cerca (o lejos) se encuentra la confiabilidad para el caso real comparada con el caso ideal. De esta comparación, la metodología nos permite analizar a grandes razgos cuantos fallos contiene el sistema y que son necesarios de corregir en la fase de pruebas.

Nuestra metodología de evaluación nos permite determinar modelos que toman en cuenta la funcionalidad particular del sistema de software evaluado en un contexto operativo. Esta es una gran diferencia con los modelos generales que han sido propuestos en las dos décadas pasadas, en las cuales no se toma en cuenta la funcionalidad particular del sistema ni su contexto de operación.

La segunda metodología se enfoca al análisis de la Confiabilidad, estimando las probabilidades de fallo de los componentes del sistemas. Para esta metodología el sistema se analiza como una caja

blanca, en donde se realiza un estudio de la funcionalidad de sus componentes y de la relación que guardan entre ellos,

El estudio en este caso se basa en la relación de propiedades intrínsecas del sistema como son la Funcionalidad y la Complejidad Ciclomática del sistema las cuales se conjuntan en un estudio que estima la Probabilidad de Fallo de los componentes de sistema de software.

De esta metodología se obtiene información para mejorar la efectividad en el proceso de evaluación, ya que permite localizar los fallos y las rutas de navegación con mayor probabilidad de incidencia de Fallos.

Otros estudios sólo se han limitado a estudiar la funcionalidad del sistema lo cual limita la objetividad de los resultados en un contexto operativo. La mayoría de estos estudios, citados en el capitulo 4, no se llevan a cabo en un contexto operativo y se limitan a los resultados de un ambiente simulado. En nuestros resultados podemos observar como la complejidad efectivamente si influye en la probabilidad de fallos de una ruta y nos da información importante para mejorar la efectividad en la cobertura de evaluación. Este hecho finalmente nos permite localizar fallos con mayor certeza lo cual sirve de gran ayuda a la fase de pruebas.

Una gran ventaja de las metodologías y las técnicas de modelado i propuestas es que los modelos que proponemos no requieren de un gran conocimiento matemático, además de que la determinación de los parámetros de entrada utiliza algoritmos de baja complejidad y con bajo tiempo de ejecución.

Otra ventaja que hace viable el uso de estas metodologías es que pueden ser aplicadas en contextos operativos. Para lograr este punto fué necesario desarrollar una compleja herramienta de evaluación que genera un conjunto de evaluadores virtuales, los cuales se encargan de probar, analizar y dar seguimiento al modelo propuesto. Esta herramienta reduce permite reducir los tiempos de este proceso, los cuales en la industria actual ocupa mucho tiempo y en la mayoría de los casos se lleva a cabo de forma manual y con grandes grupos de evaluación. Con nuestra herramienta este proceso ocupa solo unas cuantas horas, lo cual le permite ser viable en cualquier contexto operativo.

6.2. Trabajo Futuro

Los modelos propuestos en esta tesis los hemos publicado en dos artículos de congresos internacionales ([83, 84]). Sin embargo el siguiente paso, sería publicar los resultados en una revista especializada en temas de confiabilidad de software. El estar nuestro trabajo expuesto a la crítica y a la revisión nos permitirá valorarlo con mayor efectividad y así decidir cuales serían los trabajos a

futuro.

En general consideramos como trabajos futuros los siguientes:

1. El Desarrollo de una taxonomía de los modelos que han sido propuestos hasta ahora para ubicar con mas claridad en donde se encuentran nuestros modelos propuestos. En esta taxonomía es importante relacionar el tipo de aplicaciones donde se usan los modelos y las distintas técnicas de modelado.

2. Mejora a la herramienta de pruebas. La herramienta en nuestro caso ha sido una piedra angular que nos ha permitido validar nuestras hipótesis. Actualmente es una herramienta que se maneja con entradas de texto por lo que a futuro sería importante trabajar en la parte gráfica para que pudiera ser mas facil de uso.

   En este caso, también es importante buscar otras herramientas de evaluación de la confiabilidad contra quien pudieramos compararnos, a fin de mejorar la efectividad y la usabilidad. Uni de estas herramientas es la herramienta MaTeLo que discutimos en la tesis.

   En estos momentos existe una tesis de maestría cubriendo este aspecto.

3. Extensiones a la aplicabilidad de la herramienta para otros esquemas de dialogo en web.

   Actualmente la herramienta se limita a operar con el protocolo http, por lo cual un paso a futuro sería extender los protocolos de dialogo para que las aplicaciones Web que pueda evaluar la herramienta sea mucho más extensa.

4. Ampliar la metodología para sistemas de Software en general.

   Otro punto importante es aplicar la metodología en otro tipo de casos de estudio que no sean de tecnología Web.

Apéndice A

Requerimientos del Sistema

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 .^{es}tablecidas"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 .^{es}tablecidas"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 A.2.

ID	Descripción	Origen	Definición del sistema
RB0	El acceso al sistema SIV sólo se permitirá mediante clave de acceso y password		
RB1	El sistema SIV permitirá a un alumno elegir los cursos que desea tomar en un cuatrimestre, en la sección del Departamento de Ingeniería Eléctrica a que pertenezca.	Establecido	Definición del sistema
RB2	Sólo podrá hacer uso del sistema SIV aquellos alumnos que se encuentren registrados en el CINVESTAV-IPN y que estén como alumnos regulares	Establecido	Definición del sistema
RB3	Todos los accesos al sistema SIV deberán realizarse desde una interfaz gráfica mediante menús, ventanas y botones, que sea accesible desde Internet.	Establecido	Definición del sistema
RB4	El alumno podrá consultar los contenidos de cada curso y los datos que le permitan elegir su bloque de materia. Sólo se podrán inscribirse a un número máximo de cursos por cuatrimestre y dentro de las fechas establecidas.	Establecido	Definición del sistema
RB5	El sistema permitirá la creación y modificación de las bases de datos que contienen la información de los cursos y de los alumnos inscritos en cada sección del DIE del CINVESTAV-IPN.	Establecido	Definición del sistema
RB6	El acceso al SIV solo podrá hacerse mediante una clave que será inicializada por los coordinadores académicos de cada sección a cada alumno.	Establecido	Definición del sistema
RB7	Los coordinadores académicos de cada sección también contarán con su respectiva clave de acceso y podrá acceder la base de datos de alumnos y de materias con el fin de consultar, añadir o modificar estas bases de datos.	Establecido	Definición del sistema
RB8	El sistema permitirá al público en general ver los cursos que se imparten en las secciones del DIE	Establecido	Definición del sistema
RB9	Se pueden considerar las siguientes vistas al sistema: público en general, alumnos, investigadores y coordinadores académicos.	Establecido	Definición del sistema
RB10	El sistema debe presentar una advertencia al usuario avisando que no ha salvado el estado del sistema y permitiéndole que salve o descarte los cambios hechos.	Establecido	Definición del sistema

Tabla A.1: Tabla de Requerimientos Funcionales (parte 1)

ID	Descripción	Origen	Definición del sistema
RB11	El sistema permitirá al usuario investigador académico ver sus datos y los cursos que imparte, además de la lista de alumnos que toman sus cursos y los alumnos asignados para asesorias.	Propuesto	Documentos y entrevistas
RB12	El sistema permitirá al coordinador agregar, modificar sus datos y eliminar a un investigador de su sección.	Propuesto	Documentos y entrevistas
RB13	El coordinador podrá ver los cursos que imparte cada investigador de su sección	Propuesto	Documentos y entrevistas
RB14	El coordinador podrá ver todos los cursos que se imparten en el cuatrimestre.	Propuesto	Documentos y entrevistas
RB15	El sistema permitirá al alumno visualizar todos los cursos que imparte cada investigador de la sección	Propuesto	Documentos y entrevistas
RB16	El sistema debe permitir al alumno cambiar sus cursos que ha elegido para el cuatrimestre, verificando que los cambios sean dentro de las fechas establecidas.	Propuesto	Documentos y entrevistas
RB17	El sistema permitirá al público en general ver los cursos que imparte cada investigador de las secciones del DIE.	Propuesto	Documentos y entrevistas

Tabla A.2: Tabla de Requerimientos Funcionales (parte 2)

A.3. Descripción del SIV

El SIV está compuesto por los módulos descritos en la figura A.1. Así mismo, en la figura 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 A.2.

