INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

ESCOM

Trabajo Terminal

“Prototipo de editor de casos de uso para construcción asistida de casos de prueba”

2014 - B064

Presentan

Natalia Giselle Hernández Sánchez [(1)]

Sergio Ramírez Camacho [(2)]

Directores

M. en C. José Jaime López Rabadán *M. en C. Idalia Maldonado Castillo*

7 de diciembre 2015

INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO
SUBDIRECCIÓN ACADÉMICA

No. de TT: 2014-B064

7 de diciembre de 2015

Documento técnico

"Prototipo de editor de casos de uso para construcción asistida de casos de prueba"

Presentan

Natalia Giselle Hernández Sánchez (1)

Sergio Ramírez Camacho (2)

Directores

M. en C. José Jaime López Rabadán *M. en C. Idalia Maldonado Castillo*

RESUMEN

En este reporte se exponen las actividades realizadas para el desarrollo del *Prototipo de editor de casos de uso para la construcción asistida de casos de prueba*, PRISMA, el cual permite gestionar la información de algunos artefactos de software que intervienen en la documentación de un sistema, como lo son: Entidades, Reglas de negocio, Términos del glosario, Pantallas, Mensajes, Actores y Casos de uso. Así mismo, con base en la información obtenida, PRISMA permite generar propuestas de casos de prueba funcionales, facilitando así el proceso de realización de pruebas.

Palabras Clave: Documentación de Sistemas, Ingeniería de Pruebas, Ingeniería de Software, Proceso de Software.

(1) hdeznatali@gmail.com
(2) sramirezc@live.com

ESCUELA SUPERIOR DE CÓMPUTO
SUBDIRECCIÓN ACADÉMICA

DEPARTAMENTO DE FORMACIÓN INTEGRAL E INSTITUCIONAL

COMISIÓN ACADÉMICA DE TRABAJO TERMINAL

México, D.F. a 14 de diciembre de 2015

DR. FLAVIO ARTURO SÁNCHEZ GARFIAS
PRESIDENTE DE LA COMISIÓN ACADÉMICA
DE TRABAJO TERMINAL
PRESENTE

Por medio del presente, se informa que los alumnos que integran el TRABAJO TERMINAL: TT2014-B064, titulado "Prototipo de editor de casos de uso para la construcción asistida de casos de prueba" concluyeron satisfactoriamente su trabajo.

Los discos (DVDs) fueron revisados ampliamente por sus directores M. en C. José Jaime López Rabadán y M. en C. Idalia Maldonado Castillo y corregidos, cubriendo el alcance y el objetivo planteados en el protocolo original y de acuerdo a los requisitos establecidos por la Comisión que Usted preside.

ATENTAMENTE

M. en C. José Jaime López Rabadán

M. en C. Idalia Maldonado Castillo

Advertencia

Índice general

Índice de figuras

CAPÍTULO 1

Introducción

Actualmente existen herramientas que permiten automatizar diferentes actividades del proceso de software, esto permite algunas mejoras en la calidad y productividad del software. Estas actividades pueden ser ubicadas en alguna de las etapas del proceso software, que generalmente son: *Análisis*, *Diseño*, *Implementación* y *Pruebas*. El presente reporte expone las actividades realizadas y los resultados obtenidos de desarrollar una herramienta que permite asistir a la etapa de *Pruebas*, a través de la generación semiautomática de casos de prueba funcionales basados en la documentación de análisis.

1.1. Problemática

En la etapa de *Pruebas* resulta muy costoso en tiempo y recursos desarrollar las pruebas funcionales de un sistema debido a la cantidad de escenarios y validaciones que se deben de probar [1]. Además, las personas responsables de realizar las pruebas funcionales de un sistema requieren conocer el negocio, las validaciones y el comportamiento del sistema a través del estudio del documento de análisis o del sistema a probar.

Algunas de las desventajas de las herramientas que existen actualmente para la automatización de pruebas son:

- No reutilizan la información de los datos de entrada que se definieron en el documento de análisis.
- No reutilizan la información de las salidas esperadas que se describieron en el documento de análisis.
- No permiten reutilizar todas las validaciones descritas en las trayectorias de los casos de uso.
- No proveen un mecanismo que transforme las trayectorias de los casos de uso en casos de prueba funcionales.

Si existiera un base de datos con la información organizada del documento de análisis, sería posible automatizar el proceso de creación de casos de prueba funcionales y con ello posiblemente se podría disminuir el tiempo y los recursos necesarios para la etapa de *Pruebas*.

1.2. Propuesta

Se propone construir un prototipo de editor de casos de uso que permita capturar la información necesaria para la construcción de un documento de análisis, esto incluye mensajes, reglas de negocio, entidades, atributos, términos del glosario, actores, interfaces de usuario y casos de uso.

Con base en la información del documento de análisis el prototipo generará propuestas de casos de prueba funcionales.

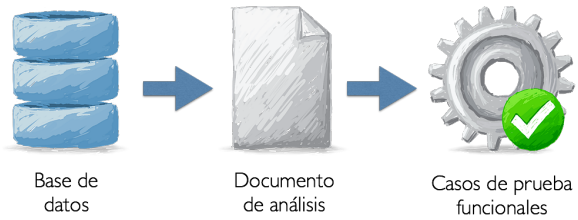

Figura 1.1: Propuesta

1.3. Objetivos

1.3.1. Objetivo general

Desarrollar un prototipo que permita la edición de casos de uso y la generación semiautomática de casos de prueba funcionales de un sistema de información que opere a través de la web, con la finalidad de asistir en las etapas de *Análisis* y *Pruebas* de software.

1.3.2. Objetivos específicos

Los objetivos específicos establecidos para el trabajo son:

1. Desarrollar un módulo que permita registrar: entidades, especificación de los atributos, reglas de negocio, interfaces, mensajes, perfiles de usuario y trayectorias de casos de uso.

2. Generar el documento de análisis de un sistema, considerando el paradigma orientado a objetos con base en la información obtenida en el módulo de captura.

3. Generar propuestas de casos de prueba funcionales de catálogos simples en un formato estándar para su uso con una herramienta de ejecución de pruebas con base en la información recolectada en el módulo de captura.

