Diseño de una estrategia de pruebas para la plataforma GENESIG y sus aplicativos.

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

Autora: Liliam Rodríguez Landestoy

Tutora: Ing. Zaily Rodríguez Luis

La Habana, 28 de junio del 2011.

DEDICATORIA

A mis padres Raquel y Jose. Gracias por existir. Ustedes conforman el regalo más precioso que me ha dado la vida. Les dedico todo el esfuerzo y la entrega de estos cinco años en el transcurso de toda mi carrera, en la cual ustedes han sido los únicos protagonistas.

A mi hermano José Javier. Tú has sido para mí la bendición más pura y linda que he recibido desde hace 15 años. Te dedico este resultado para que sea tu guía y tu motor impulsor, porque quiero que en un futuro cuelgues tu título junto al mío.

A la memoria de mi abuela Gardenia. Dondequiera que estés eres mi fuente de inspiración. A ti te dedico el triunfo de mis estudios, porque sé que estarías muy orgullosa de mí.

AGRADECIMIENTOS

Agradezco a mi madre por ser la persona más maravillosa del mundo y por haber sido excepcional conmigo durante toda mi vida, por brindarme su amor y su apoyo incondicional desde el momento en que abrí los ojos hasta hoy, gracias a ella por su fuerza y su perseverancia que han logrado convertirme en ingeniera.

A mi padre por entregarme más que su vida, por haber estado ahí siempre que lo necesité, por decirme siempre que sí, por su dedicación absoluta y su amor transparente.

A mi hermano por haberme dado su mano todos estos años, gracias por confiar en mi, por brindarme su cariño puro y su apoyo sagrado.

Gracias a toda mi familia que es enorme en especial a:

- Mis tías Xiomara y Glicer por haber sido como mis segundas madres, gracias por estar a mi lado toda mi vida, por permitirme compartir junto a ellas tantos momentos hermosos que nunca olvidaré.
- Mi tío Norberto, que a pesar de la distancia que nos separa lo he sentido muy cerca de mi todo este tiempo, gracias por sus consejos y por brindarme su ayuda espiritual y materialmente.
- A mis tíos políticos Maday y Carlitos quienes me han ayudado enormemente y me han demostrado su cariño sincero. Gracias por poder contar con ellos siempre.
- Mis primas Yudit y Yisel, que han sido las hermanas hembras que nunca tuve, mis dos primas, mis dos compañeras y mis dos amigas con las cuales he compartido mis mayores alegrías.
- Mis primos Yeny y Yeniel quienes me han apoyado increíblemente en todos los pasos que he dado. Gracias por sus correos, por sus consejos y por su disposición en todo momento de ayudarme.
- Mis tres primitos: la princesita Anna, la bella Grecia y mi macho Manue, por darme tanta alegría y felicidad, gracias por haber llegado a la familia y por haber logrado que desprenda tantas risas debido a sus ocurrencias.

A mi novio por ayudarme en cada paso que doy, por mantener esta hermosa relación que ha despertado en mí tantos sentimientos lindos y puros, logrando juntos tantas cosas maravillosas, gracias por su amor incondicional, su respeto, su comprensión y su dedicación. Sin él no hubiera cumplido mis sueños, fue mi guía y mi brazo derecho para cumplir esta meta tan importante en mi vida.

A mis suegros por sus consejos y su ayuda que lograron convertirme en una mejor persona.

Agradecimientos

A mi tutora por ayudarme en el desarrollo de este trabajo de diploma, a los miembros del tribunal por ss señalamientos contructivos que posibilitaron darle calidad a la investigación realizada y a mi co-tuto: el profesor Daniel por brindarme todo su apoyo desinteresadamente cuando más lo necesité.

A mis amigas de toda la vida Daynelis y Liset con quienes he compartido inseguridades, alegrías, disgustos y momentos inolvidables que fortalecen cada vez más nuestra amistad.

A mi gran amigo Mongui quien conocí hace muy poco tiempo y tal parece que lo conociera de toda la vida, gracias por haberme dado fuerzas, ánimos y el ímpetu necesario para vencer el último tiempo y el más difícil en la universidad.

A mis amigos de universidad que nunca olvidaré ya que son una parte muy importante en mi vida. Gracias a Milayne, Noraima, Yanet, Yordanka, Lisandra, Daniel, Jorge y Reynel porque cada uno de ellos colocó una huella diferente en mi corazón y me ayudaron a valorar las amistades y a apreciar verdaderamente a las personas.

A todos los profesores que me formaron como profesional, en especial a Osmel y a Pimienta que me brindaron su apoyo como profesor y amigo a la vez.

A todos mis vecinos en especial: Ramona, Belkis, Yanelis, Emelia, Martha, quienes forman mi otra familia. Gracias por ayudarme a lo largo de estos 5 años con sus atenciones y apoyo incondicional.

A mis costureras, gracias por su disposición, su paciencia y solidaridad en todo momento.

A todas las personas que he conocido en esta etapa tan importante de mi vida, porque cada una de ellas me enseñó algo especial y me brindó apoyo cuando lo necesité.

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

“Liliam Rodríguez Landestoy” **“Zaily Rodríguez Luis”**

_____ _____

RESUMEN

Un Sistema de información geográfica (SIG) es una tecnología de manejo de información georeferenciada, que provee las funciones y las herramientas necesarias para almacenar, analizar y desplegar la información espacial. La Universidad de las Ciencias Informáticas dispone de una plataforma para el ensamblaje de los SIG, que carece de un proceso que sea capaz de validar la calidad tanto de la plataforma como de todos sus aplicativos, el cumplimiento de los requisitos y especificaciones para ser entregado al cliente, convirtiéndose en una necesidad para conocer los posibles errores sobre los que se deben trabajar para asegurar la calidad de estos sistemas. Para suplir esta carencia fue diseñada una estrategia de pruebas, que permitió organizar y planificar las actividades llevadas a cabo durante todo el proceso. Para la siguiente investigación se utilizaron métodos teóricos, empíricos y estadísticos, los que permitieron profundizar en el estudio del flujo de trabajo de pruebas como parte del desarrollo del software. Se tomó como muestra el Sistema de información geográfica de la UCI (SIGUCI), al que se le aplicaron un conjunto de pruebas y revisiones, las cuales permitieron detectar gran cantidad de errores que deben ser corregidos para certificar la calidad de dicho producto.

PALABRAS CLAVES

Pruebas, Sistemas de información geográfica, calidad, estrategia

DATOS EN INGLÉS

A geographic information system (GIS) is a technology of georeferenced information, which provides functions and tools to store, analyze and display spatial information. Its use became more prevalent with the development of appropriate computer technology, which encouraged the development of GIS products on the market. At the University of Information Sciences built a platform for the assembly of the SIG. For its economic and social impact attaches great importance to this product because there is a process that is capable of validating the quality of both the platform and all applications not meeting the requirements and specifications to be delivered to customer becomes a necessity to know the possible errors on what should work to ensure the quality of these systems. Responding to this problem, we designed a test strategy that allowed the organization and planning activities conducted throughout the process. For the following investigation were used within the scientific methods and empirical theoretical methods, which allowed further study of related workflow tests as part of software development. Sample was taken as the geographical information system of the ICU (SIGUCI). At the same were applied to a set of tests and reviews, which allowed us to detect many errors to be corrected to certify the quality of the product.

KEY WORDS

Testing, geographic information systems, quality, strategy

TABLAS Y FIGURAS

ÍNDICE

Introducción

El inicio del desarrollo de los SIG comienza en los años 60 relacionándose con diversos campos de investigación. Surgen bajo condiciones ambientales, es decir, era necesario estudiar las características físicas y químicas de la atmósfera para contribuir con su protección. A principios de los 80, se hizo más frecuente gracias al desarrollo de una tecnología informática adecuada, que fomentó la aparición de estos productos en el mercado. El surgimiento de los SIG puso en manos de los investigadores una herramienta que permitía el procesamiento espacial de gran cantidad de información en forma de capas.

Actualmente muchos de los problemas sociales que padece el mundo contienen una dimensión geográfica valorada de crítica como resultado de la pobreza y se manifiesta en la contaminación, superpoblación, los desastres naturales y otros, por ello; para la búsqueda de soluciones técnicas es necesario el acceso a la información y el trabajo de especialistas en equipos multidisciplinarios.

La tecnología SIG, además de una disciplina académica; es una industria importante ya que sus aplicaciones son actualmente enseñadas y expandidas en todas las universidades del mundo. Los jóvenes profesionales cubanos son cada vez más conscientes de las ventajas que proporciona que un sistema almacene y manipule información geográfica, ya que cerca del 80% de la información tratada en empresas e instituciones está relacionada con localizaciones geográficas o coordenadas espaciales ***(1)***.

Cuba no ha quedado exenta de este desarrollo, en 1987 surge el Instituto de Geografía de la Academia de Ciencias de Cuba "El SIG de Cuba" con el objetivo fundamental de actualizar el atlas nacional de Cuba. En las últimas décadas su uso se ha incrementado enormemente pasando del total desconocimiento a la práctica cotidiana. Todo esto se debe a determinados factores que han fomentado su desarrollo. Entre los que se encuentran: La reducción de costos de los equipos informáticos, el uso de la información geográfica desde múltiples ámbitos para ayudar la toma de decisiones y la sociedad de la información que demanda cada vez más cualquier tipo de datos.

Por todo esto se considera que están de total actualidad y son muchas las áreas en las que intervienen. Un SIG es un sistema dinámico computacional que utiliza datos geográficos, los cuales permiten mapear cualquier tipo de información almacenada en planillas o base de datos, para lograr un mejor análisis ***(2)***. El objetivo principal de un SIG es administrar, modelar y analizar información relacionada con datos espaciales.

En estos sistemas es elevado el nivel de responsabilidad e importancia, por lo que se hace necesaria la presencia de la calidad de software, que no es más que la forma de asegurar la concordancia entre los requisitos funcionales y los de rendimiento con los estándares de desarrollo ***(2)***. Las actividades planificadas y sistemáticas que contribuyen a que el producto logre satisfacer los requisitos expuestos aseguran la calidad del software. Para garantizar la calidad adecuada de un SIG es necesario que se les realicen pruebas de software; identificando los posibles errores de implementación, calidad o usabilidad ***(3)***. Y la detección oportuna de fallos en el software con el fin de minimizar los costos de reparación. Si se detecta una falla con rapidez, su corrección será mucho más barata. La Universidad de las Ciencias Informáticas (UCI) desde su surgimiento, ha obtenido significativos logros informáticos.

Trabaja en la formación integral de profesionales que respondan a las necesidades del progreso científico técnico y socioeconómico de la sociedad; para ello cuenta con varios Centros Productivos, que a su vez están conformados por diferentes Departamentos que contienen los Proyectos especializados en la elaboración de software. En el centro GEySED de la facultad 6, ideado para el desarrollo de softwares relacionados con la Geoinformática y la captura de señales digitales, se encuentra el Departamento Geoinformática donde se desarrolla el proyecto GENESIG, encargado de la elaboración de SIG mediante una plataforma que lleva ese mismo nombre.

El mismo es sustentado por un personal altamente calificado con la función de aplicar esta ciencia. Esta herramienta para el ensamblaje de SIG es interoperable con soluciones existentes, integradas con software de gestión para la toma de decisiones. En los últimos tiempos se ha incrementado exponencialmente el uso de aplicaciones de este tipo para la toma de decisiones, lo cual está íntimamente relacionado con el incremento de la efectividad y la disminución de los costos que se obtienen con la puesta en marcha de un sistema con estas características. Constituye un producto de gran actualidad y con grandes expectativas a nivel mundial.

Por el impacto económico notable en relación con los SIG actuales se le atribuye un valor incondicional a este sistema, el cual no cuenta con un proceso que sea capaz de validar la calidad tanto de la plataforma como de todos sus aplicativos, además del cumplimiento de los requisitos y especificaciones para ser entregado al cliente. Por todo esto se convierte en una necesidad conocer los posibles errores sobre los cuáles se deben trabajar para asegurar la calidad de estos sistemas con el mínimo costo y tiempo.

Esta situación condujo a declarar el siguiente **problema científico:** ¿Cómo verificar que el Sistema GENESIG y sus aplicativos sean entregados con la calidad requerida y cumplan con las funcionalidades predeterminadas? Para resolver el problema anterior se plantea como **objeto de**

estudio de esta investigación el proceso de pruebas de calidad, como parte del desarrollo del software, enmarcando como **campo de acción** el proceso de pruebas de calidad de software a la plataforma GENESIG y sus aplicativos.

Esta investigación se traza como **objetivo general:** Desarrollar una estrategia de pruebas para la plataforma GENESIG y sus aplicativos mediante un plan de pruebas que permita certificar la calidad de estos productos. Con la finalidad de orientar, servir de guía al proceso de investigación y la solución del problema se parte de la siguiente **hipótesis:** si se lleva a cabo un correcto diseño y aplicación de pruebas de calidad a la plataforma GENESIG y sus aplicativos se podrá verificar y revelar la calidad de estos productos.

Para darle cumplimiento al objetivo propuesto, se trazaron las siguientes **tareas de investigación:**

1. Caracterizar el estado del arte de los temas relacionados con las pruebas de software para los SIG.
2. Caracterizar los Sistemas de Información Geográfica.
3. Caracterizar el proceso de prueba en los Sistemas de Información Geográfica.
4. Definir la estrategia de pruebas que se le aplicará al producto.
5. Elaborar el plan de pruebas que se le realizará al proyecto SIGUCI.
6. Seleccionar la metodología a utilizar para realizar el proceso de pruebas.
7. Elaborar el documento de no conformidades detectadas.
8. Diseñar los Casos de Prueba.
9. Validar el producto.

En el transcurso de toda la investigación se aplican los siguientes **métodos científicos:**

Métodos teóricos: estos métodos permiten profundizar en el estudio de lo relacionado con el flujo de trabajo de pruebas. Crean las condiciones para dominar y analizar las características del proceso de pruebas de software que no se pueden observar directamente.

Método histórico-lógico: permite estudiar la trayectoria de los acontecimientos relacionados con las pruebas de software, así como su funcionamiento y desarrollo. Proporciona un exhaustivo análisis de las etapas del proceso de pruebas en sucesión cronológica para conocer su evolución y luego descubrir los temas fundamentales que están incluidos en él.

Método analítico sintético: ayuda a seguir un procedimiento para dividir lo complejo del desarrollo de las pruebas de software y posteriormente establecer una unión a partir de los resultados obtenidos.

Método de modelación: este método permite realizar la modelación de la estrategia de pruebas que se le aplicará al proyecto SIGUCI.

Para la obtención y elaboración de datos empíricos relacionados con el proceso de pruebas de software se utilizan los siguientes **métodos empíricos.**

Observación bibliográfica: es el método utilizado en el transcurso de toda la investigación ya que permite la recolección de la información relacionada con los aspectos tratados con definiciones y resultados de otros autores.

La presente investigación está formada por tres capítulos, de los cuales a continuación se brinda una descripción de las funciones en particular de cada uno de ellos.

Capítulo 1: Consideraciones teóricas relacionadas con el flujo de trabajo de pruebas como parte del proceso de desarrollo de software.

Este capítulo se compone de conceptos, definiciones, además de una caracterización precisa del proceso de pruebas. En este capítulo se encuentran sentadas las bases de la investigación.

Capítulo 2: Diseño de la estrategia de prueba para la plataforma GENESIG y sus aplicativos.

En este capítulo se diseña la estrategia de pruebas en la cual se planifica todo el proceso que se va a llevar a cabo. Además de brindar los elementos necesarios sobre el Plan de Pruebas y los objetivos que persigue, así como la realización de los Casos de Prueba.

Capítulo 3: Resultados obtenidos de la ejecución de las pruebas.

Concluyendo la investigación se exponen los resultados del proceso de pruebas desarrollado al proyecto SIGUCI, además de validar la estrategia aplicada.

Capítulo 1: Consideraciones teóricas relacionadas con el flujo de trabajo de pruebas como parte del proceso de desarrollo de software.

1.1 Introducción.

El proceso de pruebas constituye básicamente una fase en el desarrollo de software destinado a probar aplicaciones de rigor. Las técnicas, métodos, pasos, herramientas y toda una variedad de acciones inquisitivas guían el proceso de pruebas. Esto permite verificar productos altamente complejos y probar la presencia de errores en los mismos. Además revelar la calidad de un producto de software y comprobar el cumplimiento de éste con las especificaciones planteadas inicialmente como objetivos. La realización de actividades organizadas y bien definidas es clave para detectar fallas en la estabilidad, escalabilidad y eficiencia de un producto.