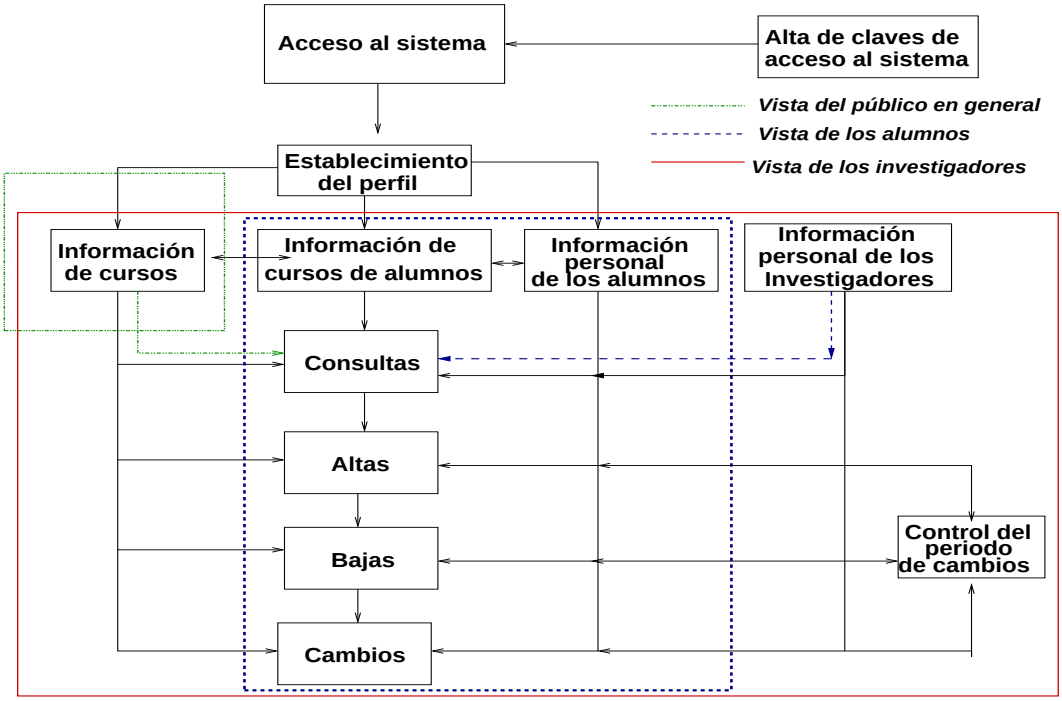

Figura A.1: Diagrama a bloques del SIV

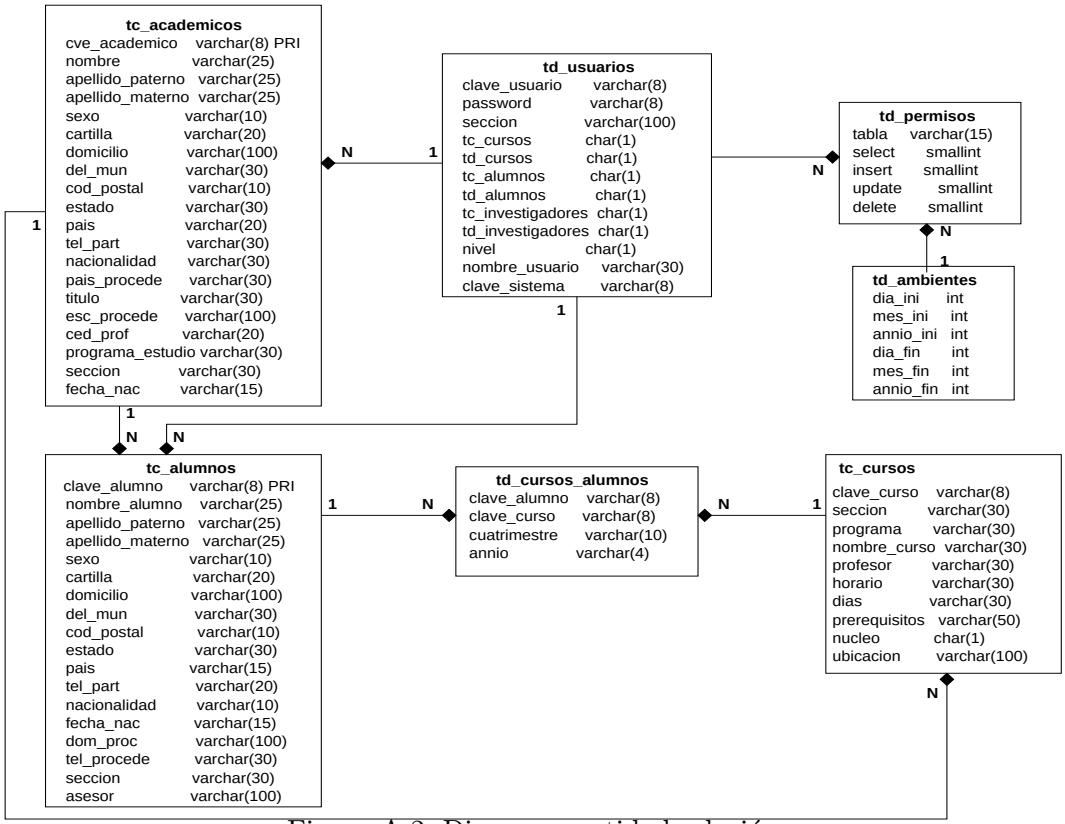

Figura A.2: Diagrama entidad relación

Apéndice B

Cadenas de Markov

B.1. Cadenas de Markov

Los eventos de una cadena de Markov en un espacio de tiempo discreto se ven limitados a ocurrir en instantes de tiempo bien establecidos $0, 1, 2 \ldots k \ldots$ De esta forma, una secuencia estocástica $X1, X2, ...$ se caracteriza por la propiedad de Markov:

$$P[X(t_{k+1}) = x \| X(t_k) = x] \tag{B.1}$$

para todos los valores del estado x, x y t_k $leqt_{k+1}$.

Para especificar el modelo de una cadena de Markov nosotros necesitamos identificar:

1. Un espacio de estados X.

2. La probabilidad inicial del estado $p_0(x) = P[X_0 = x]$, para todas las x que pertenecen a X

3. Las probabilidades de transición $p(x, x)$ donde x es el estado actual y x es el siguiente estado.

La representación del espacio de estados se hace normalmente con números naturales.

B2. Probabilidades de transición y las ecuaciones de Chapman-Kolmogorov

Debido a que el espacio de estados se representa con números enteros no negativos para representar el espacio de estados, utilizaremos los símbolos i y j para representar el estado y el estado siguiente. Tomando en cuenta este punto la probabilidad de transición la podemos definir de la siguiente manera:

$$P_ij(k) \equiv P[X_{k+1} = j|X_k = i] \tag{B.2}$$

Donde i y $j \in X$.

$P_ij(k)$ se define como una función que depende del tiempo en el instante k. Es claro que $0 \leq Pij(k) \leq 1$. También es necesario observar que para cualquier estado i y cualquier instante k:

$$\sum p_{ij}(k) = 1 \tag{B.3}$$

Para todo j.

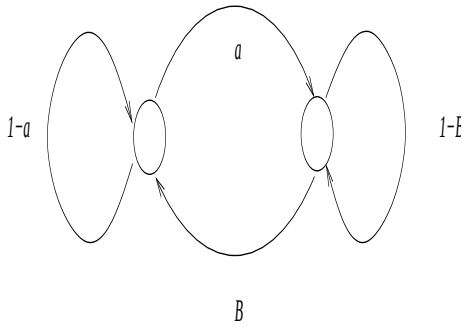

Figura B.1: Diagrama de transición de estados.

En el diagrama de la figura B.1 podemos observar la forma en la que las probabilidades de los arcos son tratadas. Podemos ver que $p_{01} + p_{00} = 1$ y $p_{01} + p_{11} = 1$.

La ecuación (2) se define para un solo paso si deseamos extender esta notación para n pasos, $n = 1, 2, ...$

$$P_{ij}(k, k + n) \equiv P[X_{k+n} = j|X_k = i] \tag{B.4}$$

Esta notación es para un espacio de eventos $[X_{k+n} = j|X_k = i]$ por encima de $[X_u = r]$para

cualquier u tal que $k < u \leq k + n$. Usando la regla de la probabilidad total llegamos a:

$$P[X_{k+n} = j|X_k = i] = P[X_{k+n} = j|X_u = r] = P_{rj}(u, k + n) \tag{B.5}$$

$$P_{ij}(k, k + n) = \sum p_{ir}(k, u)p_{rj}(u, k + n), k < u \leq k + n \tag{B.6}$$

Esta ecuación es conocida como la de ChapmanKolmogorov. En forma matricial:

$$H(k, k + n) \equiv [p_{ij}(k, k + n)], i, j = 0, 1, 2, 3, \ldots \tag{B.7}$$

Puede reescribirse como un producto matricial:

$$H(k, k + n) = H(k, u)H(u, k + n) \ldots \tag{B.8}$$

Seleccionando $u = k + n - 1$, tenemos:

$$H(k, k + n) = H(k, k + n - 1)H(k + n - 1, k + n) \tag{B.9}$$

Seleccionando $u = k + 1$, tenemos:

$$H(k, k + n) = H(k, k + 1)H(k + 1, k + n) \tag{B.10}$$

Clasificación de estados de una Cadena de Markov en un comportamiento estable:

Definición. Un estado j es alcanzable desde un estado i si $P_i^n j > 0$ para $n = 0, 1, 2, ..$

Tenemos S que es un subconjunto del espacio de estados X. Si no hay transición posible de cualquier estado en S para cualquier estado fuera de S, entonces S forma un conjunto *cerrado*.