Todos los objetivos fueron alcanzados satisfactoriamente, en el **capítulo 4** se detallan los módulos desarrollados que permitieron alcanzar cada objetivo.

1.4. Estructura del documento

El presente documento está dirigido a los sinodales del Trabajo Terminal TTB064, retoma los objetivos descritos en el protocolo, considerando las observaciones realizadas en la primera evaluación del trabajo.

Se destaca que existe un documento adicional, denominado *Documento de análisis* donde se describen los casos de uso con los que se desarrolló cada uno de los módulos.

Los capítulos de este reporte son:

- **Capítulo 2. Antecedentes.** En este capítulo se exponen algunas herramientas para automatizar el proceso de software.
- **Capítulo 3. Metodología.** En este capítulo se describe el proceso que se llevó a cabo para la construcción del prototipo.
- **Capítulo 4. Módulos desarrollados.** En este capítulo se describen los módulos y submódulos desarrollados.
- **Capítulo 5. Algoritmos desarrollados.** En este capítulo se muestran los principales algoritmos utilizados en el prototipo.
- **Capítulo 6. Pruebas.** En este capítulo se muestran las pruebas realizadas al prototipo.
- **Capítulo 7. Resultados.** En este capítulo se exponen los resultados obtenidos del desarrollo del prototipo.
- **Capítulo 8. Conclusiones.** En este capítulo se expone un breve análisis sobre los resultados obtenidos y las experiencias adquiridas.
- **Capítulo 9. Trabajo a futuro.** En este capítulo se describe la posible continuación del presente trabajo.

CAPÍTULO 2

Antecedentes

Un proceso de software es un conjunto coherente de actividades que se llevan a cabo para la producción de un software. Los procesos de software son complejos y los intentos por automatizarlos han tenido un éxito limitado debido a su inmensa diversidad. Aunque existen muchos procesos diferentes, algunas etapas son comunes entre ellos [2], [1]:

- Análisis
- Diseño
- Implementación
- Pruebas

Si se considera que el costo total del desarrollo de un sistema de software es de 100 unidades de costo, la figura 2.1 muestra cómo se gastan estas en las diferentes actividades del proceso [2].

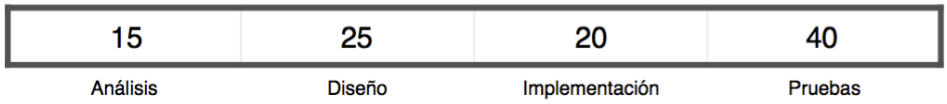

Figura 2.1: Distribución de costos

2.1. Etapa de Pruebas

La etapa de *Pruebas* consiste en verificar y validar que el software se ajusta a los requerimientos funcionales y no funcionales del sistema. Existen muchos tipos de pruebas y cada uno de ellos cuenta con un propósito específico.

2.1.1. Tipos de pruebas

Una clasificación para las pruebas de software, se conoce como niveles de prueba, y permite identificar las pruebas que se realizan según la etapa de desarrollo en la que se encuentre el sistema. Los niveles son:

- **Pruebas de componentes.** También conocidas como pruebas unitarias, buscan defectos en módulos o programas en un ambiente aislado del resto del sistema.
- **Pruebas de integración.** Buscan defectos entre el funcionamiento de 2 o más componentes.
- **Pruebas de sistema.** Buscan defectos desde un punto de vista integral.
- **Pruebas de aceptación.** No buscan encontrar defectos, sino otorgar confianza en el sistema frente a los clientes.

También es posible clasificar las pruebas según su objetivo [3], esta clasificación no es excluyente con la mostrada anteriormente, de hecho, los tipos que se presentan a continuación pueden ubicarse en cualquiera de los niveles anteriores.

- **Pruebas funcionales.** Su objetivo es demostrar que el sistema se ajusta a los requerimientos establecidos. Son pruebas referentes a "qué hace el sistema" y pueden aplicarse en cualquier nivel.
- **Pruebas de características no funcionales.** Incluye, pero no se limita, a comprobar: rendimiento, usabilidad, mantenibilidad y portabilidad. Son pruebas referentes a "cómo funciona el sistema" y pueden aplicarse en cualquier nivel.
- **Pruebas de estructura/arquitectura.** Están orientadas al funcionamiento interno del sistema y su objetivo es medir el grado en el que los casos de prueba han logrado abarcar los escenarios. Pueden aplicarse en cualquier nivel.
- **Pruebas de regresión.** Consisten en confirmar que se ha corregido un defecto de manera satisfactoria. Pueden aplicarse en cualquier nivel, y puede utilizar cualquier tipo de prueba.

2.1.2. Pruebas funcionales

El objetivo de las pruebas funcionales es demostrar que el software satisface los requerimientos establecidos. Estas pruebas también son conocidas como pruebas de entrega, y normalmente se encuentran en el conjunto de pruebas de caja negra, pues únicamente estudian el funcionamiento del sistema a través de un conjunto de entradas y un conjunto de salidas esperadas.

Según [2], la mejor aproximación para realizar pruebas de funcionalidad es utilizar la prueba basada en escenarios, en la que se idean diferentes escenarios y a partir de ellos se desarrollan diferentes casos de prueba. Para el caso de sistemas desarrollados a partir de un paradigma orientado a objetos, la creación de casos de prueba podría basarse en la especificación de casos de uso, así como en los diagramas de secuencia.

2.1.3. Pruebas automatizadas

Debido a que la etapa de pruebas presenta un costo muy alto, el desarrollo de herramientas para realizar pruebas automatizadas ha incrementado considerablemente, a continuación se muestra una tabla comparativa entre diferentes herramientas.

Herramienta	Pruebas funcionales	Bases de datos	Código abierto
Selenium	Sí	No	No
JMeter Rose	Sí	No	No
WebLOAD	Sí	No	Sí
Microsfot Word	No	No	No

Tabla 2.1: Herramientas para probar software

2.2. Etapa de Análisis

La etapa de *Análisis* consiste en identificar las necesidades del cliente y la funcionalidad que tendrá el sistema con base en los procesos del negocio para describir el comportamiento que tendrá el mismo, además de definir la información que será almacenada en una estructura de datos.