Los SIG se han desarrollado irregularmente en complejidad, tamaño y costo, por lo que se hace necesaria una correcta planificación, ejecución, control y evaluación de pruebas logrando refinación y perfeccionamiento en estos sistemas. Este capítulo aborda y profundiza los temas fundamentales relacionados con el flujo de trabajo de pruebas y su aplicación en los SIG. Diferentes conceptos, criterios, definiciones y características son expuestos por autores capacitados en lenguajes de desarrollo, métodos y técnicas especializadas de pruebas. Además se referencian valoraciones de elementos comunes como la calidad en el desarrollo de software y otras generalidades del flujo de trabajo de pruebas.

1.2 El papel de la Industria del Software.

En la comunicación, la computación y la electrónica numerosos y nuevos avances científico – técnicos han transformado y evolucionado a todos los países del mundo. Las Tecnologías de la Información y las Comunicaciones (TICs) han tenido la mayor influencia para el desarrollo continuo y social. Estas abren las puertas para formar parte de la inmensa cultura tecnológica que hoy nos rodea, además de acelerar el ritmo de las transformaciones en estructuras económicas, diseños industriales, informaciones, organización de las empresas, educación así como en la calidad de vida.

El poderoso impacto que ha tenido sobre la sociedad, hace que se pueda prescindir de ella cada vez menos. No se puede hablar de las TICs sin mencionar los medios de comunicación, la informática y otras tecnologías asociadas que forman parte de su definición. Enmarcando esta revolución de tecnologías se encuentra la industria del software, cada vez más aplicable a cualquier esfera de la ciencia de todos los países. Su posición dentro de las TICs es relevante, ya que posibilita

el desarrollo económico de cualquier nación. Los países desarrollados protagonizan la evolución de las nuevas tecnologías debido a sus potentes estrategias y métodos que les permiten liderar esta vanguardia. La industria del software en América Latina cuenta con el apoyo de instituciones y asociaciones, las cuales proveen grandes mejoras en los mercados, propiciando cambios positivos para sus competencias y alternativas de nuevos programas. En Cuba las posibilidades de transformación son frenadas por el bloqueo económico comercial y financiero; vil mecanismo que ha implantado la fuerte potencia de Estados Unidos sobre la isla. Esto ha impedido un pleno y total acceso al mercado mundial, afectando fuertemente la economía de Cuba. Por lo que se duplica la inversión de los recursos, ya que obliga a recurrir a mercados distantes para la adquisición de mercancías.

Por esta razón, se dilatan numerosos avances como la telefonía inalámbrica, la televisión digital, el internet y la computación, los cuales le proporcionan al país la creación de un elevado nivel cultural y científico. Estos logros posibilitan que Cuba sea capaz de prosperar en informatización y tecnologías. Un universo de posibilidades abren las comunicaciones satelitales, la telefonía inalámbrica, Internet, la televisión digital, la computación, a un país como Cuba, cercado por el conocido bloqueo económico, comercial y financiero con el que Estados Unidos le impide a la Isla el acceso a su mercado. No obstante, basándose sobre todo en sus recursos humanos y aprovechando sus recursos materiales y financieros, Cuba avanza en su informatización, priorizando el uso social y colectivo de las TICs.

Esto le ofrece al país una connotación diferenciada al resto de los países del mundo en cuanto a Tecnologías de la Información se trata. Es un objetivo selecto para el país incorporar las ramas sociales como la educación, salud, economía y otras existentes a las TICs, para de esta forma lograr la informatización de la sociedad. Encaminados a cumplir esta meta se han creado los Joven Club de Computación en el año 1987, proceso guiado por Fidel Castro Ruz. Estos centros ubicados en los barrios de cada municipio, han contribuido con la formación técnica integral de numerosas personas en la sociedad mediante una ardua preparación en temas de informática.

Ya sean niños, jóvenes, adultos o miembros desvinculados de labores obreras y hasta discapacitados han podido adquirir información y fortalecer sus conocimientos sobre las nuevas tecnologías. Otro paso importante fue el desarrollo de un modelo integral de informatización en los Centros de Salud Pública de todo el país. Se instaló un potente equipamiento para apoyar y potenciar la asistencia médica, la higiene y otros programas de salud imprescindibles para el uso social. Los centros de acceso público a las tecnologías gratuitamente y otros centros especializados en TICs, constituyen aspectos impulsores en cuanto a las actividades de informatización. Estudiantes

universitarios y técnicos profesionales gozan hoy de privilegios como el acceso a internet y la utilización de software educativos, los cuales posibilitan ampliar la gran comunidad tecnológica. En la industria del software se han alcanzados buenos y factibles resultados, aunque no los esperados después del empeño y esfuerzo puesto en marcha a través de las promociones, ferias, eventos y varias actividades correspondientes a este tema. Las áreas de salud, educación y deporte han sido las más vinculadas a la exportación de los productos, por lo que son responsables que el país pueda lograr una ubicación significativa en el mercado internacional.

Es notable que la causa principal por la cual Cuba aún no sea reconocida como uno de los países en el mercado internacional es el bloqueo impuesto por los Estados Unidos, con la única intención de frenar el vertiginoso desarrollo de la industria en Cuba. También son evidentes los grandes problemas de calidad existentes, lo que constituye un freno para el alcance de resultados satisfactorios ante el cliente. Es verdaderamente importante la calidad de software en el desarrollo científico – técnico, ya que con ella se lograría un incremento sostenible y permanente en la informatización de la sociedad cubana.

1.3 Calidad de Software. Definiciones.

"La Calidad de software es la concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente" ***(2)*** "El conjunto de características de una entidad que le confieren su aptitud para satisfacer las necesidades expresadas y las implícitas" ***(3)*** "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". ***(4)*** La calidad de software reúne las características y cualidades que necesita un producto para hacer válida su utilidad. Aseguran la eficiencia, corrección, flexibilidad, usabilidad e integridad de las aplicaciones en general.

Se puede afirmar que es un aspecto fundamental con máxima prioridad para un equipo de desarrollo a la hora de construir un producto de software. Es importante tener en cuenta que nunca se puede aseverar la perfección de un software, a pesar del gran empeño y esfuerzo que lleva a cabo el grupo de trabajo para asegurar sus actividades. Todos los centros de producción no solo anhelan y colocan entre sus objetivos producir software que logren cumplir con las expectativas de los clientes, sino además superarlas. Esta tarea es considerada la más compleja y difícil dentro del equipo de desarrollo. La calidad de software es medible y con grandes probabilidades a que varíe en

dependencia del software elaborado. El nivel de calidad de un software que será ejecutado una única vez, no va a ser el mismo que otro software encaminado a cumplir otras funcionalidades.

1.3.1 Factores que determinan la calidad del software.

Es de gran importancia profundizar en el conocimiento de los factores que miden y determinan la calidad del software, ya que constituyen puntos claves en el desarrollo de los mismos.

Se clasifican en tres grupos:

• Operaciones del producto: características operativas.

- Corrección
- Fiabilidad
- Eficiencia
- Integridad
- Facilidad de uso

• Revisión del producto: capacidad para soportar cambios

- Facilidad de mantenimiento
- Flexibilidad
- Facilidad de prueba

• Transición del producto: adaptabilidad a nuevos entornos

- Portabilidad
- Reusabilidad
- Interoperabilidad ***(4)***

1.4 ¿Qué son las pruebas de software?

Las pruebas de software conforman el conjunto de actividades que originan los procesos capaces de detectar los errores en un producto. Cualquier aplicación que haya sido construida necesita de las pruebas de software y se deben tener en cuenta los niveles de las mismas para su correcta ejecución. Este proceso permite indagar en cada una de las fases del software, para tales fines realiza una búsqueda consistente y eficaz, que posibilita probar cada una de las funcionalidades del sistema. Es importante aclarar que la realización de las pruebas asegura la presencia de errores en caso que éstos existan, pero nunca podrá ser capaz de garantizar que el producto esté libre de defectos.

1.4.1 Principios de las Pruebas de Software

Las pruebas de software son guiadas por normas y principios que un diseñador de casos de pruebas siempre debe dominar y tener en cuenta para lograr la efectividad de este proceso.

Según Pressman las pruebas de software deben estar regidas por los siguientes principios:

- A todas las pruebas se les debe hacer un seguimiento hasta los requisitos del cliente.
- Las pruebas deben ser planificadas antes de comenzar las iteraciones.
- El principio de Pareto es aplicable a la prueba del software. El 80% de los errores está en el 20% de los módulos. Hay que identificar esos módulos y probarlos muy bien.
- Las pruebas deben comenzar por “lo pequeño” y progresar hacia “lo grande”.
- No son posibles las pruebas exhaustivas.
- Para ser más efectivas, las pruebas deben ser conducidas por un equipo independiente al grupo de proyecto ***(4).***

1.4.2 Características de una buena prueba de software

Para que una prueba de software cumpla sus objetivos y alcance resultados satisfactorios, es necesaria una comprensión total del diseñador de casos de pruebas sobre el producto. Tener una idea de cómo podrían fallar las funcionalidades del mismo posibilita la búsqueda directa de posibles errores. La distribución de los recursos y la planeación con respecto al tiempo constituyen aspectos sumamente importantes para que las pruebas tengan éxito. Se debe analizar el propósito de cada prueba, lo que ésta garantiza, modifica y corrige, para de esta forma no repetir pruebas que persigan los mismos objetivos y de esta forma se ahorrarían factores imprescindibles y limitados como el tiempo y los recursos.

Cuando se está en presencia de pruebas similares, es viable valorar y evaluar sus alcances, así se escogería la prueba más factible a realizar, es decir, la que presenta altas probabilidades para la detección de errores en el sistema. En una línea descriptiva se encuentran dos extremos delicados que influyen en la ejecución de las pruebas. Las demasiado sencillas y las de alto grado de complejidad, por lo cual es aconsejable el punto intermedio entre ellas, es decir, una prueba cuya ejecución no sea complicada y logre sus objetivos

1.5 Flujo de trabajo de pruebas en el desarrollo de software según Proceso Racional Unificado (RUP)

Como resultado principal de la fase de implementación en el desarrollo de un software, se obtiene el modelo de implementación. Este modelo consta de subsistemas de implementación, componentes, así como la vista de arquitectura. La etapa de pruebas sigue este modelo, ya que se encarga de probar cada artefacto generado en la implementación. El flujo de trabajo de pruebas verifica cuidadosamente todos los resultados obtenidos en la implementación, sometiendo a una estructura de pruebas las construcciones que se llevan a cabo internamente y las que son entregables a los clientes.

1.5.1 Artefactos del flujo de trabajo de pruebas

Los artefactos que genera el flujo de trabajo de pruebas son:

- **Modelo de pruebas**

  Está compuesto por procedimientos, componentes, así como los casos de pruebas los cuales brindan una descripción de cómo van a ser probados los elementos de la implementación.

- **Caso de prueba**

  Especifica una vía o forma para realizarle pruebas al sistema. Incluye condiciones, aspectos de entrada y los resultados correspondientes con los que se proseguirá la prueba.

- **Procedimiento de prueba**

  Conforma las instrucciones capaces de especificar cómo se realizan los casos de prueba o cómo interactuar con una herramienta de pruebas. Describe los aspectos fundamentales para lograr un buen dominio y manejo de las pruebas.

- **Componente de prueba**

  Automatiza los procedimientos de pruebas y verifican los componentes en el modelo de implementación.

- **Plan de prueba**

  Es el documento que describe los tipos de pruebas que se llevarán a cabo, sus objetivos, los niveles, los recursos y la planificación en general del proceso de pruebas.

- **Defecto**

  Es una falla o un problema detectado durante revisiones al software. Un síntoma de un funcionamiento negativo del sistema.

- **Evaluación de prueba**

  Es el proceso que analiza el estado de los defectos encontrados, es decir, consiste en la evaluación de los resultados obtenidos en las pruebas ***(6)*.**

1.5.2 Actividades del flujo de trabajo de pruebas

Las actividades desarrolladas durante el flujo de trabajo de pruebas son:

Planificar prueba: esta actividad ejecuta las tareas que organizan y dirigen las pruebas de software. Es la encargada de planificar los recursos y esfuerzos de las pruebas en cada iteración. Describe la estrategia de prueba que identifica los tipos de pruebas que se van a realizar, cómo hacerlo y cuándo ejecutarlas.

Diseñar prueba: esta actividad permite el diseño de los casos de pruebas que se van a realizar, además de identificar los procedimientos a seguir para elaborar dichos casos de pruebas.

Implementar prueba: el objetivo que persigue esta actividad es automatizar los posibles procedimientos de pruebas, mediante la creación de nuevos componentes.

Realizar pruebas de integración: mediante las pruebas de integración se puede construir una estructura del programa reuniendo todos los módulos que se hayan probado, para verificar el funcionamiento de ellos juntos.

Realizar pruebas de sistema: permite la ejecución de las pruebas de sistema y la recopilación de sus resultados, ya que el software puede incorporarse a nuevos hardware.

Evaluar prueba: el propósito de esta actividad es evaluar sus esfuerzos. Establecen una comparación entre todos los resultados obtenidos y los especificados en el plan de pruebas. De esta forma se determina el número de pruebas que se deben realizar además del nivel de calidad de software ***(6)*.**

Todas las actividades que guían este flujo de trabajo son las encargadas de asegurar el funcionamiento correcto de las pruebas que van a ser aplicadas al software. Estas actividades permiten una correcta planificación y gestión de todos los artefactos generados, para lograr éxito en los resultados obtenidos luego que concluya todo el proceso y a la misma vez encontrar la mayor cantidad de errores en el sistema.

1.5.3 Plan de pruebas

El plan de pruebas es un documento en el que se describen todas las pruebas que van a ejecutarse en el sistema. Debe reflejarse el propósito de cada una de ellas, sus objetivos, y lo que se va a probar con la misma. Se detalla cada paso y procedimiento que se realiza, además de describir en qué consisten. Se definen los resultados esperados para comprobarlos con los obtenidos. Un plan de pruebas contiene toda la información sobre los casos de pruebas, los elementos necesarios para su ejecución, así como sus responsables y las pruebas que están planificadas.

1.5.4 Estrategia de pruebas

Las pruebas de software constituyen un conjunto de tareas y actividades, que pueden ser planificadas y organizadas antes de ejecutarse, mediante modelos y plantillas asegurando la calidad de este proceso de pruebas. Esto implica la necesidad de elaborar un documento que contenga los datos y las informaciones principales acerca de cada prueba que se le realice al sistema, así como los métodos de diseño de casos de pruebas y otros elementos claves.

Varios autores como Pressman proponen diferentes estrategias de prueba, las cuales presentan las características siguientes ***(4)***:

- Comienzan a nivel de módulo.
- Son apropiadas diferentes técnicas de pruebas.
- La prueba la lleva a cabo el responsable del desarrollo del software y en los casos en que el proyecto es grande un grupo independiente de pruebas.
- La depuración debe estar incluida en cualquier estrategia de prueba.

Una estrategia de prueba de software debe incluir pruebas de alto nivel, las cuales se encargan de validar que los requisitos implantados por el cliente se correspondan con las funcionalidades del sistema. Uno de los objetivos fundamentales que persigue una estrategia de prueba es proporcionarle al jefe de proyecto y a los probadores una guía, mediante la cual puedan darle seguimiento y medición al progreso y los problemas que se presenten en caso que ocurra una presión por parte del cliente ***(4)*.**

¿Cómo lograr una estrategia de prueba correcta?

Cuando se define una estrategia de prueba se deben tener en cuenta aspectos relevantes para eliminar algún posible fracaso en ella. Para definir la estrategia correcta es necesario:

- **Especificar los requisitos del producto de manera cuantificable antes que comiencen las pruebas.**

Partiendo que detectar los errores del software es el objetivo principal de las pruebas, es preciso aclarar que evaluar otros puntos como la portabilidad, facilidad de uso y mantenimiento, entre otros aspectos determina excelentes resultados en el proceso de pruebas.

- **Establecer los objetivos de la prueba de manera explícita.**

Es necesario describir detalladamente cada uno de los objetivos de las pruebas, su alcance, su efectividad, así como su costo y tiempo de ejecución, es decir, una exacta planificación de las mismas.