Definición. Un subconjunto S del espacio de estados X se dice *cerrado* si $p_{ij} = 0$ para cualquier $i \ni S, j \in S$.

Es el caso de un solo estado.

Definición. Un estado i se dice *absorbente* si este forma un solo elemento cerrado.

Claramente la probabilidad de estos estados es $p_{ij} = 1$. Otro caso interesante de conjuntos cerrados consiste en estados mutuamente alcanzables.

Definición. Un conjunto de estados cerrados S se dice *irreducibles* si el estado j es alcanzable desde el estado i para cualquier estado $i, j \in S$.

Definición. Una cadena de Markov se dice *irreducible* si el espacio de estados X es irreducible.

Deduciendo si una cadena no es irreducible, entonces es *reducible*.

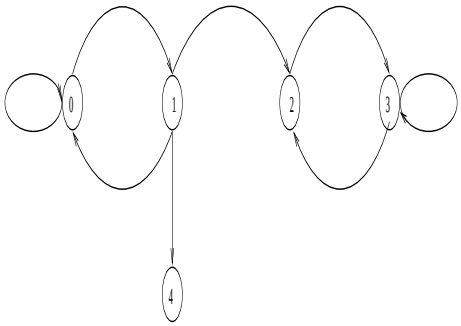

Figura B.2: Diagrama de transición de estados.

En la figura B.2 observamos una Cadena de Markov. Esta es una cadena irreducible. En donde el estado 4 es absorbente. Los estado 2 y 3 forman un conjunto irreducible

Definición. Un estado i se dice que será recurrente si su probabilidad $\pi_i = 1$. Si esta probabilidad es menor que $1\pi_i < 1$ este estado será denominado transitorio.

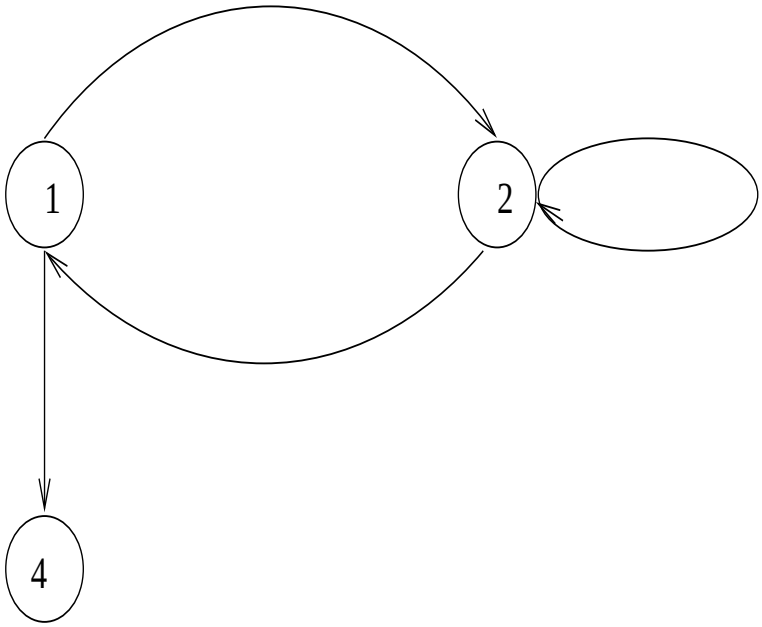

Figura B.3: Diagrama de transición de estados.

En la figura B.3 observamos una cadena, donde el estado 2 es recurrente. Los estados 0 y 1 son transitorios.

B3. Probabilidades de estado

Uno de los principales objetivos del análisis de las cadenas de Markov es la determinación de las probabilidades que tienen los estados de la cadena en instantes específicos de tiempo. Definimos es espacio de estados como sigue:

$$\pi_j(k) \equiv P[X_k = j] \tag{B.11}$$

Acorde a esto, definimos el vector de probabilidades como sigue:

$$\pi(k) = [\pi_0(k), \pi 1(k), \ldots] \tag{B.12}$$

El modelo de la Cadena de Markov esta especificado completamente si además del espacio de estados X y la matriz de probabilidades de transición $\mathbf{P}$, también especificamos un vector inicial de probabilidades, como sigue:

$$\pi(0) = [\pi_0(0), \pi_1(0), \ldots \tag{B.13}$$

Para determinar la probabilidad que toman los estado de la cadena desde el estado **i** al **j** en el tiempo **k** (en **n** pasos). Si realizamos el *análisis transitorio*, podemos auxiliarnos con la matriz $\mathbf{P}$, utilizando las definiciones de $\pi(k)$ y $\mathbf{P}$ tenemos :

$$\pi_j(k+1) = \pi(k)P, k = 0, 1, 2, \ldots \tag{B.14}$$

Tenemos que $\pi_j(k+1)$ es un típico elemento de $\pi(k+1)$. Si tomamos en cuenta la condición del evento $X_{k+1} = j$ sobre $X_k = i$ para todos los posibles estados i, tenemos:

$$\pi_j(k+1) = P[X_{k+1} = j] = \sum P[X_{k+1} = j \mid X_k = i] \bullet P[X_k = i] = \sum p_{ij} \bullet \pi_i(k) \tag{B.15}$$

En términos del vector inicial de probabilidades $\pi(0)$ y de la matriz de probabilidades de transición tenemos:

$$\pi(1) = \pi(0)P \tag{B.16}$$

Para $k = 1$

$$\pi(2) = \pi(1)P = \pi(0)P^2 \tag{B.17}$$

Por inducción matemática:

$$\pi(k) = \pi(1)P^k, k = 1, 2, \ldots \tag{B.18}$$

B.2. Análisis de los estados estables

. Para determinar la probabilidad de un estado i en un tiempo k de una cadena de Markov, es posible realizar el análisis transitorio que hemos estudiado el cual se limita a dados un número finito de pasos analizamos la cadena a cada paso k aplicando la ecuación 17. Sin embargo no es suficiente este análisis después de que el sistema ha estado en operación por un período suficientemente largo de tiempo. Nuestro estudio en este caso se centra en cuantificar el vector de probabilidades estables:

$$\pi_j =_{k\to\infty} lim\pi_j(k) \tag{B.19}$$

En este caso la existencia de límites para la expresión que marca la ecuación 18, no siempre esta garantizada. Así nosotros tenemos que encontrar los valores bajo las siguientes restricciones:

- Estimar las condiciones donde existen los límites.
- Si los límites existen, cuidar que la distribución de probabilidad conserve la expresión $\sum_j \pi_j = 1$.
- Como evaluar π_j.

Si π_j existe para un estado j, esto refleja que existe un estado estable para el estado j, lo cual se refleja en una probabilidad estable.

B.4.1 Cadenas Ergódicas

Definición. Si existe un paso $n > 0$ tal que $P_{ij}^n > 0, i, j = 0, 1, 2...n.$ la cadena de Markov, con esta propiedad, se llama ergódica. Entonces, $P_{ij}^n = \sum_k = 0(P_{ik}^n * P_{kj})$, luego $\pi_j \doteq \sum_k = 0(\pi_k * P_{kj}) y como \sum_j = 0 P_{ij}^n = 1$, entonces $\sum_j = 0\pi_j = 1$.

Teorema. Para una cadena de Markov ergódica, $\pi_j = Lim_{n\to\infty} P_{ij}^n$existe y $\pi_j(j)$ pertenece $0, \ldots, m$ es la única solución no negativa de π_j. Entonces:

$$\pi_j = \sum_{?k=0} (\pi_k * P_{kj}) y \sum_j = 0\pi_j = 1. \tag{B.20}$$

B.4.2 Límites Ergódicos en las Cadenas de Markov.

La relación fundamental en una cadena de Markov es: $P^n = H^n P_0$. Y si nuestro interés es el comportamiento asintótico de P_n, es decir $Lim_{n\to\infty} P_n$entonces el problema es encontrar las

condiciones para que este límite exista y en caso de existir, +dependerá del estado inicial del sistema

Bajo condiciones de regularidad la distribución asintótica existe y bajo estas condiciones la distribución en el límite será independiente de la distribución inicial.

