Universidad de las Ciencias Informáticas

Facultad 6

Estrategia de prueba para la Fábrica de Software del Departamento Geoinformática

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

AUTOR(ES): Elizabetha del Rosario Alarcón Bermudez

TUTOR(ES): Ing. Yesleny Becerra Torreira

La Habana, 27 de junio del 2011

“Año 53 de la Revolución”

Frase

"La calidad consiste en no tener deficiencias".

Joseph Moses Juran

DEDICATORIA

A la memoria de mi bisabuela,

Quien siempre estuvo ahí para mí, aunque ahora no esté físicamente.

AGRADECIMIENTOS

Dado el momento de agradecer, comenzaré diciendo que esta es una de las tareas más difícil, puesto que son muchas las personas que han participado para ver el sueño de Ingeniera finalmente realizado. Voy a tratar de hacerlo lo mejor posible para que no se me quede ninguno.

Ante que todo quiero agradecer a mis dos madres postizas de 5 años esas son mi tía Mary y mi prima Nurys, sin ustedes no sé que hubiese sido de mí, gracias por apoyarme en cada momento de mi vida y sobre todo cuando más lo necesitaba, siempre estaré eternamente en deuda con ustedes, pues no existen palabras para demostrar mi agradecimiento, las quiero mucho.

A mi mamá por darme la vida y tu apoyo incondicional en cada decisión que tomé, te quiero mucho.

A mis abuelos por malcriarme tanto, en especial a mi bisabuela que no se encuentra en este momento, a ella le debo mi crecimiento como persona.

Agradezco a mis hermanos de sangre Lorena, Anniel y Dianela por sacarme cada día una sonrisa, y hacerme esforzar para demostrarles que sí se puede, lo mismo para mis primitos Amanda y Anthony.

A mi tío Jabao y mi primo Georvi por ayudarme con sus críticas, me han servido de mucho, aquí está el fruto ingeniera, Tony por servir de transporte, a todos gracias,

A mi hermana Greta por ser hermana, amiga y aconsejarme en las momentos fáciles y difíciles, gracias por abrirme los ojos siempre que estuve ciega.

A mi mamá de 5 años Lele, por compartir cada derrota, cada victoria, te voy a extrañar mucho, te quiero y gracias por existir.

A mis hermanitas de aquí de la UCI Dino, Naivis, Sachita, Katy, Yara, Danieyis y Melo gracias por apoyarme y entenderme, las quiero mucho.

A mis hermanitos Osi, Juan y Joako por cada consejo dado, a todos los quiero y a ti Osi te voy a extrañar mucho.

A todas las amistades que colaboraron y fueron parte de vida tanto en la UCI como fuera de la misma, a ustedes también les debo, Yami, Baby, Los Alejandros, Jeans, Yansel, Lucho, Ari, Lisy, a todos muchas gracias por existir.

A mis compañeras de apto 107201 y 107202, principalmente mis clientes de la peluquería Nixys, Diana, Grechin, Yanet Ramos, Mailén, la Claria, Leydis, La Miche y Eliadys gracias por aguantarme y malcriarme.

A mis compañeros de grupo primero 9108, 9207, 9306, 9405 y 6504 a todos muchas gracias.

A mi familia, vecinos y amistades en general, toda aquella persona que aportó un poquito para que fuera ingeniera, a todos ustedes muchas gracias.

A mi tutora por ayudarme tanto y ser como una compañera de tesis para mí, a ti te debo mucho, gracias por ser mi tutora.

A mi papá, aunque esté brava contigo, te quiero mucho.

A la mamá de mis hermanos Yami, gracias por aconsejarme y tenerme presente como una hija más, para ti también muchas gracias, sin ti muchas cosas no hubiesen sido posibles.

A todos los profes que he tenido desde que estaba en preescolar, gracias por sus consejos, sus críticas y su educación.

A Elizabeth por corregirme en cada paso de mi tesis, cada sugerencia me ha servido de mucho esto demuestra que no fue en vano tu esfuerzo, gracias por tu apoyo.

A mi oponente y tribunal de tesis, por cada sugerencia para ser una mejor ingeniera, a ustedes también muchas gracias.

A Fidel y a la Revolución por hacer realidad mi sueño.

DECLARACIÓN DE AUTORÍA

Declaro que soy la autora de este trabajo y autorizo a la facultad 6 de la Universidad de las Ciencias Informáticas, así como al Centro Geoinformática y Señales Digitales (GEySED) para hagan el uso que estimen pertinente con el siguiente trabajo de diploma.

Para que así conste firmamos la presente a los ___ días del mes de ___ del año 2011.

Autor:

Elizabetha del Rosario Alarcón Bermudez.

Tutora:

Ing. Yesleny Becerra Torreira.

DATOS DE CONTACTO

Elizabetha del Rosario Alarcón Bermudez.

Correo: eralarcon@estudiantes.uci.cu

Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba.

Ing. Yesleny Becerra Torreira.

Ingeniera en Ciencias Informáticas.

Correo: ybecerra@uci.cu

Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba.

RESUMEN

En este trabajo de diploma se propone una estrategia de prueba a seguir, para planificar, documentar y evaluar los resultados de las pruebas aplicadas a los productos de la Fábrica de Software del departamento Geoinformática perteneciente a la facultad seis de la Universidad de las Ciencias Informáticas (UCI). Para ello se tomó un producto piloto llamado Sistema de Información Geográfico Rutas (SIGRutas) al cual se le aplicó la estrategia propuesta, con los diferentes tipos de pruebas fundamentales existentes (unidad, integración, sistema, aceptación) para su posible validación.

Se planificaron pruebas, utilizando como soporte herramientas que automatizaron el proceso de desarrollo, además se obtuvieron gran número de artefactos como: plan de pruebas, diseños de casos de pruebas, evaluaciones de pruebas, registros de pruebas y un listado con las no conformidades detectadas.

Palabras claves: calidad, pruebas de software, Fábrica de Software, estrategia de pruebas.

ABSTRACT

In this dissertation proposes a test strategy to continue to plan, document and evaluate the results of the tests applied to products of the Software Factory Geoinformatics department within the Faculty six of the Computer Science University (CSU). For this pilot took a product called Pathways Geographic Information System (SIGPathways) and applying the strategy, with different types of existing core tests (unit, integration, system, and acceptance) for possible validation.
Test were planned using automated tools to support the development process, also obtained a large number of artifacts including: test plan, test case design, test evaluation, test records and a list of nonconformities.
Keywords: quality, software testing, Software Factory, test strategy.

ÍNDICE

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

INTRODUCCIÓN

En la actualidad, a nivel mundial la producción del software crece y se desarrolla a ritmo agitado, aplicando el uso de técnicas o metodologías que permiten una mayor productividad. Todas las empresas de software ambicionan producir aplicaciones informáticas con alta calidad, en el menor tiempo posible y a costos mínimos, aumentando así la competencia entre ellas debido a que los clientes son cada vez más exigentes.

La calidad del software juega un papel importante dentro del desarrollo de soluciones informáticas, influyendo positivamente en la decisión de un cliente a la hora de seleccionar el producto que necesita. Además de cumplir con las exigencias explícitas e implícitas del consumidor debe ser confiable y aceptable por el mismo. Dicha calidad pretende dar confianza en que el software reúne las características necesarias, cumpliendo así todos los requerimientos del cliente y es aplicable a nivel de proyecto a todo el proceso de desarrollo de software.

Uno de los procesos que permite validar y verificar (V&V) los requerimientos del producto lo constituye las pruebas de software, actividad que se efectúa para dar garantía de que el software sea factible. Las pruebas que se le realizan a un producto tienen como objetivo encontrar errores en las funcionalidades del sistema, comprobar la lógica interna de los componentes, verificar el comportamiento y rendimiento de los mismos.

En un proyecto las pruebas, en ocasiones requieren un mayor esfuerzo que cualquier otra actividad de la ingeniería de software. Si se realizan sin un procedimiento, el tiempo se desperdicia y el esfuerzo es consumido inútilmente y en el peor de los casos los errores inadvertidos resultarán no detectados. Por tanto sería conveniente trazar una estrategia consecuente para comprobar el software.

La estrategia de prueba provee un mapa que detalla los pasos que se llevan a cabo como parte de la prueba, cuándo se deben planificar y efectuar esos pasos, cuánto esfuerzo, tiempo y recursos se van a necesitar. Debe facilitar una guía profesional y un conjunto de hitos para el jefe de proyecto, debido a que se debe poder medir el avance

y las dificultades deben emerger lo antes posible.

Hace ya varios años Cuba se introdujo en uno de los procesos más importantes de su historia, la informatización de la sociedad en todos los sectores dígase: Salud, Educación, Cultura, Deporte, Turismo y Prensa; siendo su pilar fundamental la UCI. Universidad surgida al calor de la Batalla de Ideas, donde existe una estrecha vinculación entre la formación y la producción de software, garantizando la práctica en proyectos reales.

La UCI cuenta con diez facultades incluyendo las facultades regionales, estas últimas se encuentran ubicadas en la provincia Granma, Ciego de Ávila y Artemisa. En una de ellas, la facultad seis, que pertenece a la sede central, existe el Centro de Desarrollo Geoinformática y Señales Digitales (GEySED), el cual tiene como misión fundamental desarrollar productos, servicios y soluciones informáticas en el campo del procesamiento de Señales Digitales y la Geoinformática, contribuyendo a la formación integral de profesionales que respondan a las necesidades del progreso científico técnico y socioeconómico, permitiendo un posicionamiento en el mercado nacional y a escala mundial.

GEySED está constituido por dos departamentos: Señales Digitales y Geoinformática, en este último se implementó una Fábrica de Software, debido a que trae como ventajas disminución de riesgos laborales, profesionales altamente calificados en el ámbito tecnológico, ya que es necesario especializar al profesional en una tarea específica del proceso, concentrando sus esfuerzos en dicha tarea, además presenta otras ventajas como la disminución de costos por inversión, siendo por esta razón las funciones principales de la Factoría de Software Aplicativos de Sistema de Información Geográfico (SIG) [1] la reutilización de código, la utilización de componentes existentes y poder integrar estos a proyectos que los puedan necesitar.

[1] Aplicativos SIG utiliza como base de apoyo la Plataforma Soberana para el desarrollo de Sistemas de

Sus objetivos principales son permitir la representación geoespacial de la información asociada a cualquier negocio que lo requiera. Proporcionar servicios de acceso a la información geográfica, para su consulta, análisis y visualización, mediante una interfaz de usuario sencilla y de fácil manejo que pueda ser utilizada por usuarios no especializados en tecnología SIG. Integrar la información socioeconómica existente (Recursos Humano, Activos Fijos, Entidades de Servicios, Lugares de interés, etc.) con la información geográfica asociada.

Las factorías de software, tienen como uno de sus objetivos, confeccionar el proceso de desarrollo de software, a través de tácticas, pautas y procedimientos determinados que cuenten con las herramientas precisas para su implantación. Una fábrica de software fundamentalmente comprende reutilización consecuente de recursos como son los requerimientos, diseño arquitectónico, software y la experiencia de los desarrolladores, entre otros.

Sin embargo en la Factoría Aplicativos SIG del departamento Geoinformática, existen deficiencias en la definición del proceso de prueba y en la planificación de las mismas, lo que trae como consecuencia encontrar errores, al culminar el desarrollo, que pudieron ser erradicados en iteraciones tempranas. Esta forma de trabajo hace que el proceso de liberación del producto se torne complejo, pues al descubrir en esta fase gran número de deficiencias, es necesario ejecutar mayor número de iteraciones, por consiguiente, no se cumple con las fechas de entrega previstas y ocasiona retrasos en la liberación del mismo, lo que promueve desconfianza por parte de los clientes.

Otra de las dificultades que se aprecia en la Factoría es que pesar que usan la metodología Proceso Unificado de Desarrollo (RUP) , no se concretan las actividades, el flujo de procesos, ni los roles, afectándose la eficiencia, la calidad y el ciclo de vida del producto. Lo anteriormente dicho trae consigo que no exista documentación completa de los productos, que el trabajo en equipo no esté debidamente organizado y que no se conozcan cuales son exactamente las necesidades finales del cliente.

Por las razones anteriormente expuestas se identificó como **problema a resolver**: ¿Cómo elevar la calidad del proceso de desarrollo en la Fábrica de Software del departamento Geoinformática, permitiendo la verificación y validación de los requisitos del cliente así como la correcta utilización de los recursos de la fábrica?

Por tanto el **objeto de estudio** estará basado en el Proceso de prueba para una Fábrica de Software: y que tendrá como **objetivo general**: Desarrollar una estrategia de pruebas para la Fábrica de Software del departamento Geoinformática que permita la verificación y validación de los requisitos del cliente así como la correcta utilización de los recursos de la fábrica.

De ello se deriva el **campo de acción**: Proceso de prueba para una Fábrica de Software en el departamento Geoinformática.

La **idea a defender** formulada, basada en el problema y en el objetivo propuesto como meta, plantea que la elaboración de una estrategia de pruebas para la Fábrica de

Software del departamento Geoinformática permitirá la verificación y validación de los requisitos del cliente así como la correcta utilización de los recursos de la fábrica.

Para dar cumplimiento al objetivo planteado se debe cumplir con las siguientes **tareas de investigación**:

- Descripción del proceso de pruebas que está definido actualmente en la UCI.
- Caracterización de los modelos de pruebas existentes.
- Comparación de estrategias de prueba aplicadas a las fábricas de software a nivel mundial.
- Elaboración de la estrategia de pruebas para la Fábrica de Software del departamento Geoinformática.
- Validación de la estrategia propuesta.

Como **posibles resultados** se espera:

1. Estrategia de prueba para la Fábrica de Software del departamento Geoinformática.

2. Resultados de la puesta en práctica de la estrategia propuesta.

Métodos de investigación:

Para el desarrollo de la investigación científica se utilizan diferentes Métodos Teóricos y Empíricos que permiten analizar la información referente al tema, esquematizarla y llevarla al caso específico que se estudia con el objetivo final de alcanzar una solución. Para el desarrollo de la fundamentación teórica los Métodos que se utilizan son:

Métodos teóricos:

- Analítico-Sintético: a través de este método se puede llevar a cabo el estudio analizando documentos ya existentes que se relacionen con el proceso de prueba para una fábrica de software.
- Análisis Histórico-Lógico: este método permitirá examinar teóricamente cómo han evolucionado las estrategias y fábricas de software ya existentes para luego adaptarlas a la situación problemática planteada.
- Modelación: este método permitirá crear la propuesta y la estrategia que ofrezca la información sobre un proceso de prueba para la fábrica de software.

Métodos empíricos:

- Entrevista: Este método permite obtener información sobre el proceso de prueba que se está llevando a cabo actualmente en el Centro, cuyo éxito depende fundamentalmente del grado de comunicación que se logre alcanzar en una conversación planificada con el jefe de proyecto de la fábrica y los integrantes del Grupo de Calidad[2].

La investigación está estructurada en tres capítulos:

Capítulo 1: Fundamentación Teórica. Emprende conceptos relacionados con el contenido de Pruebas, Estrategia de pruebas y Factoría de Software. Se mencionan los tipos de prueba más representativos a nivel internacional, las características fundamentales de cada uno. Además se abordan aspectos sobre la definición de Factoría de Software.

Capítulo 2: Propuesta de la solución. Se expondrá la propuesta de solución a la problemática planteada, elaborando una estrategia de pruebas de calidad para llevar a cabo la investigación científica.

Capítulo 3: Aplicación de la propuesta. Recogerá los resultados que proyecte la aplicación de la estrategia propuesta, fundamentándose con los diseños de casos de prueba y una lista de las no conformidades.

[2] El Grupo de Calidad del Centro cuenta con una Asesora de Calidad, 41 estudiantes (de ellos 10 son aprendices de tercer año, 14 de 4to año y 17 de 5to año) y 9 profesionales, donde cada miembro ocupa un rol significativo.
Este Grupo de Calidad se ha trazado como objetivos fundamentales:
1. Mantener un sistema de revisiones internas que garanticen elevar los niveles de calidad del proceso de desarrollo de software del centro.
2. Guiar el aseguramiento de la calidad del proceso de desarrollo de software en los proyectos del centro.
3. Fomentar el desarrollo investigativo y ejecutar actividades de formación y superación de cada una de las temáticas y área de trabajo del grupo de calidad.
4. Ejecutar pruebas de liberación, aceptación y pilotos al 100% de los entregables de todos los proyectos del Centro.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1. Introducción.

En este capítulo se sistematizarán una serie de conceptos significativos relacionados con la obtención de una estrategia de prueba de una Factoría de Software para el departamento Geoinformática. Se describen los tipos de prueba más representativos encontrados durante la consulta bibliográfica, los principales conceptos asociados con la calidad. También se presentarán definiciones de estrategia y características de los modelos de prueba.

1.2. Conceptos asociados.

1.2.1. Calidad.

Con el objetivo de estudiar la medida de la calidad del software se hace imprescindible definir este atributo. Si se refiere a un producto, la calidad es diferenciarse cualitativa y cuantitativamente respecto de algún atributo requerido. Además de satisfacer las necesidades y requisitos solicitados por el cliente. Esto quiere decir que la calidad de un producto depende de la forma en que éste responda a las particularidades del comprador, y se puede decir que es satisfactoria cuando cumple estrictamente con las exigencias del mismo. La calidad es ante todo, satisfacción del cliente.