Las necesidades del cliente se transforman en requerimientos funcionales y no funcionales, estos describen el comportamiento del sistema, parte de este comportamiento es modelado mediante casos de uso (considerando el paradigma orientado a objetos). El producto de la etapa de Análisis es un documento donde se incluye la definición de los casos de uso, la cual requiere la descripción de entidades, especificación de los atributos, reglas de negocio, interfaces, mensajes y perfiles de usuario que intervienen en el sistema.

La etapa de *Análisis* es muy importante ya que en ella se define el comportamiento del sistema y los errores u omisiones generados durante esta etapa se propagarán a todas las demás etapas del proceso de construcción del software [2], [4].

Algunas de las herramientas que se utilizan para realizar la documentación de análisis, son:

Herramienta	UML	Bases de datos	Código abierto
Visual Paradigm	Sí	No	No
Rational Rose	Sí	No	No
Argo UML	Sí	No	Sí
Microsfot Word	No	No	No

Tabla 2.2: Herramientas para documentar software


ESCOM

CAPÍTULO 3

Metodología

En este capítulo se describen las actividades realizadas a lo largo del desarrollo del trabajo que permitieron alcanzar los objetivos planteados. PRISMA se dividió en dos incrementos:

1. Incremento 1. Editor de casos de uso
2. Incremento 2. Generador de casos de prueba

En este capítulo se describen las principales actividades realizadas para cada uno de ellos.

3.1. Incremento 1. Editor de casos de uso

En la figura 3.1 se muestran las principales actividades que se llevaron a cabo para la construcción del primer incremento del prototipo.

I. Análisis de diferentes documentos de análisis

Dado que las pruebas funcionales de un sistema pueden basarse en la especificación de casos de uso, se realizó un estudio de la documentación de casos de uso de diferentes sistemas, con el objetivo de conocer cómo se estructuraban y qué información requerían. El resultado de esta actividad permitió establecer la información con la que PRISMA operaría:

- Reglas de negocio
- Mensajes
- Actores
- Términos del glosario
- Entidades
- Pantallas

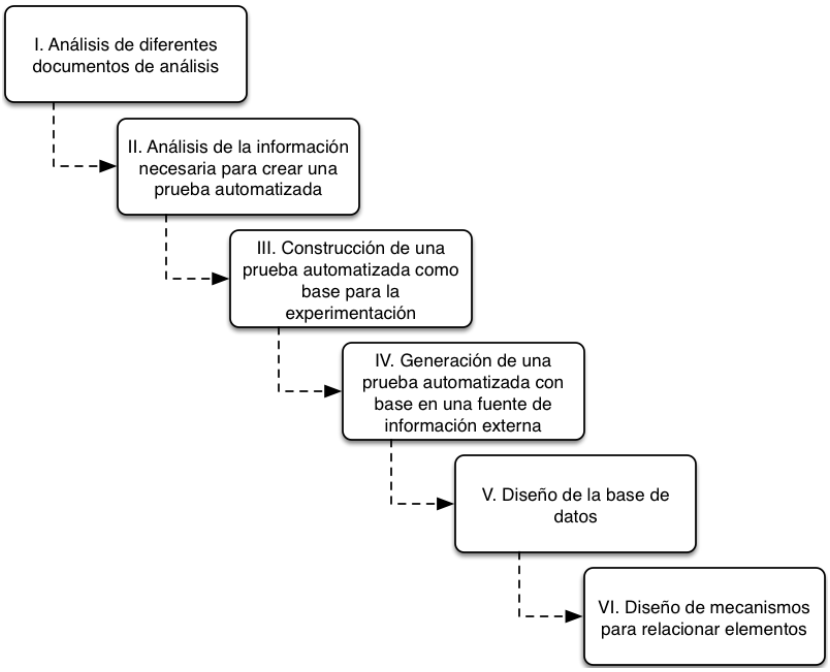

Figura 3.1: Metodología: Editor de casos de uso

II. Análisis de la información necesaria para crear una prueba automatizada

Posteriormente se estudió la estructura y la información necesaria para construir un caso de prueba funcional, esperando que los resultados obtenidos presentaran una estrecha relación con los resultados del análisis de la estructura e información necearia para un caso de uso.

Los resultados obtenidos fueron positivos, ya que gran parte de la información requerida para construir un caso de prueba funcional, coincidía con la información necesaria para la construcción de un caso de uso.

III. Construcción de una prueba automatizada como base para la experimentación

Con los resultados obtenidos anteriormente, se intentó realizar una prueba automatizada, utilizando exclusivamente la información proporcionada por la especificación de casos de uso, el resultado de este ejercicio mostró que aunque la información que proporciona este documento es bastante útil, no es suficiente para la construcción de la prueba.

IV. Generación de una prueba automatizada con base en una fuente de información externa

Debido a que la prueba automatizada creada anteriormente fue construida directamente sobre una herramienta de pruebas, se realizó un ensayo para demostrar que era posible crearla de forma semiautomática desde una herramienta externa. Para ello se implementó un breve sistema, en el cual se solicitaba información referente a la prueba, para posteriormente utilizarla y generar la prueba

automatizada. Los resultados fueron satisfactorios, se demostró que era posible generar pruebas desde una herramienta externa para una posible ejecución con alguna herramienta de pruebas.

V. Diseño de la base de datos

Con el análisis realizado anteriormente, se propuso como primer incremento la construcción de un editor de casos de uso que permitiera gestionar diferentes elementos contenidos en la documentación de análisis de un sistema. Con ello se diseñó una base de datos que permitiera alojar la información y que además modelara la gran cantidad de relaciones que tiene un caso de uso con diferentes elementos del documento análisis.

VI. Diseño de mecanismos para relacionar elementos

Como se mencionó anteriormente, un caso de uso se encuentra interrelacionado con una gran cantidad de elementos incluidos en el documento de análisis, por lo que resultó importante establecer mecanismos que elevaran la usabilidad al sistema. En general, este mecanismo consiste en poder seleccionar elementos previamente registrados en medio de una redacción abierta. En el **capítulo 5** se muestra a detalle este funcionamiento.