Teorema básico del límite para Cadenas de Markov: En una cadena de Markov recurrente irreducible y aperiódica se tiene que:

$$Lim_{n\to\infty}P_{ii}^{n} =| \sum_{n=1}^{\infty} np_{ii}^{n} \tag{B.21}$$

Siendo p_{ii} la probabilidad de que el proceso regrese al estado i dado que comienza en el estado i. Además

$$Lim_{n\to\infty}P_{ij}^{n} = Lim_{n\to\infty}P_{ii}^{n} \tag{B.22}$$

Del teorema se deduce:

Si $H = P_{ij}$ es la matriz de transición de una cadena de Markov, y si suponemos que esta cadena es recurrente, irreducible y aperiódica, se nos garantiza la existencia de la matriz $H^{(\infty)}$ donde la entrada j, i es $P_{ij}^{(\infty)} = Lim_{n\to\infty}P_{ij}^{n}$, pero como $P_{ij}^{(\infty)} = P_{ii}^{(\infty)}$ se concluye que la matriz $H^{(\infty)}$ tiene sus columnas iguales, esto es de la forma :

$$H^{\infty} = \begin{vmatrix} P_{00}^{\infty} & P_{00}^{\infty} & P_{00}^{\infty} & \dots \\ P_{11}^{\infty} & P_{11}^{\infty} & P_{11}^{\infty} & \dots \\ P_{22}^{\infty} & P_{22}^{\infty} & P_{22}^{\infty} & \dots \\ P_{ii}^{\infty} & P_{ii}^{\infty} & P_{ii}^{\infty} & \dots \end{vmatrix} \tag{B.23}$$

Sea C una cadena *irreducible y recurrente*, entonces $P_{ij}^{n} = 0$ para i pertenece a C y j no pertenece a C dado n. Esto es, toda vez que entremos en C no es posible abandonarlo, luego la matriz P_{ij} con i, j perteneciendo a C estará asociada a una cadena de Markov irreducible y recurrente luego el teorema básico es aplicable si la clase resulta ser aperiódica.

Si $Lim_{n\to\infty}P_{ii}^{n} = \pi = 0$ y la clase es recurrente se dirá entonces que la clase es débilmente ergódica o nula recurrente.

Supongamos que $Lim_{n\to\infty}P_{ii}^{n} = \pi_i > 0$ para algún i en una clase *aperiódica recurrente*, entonces $\pi_j > 0$ para todo j que esté en la clase de i. Si este es el caso diremos que la clase es positiva recurrente o fuertemente ergódica.

El valor $\sum_{n=1}^{\infty} np_{ii}^{n} = m_i$. Se define como el tiempo medio de recurrencia del estado i.

Bajo las condiciones de regularidad del teorema anterior, el $Lim_{n\to\infty} p_{ii}^n = \frac{1}{m_i} = \pi_i$. El calculo de los $\pi_i$se obtiene de:

Teorema. En una clase aperiódica positiva recurrente con estados $i = 0, 1, 2....$ se tiene que

$$Lim_{n\to\infty} P_{ii}^n = \pi_i = \sum_{k=0}^{\infty} p_{kj}\pi_k; \sum_{k=0}^{\infty} \pi_k = 1 \tag{B.24}$$

Cualquier conjunto π_i que satisfaga (20) se llama *probabilidad de distribución estacionaria* de la cadena de Markov.

Observe que si P_{ij} es la matriz de transición asociada a una clase recurrente ergódica positiva, entonces la ecuación $\pi_i = \sum_{k=0}^{\infty} P_{kj}\pi_k$ llevada a su forma matricial:

$$\begin{vmatrix} P_{00} & P_{10} & P_{20} & \dots \\ P_{01} & P_{11} & P_{21} & \dots \\ P_{02} & P_{12} & P_{22} & \dots \\ \dots & \dots & \dots & \dots \end{vmatrix} \begin{vmatrix} \pi_1 \\ \pi_2 \\ \pi_3 \\ \dots \\ \pi_i \end{vmatrix} = \begin{vmatrix} \pi_1 \\ \pi_2 \\ \pi_3 \\ \dots \\ \pi_i \end{vmatrix}$$

Establece claramente que $(\pi_1, \pi_2, \pi_3, \dots....)^t$ es un auto vector de la matriz P_{ij} asociado al autovalor 1. En este caso la matriz de Markov tiene un autovalor igual a 1, este valor será el mayor, en valor absoluto, si la matriz es primitiva, esto es si todas sus entradas son positivas para la potencia n de la matriz.

B.5. Resolución de sistema de ecuaciones lineales.

El objetivo de este apartado es examinar los aspectos numéricos que se presentan al resolver sistemas de ecuaciones lineales de la forma:

$$\begin{aligned} a_{11}x_1+ \quad & a_{12}x_2+ \quad a_{13}x_3+ \quad \dots \quad a_{1n}x_n \quad = b_1 \\ a_{21}x_1+ \quad & a_{22}x_2+ \quad a_{23}x_3+ \quad \dots \quad a_{2n}x_n \quad = b_2 \\ a_{31}x_1+ \quad & a_{32}x_2+ \quad a_{33}x_3+ \quad \dots \quad a_{3n}x_n \quad = b_3 \\ & \dots \\ a_{n1}x_1+ \quad & a_{n2}x_2+ \quad a_{n3}x_3+ \quad \dots \quad a_{nn}x_n \quad = b_n \end{aligned} \tag{B.25}$$

Se trata de un sistema de n ecuaciones con n incógnitas, $x_1, x_2, \dots, x_n$. Los elementos a_ij y b_i son números reales fijados. El sistema de ecuacionesáse puede escribir, empleando una representación matricial, como la siguiente

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \dots & & & & \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{vmatrix} \begin{vmatrix} x_1 \\ x_2 \\ x_3 \\ \dots \\ x_n \end{vmatrix} = \begin{vmatrix} B_1 \\ B_2 \\ B_3 \\ \dots \\ B_n \end{vmatrix} \tag{B.26}$$

Entonces podemos denotar estas matrices por A, x y b de forma que la ecuación se reduce simplemente a:

$$Ax = b \tag{B.27}$$

Los métodos de resolución de sistemas de ecuaciones se pueden dividir en dos grandes grupos:

- Los Métodos exactos o algoritmos finitos que permiten obtener la solución del sistema de manera directa.
- Los Métodos aproximados que utilizan algoritmos iterativos e infinitos y que calculan las soluciones del sistema por aproximaciones sucesivas.

Al contrario de lo que pueda parecer, en muchas ocasiones los métodos aproximados permiten obtener un grado de exactitud superior al que se puede obtener empleando los denominados métodos exactos, debido fundamentalmente a los errores de truncamiento que se producen en el proceso. De entre los métodos exactos analizaremos el método de Gauss y una modificación de éste denominado método de Gauss-Jordan. Entre los métodos aproximados nos centraremos en el estudio de los métodos de Richardson, Jacobi y Gauss-Seidel

B.2.1. Métodos de resolución exacta

Antes de abordar el estudio de los métodos de resolución exacta de sistemas de ecuaciones lineales, analizaremos algunas propiedades y relaciones útiles que caracterizan a estos sistemas.

B.5.1 Sistemas de solución inmediata.

Analizaremos previamente un sistema que sea fácil de resolver. Por ejemplo, supongamos que la matriz A de nxn presenta estructura diagonal, es decir, todos los componentes distintos de cero se

encuentran sobre la diagonal principal. El sistema de ecuacionesátoma por tanto la forma:

$$\begin{vmatrix} a_{11} & 0 & 0 & \dots & 0 \\ 0 & a_{22} & 0 & \dots & 0 \\ 0 & 0 & a_{33} & \dots & 0 \\ \dots & & & & \\ 0 & 0 & 0 & \dots & a_{nn} \end{vmatrix} \begin{vmatrix} x_1 \\ x_2 \\ x_3 \\ \dots \\ x_n \end{vmatrix} = \begin{vmatrix} B_1 \\ B_2 \\ B_3 \\ \dots \\ B_n \end{vmatrix} \tag{B.28}$$

En este caso el sistema se reduce a n ecuaciones simples y la solución es:

$$x = \begin{vmatrix} b_1 & a_{11} \\ b_2 & a_{22} \\ b_3 & a_{33} \\ \dots & \\ b_n & a_{33} \end{vmatrix} \tag{B.29}$$

B.5.2 La factorización LU

Supongamos que A se puede factorizar como el producto de una matriz triangular inferior L con una matriz triangular superior U:

$$A = LU \tag{B.30}$$

En este caso, el sistema de ecuaciones dado podría representarse en la forma: á

$$Lux = b \tag{B.31}$$

Si denominamos z a la matriz columna de n filas resultado del producto de las matrices Ux, tenemos que la ecuación 30 se puede reescribir del siguiente modo:

$$Lz = b \tag{B.32}$$

A partir de las ecuaciones 30 y 31, es posible plantear un algoritmo para resolver el sistema de ecuaciones empleando dos etapas:

- Primero obtenemos z aplicando el algoritmo de sustitución progresiva en la ecuación 31.
- Posteriormente obtenemos los valores de x aplicando el algoritmo de sustitución regresiva a la ecuación

$$Ux = z \tag{B.33}$$

Si tenemos:

$$L = \begin{vmatrix} l_{11} & 0 & 0 & \dots & 0 \\ l_{21} & l_{22} & 0 & \dots & 0 \\ l_{31} & l_{32} & l_{33} & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ l_{n1} & l_{n2} & l_{n3} & \dots & l_{nn} \end{vmatrix} \quad U = \begin{vmatrix} u_{11} & u_{12} & u_{13} & \dots & u_{1n} \\ 0 & u_{22} & u_{23} & \dots & u_{2n} \\ 0 & 0 & u_{33} & \dots & u_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & u_{nn} \end{vmatrix} \tag{B.34}$$

tales que cumplan la ecuacióná29. Cuando esto es posible, decimos que A tiene una descomposición LU. Se puede ver que la ecuación anterior no determina de forma única a L y a U. De hecho, para cada i podemos asignar un valor distinto de cero a l_{ii} o u_{ii} (aunque no ambos). Por ejemplo, una elección simple es fijar $l_{ii} = 1$ para $i = 1, 2, \dots, n$ haciendo de esto modo que L sea una matriz *triangular inferior unitaria*. Otra elección es hacer U una *matriz triangular superior unitaria* (tomando $u_{ii} = 1$ para cada i).