- **Comprender qué usuarios van a manejar el software y desarrollar un perfil para cada categoría de usuario.**

Los casos de uso deben describir los escenarios para reducir el esfuerzo general de las pruebas.

- **Desarrollar un plan de prueba que profundice fundamentalmente la prueba de ciclo rápido.**

La prueba de ciclo rápido es de gran importancia ya que posibilita el control de los niveles de calidad y las correspondientes estrategias de pruebas.

- **Construir un software robusto, diseñado para probarse a sí mismo.**

El software debe contar con técnicas de depuración anti errores, posibilitando que el mismo pueda diagnosticar clases de errores.

- **Usar revisiones técnicas formales efectivas como filtro antes de la prueba.**

Con las revisiones técnicas formales se reduce el esfuerzo en la realización de las pruebas, además persiguen los mismos objetivos que éstas últimas, encaminadas a lograr software de alta calidad.

- **Desarrollar un enfoque de mejora continua al proceso de prueba, midiéndose la estrategia de prueba.**

Las métricas desarrolladas durante el proceso de pruebas son importantes para el control de las estadísticas generales ***(4)*.**

1.5.5 Pruebas en el desarrollo de un software

El ciclo de vida de un software está estructurado por las fases de inicio, elaboración, construcción y transición, además de diferentes flujos de trabajo. Las pruebas pueden ser planificadas al definirse el sistema en la fase de inicio, aunque mayormente se llevan a cabo en los resultados de la fase de implementación. Por tanto la realización de las pruebas es centrada en las fases de elaboración y construcción. Aquí se detecta la mayor cantidad posible de errores en la arquitectura y

desarrollo. En la fase de transición se trabaja en la corrección de defectos encontrados. Algunos de los casos de pruebas pueden ser utilizados como casos de pruebas de regresión, los cuales especifican los procedimientos a seguir para realizarle estas pruebas a construcciones posteriores. Las de regresión aumentan notablemente según sus iteraciones. El modelo de pruebas de software cambia constantemente por varios puntos, pero es importante y necesario mantenerlo durante todo el ciclo de vida del software.

1.6 Elementos en el Proceso de Pruebas

1.6.1 Niveles de pruebas.

Los niveles de prueba son los escenarios donde se aplican las pruebas. Están estructurados de forma ascendente para lograr concordancia y organización en los procesos de pruebas. Todos deben realizarse con el objetivo de garantizar la calidad del software de forma continua e incremental.

Los niveles de pruebas son los siguientes:

Pruebas de Unidad

Estas pruebas centran el proceso de validación en los componentes o módulos, es decir, menores unidades del diseño de software. Verifican las interfaces de cada módulo con el objetivo de asegurar una fluidez total en todos los datos que interactúan con la unidad que se está probando. Conforman los algoritmos necesarios para verificar la existencia de uno de los pilares más importantes de la seguridad informática: la integridad de la información. Además se prueban los valores límites asegurando las funcionalidades de los módulos teniendo en cuenta las restricciones de procesamiento.

Es necesario que el flujo de datos en las interfaces de los módulos quede asegurado y protegido de posibles errores, ya que esto es primordial para realizar las demás pruebas, pues si la entrada de datos es incorrecta, las demás pruebas que le siguen no tienen validez alguna. No sólo es recomendable probar durante la prueba de unidad la información interna de cada módulo, sino también la global que de cierta forma interactúe con el módulo que se esté probando. Los errores causados por comparaciones y cálculos incorrectos influyen grandemente en los caminos de ejecución. Para detectar esto en el tiempo correcto sería esencial un buen diseño de casos de pruebas.

Pruebas de Integración

Con la prueba de unidad se verifica cada módulo o componente por separado, pero es necesario comprobar que unidos también funcionen correctamente. Cuando se combinan todos, pueden existir problemas debido a su interacción, provocando pérdida de datos en alguna interfaz. Si las funciones de cada módulo en conjunto llegan a transformar la función principal, provocaría problemas en las etapas posteriores del ciclo de vida del software.

Las pruebas de integración se realizan durante la construcción del sistema, con el objetivo de crear la estructura del programa. Así se comprobaría la unión de todos los componentes mediante sus interfaces, además de asegurar que cumplen con las funcionalidades requeridas. Por todo lo antes demostrado es notable que con la técnica de integración incremental queden resueltas las posibles fallas a ocurrir durante la interacción de los módulos del sistema.

Esta técnica traza las estrategias siguientes:

Integración descendente: consiste en unir todos los módulos, comenzando por el programa principal y luego continuando con el resto, transitando por una jerarquía de control. De esta forma todos los módulos subordinados se incorporarían al módulo principal y así se conformaría una mejor estructura. Utiliza una Integración primero en profundidad integrando todos los módulos que se encuentren en un mismo camino de control principal de la estructura. Para realizar la selección se basa en las características de la aplicación. Mediante la Integración primero en anchura se mueve por la estructura horizontalmente y va integrando los módulos subordinados de cada nivel.

Integración ascendente: comienza la construcción de las pruebas a partir de los módulos en los niveles más bajos de la estructura. Primeramente los módulos de bajo nivel se combinan formando grupos, posteriormente se escribe un controlador con el objetivo de gestionar los casos de pruebas. Luego se les realizan las pruebas a estos grupos para una vez que concluyan, eliminar los controladores y combinar los grupos de abajo hacia arriba a través de la estructura del programa.

Pruebas de Validación

Luego de terminadas las pruebas de integración, ya el software se encuentra estructurado en un paquete, se han detectado los errores de interfaz, además de haberse corregido los mismos, por lo que se hace necesaria la realización de pruebas finales al software. Esta es la etapa de validación, en la cual es sumamente importante el conjunto de requisitos de software. La validación es conseguida siempre que el sistema cumpla con las expectativas del cliente. Para la validación del software es necesaria la realización de pruebas de caja negra, demostrando la excelencia en las funcionalidades del sistema.

Pruebas de Sistema

Se realizan durante la construcción del sistema trazándose como objetivo fundamental probar completamente el sistema verificando su integridad y funcionalidad, para de esta forma garantizar la calidad del software ante el cliente. Constituye una etapa vital en el ciclo de vida del software, ya que aquí es donde queda detectada la mayor cantidad de errores. Estas pruebas permiten ejercitar el sistema comprobando la integración de la información, el funcionamiento de las interfaces, entre otros aspectos relevantes que propician eficiencia y efectividad en sus procesos.

En este nivel se realizan pruebas como:

- **Prueba de recuperación:** es importante que los sistemas cuando detecten errores puedan recuperarse de los mismos para darle continuidad a sus procesos. En muchos casos los sistemas deben ser tolerante a fallas, para que no se afecte totalmente su funcionamiento. Esta prueba permite forzar el fallo en el sistema de diferentes formas. Le permite verificar que la recuperación que realice sea la indicada. En caso que el sistema cuente con una recuperación automática, se deben evaluar los mecanismos que utiliza, mientras que si la recuperación depende de alguna función humana se necesita comprobar el tiempo de reparación.
- **Prueba de seguridad:** todo sistema que maneje información importante es sensible a entradas inapropiadas de usuarios, cuya intención es dañar, falsificar o adquirir datos privados. Mediante esta prueba se verifican los mecanismos de protección, es decir, se comprueba que sean capaces de proteger la información de cualquier acceso indebido al sistema. El responsable de esta prueba debe actuar como cualquier individuo, con la intención de acceder al sistema a través de cualquier medio.
- **Prueba de resistencia (Stress):** mediante esta prueba se puede probar el funcionamiento del sistema enfrentando a los programas con situaciones poco comunes utilizando la mayor cantidad de recursos.
- **Prueba de rendimiento:** esta prueba se encarga de probar el rendimiento del software en tiempo de ejecución. Está presente en todo el proceso de pruebas, incluso en las pruebas de unidad se debe verificar el rendimiento de cada módulo individual. No se asegura el rendimiento del sistema hasta que no se realice esta prueba con todos los módulos unidos. Estas pruebas necesitan de instrumentos de hardware y software debido a que muchas veces se mide la utilización de recursos.

- **Pruebas de funcionalidad:** están enfocadas a validar las funcionalidades del sistema, así como sus métodos, servicios, casos de uso y especificaciones importantes que definen la calidad de su ejecución.
- **Prueba de usabilidad:** se realizan comprobando factores humanos, relacionados con la estética, verificando consistencia y utilidad en las interfaces de usuarios y en los materiales o documentación de los mismos.

Pruebas de Aceptación

Las pruebas de aceptación permiten validar que el sistema cumpla con el funcionamiento esperado y deseado tanto por los clientes como por los desarrolladores del software. Su objetivo principal es permitirle al usuario del sistema determinar su aceptación, teniendo en cuenta dos aspectos claves para garantizar su calidad: funcionalidad y rendimiento. El usuario del sistema es el encargado de definir estas pruebas, además de ejecutarlas y darles su aprobación final.

1.6.2 Diseño de Casos de Prueba

Varias personas en el mundo tergiversan los objetivos y la importancia que tienen los casos de pruebas. Su realización y aplicación lleva implícito gran esfuerzo y dedicación, al punto que puede ser comparable con el empeño con que se diseñó el producto inicialmente. Considerar las pruebas como un proceso insignificante y secundario es un error grave, capaz de influir en el buen funcionamiento de cualquier software, dando paso al freno absoluto de la calidad del mismo. Los objetivos de las pruebas de software son fundamentales para el diseño de los casos de pruebas, ya que lo factible sería pensar siempre en encontrar y detectar la cantidad máxima de errores en el menor tiempo y costo posible.

Existen dos formas para probar los productos de software. La primera vía es realizar pruebas que detecten errores en las funcionalidades del sistema llevadas a cabo en la interfaz del software. Estas pruebas se denominan Pruebas de Caja Negra. La otra forma es asegurar todos los componentes internos del sistema. Consiste en probar que el funcionamiento interno se ajuste a lo especificado en un inicio. Estas pruebas son las Pruebas de Caja Blanca.

Casos de Prueba

Los casos de pruebas no son más que un conjunto de condiciones representadas como variables. Las mismas le permiten a la persona que ejerce el rol de analista, determinar si los requerimientos cumplen sus funcionalidades correctamente. El software es desarrollado con la intención que sus funciones tengan resultados satisfactorios basándose en las especificaciones del cliente y los casos de pruebas se encargan de validar y asegurar que esto ocurra. Es importante

conocer, dominar, interpretar y entender las funciones de cada requerimiento, ya que por cada uno de ellos debe existir un caso de prueba. Cuando se escribe un caso de prueba se debe tener en cuenta, que es necesario incluir en él una descripción detallada de cada funcionalidad que éste va a probar.

En caso de que uno o varios requisitos contengan otros secundarios, entonces cada uno de ellos debe tener un caso de prueba. Diferentes metodologías aceptan y proponen elaborar dos casos de prueba por cada requisito. Uno positivo, con el fin de probar valores aceptados, coherentes y correctos y el otro para realizar todo lo contrario. El responsable de las pruebas debe haber realizado una excelente investigación o tener consigo una potente documentación que le sirva de guía y apoyo para ejecutar correctamente su función. Los casos de pruebas deben estar constituidos por pautas fundamentales que no se pueden omitir. Primeramente una correcta planificación de la o las pruebas que se van a realizar.

En este punto es importante la organización de las actividades que se llevarán a cabo así como su secuencia y distribución. Otro paso a seguir es el diseño de los casos de prueba, la ejecución de las mismas y finalmente su evaluación. Los casos de pruebas reúnen las mismas características que debe tener una buena prueba de software. Un caso de prueba:

- Nunca debe tener redundancia, es preciso, claro y específico.
- Debe ser representativo, capaz de explicar claramente su objetivo y demostrar su función.
- Requiere de un punto medio entre lo simple y lo complejo. No debe tocar ninguno de estos dos extremos.

Estructura de los casos de Prueba

Los casos de prueba escritos consisten en tres partes con subdivisiones:

- **Introducción/visión general**
  - Identificador
  - Caso de prueba dueño/creador
  - Versión
  - Nombre
  - Identificador de requerimientos
  - Propósito
  - Dependencias
- **Actividades de los casos de prueba**
  - Ambiente de prueba/configuración
  - Inicialización

- Finalización
- Acciones
- Descripción de los datos de entrada

➢ **Resultados**

- Resultados esperados
- Resultados reales contienen una breve descripción de lo que el analista encuentra después que los casos de prueba se hayan completado. Esto se sustituye a menudo con un Correcto/Fallido. Si un caso de prueba falla, frecuentemente la referencia al defecto implicado se debe enumerar en esta columna ***(4)*.**

1.6.3 Técnicas de Diseño de Casos de Prueba

Las técnicas de diseño de casos de pruebas constituyen el conjunto de actividades planificadas, organizadas y secuenciales que facilitan y ayudan al desarrollo de un correcto proceso de pruebas, permitiendo así obtener como resultado un software de alta calidad. Los tres enfoques de pruebas que se identifican son:

- Técnica de prueba de caja blanca: consiste en centrarse en la estructura interna, es decir, en la implementación del programa para de esta forma elegir los casos de prueba.
- Técnica de prueba de caja negra: centrada en las funcionalidades del software para derivar los casos de prueba.
- El enfoque aleatorio: consiste en la utilización de modelos que generalmente son estadísticos representando las posibles entradas al programa creando a partir de ellos los casos de prueba ***(7)*.**

Pruebas de Caja Blanca

La Prueba de Caja Blanca es un método utilizado para realizar el diseño de casos de prueba usando una estructura procedimental para su derivación. Mediante los métodos de esta prueba el ingeniero de software puede derivar casos de prueba que garanticen que:

- Se ejercitan al menos una vez todos los caminos independientes de cada módulo.
- Se ejercitan todas las decisiones lógicas en sus caras verdaderas y falsas.
- Se ejecutan todos los bucles en sus límites.
- Se ejercitan las estructuras de datos internas para asegurar su validez.

En ocasiones ocurre un gasto de tiempo y energía probando la lógica del programa, sin embargo una vía más factible sería verificando el cumplimiento pleno de las funcionalidades a partir

de las pruebas de caja negra. No siempre es así, pues muchas veces se hace necesaria la realización de las pruebas de caja blanca por diferentes situaciones, entre las cuales están: ***(4)***

- Los errores lógicos y las suposiciones incorrectas son inversamente proporcionales a la probabilidad de que se ejecute un camino del programa.
- Existen ocasiones en las que se cree que un camino lógico tiene pocas posibilidades de ejecutarse aunque éste se puede ejecutar de forma normal.
- Los errores tipográficos son aleatorios. Muchos de estos errores solo podrán ser detectados mediante esta prueba.

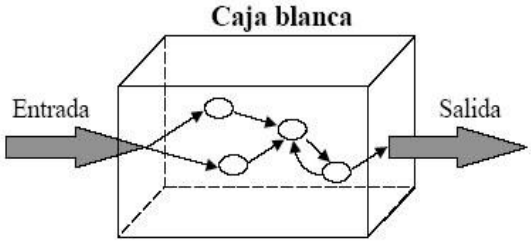


Figura 1: Prueba de caja blanca

Dentro de las técnicas de caja blanca se encuentran las siguientes pruebas:

Prueba de camino básico: Este método permite medir la complejidad lógica de un diseño procedimental y a la vez usar esta medida como una guía de apoyo para definir conjuntos básicos de caminos de ejecución. Los casos de pruebas que se obtienen a partir del conjunto básico garantizan que se ejecuten al menos una vez cada sentencia del programa. La notación del grafo de flujo forma parte importante de esta técnica, ya que permite representar el flujo de control lógico.

Pruebas de Condición: Esta prueba ejercita las condiciones lógicas que contienen los módulos de los programas. Una condición está formada tanto por un operador lógico, una variable lógica, operadores relacionales o una operación aritmética. Cuando estamos en presencia de una condición con errores, es porque uno o más de sus componentes son incorrectos. Las condiciones pueden presentar diferentes tipos de errores, entre los que se encuentran:

- Error en un operador lógico
- Errores en variables lógicas.
- Errores en paréntesis lógicos.
- Errores en operadores relacionales.
- Errores en una expresión aritmética.

Prueba del flujo de datos: Mediante esta prueba se seleccionan los caminos de prueba de un programa teniendo en cuenta la ubicación de las definiciones y los usos de todas las variables. Existen varias estrategias del flujo de datos, una de ellas se basa en que se debe cubrir al menos una

vez cada cadena de definición. Todavía no puede asegurarse por completo el cubrimiento en los casos de una ramificación en construcciones como when – else. Por lo que se llega a la conclusión que esta estrategia es factible para la selección de caminos de prueba en programas que presenten las sentencias “if” o “bucles anidados”.