La gran mayoría de los autores consideran que la calidad debe ser definida desde el punto de vista del consumidor, ratificando este planteamiento Feigenbam (1) plantea que la calidad es “Satisfacción de las expectativas del cliente”, igualmente Jurán (2) opina que “Calidad es adecuación al uso del cliente”. Por otra parte, La Real Academia de la Lengua Española (3) define que la calidad es una “Propiedad o conjunto de propiedades inherentes a una cosa que permiten apreciarla como igual, mejor o peor que las restantes de su especie”. Donde las propiedades son características tales como: color, tamaño, forma, etc.

Edwards Deming, comenta en su libro "Out of the Crisis", publicado en 1982: “El control de calidad no significa alcanzar la perfección. Significa conseguir una eficiente producción con la calidad que espera obtener en el mercado.” [3]

[3] Deming, Edwards, *Out of the Crisis*, Cambridge University Press, 1986.

Joseph Juran. En 1986, escribió La trilogía de Juran, esta trilogía es Planificación de la Calidad, Control de Calidad y Mejora de la Calidad. Se amplía el enfoque de la calidad más allá del producto y la satisfacción del cliente hacia lo que significa calidad en los procesos y el papel de la mejora de procesos en la calidad final de la producción en general: producto, satisfacción del cliente y procesos involucrados.

La Planificación de la Calidad, se definió como el "rendimiento del producto que da como resultado la satisfacción del cliente; libertad de deficiencias en el producto, que evita la falta de satisfacción del cliente". El Control de la Calidad se entiende "como un proceso que debe seguir toda empresa para asegurarse que sus productos o servicios mantengan un nivel mínimo de Calidad, el cual es definido por la propia empresa, de acuerdo a las características de lo que genera, de las características de sus clientes y de los objetivos de eficiencia que se hayan planteado y que deban alcanzar con regularidad".

La Mejora de la Calidad es el proceso de elevarse a niveles de rendimiento sin precedente. Un programa de este tipo incluye demostrar las necesidades de las mejoras, identificar proyectos específicos para la mejora, organizar el apoyo para los proyectos, diagnosticar las causas, dar remedios para las causas, demostrar que los remedios son efectivos bajo las condiciones de operación y proporcionar el control para mantener las mejoras.[4]

De lo comentado por cada uno de estos autores se puede evidenciar que todas las definiciones incluyen la participación del cliente y en el contexto actual la calidad persigue los siguientes objetivos:

- Satisfacción de los consumidores o compradores.
- Eficiencia en la utilización de los recursos humanos.
- Reducción del costo de operaciones.

A continuación se ofrecen otras definiciones de asociaciones y expertos reconocidos en el mundo de la calidad.

La ISO 9000 (4) define Calidad como "Grado en el que un conjunto de características inherentes cumple con los requisitos".

[4] Jurán, Joseph, *Juran´s Quality Handbook*, Ed. McGraw Hill, 1998

“El conjunto de características de una entidad que le confieren su aptitud para satisfacer las necesidades expresadas y las implícitas”. (4)

De este modo se coincide con el criterio de Pressman (6), quien plantea que un producto tiene calidad cuando cumpla con los siguientes elementos:

- Se cumple el propósito para el cual fue previsto.
- Satisfacción total de las expectativas de los clientes y consumidores finales.
- Es producido en el tiempo pronosticado y al presupuesto acorde para el mismo.

Estos criterios sobre la calidad de un producto se toman en cuenta cuando se trata de desarrollo de software que se dedican a ello. En la actualidad en el ámbito de la informática a partir del aumento de los productores de software y la complejidad del proceso, existe un incremento de la competencia. Razón por la cual la calidad comenzó a ser un punto esencial en la creación del producto para garantizar la rentabilidad y en mantenimiento en el mercado.

Un aspecto de interés que es necesario resaltar es la evolución en el proceso de validación de los productos informáticos, pues se ha podido detectar que pese a que inicialmente eran los propios desarrolladores los encargados de este proceso, en los últimos años esta se realiza mediante equipos independientes que se rigen por metodologías, normas y estándares.

1.2.2. Calidad del Software.

La calidad del software es de significativa importancia debido a que proporciona un correcto desarrollo y funcionamiento de un software determinado, este proceso se ha convertido en unos de los problemas más trascendentales de la era informática con los que se han afrontado los fabricantes de los mismos y el cual por ende preocupación para especialistas, ingenieros, investigadores y comercializadores de software, de ahí muchos han encaminado sus más grandes esfuerzos a indagar entorno a:

- ¿Cómo obtener un software con calidad?
- ¿Cómo evaluar la calidad del software?

Como resultado de estas investigaciones numerosos han sido los resultados del término Calidad de Software, los más generalizados de acuerdo al Institute of Engineers Electrician and Electronic, Std. 610 (5) son: “La calidad del software es el grado con el que un sistema, componente o proceso cumple los requerimientos

especificados y las necesidades o expectativas del cliente o usuario". Otro concepto es el proporcionado por el autor Roger S. Pressman (6), una autoridad universalmente reconocida en el avance de procesos de software y en tecnologías de Ingeniería de software, se refiere a la calidad del software como: "Concordancia del software producido con los requerimientos explícitamente establecidos, con los estándares de desarrollo prefijados y con los requerimientos implícitos no establecidos formalmente, que desea el usuario".

Esta última definición es una de la más acertada ya que en su mayoría los problemas con la calidad del software no se encuentran en el software, sino en la vía como las personas interpretan y aplican el concepto de calidad. No sólo se trata de desarrollar un producto que responda a los requerimientos especificados, sino que el equipo de desarrollo sea capaz de comprender e incorporarle aquellos requerimientos que el consumidor desea pero no sabe expresar.

1.2.3. Aseguramiento de la calidad.

El proceso de Aseguramiento de la Calidad o SQA por sus siglas en inglés (Software Quality Assurance) es una actividad de soporte, protección que se realiza durante todo el ciclo de vida del software, que si se aplica desde tempranas etapas supondrá grandes ventajas como: reducción de cantidades repetitivas de trabajo a realizar, costes bajos, mejoría en el tiempo de llegada del producto al mercado y unas de las cosas más importantes satisfacción del cliente.

Según Lovelle (8) el SQA se tiene en cuenta en métodos y herramientas de análisis, diseño, programación y prueba, así como en inspecciones técnicas formales en todos los pasos del proceso de desarrollo del software, además en las estrategias de prueba multiescala, el control de la documentación del software y los cambios realizados, los procedimientos para ajustarse a los estándares (y dejar claro cuando se está fuera de ellos), los mecanismos de medida y también en los registros de auditorías y realización de informes .

Monsalve (9) plantea que existen tres aspectos muy importantes con relación al SQA:

- La calidad no se puede probar, se construye.
- El SQA no es una tarea que se realiza en una fase particular del ciclo de vida de desarrollo.

- Las actividades asociadas con el SQA deben ser realizadas por personas que no estén directamente involucradas en el esfuerzo de desarrollo.

Pressman (6) plantea que SQA comprende una gran variedad de tareas asociadas:

- Preparar un plan de SQA para un proyecto.
- Participar en el desarrollo del proceso de descripción del proyecto de software.
- Revisar las actividades de ingeniería del software para verificar su consistencia con el proceso de software definido.
- Auditar el producto de software para verificar el cumplimiento del proceso de software definido.
- Asegurar que las divergencias en el trabajo de software sean documentadas de acuerdo a los estándares definidos. (8)
- Almacenar cualquier inconformidad y reportarla a la gerencia media.

El aseguramiento es indispensable para tener la total confianza en que el software cumplirá con los requerimientos de calidad, siempre teniendo en cuenta que el proceso de SQA no sólo favorecerá garantizar la calidad el producto, sino que mejorará la planificación y la ejecución de un programa de software.

Aun existiendo diversos juicios de cómo asegurar la calidad del software, todavía muchos de ellos coinciden en que el aseguramiento debe realizarse desde el inicio del proceso de desarrollo del producto, pues la prevención de fallos trae consigo menos costes que su corrección.

1.2.4. Control de la calidad.

Durante la elaboración de un producto se realizan múltiples actividades, una de ellas es el control de la calidad, cuyo propósito fundamental es asegurar la continua satisfacción de los usuarios, mediante la constante supervisión de la calidad del software y las prestaciones que ofrece el mismo.

Numerosos autores han abordado acerca del tema y han expresado conceptos tales como:

Pressman (6) declara que: “El control de calidad es una serie de inspecciones, revisiones y pruebas utilizadas a lo largo del proceso del software, para asegurar que cada producto cumple con los requisitos que le han sido asignados.”

De ahí se deduce que el control de la calidad se centra en cumplir los estándares establecidos para el desarrollo del software, a partir de las exigencias de calidad propuestas por el cliente.

Según Monsalve (9), el control de la calidad se relaciona con la vigilancia permanente de todo el proceso de desarrollo y el ciclo de vida del software. Se logra mediante la observación constante del cumplimiento de cada una de las fases y actividades involucradas en el proceso de desarrollo. Permite reducir los defectos producidos durante el ciclo de vida del producto.

El SQA desde tempranas etapas del ciclo de vida del software implica que los costes del control en fases posteriores tienden a disminuir, pues existirían menos aspectos que controlar.

Monsalve (9) declara que: para realizar un control de calidad deben ejecutarse frecuentes inspecciones a las metodologías de trabajo y el uso de las herramientas, revisiones de prototipos y de las pruebas formales de los productos finales.

El control de la calidad tiene cierta relación con la informática, pues resulta que su supervivencia depende de los productos informáticos para llegar a un correcto funcionamiento de la aplicación y que tiene como ventaja señalar los errores, a fin de que se pueda reparar y evitar su repetición. De ahí la coincidencia con lo planteado por Monsalve de la necesidad de realizar pruebas para mantener bajo control el proceso de desarrollo y así eliminar las causas de defectos en todo el ciclo de vida del software.

Se puede evidenciar en cada una de las definiciones antes planteadas que las pruebas de software juegan un papel elemental en el control de la calidad.

1.2.5. Pruebas de software.

Proceso que entra dentro del control de la calidad. La prueba de software es un elemento censor en la garantía de la calidad. Estas permiten la verificación y validación del software que fue construido anteriormente. Mediante la primera asegura que el software implementa correctamente una funcionalidad en específico y la última se comprueba que el producto compensa las exigencias del comprador.

Las pruebas de software son una serie de herramientas, prácticas y procedimientos que valoran el desarrollo de un programa determinado. Incluyen las operaciones del sistema bajo ciertas condiciones controladas y evaluando los efectos, es por eso que

la realización de ellas es factor imprescindible, pues antes de la liberación de un producto se hace necesario tener total seguridad que este se halla libre de errores, aunque se considera que existe alguna probabilidad de no estar 100% libre de errores. Pero se puede decir que mientras se realicen mayor cantidad de pruebas a los productos se logrará un producto con la calidad deseada o más aproximada a la satisfacción de las necesidades del cliente.

Para la realización de pruebas de software existen varias normas y procedimientos que toman en cuenta los objetivos de las mismas, para garantizar que sean idóneas. Probar es el proceso de ejecución de un programa con la intención de descubrir un error.

Citando a Ivar (11) y otros autores una prueba tiene éxito si se descubre un error no detectado hasta entonces.

A modo de resumen se puede decir que existen pruebas para cada etapa del ciclo de desarrollo de un software, el objetivo de cada tipo de pruebas es garantizar la calidad de acuerdo a la etapa de desarrollo del producto, para lograrlo se hace necesario cumplir una serie de parámetros que plantea OCENET (12) que el objetivo general de la etapa de pruebas es garantizar la calidad del software desarrollado y para alcanzarlo se hace necesario cumplir una serie de parámetros:

- Planificar las pruebas necesarias en cada iteración, incluyendo las pruebas de integración y las pruebas de sistema. Las pruebas de integración son necesarias para cada construcción dentro de la iteración, mientras que las pruebas de sistema son necesarias sólo al final de la iteración.

- Diseñar e implementar las pruebas creando los casos de prueba que especifican qué probar, creando componentes de pruebas ejecutables para automatizar las pruebas.

Ivar (11) y otros autores plantean que realizar las diferentes pruebas y manejar los resultados de cada una, de forma sistemática. Las construcciones en las que se detectan defectos son probadas de nuevo y posiblemente devueltas a otro flujo de trabajo, como diseño o implementación, de tal forma que los defectos importantes puedan ser arreglados.

1.2.6. Verificación y validación.

Pressman (6) plantea que “La prueba del software es un elemento de un tema más amplio que, a menudo, es conocido como verificación y validación (V&V)”.

Según Pressman (6) verificación se refiere al conjunto de actividades que aseguran que el software implementa una función específica. La validación se refiere a un conjunto diferente de actividades que aseguran que el software construido se ajusta a los requisitos del cliente.

Wallance (20) declara que la V&V abarcan una amplia lista de actividades SQA que incluye: revisiones técnicas formales, auditorías de calidad y de configuración, monitorización de rendimientos, simulación, estudios de factibilidad, revisión de la documentación, revisión de la base de datos, análisis algorítmico, pruebas de desarrollo, pruebas de validación y pruebas de instalación.

Después de analizar las bibliografías citadas se considera que lo incluido por Wallance en su definición, forma parte de lo que conduce a la calidad del software y a su vez se puede confirmar en la etapa de pruebas. Cuando se trata de V&V que se encuentren ligadas al control de la calidad, es necesario incluir además las pruebas y el proceso de revisión.

1.2.7. Estrategia.

El concepto estrategia según la bibliografía analizada es bastante antiguo, pues surgió desde los tiempos de la Antigua Grecia cuando los generales griegos dirigían sus ejércitos tanto en las conquistas como en la defensa de las ciudades. Desde esta época la definición de estrategia ya tenía tantos componentes de planeación como de toma de decisiones.

Según el diccionario de la Real Academia Española (3) estrategia es definida como “Arte de dirigir operaciones militares”, “Arte, traza para dirigir un asunto”. Pero aún así el diccionario de OCENET (12) lo define como: “Conjunto de líneas maestras para la toma de decisiones que tienen influencia en la eficacia a largo plazo de una organización.”

En el ámbito de las ciencias informáticas, la estrategia ocupa un lugar destacado para organizar el proceso de desarrollo de software y en especial, para el control de la calidad una vez obtenido el producto, pues cuando esta es bien concebida provee un mejor aprovechamiento de los recursos de un proyecto determinado ya sean

materiales o humanos. Constituye una guía a seguir para cumplir metas y objetivos ya propuestos con anterioridad.

En este estudio enfocado en el proceso de evaluación de la calidad del software, se considera que la estrategia no es más que el proceso seleccionado previamente para alcanzar el objetivo de detectar los errores de un producto informático y comprobar si cumple los fines que se le atribuyen mediante pruebas planificadas.

1.2.8. Estrategia de pruebas.

Pressman (6) plantea que una estrategia de prueba debe incluir pruebas de bajo nivel (unitarias, integración) verificando que todos los pequeños segmentos del código fuente se han implementado correctamente, así como la pruebas de alto nivel (sistema, aceptación) validando las principales funciones del sistema frente a los requisitos del cliente. Debe proporcionar una guía al profesional y un conjunto de hitos para el jefe de proyecto.

La planificación de una buena Estrategia de Prueba puede disminuir representativamente el esfuerzo preciso para el desarrollo de las pruebas proporcionadas, reducir el tiempo de elaboración y ejecución de las mismas y minimizar los altos costos que se forman. Es válido aclarar que toda estrategia de pruebas debe llevar a cabo una planificación de pruebas, diseño de casos de prueba, ejecución de las mismas, y por último, que no por eso es un factor menos importante la agrupación y evaluación de los datos obtenidos.

La estrategia que se ha de seguir al evaluar dinámicamente un sistema de software debe comenzar por el análisis de la calidad de los componentes más simples y pequeños hasta los de mayor complejidad, e ir progresando continuamente para alcanzar probar el software en su conjunto.

De la investigación realizada se tienen como resultados que la formulación de la estrategia de pruebas de software debe establecer claramente los pasos a tener en cuenta para la realización de las pruebas, tales como:

- Identificar la etapa del desarrollo del software y definir las pruebas que se aplicarán.
- Propuesta del instrumento de medición.
- El diseño y registro de casos de prueba.

- Aplicación de las pruebas propuestas.
- Documentación y evaluación de los resultados de la implementación de la estrategia propuesta.

1.2.9. Tipos de prueba.

1.2.9.1. Pruebas de unidad.

Citando a Pressman (6) es la escala más pequeña de la prueba, está basada en la funcionalidad de los módulos del programa, como funciones, procedimientos, módulos de clase, etc. En ciertos sistemas también se verifican o se prueban los drivers y el diseño de la arquitectura.

Durante las pruebas de unidad se cometen numerosos errores, por esto se hace necesario en este nivel diseñar casos de pruebas que descubran deficiencias como las que se muestran a continuación:

- Comparaciones entre tipos de datos distintos.
- Variables o comparaciones erróneas.
- Operadores lógicos de origen incorrecto.
- Terminación de bucles inapropiada o irreal.
- Bucles que manipulan variables transformadas de forma inadecuada.
- Fallo de salida cuando se halla en una iteración divergente.

