ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA.

Desarrollo de una herramienta web de apoyo a la ejecución de experimentos en pruebas de software.

TRABAJO DE TITULACIÓN

PREVIA LA OBTENCIÓN DEL TÍTULO DE:

INGENIERO EN SISTEMAS INFORMÁTICOS

Presentado por:

GLORIA JIMENA HERNÁNDEZ GUATO

RIOBAMBA-ECUADOR

2017

Agradezco a Dios y a mi madre por brindarme sus bendiciones y apoyo incondicional, a mi hermana y amigos por estar conmigo cuando los necesité, a mis pequeñas hijas y a mi amado esposo por su paciencia y cariño, a la ESPOCH institución que me abrió sus puertas y, mis queridos maestros quienes permitieron mi formación académica, de manera especial al director y miembro de tesis, que con sus conocimientos fueron una guía en el desarrollo de la misma; pues gracias a todos ellos he luchado constantemente para cumplir mis metas, y hacer posible este sueño.

TABLA DE CONTENIDOS

Contenido

INDICE DE TABLAS

INDICE DE GRÁFICOS

RESUMEN

En el presente trabajo, se ha desarrollado un sistema web como herramienta de apoyo a la experimentación de software en el área de las pruebas de software. ……..

SUMMARY

INTRODUCCIÓN

Actualmente, “Hay una creciente comprensión en la Ingeniería de Software, que los estudios empíricos son necesarios para desarrollar o mejorar procesos, métodos y herramientas para el desarrollo y mantenimiento de software” (1).

La experimentación en ingeniería de software, permite identificar las relaciones causa-efecto, mediante la realización de experimentos controlados donde sólo unas pocas variables cambian. “Experimentos controlados en Ingeniería de software involucran a menudo a los estudiantes que resuelven tareas de papel en un entorno de aula (2).

Como apoyo a la experimentación en ingeniería de software ya existen algunas herramientas que se han desarrollado las cuales se encuentran en el mercado como: SimGrid, Sese y Marmoset.

SimGrid es un conjunto de herramientas que proporciona funcionalidades básicas para la simulación de aplicaciones distribuidas en entornos distribuidos heterogéneos. El objetivo específico del proyecto es facilitar la investigación en el área de sistemas paralelos y distribuidos que sean a gran escala, tales como Grids, sistemas P2P y Cloud (3).

Sin embargo SESE es una herramienta para ejecutar experimentos controlados realistas, para evaluar empíricamente las tecnologías de ingeniería de software. El realismo puede ser incrementado utilizando profesionales además de estudiantes, las herramientas reales de desarrollo, en lugar de papel y lápiz, tareas más grandes y un entorno de trabajo típico en lugar de un aula (4).

Una de las áreas en ingeniería de software, donde se han realizado experimentos son las pruebas de software, las cuales están diseñadas para realizar la detección de defectos de forma estructurada y facilitar el estudio de software.

Marmoset, es una herramienta que apoya a los docentes de informática que mantienen fuertes opiniones sobre el enfoque correcto, para la enseñanza de la programación a nivel introductorio (5).

Las pruebas de software ayudan a proporcionar información objetiva e independiente sobre la calidad del producto software (6).

Algunas técnicas para aplicar pruebas de software son: La lectura de código por abstracción sucesiva, pruebas funcionales “caja negra” y pruebas estructurales “caja blanca”.

Aunque existen algunas herramientas de soporte a la experimentación en ingeniería de software, no se han encontrado herramientas de apoyo a la comparación de técnicas de pruebas de software, es por ello que los experimentos en esta área suelen realizarse de forma manual.

Al realizar experimentos de software de manera manual, existe una pérdida de tiempo y la falta de control en la recolección de métricas, por lo que no se conoce como influye el uso de un sistema web para apoyar la realización de experimentos en el área de pruebas de software.

Para la sistematización del problema se propone desarrollar un sistema web para apoyar la realización de experimentos en el área de las pruebas de software, por lo cual es necesario realizar las siguientes preguntas: ¿Qué es la experimentación en ingeniería de software?, ¿Qué herramientas de soporte a la experimentación en ingeniería de software existen?, ¿Qué son las pruebas de software?, ¿Qué técnicas de pruebas de software existen?, ¿Cuáles son las variables y métricas de las técnicas de pruebas de software que se recolectarán?.