Prueba de Bucles: Estas pruebas no son calificadas con la importancia que realmente presentan en el proceso de las pruebas del software, sin embargo constituyen la piedra angular de la mayoría de los algoritmos que se implementan en los softwares. Los bucles complejos son otra zona de debilidades y propensa a fallas y errores. Esta técnica de caja blanca está centrada específicamente a validar las construcciones de bucles que se realicen y pueden definirse en cuatro clases fundamentales de bucles:

- Bucles simples
- Bucles concatenados
- Bucles anidados
- Bucles no estructurados ***(4)***

Pruebas de Caja Negra

Las pruebas de caja negra son aquellas que se centran en los requerimientos funcionales del software, permitiendo derivar condiciones de entrada que ejerciten prácticamente todos los requerimientos funcionales de un programa. Estas pruebas persiguen como objetivo, descubrir varios tipos de errores, diferentes a los que se puedan detectar mediante los métodos de caja blanca. Los errores que identifican las pruebas de caja negra pueden ser de interfaz, funciones incorrectas o que falten, errores tanto en las estructuras de datos como en el acceso a las bases de datos, errores de rendimiento y de inicialización o terminación. Estas pruebas se realizan después de la fase de la prueba, ya que al desarrollar pruebas de caja negra se ignora completamente la estructura de control centrando su atención solo en el campo de la información.

Al aplicar las técnicas de pruebas de caja negra se pueden obtener casos de pruebas capaces de satisfacer criterios como:

- Casos de Prueba que reducen, en un coeficiente que es mayor que uno, el número de casos de prueba adicionales que se deben diseñar para alcanzar una prueba razonable.
- Casos de Prueba que digan algo sobre la presencia o ausencia de clases de errores en lugar de errores asociados solamente con la prueba que se esté realizando ***(7)*.**

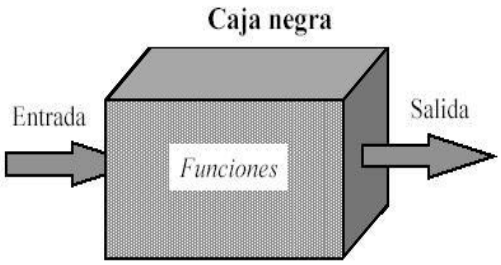


Figura 2: Prueba de caja negra

El método de prueba de caja negra contiene técnicas que son muy utilizadas en la actualidad, debido a sus grandes ventajas para probar las funcionalidades de un software. Entre ellas se encuentran:

Técnica de Partición Equivalente:

Constituye un método de caja negra el cual divide el campo de entrada de un programa en diferentes clases de datos, y a partir de éstas derivar los casos de pruebas. Esto permite que se descubra inmediatamente una clase de errores reduciendo el número total de casos de pruebas a desarrollar.

Análisis de valores límites

Generalmente los errores tienden a producirse más en los valores límites del campo de entrada que en cualquier otra parte, por lo que se ha diseñado la técnica de Análisis de Valores Límites (AVL). Esta prueba complementa a la partición equivalente. Permite seleccionar todos los casos de prueba que ejerciten los valores límites y no solo se concentra en los campos de entrada, sino también en los de salida.

Las directrices de AVL son similares en muchos aspectos a las que proporciona la partición equivalente:

- Si una condición de entrada especifica un rango delimitado por los valores a y b, se deben diseñar casos de prueba para los valores a y b, además para los valores justo por debajo y justo por encima de a y b, respectivamente.
- Si una condición de entrada especifica un número de valores, se deben desarrollar casos de pruebas que ejerciten los valores máximo y mínimo. También se deben probar los valores justo por encima y justo por debajo del máximo y del mínimo.

- Aplicar las directrices 1 y 2 a las condiciones de salida. Se deben diseñar casos de pruebas que creen un informe de salida que produzca el máximo (y el mínimo) número permitido de entradas en la tabla.
- Si las estructuras de datos internas tienen límites preestablecidos, hay que asegurarse de diseñar un caso de prueba que ejercite la estructura de datos en sus límites.

Por lo general casi todos los ingenieros de software utilizan alguna forma de AVL, debido a que es una de las pruebas más completas. Esto posibilita que tenga mayor probabilidad a la hora de la detección de errores, aspecto importante para lograr calidad en el producto ***(4)*.**

Prueba de comparación

Los software han presentado situaciones, en los que su fiabilidad pasa por momentos realmente críticos. Una de las causas que originan este problema es la utilización de software y hardware redundante. En estos casos varios equipos de ingeniería de software desarrollan versiones independientes de una aplicación por separado usando las mismas especificaciones. Cuando esto sucede es necesario probar con los mismos datos de prueba todas las versiones existentes verificando la salida, para posteriormente ejecutar las versiones en paralelo y realizar una comparación en tiempo real de los resultados para garantizar la consistencia.

Prueba de tabla ortogonal

Existen aplicaciones con su dominio de entrada extremadamente limitado, esto quiere decir que la cantidad de parámetros de entrada es pequeña. Cuando esto sucede es necesario comprobar cada paso en el proceso del dominio de entrada. Este método es útil cuando se deseen encontrar errores asociados a fallos localizados.

1.7 Caracterización del rol de Diseñador de Casos de Prueba

El equipo de desarrollo que tiene la responsabilidad de construir software, alcanzando la mayor y mejor calidad posible, necesita contar con un personal altamente calificado, que sea capaz de detectar posibles fallas o errores que se presenten en la elaboración del producto. Los grupos de trabajo especializados en las pruebas de software, cuya misión es garantizar que el producto final sea entregado al cliente libre de defectos, son dirigidos precisamente por un responsable que se encargará de la realización de las pruebas, quien desempeña el rol de diseñador de casos de pruebas.

Las actividades y tareas que son ejecutadas por este importante rol guían un complejo proceso en el cual se identifican, definen y se llevan a cabo las pruebas requeridas. Una vez realizadas, se documentan las no conformidades encontradas y posteriormente se discuten con el

equipo de desarrollo. quien argumentará las causas y razones que propiciaron estas no conformidades. Finalmente el diseñador de casos de pruebas debe asegurarse que queden definidas las estrategias pertinentes para la mejora de los procesos defectuosos logrando entregar en perfecto estado el software final.

1.8 Generalidades de los SIG

1.8.1 Definiciones de los SIG

Un SIG es definido como:

"Es un sistema de hardware, software y procedimientos, diseñados para soportar la captura, el manejo, la manipulación, el análisis, el modelado y el despliegue de datos espacialmente referenciados (georreferenciados), para la solución de los problemas complejos del manejo y planeamiento territorial" ***(1).*** Este sistema de computación dinámico facilita más funcionalidades y mejores resultados que los mapas estáticos, además son capaces de producir estos últimos. Gestiona el mapeo de toda la información que se relacione con la geografía, es decir, posibilita su almacenamiento, manipulación, análisis, comprobación y selección realizando una correcta estructuración de estos datos.

1.8.2 Funcionamiento de los SIG

Los SIG estructuran toda su información en capas (layers), donde cada una de ellas contiene y muestra un determinado tipo de información, ya sean mapas o regiones de los mismos. Existen dos tipos fundamentales de datos geográficos, el modelo raster y el modelo vectorial. Todo esto conlleva a que puedan identificarse dos tipos de formatos, los cuales serían: raster y vectorial. El formato vectorial contiene toda su información mediante líneas, puntos y polígonos, es decir, que cuando se hace referencia a una propiedad o característica puntual, se describe solo mediante un punto.

Las calles, mares, ríos entre otras características lineales se representarán a través de las líneas y las zonas, regiones y territorios entre otros se almacenan con la forma de polígonos. Este modelo se utiliza para geometrías sencillas y proporciona una cómoda edición, es decir, eliminar copiar o modificar objetos en el mapa sin alterar la información. Cuando se amplían los objetos no se produce pérdida de resolución y su tamaño de archivo es pequeño ya que solo incluye fórmulas y procedimientos matemáticos. El formato raster está formado por un conjunto de celdas o píxeles que permiten la representación de la información como una imagen escaneada. Este modelo ha

evolucionado de forma increíble. Es más utilizado en la construcción de mapas complejos que presenten gran variedad de colores.

Los SIG realizan la ejecución de seis procesos fundamentales: ***(1)***

- **Ingreso:** para que los datos geográficos puedan utilizarse en los SIG deben ser convertidos a un formato digital, es decir, a archivos de computación. Existen SIG que tienen implementadas funcionalidades que permiten automatizar este proceso.
- **Manipulación:** los tipos de datos requeridos para cualquier proyecto de SIG deben ser manipulados, ya que para que sean compatibles con el sistema, es necesario transformarlos hasta que se encuentren en la misma escala.
- **Manejo/Administración:** cuando un proyecto tiene gran variedad de datos e información y la cantidad de usuarios que interactúan con ellos es elevada, se recomienda el uso de un software que permita almacenar, organizar y manejar los datos. Para esto existen los sistemas de manejo de bases de datos.
- **Consulta:** las consultas brindan informaciones necesarias a los analistas o administradores utilizando distintos niveles de datos.
- **Análisis:** es necesario llevar a cabo procesos de análisis de datos geográficos. Esto se logra mediante la búsqueda de patrones que construyan escenarios con mayores funcionalidades.
- **Visualización:** se considera mucho más factible la visualización de datos geográficos a través de los mapas, ya que resultan más atractivos y específicos para los usuarios. En la actualidad muchos SIG cuentan con herramientas potentes que extienden el arte de la cartografía.

1.8.3 Componentes de los SIG

El funcionamiento de un SIG depende de 5 componentes claves:

Hardware: es la computadora ya sea servidor o PC cliente en la que opera un SIG, las cuales pueden ser individuales o estar ubicadas en red. Estas deben brindar determinadas propiedades que permitan satisfacer las necesidades de la aplicación.

Software: el software de un SIG es responsable del almacenamiento, análisis y muestra de toda la información geográfica que presente la aplicación. Deben proporcionar herramientas como sistemas gestores de base de datos, funciones para insertar y manipular datos geográficos, herramientas que implementen consultas y la visualización de información geográfica así como una interfaz gráfica de usuario para facilitar el acceso de este a las herramientas.

Datos: en los datos recae la mayor importancia de un SIG, ya que estos pueden asegurar gran parte del presupuesto de la implementación. Es necesario que los datos cuenten con la calidad requerida, principalmente los de base.

Personas: las personas en el funcionamiento de un SIG juegan un papel fundamental, ya que los usuarios son necesarios para que logren actualizar toda la información que se maneja en la aplicación. Se requiere de personal calificado para que utilice el hardware, además de investigar y gestionar las herramientas de software.

Métodos: un SIG necesita del correcto diseño de un plan, contenido por reglas que constituyen los modelos y los estándares para guiar las tareas de cada organización ***(1)*.**

Un SIG que no conste con al menos uno de los componentes expuestos anteriormente carecería de funcionalidades importantes y necesarias para el usuario. Cada componente lleva implícitos tareas, actividades e incluso otros componentes que dan paso a la calidad que se debe alcanzar para cumplir todos los requisitos y objetivos del sistema satisfactoriamente.

1.8.4 Importancia de los SIG

Desde tiempos remotos representar la superficie terrestre junto con todos los objetos y elementos que existían sobre ella se convirtió en una necesidad para el hombre, que desde el surgimiento de la Cartografía como ciencia, ha alcanzado un alto nivel de desarrollo, ya que muchos de los progresos en la actualidad se han logrado gracias a ella. Numerosas herramientas informáticas y técnicas de la computación son de gran uso en la cartografía, permitiendo de esta forma la representación, modelación, cálculos que determinan referencias geográficas entre otras funcionalidades, posibilitando que estos sistemas faciliten la toma de decisiones con respecto a datos espaciales. Una completa fusión de tecnologías ha dado surgimiento a los SIG, los cuales representan una de las más potentes combinaciones de hardware y software dentro de la revolución científico – técnica.

La tecnología de los SIG funciona como una industria poderosa, manejadora de grandes capitales que involucra a casi todo el mundo. Su uso se expande y se enseña en todas las escuelas y universidades con el propósito de alimentar y fortalecer los conocimientos sobre esta importante herramienta. Con el transcurso de los años jóvenes y adultos profesionales y no profesionales han aprendido a valorar la importancia y utilidad que presenta un SIG para el manejo, control y gestión de la información geográfica. En la actualidad el mundo enfrenta una gama de problemas los cuales frenan el desarrollo tecnológico, político y social de cualquier país. Muchas de estas problemáticas se deben a los desastres naturales, la contaminación y otros temas fundamentales que tienen

dimensiones geográficas críticas. Para darle solución a dificultades referentes a las ramas del saber, es necesario acceder a información relacionada con la geografía y sólo un SIG permite la interacción y manipulación de estos datos geográficos. Un SIG tiene aplicaciones en todas las esferas y los ámbitos sociales. Estos se encargan de analizar las informaciones geológicas y sísmicas. Disímiles situaciones de localidades se presentan a diario en la vida de las personas que pueden ser solucionadas gracias a estos sistemas que posibilitan la realización de consultas geográficas, mejorar la integración organizacional entre otras funcionalidades dando paso a ejecutar las acciones correctas e indicadas en cada caso.

1.9 Herramientas para Pruebas Automatizadas.

Para el desarrollo de las pruebas automáticas de software se ha implementado un gran número de herramientas persiguiendo diferentes objetivos, entre ellos facilitar, agilizar y asegurar este complejo proceso. Todas se desarrollan en varios tipos de escenarios o sistemas operativos. Una muestra de algunas de estas herramientas son las siguientes:

Junit: Herramienta especialmente diseñada con el objetivo de implementar y automatizar la realización de pruebas de unidad en el lenguaje Java.

Selenium: Herramienta de Software Libre que se utiliza para la realización de pruebas basadas en aplicaciones Web. Las pruebas de Selenium son ejecutadas directamente en un navegador y facilitan las pruebas de compatibilidad en navegadores. Además se realizan como pruebas funcionales de aceptación de aplicaciones Web. Esta herramienta posee un ambiente de desarrollo llamado Selenium IDE, que facilita el registro de pruebas de aceptación incluyendo su depuración.

JTest: Esta herramienta se utiliza para la ejecución de pruebas funcionales, la cual permite ofrecer avances para las industrias de software. Estos avances se concentran en la realización de pruebas, para ayudar a los equipos a verificar automáticamente la funcionalidad de aplicaciones complejas. Para reducir la complejidad de las pruebas, Jtest ofrece la generación y la ejecución automatizada de los casos de prueba a través de la simulación de un entorno real, posibilitando una prueba en tiempo de ejecución. Permite detectar errores en el código que pueden pasar por inadvertidos durante etapas avanzadas en el desarrollo del software.

PHPUnit: Es una herramienta para la ejecución de pruebas al desarrollo de programas/sistemas implementados específicamente utilizando el lenguaje PHP.

JMeter: Herramienta desarrollada en el lenguaje java dentro del proyecto Jakarta. Permite la realización de pruebas de rendimiento y pruebas funcionales para aplicaciones web. El JMeter

posibilita generar a través de una navegación de usuario un caso de prueba. Su arquitectura ha evolucionado, no sólo para llevar a cabo pruebas en componentes habilitados en Internet (HTTP), sino además en bases de datos, requisiciones FTP, entre otras posibles pruebas. Es destacada por su versatilidad, estabilidad y por ser de uso gratuito. Es utilizada por el equipo de desarrollo de pruebas en la UCI ***(8).***

1.10 Estado Actual de las Pruebas de Software

Con el transcurso de los años se ha incrementado el control de la calidad en los productos de software, formando todo un proceso que se encuentra en constante cambio y perfeccionamiento. Aquellas aplicaciones que presenten fallas tanto en sus funcionalidades, su implementación como en el rendimiento o cualquier otro aspecto importante generan riesgos, amenazas y pérdidas al negocio, perjudicando de esta forma las empresas y clientes que interactúan en él. En el mundo existe insuficiencia en cuanto a la realización de pruebas a los productos de software, trayendo consigo falta de calidad en los mismos. Según estudios realizados de personas altamente calificadas, el 90 % de las aplicaciones no llevan a cabo un proceso de pruebas.