3.2. Incremento 2. Generador de casos de prueba

En la figura 3.2 se muestran las principales actividades que se llevaron a cabo para la construcción del segundo incremento del prototipo.

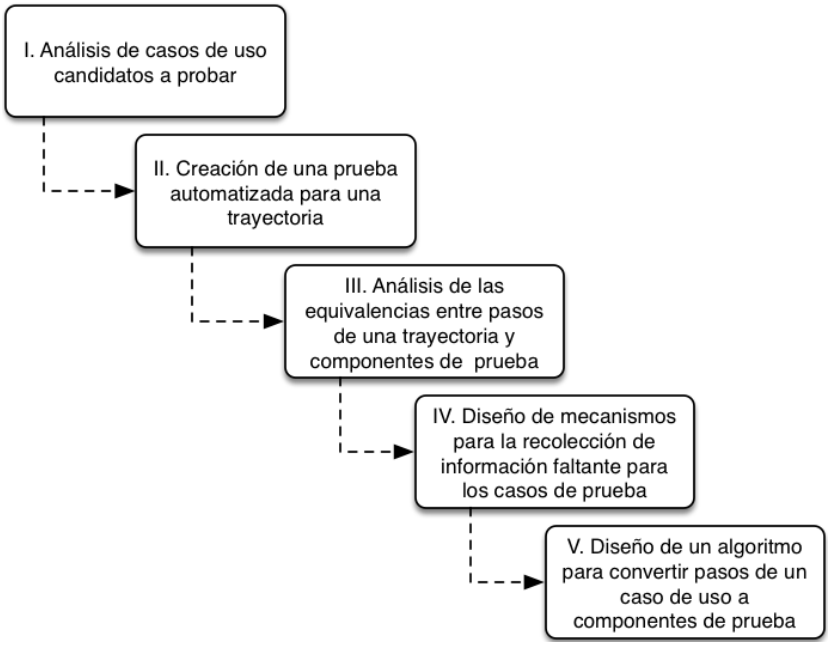

Figura 3.2: Metodología: Generador de casos de prueba

I. Análisis de las trayectorias a probar

Dado que el comportamiento de un caso de uso puede modelar un conjunto muy amplio de escenarios, se realizó un análisis de diferentes trayectorias, buscando comportamientos en común y altamente repetitivos.

El resultado de este análisis permitió definir un modelo general de trayectorias candidatas para la creación de casos de prueba funcionales. Posteriormente se analizaron nuevamente diferentes trayectorias, pero esta vez basando la búsqueda en el modelo obtenido del análisis anterior, esto con la finalidad de identificar claramente qué casos de uso realmente eran candidatos y con ello definir un vector de pruebas.

II. Creación de una prueba automatizada para una trayectoria

Con un espacio de búsqueda reducido, se escogió la trayectoria de un caso de uso para utilizarlo como base para la generación de casos de prueba muestra. Se buscó que el caso de uso presentara la mayor cantidad de escenarios para así observar los componentes de prueba necesarios para cada uno de ellos. Este ejercicio reforzó el análisis realizado durante el primer incremento para determinar la información necesaria para construir la prueba automatizada y permitió definir claramente qué información no aportaba el editor para la construcción de la prueba.

III. Análisis de las equivalencias entre pasos de un caso de uso y componentes de prueba

Con la prueba creada, el siguiente paso consistió en analizar a detalle la relación existente entre los pasos de una trayectoria y los componentes de prueba. Para ello se estudió cada paso y se establecieron diferentes categorias, a las cuales a su vez se les asocio un conjunto de componentes de prueba, por ejemplo, para un paso que consiste en oprimir un botón, se le asocio una petición HTTP.

IV. Diseño de mecanismos para la recolección de información faltante para los casos de prueba

La siguiente tarea consistió en establecer la manera en que la información faltante sería solicitada al usuario, entre los datos que el editor de casos de uso no aportaba se encuentran por ejemplo los nombres de los input de los formularios. Por otro lado, para el caso de los datos de entrada, se diseñaron mecanismos para su generación automática, con base en la especificación registrada en el editor de casos de uso.

Los mecanismos utilizados para solicitar información no resultaron triviales debido a que gran parte de la información depende de la especificación del caso de uso, es por ello que su comportamiento es completamente dinámico.

V. Diseño de un algoritmo para convertir pasos de una trayectoria a componentes de prueba

Finalmente, con el análisis de las equivalencias entre pasos y componentes de prueba, se prosiguió a diseñar un algoritmo que permitiera transformar los pasos de una trayectoria a casos de prueba funcionales.

Inicialmente se propuso realizar una traducción lineal de los pasos, es decir, encontrar un paso y en ese mismo momento transformarlo a componentes de prueba, sin embargo con esa idea no resultó posible generar la prueba de forma automática pues la transformación dependía del contexto. Debido a

esto, se implementó un algoritmo que permitiera realizar esta conversión. En el **capítulo 5** se muestra a detalle el funcionamiento del algoritmo.

CAPÍTULO 4

Módulos desarrollados

4.1. Módulo 1. Editor de casos de uso

Este módulo lleva por nombre "Editor de casos de uso", y permite gestionar proyectos, módulos, reglas de negocio, mensajes, términos del glosario, actores, entidades, pantallas y casos de uso.

Con base en esta información, este módulo también permite generar el documento de análisis del sistema, además implementa un proceso para el seguimiento de un caso de uso, así como la gestión de los participantes del proyecto.

En la figura 4.1 se muestran los submódulos que componen al "Editor de casos de uso".

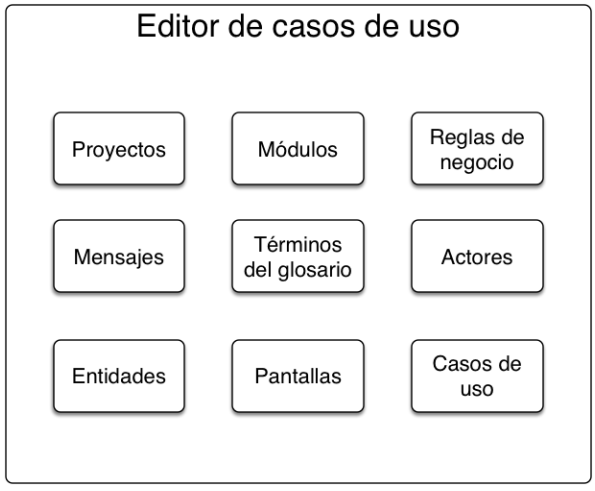

Figura 4.1: Módulo 1. Editor de casos de uso

Proyectos

Un proyecto, sirve para organizar los diferentes elementos que se gestionan a través del editor de casos de uso.