JUSTIFICACIÓN DEL TRABAJO DE TITULACIÓN

JUSTIFICACIÓN TEÓRICA

“La ingeniería de software no es diferente de otras ciencias, este campo ha ido evolucionando para adoptar aproximaciones experimentales” (6), y “la ingeniería de software está preocupada sobre los modelos apropiados para ser aplicados al desarrollo de sistemas de software” (7), mientras más grandes son los sistemas de software se requiere de varios programadores, profesionales, largos periodos de tiempo, esto significa grandes cantidades de recursos económicos, es por ello que a través de la experimentación en Ingeniería de Software, se puede tener mayor conocimiento sobre el proceso de desarrollo de software.

El aula es un entorno donde se puede conducir experimentos en ingeniería de software, por varias razones:

- La mayoría de investigadores están localizados en universidades.
- La formación del curso puede ser integral.
- Las tareas requeridas pueden ser fácilmente integradas.
- Los temas están relacionados en tareas de programación idénticas, lo cual generalmente no es cierto en la industria.

Los resultados obtenidos directamente de estos estudios, son útiles para determinar reglas y diseños experimentales, los cuáles más tarde pueden ser aplicados a temas de la industria.

Para la realización del Sistema Web se utilizará el lenguaje de programación PHP, AJAX (Asynchronous JavaScript And XML) que es una técnica de desarrollo web para crear aplicaciones interactivas o RIA, además se utilizará la metodología SCRUM.

JUSTIFICACIÓN METODOLÓGICA

Se conoce la existencia de varias metodologías web, sin embargo al desarrollar una aplicación no siempre se cuenta con una metodología adecuada a pesar de su existencia, es por este motivo que se aplicará la metodología SCRUM.

SCRUM es una metodología ágil que se usa para minimizar los riesgos durante la realización de un proyecto, pero de manera colaborativa.

Con entregas mensuales o quincenales de resultados, los requisitos más prioritarios en ese momento, se completan lo cual proporciona las siguientes ventajas:

- Gestión regular de las expectativas del cliente y basada en resultados tangibles.
- Resultados anticipados (time to market).
- Flexibilidad y adaptación respecto a las necesidades del cliente, cambios en el mercado, etc.
- Gestión sistemática del Retorno de Inversión (ROI).
- Mitigación sistemática de los riesgos del proyecto.
- Productividad y calidad.

- Alineamiento entre el cliente y el equipo de desarrollo.
- Equipo motivado**.** (8)

JUSTIFICACIÓN APLICATIVA

Se ha detectado la necesidad de realizar una aplicación web, en apoyo a la experimentación en ingeniería de software, se va automatizar la aplicación de pruebas de software, disminuir el tiempo debido a la herramienta de soporte.

El principal beneficio es disminuir el tiempo e incrementar el uso de una herramienta que apoyará a los docentes e investigadores que lleven a cabo experimentos en el área de las pruebas de software.

El sistema web tendrá varios roles, definidos en sesiones donde el docente podrá ingresar y parametrizar las técnicas de las pruebas de software a evaluar. Una vez definidas las variables del experimento, los estudiantes podrán ingresar al sistema y deberán aplicar alguna de las técnicas de pruebas de software, las métricas recolectadas se guardarán en una base de datos para posteriormente ser analizadas por el sistema web.

OBJETIVOS

OBJETIVO GENERAL

Implementar un sistema web en apoyo a la experimentación en ingeniería de software para la realización de experimentos en el área de pruebas de software.

OBJETIVOS ESPECÍFICOS

1. Investigar sobre la experimentación en ingeniería de software para el desarrollo del sistema.
2. Determinar las variables y métricas de las técnicas de pruebas de software que se recolectarán a través de los formularios automatizados.
3. Investigar las tres técnicas de pruebas de software: lectura de código por abstracciones sucesivas, caja negra y caja blanca que se incorporará en el sistema web.
4. Evaluar la usabilidad del Sistema web.

CAPÍTULO I

MARCO TEÓRICO DE REFERENCIA

La experimentación en ingeniería de Software

La organización del conocimiento ha dejado tratar niveles más altos de abstracción del problema y el espacio de solución, dicho conocimiento está basado en la retroalimentación y el aprendizaje aplicando experimentos y analizando los resultados. La experimentación ha sido utilizada en muchos campos, p. ej., física, medicina, fabricación.