Esto se debe a la falta de recursos y la baja disponibilidad de tiempo. Cuba actualmente está prestando gran interés a este aspecto delicado e importante que afecta la industria del software y por ende frena el desarrollo científico técnico en la sociedad. Es una gran necesidad entrar al mercado mundial con productos que presenten máxima calidad, ya que las exigencias de la industria de software crecen cada día más. Muchas empresas ignoran el proceso de pruebas disminuyéndole la gran importancia que este posee. A causa de esto elaboran software básicamente improvisados sin una estructura organizativa y un control en cada uno de sus procesos, o incluso sin seguir una metodología de desarrollo de software.

Otro de los caso frecuentes en instituciones inmaduras del país es el poco rigor que imponen los representantes de cada institución, la falta de ética en darle continuidad a los procesos y a las fases del software. Otro punto clave que determina la eliminación de las pruebas de software es la irregularidad al darle seguimiento a un plan, el incumplimiento de acuerdos y requisitos propuestos por el cliente. Todo esto implica que los desarrolladores del proyecto sientan una presión demasiado fuerte, capaz de eliminar las revisiones y pruebas con el único objetivo de acortar el tiempo, pero sin importar la calidad del producto final.

Algunas cifras significativas que demuestran lo anteriormente planteado se exponen a continuación:

- El 90% de los proyectos no alcanzan los objetivos.

- El 40% fracasan por completo.
- El 50% no se entregan nunca.
- Gastos de adaptación tecnológica al año.
- Coste de demandas y litigios legales añadidos ***(9)*.**

La informática evoluciona continuamente a una velocidad impresionante, lo que exige un ritmo acelerado y una calidad excelente en las producciones. De esta forma se puede ganar un espacio en las grandes competencias internacionales. Las pruebas de software juegan un papel principal para darle cumplimiento a estos objetivos. Si se realizan de una forma eficiente y cautelosa se obtendrá como resultado el producto con calidad ideal, evitando las graves consecuencias que traería consigo el incumplimiento en los requisitos esperados.

1.11 Conclusiones

El análisis realizado en este capítulo ha brindado una panorámica de manera clara, explícita y elocuente de la importancia de la calidad del software. Se han mencionado distintos modelos de calidad, además de caracterizar las metodologías de desarrollo de software; aspecto sumamente importante para el funcionamiento de cualquier sistema. Posteriormente fueron fundamentados los principios y características de una buena prueba, además de argumentar las aplicaciones de los SIG, así como su funcionamiento y características en general. Los temas que se han abarcado como todo lo relacionado con el proceso de pruebas en un software, los niveles en los que se dividen las mismas, los métodos y técnicas que se desarrollan para su ejecución, además de conceptos y definiciones a los que se hacen referencia, posibilita nutrir de conocimiento y actualizar a todos los interesados en construir productos con un alto nivel de calidad.

CAPÍTULO 2: Diseño de la estrategia de prueba para la plataforma GENESIG y sus aplicativos.

2.1 Introducción

En el presente capítulo se describe a la plataforma GENESIG y a uno de sus aplicativos: SIGUCI. De estos productos se caracterizan su composición, requisitos funcionales, casos de uso, entre otros aspectos. Se muestra además la estrategia a seguir para la ejecución de las pruebas, donde se especifica su objetivo, las fases que la componen, las actividades que integran cada fase, los involucrados y los entregables. Además se precisa un plan de pruebas como base, para todo el proceso que se llevará a cabo durante la investigación.

2.2 Plataforma soberana GENESIG

GENESIG es una herramienta interoperable con soluciones existentes para el montaje de SIG. Tiene integración con software de gestión para la toma de decisiones. Presenta una estructura modular con requisitos funcionales mínimos que consta con más de 50 funcionalidades incorporadas.
El sistema está compuesto por 9 módulos:
Navegación, Selección, Consulta espacial, Análisis, Servicio, Consultas, Estructura y composición, Catálogo de mapas y Configuración.

2.2.1 Tecnologías utilizadas

Las tecnologías que fueron utilizadas para la realización de la Plataforma soberana GENESIG han alcanzado gran auge en la industria del software, debido a las grandes ventajas que poseen. La siguiente tabla muestra los lenguajes de programación, servidores y otras herramientas gestoras que se utilizaron para ejecutar las funciones de cada capa que componen la arquitectura de este sistema.

Lógica y negocio	Presentación	Datos	Servidores
PHP	Ext JS	Postgre SQL	Map server
Symfony	JAVA SCRIPT	PostGis	Apache

Tabla 1: Tecnologías y herramientas

2.3 Aplicativo SIGUCI

Para probar esta estrategia se toma como muestra el aplicativo SIGUCI, el cual surge como necesidad de contar con una herramienta que sirva de ayuda en la toma de decisiones en la UCI.

Está implementado con herramientas y tecnologías libres y su objetivo fundamental es realizar la representación geoespacial de la información asociada a la universidad, permitiendo además realizar un análisis sobre dicha información.

2.4 Características a probar.

Cuando se va a probar un software es necesario tener en cuenta varias características dependiendo de las condiciones del producto y además del objetivo que persiguen las pruebas a realizar. Luego de analizar las propiedades del aplicativo SIGUCI se determinó que teniendo en cuenta que el cliente pretende asegurarse que las funcionalidades en el sistema se realicen de manera correcta, se convierte en una función principal verificar que:

- Sea correcta la implementación de los requisitos funcionales.
- Cuente con una interfaz amigable, comprensible y factible para el usuario.
- Presente una documentación legible y entendible ante los usuarios.
- Tenga implementado correctamente mecanismos que garanticen la seguridad de la aplicación.
- El rendimiento y la carga de trabajo que puede soportar el sistema.

Pudiendo observar que el software a probar es un sistema de información geográfica es necesario verificar que además de las condiciones anteriores sea capaz de gestionar correctamente toda la información que almacena. Para esto es imprescindible que el sistema cuente con:

- Correcta entrada de la información.
- Eficiente almacenamiento y manipulación de la información.
- Calidad en la representación y salida de los datos georeferenciados.
- Válida utilización de la fuente cartográfica.

2.5 Plan de prueba

Con el plan de pruebas se pretende organizar cada una de las actividades que guiarán el proceso de pruebas. Para el aplicativo SIGUCI se ha definido un plan que contiene todos los elementos que serán probados. Se determina la estrategia de pruebas que se aplicará para este sistema, logrando un óptimo diseño de casos de prueba.

2.5.1 Recursos para la ejecución de las pruebas

Para el desarrollo de las pruebas es necesario contar con una serie de recursos tanto de hardware como de software los cuales serán mencionados a continuación:

Hardware:

Para las PCs clientes:

- Se requiere tengan tarjeta de red.
- Al menos 128 MB de memoria RAM.
- Se requiere al menos 100 MB de disco duro.
- Procesador 512 MHz como mínimo.

Para los servidores:

- Se requiere tarjeta de red.
- El Servidor de Mapas tenga como mínimo 2GB de RAM y 2GB de disco duro.
- El Servidor de Base de Datos tenga como mínimo 2GB de RAM y 10GB de disco duro.
- Procesador 3 GHz como mínimo.

Software:

Para las PCs clientes:

- Un Navegador como Mozilla Firefox, Zafari u otro navegador que cumpla con los estándares W3C.
- Sistema operativo: GNU/Linux y Windows.

Para los Servidores:

- Sistemas operativos GNU/Linux o Windows Server 2000 o superior.
- Servidor Web Apache 2.0 o superior, con módulo PHP 5 configurado con la extensión pgsql incluida.
- PostgreSQL como Sistema Gestor de Base de Datos.
- PostGis como extensión de PostgreSQL como soporte de datos espaciales.
- PgRouting como extensión de PostgreSQL para análisis de rutas.
- MapServer 5.2.2 o superior, con extensión PHP mapscript.

2.5.2 Requisitos funcionales.

Los requisitos planteados por los clientes en un inicio son los elementos fundamentales que se deben tener en cuenta a la hora de realizar los Casos de Prueba. El producto SIGUCI cuenta con las

siguientes funciones para la representación, modelación y análisis de la información geográfica en la UCI: Realizar navegación, Buscar persona, Localizar estructura, Buscar edificio, Mapa temático, Medir distancia, Calcular área, Exportar mapa.

2.5.3 Estrategia de pruebas

La estrategia de pruebas de software integra un conjunto de actividades que describen los pasos que se deben llevar a cabo en un proceso de pruebas. Está compuesta por su objetivo general, ocho fases, las actividades que se encuentran en cada una de las fases, los involucrados que intervienen y los entregables obtenidos como se muestra en la siguiente figura.

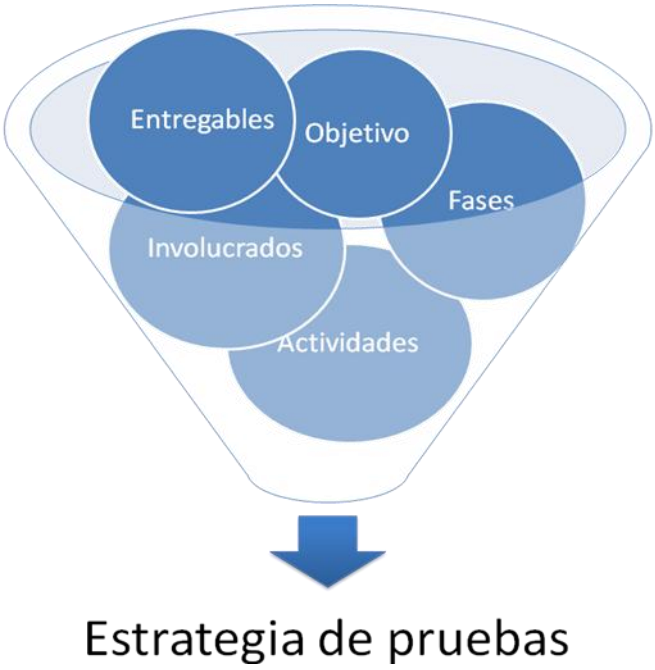


Figura3 Componentes de la estrategia de pruebas

Objetivo de la estrategia de pruebas.

El objetivo principal que persigue la estrategia de pruebas es establecer un procedimiento que permita organizar y planificar las actividades que se llevarán a cabo para lograr que los resultados de las pruebas sean satisfactorios.

Involucrados en la estrategia de pruebas.

Para la realización de las pruebas se llevó a cabo un proceso de planificación en el que participaron el equipo de desarrollo y el responsable de las pruebas. Estas fueron realizadas por un probador que fue el encargado de las actividades planificadas. Un integrante del equipo de desarrollo

estuvo presente durante todo el proceso, el cual veló por el correcto funcionamiento del sistema. Intervinieron además expertos en los temas que fueron estudiados, como usabilidad, cartografía, geología y pruebas de software.

Fases de la estrategia de pruebas.

Las fases que forman parte de la estrategia propuesta conforman un método de evaluación representado por listas de chequeos y casos de pruebas que permiten asegurar y garantizar la calidad durante la ejecución de las pruebas. Las mismas se muestran a continuación en la siguiente figura:

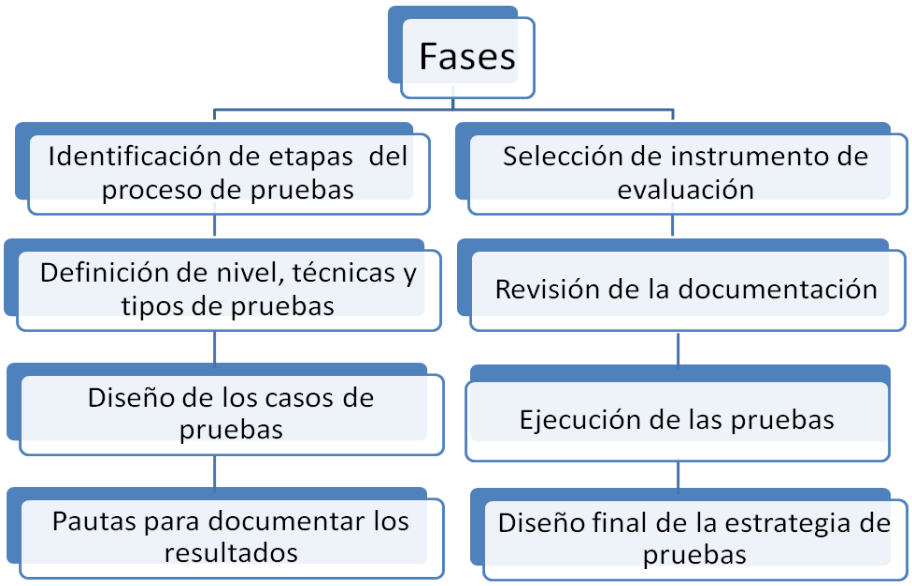


Figura 4 Fases de la estrategia de pruebas.

2.5.4 Identificación de etapas del proceso de pruebas.

Fase	Actividades
Identificación de etapas del proceso de pruebas 	1- Descripción de etapas en el proceso de pruebas
	2- Definición de iteraciones en el proceso de pruebas
	3- Descripción de iteraciones en el proceso de pruebas

Tabla 2 Actividades de la primera fase de la estrategia de pruebas

En esta fase se describen las etapas por las que transitará el proceso de pruebas. Las pruebas se realizarán por iteraciones. Cuando termine cada una de ellas se le hará entrega al equipo de desarrollo el documento de No Conformidades. Luego quedará establecido un plazo para solucionar los errores que fueron detectados para proceder al comienzo de otra iteración. Los casos de pruebas y las No Conformidades estarán almacenados en un servidor que se encuentra a disposición del equipo de desarrollo.

2.5.5 Definición de nivel, técnicas y métodos de pruebas.

Fase	Actividades
Definición de nivel, técnicas y métodos de pruebas 	1- Definir las características a probar en el producto
	2- Definir niveles de pruebas
	3- Definir técnicas de pruebas
	4- Definir tipos de pruebas en el nivel seleccionado

Tabla 3 Actividades de la segunda fase de la estrategia de pruebas

Las pruebas fueron aplicadas en el nivel de sistema, lo que permite probar el funcionamiento del producto SIGUCI en su totalidad, después de integrados sus módulos y a través de la interfaz de usuario. Se define este nivel debido a que se pretende probar la integridad y funcionalidad del sistema, tratando de encontrar la mayor cantidad de errores. Las técnicas de pruebas de dicho nivel permiten que el sistema se ejercite, además se comprueba la integración de la información y el funcionamiento de las interfaces.

Cada nivel de prueba contiene métodos y técnicas las cuales están diseñadas con el objetivo de encontrar posibles errores del software en su respectivo nivel. Dentro de los tipos de prueba que se realizan en el nivel de Sistema, las escogidas que se aplicaron al proyecto SIGUCI fueron las siguientes:

- **Pruebas exploratorias:** son las primeras que se realizan al sistema con el propósito de encontrar errores superficiales que puedan existir. Es una prueba sencilla, por lo que no se utilizan métodos para su ejecución. Esto no significa que sean menos importantes. Esta

prueba fue escogida ya que permite la detección de posibles errores que tal vez no sean identificados mediante otras pruebas.

- **Pruebas Funcionales:** se decidió realizar estas pruebas con el objetivo de verificar que el sistema tenga implementado adecuadamente cada uno de los requisitos establecidos por el cliente. Se escogió el método de Caja Negra, haciendo uso de la técnica de Partición de Equivalencia. Esta técnica se basa en una evaluación de las clases de equivalencia para una condición de entrada.
- **Pruebas de seguridad:** estas pruebas presentan un alto nivel de importancia, ya que todo sistema computacional que maneje información sensible constituye un blanco de entradas indebidas o ilegales al mismo. Otra de las razones principales por las que escogimos las pruebas de seguridad es debido a que nunca se podrá construir una aplicación segura sin realizarle este tipo de pruebas. Mediante ellas se puede verificar que el sistema tenga mecanismos incorporados que protejan la aplicación de accesos impropios de los usuarios. Para que estas pruebas logren su objetivo, requieren de tiempo y recursos suficientes que permitan una correcta ejecución. La comprobación de seguridad no garantiza que puedan comprobarse todas las formas en que un atacante puede colgar una aplicación, pero sí ha demostrado ser un factor clave para cualquier organización que necesita confiar en el software que usa o produce.
- **Pruebas de carga y estrés:** estas pruebas se ejecutan para poder conocer los límites de carga que soporta el sistema en una situación determinada. El aplicativo SIGUCI presenta grandes aplicaciones en la Universidad de las Ciencias Informáticas y gran cantidad de usuarios interactúan con el sistema por lo que se considera importante determinar hasta qué punto puede cargarse de trabajo el software sin que ocurran fallas.
- **Pruebas de usabilidad:** las pruebas de usabilidad permiten que un sistema sea funcionalmente correcto, eficiente, que para los usuarios sea fácil de aprender y recordar, además de presentar tolerancia a los errores. Estas pruebas fueron escogidas debido a que la usabilidad es un factor estratégico fundamental para conseguir el máximo aprovechamiento de recursos que brinda una aplicación. Un sistema usable reduce los costos de producción, mantenimiento y soporte, también es capaz de aumentar la productividad disminuyendo esfuerzos. El aplicativo SIGUCI contiene mucha información y si no está organizada, procesada y disponible para las personas en un formato que les permita tomar decisiones, se convertiría en un freno para el conocimiento y no en un beneficio.