2.2.3 Eliminación Gaussiana básica Ilustraremos el método de Gauss aplicando el procedimiento a un sistema de cuatro ecuaciones con cuatro incógnitas: Paso 1:

$$\begin{vmatrix} 6 & -2 & 2 & 4 \\ 12 & -8 & 6 & 10 \\ 3 & -13 & 9 & 3 \\ -6 & 4 & 1 & -18 \end{vmatrix} \begin{vmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{vmatrix} = \begin{vmatrix} 12 \\ 34 \\ 27 \\ -38 \end{vmatrix}$$

En el primer paso, multiplicamos la primera ecuación por $\frac{12}{6} = 2$ y la restamos a la segunda, después multiplicamos la primera ecuación por $\frac{3}{6} = \frac{1}{2}$ y la restamos a la tercera y finalmente multiplicamos la primera ecuación por -6/6=-1 y la restamos a la cuarta. Los números $2, \frac{1}{2} y - 1$ son los **multiplicadores** del primer paso del proceso de eliminación. El número 6 es el **elemento pivote** de este primer paso y la primera fila, que no sufre modificación alguna, se denomina **fila pivote**. El sistema en estos momentos tiene el siguiente aspecto:

Paso 2:

$$\begin{vmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & -12 & 8 & 1 \\ 0 & 2 & 3 & -14 \end{vmatrix} \begin{vmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{vmatrix} = \begin{vmatrix} 12 \\ 10 \\ 21 \\ -26 \end{vmatrix}$$

En el siguiente paso del proceso, la segunda fila se emplea como **fila pivote** y -4 como **elemento pivote**. Aplicamos del nuevo el proceso: multiplicamos la segunda fila por $\frac{-12}{4} = -3$ y la restamos de la tercera y después multiplicamos la segunda fila por $\frac{2}{-4} = \frac{-1}{2}$ y la restamos a la cuarta. Los multiplicadores son en esta ocasión 3 y $\frac{-1}{2}$y el sistema de ecuaciones se reduce a:

Paso 3:

$$\begin{vmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & 5 \\ 0 & 0 & 4 & -13 \end{vmatrix} \begin{vmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{vmatrix} = \begin{vmatrix} 12 \\ 10 \\ -9 \\ -21 \end{vmatrix}$$

El último paso consiste en multiplicar la tercera ecuación por $\frac{4}{2} = 2$ y restarla a la cuarta. El sistema resultante resulta ser:

Paso 4:

6 -2 2 4 X1

12 0 -4 2 2 x2

10 0 0 2 -5 x3 = -9 0 0 0 -3 x4

-3

$$\begin{vmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & 5 \\ 0 & 0 & 0 & -3 \end{vmatrix} \begin{vmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{vmatrix} = \begin{vmatrix} 12 \\ 10 \\ -9 \\ -3 \end{vmatrix}$$

El sistema resultante es triangular superior y equivalente al sistema original (las soluciones de ambos sistemas coinciden). Sin embargo, este sistema es fácilmente resoluble aplicando el algoritmo de **sustitución regresiva** explicado en el apartado anterior. La solución del sistema de ecuaciones resulta ser:

$$X = \begin{vmatrix} 1 \\ -3 \\ -2 \\ -1 \end{vmatrix}$$

Apéndice C

Estudio de Whittaker y Thomason.

C.1. Un Modelo con Cadenas de Markov para Pruebas de Software Estadísticas.

Whittaker y Thomason proponen un modelo estocástico para la planeación de la etapa de pruebas basado en Cadenas de Markov. Este trabajo propone que las evaluaciones estadísticas puedan ser diseñadas en base de un modelo estocástico. Para los autores, los modelos estocásticos son capaces de modelar sistemas con múltiples distribuciones probabilísticas en términos de variables aleatorias. Idealmente los parámetros de entrada del modelo son establecidos usando información de diversas fuentes, incluyendo funciones que describen el comportamiento del software, así como el uso de patrones de versiones anteriores.

Este modelo tiene elementos de un dominio d, que se determina a partir de la función de entrada y de una relación probabilística definida en estos elementos. Del dominio d se determinan las secuencias de pruebas, considerando que una secuencia finita puede considerarse también como una prueba. Las secuencias de entrada son aplicadas al modelo y los resultados de estas pruebas se aplican a un segundo modelo. Este segundo modelo se analiza y se calcula entonces un estimado del lugar donde se origina el fallo.

La cadena desarrollada para un sistema de software se compone de estados, los cuales deben representar el funcionamiento del sistema. Los estados deben mantenerse constantes durante el fun-

cionamiento del sistema. Las transiciones de los estados son cuantificadas mediante probabilidades de transición. Para determinar el conjunto de estados, debemos considerar cada entrada y la información necesaria para aplicarla. Es posible que ciertos modos causen entradas menos probables e incluso ilegales. Así un modo representa un estado ó un conjunto de estados.

Los estados que los autores identifican son los estados inicial, final y los representados en la cadena. La cadena está completamente definida cuando se establecen las probabilidades de transición y estos valores de probabilidad deben representar los mejores estimados del uso real.

C.2. Caso de Estudio

En el artículo los autores presentan un ejemplo de su aplicación el cual está conformado por el menu descrito por la Figura C.1. Este sistema tiene varias opciones y los usuarios se pueden dezplazar en el hacia arriba, hacia abajo y pueden seleccionar una opción. En la opción *Selecionar Proyecto*, el proyecto se selecciona y de acuerdo a este las otras opciones (Introducir Datos, Analizar Datos e Imprimir) del menu se activan. La opción salir siempre esta activa. Las variables a analizar en este sistema son:

- *La posición del cursor* (su abreviacion CL CursorLocation), la cual toma los valores *Sel, Ent , Anl, Prt ó Ext* para cada opción de menu.
- *Proyecto Definido* (el cual es abreviado con PD), el cual toma los valores *Si o No.*

C.3. Diseño de la Cadena de Markov.

C.3.1. Estimación de las probabilidades de entrada de la cadena de Markov

El espacio de estados es el siguiente:$\{(CL = SI, PD = NO)$, $(CL = Ent, PD = NO)$, $(CL = Anl, PD = NO)$, $(CL = Prt, PD = NO)$, $(CL = Ext, PD = NO)$, $(CL = Sel, PD = NO)$, $(CL = Sel, PD = Si)$, $(CL = Ent, PD = Si)$, $(CL = Anl, PD = Si)$, $(CL = Prt, PD = Si)$, $(CL = Ext, PD = Si)$.