Un ejemplo muy claro es dentro del área de la medicina, donde los objetivos del investigador es comprender el funcionamiento del cuerpo humano, mediante la experimentación se conoce los efectos de los fármacos y así se proporciona el conocimiento necesario. Sin embargo los primeros investigadores aplicaron varios procesos a hierbas y entregaron el conocimiento a menudo en secreto que paso de generación a generación. La medicina como campo no progreso hasta que "la aplicación de medicaciones y procedimientos formó una base para evolucionar el conocimiento de la relación entre los efectos y la solución." (9)

La experimentación permite el control en casos de estudio, porque se construyen los modelos con capacidades predictivas y así entender la relación entre el proceso y el producto. (1)

La ingeniería de software no difiere mucho de otras ciencias por lo que también se puede aplicar dentro de un laboratorio, esto implica un componente experimental para comprobar o rectificar teorías, al explorar nuevos ámbitos. La función del investigador es entender la naturaleza de los procesos y productos, y la relación entre ellos. El experimento es basado en técnicas para proporcionar una visión del trabajo real y responde al cómo, cuándo y los límites.

Actualmente en la ingeniería de software es imprescindible conocer para el desarrollo de nuevas soluciones tecnológicas, el cómo hacer, cuánto tiempo tardará y cuánto va a costar el sistema, esto es un gran problema cuando no contamos con investigaciones

previas, es por ello que la experimentación en ingeniería de software nos permite conocer de forma más rigurosa el proceso de desarrollo software. Un ejemplo podría ser el estudio de mejoras a los métodos que son utilizados en el desarrollo de software o la prueba de alguna herramienta nueva.

Los estudios empíricos en ingeniería de software permiten ampliar o mejorar procesos, métodos y herramientas para el desarrollo y mantenimiento de software (1).

El método clásico para identificar las relaciones causa-efecto, es conducir experimentos controlados donde sólo unas pocas variables cambian. "Experimentos controlados en ingeniería de software involucran a menudo a los estudiantes que resuelven tareas de papel en un entorno de aula" (2). Los experimentos en ingeniería de software serían más reales si se ejecutarán tareas reales, con sistemas reales y con profesionales que utilicen herramientas de apoyo, en dicho ambiente.

Pasos para la realización de un experimento

La realización de experimentos realistas requiere una buena gestión de las ocupaciones, un procedimiento experimental típico es el siguiente.

1: Definir el experimento

- Diseñar un nuevo experimento con los requisitos.
- Cuestionarios para recopilar información de antecedentes (nombre, afiliación, Dirección, dirección de correo electrónico, cuenta bancaria si los sujetos son pagados Individualmente, educación, experiencia laboral, etc.),
- Entorno de PC y herramientas.
- Descripciones de tareas y archivos a descargar, etc.

2: Definir, reunir y asignar temas

Definir el tipo y el número de temas que deben participar en el experimento, y reclutarlos. Normalmente, un experimento controlado consiste en dos o más tratos. El tratamiento apropiado debe asignarse a los grupos.

3: Cada sujeto ejecuta el experimento

Distribuir los cuestionarios y otros documentos pertinentes definidos en el paso 1 a los sujetos. En muchos experimentos, es necesario ingresar la fecha en la que el sujeto comienza a leer una descripción de la tarea y cuando la solución de tarea está terminada.

4: Monitorear el experimento

Para asegurar que los sujetos funcionen correctamente y que los datos son apropiados, el investigador debe monitorear el progreso de cada tema.

5: Recoger resultados: Cuando un sujeto ha terminado las tareas, sus resultados son recogidos y almacenados en un lugar seguro. Cuando todos los temas hayan terminado, el investigador puede iniciar el análisis (10).

Herramientas de apoyo a la experimentación en ingeniería de software.

La experimentación en ingeniería de software, ha permitido desarrollar técnicas y métodos para desarrollar software de calidad, pero es necesario el uso de herramientas, que permiten replicar varias veces los experimentos y obtener conclusiones que ayuden a los investigadores. Actualmente ya existen algunas herramientas de apoyo a la experimentación en ingeniería de software entre las que se destacan: Singrid, SESE y Marmoset.

SINGRID

Es posible llevar a cabo un gran número de experimentos consecutivos y obtener resultados teóricos o analíticos para comparar el rendimiento de algoritmos dirigidos a sistemas.