Debido a que la plataforma GENESIG y sus aplicativos proveen las funciones y las herramientas necesarias para almacenar, analizar y desplegar la información geográfica, fue necesario realizar pruebas que permitan garantizar la correcta ejecución de las principales funcionalidades de los SIG. Por lo explicado anteriormente se redactó una lista de chequeo, la cual evalúa los siguientes parámetros que se muestran a continuación en el siguiente diagrama:

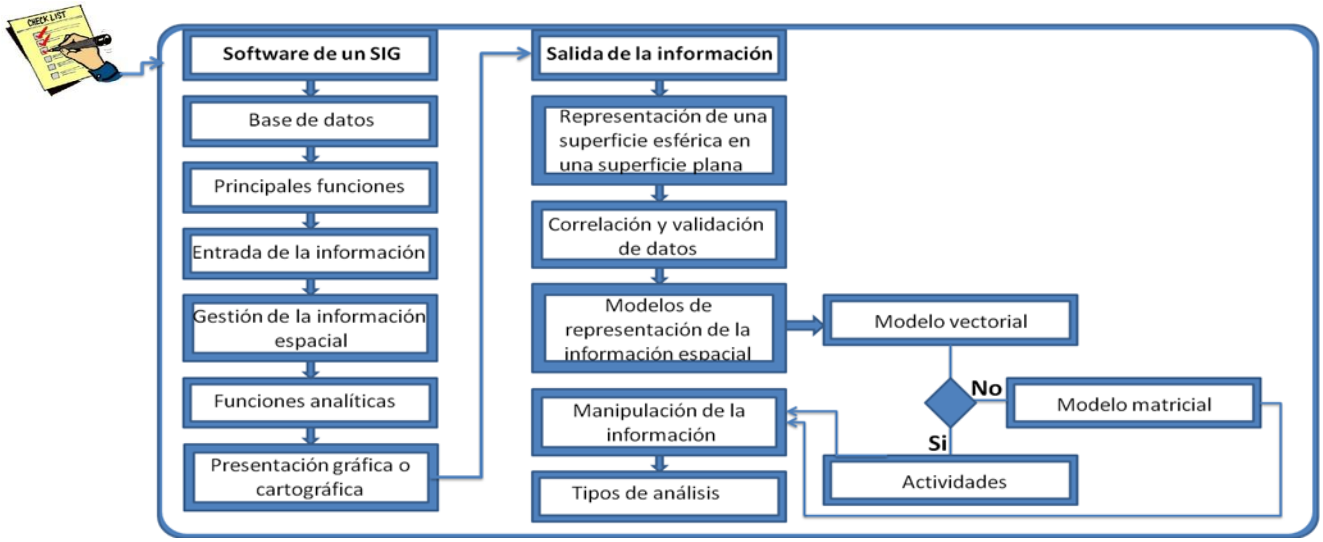


Figura 5 Diagrama de las Pruebas básicas de las principales funciones de un SIG

Como técnicas de diseño de casos de pruebas se escogió:

Pruebas de caja negra: esta técnica permite identificar errores de interfaz y funciones incorrectas. Las pruebas de caja negra están centradas en los requisitos funcionales del software. Como método de prueba de caja negra se escogió la técnica de partición equivalente, ya que permite que se pueda descubrir una clase de errores reduciendo el número total de casos de pruebas a realizar. Divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada.

2.5.6 Selección de instrumento de evaluación.

Fase	Actividades
Selección de instrumento de evaluación 	1- Seleccionar instrumento de evaluación.
	2- Establecer la cantidad de listas de chequeos que se utilizarán.
	3- Definir objetivos de las listas de chequeo.
	4- Establecer las variantes utilizadas para elaborar las listas de chequeos.
	5- Definir los elementos que conforman las listas de chequeos.

Tabla 4 Actividades de la tercera fase de la estrategia de pruebas

Como instrumento de evaluación se escogieron las listas de chequeo. Son un conjunto de preguntas que persiguen como objetivo verificar el cumplimiento de las políticas y reglas establecidas. Existen varios tipos de listas de chequeos, pueden ser breves o extensas y sencillas o complejas, todo depende del fin que éstas tengan. Algunas de las claves del éxito de una lista de chequeo, en cuanto a aceptación e incorporación para su uso, es que tenga estas características ***(12)***:

- Quien deba responderla la entienda fácilmente.
- No consuma mucho tiempo llenar el formulario, (solo el estrictamente necesario para responderla a conciencia).
- Quien debe leer las respuestas, también lo pueda hacer de una manera rápida y clara (evitando incluso el problema de redacciones confusas).

Se puede hacer uso de alguna Lista de Chequeo ya elaborada o se puede diseñar una propia ateniéndose a las peculiares necesidades de la organización. De ser posible, siempre es preferible elegir esta última opción, debido a que redactando la lista se puede contemplar de manera más precisa los intereses de quienes van a usarlas y las particularidades de la institución donde se va a aplicar ***(12).***

2.5.7 Revisión de la documentación.

Fase	Actividades
Revisión de la documentación 	1- Realizar reunión de apertura.
	2- Definir los documentos que van a ser revisados.
	3- Determinar los involucrados que participarán en la revisión.
	4- Realizar la revisión
	5- Elaborar la lista de recomendaciones.
	6- Realizar el informe final.
	7- Realizar el acta de reunión de cierre.

Tabla 5 Actividades de la cuarta fase de la estrategia de pruebas

La documentación es un aspecto crucial en un producto de software. Constituye un excelente medidor de la organización por parte del grupo de desarrollo y del producto en sí. Esta debe presentar las mismas exigencias que el software producido, ya que es un complemento del mismo. Es importante que la plataforma GENESIG y sus aplicativos cuenten con una documentación clara, precisa y completa. Por tales motivos es imprescindible realizar revisiones a toda la documentación del producto. Para ello se planificó el siguiente cronograma de actividades:

Actividad	Objetivo	Día	Hora	Lugar
Reunión de Apertura.	Aprobar objetivos, criterios y cronograma de la revisión.	15/03/11	9:00 am	Local del Proyecto
Revisión de la Documentación con el proyecto.	Reunir evidencias objetivas de la adherencia del proyecto a los criterios de evaluación.	15/03/11	9:30 am – 11:30 am	Local del Proyecto
Conciliación de los hallazgos con el equipo de proyecto.	Conciliar los hallazgos detectados con el equipo de proyecto.	15/03/11	11:30 pm – 12:30 pm	Local del Proyecto
Elaborar los documentos Recomendaciones e Informe Final de la revisión.	Preparar la documentación de la revisión	15/03/11	2:00 pm -3:00 pm	Local de Revisores
Reunión de Cierre de la Revisión.	Presentar los hallazgos y el informe de la revisión	15/03/11	3:30 pm	Local del Proyecto

Tabla 6: Cronograma de actividades

Para el proceso de revisión de la documentación se utilizó una computadora, la documentación y el registro de no conformidades.

2.5.8 Diseño de los casos de pruebas.

Fase	Actividades
Diseño de los casos de pruebas 	1- Definir la planilla de casos de prueba a utilizar
	2- Definir los documentos que van a ser revisados

Tabla 7 Actividades de la quinta fase de la estrategia de pruebas

Los casos de pruebas se diseñaron sobre una planilla definida por el centro de calidad para soluciones tecnológicas (CALISOFT) de la UCI. Presenta un modo estándar, para que pueda ser utilizado en todos los proyectos producidos en la misma. Estas planillas presentan modificaciones con el objetivo de obtener un mejor diseño de los casos de prueba y así se garantiza que se detecte la mayor cantidad de errores en los productos. La planilla utilizada para estos casos de prueba fue la propuesta para el programa de mejora que se lleva a cabo actualmente en la universidad. A continuación se mostrarán los casos de pruebas diseñados correspondientes al proyecto SIGUCI.

Diseño de casos de pruebas de caja negra.

Con el objetivo de lograr una mejor identificación se les nombró a los casos de pruebas del mismo modo que los casos de usos. A continuación se muestran 3 diseños de casos de pruebas correspondientes a 3 casos de uso.

DCP_Buscar edificio.

Descripción General.

El caso de uso se inicia cuando el usuario desea localizar un edificio, para ello deberá introducir el número del mismo y termina cuando el sistema realiza la operación.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central
Buscar edificio.	EC 1.1: Buscar edificio correctamente.	Esta funcionalidad permite buscar un edificio seleccionado por el usuario, además muestra los datos generales del mismo y lo ubica en el mapa.	• El usuario selecciona la opción: Buscar edificio. • El sistema muestra una interfaz con los campos necesarios para la búsqueda. • El usuario introduce los datos. • El usuario presiona el botón: buscar. • El sistema muestra los datos del edificio y señala en el mapa su ubicación.
	EC 1.2: Buscar edificio con error en la entrada de datos.	El sistema no muestra los resultados ya que el usuario no introdujo el número de edificio.	• El usuario selecciona la opción: Buscar edificio. • El sistema muestra una interfaz con los campos necesarios para la búsqueda. • El usuario no introduce el número de edificio. • El sistema mantiene desactivado el
	EC 1.3: Ver detalles de un edificio.	Esta funcionalidad permite mostrar al usuario detalles de la información relacionada con el edificio seleccionado.	• El usuario selecciona la opción: Buscar edificio. • El sistema muestra una interfaz con los campos necesarios para la búsqueda. • El usuario introduce los datos. • El usuario presiona el botón: buscar. • El usuario presiona la opción: ver detalles. • El sistema muestra una ventana con la información de todos los apartamentos de dicho edificio.
	EC 1.4: Cerrar ventana Edificio	Esta funcionalidad permite que se cancele la búsqueda de un edificio.	• El usuario selecciona la opción: Buscar edificio. • El sistema muestra una interfaz con los campos necesarios para la búsqueda. • El usuario introduce los datos. • El usuario presiona el botón: buscar. • El usuario cierra la ventana Edificios • El sistema cancela la Búsqueda de

			personas.

Tabla 8: Secciones a probar en el CU: Buscar edificio

No	Num_edificio	Clasificación	Valor Nulo	Descripción
1	*Numero del edificio*	*Campo de texto*	*No*	*Solo deben introducirse números.*

Tabla 9: Descripción de variables del CU Buscar edificio

Diseño de casos de pruebas de Carga

CU: Realizar navegación

ID del escenario	Escenarios de la sección	Carga de Trabajo	Descripción	Resultado esperado	Resultado de la prueba
1	EC 1.1: Acercar	50 conexiones.	Escenario para acercar una región del mapa seleccionada por el usuario.	El sistema procesa la información a partir de la acción realizada por el usuario y actualiza la visualización del mapa con un buen rendimiento sin errores.	Satisfactorio
2	EC 1.2: Alejar	50 conexiones.	Escenario para alejar una región del mapa seleccionada por el usuario.	El sistema realiza la operación de Alejar seleccionada por el usuario disminuyendo el tamaño y aumentando la escala puntualmente con un buen rendimiento sin errores.	Satisfactorio
3	EC 1.3: Recentrar Mapa	50 conexiones.	Escenario para recentrar el mapa.	El sistema obtiene las coordenadas (x, y) del punto donde el usuario hizo clic y lo traslada al centro de la pantalla sin cambiar la escala del mapa con un buen	No Satisfactorio

				rendimiento sin errores.	
4	EC 1.4: Ver Todo	50 conexiones.	Escenario para visualizar todo el mapa.	El sistema visualiza el mapa según el extend inicial de la aplicación con un buen rendimiento sin errores.	Satisfactorio
5	EC 1.5: Mover el mapa	50 conexiones.	Escenario para mover el mapa.	El sistema permite obtener un nuevo centro de pantalla a donde se movería el mapa con un buen rendimiento sin errores.	No Satisfactorio
6	EC 1.6: Anterior	50 conexiones.	Escenario para visualizar el mapa anterior.	El sistema comprueba que tenga una visualización anterior y muestra el mapa anterior con un buen rendimiento sin errores.	Satisfactorio
7	EC 1.7.: Siguiente.	50 conexiones.	Escenario para visualizar el mapa siguiente.	El sistema comprueba que tenga una visualización siguiente y muestra el mapa siguiente con un buen rendimiento sin errores.	Satisfactorio
8	EC 1.8.: Navegar a través del Mapa de Referencia.	50 conexiones.	Escenario para navegar a través del mapa de referencia.	Permite mover al centro de la pantalla el punto seleccionado sin modificar la escala con un buen rendimiento sin errores.	Satisfactorio

Tabla 10 DCP de carga para el CU: Realizar Navegación

Diseño de casos de pruebas de estrés.

CU: Mapa temático

ID del escenario	Escenarios de la sección	Carga de Trabajo	Descripción	Resultado esperado	Resultado de la prueba
1	EC 1.1: Corocromático y coropleta	150 conexiones.	Escenario para realizar una tematización corocromática y coropleta	El sistema muestra la tematización especificada con un buen rendimiento sin	No Satisfactorio
2	EC 1.2: Gráficas dinámicas	150 conexiones.	Escenario para realizar una tematización de Grñaficas dinámicas.	El sistema muestra la tematización especificada con un buen rendimiento sin	No Satisfactorio
3	EC 1.2: Símbolo proporcional	150 conexiones.	Escenario para realizar una tematización de Símbolo proporcional.	El sistema muestra la tematización especificada con un buen	No Satisfactorio

Tabla 11 DCP de estrés para el CU: Mapa temático

2.5.9 Ejecución de las pruebas.

Esta fase es de gran importancia y constituye la esencia de la estrategia de pruebas. Aquí es donde se aplican las pruebas que fueron escogidas anteriormente al aplicativo SIGUCI tomado como muestra. Cada una de las actividades se describirá a continuación.

Fase	Actividades
Ejecución de las pruebas 	1- Realizar pruebas exploratorias
	2- Realizar pruebas funcionales
	3- Realizar pruebas de usabilidad
	4- Realizar pruebas de carga y estrés
	5- Realizar pruebas de seguridad
	6- Realizar pruebas de elementos básicos de un SIG

Tabla 12 Actividades de la sexta fase de la estrategia de pruebas

Pruebas exploratorias

Para la realización de las pruebas exploratorias se tuvieron en cuenta algunos aspectos de la usabilidad con el objetivo de hacer más factible el resultado de estas pruebas. El probador que llevó a cabo esta actividad tuvo una preparación por parte del líder del proyecto sobre el sistema. Además participó en talleres sobre los SIG, pruebas, revisiones y aplicaciones web. El procedimiento de esta prueba se llevó a cabo en un día. Estas pruebas fueron desarrolladas mediante una interacción exploratoria del probador con el sistema. Las no conformidades encontradas fueron registradas en una lista de recomendaciones.

Pruebas funcionales

Las pruebas funcionales que se le realizaron a SIGUCI fueron pruebas de Caja negra. Un probador y el jefe de proyecto fueron los protagonistas de la actividad. Estas pruebas fueron basadas en los diseños de casos de prueba que se explicaron con anterioridad, donde se recogen los escenarios correspondientes a los casos de uso del sistema y las clases de datos utilizadas para probar estos escenarios.

Para la realización de estas pruebas se ejecutó cada caso de uso, flujo de caso de uso o función usando datos válidos e inválidos. Se eligieron correctamente los valores en las entradas abarcando el máximo número de combinaciones de datos de entrada sin que el número de casos de pruebas sea muy elevado. Se realizaron 2 iteraciones, cada una tuvo una duración de 7 días. Mediante estas pruebas se verificaron todos los requisitos especificados por el cliente.