Para comprobar los componentes que han sido implementados como módulos individuales, se le ejecutan pruebas o técnicas de caja negra que son aquellas pruebas que se le hacen a la interfaz de la aplicación, las cuales verifican el comportamiento de la unidad perceptible encontrada en la parte externa y las pruebas de estructura, o de caja blanca son aquellas que se le hacen directamente al código, que comprueban la implementación interna de la unidad.

1.2.9.2. Pruebas de integración.

Después de haber realizado la prueba de unidad a todos los módulos aparece la incógnita “Si todos funcionan bien por separado ¿Por qué dudar de que no funcionen todos juntos?”. Pues por supuesto que el problema está en colocarlos todos juntos, es decir a lo que llamamos interacción. Los datos pueden ser perdidos en una interfaz; un

módulo puede no tener un resultado favorable sobre otro; puede no producirse la función esencial que en realidad se desea; y así sucesivamente pueden ocurrir una serie problemas relacionados con esto.

La prueba integración según Pressman (6) es una técnica sistemática para construir la estructura del programa mientras que, al mismo tiempo, se llevan a cabo pruebas para detectar errores asociados con la interacción. Es decir que después de haber ocurrido la interacción se realizan un conjunto de pruebas para mostrar todas las deficiencias que tiene esta integración. El objetivo principal de esta prueba es seleccionar los módulos que fueron probados mediante la prueba de unidad y construir la estructura de un programa que coincida totalmente con lo que prescribe el diseño.

Citando a Ivar (11) y otros autores: “La mayoría de los casos de prueba de integración pueden ser derivados de las realizaciones de caso de uso-diseño, ya que las realizaciones de casos de uso describen como interaccionan las clases, los objetos y los componentes”.

Las pruebas de integración se dividen en dos la incremental y la no incremental, no obstante seleccionar una estrategia de integración depende mucho de las particularidades que tenga el producto, además de la planificación del proyecto, por lo general se tiende a aplicar la integración incremental, o sea, a juntar todos los módulos existentes anticipadamente. Lo cual se fundamenta en probar el programa completo, ya combinado.

Este proceso usualmente provoca desconcierto, puesto que se haya gran suma de errores y su corrección se hace bastante dificultosa, ya que no es lo mismo aislar errores a un programa, que recién comienza a otro que ya está completo. El problema aparece que en cuanto se detectan esas deficiencias pues aparecen otras nuevas y el proceso se repite en lo que parece ser un ciclo que no tiene fin.

Se emplea integración incremental cuando el programa se construye y se prueba en segmentos un poco más pequeños. Este tipo de integración tiene como ventajas que como los módulos son pequeños y funcionales hacen que los errores sean más fáciles de aislar y corregir. Existen dos estrategias de integración incremental:

- Integración descendente.
- Integración ascendente.

1.2.9.2.1. Integración descendente.

La prueba de integración descendente es un planteamiento incremental a la cimentación de la estructura de programas, en cual se componen los módulos moviéndose en recorrido hacia abajo por la jerarquía de control iniciando con el módulo principal. Los módulos dependientes al módulo de control principal se agregan en la estructura, de forma primero-en-profundidad, ó primero-en-anchura. Ver Ilustración 1.

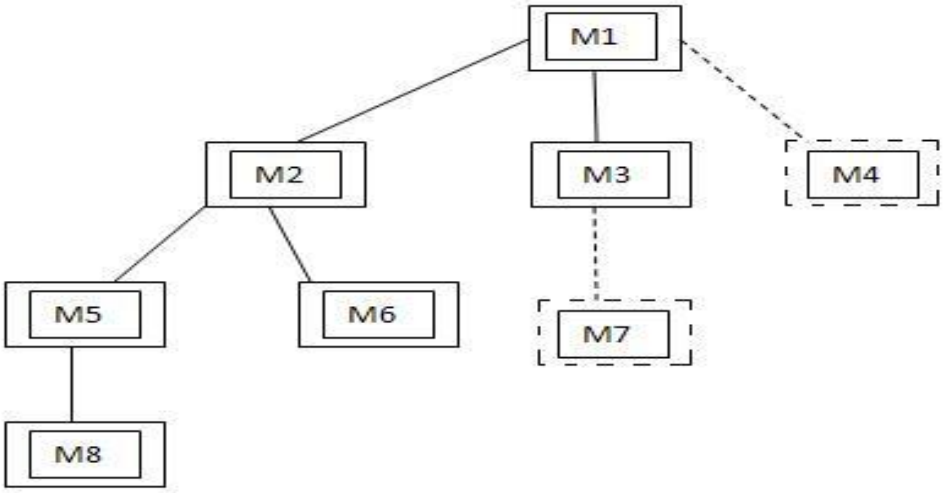

Ilustración 1 Integración descendente Fuente: Elaboración Propia

Antes de ejecutar las pruebas de integración descendentes es preciso instaurar resguardos, los cuales son una sucesión de programas que suplantan los módulos del nivel bajo. Esto se hace necesario cuando se solicita un proceso de los niveles más bajos de la categoría para conseguir comprobar apropiadamente los niveles superiores.

Para solucionar este problema se asumen tres opciones según Pressman (6):

- Retrasar muchas de las pruebas hasta que los resguardos sean reemplazados por los módulos reales.
- Desarrollar resguardos que realicen funciones limitadas que simulen los módulos reales.
- Integrar el software desde el fondo de la jerarquía hacia arriba.

1.2.9.2.2. Integración ascendente.

La prueba de la integración ascendente, como bien su nombre muestra, la construcción y las pruebas empiezan con los módulos atómicos, o sea módulos de los niveles más bajos de la organización del programa. Dado que los módulos se integran de abajo hacia arriba, de forma ascendente, el proceso propuesto de los módulos sometidos está en disposición constantemente y excluye la necesidad de resguardos. La principal desventaja de este tipo de integración es que “el programa como entidad no existe hasta que se haya añadido el ultimo módulo”.

Se puede efectuar una estrategia de integración ascendente a través de los pasos que se muestran a continuación citados de Pressman (6):

- Se combinan los módulos de bajo nivel en grupos que realicen una subfunción específica del software.
- Se escribe un controlador (un programa de control de la prueba) para coordinar la entrada y la salida de los casos de prueba.
- Se prueba el grupo.
- Se eliminan los controladores y se combinan los grupos moviéndose hacia arriba por la estructura del programa.

Mientras la integración avanza hacia arriba se hace innecesario cada vez más el uso de los revisores de prueba.

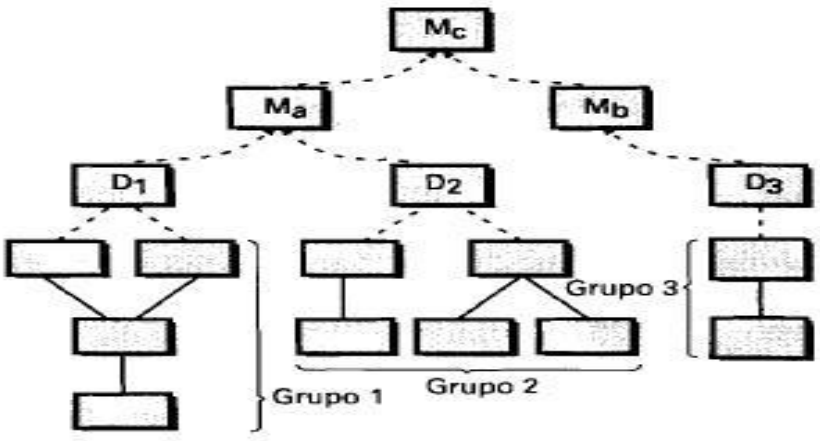

Ilustración 2 Integración ascendente tomado de Pressman (6).

1.2.9.2.3. Prueba de regresión.

La prueba de regresión reside en ejecutar nuevamente un subconjunto de pruebas que anteriormente se hayan llevado a cabo, para cerciorarse de que los cambios no han irradiado efectos adyacentes no deseados. Cada vez que se incrementa un nuevo módulo como segmento de una prueba de integración el software sufre determinados cambios.

1.2.9.2.4. Prueba de humo.

La prueba de humo es un procedimiento de prueba de integración frecuentemente utilizado cuando se ha desplegado un software empaquetado. Es delineado como un mecanismo para proyectos críticos por tiempo, admitiendo que el equipo de desarrollo del producto evalúe su proyecto sobre una base consistente.

1.2.9.3. Pruebas del sistema.

La prueba del sistema, está establecida por un conjunto de pruebas disímiles cuyo principal propósito, es practicar intensamente el sistema basado en computadora. Aunque cada prueba posee una intención diferente, todas están estrechamente relacionadas, verificando que se han constituido debidamente todos los componentes del sistema y que efectúan las funciones adecuadas. Dentro de las pruebas de sistema se encuentran las pruebas que se exponen a continuación:

1.2.9.3.1. Prueba de recuperación.

La prueba de recuperación es una de las pruebas del sistema, la misma fuerza el fallo del software de formas diversas e identifica si la recuperación se lleva a cabo acertadamente.

1.2.9.3.2. La prueba de seguridad.

La prueba de seguridad pretende verificar que los mecanismos de protección afiliados en el sistema lo resguardarán, incluso, de accesos inconvenientes.

1.2.9.3.3. La prueba de resistencia (stress).

La prueba de resistencia compone un sistema de forma que solicite recursos en cantidad, frecuencia o volúmenes inauditos como por ejemplo diseñar pruebas especiales que generen diez, interrupciones por segundo, cuando las normales son una o dos.

1.2.9.3.4. La prueba de rendimiento (carga).

La prueba de rendimiento es utilizada para experimentar el provecho del software en tiempo de ejecución dentro del contenido de un sistema completado.

Las pruebas de sistema se emplean para probar que este marcha adecuadamente como un todo. Cada prueba de este tipo, prueba primordialmente mezclas de casos de uso instanciados bajo circunstancias desemejantes. Estas condiciones recogen diferentes configuraciones de hardware como por ejemplo: procesadores, memoria principal, discos duros, etc.; desiguales niveles de carga del sistema, números de actores y tamaños de la base de datos. (12)

1.2.9.4. Pruebas de aceptación.

Generalmente estas pruebas se aplican para favorecer que el usuario valide todos los requerimientos. Las realiza el cliente final en vez de hacerlo el responsable del desarrollo del sistema, una prueba de aceptación puede ir desde un informal hasta la ejecución consecuente de una cadena de pruebas bien planificadas. De hecho, esta puede tener términos a lo largo de semanas o meses, revelando así errores recolectados que pueden ir degradando el sistema.

1.2.9.4.1. Pruebas alfa y beta.

Según Pressman (6) son procesos transcurridos por los desarrolladores para manifestar errores donde parezca que solo el cliente final puede descubrir. La prueba alfa es llevada a cabo, por un cliente, en el lugar de desarrollo. Se explota el software de forma natural con el desarrollador como espectador del usuario e inspeccionando los errores y a su vez las dificultades de uso. Las pruebas alfa se realizan en un entorno controlado. Sin embargo, la prueba beta es efectuada por los clientes finales del producto en los lugares de trabajo de los usuarios. A discrepancia de la prueba alfa, el desarrollador no está presente habitualmente. Así, la prueba beta es una aplicación en vivo del software en un contexto que no puede ser observado por el desarrollador.

1.2.10. Métodos de prueba.

Los métodos de pruebas del software son aquellos que permiten diseñar pruebas donde se puedan encontrar disímiles tipos de errores en el menor tiempo posible y como parte de los métodos, las pruebas de evaluación dinámica facilitan diferentes razonamientos para componer casos de prueba que induzcan a fallos en los

programas. Se clasifican en dos grupos pruebas de caja blanca o estructural y pruebas de caja negra o funcional.

1.2.10.1. Pruebas de caja blanca o estructural.

Las pruebas de caja blanca, también conocidas como pruebas de caja transparente o de cristal, se concentran en el diseño de casos de pruebas atendiendo el comportamiento interno y la estructura de control del programa. Luego de ser diseñadas se prevé obtener casos de pruebas que se ejecuten al menos una vez, todas las sentencias del programa y las condiciones lógicas ya sean verdaderas o falsas. Son denominadas además como estructurales o de cobertura lógica, en ellas se procura examinar todo lo referente a la estructura interna del código, exceptuando complementos referidos a fundamentos de entrada o salida. Para esta prueba es imprescindible tener en cuenta estos tres aspectos:

- Conocer el desarrollo interno del programa, determinante en el análisis de coherencia y consistencia del código.
- Considerar las reglas predefinidas por cada algoritmo.
- Comparar el desarrollo del programa en su código con la documentación pertinente. La primera parte de esta prueba es el análisis estático.

1.2.10.1.1. Técnicas de Prueba de caja blanca.

En la actualidad existen diversas técnicas de prueba de Caja Blanca, como por ejemplo la prueba del camino básico, la prueba de condición, la prueba de flujo de datos y la prueba de bucles.

Citando a Pressman (6) la prueba del camino básico: Esta prueba posibilita que el diseñador de casos de prueba pueda alcanzar una medida de la complejidad lógica de un diseño procedimental y el uso de esa medida como guía para el esclarecimiento de un conjunto elemental de vías de ejecución. Los casos de prueba obtenidos del conjunto básico certifican que durante la ejecución de la prueba se produce al menos una vez cada sentencia del programa. Para poder llevar a cabo esta técnica es necesario tomar en consideración los siguientes pasos:

1. Luego de haber precisado el diseño o el código fuente, se traza el grafo de flujo asociado.

2. Se calcula la complejidad ciclomática del grafo.

3. Se establece un conjunto básico de vías independientes.

4. Se disponen los casos de prueba que exijan la ejecución de cada camino del conjunto básico.

La prueba de condición: Es un método de diseño de casos de prueba que visualiza el contexto lógico contenido en el módulo de un programa.

La prueba de flujo de datos: con esta técnica se seleccionan caminos de prueba de un programa acorde con el sitio en donde se encuentren las definiciones y los usos de las inconstantes o variables del programa.

La prueba de bucles: como su nombre indica se centra únicamente en la eficacia de las construcciones de bucles.

1.2.10.2. Pruebas de caja negra o funcional.

Citando a Pressman (6) las pruebas de caja negra, igualmente nombradas como pruebas de comportamiento o funcionalidad, tienen como objetivo principal realizar pruebas a la interfaz de usuario del software desarrollado, es decir, que se centra generalmente en los requisitos funcionales del producto. Permite al ingeniero de software adquirir conjuntos de condiciones de entrada que preparen plenamente los requisitos funcionales de un programa. Las pruebas de caja negra aspiran hallar fallas de las siguientes cualidades:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a Bases de Datos externos.
- Errores de rendimientos.
- Errores de inicialización y de terminación.

1.2.10.2.1. Técnicas de Prueba de caja negra.

Citando a Pressman (6) la Técnica Partición de la Equivalencia: es un método de prueba de la caja negra que divide el dominio de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de error (por ejemplo procesamiento incorrecto de todos los datos de carácter) que de otro modo requerirían la ejecución de muchos

casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar.

Resumiendo es aquella técnica que permite definir casos de pruebas por cada requisito funcional, reduciendo la cantidad de los mismos, y a su vez la verificación del sistema con valores válidos e inválidos para condiciones de entrada.

1.2.11. Modelos de prueba.

La ingeniería cuenta con diversos modelos de prueba de software a continuación se establecen características de cada uno de ellos:

1.2.11.1. Modelo en “V”.

En el documento el Proceso de la Prueba del Software el profesor Luis Vinicio (22) expone su criterio acerca del Modelo en “V”: a través de este modelo se describe a nivel muy elevado de abstracción, las etapas del ciclo de vida del software en las que se incluye la prueba. En el mismo la prueba se inicia con las revisiones técnicas a los requerimientos y delineando los primeros casos de prueba de aceptación. A continuación se revisa que la arquitectura satisfaga los requerimientos y se definen los casos de prueba de sistema; luego se examina la modularidad del diseño y se esbozan casos de prueba de integración, por último se analizan los algoritmos y al desarrollo de los casos de prueba de unidad. En este modelo los documentos que se generan de estos procesos suelen ser vulnerables a volverse precipitadamente frágiles (por consecuencia de cambios) o sin modificar (por el conflicto para ejecutar esos cambios).

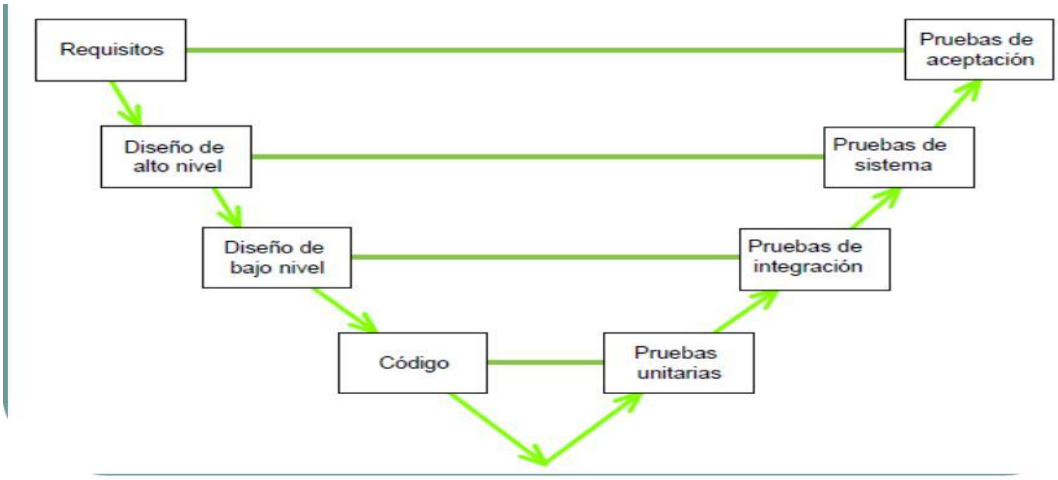

Ilustración 3 Modelo en V tomado de El proceso de la prueba del software (22).