El proyecto SimGrid se inició en 1999 y permite el estudio de algoritmos de programación para plataformas heterogéneas. SimGrid v1, facilitó el prototipo de programación heurística y probarlos en una variedad de aplicaciones (Expresados como gráficos de tareas) y plataformas. En 2003, SimGrid v2, amplió las capacidades de Su

predecesor en dos maneras principales. En primer lugar, el realismo del motor de simulación se mejoró mediante la transición desde un modelo de agujero de gusano a un modelo analítico. En segundo lugar, una se agregó la API para estudiar la programación no centralizada y otro tipo de procesos secuenciales concurrentes (CSPs) (3).

SESE

SESE, se construyó sobre un producto comercial estándar de KompetanseWeb, que es utilizado por varias grandes organizaciones noruegas. SESE fue y sigue siendo desarrollado a través de un estrecho contacto entre Simula Research Laboratory (SRL) y KompetanseWeb. El desarrollo de la funcionalidad adicional requerida en SESE.

SESE es una herramienta para ejecutar experimentos controlados realistas, para evaluar empíricamente las tecnologías de ingeniería de software. El realismo puede, por ejemplo, ser incrementado utilizando profesionales además de estudiantes, herramientas reales de desarrollo en lugar de papel y lápiz, tareas más grandes y un entorno de trabajo típico en lugar de un aula. La logística de ejecutar experimentos realistas es mucho más compleja que para los experimentos sencillos de estudiantes que emplean lápiz y papel (4).

Para el experimento se incluyeron 130 profesionales en desarrollo de Java, nueve de asesoría diferentes compañías y 60 alumnos por tema. El experimento tuvo lugar durante un periodo de dos meses y estuvo organizado en 12 sesiones separadas de un día, con diferentes participantes. Durante el-sesión de día, cada tema tuvo que solucionar seis tareas de programación en su ordenador que utiliza la herramienta de desarrollo Java. Mientras este experimento era de escala más grande y en muchas maneras más realistas que la mayoría de experimentos de ingeniería del software, aparecieron nuevos retos:

- El material experimental como cuestionarios, descripciones de tarea, código y herramientas, tuvo que ser distribuido a cada tema de modo oportuno.
- El diseño experimental es decir todo material no podía ser distribuido inmediatamente, porque cada profesional desarrollador se sentaba en su ubicación

de oficina habitual mientras participaban del experimento, y controlar el progreso individual era crucial.

- Los resultados como respuestas a cuestionarios y soluciones de programa, tuvieron que ser almacenado para análisis futuros.

El estudio implicada en correr los experimentos y motivó la necesidad para una herramienta que podría automatizar algunos de aquellos procesos, por lo tanto, se desarrolló la web basándose en el "Simulador de Entorno para Soporte de un Experimento" (SESE) con la colaboración de una compañía de desarrollo externa.

SESE permitió definir experimentos, incluyendo todo el detalle de cuestionarios, descripciones de tareas y código necesario, asignar temas a una sesión de experimento dada, corrido y controlar cada sesión de experimento y recoger los resultados de cada tema para análisis.

Uno de los objetivos que se buscaba con SESE es hacer experimentos de ingeniería del software reales y así posiblemente generalizar los resultados a una práctica industrial a escala mundial.

Marmoset

Es una herramienta que permite a los docentes de informática involucrar a los estudiantes novatos al desarrollo de software. Los estudiantes envían versiones de sus proyectos a un servidor central, que automáticamente los prueba y registra los resultados.

Marmoset recolecta copias de código mientras los estudiantes trabajan en los proyectos y presenta un esquema de datos que permite una gran variedad de consultas (5).

Pruebas de software

Las pruebas de software están diseñadas para realizar la detección de defectos de forma estructurada y facilitar el estudio de software. En las pruebas de software no se quita ninguna información después de que los sujetos la reciben (6). Existen varias técnicas

de pruebas de software entre las que se destacan: la lectura de código por abstracciones sucesivas, caja negra y caja blanca.

Lectura del código por abstracciones sucesivas.

En la lectura de código, no se utiliza el computador. Se inicia construyendo casos de prueba, los sujetos ven el código fuente impreso, pero no ven las especificaciones. “Los sujetos leen el código fuente y escriben sus propias especificaciones del código, esto es basado en la técnica de lectura por abstracciones sucesivas” (15).