- Registrar, modificar y elminar proyectos (Administrador).
- Seleccionar el personal participante para el proyecto (Líder de proyecto).

Módulos

Este módulo permite:

- Registrar, modificar y eliminar módulos (Analista).

Una vez registrado un módulo es posible gestionar casos de uso y pantallas.

Reglas de negocio

Este módulo permite:

- Registrar, modificar, consultar y eliminar regla de negocio (Analista).

Mensajes

Este módulo permite:

- Registrar, modificar y eliminar mensajes (Analista).
- Registrar y modificar parámetros (Analista).

Términos del glosario

Este módulo permite:

- Registrar, modificar, consultar y eliminar términos del glosario (Analista).

Actores

Este módulo permite:

- Registrar, modificar, consultar y eliminar actores (Analista).

Entidades

Este módulo permite:

- Registrar, modificar, consultar y eliminar entidades (Analista).
- Registrar, modificar y eliminar atributos (Analista).

Pantallas

Este módulo permite:

- Registrar, modificar, consultar y eliminar pantallas (Analista).
- Registrar, modificar y eliminar acciones (Analista).

Casos de uso

Este módulo permite:

- Registrar, modificar, consultar y eliminar casos de uso (Analista).
- Registrar, modificar y eliminar trayectorias (Analista).
- Registrar, modificar y eliminar pasos (Analista).
- Registrar, modificar y eliminar puntos de extensión (Analista).
- Revisar casos de uso (Analista).
- Liberar casos de uso (Líder).
- Corregir casos de uso (Analista).

4.2. Módulo 2. Generador de casos de prueba

Este módulo lleva por nombre "Generador de casos de prueba", y permite:

- Generar en un archivo con la estuctura de un caso de prueba funcional.
- Generar los archivos correspondientes a los datos de entrada de prueba.

Para ello se requiere contar con la especificación completa de un caso de uso y configurar un conjunto de parámetros para la prueba.

El Generador de casos de prueba, se encarga de traudcir los pasos de un caso de uso a componentes de prueba particulares para JMeter. Este caso de prueba funcional consiste en un conjunto de archivos, uno de ellos describe la estructura de la prueba (extensión .jmx) y el resto contiene las entradas que serán utilizadas (extensión .csv), estos ficheros serán descargados en el equipo cliente para que la prueba pueda ser ejecutada con JMeter.

CAPÍTULO 5

Algoritmos desarrollados

Descripción del algoritmo Referencias a parámetros

En el *Módulo 1 Editor de casos de uso* se implementó un mecanismo que permite seleccionar algunos elementos previamente registrados en medio de una redacción abierta. Este mecanismo surge de la necesidad de poder agilizar el uso de elementos como reglas de negocio o mensajes mientras se redacta un caso de uso, además de mantener consistente la información previniendo futuros cambios en los elementos utilizados.

Cuando el usuario escribe algún token (*RN·*, *ENT·*, *CU·*, *MSG·*, etc.) en un campo de texto válido, el sistema muestra una lista ubicada en la posición actual del cursor, esta lista muestra diferentes resultados según el token utilizado, por ejemplo, para el token *MSG·* el sistema mostraría una lista de mensajes. Posteriormente, cuando el usuario elige algún elemento de la lista, el sistema se encarga de incluir una cadena de texto relativa al elemento seleccionado. Cuando el usuario decide guardar la información, el sistema ejecuta un algoritmo previo al almacenamiento, que permite establecer relaciones lógicas con los elementos seleccionados y así mantener la información consistente ante cualquier cambio, a este algoritmo se le llamó **Referencias a parámetros**. Para explicar el funcionamiento, se utilizará como ejemplo la redacción de un paso que hace referencia a un mensaje y a una pantalla.

1. Inicialmente el algoritmo se encarga de buscar los tokens utilizados en la cadena en cuestión, respondiendo con un conjunto de cadenas correspondientes a los tokens ingresados, como se muestra en la figura 5.1.

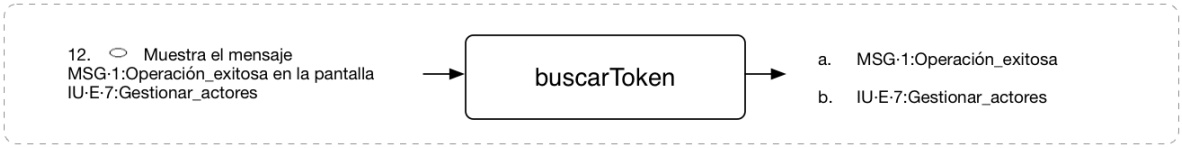

Figura 5.1: Algoritmo Referencias a parámetros: Buscar token

2. Para cada token encontrado se realiza una segmentación con base en una determinada estructura, esta segmentación permite obtener todos los segmentos utilizados en el token. El primer segmento corresponde al tipo de elemento seleccionado y el resto de la estructura es determinada por el tipo encontrado. En la figura 5.2 se muestra el resultado que se obtendría de segmentar el token para un mensaje.

Figura 5.2: Algoritmo Referencias a parámetros: Segmentar token

3. Cuando el token se encuentra segmentado, se busca el objeto al que hace referencia, en este ejemplo se busca el mensaje número "1" cuyo nombre es "Operación exitosa". En la figura 5.3 se ilustra el resultado de esta búsqueda con base en los segmentos.