1.2.11.2. Modelo espiral.

Según Pressman (6) es un modelo de proceso de software evolutivo que conjuga la naturaleza iterativa de construcción de prototipos con los aspectos controlados y sistemáticos del modelo lineal secuencial.

Dicho modelo es bastante favorable pues como su nombre indica el software irá evolucionando a medida que progresa el proceso, y el cliente y el desarrollador comprenden y reaccionan ante riesgos por cada una de las fases evolutivas. Proporciona además versiones cada vez más completas del producto, por estas características este será el modelo que sustentará la estrategia propuesta.

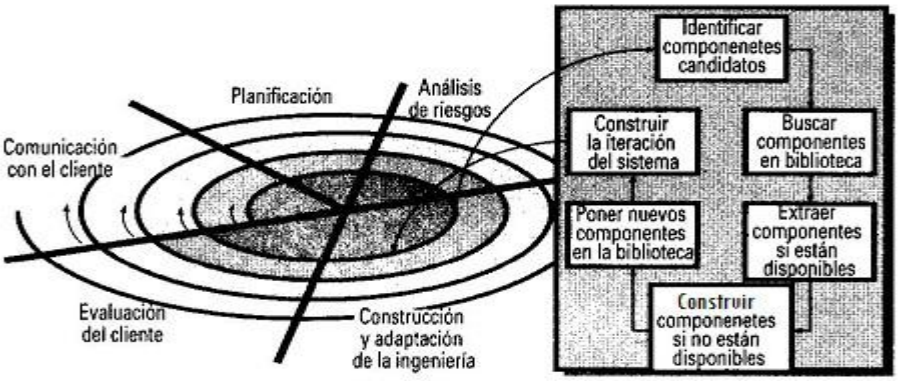

Ilustración 4 Modelo espiral tomado de Pressman (6)

1.2.11.3. Modelo en cascada.

Según Iribarne (19) tradicionalmente los ingenieros del software han seguido un enfoque de desarrollo descendente (top-down) basado en el ciclo de vida en cascada (análisis-diseño-implementación) para la construcción de sus sistemas, donde se establecen los requisitos y la arquitectura de la aplicación y luego se va desarrollando, diseñando e implementando cada parte software hasta obtener la aplicación completa implementada.

Lo que plantea este modelo es que su aplicación se hará de forma descendente por todo el ciclo de vida del proyecto, a través de sus fases y por cada una de actividades de estas fases, hasta llegar a obtener la aplicación completa. Sin dejar de mencionar por supuesto, las pruebas a las que estará sometido el software. A continuación se muestra una figura de dicho modelo:

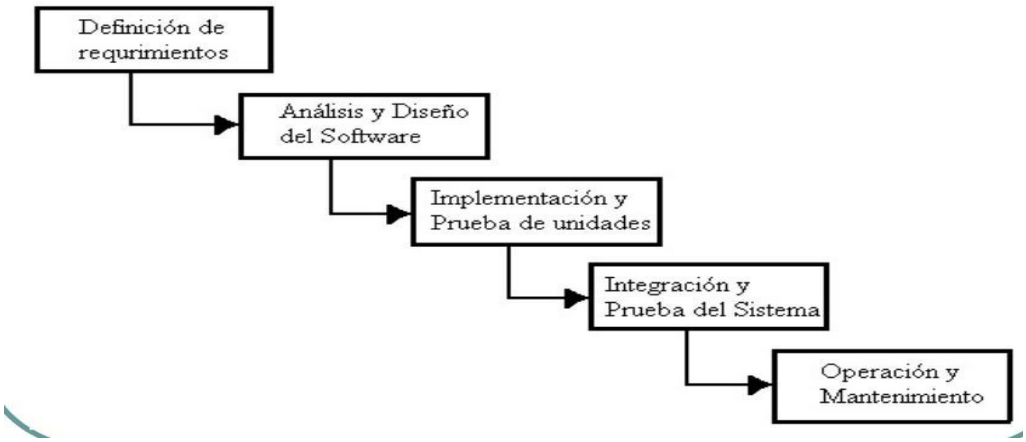

Ilustración 5 Modelo en cascada tomado de Pressman (6)

1.2.12. Factoría de Software.

Según Ivan Aaen (13), “la fábrica es una organización habitada por personas comprometidas en un esfuerzo común, el trabajo es organizado de una manera o de otra, se usa la estandarización para la coordinación y formalización donde la sistematización es importante.”

El proceso fabril habitual maneja la técnica de línea de producción y montaje, con el único objetivo de disminuir los costes de producción por el camino de la estandarización de productos y procesos. La producción por montaje se determina relacionando secuencias de pasos que confluyen hacia una línea incesante en la que se acoplan los productos obtenidos en la última etapa.

Si se aplica el concepto de fábrica en el ámbito de la producción de software, se considera muy atinado el planteamiento de Certum Arnoldo Díaz (14), quien expresa que “el fundamento para establecer fábricas de software, fue basado en tratar de obtener los beneficios que las líneas de producción industrial produjeron en la calidad de los productos así como la productividad lograda.”

Según lo expuesto por Arnoldo para el éxito de las Factorías de Software es radical alimentarse de esta experiencia, aprovechándola para su producción a través de la determinación del trabajo por líneas de producción, en el cual cada una posea un proceso y un flujo adecuadamente concretos.

1.2.12.1. Definiciones de Factoría de Software.

A finales de los años 1960 germina en la industria del software la definición de Fábrica de Software para referirse al proceso de desarrollar y reutilizar componentes de

software que pueden ser comunes a cualquier otro producto informático que se desarrolle. Este concepto ha progresado como resultado de la definición de prácticas y procedimientos, conjuntamente con la aparición de diferentes herramientas de software. No obstante, el concepto de Factoría de Software no es desconocido, a partir de las últimas décadas ha alcanzado un notable impacto con el perfeccionamiento de modelos. A continuación se expresan elementos a considerar por una Factoría de Software que exponen diversos autores:

- A principio de los años 90 Fernstrom (15) afirma “Una organización fabril para el desarrollo de software debe tener claro el asunto del “software único”, es decir, todo software es único, pero algunas partes de ellos se pueden repetir en varios proyectos. El proceso industrial debe contener el desarrollo, almacenamiento y montaje de partes reutilizables en un producto”.

- Basili (16) en 1992 menciona que “Una organización con características de Factoría de Software debe poseer una estructura de construcción de software basada en componentes. Los componentes utilizados en la construcción del software pueden ser desarrollados por una unidad de producción de componentes (factoría de componentes). La factoría de componentes es la base para la implementación de una Factoría de Software.”

El propio autor antes citado agrega que “Una Factoría de Software debe poseer un conjunto de herramientas estandarizadas para la construcción de software, bases históricas para ser usadas en la dirección de proyectos y principalmente, poseer un alto grado de reutilización de código en el proceso de desarrollo de un determinado software, apoyado en una base de componentes reutilizables.”(16)

Estos autores coinciden en que conviene que las Factorías de Software sean tuteladas por metodologías de avance de software, las cuales son examinadas en la disciplina de la Ingeniería de Software y orientadas al continuo progreso de sus procesos. También consideran que se deben precisar los roles y ocupaciones que conlleven a originar productos en máximas cantidades.

En las fábricas de software el incremento en la productividad tiene su apoyo en la reutilización de numerosos de los componentes que completan un software y del conocimiento tomado, con el resultado de realzar la calidad, restringir los peligros, extender la productividad y disminuir el período de desarrollo.

La labor primordial de una Factoría de Software es delimitar un proceso patrón para el perfeccionamiento del software, donde haya una estrecha comunicación con el cliente y las estimaciones que se efectúen apoyadas en los antecedentes históricos. Reutilizar un componente avala que el producto en construcción conserve una calidad igual que el de su predecesor.

1.2.13. Metodología de desarrollo de software de un producto en AplicativosSIG.

La Factoría de Software AplicativosSIG se basa en la Metodología RUP, la cual cuenta con cinco fases de desarrollo: Inicio, Elaboración, Construcción y Transición. En ella se cuenta con cinco grupos de Factoría en funcionamiento como las fases de RUP, el producto que se esté desarrollando debe transitar por cada uno de estos grupos, para la posterior terminación de la iteración y llegar a su liberación. Cada grupo tiene designado un líder, el cual planea las actividades y tareas a realizar en el mismo. Este proceso de desarrollo fue implementado por la fábrica, por lo que a continuación se explican para lograr un mejor entendimiento.

Grupos	Descripción	Rol	Herramientas
Ingeniería	Este grupo se divide en dos categorías Preliminar y Requisitos. En el primero se delimitan los convenios con el cliente, el alcance del proyecto, la identificación de los posibles riesgos, la estimación de los recursos del proyecto. En el último los desarrolladores y clientes deben definir las características que tendrá el sistema, y la confección de la documentación del expediente del proyecto.	Líder de Grupo. Analista. Planificador.	Visual Paradigm.
Integración y componente	Se integra y se crean los componentes. Además se ensambla la Interfaz de Usuario con la Base de Datos.	Líder de Factoría. Líder de Grupo Desarrollador. Planificador.	Netsbeans. PHP como Lenguaje de Programación.
Base de datos espaciales	En este grupo se llevan a cabo las acciones de cartografía, funciones y almacenamiento de los SIG.	Líder de grupo. Diseñador de Base de Datos Espaciales.	PGAdmin. PostgreSQL. PostGIS.

		Planificador.	
Interfaz de usuario	El grupo se encarga del diseño, las imágenes y los componentes visuales de los SIG.	Líder de grupo. Diseñador de Interfaz de usuario. Planificador.	Biblioteca EXT. Biblioteca EXT.JS.
Gestión	Gestionar los Recursos ya sean materiales o humanos, el marketing y la contabilidad.	Líder de grupo. Planificador.	REDMINE.

Tabla 1 Grupos de la Factoría AplicativosSIG. Fuente: Elaboración Propia.

1.2.14. Herramientas a utilizar

La amplia variedad de tecnologías existentes que permiten automatizar en cierta medida el proceso de prueba, es de gran utilidad para decidir qué herramienta utilizar, a continuación se muestran alguna de ellas:

1.2.14.1. Herramienta para el análisis estático de código fuente.

Jtest: posee diversas características entre ellas la comprobación automática de la construcción del código fuente, es decir, las pruebas de caja blanca, la correcta funcionalidad del código, aquí entraría directamente las llamadas pruebas de caja negra, y por último el mantenimiento de la integridad del código, o sea las pruebas de regresión. Por lo general esta herramienta se aplica sobre clases Herram Java y JavaScript.

Ilustración 6 Herramientas para el análisis estático de código fuente tomado de Natalia (23)

1.2.14.2. Herramienta para pruebas de carga y stress.

OpenLoad Tester: es una herramienta de optimización basada generalmente en el navegador para pruebas de carga y stress de sitios web dinámicos. La misma permite la elaboración de escenarios de ejecución y a su vez poder ejecutarlos de forma reiterada, figurando la carga de un contexto de producción real de la aplicación como si estuvieran usándola muchos usuarios al mismo tiempo.

Ilustración 7 Herramienta para pruebas de carga y stress tomado de Natalia (23)

Otra de las herramientas utilizadas para este tipo de pruebas es la denominada: Benchmark Factory la cual es la más idónea para simular el acceso de millares de usuarios a sus servidores de Base de Datos, archivos, internet y correo, aislando así las deficiencias asociadas con la sobrecarga del sistema.

Ilustración 8 Herramientas para pruebas de carga y stress tomado de Natalia (23)

Shadow Security Scanner: Es un completísimo y efectivo escáner de vulnerabilidades para la plataforma de Windows nativa, aunque también examina servidores de cualquier otra plataforma revelando brechas en Unix, Linux, FreeBSD, OpenBSD y Net BSD. Por su arquitectura, Shadow Security Scanner también descubre fallos en CISCO, HP y otros equipos de la red.

Es una herramienta que permite escanear la red en busca de debilidades, lo que proporciona una ayuda a la hora de establecer una evaluación al nivel de seguridad que posee el sistema, a través de la red y a su vez impide el aparecimiento de algún inconveniente que ponga en riesgo el correcto funcionamiento del sistema.

JMeter: El JMeter es una herramienta libre, además es una herramienta Java, que permite realizar pruebas de rendimiento y pruebas funcionales sobre aplicaciones web. Es una herramienta de carga para llevar acabo simulaciones sobre cualquier recurso de software. De las herramientas gratis, es la más completa y útil para realizar pruebas de carga.

Se caracteriza por sus funcionalidades para realizar pruebas de stress, carga y rendimiento a las aplicaciones de forma independiente, al permitir aislar los subsistemas de aplicación. Puede ser configurable debido a que permite delimitar la cantidad de usuarios que van a figurar y las pruebas que se aplicarán al marco de trabajo.

Para lograr un mejor entendimiento del funcionamiento de la herramienta JMeter se muestran algunas de sus características:

- Actúa como un proxy local cimentando un camino de navegación aleatoria y así figurar la visita del usuario, por donde transita toda la información en tiempo real.
- Reconoce estudiar su utilidad, dados determinados sucesos como la navegación en el sistema.
- Enumera los errores ocasionados durante la transferencia de datos del servidor al navegador, que por lo general no son visibles para el cliente pero si pueden provocar grandes dificultades ya sea a corto o largo plazo.
- El muestreo de la información a través de tablas y gráficas facilita la comprensión del flujo de datos y sus características.

Ilustración 9 Herramienta para pruebas de carga y stress. Tomado: Herramienta JMeter.

1.2.14.3. Herramientas de diagnóstico.

PerformaSure: como su nombre indica es una herramienta de diagnóstico de rendimiento utilizada para el análisis en entornos distribuidos J2EE, permitiendo la búsqueda las carencias de rendimiento descubiertos en tiempo real, desde la avenencia del usuario final en fase de producción, hasta la línea de código que concibe el problema.

1.3. Objeto de estudio

1.3.1. Descripción general

En la actualidad las Factorías son un elemento fundamental en la reutilización, ya sea código, recursos, software, la experiencia de los desarrolladores, entre otros. Las mismas tienen múltiples ventajas, una de las más importantes es la de personal altamente calificado en el ámbito de las tecnologías, debido a que se le hace necesario especializarse en un rol específico del ciclo de vida del producto,

centralizando todos sus esfuerzos en esa ocupación definida. Por ello es necesario un especialista en el proceso de pruebas.

El especialista en el proceso de pruebas de una Fábrica de Software, es el encargado específicamente de encontrar errores en las funcionalidades del sistema, para lograr que llegue con la menor cantidad de errores posibles a manos del cliente, cumpliendo sus exigencias. Además de elaborar todos los documentos que contiene este proceso, revisarlos y aplicar la estrategia de pruebas.

Con una estrategia de prueba en el nuevo modelo de formación se logrará una mejor organización y utilización de los recursos, sin dejar de mencionar la ejecución de pruebas en cada una de las fases del ciclo de vida del producto. Pues con anterioridad se esperaba la terminación total del software, para luego ser sometido al proceso de prueba, lo que provocaba llevar a cabo un gran número de iteraciones y la entrega tardía del mismo.

1.3.2. Situación problemática

En muchas ocasiones los proyectos de la UCI no cuentan con una estrategia para llevar a cabo el proceso de prueba, ni con una guía oficial para orientar a los interesados en este tema. Lo que trae consigo que estos procedimientos no se hagan durante todo el ciclo de vida del software, sino al terminarse el producto. AplicativosSIG es un ejemplo de lo anterior expuesto, pues no cuenta con una planificación de cuando pueden ejecutarse las pruebas, provocando encontrar errores al realizarse las pruebas de liberación, y a su vez entrega tardía del producto. De ahí proviene la desconfianza de los clientes, retrasos en la liberación, porque es necesario realizarse más iteraciones, también mucho trabajo debido a que las actividades y los roles no fueron organizados, además de no saberse cuales son exactamente las necesidades del cliente, por no tener una documentación concreta de lo que este necesita y de los productos que se ofrecen. Todas estas necesidades conllevan a la afectación de la eficiencia, la calidad y el ciclo de vida del software.

Las pruebas de liberación son ejecutadas por el Centro para la Excelencia en el Desarrollo de Proyectos Tecnológicos (CALISOFT).

CALISOFT es la continuidad de la Dirección de Calidad de Software que pertenecía a la Infraestructura Productiva (IP) de la UCI, ampliando sus perspectivas y servicios, de forma que se abran las puertas a otras organizaciones productoras de software en Cuba. El centro cuenta con cinco áreas fundamentales entre ellas el Departamento

de Pruebas de Software, el cual incluye el Laboratorio Industrial de Pruebas de Software (LIPS), donde se llevan a cabo las pruebas de liberación. Con la creación del LIPS se asegura que cada artefacto haga lo adecuado en todo momento, ya que hace posible una gestión integral del valor añadido mediante el cumplimiento y la superación de las expectativas del cliente.