Los sujetos identifican subprogramas principales (líneas consecutivas de código), escriben una especificación para el sub-programa lo más apropiado, agrupan los subprogramas y sus especificaciones, este proceso se repite hasta que hayan abstraído todo el código fuente.

Después de escribir sus propias especificaciones, los sujetos permiten que sus especificaciones sean copiadas, reciben la especificación real a cambio y comienzan con la detección de fallas.

En la detección de fallas, los sujetos comparan la especificación real con la suya para observar las inconsistencias, entre el comportamiento especificado y el esperado del programa. Los sujetos aíslan los fallos de inconsistencia, para ello no se especifica ninguna técnica especial en el aislamiento de fallos. Finalmente, los sujetos hacen una lista de inconsistencias identificadas y fallas aisladas (11).

Pruebas funcionales o caja negra.

Primeramente los sujetos reciben la especificación del sistema, pero no ven el código fuente. Identifican clases de equivalencia en los datos de entrada y construyen Casos de prueba utilizando las clases de equivalencia, prestando especial atención a los valores límite. Después, los sujetos ejecutan sus casos de prueba en el equipo. Se les instruye para no generar pruebas adicionales durante el paso 2, pero no se puede prevenir ni medir esto hasta que se haya concluido cuando los sujetos imprimen sus resultados y

cierran la sesión del computador. “Para la detección de fallas, los sujetos usan la especificación real para observar los fracasos que se revelaron en su ejecución” (15).

Después de registrar las fallas, los sujetos entregan una copia de su salida impresa y reciben el código fuente impreso para intercambiar y comenzar a aislar los defectos. Los sujetos usan el Código fuente para aislar las fallas que causaron los defectos observados. No se especifica ninguna técnica especial para la actividad de aislamiento de fallas. Finalmente, los sujetos entregan una lista de fallas observadas y fallas aisladas. (12)

Pruebas estructurales o caja blanca.

Al iniciar la ejecución de la prueba, los sujetos usan una versión establecida del programa para ejecutar sus casos de prueba y ver los informes de valores de cobertura alcanzados.

Los sujetos desarrollan pruebas adicionales de casos hasta que alcanzan el 100% de cobertura, o creen que no pueden lograr una mejor cobertura. Después de ejecutar los casos de prueba, los sujetos desconectan la computadora, realizan a mano una copia de su salida, entonces reciben una copia de la especificación a cambio y comienza la detección de fallas. Los sujetos utilizan la especificación para observar fallas en su salida (13).

Los sujetos aíslan las fallas que causaron las fallas observadas, no se especifica ninguna técnica para el aislamiento de fallas. Por último, los sujetos escriben una lista de las Fallas observadas y fallas aisladas (15).

Evaluación empírica de tres técnicas de pruebas de software

Las técnicas de lectura de código por abstracciones sucesivas, pruebas funcionales utilizando clases de equivalencia con valores límite, y pruebas estructurales con el objetivo de 100% cobertura de casos de prueba, difieren en su efectividad que es el porcentaje de fallas observadas y fallas aisladas mientras que la eficiencia es el número de fallas o división de fallas por tiempo requerido para detectarlas.

En la investigación ya realizada por Erik Kamsties y Christopher M. Lott se determinó que las técnicas difieren en la efectividad, el aislamiento de fallas de diferentes tipos, y que la motivación al sujeto le permite desarrollar habilidades para detectar la eficacia y eficiencia.

Un primer experimento controlado fue realizado a principios de los ochenta para evaluar la efectividad y eficiencia de 50 estudiantes que utilizaron las tres técnicas de detección de defectos para observar fallas y aislar las fallas en programas pequeños.

Las tres técnicas fueron la lectura de código por abstracciones sucesivas, pruebas funcionales o caja negra y pruebas estructurales o caja blanca. Dos repeticiones internas mostraron que los sujetos relativamente inexpertos eran igualmente de efectivos en la observación de fallas y aislamiento fallas con las tres técnicas.

En la investigación realizada por Erik Kamsties y Christopher M. Lott, los sujetos fueron más eficientes en ambas tareas cuando utilizaron las pruebas funcionales y existen algunas diferencias significativas entre las técnicas en cuanto a eficacia, se observaron fallas aislantes de diferentes tipos.

Estos resultados sugieren que los sujetos inexpertos al aplicar la técnica de lectura de código lo realizan eficazmente como una validación basada en la ejecución de la técnica, pero son más eficientes cuando utilizan pruebas funcionales.