Pruebas de Usabilidad

Las pruebas de usabilidad fueron ejecutadas con usuarios reales, a los cuales se les presentaron un conjunto de tareas a realizar con el prototipo del sistema. Estas tareas fueron basadas en una lista de chequeo con interrogantes relacionadas con los factores de usabilidad en aplicaciones web. Los comentarios de los usuarios que participaron fueron recogidos para optimizar el resultado de dicha prueba. Como complemento a esta prueba se realizó la evaluación de un experto en usabilidad. El mismo proporcionó una información completamente diferente a la que se obtuvo de los usuarios finales, pues es más viable y precisa.

Como parte del procedimiento que se llevó a cabo se le explicó al experto el modo de funcionamiento del sistema, además de entrenarlo en las funciones principales. Se le informa acerca del dominio de aplicación y el experto revisa el sistema. Una vez finalizada la revisión el experto elabora un informe con los problemas identificados y sugerencias. Este informe fue comparado y analizado junto al que se obtuvo como resultado de las acciones de los usuarios. De esta forma se

elaboró un documento final que recogía todas las no conformidades detectadas en el transcurso de la ejecución de las pruebas.

Pruebas de Carga.

Las pruebas de carga fueron realizadas usando la herramienta Jmeter. Es una herramienta automatizada para aplicaciones web muy versátil, capaz de simular usuarios, ahorrar recursos y tiempo, es gratis y libre. Esto condujo a tener en cuenta determinados requisitos para que los resultados sean satisfactorios.

Sobre el aplicativo SIGUCI que fue la aplicación sometida a las pruebas fue necesario conocer en cuanto al software:

- Datos de software de la aplicación.
- Tipo de conexión y velocidad.

Sobre hardware:

- Datos sobre la velocidad y la capacidad del servidor de la aplicación.
- Distancia del servidor al laboratorio de pruebas.

Este proceso consta de dos elementos fundamentales:

- Grabación de los escenarios de pruebas.
- Confección de los planes de pruebas.

Para la realización de la grabación de los escenarios de prueba es necesario configurar el navegador Firefox con el proxy definido en el JMeter, y luego se pone a correr la aplicación realizando las tareas en ella que se desea que sean grabadas por el programa para su posterior simulación. De esta forma queda grabado el escenario de prueba. Para la confección de los planes de prueba se utilizan los elementos definidos en el procedimiento.

Para interpretar los resultados del JMeter se debe conocer:

- Cantidad de usuarios concurrentes que necesita soportar la aplicación.
- Tiempo de respuesta en situaciones de carga de una transacción.

Además el proyecto en el documento de los casos de prueba debe describir los escenarios significativos que serán sometidos a la prueba. Dígase camino básico a realizar para llegar hasta esos escenarios. En este caso se deben mostrar los lugares más críticos de la aplicación y crear flujos describiendo toda la funcionalidad.

Pruebas de Seguridad

Las pruebas de seguridad al producto SIGUCI fueron basadas en la guía de OWASP (Proyecto Abierto de Seguridad para Aplicaciones Web). Se utilizó esta guía ya que permite mejorar las habilidades para probar. Además varias de las pruebas aquí descritas no son complicadas ni requieren habilidades especiales o herramientas. Con OWASP se pueden expandir los casos de comprobación aplicados a los programas. También se detectan vulnerabilidades ahorrando tiempo y recursos considerables.

A la aplicación SIGUCI se le realizaron pruebas de intrusión, las mismas son las encargadas de comprobar la seguridad de una red. Esta técnica es el arte de comprobar un sistema en ejecución, sin conocer su funcionamiento interno, con el objetivo de encontrar vulnerabilidades de seguridad. Para el desarrollo de estas pruebas se realizó un análisis activo de la aplicación en busca de cualquier debilidad, fallos técnicos o vulnerabilidades. Las incidencias de seguridad encontradas fueron recogidas en un documento junto con una propuesta para su mitigación o una solución técnica. Las pruebas que se le aplicaron a SIGUCI fueron divididas en 4 subcategorías.

Pruebas de Autenticación.

- Transmisión de credenciales a través de un canal cifrado (OWASP-AT-001)

  Se comprobó que los datos de autenticación del usuario sean transferidos a través de un canal cifrado para evitar que sean interceptados por usuarios maliciosos. Con esto se pretende entender si los datos viajan no cifrados desde el servidor web al servidor o si la aplicación utiliza los mecanismos de seguridad apropiados.

- Pruebas de diccionario sobre cuentas de Usuario o cuentas predeterminadas (OWASP-AT-003).

  Se realizaron varias combinaciones de nombre de usuario/contraseña predeterminadas y que pueden ser empleadas para obtener acceso a la infraestructura de la red interna para obtener privilegios y robar datos. Esta comprobación se realizó con una herramienta automatizada llamada Brutus. Se elaboraron dos documentos de textos con los posibles usuarios y contraseñas, los cuales fueron cargados por esta herramienta. De esta forma se comprobó cualquier posible entrada al sistema con estos usuarios y contraseñas.

Gestión de Sesiones:

Primeramente se comprobó la presencia de la función de cierre de la sesión, comprobando que la aplicación proporciona un botón de cierre de sesión y que esté presente en las páginas que requieran autenticación. Mediante esta prueba también se comprueba lo que le ocurre a los testigos de sesión cuando la función de cierre es invocada. Se verificó que las cookies fueran borradas.

También se realizó la prueba de cierre de sesión por tiempo expirado. Se comprobó si existe un tiempo de expiración registrando un usuario en la aplicación y esperar un tiempo determinado para ver si se dispara el cierre de sesión por tiempo. Luego que haya pasado este tiempo todos los testigos de sesión deberán ser destruidos.

Pruebas de elementos básicos de un SIG.

Estas pruebas fueron ejecutadas directamente a la aplicación. Estuvieron presentes un implementador del sistema, el analista principal, un diseñador de base de datos y el jefe de proyecto junto al probador encargado de la actividad. Se recibió con anterioridad una capacitación sobre las características fundamentales, las funciones, la composición y otras informaciones generales de los SIG. Mediante la lista de chequeo elaborada se le dio respuesta a cada una de las interrogantes planteadas. Se identificaron las deficiencias e inconformidades en un documento, que posteriormente fue revisado y validado por expertos en el tema.

2.5.10 Pautas para documentar los resultados.

Fase	Actividades
Pautas para documentar los resultados 	1- Definir una vía para documentar los resultados
	2- Definir los elementos que componen el documento de no conformidades
	3- Realizar el control de versiones

Tabla 13 Actividades de la séptima fase de la estrategia de pruebas

La plantilla de No Conformidades es el documento que recoge los errores que son detectados durante la realización y aplicación de las pruebas del software, además de la revisión de la documentación del sistema. Es necesario elaborar un documento por cada iteración que se lleve a cabo, controlándose a través de versiones, en las cuales se van eliminando los errores. Finalmente quedarán erradicados todos los defectos que presente el elemento al que se le realizó la prueba.

Los casos de prueba cuentan con una sección dedicada a documentar los errores que fueron encontrados en la funcionalidad que se esté probando, pero la planilla de No conformidades registra los defectos generales de todas las funcionalidades que se probaron en el sistema y finalmente ésta se les entrega oficialmente al equipo de desarrollo.

Entregables

Los entregables y artefactos que quedaron establecidos son: Plan de Pruebas, Cronograma de actividades, Plantilla de diseño de casos de pruebas, Plantilla de no conformidades, Lista de recomendaciones y Listas de chequeo.

2.5.11 Diseño final de la estrategia de pruebas.

Todo lo anteriormente presentado soporta el diseño de la estrategia de prueba. Esta consiste en un conjunto de acciones organizadas y secuenciales que guían el proceso de pruebas. A continuación se presentan las actividades con su descripción, participantes y entregables, llevadas a cabo para el diseño de la estrategia de pruebas para la plataforma GENESIG y sus aplicativos.

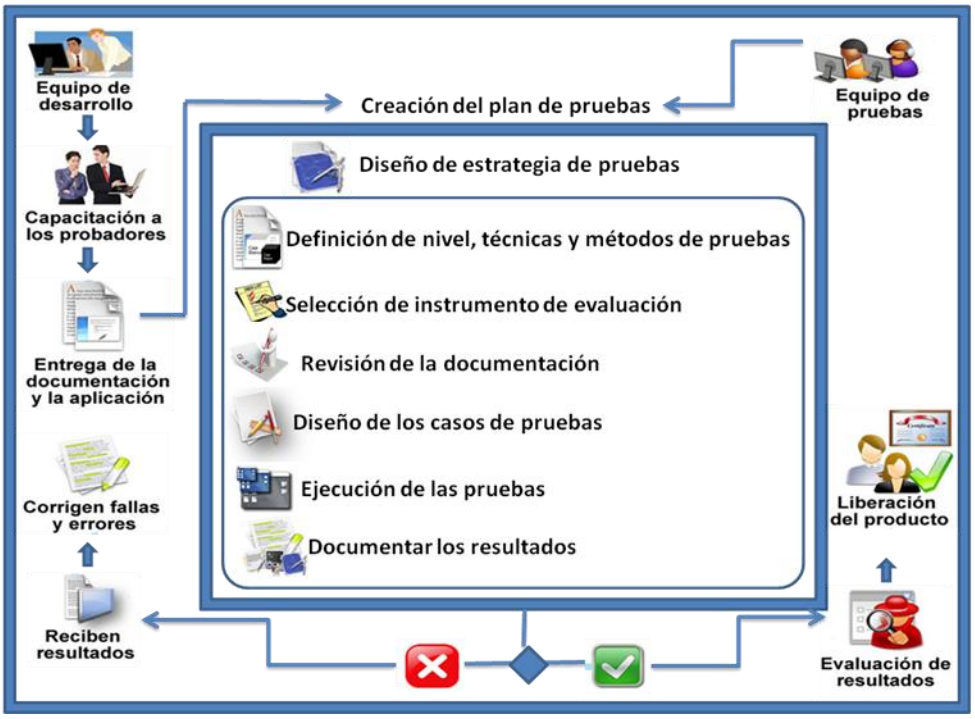


Figura 6 Diseño final de la estrategia de pruebas

2.6 Conclusiones

En este capítulo fueron detalladas las características de la Plataforma soberana de desarrollo GENESIG y de uno de sus aplicativos: SIGUCI. Además todo el proceso de pruebas al cual fue sometido. Se definió la metodología de prueba que guiaron las actividades realizadas. Se mostró la estrategia que se llevó a cabo para la ejecución de las pruebas, y fueron realizados los artefactos que

estaban definidos como entregables. Se muestra cómo se diseñaron los casos de pruebas aplicados al sistema, teniendo en cuenta la premisa que con un correcto diseño de casos de prueba, se podrán encontrar la mayor cantidad de errores, posibilitando esto la calidad del producto.

Capítulo 3: Resultados obtenidos de la ejecución de las pruebas.

3.1 Introducción

La ejecución del proceso de prueba a la aplicación SIGUCI produjo la detección de fallas y errores que opacan la calidad de dicho software. Las mismas fueron registradas en el documento de no conformidades y posteriormente fue entregado al equipo de desarrollo, quien es el encargado de darles solución. Para que un proceso de pruebas tenga éxito es necesario un análisis final de los resultados arrojados.

Es imprescindible la evaluación del producto que ese está probando teniendo en cuenta todos los defectos encontrados durante el proceso. El presente capítulo se encarga de analizar las no conformidades encontradas en las distintas pruebas realizadas al aplicativo SIGUCI, además se brinda una valoración del producto general tomando como base los parámetros de calidad siguientes: seguridad, portabilidad, usabilidad y confidencialidad; utilizando para ello una lista de chequeo.

3.2 Resultado de las pruebas aplicadas a SIGUCI

3.2.1 Pruebas exploratorias

Con la aplicación de estas pruebas se encontraron 7 no conformidades que no habían sido detectadas, lo que ratifica su importancia. La lista de recomendaciones **(Anexo 1)** fue entregada al equipo de desarrollo y se estableció como plazo para erradicar las no conformidades un tiempo estimado de 72 horas.

Los principales problemas encontrados fueron:

- La aplicación no se muestra en el navegador: Internet explorer.
- Existen espacios en blanco en algunas interfaces.
- No se identifica al usuario cuando se encuentra en la aplicación.

3.2.2 Pruebas de Caja negra.

Las pruebas funcionales al sistema se realizaron en dos iteraciones como se especificó en capítulos anteriores, para verificar que las funcionalidades implementadas fueron las acordadas con el cliente y que respondían correctamente a sus necesidades. La siguiente tabla muestra el resumen de los defectos encontrados como resultado de estas pruebas.

Aplicación	Iteración	No conformidades	Significativas	No significativas	Recomendaciones
	1	7	6	1	4
	2	5	4	1	4
Total	**2**	**12**	**10**	**2**	**8**

Tabla 14 Resumen estadístico de las no conformidades en las pruebas funcionales

De las no conformidades **(Anexo 2)** encontradas las que están clasificadas como significativas están relacionadas con problemas de funcionalidad del sistema, que en ocasiones no coincidían con las descripciones de los casos de uso a los que responden. Los errores clasificados como no significativos están relacionados con el idioma, principalmente textos o mensajes de errores. Las recomendaciones que fueron redactadas perseguían como objetivo principal proponer mejoras para el sistema y prevenir la ocurrencia de fallas en un futuro. Una vez que el probador le hizo entrega al equipo de desarrollo los defectos encontrados en una iteración contaban con 72 horas para resolverlos.

Los principales problemas que se detectaron durante la aplicación de estas pruebas son:

- No se muestran correctamente todos los datos del edificio que busca el usuario.
- No se encuentran las imágenes en las descripciones de algunas entidades.
- La interfaz correspondiente a la sección “Mapa temático” no se muestra en la aplicación, por lo que no se puede probar ningún escenario de esta sección.
- Algunos errores que se muestran para especificar los valores que el usuario debe entrar son descritos en inglés.

3.2.3 Pruebas de usabilidad.

Las pruebas de usabilidad permitieron detectar fallas e ineficiencias en la aplicación. En la tabla que se muestra a continuación se encuentra un resumen de las no conformidades encontradas durante la realización de estas pruebas.

Aplicación	Iteraciones	No conformidades	Impacto Alto	Impacto Medio	Impacto Baja	Recomendaciones
	1	24	20	4	0	24
	2	19	18	1	0	19

Tabla 15 Resumen estadístico de las no conformidades en las pruebas de usabilidad

Los principales problemas que se encontraron se especifican a continuación.

- No cuenta con una ayuda para los usuarios que interactúan con el sistema.
- El sitio no presenta contactos para conocer a los administradores del mismo.
- Existen mensajes de error en inglés.
- La aplicación no contiene la fecha.
- No se brinda la opción de aumentar o disminuir el tamaño de letra.
- El sitio no presenta enlaces.
- Contiene demasiados elementos en el menú de navegación.

3.2.4 Pruebas de Carga y estrés.

Como se especificó en el capítulo anterior para las pruebas de carga se utilizó la herramienta JMeter. Los resultados arrojados durante esta prueba se presentan a continuación.

Funcionalidades	Número de usuarios	% de error	rendimiento	Kb/sec	Uso del CPU	Memoria física
Autenticación del usuario	(carga) 50	4.58	52.9	5113.9	50%	60%
	(estrés) 150	100	48.4	475	50%	60%
Buscar edificio	(carga) 50	45.45	62.1	50.3	55%	60%
	(estrés) 150	100	5.5	28.5	50%	64%
Buscar persona	(carga) 50	100	564.8	5889.3	60%	70%
	(estrés) 150	100	7.8	195.6	45%	55%
Calcular área	(carga) 50	6.41	38.5	349.7	52%	60%
	(estrés) 150	100	12.5	264.1	45%	50%
Exportar mapa	(carga) 50	6.37	36.9	333.8	45%	50%
	(estrés) 150	100	4.2	85.4	50%	60%
Localizar estructura	(carga) 50	7.10	23.1	282.3	50%	60%
	(estrés) 150	100	9.0	170.5	50%	60%
Mapa temático	(carga) 50	2.56	455.9	4695.8	50%	60%
	(estrés) 150	100	7.7	183.4	50%	60%
Medir distancia	(carga) 50	40.0	101.0	61.6	45%	55%
	(estrés) 150	100	6.4	164.8	50%	60%
Realizar navegación	(carga) 50	17.68	24.0	173.6	50%	60%

	(estrés) 150	100	7.7	223.7	50%	60%

Tabla 16 Resultados de las pruebas de carga y estrés

Durante las pruebas de carga, analizando los valores de la cantidad de peticiones y rendimiento se puede concluir que el sistema responde satisfactoriamente. La prueba se realizó con 50 usuarios concurrentes y mantuvo un rendimiento estable. Los resultados obtenidos demuestran que las funcionalidades cumplen con los requerimientos establecidos exceptuando la funcionalidad: Buscar persona, la cual alcanzó un 100% de error con un rendimiento desfavorable.