1.4. Análisis de otras soluciones existentes

Se realizó un análisis detallado de las estrategias existentes en la universidad con el objetivo de encontrar alguna que se ajustará a las características de una fábrica de software. A continuación se mostrarán algunas descripciones de las estrategias usadas en la universidad.

a) La estrategia de prueba para Juegos de Realidad Virtual propuesta por la facultad cinco no se ajusta a la Fábrica de Software del departamento Geoinformática, ya que no examina dentro de los tipos de prueba de sistema, las pruebas de seguridad que son un elemento fundamental de la protección incorporadas al sistema. La prueba de resistencia y la anterior mencionada, son muy indispensable en las necesidades de la Fábrica por lo que se debe tener en cuenta para la propuesta de la estrategia.

   Esta estrategia plantea: un acumulado de listas de chequeo, conjuntamente con la aplicación de cuatro tipos de pruebas. La misma está precisada incrementalmente de adentro hacia afuera, es decir a través de la prueba de unidad constituida para valorar el correcto funcionamiento de los módulos, la prueba de integración, la prueba de sistema y por último la prueba de aceptación.

   Listas de chequeo:

   1. Revisión al Guión Técnico: Aplicar la lista de Chequeo propuesta.
   2. Revisión a los Requerimientos: Aplicar la lista de Chequeo propuesta.

   Tipos de prueba:

   1. Prueba de unidad: probar cada uno de los módulos por separado.
   2. Prueba de integración: probar la interacción de los módulos.
   3. Pruebas de sistema: verificar que el sistema cumple con los requisitos.

4. Pruebas de aceptación: validar el producto con el cliente.

b) La estrategia de prueba para el producto Heredia no es aplicable para AplicativosSIG, ya que Heredia es un libro virtual, donde no se realiza una integración de módulos de funcionalidad, por lo que se concurrirá a la aplicación de la estrategia, de acuerdo a la aplicación y los tipos de pruebas a ejecutar. Es importante esclarecer que la estrategia del producto no es manejable para adaptarla a ningún otro proyecto, mucho menos a una fábrica ya que plantea pruebas trazadas únicamente para el producto Heredia.

   La estrategia plantea: Los tipos de pruebas efectuados como son la documentación, validación, usabilidad, instalación y perfil de desempeño.

1.5. Conclusiones parciales

En el presente capítulo, se examinaron las definiciones del término calidad, que varían desde diferentes aspectos. Se muestra además la correspondencia, entre el término “calidad” y la aparición de otras definiciones, tales como “Aseguramiento de calidad”, “calidad de Software” y “Control de la calidad” pertenecientes al ámbito de la informática y que es de interés para esta investigación. Se analizaron varios modelos de prueba y tipos de pruebas de software donde cada uno presentaba características importantes para determinar cuáles son los convenientes para desarrollar una metodología que permita realizar la verificación y validación de la calidad de los productos informáticos. Quedaron establecidos los pasos que van a guiar la confección de la estrategia a proponer, sentando así las bases para la confección de la propuesta de solución. Como resultados se obtienen los conceptos asociados para un mejor entendimiento del trabajo en cuestión.

CAPÍTULO 2: PROPUESTA DE LA SOLUCIÓN

2.1. Introducción

En este Capítulo se describe una estrategia de pruebas, con el objetivo de maximizar la calidad en el proceso de desarrollo de software del Centro GEySED. Para obtener un producto de alta calidad es necesario durante la realización de las pruebas se lleve a cabo una buena estrategia de pruebas, la misma abarca la planificación, la ejecución y los resultados, teniendo en cuenta los recursos que se van a emplear tanto humanos como materiales. Siempre especificando los artefactos que se generan al aplicarse la estrategia y los roles que intervienen en la confección de cada de unos de ellos, conjuntamente con las herramientas que automatizarán y acelerarán la aplicación de las mismas.

2.2. Descripción de la estrategia.

La Factoría AplicativosSIG cuenta con ocho productos, los que se nombran a continuación: SIGSalud, SIG-INSMET, SIG-Verano, SIG-INRH, SIGRutas siendo este último utilizado como piloto para la aplicación de la estrategia de prueba que se propone.

SIGRutas permite consultar y analizar la información referente a las rutas de transporte obrero de la UCI. El mismo presenta cuatro módulos en general, ellos son:

- Módulo de navegación: Este módulo contempla el trabajo con los mapas y es fundamental su presencia no solo en esta aplicación sino en toda aquella que trabaje con información geoespacial.
- Módulo de localización: En esta parte se han agrupado las funcionalidades asociadas con la localización de los elementos georreferenciables.
- Módulo de opciones: Este módulo contempla una serie de funcionalidades propias del trabajo con este sistema. Ejemplo: Localizar parada más cercana.
- Módulo administración: Interactúa con datos y funciones críticas del sistema, tales como agregar y modificar los datos de los elementos existentes en la base de datos así como agregar o desactivar funcionalidades del mismo sistema.

Por las características antes descritas se hace necesario realizar pruebas que validen la calidad durante todo el proceso de desarrollo. Para la correcta aplicación de la estrategia, inicialmente se deben especificar las fases en las que se aplicarán las pruebas, para ello se tiene en cuenta que las pruebas, según la metodología RUP que es por la que se rigen todos los productos de la Fábrica donde tiene lugar la investigación, se aplican en la fase Construcción y Transición. Pero esta vez se empezará desde la fase Inicial y teniendo en cuenta los grupos hechos por la Factoría.

En la fase Inicial se diseñarán los casos de pruebas para definir la misión de evaluación. Luego en la de Elaboración se trazará el plan de pruebas definiendo también la misión de evaluación, además de probar y evaluar pruebas, verificando así el enfoque de la misma. En la fase de Construcción se aplicarán las pruebas unitarias para comprobar el funcionamiento de cada uno de los módulos por separado, seguidamente las pruebas de integración para verificar el funcionamiento correcto del ensamblaje del sistema completo y pruebas de sistema para comprobar la correspondencia con las características establecidas. Por último en la fase de Transición se ejecutarán las pruebas de aceptación, donde el cliente evaluará su total conformidad con el producto final.

De acuerdo a los grupos de la Factoría, las pruebas serán aplicadas en el grupo de Ingeniería, donde se diseñarán los casos de pruebas, se esbozará el plan de pruebas y la ejecución de las pruebas unitarias; y en el Integración de componentes se llevarán a cabo las pruebas de integración, sistema y aceptación.

En la estrategia propuesta a cada tipo de prueba se le realizarán tres iteraciones, aunque al ejecutarse las mismas minimicen las posibilidades de detectar no conformidades, pero sí asegurando un producto con los mejores resultados y a su vez validando completamente el éxito de la prueba. Realizada la primera iteración deben obtenerse las no conformidades, las cuales serán entregadas a los desarrolladores, para luego ser corregidas en dependencia del nivel de complejidad que posean.

2.2.1. Para cada iteración se debe:

Establecer la configuración del entorno de prueba, identificar los objetivos y los artefactos que se entregan en las pruebas, aplicar la estrategia de prueba de acuerdo a la prueba que se esté aplicando y por último, mostrar el resultado de la prueba, ya sea con las no conformidades encontradas o con un resumen de las mismas.

La estrategia de prueba debe definir las tácticas en cuanto a los aspectos que a continuación se mencionan:

- ¿Qué metodología de pruebas se va a usar?
- ¿Cuál será el entorno de pruebas?
- ¿Cómo deberá estar conformado el equipo de calidad?
- ¿Qué técnicas específicas de pruebas serán usadas para probar el software?
- ¿Qué herramientas se usarán?

2.3. Metodología de pruebas

El problema del software se reduce a la dificultad que afrontan los desarrolladores para sistematizar las innumerables cadenas de trabajo de un proyecto. La entidad desarrolladora de software solicita una forma fusionada de trabajar. Necesita un procedimiento usual de trabajo, que componga las múltiples fases del proceso de desarrollo del software, un proceso que:

- Provea una guía para dictaminar los movimientos de un equipo.
- Detalle los artefactos que convienen desplegarse.
- Administre las labores de cada desarrollador de manera independiente y en equipo.
- Convide criterios para el registro, la comprobación de los productos y actividades del proyecto.

Para la elaboración de la estrategia se utilizará la metodología de desarrollo basada en RUP adaptada a la Fábrica AplicativosSIG, debido a que esta es la usada en el desarrollo de esta Factoría de Software.

2.4. Entorno de pruebas.

El entorno de prueba está establecido por todos los activos fijos tangibles, que tenga a su disposición el equipo de calidad. Los recursos varían de acuerdo a las necesidades actuales que tenga el proyecto.

Los recursos humanos del proyecto integran también parte del entorno de prueba, pues los trabajadores son el eslabón más importante en el proceso de desarrollo del

software. Formalizar los recursos humanos del proyecto, es un paso vital para la estrategia que se planea.

Para la Factoría se precisan veintitrés máquinas clientes, siempre teniendo en cuenta la cantidad de recursos humanos existentes en el proyecto y la cantidad de equipos de trabajo que le fueron establecidos. Estas deben estar conectadas mediante un switch al servidor de aplicaciones, a través del protocolo http y este a su vez se conecta con el servidor de base de datos, para garantizar el acceso a la aplicación sin problemas.

2.5. Descripción de los roles de la fase de prueba de la Factoría.

La estrategia de la metodología RUP, contempla la presencia de cinco roles responsables de la aplicación de las pruebas al software. Los mismos serán ocupados de acuerdo a los conocimientos que debe poseer el individuo. Seguidamente se refieren dichos roles y competencias:

- Administrador de calidad: quien responde por el trabajo del equipo, contribuye con el cumplimiento del plan de pruebas y la correcta consumación de la estrategia de prueba. Es una persona bien instruida, su objetivo principal es cerciorarse que la aplicación concuerda con los requerimientos y detectar la mayor cantidad de errores. Evalúa los efectos que se consiguen al perpetrar las pruebas de calidad. También estará facultado para elegir las pruebas que se ejecutarán y monitorear la aplicación de las mismas.

- Diseñador de pruebas: este rol deberá ser ocupado por alguien que tenga un considerable conocimiento de las herramientas de pruebas, puesto que será el máximo responsable en la selección de las herramientas con las que se efectuarán las pruebas. Tendrá conocimiento de la ingeniería de software, de la metodología basada en componentes, pruebas de software y del negocio, debido a que es el encomendado de diseñar los casos de prueba.

- Probador: este rol será el responsable de ejecutar las pruebas tal y como fueron planificadas, para ello deberá tener conocimientos elementales de la metodología basada en RUP, de las pruebas de software, y muy importante, debe ser un especialista en las herramientas escogidas para la ejecución automática de las pruebas.

- Analista de prueba: quienes asuman este rol deben tener total dominio de las características del sistema creado en el proyecto Factoría. Identificar y delimitar las pruebas que se requieren. Monitorear el progreso de las mismas y el

resultado en cada ciclo de prueba. Evaluar la calidad total experimentada como un resultado de las actividades de prueba.

- Revisor técnico: es el encargado de evaluar la calidad de los productos desarrollados en cuanto al cumplimiento del proceso desarrollo del software. Debe tener conocimiento de cómo elaborar el Plan de Revisiones, las listas de chequeos que se aplican a los artefactos generados durante el ciclo de vida de los productos y que se encuentran recogidos en el expediente de proyecto obteniéndose luego las no conformidades.

Las competencias antes expuestas, serás desempeñadas por quienes integren el equipo de pruebas, el cual será responsable de desplegar los artefactos que se logran de este flujo de trabajo, a medida que se avance en cada paso de la estrategia.

Los integrantes del equipo de calidad serán seleccionados del personal del departamento Geoinformática, con el objetivo de que las pruebas se realicen puramente de forma independiente y teniendo en cuenta las características de las fábricas de software se propone la presencia de tres roles en la realización de las pruebas: Administrador de la calidad, Diseñador de pruebas y Probador.

- Administrador de la calidad: desempeñará todas las actividades propuestas por RUP, además el rol del Analista de prueba y de Revisor técnico, pues es el responsable de guiar todo el proceso del equipo de pruebas, a su vez está autorizado a identificar y delimitar las pruebas que se efectuarán, competencia perteneciente al Analista de prueba, y elaborar un Plan de Revisiones identificando las listas de chequeos a utilizar para identificar las no conformidades actividades que realiza el Revisor técnico.

- Diseñador de pruebas: será el Analista del sistema pues tienen conocimiento pleno de los requisitos y las funcionalidades que ejecutará el mismo, permitiéndole confeccionar los diseños de casos pruebas.

- Probador: ejercerá el rol de Desarrollador pues tienen dominio del código fuente y de la interfaz, incluyéndose probadores externos del Grupo de Calidad del departamento Geoinformática.

2.6. Actividades definidas para el proceso desarrollo de cada prueba.

Estas actividades que se concretan a continuación son pasos que se formulan para las pruebas que componen la estrategia. Las actividades planificar, diseñar e implementar

prueba son comunes en cada tipo de prueba que se hace, sin embargo la actividad implementar prueba sólo se realiza en el tipo de prueba de sistema, puesto que en él se realizan pruebas automatizadas que generan componentes de pruebas.

2.6.1. Actividad: Planificar prueba

El propósito de la planificación de la prueba es determinar y calcular planificar los esfuerzos que se requieren para cada iteración de las pruebas, cumpliendo las tareas que a continuación se mencionan:

- Describir una estrategia de prueba.
- Estimando las exigencias para el esfuerzo de una prueba, dígase recursos humanos y sistemas necesarios.
- Planear el esfuerzo de una prueba.

Los desarrolladores de pruebas desarrollan una estrategia de prueba para cada iteración donde definen que tipo de pruebas ejecutar, cómo y cuándo ejecutar dichas pruebas.

2.6.2. Actividad: Diseñar prueba

El objetivo de diseñar pruebas es sencillamente identificar y definir los casos de pruebas para cada construcción, identificar y estructurar las tácticas de pruebas detallando cómo efectuar los casos de pruebas.

2.6.3. Actividad: Implementar prueba

En esta actividad lo más importante es automatizar los procedimientos de prueba creando componentes de prueba. Dicha implementación no se realiza en todas las pruebas, sólo en las pruebas que son automatizadas. Ver Ilustración 9.

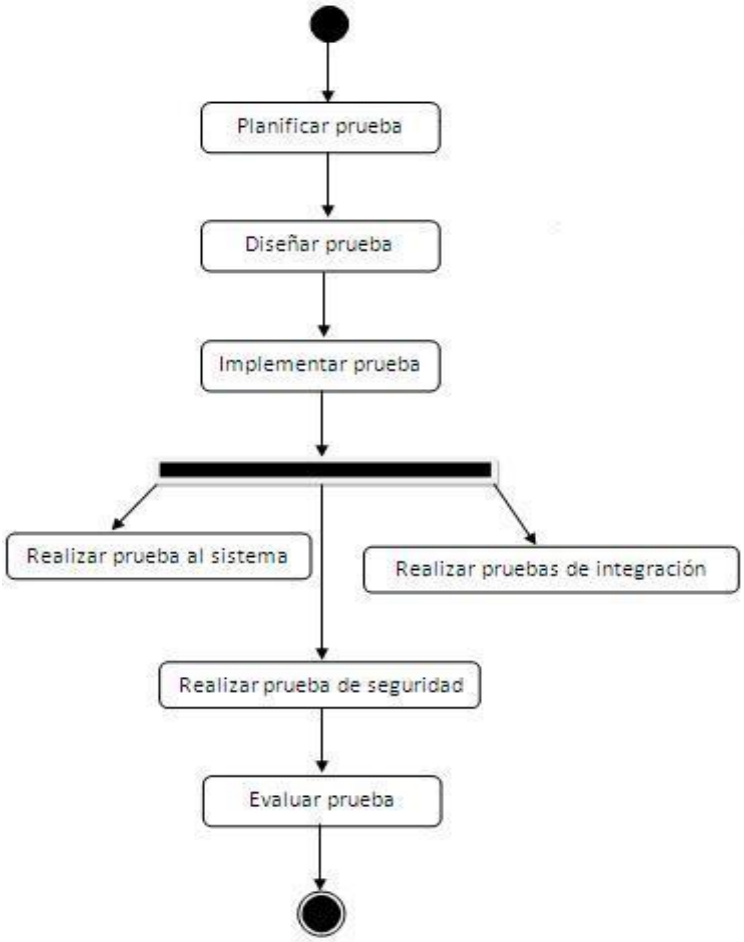

Ilustración 10 Actividades definidas para el proceso de desarrollo. Fuente: Elaboración propia.

2.7. Descripción de los artefactos

Como resultado del proceso, para esta estrategia, se lograrán diferentes artefactos, los cuales serán manifestados como lo establece la versión 2.0 del expediente de proyecto, actualmente vigente en la UCI y creado por el centro Calisoft. Seguidamente la descripción de cada uno de los artefactos generados:

2.7.1. Plan de pruebas:

Esta plantilla se realiza por el administrador de calidad y en su interior llevará:

- Nombre del proyecto.
- Nombre del producto.
- Versión actualizada del documento.