Los desarrolladores de software tienen muchas armas en su arsenal de técnicas para detectar defectos en su Programas. Pueden ejecutar un programa en una computadora para revelar y observar fallas, con sus ojos agudos para aislar fallos leyendo el código fuente.

En la ingeniería de software, “los desarrolladores de software necesitan evidencia empírica para ayudarles Decidir qué técnica de detección de defectos para aplicar en diversas condiciones” (14).

Para contribuir a esta base de evidencia, han replicado y ampliado un experimento que evalúa tres técnicas de detección de defectos, (15), específicamente lectura de código por abstracciones sucesivas (11), pruebas funcionales o caja negra (12), y pruebas estructurales o caja blanca (13). Por supuesto no hay ninguna técnica infalible.

El desarrollador puede detectar la mayoría de los defectos es decir, máxima efectividad y condiciones en las que la técnica ayuda al desarrollador a detectar los defectos más rápidamente es decir, la eficiencia. Tales condiciones pueden incluirse "cuando los desarrolladores tienen poca experiencia con el estándar de Lenguaje de programación o cuando la mayoría de las fallas son causadas por mal manejo de casos de prueba.

Un fallo es el comportamiento de un programa que se desvía visiblemente de la especificación. Una falla es cualquier problema en el programa. Código fuente que puede manifestarse en un error durante la ejecución. Usamos el defecto como un término general que abarca tanto Fallas y fallas. Véase también (16).

El experimento consistió en agregar fallas después de revelar y observar fracasos. En segundo lugar se entregó materiales necesarios. El experimento ofrece una oportunidad para los desarrolladores de software actuales de evaluar su desempeño cuantitativamente, mediante la experiencia educativa.

Según Watts Humphrey (17). Un experimento se convierta en un ejercicio estándar que los desarrolladores utilizan para evaluar y afinar sus habilidades de detección de defectos.

Para medir las diferencias individuales y hacer el experimento interesante para los sujetos, cada Sujeto aplica cada técnica de detección de defectos una vez. Porque los efectos de aprendizaje hacen ridículo Una persona para buscar fallos y fallas en el mismo programa más de una vez, se necesita al menos tres diferentes programas defectuosos, una segunda variable independiente. Para permitir la medición de cualquier maduración Efectos entre los sujetos, manipulamos el orden de aplicación de las técnicas, un tercero independiente variable.

Debido a que tres ejercicios son demasiado para un solo día, el experimento debe Varios días. Sin embargo, esto no se considera una cuarta variable independiente porque todos los sujetos ven El mismo programa el mismo día; Es decir, las dos variables "programa" y "día" se confunden. Los experimentos de B & S y K & L son idénticos con respecto al diseño básico, Las técnicas de detección y la hipótesis relativa a la revelación y la observación de los fracasos.

Diferencias Incluyen el paso adicional de aislamiento de fallas y hipótesis asociadas, el lenguaje de programación (C versus FORTRAN), los programas y fallas (instrumentos),

y metas de cobertura más específicas para Pruebas estructurales (rama de 100%, condición múltiple, bucle y cobertura de operador relacional versus Cobertura del estado del 100%).

No intentar analizar la influencia que el tipo de software tuvo sobre Las variables dependientes, ni analizamos el costo de los recursos informáticos. La dramática caída de Los costos de hardware a lo largo de los 12 años entre los dos experimentos hace que la pregunta relativamente poco interesante. Finalmente, el experimento B & S no informa sobre la motivación del sujeto.

CAPITULO II

MARCO METODOLÓGICO

MÉTODOS Y TÉCNICAS

La investigación a realizarse será del tipo aplicativa, porque se busca la generación de conocimiento a partir de la aplicación directa al problema, no existe una herramienta web que apoye la realización de experimentos en pruebas de software.

Se tomará fundamentalmente hallazgos tecnológicos previos, se realizará el enlace entre la teoría y el sistema web. Es por ello que el presente anteproyecto se convertirá en una fuente de información secundaria porque, no se está generando un conocimiento nuevo, pero se toma la información existente y se aplicará en el desarrollo del sistema.

El sistema web no se aplicará en una empresa, sino en un contexto académico porque será una herramienta que apoye a la experimentación de software, se utilizará la metodología ágil SCRUM.