En el diagrama de transición (descrito en la Figura C.2) una secuencia va desde el estado *Sin Invocar* hasta el estado *Termino*, lo cual representa la ejecución de una prueba. Un conjunto de secuencias son aplicadas para los casos de prueba. Las secuencias se establecen de acuerdo a las probabilidades que son asignadas en el diseño de la Cadena de Markov. Un ejemplo detallado de

From State	Transition Stimuli	To State	Unif Prob	Est. Prob
Sin Invocar	Invocar	$(CL = Sel, PD = NO)$	1	1
$(CL = Sel, PD = NO)$	$\downarrow$	$(CL = Ent, PD = NO)$	$\frac{1}{3}$	$\frac{1}{10}$
	$\uparrow$	$(CL = Ext, PD = NO)$	$\frac{1}{3}$	$\frac{1}{10}$
	$\rightarrow$	$(CL = Sel, PD = NO)$	$\frac{1}{3}$	$\frac{8}{10}$
$(CL = Ent, PD = NO)$	$\downarrow$	$(CL = Anl, PD = NO)$	$\frac{1}{3}$	$\frac{1}{3}$
	$\uparrow$	(CL=Sel, PD=NO)	$\frac{1}{3}$	$\frac{1}{3}$
	$\rightarrow$	(CL=Ent, PD=NO)	$\frac{1}{3}$	$\frac{1}{3}$
$(CL = Anl, PD = NO)$	$\downarrow$	$(CL = Prt, PD = NO)$	$\frac{1}{3}$	$\frac{1}{3}$
	$\uparrow$	$(CL = Ent, PD = NO)$	$\frac{1}{3}$	$\frac{1}{3}$
	$\rightarrow$	$(CL = Anl, PD = NO)$	$\frac{1}{3}$	$\frac{1}{3}$
$(CL = Prt, PD = NO)$	$\downarrow$	$(CL = Ext, PD = NO)$	$\frac{1}{3}$	$\frac{1}{3}$
	$\uparrow$	$(CL = Anl, PD = NO)$	$\frac{1}{3}$	$\frac{1}{3}$
	$\rightarrow$	$(CL = Prt, PD = NO)$	$\frac{1}{3}$	$\frac{1}{3}$
$(CL = Ext, PD = NO)$	$\downarrow$	$(CL = Sel, PD = NO)$	$\frac{1}{3}$	$\frac{1}{3}$
	$\uparrow$	$(CL = Prt, PD = NO)$	$\frac{1}{3}$	$\frac{1}{3}$
	$\rightarrow$	Termino	$\frac{1}{3}$	$\frac{1}{3}$
$(CL = Sel, PD = Si)$	$\downarrow$	$(CL = Ent, PD = Si)$	$\frac{1}{3}$	$\frac{5}{9}$
	$\uparrow$	$(CL = Ext, PD = Si)$	$\frac{1}{3}$	$\frac{3}{9}$
	$\rightarrow$	Seleccionar Proyecto	$\frac{1}{3}$	$\frac{1}{9}$
$(CL = Ent, PD = Si)$	$\downarrow$	$(CL = Anl, PD = Si)$	$\frac{1}{3}$	$\frac{4}{7}$
	$\uparrow$	$(CL = Sel, PD = Si)$	$\frac{1}{3}$	$\frac{1}{7}$
	$\rightarrow$	Enter Data	$\frac{1}{3}$	$\frac{2}{7}$
$(CL = Anl, PD = Si)$	$\downarrow$	$(CL = Prt, PD = Si)$	$\frac{1}{3}$	$\frac{3}{6}$
	$\uparrow$	$(CL = Ent, PD = Si)$	$\frac{1}{3}$	$\frac{1}{6}$
	$\rightarrow$	Analyz Data	$\frac{1}{3}$	$\frac{2}{6}$
$(CL = Prt, PD = Si)$	$\downarrow$	$(CL = Ext, PD = Si)$	$\frac{1}{3}$	$\frac{2}{6}$
	$\uparrow$	$(CL = Anl, PD = Si)$	$\frac{1}{3}$	$\frac{1}{6}$
	$\rightarrow$	Print Report	$\frac{1}{3}$	$\frac{2}{6}$
$(CL = Ext, PD = Si)$	$\downarrow$	$(CL = Sel, PD = Si)$	$\frac{1}{3}$	$\frac{1}{6}$
	$\uparrow$	$(CL = Prt, PD = Si)$	$\frac{1}{3}$	$\frac{2}{6}$
	$\rightarrow$	Termino	$\frac{1}{3}$	$\frac{3}{6}$
Select Project	cancel	$(CL = Sel, PD = No)$	$\frac{1}{2}$	$\frac{1}{8}$
	select	$(CL = Sel, PD = Yes)$	$\frac{1}{2}$	$\frac{7}{8}$
Enter Data	data	$(CL = Ent, PD = Si)$	1	1
Anlyz Data	analyze	$(CL = Anl, PD = Si)$	1	1
Print Report	print	$(CL = Prt, PD = Si)$	1	1
Termino	null	Sin Invocar	1	1

Tabla C.1: Tabla de probabilidades de transición de los componentes de la aplicación de software

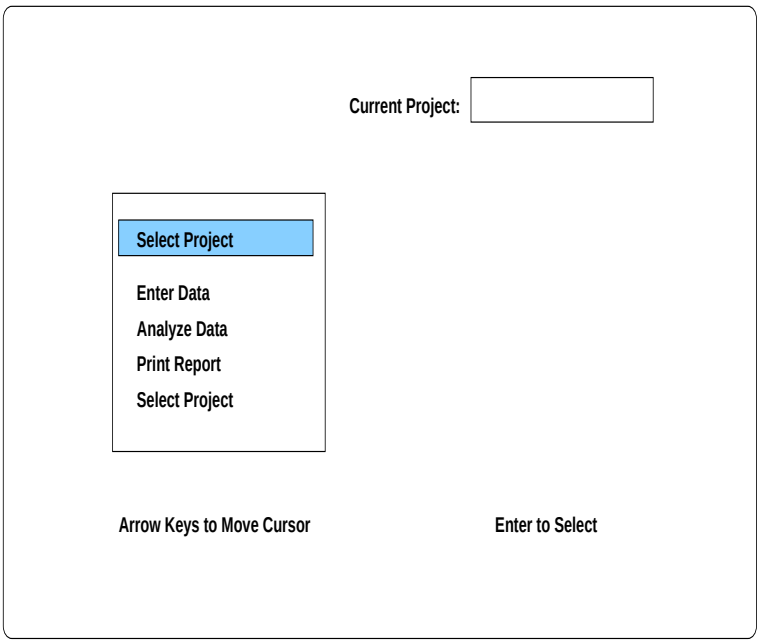

Figura C.1: Un ejemplo de un sistema de software

este proceso lo podemos apreciar en el trabajo [100]. En este trabajo Whittaker ejecuta paso a paso el diseño de la cadena y la obtención de los datos de entrada.

En la Tabla C.1 tenemos las probabilidades asignadas para el ejemplo de la Figure C.2. En la primera columna de la Tabla se describe el estado de procedencia de la transición. En la segunda columna, se describe el estímulo para generar la transición. En la tercera columna se describe el estado hacia donde se dirige la transición. En la cuarta columna, se describe la Probabilidad uniforme y en la quinta columna se describe la Probabilidad estacionaria.

El detalle para obtener el vector de soluciones π se encuentra en la fuente bibliogrfica antes mencionada [100]. En este artículo podemos observar que la cadena de Markov se representa mediante una matriz de estados U. A partir de esta matriz se obtiene un sistema de ecuaciones, el cual debe resolverse. La solucion de este sistema de ecuaciones se representa mediante el vector de soluciones π.

C.3.2. Generación de las secuencias de la cadena de Markov.

En el trabajo de Whittaker, también se argumenta que la relación de las secuencias con los estados a evaluar esta de acuerdo a las relaciones descritas en la Tabla C.2. En esta Tabla se describen el estado i, su solución estacionaria π_i, el numero de estados x_i necesarios para llegar al