- Control de versiones: tabla que deberá actualizar con cada cambio que se efectúe en el documento.
- Introducción: dentro de este epígrafe se detallará el alcance del documento, las definiciones, acrónimos y abreviaturas que se emplean, además de las referencias hechas a otros documentos.
- Roles y responsabilidades: se describirán los roles existentes en el proyecto para la realización de las pruebas y las responsabilidades específicas de cada uno.
- Escenario de pruebas: contendrá el diagrama de despliegue y los recursos del sistema con los que cuenta el proyecto.
- Requerimientos a probar: llevará la lista de los requisitos o una referencia del documento “Especificación de requisitos”.
- Estrategia de pruebas de aceptación: donde se detalla el flujo de trabajo al que se le dará continuidad para la ejecución de las pruebas.
- Evaluación de las pruebas: se especificarán los criterios de evaluación para las pruebas, así como la clasificación de las no conformidades, pedidos de cambios y las listas de chequeo.
- Cronograma: cronograma de ejecución de las pruebas planificadas.

2.7.2. Caso de prueba

En la estrategia se alcanzarán dos casos de pruebas distintos, existirán para las pruebas de unidad y stress. La plantilla para estas pruebas en su contenido tendrá los siguientes aspectos:

- Nombre del proyecto
- Nombre del producto y versión actualizada del mismo.
- Nombre del caso de prueba coincidiendo con el requisito al que se está haciendo referencia y la versión actualizada del mismo.
- Control de versiones: tabla que se actualizará con cada cambio que se realice en el documento.

- Descripción general: descripción general del comportamiento del requisito referenciado.
- Condiciones de ejecución: precondiciones que cumplirá el requisito para ejecutarse.
- Secciones: tabla que tiene las secciones determinadas, de cada una se registrará el nombre, los escenarios que posee, descripción de la funcionalidad y el flujo central de eventos. Para cada sección, los escenarios van a ser flujo básico[5] y flujos alternos[6].
- Registro de defectos: se hace referencia a la plantilla “No conformidades” referente a este caso de prueba.

2.7.3. No conformidad

Artefacto confeccionado por el probador y que está compuesto con elementos siguientes:

- Nombre del proyecto.
- Nombre del producto.
- Versión actualizada del documento.
- Control de versiones: tabla que se actualizará con cada cambio que se realice en el documento.
- Descripción general: descripción de los aspectos generales a tener en cuenta al realizar el diseño de pruebas, incidentes en el instante de su desarrollo y otros aspectos importantes.
- Registro de defectos y dificultades detectadas: tabla que contendrá el elemento que se prueba, número de la no conformidad, descripción de la no conformidad de forma clara y concisa, descripción del aspecto correspondiente teniendo en cuenta el escenario donde se encuentra la deficiencia, etapa de detección,

[5] El flujo básico debe cubrir lo que sucede “normalmente” cuando se realiza el caso de uso.

[6] Los flujos alternativos de eventos cubren el comportamiento de una opción o excepción relativa al comportamiento normal, además de variaciones de este. Se puede pensar en los flujos alternativos como "desvíos" del flujo básico de eventos.

clasificación de la no conformidad, estado de la no conformidad y respuesta del equipo de desarrollo.

- Anexos: apoyo visual que se crea, necesario para hacer que no conformidad se pueda entender mejor.

2.7.4. Registro de prueba unitaria e integración.

En esta plantilla se registra los resultados obtenidos de las pruebas unitarias y de integración realizada, para ello almacenará los siguientes elementos:

- Nombre del proyecto.
- Nombre del producto.
- Versión actualizada del documento.
- Introducción: dentro de este acápite se describirá el alcance del documento, las definiciones, acrónimos y abreviaturas que se utilizan y las referencias hechas a otros documentos.
- Registro de prueba unitaria o integración: este acápite recoge los detalles de las pruebas, como el tipo de prueba ya sea de unidad o integración, nombre de la prueba, estado, tipo, última ejecución, ejecutado por, verificado por, descripción de la prueba, criterio de entrada, datos de aceptación y resultado.

2.7.5. Evaluación de la prueba:

Este documento es redactado después de haberse finalizado con cada tipo de pruebas a ejecutar, donde el probador comparará el resultado obtenido con el resultado esperado y llegarán a conclusiones el resto del equipo de pruebas conjunto con el probador. Sus elementos serán:

- Nombre del proyecto.
- Nombre del producto.
- Versión actualizada del documento.
- Control de versiones: tabla que se actualizará con cada cambio que se realice en el documento.

- Introducción: dentro de este acápite se describirá el alcance del documento y las referencias hechas a otros documentos.
- Recursos: se refleja el escenario de prueba en cuestión. Se listan, también, las herramientas y los artefactos que se utilizaron como apoyo.
- Resumen del estado del artefacto: en este acápite se reflejan las no conformidades encontradas en las pruebas en cuestión, especificando la cantidad de cada tipo. Además, se realiza un cronograma de los artefactos que se generarán para dicha prueba.

2.8. Propuesta de prueba.

2.8.1. Descripción de la aplicación de pruebas de unidad.

Se realizarán pruebas de unidad para cada uno de los módulos de la fábrica, de forma independientes, la misma se ejecutará durante la fase de Construcción y a la par del proceso de implementación, a medida que se realice se irán diseñando los casos de pruebas, en un estimado de quince días. Los probadores serán los encargados de ejecutar las pruebas, actualizar los diseños de caso de prueba y emitir las no conformidades encontradas, las cuales serán registradas en el documento “No conformidades” dejando así evidencia de las deficiencias que fueron encontradas y a su vez garantizar que cuando se haga una prueba exploratoria queden resueltos estos problemas.

Para esta prueba debería ejecutarse pruebas de caja blanca y caja negra. Debido al corto tiempo de vida de la Fábrica de Software, la estrategia solo contempla la aplicación de pruebas de caja negra y de caja blanca en las etapas de implementación del módulo.

Para ejecutar la prueba de caja negra, el primer paso es que después de terminada la implementación del módulo, el Diseñador de prueba elabora un caso de prueba para cada requisito funcional, definiendo los escenarios y las secciones a probar. En el caso de la prueba de caja blanca, el objetivo sería trabajar directamente con el código, garantizando que este sea lo más óptimo posible.

Luego de haberse realizado esta prueba ya el módulo estará listo para aplicarle la prueba de integración y quedará confeccionado el “Registro de las pruebas unitarias o integración” donde se recogerán los resultados obtenidos de forma general.

2.8.2. Descripción de la aplicación de pruebas de integración.

Al realizarse la prueba de integración las cuales son de tipo desarrollador, debido a que los programadores son los máximos responsables de verificar que el módulo en que están trabajando no tenga ningún tipo de problemas, para su posterior integración al subsistema. Las mismas se realizarán conjunto con las pruebas de unidad.

El objetivo principal de esta prueba es probar cómo los componentes funcionan en conjunto, los responsables de ejecutar este tipo de pruebas serán los Desarrolladores y el Arquitecto del proyecto. Comienzan en la fase de Construcción del producto, a medida que los desarrolladores vayan terminando los módulos que conforman los subsistemas.

2.8.3. Descripción de la prueba de Sistema.

Para esta prueba el probador se facultará de comprobar la respuesta del sistema ante una situación anormal que se pueda presentar, para la misma se utilizará la herramienta JMeter para aparentar estas situaciones, siempre teniendo un apoyo en la descripción del caso de prueba elaborado anteriormente por el diseñador de prueba.

A medida en que se efectúa la prueba, si se encuentran deficiencias, los resultados serán registrados en el documentos de “No conformidades” de igual forma que en las pruebas anteriores.

2.8.3.1. Descripción de pruebas de seguridad.

Mediante esta prueba el probador verificará el nivel de eficiencia del control de acceso de usuarios de la aplicación, es decir, se asegurará que al sistema accede el personal autorizado con los permisos adecuados para cada usuario.

Sí no existe la seguridad requerida, aparecerá una no conformidad, lo provocará una nueva ejecución de la prueba garantizando que el error fue corregido. La prueba solo será detenida si la comprobación del sistema pasa favorablemente la prueba.

Esta prueba puede realizarse también de forma automática, evaluándose la efectividad del encriptamiento de la contraseña del sistema, así como la seguridad de la base de datos, todo esto haciendo uso del Shadow Security Scanner, pero para la propuesta estas pruebas se realizarán de forma manual.

2.8.4. Descripción de la prueba de aceptación.

Es la prueba final antes del despliegue del sistema, con el objetivo de verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales fue concebido, además su posterior reutilización. El responsable de ejecutar la prueba serán el cliente y el Analista, el responsable de diseñar la prueba el Diseñador de pruebas. Para saber si el software cumple con los requisitos del cliente, pues se necesita su total aprobación, se debe confeccionar el Plan de Pruebas de Aceptación que contendrá los siguientes elementos:

- Nombre del proyecto.
- Nombre del producto.
- Versión actualizada del producto.
- Control de versiones: tabla que se actualizará con cada cambio que se realice en el documento.
- Introducción: dentro de este acápite se describirá el alcance del documento, los objetivos, una descripción del proyecto, el producto y la evolución del plan.
- Recursos: se resumirán los responsables de la aplicación de cada una de las pruebas así como los escenarios de pruebas y los recursos.
- Estrategias de pruebas de aceptación: en su contenido tendrá la representación del flujo de seguir, la descripción de las estrategias y tipos de pruebas y los datos de pruebas.
- Documentos generados durante la realización de las pruebas: se relacionan todos aquellos documentos generados durante la realización de las pruebas.
- Evaluación de las pruebas: se describen los criterios de evaluación para las pruebas, como clasificación de las no conformidades y pedidos de cambio.
- Cronograma de trabajo: cronograma de ejecución de las pruebas.
- Anexos.

2.9. Herramientas de pruebas.

Dentro de la propuesta quedarán definidas las herramientas de pruebas que se utilizarán en la Fábrica AplicativosSIG, entre estas herramientas se emplearán las de tipo manuales y las automatizadas.

2.9.1. Herramientas manuales.

Las herramientas manuales que se utilizarán serán los casos de prueba, debido a que tiene como ventajas:

- Son fáciles de definir, debido a que permite concretar el conjunto de pasos de cualquier abstracción, sin dejar de mencionar que propician delimitar las condiciones que se prefieran de superación o error de las pruebas.
- Pueden actuar como medio de comunicación en una fase temprana del proyecto.
- Es muy fácil empezar a escribir casos de pruebas manuales al principio del proyecto, antes de que se haya implementado el código, sólo con la obtención de los requisitos funcionales.

Casos de pruebas

Un caso de prueba es un acumulado de entrada, condiciones de ejecución y resultados deseados, desplegados para desempeñar un objetivo en particular o una función esperada. Los mismos deben expresar:

- Si el producto satisface las necesidades del cliente, tal y como se detalla en las descripciones de los requerimientos.
- Si el producto responde a lo que se desea, justo como se describe en la especificación funcional del diseño.

Se realizan tres iteraciones, durante la primera se obtienen las no conformidades que se clasifican en:

- Significativa: si inciden directamente en el proceso o cuando es problema de redacción, correspondencia con otra documentación, formato o error técnico.
- No significativa: cuando su impacto no tiene grandes repercusiones.

- Recomendación: es cuando se encuentra en el documento pero no con los elementos que debe llevar totalmente completos, por ello se le hacen sugerencias del revisor para mejorar la calidad del documento.

Posteriormente se le debe dar tratamiento a estas no conformidades obtenidas, para ello, en la segunda iteración su clasificación cambiaría en dependencia del tratamiento dado, esta vez se especificarán en:

- Resueltas: es cuando ya fueron resueltas estas no conformidades.
- No procede: se utiliza para especificar que el indicador a evaluar no se puede aplicar en ese caso.
- Pendiente de solución: como su nombre indica está en espera de solución por el equipo de trabajo.

2.9.2. Herramientas automatizadas.

Las herramientas automatizadas de pruebas guían hacia múltiples ventajas y desventajas, por ello cuando se trata de pruebas, es necesario conocerlas para así seleccionar los caminos más óptimos para la solución a problemas que se puedan presentar.

A partir del análisis realizado durante la investigación se determinó que se ejecutarán herramientas automatizadas a las pruebas que resulten muy complejas para su realización de forma manual. De la lista de herramientas analizadas se ha hecho énfasis total de las libres y de código abierto, pues las mismas representan un ahorro económico y amplias posibilidades de personalización a los intereses tecnológicos del centro. La herramienta elegida es: JMeter, teniendo en cuenta que de ella existe una documentación bastante abarcadora sobre su modo de empleo y los requerimientos no funcionales que ella requiere.

2.10. Conclusiones parciales.

En el desarrollo de este capítulo, quedó propuesta la estrategia de pruebas de software a seguir en la Fábrica de Software del departamento Geoinformática, por lo que se establece una guía que garantiza la organización del trabajo, ganando en tiempo y certificando que el producto final tendrá el mínimo de defectuosidad, favoreciendo el aseguramiento de la calidad del mismo.

CAPÍTULO 3 APLICACIÓN DE LA PROPUESTA

3.1. Introducción.

En este capítulo se confeccionará el proceso de aplicación de pruebas planificadas y organizadas en el Capítulo 2 al producto SIGRutas, mostrándose los artefactos generados y los resultados obtenidos. Luego se hace una valoración de cuán efectiva fue la aplicación de dicha estrategia.

3.2. Establecimiento del Plan de Pruebas.

El primer paso a ejecutar en la aplicación de esta propuesta, es establecer el Plan de Pruebas, debido a que será el artefacto que guiará todo el proceso de ejecución de las pruebas. En dicho documento quedan originados elementos como los que se muestran a continuación:

- Despliegue del sistema.

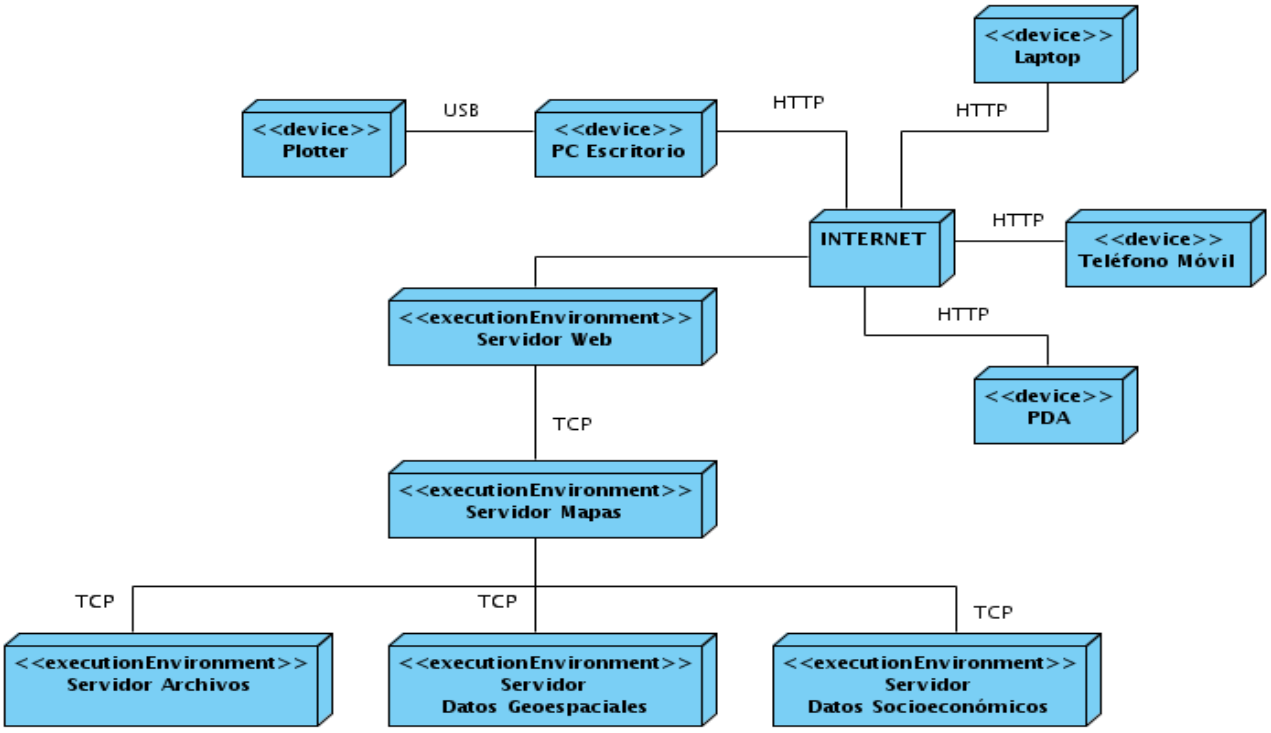

Ilustración 11 Diagrama de despliegue recogido en el "Plan de Pruebas". Fuente: Doc. Arquitectura de Software del producto SIGRutas.

- Roles y responsabilidades.

Rol	Cantidad	Responsabilidad
Administrador de la calidad	1	Asegurar la calidad en el ciclo de vida del software en la Factoría del departamento Geoinformática. Confección del documento Plan de pruebas, estrategias de pruebas, las listas de chequeo. Identificar y definir las pruebas necesarias a ejecutar. Recopila los datos de prueba en cada ciclo de prueba.
Diseñador de pruebas	1	Identificar las herramientas y técnicas necesarias para implementar las pruebas. Diseñar los casos de pruebas.
Probador	3	Probar los casos de uso.

Tabla 2 Roles y Responsabilidades. Fuente: Elaboración Propia.