Para comprobar la usabilidad del sistema web se aplicará la estadística inferencial para tomar una pequeña población, se realizarán encuestas a los docentes de la FIE de la ESPOCH.

RECURSOS NECESARIOS

Los recursos son materiales necesarios para producir la herramienta web, es por ello que dentro de un proyecto es necesario identificar los recursos necesarios, para la realización del mismo.

HARDWARE

Para el desarrollo del sistema se utilizará una laptop con las siguientes características.

Nombre	Descripcion
Procesador	Intel(R) Core TM i5

Memoria RAM 8,00 GB

Tipo de Sistema Sistema operativo de 64 bits, procesador de x64

SOFTWARE

Las tecnologías que se utilizarán para realizar el sistema, son las siguientes:

PHP: Es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico.

MySQL: Es un gestor de bases de datos de una aplicación, además es una herramienta cliente/servidor.

AJAX (Asynchronous JavaScript And XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA.

Variable

Tabla1-2: Descripción de la Variable

Variable	**Tipo**	**Concepto**
Usabilidad	**Compleja**	Un conjunto de atributos relacionados con el esfuerzo necesario para su uso, y en la valoración individual de tal uso, por un establecido o implicado conjunto de usuarios.

Fuente: Estándar ISO 9126

Operacionalización de la Variable

Tabla2-2: Operacionalizacion de la variable

Variable	**Categoría**	**Indicadores**	**Técnicas**	**Fuente de investigación**
Usabilidad	**Categoría 1**	• Aprendizaje- Atributos del software que se relacionan al esfuerzo de los usuarios para reconocer el concepto lógico y sus aplicaciones.	**Encuesta**	**Docentes de la Escuela de Ingeniería en sistema de la**

		• Comprensión - Atributos del software que se relacionan al esfuerzo de los usuarios para reconocer el concepto lógico y sus aplicaciones. • Operatividad - Atributos del software que se relacionan con el esfuerzo de los usuario para la operación y control del software. • Atractividad		**ESPOCH.**

Fuente: Hernández G.

Bibliografía

1. *The Role of Experimentation in Software Engineering: Past, Current, and Future.* **Brasili, Victor.** Berlin, Alemania : Proceedings of the 18th International Conference on Software Engineering, Marzo 1996. ISBN 0-7695-1796-X.

2. **Potts, C.** *Software-Engineering Research Revisited.* s.l. : IEEE Software, 1993. . Vol. 10, No. 5, pp. 19-28, .

3. *Singrid: A generic framework for large-scale distributed experiments.* **Casanova, Henri, Legrand, Arnaud and Quinson, Martín.** s.l. : 10 th IEEE International Conference on Computer Modeling an Simulation- EUROSIM/UKSIM, 2008. inria-00260697.

4. *SESE – An Experiment Support Environment for Evaluating Software Engineering Technologies.* **Arisholm, Erick, et al., et al.** s.l. : KompetanseWeb AS, Tullinsgate 6, NO-0166 Oslo, Norway yngve@kompetanseweb.no.

5. *Software repository mining with Marmoset: an automated programming project snapshot and testing system.* **J, Spacco, et al., et al.** St. Louis, Missouri : Proceedings of the 2005 International Workshop on Mining Software Repositories, 2005. 1-5.

6. *Experimental models for validating computer technology.* **Zelkowitz, M. V. and Wallace, D.** s.l. : IEEE Computer 31, 5, May, 1998. 23-31.

7. *An Environment of Conducting Families of Software Engineering Experiments.* **Hochstein, Lorin, et al., et al.** University of Nebraska – Lincoln : Technical Report CS-TR-4877, May 2007. UMIACS-TR-2007-28.

8. **Albaladejo, Xavier.** Proyectos agiles.org . [Online] [Cited: 01 2017, 30.] https://proyectosagiles.org/que-es-scrum/.

9. **Reggia, J.** Computer-assisted medical decision making in Application of Computers. [book auth.] M. Schwartz. *Application of Computers.* New York : 198-213, 1982.

10. *The Discipline of Systematic Development of Web Applications.* . **Kappel, G, et al., et al.** s.l. : John Wiley & Sons. , 2006.

11. *Structured Programming: Theory and Practice.* **Richard, Linger, Harlan, Mills and Bernard, Witt.** s.l. : Addison-Wesley Publishing Company, 1979.