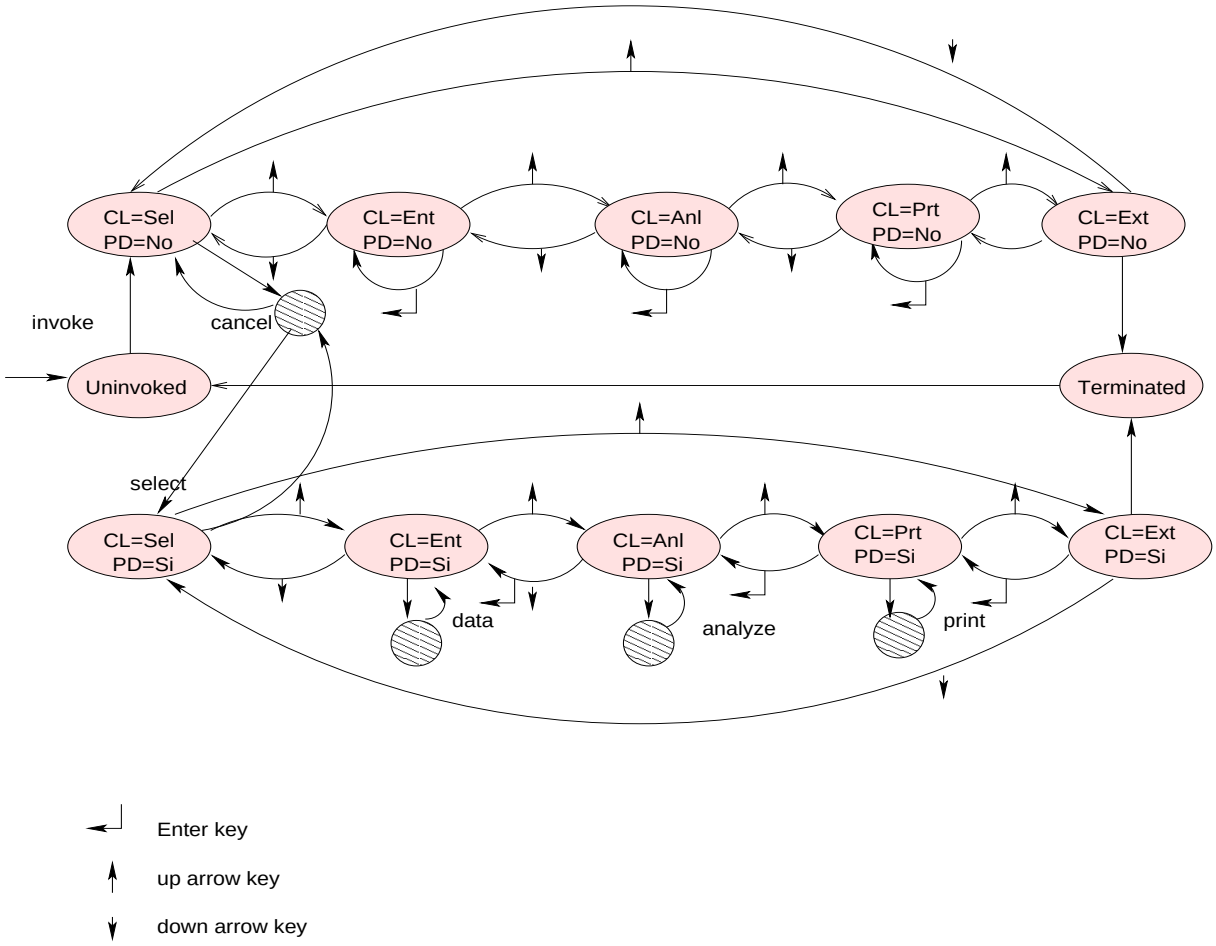

Figura C.2: Cadena del menú del sistema de software

State	π_i	$xi = frac1\pi$	$s = fracx_i/x_term$

Tabla C.2: Relaciones con los estados del vector de soluciones π

estado i y finálmente el número esperado de secuencias s de ejecución para el estado. Todos estos valores son descritos y justificados por Whittaker.

C.3.3. Desarrollo de las funciones de cobertura del caso de estudio SIV, de acuerdo al vector de soluciones π_i.

Para la cobertura de evaluación el autor utiliza las ecuaciones que se presentan en la Tabla C.3, en donde se describen 1). el nombre de la función, 2). la relación matemática de la función y 3). una breve descripción de la función en cuestión.

Es importante señalar que estas funciones se resuelven mediante métodos númericos y en algunos

Resultado	**Ecuación de Probabilidad ó Media**	**Interpretación de la Media**
Cadena Recurrente		
Distribucion estacionaria π	$\pi_j = \sum_i U_{ij}$	π_j es la taza de aparicióna asintótica del estado j en un gran número de secuencias de U
Tiempo de recurrecia para el estado j	$m_{jj} = \frac{1}{\pi_j}$	El número medio de transiciones entre las ocurrencias del estado j en un gran número de secuencias de U.
Número de ocurrencias del estado i entre las ocurrencias del estado j	$m_{jj}\pi_i = \frac{\pi_i}{\pi_j}$	El número medio de ocurrencias del estado i entre el estado j.
Tiempo de primera pasada	$m_{ij} = 1 + \sum_{k \neq j} U_{ik} m_{kj}$	El número medio de las transiciones hasta el estado j que ocurre desde el estado i.

Tabla C.3: Relaciones y Funciones relacionadas con la cobertura de evaluacion del sistema.

casos (como en la determinación de los *tiempos de primera pasada*), se resuelven mediante la solución de sistemas de ecuaciones lineales. También es importante mencionar que cada medida se relaciona con la matriz de transiciones U.

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

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

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

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

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

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

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

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

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

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

[25] Brown A., Kar G., and Keller A. .An Active Approach to Characterizing Dynamic Dependencies for Problem Determination in a Distributed Eviroment". Seventh IFIPIEEE International Symposium on Integrated Network Management, Seattle. WA, May 2001.

[26] V.R. Basili, L.C. Briand and W. Melo. .A Validation of object-oriented design metrics as quality indicators". International Conference on Comunications. Vancouver Canada 1999, pp 1547-51.

[27] J. Choi, M. Choi and S. Lee. .An Alarm Correlation and Fault Identification Scheme Based on OSI Managed Object Classes". International Conference on Comunications. Vancouver Canada 1999, pp 1547-51.

[28] Joâo W. Cangussu, Richard M. Karcich. "Software Release Control using Defect Based Quality Estimation". 15th International Symposium on Software Reliability Engineering (ISSRE'04). 1071-9458/04

[29] Dow, H.E., Murphy, J.S., "Detailed Product Knowledge is not Required for a Successful Formal Software Inspection". Seventh Software Engineering Process Group Meeting, Boston, MA, 1994.

[30] D.G. Luenberger, *"Introduction to Dinamic Systems:Theory, models and applications".* ISBN: John Willey & Sons, 1979.

[31] R.A. DeCarlo *"Linear systems:A state variable approach with numerical implementation".* ISBN: Upper Saddle River, New Jersey, Prentice Hall, 1989.

[32] Breiman, L., J. Friedman, R. Olshen, and C. Stone. Çlassification and Regression Trees". ISBN: 0534980538 *Chapman and Hall, 1984.*

[33] Fagan M.E. .Advances in Software Inpections". *IEEE Transactions on Software Engineering*, Vol. 12, pp. 744-751, 1986.

[34] B. Gruschke. "Itegrated Event Management:Event Correlation Using Dependency Graphs". Proceedings of 9th IFIPIEEE International Workshop on Distributed System Operation and Management (DSOM'98).1998.

[35] "IEEE Standard Glossary of Software Engineering Terminology". IEEE STD 610.12-1990.

[36] S. Katker and M. Paterok. "Fault Isolation and Event Correlation for Integrated Fault Management". Proceedings of Fifth IFIPIEEE International Symposium on Integrated Network Management (IMV), San Diego, CA.,1997, 538-596.

[37] C. Ebert. Classification techniques for MetricBased Software Develpment. Software Quality Journal, 5(4):255-272, Dec 1992

[38] Jelinski, Z. And P. Moranda, Software Reliability Research, Statistical Computer Performance Evaluation, W. Freiberger, Ed. New York:academic Press, 1972, pp. 465 - 484.

[39] Littlewood, B. Stochastic Reliability Growth: A Model for Fault Removal in Computer Programs and Hardware Designs IEEE Transactions on Reliability, Vol. SE-10, 1981, pp.313-320.

[40] Goel, A. L., and Okumoto, An Analysis of Recurrent Software Failures in Real-Time Control Systems, Proccedings ACM Annual Technology Conference, Washintong, D.C., 1978, pp. 496-500.

[41] Goel, A. L., and Okumoto, A Time-Dependent Error-Detection Rate Model for Software Reliability, Vol. R-28, 1979, pp. 206-211.

[42] Musa, J. D., and K. Okumoto, A Logarithmic Poisson Execution Time Model for Software Reliability Measurement, Proceedings Seventh International Conference on Software Engineering, Los Alamitos, Calif.:IEEE Computer Society Press, 1983, pp. 230-238.

[43] Yamada, S. M. Ohba ans Osaki, S-shape Reliability Growth Modeling for Software Error Detection, IEEE Transactions on Reliability, Vol. R-32, 1983, pp.475-478.

[44] Ohba, M., Software Reliability Analysis Models, IBM Journal of Research and Development, Vol.28, 1984, pp. 428-443.

[45] Musa, J.D. Operational profiles in software-reliability engineering Software, IEEE Volume 10, Issue 2, Mar 1993 Page(s):14 - 32 Digital Object Identifier 10.1109/52.199724

[46] T.M. Khoshgoftaar, E.B. Allen, K.S. Kalaichelvan and N. Goel. ."Early Quality Predictions: A case study in telecomunications". *IEEE Software, 13(1):65-71, Jan. 1991.*

[47] T.M. Khoshgoftaar, E.B. Allen, F.D. Ross, R.D. Munik oti, N. Goel and A. Nandi. "Prediting Fault-Prone Modules with Cased-Based Reasoning". Eighth International Symposium on Software Reliability Engineering (ISSRE'97). pp. 27 Nov. 1997.

[48] T.M. Khoshgoftaar and D. L. Lanning. .A Neuronal Network Approach for Early Detection of Program Modules Having High Risk in the Maintenance Phase". *Journal of Systems and Software, 29(1):85-91, April. 1995.*

[49] T.M. Khoshgoftaar and N. Seliya. "Tree-Based Software Quality Estimation Models For Fault Prediction". Proceeding the Eighth International Symposium on Software Metrics (METRICS'02). pp. 203 June 2002.