- Recursos del sistema.

Ordenadores clientes	
Cantidad	2
Descripción	2.20Ghz, 0.97 GB de RAM, Intel (R) Core (TM) Duo CPU E4500@ 2.20Ghz.
Software base	Sistema Operativo Linux Versión 8.10, JMeter, Data Generator PostgresSQL.

Tabla 3 Recursos del sistema. Fuente: Elaboración Propia.

- Servidores

Recurso	Tipo
Servidor de Base de Datos (Postgre SQL v8.3).	Procesador Pentium IV 2x2 cache. 3 GHz como mínimo. Memoria 2GB de RAM. Almacenamiento en disco con capacidad igual o superior a los 10 GB (como mínimo). Tecnología de respaldo de datos históricos.
Sistemas de Información Geográfica (PostGIS)	Procesador Pentium IV ó superior, 3 GHz como mínimo. Memoria 2GB de RAM Disco duro de 5GB mínimo libre.
Servidor de aplicaciones Apache v2.5.6	Procesador Pentium IV 2x2 cache. 3 GHz como mínimo Memoria 2GB de RAM.

Tabla 4 Servidores. Fuente: Elaboración Propia.

- Requisitos a probar

RF1.Realizar Zoom int al mapa.

RF2.Centrar un área del mapa.

RF3.Realizar paneo sobre el mapa.

RF4.Medir Distancia del punto de partida al punto de llegada.

RF5.Localizar parada más cercana.

RF6.Localizar ruta por dirección.

RF7.Realizar Zoom out sobre el mapa.

RF8.Ver todo el mapa.

- Estrategia de pruebas de aceptación

Para la posterior ejecución de las pruebas de aceptación se establecerán inicialmente, pruebas unitarias para comprobar que funcionan por separado cada uno de los módulos del proyecto. Seguidamente se realizarán pruebas de integración para demostrar el funcionamiento de los módulos en conjunto. Luego se le aplicarán pruebas de sistema. Dentro de estas se seleccionaron pruebas de stress y seguridad, las cuales se efectuarán de forma manual y las de un mayor nivel de complejidad de forma automatizada con la utilización de la herramienta JMeter. Finalizada las pruebas anteriormente mencionadas, sólo resta la implementación de las pruebas de aceptación donde el propio cliente dará la aprobación del producto.

3.3. Aplicación de las pruebas

3.3.1. Diseños de casos de prueba

Definido el Plan de Prueba, se llevan a cabo a las pruebas unitarias. Inicialmente se realizaron los diseños de casos de pruebas, con la plantilla antes descrita en el Capítulo 2, para los casos de uso más críticos, teniendo un total de trece diseños de casos de pruebas.

A continuación se muestran algunos aspectos que se muestran en el DCP_Localizar Parada.

- Descripción general

El caso de uso se inicia cuando el usuario selecciona la localización de paradas, para ello el usuario debe ingresar la parada que desea localizar, en dependencia del tipo de ruta, y termina cuando realiza la operación de localizar la parada.

- Condiciones de ejecución

El usuario debe estar autenticado.

- Secciones

Nombre de la sección	**Escenarios de la sección**	**Descripción de la funcionalidad**
SC 1: Localizar Parada	EC 1.1: Seleccionar la opción Localizar Parada.	El sistema muestra la lista de municipios en una pestaña en el panel izquierdo.
	EC 1.2: Desplegar Municipios.	El sistema muestra la lista de municipios.
	EC 1.3: Contraer Municipios.	El sistema contrae la lista desplegada de municipios.
	EC 1.4: Desplegar Paradas de un Municipio.	El sistema muestra el listado de las paradas dentro del municipio.
	EC 1.5: Contraer Paradas.	El sistema contrae la lista desplegada de las paradas.
	EC 1.6: No se selecciona la parada a localizar.	El sistema muestra los botones de “Localizar” y “Datos de parada” desactivados.
	EC 1.7: Seleccionar parada.	El sistema activa los botones “Localizar” y “Datos de parada”.
	EC 1.8: Localizar parada mediante la opción “Localizar”.	El sistema coloca una tachuela sobre el mapa señalando la parada seleccionada.
	EC 1.9: Mostrar datos de la parada a través del botón “Datos de la parada”.	El sistema muestra una ventana que contiene la dirección de la parada y las rutas que hacen parada en ese lugar en listas desplegables.
	EC 1.10: Mostrar datos de la parada mediante el posicionamiento del cursor sobre la tachuela en el mapa.	El sistema muestra una ventana que contiene la dirección de la parada y las rutas que hacen parada en ese lugar.

	EC 1.11: Desplegar dirección de la parada.	El sistema despliega la dirección de la parada y las rutas que hacen parada en ese lugar.
	EC 1.12: Contraer dirección de la parada.	El sistema contrae la dirección de la parada.
	EC 1.13: Desplegar rutas que hacen parada en ese lugar.	El sistema despliega las rutas que hacen parada en ese lugar.
	EC 1.14: Contraer rutas que hacen parada en ese lugar.	El sistema contrae las rutas que hacen parada en ese lugar.
	EC 1.15: Localizar parada mediante doble clic.	El sistema coloca una tachuela sobre el mapa señalando la parada seleccionada.
	EC 1.16: Cerrar información de parada.	El sistema cierra la información de la parada.

Tabla 5 Secciones a probar en el DCP_Localizar Parada. Fuente: Elaboración Propia

- Descripción de variables.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1.	Municipio	Lista desplegable	No	Se tiene que seleccionar la opción Municipio.
2.	Parada	Lista desplegable	No	Se tiene que seleccionar la opción Parada.

- SC1: Localizar Parada.

Escenario	Variable 1 Municipio	Variable 2 Parada	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
EC 1.1: Seleccionar la opción Localizar Parada	NA	NA	El sistema muestra la lista de municipios en una pestaña en el panel izquierdo.		• Seleccionar la opción Localización. • Seleccionar la opción Localizar Parada.

Escenario	Variable 1 Municipio	Variable 2 Parada	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
EC 1.2: Desplegar Municipios.	NA	NA	El sistema muestra la lista de municipios.		• Seleccionar la opción Localización. • Seleccionar la opción Localizar Parada. • Seleccionar la opción de desplegar los municipios.
EC 1.3: Contraer Municipios.	NA	NA	El sistema contrae la lista desplegada de municipios.		• Seleccionar la opción Localización. • Seleccionar la opción Localizar Parada. • Seleccionar la opción de contraer los municipios.
EC 1.4: Desplegar Paradas de un Municipio.	NA	NA	El sistema muestra el listado de las paradas dentro del municipio		• Seleccionar la opción Localización. • Seleccionar la opción Localizar Parada. • Seleccionar la opción de desplegar los municipios. • Seleccionar la opción de desplegar las paradas.
EC 1.5: Contraer Paradas.	NA	NA	El sistema contrae la lista desplegada de las paradas.		• Seleccionar la opción Localización. • Seleccionar la opción Localizar Parada. • Seleccionar la opción de desplegar los municipios. • Seleccionar la opción de contraer las paradas.

Escenario	Variable 1 Municipio	Variable 2 Parada	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
EC 1.6: No se selecciona la parada a localizar.	V (Artemisa)	I ()	El sistema muestra los botones de “Localizar” y “Datos de parada” desactivados.		• Seleccionar la opción Localización. • Seleccionar la opción Localizar Parada. • Seleccionar la opción de desplegar los municipios. • Seleccionar la opción de desplegar las paradas. • Seleccionar un municipio.
EC 1.7: Seleccionar parada.	V (Artemisa)	V (Hospital Artemisa)	El sistema activa los botones “Localizar” y “Datos de parada”.		• Seleccionar la opción Localización. • Seleccionar la opción Localizar Parada. • Seleccionar la opción de desplegar los municipios. • Seleccionar la opción de desplegar las paradas. • Seleccionar un municipio. • Seleccionar la parada a localizar.
EC 1.8: Localizar parada mediante la opción “Localizar”.	V (Artemisa)	V (Hospital Artemisa)	El sistema coloca una tachuela sobre el mapa señalando la parada seleccionada.		• Seleccionar la opción Localización. • Seleccionar la opción Localizar Parada. • Seleccionar la opción de desplegar los municipios. • Seleccionar la opción de desplegar las paradas. • Dar clic en el botón “Localizar”.

Escenario	Variable 1 Municipio	Variable 2 Parada	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
EC 1.9: Mostrar datos de la parada a través del botón “Datos de la parada”.	V (Artemisa)	V (Hospital Artemisa)	El sistema muestra una ventana que contiene la dirección de la parada y las rutas que hacen parada en ese lugar en listas desplegables: Dirección y Las siguientes rutas hacen parada en este lugar.		• Seleccionar la opción Localización. • Seleccionar la opción Localizar Parada. • Seleccionar la opción de desplegar los municipios. • Seleccionar la opción de desplegar las paradas. • Dar clic en el botón “Localizar”. • Dar clic en el botón “Datos de la parada”.
EC 1.10: Mostrar datos de la parada mediante el posicionamiento del cursor sobre la tachuela en el mapa.	V (Artemisa)	V (Hospital Artemisa)	El sistema muestra una ventana que contiene la dirección de la parada y las rutas que hacen parada en ese lugar.		• Seleccionar la opción Localización. • Seleccionar la opción Localizar Parada. • Seleccionar la opción de desplegar los municipios. • Seleccionar la opción de desplegar las paradas. • Dar clic en el botón “Localizar”. • Posicionar el cursor sobre la tachuela en el mapa.

Escenario	Variable 1 Municipio	Variable 2 Parada	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
EC 1.11: Desplegar dirección de la parada.	NA	NA	El sistema despliega la dirección de la parada y las rutas que hacen parada en ese lugar.		• Seleccionar la opción Localización. • Seleccionar la opción Localizar Parada. • Seleccionar la opción de desplegar los municipios. • Seleccionar la opción de desplegar las paradas. • Dar clic en el botón “Datos de la parada”. • Seleccionar la opción de desplegar la dirección de la parada.
EC 1.12: Contraer dirección de la parada.	NA	NA	El sistema contrae la dirección de la parada.		• Seleccionar la opción Localización. • Seleccionar la opción Localizar Parada. • Seleccionar la opción de desplegar los municipios. • Seleccionar la opción de desplegar las paradas. • Dar clic en el botón “Datos de la parada”. • Seleccionar la opción de contraer la dirección de la parada.

Escenario	Variable 1 Municipio	Variable 2 Parada	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
EC 1.13: Desplegar rutas que hacen parada en ese lugar.	NA	NA	El sistema despliega las rutas que hacen parada en ese lugar.		• Seleccionar la opción Localización. • Seleccionar la opción Localizar Parada. • Seleccionar la opción de desplegar los municipios. • Seleccionar la opción de desplegar las paradas. • Dar clic en el botón “Datos de la parada”. • Seleccionar la opción de desplegar la dirección de la parada.
EC 1.14: Contraer rutas que hacen parada en ese lugar.	NA	NA	El sistema contrae las rutas que hacen parada en ese lugar.		• Seleccionar la opción Localización. • Seleccionar la opción Localizar Parada. • Seleccionar la opción de desplegar los municipios. • Seleccionar la opción de desplegar las paradas. • Dar clic en el botón “Datos de la parada”. • Seleccionar la opción de contraer la dirección de la parada.
EC 1.15: Localizar parada mediante doble clic.	V (Artemisa)	V (Hospital Artemisa)	El sistema coloca una tachuela sobre el mapa señalando la parada seleccionada.		• Seleccionar la opción Localización. • Seleccionar la opción Localizar Parada. • Seleccionar la opción de desplegar los municipios. • Seleccionar la opción de desplegar las paradas. • Dar doble clic en el botón sobre la parada.

Escenario	Variable 1 Municipio	Variable 2 Parada	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
EC 1.16: Cerrar la información de la parada.	NA	NA	El sistema cierra la información de la parada.		• Seleccionar la opción Localización. • Seleccionar la opción Localizar Parada. • Seleccionar la opción de desplegar los municipios. • Seleccionar la opción de desplegar las paradas. • Localizar la parada. • Seleccionar la opción “X.

Tabla 6 Escenarios de la sección Localizar Parada. Fuente: Elaboración Propia.

Al aplicarse estas pruebas se detectan diferentes deficiencias en el sistema para cada uno de los casos de pruebas, de ahí quedaría confeccionado el documento con las no conformidades detectadas en cada una de las iteraciones.

En general para el SIG tomado como muestra se obtuvo un total de 17 no conformidades en la prueba exploratoria, 10 en la primera iteración y 5 en la segunda iteración. Los resultados de las pruebas unitarias quedan como se muestra en el gráfico a continuación:

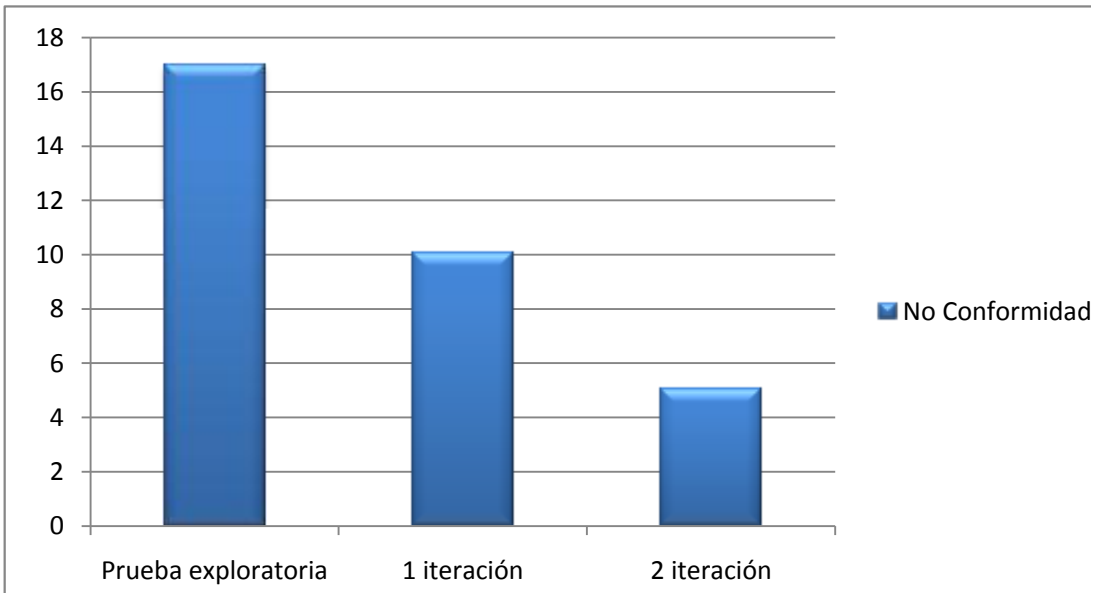

Ilustración 12 Resultados de las pruebas unitarias para SIGRutas. Fuente: Elaboración Propia.

3.4. Pruebas de sistema.

Dentro de las pruebas de sistema se aplican otros tres tipos de pruebas, que a continuación se mencionan pruebas de carga, stress y seguridad. Para cada unas de ellas se procede de forma diferente.

3.4.1. Pruebas de carga y stress.

Al aplicarse las pruebas de carga para un total de 45 usuarios solicitando reportes de Mover Mapa la aplicación generó un total de 4545,7 kb/seg de transferencia de datos lo que incurrió en un rendimiento de 352,7 seg. Se demuestra que la aplicación es estable ya que para todos estos usuarios, se mantuvo prestando servicios todo el tiempo sin incurrir en fallos para un de 0,78% de error. Para una mayor especificación citar Anexos 3.

Después de estresada la aplicación inicialmente para 100 usuarios estableciendo las mismas peticiones mencionadas anteriormente se obtiene que la aplicación generó 104kb/seg de transferencia de datos lo que incurrió en un rendimiento de 19,8 seg. Aquí se demuestra que aún la aplicación sigue mostrando sus servicios sin problemas para un 22,22% de error. Seguidamente con 150 usuarios no se obtienen los mismos resultados pues estos son más desfavorables, ya que la transferencia de datos es de 30,7kb/seg y su rendimiento es 6,6 seg, lo que demuestra que la aplicación no es estable y se demora mucho en responder las peticiones para un 100% de error.

3.4.2. Pruebas de seguridad.

Luego de establecidas estas pruebas se obtiene como resultado que el sistema cuenta con la correcta autorización por cada uno de los usuarios que acceden a la aplicación, es decir, los permisos y privilegios fueron dados según su correspondencia. Con el resultado alcanzado se concluye que la prueba no desempeñó su objetivo fundamental, el de detectar deficiencias en el sistema, pero sí se demuestra el exitoso funcionamiento de la aplicación.

3.5. Pruebas de aceptación.

Las pruebas de aceptación son pruebas directamente ejecutadas con el cliente, por lo que teniéndose un plan de pruebas de aceptación, aún no se puede proceder a establecer las pruebas ya que el producto debe estar totalmente terminado, y hechas cada una de las pruebas anteriores. Como SIGRutas no se encuentra aún culminado,

para el trabajo de diploma en cuestión, no se cuenta con los resultados obtenidos de las pruebas de aceptación, pero la constitución de su plan se lleva a continuación.