12. *The Art of Software Testing.* **Glenford, Myers.** Nueva York : John Wiley & Sons, 1979.

13. *Functional program tests.* **William, Howden.** s.l. : IEEE Transactions on Ingeniering Software, marzo de 1980. SE-6: 162-169.

14. *Experimentation in software engineering.* **Victor R. Basili, Richard W. Selby, y David H. Hutchens.** s.l. : IEEE Transactions on Software Engineering, julio 1986. SE-12 (7): 733-743.

15. *Comparing the efficiency of technical software testting.* **Victor, R Basili and Selby, Richard.** s.l. : IEEE Transactions on Software Engineering , diciembre, 1987. 13 (12): 1278-1296.

16. *Estándar Glosario de software de Ingeteam niería Terminología de 1983.* **IEEE.**

17. *A discipline of Software Engineering.* **Watts, Humphrey.** s.l. : Addison-Wesley, 1995.

18. *An Empirical Evaluation of Three Defect-Detection Techniques.* **Erik, Kamsties and Christopher M., Lott.** Kaiserslautern, Germany : Software Technology Transfer Initiative, May 30, 1995.

19. **Del Valle Rodriguez, Ana Nieves.** Metodologías de Diseño usadas en Ingeniería Web. *Su vinculación con la NTics. (Postgrado). Universidad Nacional de la Plata.* . 2009.

20. **Loucopoulos, Pericles and Karakostas, Vassilios.** *System Requirements Engineering.* Inc. New York : McGraw-Hill, 1995. 160.

21. **Oliveros, Alejandro, et al., et al.** *Requerimientos para aplicaciones web.* s.l. : XIII Workshop de Investigadores en Ciencias de la Computación, 2011.

22. *Desarrollo dirigido por modelos de aplicaciones web que integran.* **Quintero, R.** Sevilla-Valencia : s.n., 2008.

23. *On testing non-testable programs.* **Elaine J., Weyuker.** s.l. : Computer Journal 25(4), Noviembre 1982. 465–470.

GLOSARIO

Aplicación web a aquellas herramientas que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una intranet mediante un navegador.

Clase-Responsabilidad-Colaborador (CRC): identificación de verbos y sustantivos, entre otras técnicas, que permiten determinar las clases, métodos y atributos.

Contexto: Entidad que posee un conjunto de circunstancias (materiales o abstractas) que se producen alrededor de un hecho, o evento dado.

Controlador: Según el patrón MVC el controlador maneja la funcionalidad involucrada desde que el usuario entra a la web hasta que este sale, cada clic que ejecuta la conexión pasa por este componente, y según lo que suceda ejecutará la página JSP para mostrarle una salida al usuario.

Enterprise Architect: Herramienta de análisis y diseño con UML, que posee un modelado avanzado para negocios y sistemas de software.

Flujos de trabajo (*Workflow*): Estudio de los aspectos operacionales de una actividad de trabajo: cómo se estructuran las tareas, cómo se realizan, cuál es su orden correlativo, cómo se sincronizan, cómo fluye la información que soporta las tareas y cómo se le hace seguimiento al cumplimiento de las tareas.

Interfaz de usuario es el medio con que el usuario puede comunicarse con una máquina, un equipo o una computadora.

Modelo de datos es un lenguaje orientado a hablar una Base de Datos.

Metodología hace referencia al conjunto de procedimientos racionales utilizados para alcanzar una gama de objetivos.

Modelo del producto: Es una abstracción de las propiedades comunes que se encuentran en cualquier producto desarrollado con el uso de un método, permite capturar los conceptos y patrones arquitecturales genéricos de cualquier aplicación web.

Navegación: Facilidad con la que un usuario puede desplazarse por todas las páginas que componen una aplicación web.

Programación es el proceso de diseñar, codificar, depurar y mantener el código fuente de programas computacionales.

Puntos de control: Son mecanismos que verifican que se cumplen las condiciones o requerimientos necesarios para permitir el paso de una fase a otra.

Productividad: Definida como la relación entre los resultados y el tiempo utilizado para obtenerlos: cuanto menor sea el tiempo que lleve obtener el resultado deseado, más productivo es el sistema.

Primitivas de Abstracción: Se define como primitiva a los principales elementos que permiten el diseño de una aplicación web.

Requerimientos: es una necesidad documentada sobre el contenido, forma o funcionalidad de un sistema o un software.