Figura 5.3: Algoritmo Referencias a parámetros: Buscar objeto

4. Con base en el objeto encontrado se crea una referencia, la cual cuenta con un objeto origen y un objeto destino, en este ejemplo, el *paso número 12* y el *mensaje número 1* respectivamente, como se muestra en la figura 5.4. Esta referencia permite identificar a nivel base de datos las relaciones que existen entre determinados elementos del sistema, permitiendo mantener únicamente una referencia lógica entre ellos, por lo que si uno de ellos actualiza su información, el cambio se verá reflejado también en los elementos con los que se encuentra relacionado.

Figura 5.4: Algoritmo Referencias a parámetros: Crear referencia

5. Finalmente la cadena original es codificada para preservar en el texto únicamente los identificadores de los elementos a los que se hicieron referencia, como se muestra en la figura 5.5. Despúes de este procesamiento, las siguientes tareas para el sistema consisten únicamente en el almacenamiento de toda la información.

Figura 5.5: Algoritmo Referencias a parámetros: Codificar cadena

Descripción del algoritmo Generador de casos de prueba

Cuando se desea generar un conjunto de casos de prueba, se requiere configurar una serie de parámetros que son determinados por la estructura del caso de uso a probar, a este caso de uso se le llamará caso de uso base. Para determinar qué parámetros deben ser configurados, el sistema establece una ruta de los casos de uso que deben ser ejecutados para llegar al caso de uso base, en el ejemplo de la figura 5.6 se muestra la ruta necesaria para ejecutar el caso de uso "Registrar persona". Se asume que cada caso de uso que forme parte de la ruta debe ser configurado para poder probar el caso de uso base.

Cuando esta configuración es realizada, se procede a configurar el caso de uso base, parte de esta configuración consiste en la generación automática de las entradas, con base en la especificación de cada atributo y en las reglas de negocio utilizadas. Finalemente, el sistema genera un conjunto de archivos en los que se definen los casos de prueba funcionales, estos archivos son descargados en el equipo cliente para que pueda ejecutar la prueba con JMeter.

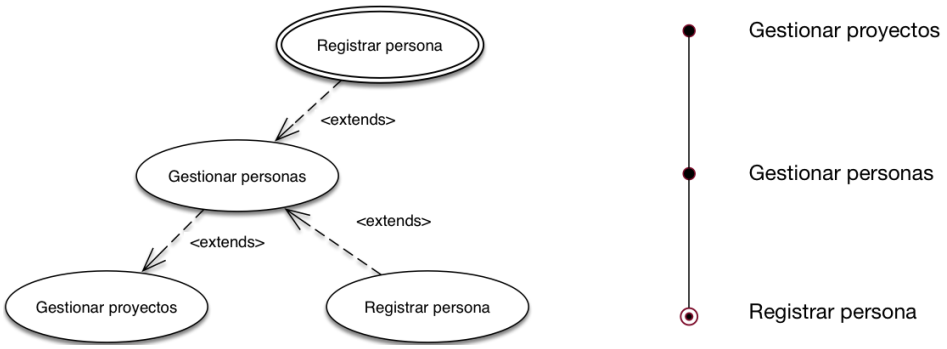

Figura 5.6: Algoritmo Generador de casos de prueba: Buscar ruta

El algoritmo **Generador de casos de prueba** se encarga de analizar los pasos de cada caso de uso y generar su equivalente en componentes de prueba y así generar los casos de prueba funcionales que serán descargados al cliente. A continuación se explica el funcionamiento:

1. Para iniciar el algoritmo es necesario especificar el conjunto de pasos correspondientes a la trayectoria principal y el paso a analizar, a este último se le llamará *paso A*.
2. Se determina el tipo del *paso A*, los tipos pueden ser:
   - Oprimir botón
   - Mostrar pantalla
   - Mostrar mensaje
   - Validar regla de negocio

   Para determinar el tipo del paso, se considera quién realiza el paso, qué verbo utiliza y qué referencias tiene.
3. Se calcula cuál es el siguiente paso, que a partir de ahora se le llamará *paso B*.
4. Si el *paso A* es de tipo "Oprimir botón" se deben considerar los siguientes casos:

a) Si el *paso B* es de tipo "Validar regla de negocio", se deben crear una serie de peticiones HTTP con la indicación de que el conjunto de entradas a utilizar, está definido por la regla de negocio empleada. Cada tipo de regla de negocio establece un funcionamiento particular:

- **Datos obligatorios.** Indica que deben generarse tantas peticiones HTTP, como entradas requeridas tenga el caso de uso. Cada petición HTTP especifica un conjunto de entradas en el que una de ellas se omite para validar el correcto funcionamiento de la aplicación. Por ejemplo, para un caso de uso con 2 entradas requeridas, esta regla de negocio generaría los escenarios que se muestran en la imagen 5.7.

- **Verificación de catálogos.** Indica que se debe generar una petición JDBC que permita extraear información mínima para el sistema, con el objetivo de anticipar si es que el sistema debe operar normalmente o debe alertar sobre la falta de información necesaria para operar.

- **Unicidad de elementos.** Indica que se debe generar una petición JDBC que permita extraer la información de algún registro en particular, con el objetivo de anticipar si es que el sistema debe alertar sobre un posible duplicado u operar normalmente.

- **Tipo de dato correcto.** Indica que deben generarse tantas peticiones HTTP, como entradas de tipo entero, fecha, booleano o flotantes tenga el caso de uso. Cada petición especifica un conjunto de entradas en el que cada una de ellas se genera con un tipo incorrecto para validar el correcto funcionamiento de la aplicación.

- **Longitud correcta.** Indica que deben generarse tantas peticiones HTTP, como entradas de tipo cadena, entero o flotante tenga el caso de uso. Cada petición HTTP especifica un conjunto de entradas en el que cada una de ellas se genera con más caracteres de los permitidos para validar el correcto funcionamiento de la aplicación.

Inicialmente se había planteado considerar reglas de negocio referentes a operaciones aritméticas, sin embargo no se consideraron debido a que para lograr comprender un tipo de regla de negocio como este se requería más tiempo del establecido para desarrollar el módulo.