[50] R.W. Selby and A. A. Porter. "Learning from examples: Generation and evaluation of desicion trees for software resources analisys" *IEEE Transactions on Software Engineering.* 14(12):1743-1756, Dec. 1988

[51] J. Troster and J. Tian. "Measurement and defect modeling for a legacy software system". *Annuals of Software Engineering.* 1:95-118, 1995

[52] Vittorio Cortesella, Katerina Goseva-Popstojanova. "Model-Based Performance Risk Analysis". *IEEE Transactions on Software Engineering.* 0098-5591 January 2005,

[53] Podgufski, A., Leon, d., Francis, P., Minch M., Sun, J., Wang, B. and Masri, W. Automated Support For Classifiying Software Failure Reports. 25th International Conference on Software Engineering (Portland OR. May 2003).

[54] J. C. Munson and T.M. Khoshgoftaar. "The Detection of Fault Prone Programs". *IEEE Transactions on Software Engineering, 18(5):423-433, May. 1992..*

[55] C. Ebert. Çlassification Techniques for Metric Based Software Development". *Software Quality Journal, 5(4):255-272, Dec. 1996..*

[56] N. Fenton and M. Neil. "Software Metrics and Risk". Proceeding 2nd European software Measurement Conference, TI-KVIV. Amsterdam, Oct. 1999.

[57] S. S. Gokhale and M.R. Lyu Regression tree modeling for the prediction of software quality". Proceeding the Third ISSAT International Conference on Reliability and Quality in Design. pp. 31-36, Anaheim, CA. Mar 1997.

[58] L. Guo, B. Cukic and H. Singh. "Prediction Fault Prone Modules by the Dempster-Shafer Belief Networks". Proceeding 18th IEEE International Conference on Automated Software (ASE 2003). October 2003.

[59] L. Guo, Yan Ma, Bojan Cukic and Harshinder Singh. Robust Prediction of Fault - Proneness by Random Forests". Proceeding 15th International Symposium on Software Reliability Engineering (ISSRE'04). 1071-9458/04.

[60] Per Runeson, Anneliese Andrews. "Detection or Isolation of Defects? An Experimental Comparison of Unit Testing and Code Inspection". 14th International Symposium on Software Reliability Engineering (ISSRE'03)

[61] Nachiappan Nagappan, Laurie Williams , John Hudepohl, Will Snipes, Mladen Vouk "Preliminary Results On Using Static Analysis Tools For Software Inpection". 15th International Symposium on Software Reliability Engineering (ISSRE'04). 1071-9458/04

[62] Patrick Francis, David Leon, Melinda Mich, Andy Podgurski. "Tree-Based Methods for Classifying Software Failures". 15th International Symposium on Software Reliability Engineering (ISSRE'04). 1071-9458/04

[63] Kelly, D., Shepard, T. .A Case Study in the use of Defects Classification in Inspections". 25th IBM Centre for Advanced Studies Conference, 2001 pp. 26-39.

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

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

[66] ana Taghdiri, "Inferring Specifications to Detect Errors in Code". *Automated Software Engineering, 19th International Conference on (ASE'04)*

[67] Jain, A.K. and Dubes, R.C. .Algorithms for Clustering Data". ISBN:0-13-022278-X *Prentice Hall, 1988.*

[68] John D. Musa Software Reliability Engineering: More Reliable Software Faster and Cheaper Second Edition ISBN: 1-4184-9387 Authorhouse, 2004.

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

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

[71] Thomas McCabe. Çyclomatic Complexity and the Year 2000". *IEEE Software, 0740-7459/96*, May 1996.

[72] S. Yemini, S. Kliger, "High Speed and Robust Event Correlation", *IEEE Comunication Magazine*, vol. 34, no. (5), pp. 82-90 , May 1996.

[73] Jeff Tian, Li Ma, Zhao Li and A. Gunes Koru. .A Hierarchical Strategy for Testing Web-Based Applications and Ensuring Their Reliability", Proceedings of the 27th Annual International Compurer Software and Applications Conference (COMPSAC 03). *Computer IEEE* 0730-3157/03.

[74] Jeff Tian. "Quality-Evaluation Models and Measurements". *IEEE Software, May/June 2004 (Vol.21, No. 3).*

[75] http:// www.mysql.com

[76] http:// www.apache.org

[77] http:// www.php.com

[78] http:// www.sun.com

[79] J. M Jézéquel and M. Meyer. Design by Contract: The Lessons of Ariane. IEEE Computer, 30(1):129-130,1997.

[80] N.G. Levenson and Turner. An Investigation of the Therac-25 Accidents. IEEE Computer ,26(7):18-41,1993

[81] Hans Van Vliet. Software Engineering, Principles and Practice. John Wiley & Sons Ltd. ISBN: 0-471-97508

[82] A. M. Law, W. David Kelton. Simulation Modeling and Analysis. .

[83] Leticia Dávila-Nicanor, Pedro Mejía-álvarez. Reliability Improvement of Web-Based Software Applications". Quality Software, 2004, QSIC 2004. Proceedings Fourth International Conference IEEE. Germany. Pp. 180-188 10.1109QSIC04.

[84] Leticia Dávila-Nicanor, Pedro Mejía-álvarez. Reliability Evaluation of Web-Based Software Applications". Sixth Mexican International Conference on Computer Science. Puebla, Mexico September 2005.

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

[86] Swapna S. Gokhale, Michael R. Lyu and Kishor S. Trivedi. *Reliability Simulation of Component-Based Software System.* Proc. 9th International Symposium on Software Reliability Engineering (ISSRE'98). Paderborn, Germany, Nov. 1998, pp. 192-201.

[87] S. S. Gokhale and M.R. Lyu *Regression tree modeling for the prediction of software quality.* Proceeding the Third ISSAT International Conference on Reliability and Quality in Design. pp. 31-36, Anaheim, CA. Mar 1997.

[88] K. Goseva-Popstojanova and K. Trivedi. *Archtecture-based approach to reliability assessment of software systems.* Performance Evaluation an International Journal. Rome, Italy, July 2002, pp. 302-309.

[89] "IEEE Standard Glossary of Software Engineering Terminology". IEEE STD 610.12-1990.

[90] J. Offutt. "Quality Attributes of Web Software Applications". IEEE Software, 0740-7459/02. MarchApril 2002.

[91] J. C. Laprie, K. Kanoun, C. Beounes, and M. Kaaniche. *The KAT (Knowledge-Action-Tranformation) Approach to the Modeling and Evaluation of Reliability and Availability Growth.* IEEE Transactions on Software Engineering. SE-17(4):379-382,1991.

[92] B. Littlewood. A Semi-Markov Model for Software Reliability with Failure Costs. Proc. 9th International Symposium Comput. Software Engineering, pages 281-300, Plytechnic Institute of New York, April 1976.

[93] John D. Musa Software Reliability Engineering: More Reliable Software Faster and Cheaper, Second Edition , ISBN: 1-4184-9387 Authorhouse, 2004.

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

[95] S. Sanyal, V. Shah and S. Bhattacharya. "Framework of Software Reliability Engineering Tool". Proc. IEEE High-Assurance Systems Engineering Workshop (HASE'97), Washington DC, 1997. 1997 pp. 114-119.

[96] Jeff Tian, Li Ma, Zhao Li and A. Gunes Koru. .A Hierarchical Strategy for Testing Web-Based Applications and Ensuring Their Reliability", Proceedings of the 27th Annual International Computer Software and Applications Conference (COMPSAC 03). Computer IEEE 0730-3157/03.

[97] Jeff Tian. "Quality-Evaluation Models and Measurements". IEEE Software, May/June 2004 (Vol.21, No. 3).

[98] Sherif Yacoub, Bojan Cukic and Hany H. Ammar. A ScenarioBased Reliability Analysis Approach for Component-Based Software. IEEE Transactions on Reliability. Vol. 53, No. 4, December 2004.0018952904.

[99] James A. Whittaker y Michael G. Thomason, A Markov Chain Model for Statistical Software Testing, IEEE Transactions on Software Engineering, October 1994.

[100] James A. Whittaker Markov Analisis of Software Specifications. ACM Transactions on Software Engineering and Methodology, Vol. 2, No. 1, January 1993.

[101] Helene Le Guen, Raymond Marie, Thomas Thelin Reliability Estimation for Statistical Usage Testing Using Markov Chains IEEE ISSRE'04

[102] Pozniak Gorbatch, Alexander Semionovich. Self-learning control of finite Markov chains. ISBN: 082479429X (alk. paper)

[103] Cassandras, Cristos G., StÃ©ane Lafortune. Introducction to Discrete Event Systems. ISBN:0-7923-8609-4(alk paper)

[104] http:// www.mysql.com

[105] http:// www.apache.org

[106] http:// www.php.com

[107] http:// www.sun.com