Al llevarse a cabo este tipo de prueba es necesario guiarse por un plan, por lo que queda diseñado de la siguiente forma:

- Introducción

Este documento se elabora fundamentalmente para delimitar el plan de pruebas de aceptación del producto SIGRutas. En el documento descripción de los casos de uso del sistema, quedan descritos los casos de usos, que serán sometidos a proceso de aceptación para los módulos que se mencionan a continuación:

- Navegación
- Localización
- Opciones
- Administración

Seguidamente como se muestra en la tabla, los artefactos que servirán de apoyo a las pruebas.

Artefactos a probar.	**Documentos precisos.**
Aplicación Web	- Especificación de requisitos. - Especificación de casos de uso del sistema. - Especificación de los requisitos no funcionales. - Casos de uso del negocio. - Casos de prueba.
Manual del usuario	- Aplicación.
Manual de instalación	- Instalador de la aplicación.
Glosario de términos	-
Prototipos no funcionales	- Especificación de casos de uso del sistema. - Especificación de requisitos.
Ayuda	- Manual de usuario. - Aplicación.

Tabla 7 Artefactos de apoyo a las pruebas de aceptación. Fuente: Elaboración Propia.

- Objetivos

Como objetivo fundamental se tiene que mediante este plan se lograrán determinar los recursos necesarios para llevar a cabo las pruebas de aceptación, además se delimitarán los elementos de pruebas y sus prioridades.

- Alcance

El alcance estará dado en la revisión de las funcionalidades de cada caso de uso del sistema, por lo que dichas pruebas se ejecutarán sólo después de haberse realizado las pruebas de unidad, integración y sistema.

- Identificación del proyecto

Aplicativos de Sistema de Información Geográfica cuenta con ocho productos, uno esos productos SIGRutas, el cual se encuentra en su primera versión. SIGRutas cuenta con cuatro módulos y ocho casos de uso, estos últimos se localizan en los documentos especificación de requisitos de SIGRutas.

- Estrategia de evolución del plan

El plan se estará verificando constantemente, siempre al finalizar cada paso de trabajo se hará un seguimiento de sí se está llevando a cabo tal y como se planificó, además si es necesario realizar algún cambio en el cronograma de las pruebas, el cual será informado de inmediato al equipo de pruebas.

- Aprobación de los involucrados

Los principales involucrados serán como se muestran en la siguiente tabla, el jefe de proyecto de AplicativosSIG y la asesora de calidad del centro de desarrollo GEySED, ellos y otros del equipo de desarrollo serán los implicados en las pruebas de aceptación.

Nombre y Apellidos	**Responsabilidad**
Adrián Gracia	Jefe de Proyecto.
Yesleny Becerra	Asesora de calidad del centro GEySED.

Tabla 8 Involucrados en las pruebas de aceptación. Fuente: Elaboración Propia.

- Roles y responsabilidades

A continuación se exponen los roles, total de personal y responsabilidades implicados en las pruebas.

Rol	**Cantidad**	**Responsabilidad**
Líder de Factoría	1	Examinar el cumplimiento del “Plan de pruebas de aceptación”.
Especialistas de calidad	2	Inspeccionar la ejecución de las pruebas, llevando un control de

		las no conformidades detectadas. Diseñar las pruebas de software, valorar dicho proceso y sus resultados. Registrar, monitorear y establecer el "Plan de pruebas de aceptación".
Expertos funcionales (usuarios finales)		Efectuar las pruebas al producto, comprobando y aprobando las funcionalidades de la aplicación. Confeccionar en conjunto con los especialistas de calidad el informe de las no conformidades encontradas y si es necesario algún pedido de cambio.
Equipo de trabajo SIGRutas		Guiar al equipo de pruebas de cómo interactuar con la aplicación. Registrar las no conformidades y los pedidos de cambio, si existen, para finalmente confeccionar el "Acta de aceptación" del producto.

Tabla 9 Roles y Responsabilidades para pruebas de aceptación. Fuente: Elaboración Propia.

- Estrategia de las pruebas de aceptación

Primeramente se orientará una reunión de inicio en donde se le hará la transferencia al cliente de todos los entregables y descripciones precisas para la ejecución de las pruebas. El cliente tendrá un período de tiempo para establecer la revisión y a su vez hacer la entrega de las no conformidades. Seguidamente el equipo de desarrollo deberá dar solución a estas no conformidades y las respuestas pertinentes. Luego solo queda la aceptación del producto, la cual se hará en una reunión minuta para tener la aprobación del software.

- Descripción de las estrategias y tipos de pruebas.

Muchas de las pruebas se harán de forma manual, solo serán automatizadas las pruebas de carga y stress, se examinará las funcionalidades descritas y desarrolladas según la metodología utilizada por la Factoría. Las mismas estarán organizadas por fases de acuerdo a la etapa perteneciente.

Fase uno: formación del escenario de pruebas, además del entrenamiento del equipo de pruebas.

Fase dos: ejecución de las pruebas de funcionalidad (caja negra).

Fase tres: aplicación de las pruebas de seguridad con el objetivo de verificar que los usuarios tienen los accesos permitidos de acuerdo a sus privilegios, además que no haya entrada de intrusos.

Fase cuatro: conformidad con los documentos entregables, aprobación del producto y las firmas correspondientes por su aceptación.

3.6. Evaluación de los resultados de las pruebas

Después de aplicadas cada una de las pruebas propuestas se llega a una evaluación de los resultados obtenidos, donde se recopilan en general las dificultades del sistema y si las pruebas fueron fallidas o no.

Al ejecutarse las pruebas unitarias se obtuvo un documento denominado “Evaluación de pruebas unitarias”, en el que quedan plasmadas todos los problemas hallados, el mismo puede ser analizado en el Anexo 2 del siguiente trabajo.

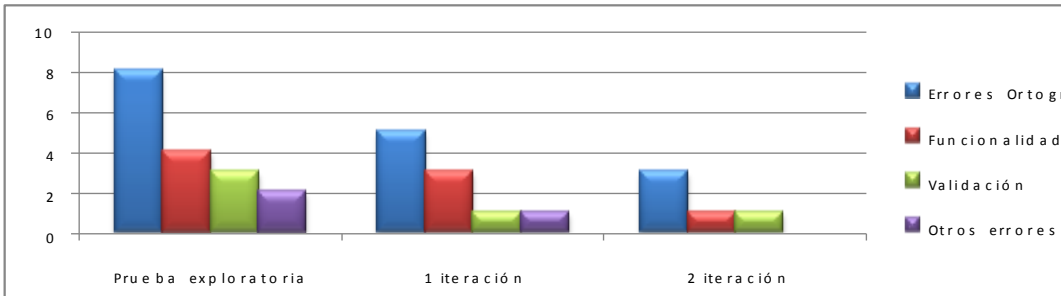

Ilustración 13 Evaluación de las pruebas unitarias. Fuente: Elaboración Propia.

De acuerdo a la gráfica mostrada anteriormente se puede concluir que luego de realizadas las pruebas unitarias las principales dificultades encontradas son de errores ortográficos, lo que representa que al terminar cada iteración solo fueron resueltas algunas de ellas y no todas. Durante la prueba exploratoria fueron encontradas 17 no conformidades, diez significativa lo que representa en 59%, seis recomendaciones representando un 35% y las no significativas el resto con un 6%, clasificadas como se observa en la siguiente figura.

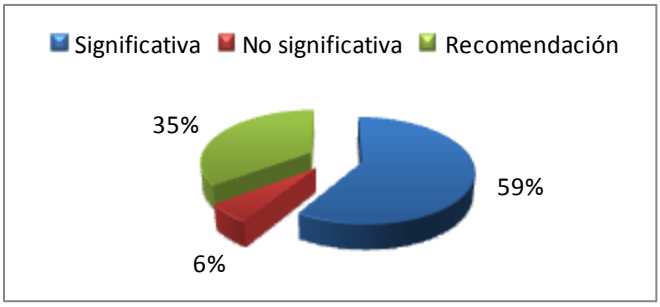

Ilustración 14 Resultados de la prueba exploratoria. Fuente: Elaboración Propia.

Al terminar la prueba exploratoria y ejecutada la primera iteración quedan los resultados y las clasificaciones. En la siguiente imagen se muestra claridad que el 70%

de las No conformidades quedaron resueltas para un total de siete, 10% Procede y el 20% Pendiente de solución.

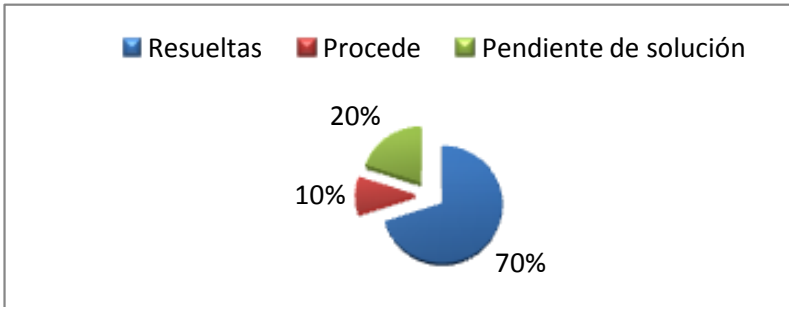

Ilustración 15 Resultados de la primera iteración. Fuente: Elaboración Propia.

Seguidamente se realiza la segunda iteración para tener los siguientes resultados, 60% de resueltas y el 40% Pendientes de solución por el grupo de trabajo de SIGRutas.

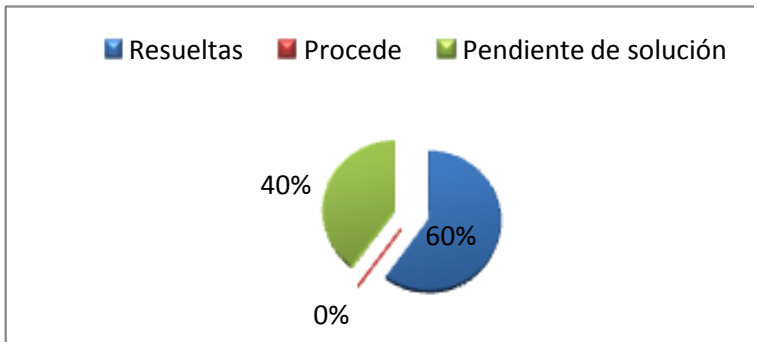

Ilustración 16 Resultados de la segunda iteración. Fuente: Elaboración Propia.

Actualmente el equipo de trabajo se encuentra inmiscuido en la solución de las No Conformidades que quedan Pendiente de Solución, para posteriormente hacer la entrega de los módulos probados al centro de calidad para soluciones tecnológicas de la universidad (Calisoft).

3.7. Conclusiones parciales.

Los resultados arrojados son una evidencia de que la estrategia de prueba propuesta y aplicada, es totalmente ajustada a la Fábrica de Software del departamento Geoinformática, la misma puede ser aplicada a otros productos de la fábrica. Hasta este período solo se han obtenido resultados parciales, pues se necesita en su totalidad aplicarse a todos los productos que están dentro de la fábrica. Obtenidos los resultados de las pruebas se concluye con que la aplicación cuenta con 100% de seguridad pues así lo muestran sus resultados, en cuanto a las pruebas de carga stress, quedó evidenciado como el sistema responde a las peticiones hechas por los

usuarios en el caso de la prueba de carga para 45 usuarios con un 0,78% de error y en las de stress se verificó que con 150 usuarios la aplicación se demora en responder, por lo que su error es de un 100%, lo que demuestra que la misma no está lista para esta cantidad de peticiones. En las pruebas unitarias las no conformidades más vistas fueron las de errores ortográficos, y que son clasificadas de significativas por lo que sí inciden directamente en el proceso. Por último y no menos importante las pruebas de aceptación, de estas no se obtuvieron resultados, puesto que el producto no se encuentra terminado, pero aún así quedó plasmado el plan de pruebas de aceptación.

CONCLUSIONES GENERALES

- Los objetivos planteados para este trabajo de diploma quedaron plenamente cumplidos a través de las actividades efectuadas. La estrategia propuesta aún está siendo implantada en la Fábrica con resultados satisfactorios, pues ha reducido el número de no conformidades del producto SIGRutas y de otros que luego pasarán al proceso de liberación como es el caso de SIGSalud.
- Se adquirieron conocimientos importantes sobre las Factorías de Software aplicables al departamento con el objetivo de organizar y elevar la producción de software.
- Probar parte de la estrategia diseñada para la Factoría sirvió de gran utilidad para la organización interna del trabajo.
- La estrategia definida proporcionó la planificación, verificación y validación del proceso de desarrollo del software, obteniéndose un mayor número y variedad de deficiencias con el mínimo consumo de tiempo y esfuerzo, además de lograr cumplir con las expectativas del cliente.
- Los resultados arrojados quedaron registrados evaluando la importancia de la estrategia que se propone en el trabajo de diploma, ya que ayudará a elevar la eficiencia y calidad del ciclo de vida de los productos de AplicativosSIG.
- Cómo parte de las pruebas realizadas se puede concluir que la aplicación es 100% segura, que no es posible tener una conexión de 150 usuarios debido a que esta no se encuentra totalmente en disposición de responder a esa cantidad de peticiones, y después de resueltas las no conformidades detectadas en la segunda iteración, el producto estará listo para pasar a las pruebas de liberación.

RECOMENDACIONES

Luego de aplicada la estrategia se recomienda:

- Aplicar la estrategia expuesta en este documento desde sus inicios a todos productos que se desarrollan en la Factoría, pues la misma fue creada de manera flexible a cambios, lo que permite que pueda ser adaptada a cualquier producto de una Fábrica.
- Estudiar y profundizar otras pruebas que se puedan aplicar a los productos de la Fábrica.
- Capacitar a todo el personal que labora en el Grupo de Calidad con el fin de aumentar la confiabilidad de las pruebas realizadas.

BIBLIOGRAFÍA

1. (13) Aaen, Ivan, p. b., Lars Mathiassen. The Software Factory: Contributions and Illusions: Proceedings of the Twentieth Information System Research Seminar. Scandinavia, Oslo: s.n., 1997.
2. (18) Almenares, Liudmila Sánchez. Como realizar pruebas de carga y stress en JMeter. Ciudad de la Habana: s.n., 2008.
3. (16) Basili, V. R. C., G. y Cantone. Reference Architecture for the Component Factory ACM Transaction on Software Engineering and Methodology. 1992.
4. (22) Carrillo, Luis Vinicio León. El proceso de la prueba de software. 2005.
5. Calisoft, Centro de Calidad para Soluciones Informáticas. [En línea] Universidad de las Ciencias Informáticas. http://calisoft.uci.cu.
6. (14) Díaz Olavarrieta, Arnoldo, a. b. p. [En línea] “Dime qué tienes, qué produces y te diré qué eres". http://www.certum.com/Publicaciones/FabSoft.pdf.
7. (1)Feigenbam, Edward Albert. 3ecubo. [En línea] http://www.3ecubo.es/calidad.html.
8. (15) Fernstrom, C. N., K. H y Ohlsson, l. Software Factory Principles, Architecture and Experiments. s.l.: No. 2, 1992.
9. (21) Gurices, Los. Metodología de Desarrollo de Software basada en Componentes. 2010.
10. (9) H, Monsalve J. Sistemas de Información III. [En línea] http://prof.usb.ve/lmendoza/Documentos/PS-6117%20%28Teor%EDa%29/PS6117%20Calidad%20del%20Software.pdf.
11. (11) Jacobson Ivar, Booch Grady, Rumbaugh James. El Proceso Unificado de Desarrollo de Software Madrid: Addison Wesley: s.n., 2000.
12. (2) Jurán, Joseph M. Juran en el liderazgo para la calidad. 1989.
13. (7) Lovelle Cueva, Juan Manuel. Calidad del Software. España: s.n., 1999.

14. (8) Mendoza, Luis Eduardo. [En línea] http://prof.usb.ve/lmendoza/Documentos/PS-6117%20%28Teor%EDa%29/PS6117%20Calidad%20del%20Software.pdf.

15. (10) Myers, Glen. The Art of Software Testing 1979.

16. (23) Natalia, Juristo, Moreno, Ana y Vegas, Evaluación de Software. Sira. 2006.

17. (4) Normas iso 9000 y calidad. [En línea] http://normas-iso-9000.blogspot.com/2007/11/cmo-se-define-la-calidad.html.

18. (12) OCENET de Administración de Empresas. [En línea] http://ocenet.oceano.com.

19. (6) Pressman, Roger S. Ingeniería de Software. Un Enfoque Práctico. s.l.: Mc Graw-Hill, 1998.

20. (3) Real Academia Española. [En línea] http://buscon.rae.es/draeI/.

21. (17) Ríos, Yanosky, M. M. Modelo funcional de la Factoría de Software de la UCI para la línea Carrefour. Ciudad Habana, Cuba, Universidad de las Ciencias Informáticas: s.n., 2005.

22. (5) Rojas Salamanca, Saulo Ernesto, Borja Parra, Juan José. Calidad del Software: Camino hacia una verdadera Industria de Software. 38, 1999.

23. (19) Vallecillo, Antonio e Iribarne, Luis. Elaboración de aplicaciones software a partir de componentes COTS.

24. Vallecillo, Antonio, Bertoa Manuel. Aspectos de Calidad en el Desarrollo de Software Basado en Componentes. 2002.

25. (20) Wallance, D. R., R. U. Fuji. Software Verification and Validation: An Overview, IEEE Software. 1989.