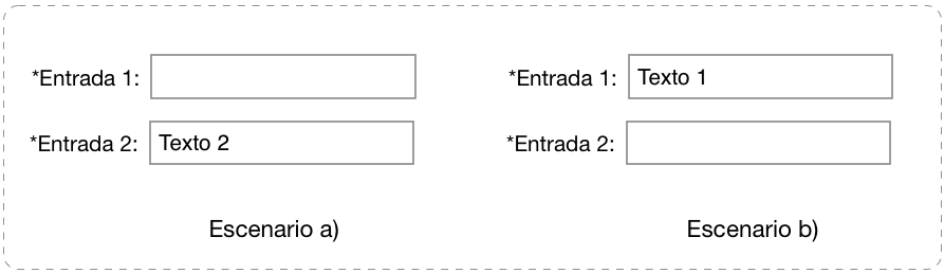

Figura 5.7: Algoritmo Generador de casos de prueba: Datos obligatorios

Posteriormente se elimina el *paso B* del conjunto de pasos y se repite el proceso, indicando que el *paso A* continúa siendo el *paso A*. Esto permite continuar analizando las posibles reglas de negocio que intervienen con la petición HTTP.

b) Si el *paso B* no es de tipo "Validar regla de negocio" simplemente se crea una petición HTTP con un conjunto de entradas válidas, se elimina el *paso A* del conjunto de pasos y se repite el proceso, indicando que el nuevo *paso A* ahora es el *paso B*. Esto permite continuar analizando los siguientes pasos de la trayectoria.

5. Si el *paso A* es de tipo "Mostrar pantalla" se debe crear una aserción con base en el patrón definido para la pantalla. Posteriormente se elimina el paso y se repite el proceso, indicando que el nuevo *paso A* ahora es el *paso B*.

6. Si el *paso A* es de tipo "Mostrar mensaje" se debe crear una aserción con base en el patrón definido por la redacción del mensaje. Posteriormente se elimina el paso y se repite el proceso, indicando que el nuevo *paso A* ahora es el *paso B*.

7. Si el tipo del *paso A* no está especificado, se elimina y se repite el proceso, indicando que el nuevo *paso A* ahora es el *paso B*.

8. Si el *paso A* no existe, se termina el proceso.

CAPÍTULO 6

Pruebas

En este capítulo se describen las pruebas realizadas por cada incremento:

- Editor de casos de uso
- Generador de casos de prueba

Las pruebas que se llevaron a cabo permitieron comprobar el funcionamiento de la herramienta, así como detectar los diferentes errores de todo el proceso de desarrollo. Los errores se clasificaron en errores de funcionalidad, excepciones, errores visuales y mejoras al sistema.

6.1. Editor de casos de uso

Para probar el *Editor de casos de uso* se realizaron registros de diferentes elementos: actores, mensajes, reglas de negocio, entidades, atributos, términos del glosario, pantallas y casos de uso de dos sistemas distintos:

- E-Commerce
- PRISMA

En la tabla 6.1 se muestra la cantidad de elementos registrados con la herramienta en relación al total de elementos pertenecientes al sistema PRISMA.

Elemento	Total registrado
Casos de uso	9 / 83
Pantallas	10 / 70
Mensajes	10 / 36
Entidades	6 / 27
Reglas de negocio	10 / 26
Actores	3 / 3
Términos del glosario	1 / 30

Tabla 6.1: Elementos de PRISMA

En la tabla 6.2 se muestra el total de elementos registrados en la herramienta pertenecientes a E-Commerce.

Elemento	Total registrado
Casos de uso	4
Pantallas	5
Mensajes	9
Entidades	2
Reglas de negocio	6
Actores	2
Términos del glosario	0

Tabla 6.2: Elementos de E-Commerce

Resultados

Con base a los registros realizados fue posible generar la base de datos y el documento de análisis de manera satisfactoria para los dos sistemas.

Al realizar las pruebas mencionadas, se detectaron diferentes tipos de errores. Los resultados de las pruebas realizadas al *Editor de casos de uso* se muestran en la tabla 6.3, donde también se muestra el total de errores corregidos.

Tipo de error	Total errores detectados	Total errores corregidos
Excepción	3	3

Tipo de error	Total errores detectados	Total errores corregidos
Funcionalidad	59	50
Visual	12	11
Mejora al sistema	2	0

Tabla 6.3: Resultados de las pruebas del Editor de casos de uso

6.2. Generador de casos de prueba

Para probar el *Generador de casos de prueba* se utilizaron algunos casos de uso de los registrados en las pruebas del editor. Los casos de uso que se probaron pertenecen a los sistemas:

- E-commerce
- PRISMA

Desarrollo de la prueba realizada al caso de uso CUE2.1 Registrar persona

A continuación se describirá a detalle el desarrollo de una prueba realizada al *Generador de casos de prueba*, con la finalidad de mostrar el número casos de prueba que se pueden generar, mostrar el tiempo de generación de los casos de prueba, así como mostrar la cantidad de valores que el usuario debe cambiar directamente en JMeter para completar al 100 % una prueba, de un caso de uso con determinadas características.

Características del caso de uso

Esta prueba consiste en la configuración y generación de los casos de prueba para el caso de uso **CUE2.1 Registrar persona**, el cual posee las siguientes características:

- Tiene un total de 6 entradas de tipo cadena, de las cuales 5 son obligatorias y una es opcional.
- Las reglas de negocio que son validadas son: Datos obligatorios, unicidad de parámetros, tipo de dato correcto y longitud correcta.
- Los casos de uso previos que deben ser configurados son: CUE13 Iniciar sesión, CUE1 Gestionar proyectos de Administrador y CUE2 Gestionar personal.

Total de casos de prueba generados

Como resultado de esta configuración se generaron 18 casos de prueba, para completar la prueba fue necesario ingresar 6 valores directamente en JMeter, estos valores no podían ser ingresados a la configuración.

Tiempo de respuesta

Se solicitó 5 veces la generación de la prueba, el tiempo promedio de respuesta del sistema fue de 4.26 segundos.

Resultados

De las pruebas realizadas al generador, los resultados de las se muestran en la tabla 6.4, donde también se muestra el total de errores corregidos.

Tipo de error	Total errores detectados	Total errores corregidos
Excepción	3	3
Funcionalidad	25	17
Visual	13	12
Mejora al sistema	2	0

Tabla 6.4: Resultados de las pruebas del Generador de casos de prueba

CAPÍTULO 7

Resultados

En este capítulo se describen los resultados más relevantes que se obtuvieron a lo largo del desarrollo del proyecto.

7.1. Prototipo

El desarrollo del proyecto permitió la construcción de PRISMA, un prototipo de editor de casos de uso que permite construir casos de prueba con base en la información del análisis. Algunas de las características de la herramienta son:

- Permite recabar los elementos que componen un documento de análisis.
- PRISMA permite la interacción de tres perfiles de usuario distintos.
- El sistema permite generar el documento de análisis en dos formatos distintos: pdf y docx.
- El Editor de casos de uso construye la base de datos de manera adecuada.
- PRISMA permite modelar el proceso de revisión de los casos de uso, así como gestionar las actividades de los colaboradores.

Algunas de las características se detallan a continuación.

Registro de elementos

PRISMA permite gestionar los elementos que describen el comportamiento de un sistema y que son recabados en la etapa de análisis: mensajes, reglas de negocio, entidades, actores, términos del glosario, pantallas y casos de uso. Con el registro de los elementos mencionados es posible modelar las diversas estructuras de información de múltiples sistemas web.

Gestión de las actividades de los colaboradores

El diseño de la herramienta soporta tres perfiles de usuario diferentes:

- Administrador
- Líder de análisis
- Analista

Estos tres perfiles permiten que existan usuarios con diferentes responsabilidades y niveles de acceso. El Administrador es el encargado de gestionar los proyectos, así como el personal de la organización. El Líder de análisis y el Analista participan en el proceso de revisión de los casos de uso siendo cada uno colaborador de los proyectos.

7.2. Documento de análisis

Se construyó un documento de análisis que contiene las especificaciones del comportamiento de PRISMA. Dentro de este documento se describen los siguientes elementos:

- Modelo de negocio
- Modelo de comportamiento
- Modelo de interacción con el usuario

7.3. Fórmula para el cálculo de los casos de prueba generados por el sistema

Un resultado interesante fue la fórmula que permite calcular el número de casos de prueba que pueden ser generados por PRISMA, el resultado del cálculo como se observa en la figura 7.1 depende de las reglas de negocio asociadas al caso de uso a probar, así como el tipo y la cantidad de entradas asociadas.

$$NCP = 1 + RNVC + RNDI\ (B + E + FL + FE) + RNLI\ (C + E + FL) + RNDO \cdot DO$$

Donde:

- NCP = Número de casos de prueba
- RNVC = Número de reglas de negocio de verificación de catálogos
- RNDI = Número de reglas de negocio de datos incorrectos
- B = Número de entradas de tipo booleano
- E = Número de entradas de tipo entero
- FL = Número de entradas de tipo flotante
- FE = Número de entradas de tipo fecha
- RNLI = Número de reglas de negocio de longitud inválida
- C = Número de entradas de tipo cadena
- RNDO = Número de reglas de negocio de datos obligatorios
- DO = Número de entradas obligatorias

Figura 7.1: Fórmula para calcular los casos de prueba generados

CAPÍTULO 8

Conclusiones

Las pruebas de software con frecuencia representan un costo muy elevado en relación al invertido en las demás actividades del proceso de software. Debido a ello, se han realizado diversos intentos por automatizar su proceso, como lo son herramientas de software o metodologías de pruebas.

El desarrollo del presente trabajo demostró que es posible automatizar las pruebas de software a través de mecanismos semiautomáticos que reutilizan la información de la documentación de análisis.

Debido a que la especificación de casos de uso es realizada por diferentes analistas, con frecuencia se encuentran diferentes redacciones para un mismo objetivo, lo cual complica bastante el entendimiento de la información para una computadora, por lo que analizar los pasos de un caso de uso y traducirlos a componentes de pruebas resulta altamente complejo.

Por otro lado, se identificó que tener la información de la documentación de análisis en una base de datos permite mantener la información organizada y centralizada. En general el contar con una base de datos de esta información abre las puertas a otros sistemas para automatizar algún proceso dentro del desarrollo de un software, en nuestro caso, por ejemplo, permitió automatizar la generación de casos de prueba funcionales.

CAPÍTULO 9

Trabajo a futuro

A lo largo del desarrollo del prototipo se identificaron diferentes áreas de oportunidad para el prototipo, a continuación se explica cada una de ellas:

Soporte para reglas de negocio con base en algún estándar

Actualmente el sistema únicamente brinda soporte para un conjunto limitado de reglas de negocio. Otorgar soporte para reglas de negocio con base en un estándar, permitiría al prototipo identificar con mayor facilidad el funcionamiento descrito en la regla de negocio, y con ello el espacio de escenarios válidos para la generación de casos de prueba aumentaría.

Crear un estándar para la redacción de los casos de uso

Dado que el idioma español es ambiguo, resulta muy complicado comprender de manera explícita lo que indican los pasos de un caso de uso, frecuentemente para comprenderlo se recurre al contexto, sin embargo para un sistema informático esta tarea no es trivial. Crear un estándar que establezca la manera en que se debe redactar un caso de uso, aportaría una mejor comprensión de las redacciones y con ello el espacio de escenarios a probar también podría crecer.

Automatizar el proceso de configuración de una prueba

Para que el prototipo genere casos de prueba funcionales, es necesario que el usuario indique los valores de una gran cantidad de parámetros. Una parte de estos valores suelen localizarse en diferentes artefactos de software desarrollados durante la etapa de *Implementación*. Diseñar un mecanimo para analizar estos artefactos de software, permitiría automatizar el proceso de configuración de la prueba, por lo que se podría reducir considerablemente el tiempo dedicado a esta tarea.

Bibliografía

[1] R. Pressman, *Software Engineering: A Practitioner's Approach*, ser. McGraw-Hill series in computer science. Boston, 2005.

[2] I. Sommerville, *Software Engineering*, ser. International Computer Science Series. Addison-Wesley, 1992.

[3] *Foundation Level Syllabus*, 2011. [Online]. Available: http://www.istqb.org/downloads/finish/16/15.html

[4] K. Kendall, J. Kendall, and A. Ramos, *Análisis y diseño de sistemas.* Pearson Educación, 2005.