La prueba de estrés fue realizada con el objetivo de conectar 150 usuarios concurrentes, para esto fue llevada a cabo con 15 usuarios que se conectaban cada 5 segundos en un ciclo que se repite 10 veces, intentando lograr 150 usuarios en el último ciclo. Al culminar la prueba se pudo concluir que el rendimiento del sistema no fue estable, alcanzando valores inaceptables y posibilitando un lento avance de la aplicación que tardó 2 minutos en reponerse.

Por todo lo anterior el sistema soporta hasta 100 usuarios concurrentes presentando un buen rendimiento y un tiempo de respuesta dentro de los parámetros normales. Un valor por encima de este no se acepta. Es necesario que se realice una revisión y verificación de los vínculos que se muestran a continuación, los cuales dan error, ya que presentan un tiempo de respuesta inaceptable por encima de los valores esperados.

links con errores	**Comentarios**
/sig_uci/htdocs/index.php	Error en la página de inicio.
/sig_uci/htdocs/generated/images/4dc96a13_725_2f.png	Error de imagen
/sig_uci/htdocs/generated/images/4dcc02b2_1c2d_1.png	Error de imagen
/sig_uci/htdocs/generated/images/4dcc02b2_1c2d_0.png	Error de imagen
/sig_uci/htdocs/generated/images/4dcc02b2_5b0_18.png	Error de imagen
/sig_uci/htdocs/generated/images/4dcc02b2_5b0_19.png	Error de imagen
/sig_uci/htdocs/generated/images/4dcc0eea_25d3_4.png	Error de imagen

/sig_uci/htdocs/generated/images/4dcc0eeb_25d3_7.png	Error de imagen
/sig_uci/htdocs/generated/images/4dcc0eeb_25d3_6.png	Error de imagen
Error de imagen	Error de imagen
/safebrowsing/downloads?client=navclient-auto	Error en la descarga de una imagen.
/sig_uci/htdocs/generated/images/4dcc5a58_6249_13.png	Error de imagen
/sig_uci/htdocs/geoweb/plugins/buscarPersona/htdocs/gfx/tb9a.png	Error de imagen
/sig_uci/htdocs/images/default/tree/folder.gif	Error de imagen
/sig_uci/htdocs/geoweb/plugins/medirDistancia/htdocs/gfx/tb3e.png	Error de imagen

Tabla 17 Vínculos con errores en la aplicación

De forma general los problemas principales encontrados durante la aplicación de estas pruebas son:

Errores en vínculos de la aplicación:

Se encuentran errores al 100% en los vínculos que se mencionaron anteriormente, pues no responden a ninguna petición, logrando retrasos en la respuesta del rendimiento del sistema.

Errores en el sistema por la carga en la herramienta:

El sistema presentó errores de respuesta debido a que la herramienta JMeter muestra grandes niveles de estrés cuando se sobrecarga la concurrencia, lo que obliga al uso de la misma en computadoras con memorias RAM altas y de gran velocidad.

Problemas de resistencia de la Base de Datos.

En ocasiones cuando se realizaban pruebas de estrés, l base de datos dejaba de responder por la cantidad de peticiones que recibía simultáneamente. Esto provocó la caída de la aplicación y fallas en las pruebas por demasiados errores.

Problemas del navegador seleccionado.

Las pruebas realizadas con el navegador Firefox presentan un por ciento de errores superior a cuando son realizadas con el internet explorer, ya que ocurren errores en elementos del mismo que no pueden ser reconocidos cuando se le da respuesta a las peticiones.

A continuación se muestran los resultados obtenidos en las pruebas de carga y estrés.

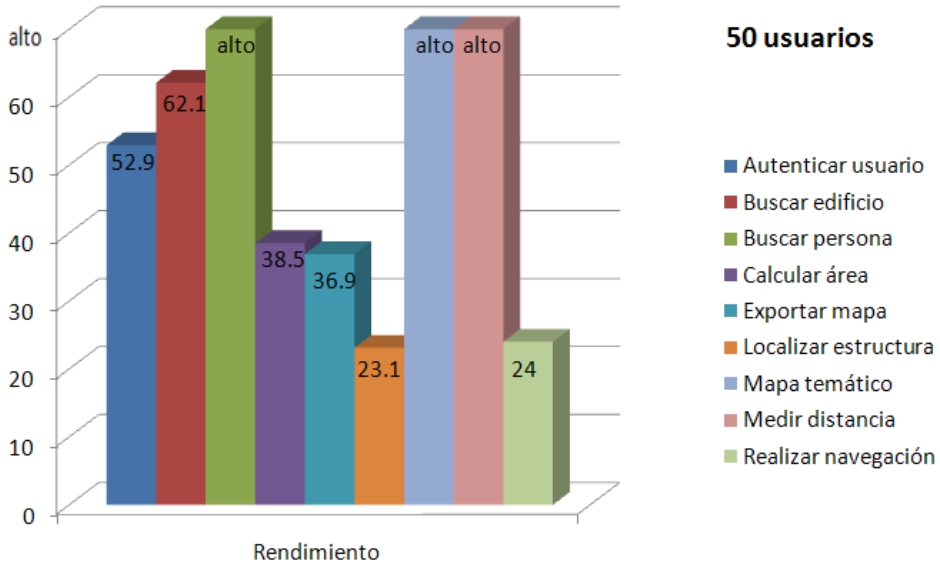


Figura 7 Valores de rendimiento en las pruebas de carga

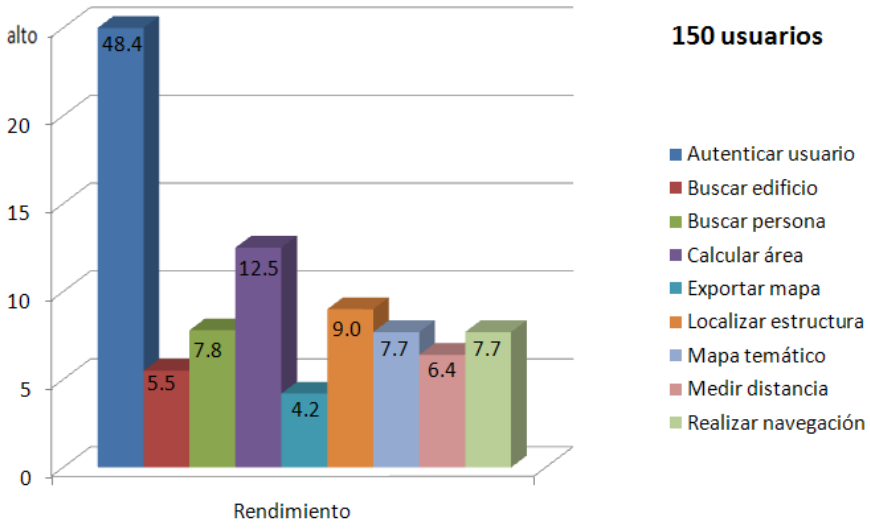


Figura 8 Valores de rendimiento en las pruebas de estrés

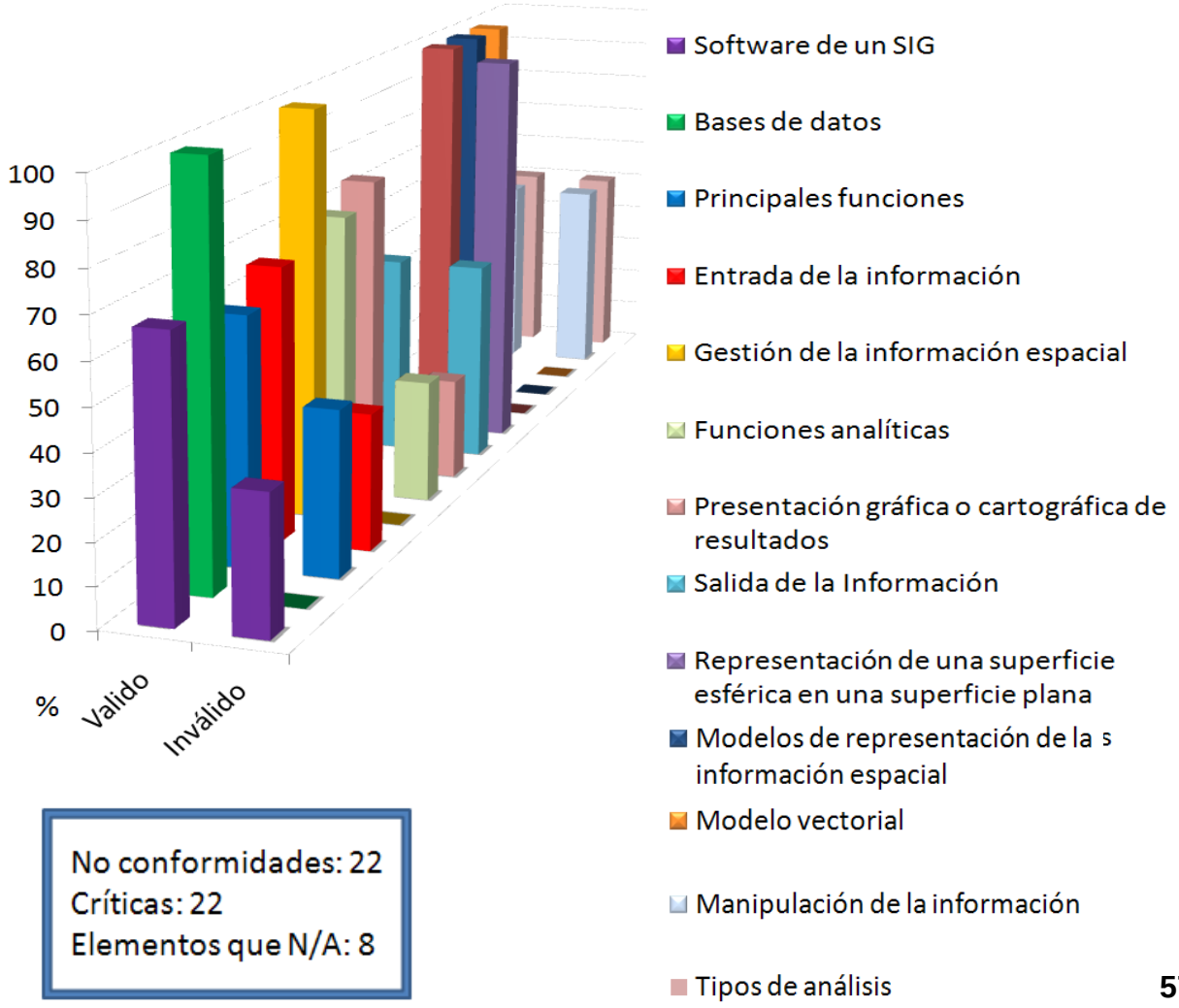
3.2.5 Pruebas de seguridad.

Mediante las pruebas de Transmisión de credenciales a través de un canal cifrado se pudo detectar que el aplicativo SIGUCI no utiliza un protocolo seguro para transmitir los datos cifrados, lo cual da paso a que no estén protegidos por algún atacante. Las pruebas de diccionario sobre cuentas de Usuario o cuentas predeterminadas permitieron detectar que la aplicación no es sensible a entradas de usuarios.

De los usuarios y contraseñas que se encontraban en los archivos cargados por la herramienta Brutus ninguno de ellos fueron capaces de entrar al sistema. Esto demuestra que no existen deficiencias en el control de la entrada de usuarios al sistema. Al realizarse las pruebas de gestión del Caché de Navegación y de salida de sesión se pudo detectar una falla grave en la aplicación. No cuenta con un botón de cierre de sesión, por lo que las cookies no son eliminadas y muestran datos importantes del usuario. Tampoco presenta un tiempo de expiración de sesión, esto contribuye aún más a la presencia de inseguridad en este sistema.

3.2.6 Pruebas de elementos básicos de un SIG.

Al aplicar la lista de chequeo expuesta en el capítulo anterior, la cual permite verificar la correcta implementación de las principales funciones que deben estar presentes en un SIG, se pudo detectar una gran deficiencia de estos elementos en el SIGUCI. Se realizó una sola iteración de esta prueba, donde se recogieron 22 no conformidades. A continuación se muestra una gráfica que expresa los resultados alcanzados.



Como se puede apreciar en la figura anterior, los elementos: Base de Datos, Gestión de la información espacial, Correlación y validación de datos, Modelos de representación de la información espacial y Modelo vectorial fueron evaluados con el 100% de validez. Esto significa que el sistema cuenta con una correcta implementación de estas funcionalidades, ya que no se encontraron no conformidades al respecto.

Además se puede observar que en cuanto a la Salida de la información, la Manipulación de la información y los Tipos de análisis se obtuvo un 50% de errores. Con este resultado se demuestra que solo la mitad de estos elementos se encuentran funcionalmente correctos.

Los principales problemas encontrados durante la realización de estas pruebas son:

- **Se utiliza una fuente cartográfica desconocida.**

Esto constituye un error grave en cualquier SIG, ya que al no conocer la fecha de la fuente cartográfica es imposible afirmar la actualidad de la misma. De esta forma existe la posibilidad que todas las funcionalidades y bondades que brinda el sistema sean incorrectas, ya que este elemento es fundamental en el desarrollo de un SIG.

- **No permite encontrar un lugar donde se cumplan ciertas condiciones.**

Esta función forma parte de los requisitos principales que todo SIG debe cumplir. Es importante que el usuario pueda identificar un lugar específico dadas ciertas condiciones como parámetro.

- **No se utiliza ninguna técnica para la teledetección y fotointerpretación de imágenes.**

Estas técnicas permiten obtener información en formato digital directamente, además de incluir un grupo de funciones que posibilitan el intercambio de ficheros con otros programas y entornos diferentes, por lo que se hace necesaria su presencia en los SIG.

- **No presenta funciones analíticas como: vecindad, conectividad y proximidad.**

La iteración lógica de funciones analíticas permite la creación de modelos para simular distintos procesos y evaluar diferentes aspectos que nos ayuden en la toma de decisiones y en la selección de alternativas.

3.3 Resultados de la revisión.

Al realizar la revisión a la documentación del sistema se debe verificar que la misma se encuentre en su totalidad y tener en cuenta aspectos significativos como la redacción, ortografía, estructura de la documentación, entre otros. Los defectos encontrados en cualquier documento deben corregirse por muy insignificantes que puedan parecer, la documentación de un sistema es el reflejo

del propio sistema y debe estar a la altura del mismo. La tabla que se muestra a continuación resume las no conformidades **(Anexo 3)** encontradas en cada uno de los documentos revisados en las dos etapas o iteraciones que se realizaron las pruebas. Además se especifican las recomendaciones realizadas al equipo de desarrollo con el objetivo de entregar al cliente un producto con mayor calidad en la lista de recomendaciones.

Documentos	Iteración	No conformidades	Significativas	No significativas	Recomendaciones
Proyecto Técnico	1	1	1	0	1
	2	7	6	1	7
Minuta de Reunión	1	3	3	0	3
	2	1	1	0	1
Glosario de términos	1	1	1	0	1
	2	1	1	0	1
Plan de desarrollo de software	1	2	2	0	2
	2	4	3	1	3
Plan de mitigación de riesgos	1	0	0	0	0
	2	1	1	0	1
Arquitectura de software	1	1	1	0	1
	2	1	1	0	1
Modelo de diseño	1	1	1	0	1
	2	1	1	0	1
Casos de prueba	1	0	0	0	0
	2	1	1	0	1
Plan de capacitación	1	1	1	0	1
	2	1	1	0	1
Plan de pruebas	1	0	0	0	0
	2	2	2	0	1
Modelo de sistema	1	0	0	0	0
	2	15	14	1	4
Especificación	1	1	1	0	1

de requisitos	2	3	3	0	1
Modelo de dominio	1	0	0	0	0
	2	1	1	0	1

Tabla 18 Informe estadístico de los resultados de la revisión

Los principales problemas que se detectaron fueron:

- La forma de presentar la información tiene que estar en presente.
- Los requisitos no están redactados de forma simple y clara pues ¨ONRM¨ no aparece en la sección Definiciones, acrónimos y abreviaturas y aparece en la mayoría de los requisitos
- No se han identificado los requerimientos de seguridad (confidencialidad, integridad, disponibilidad).
- Presentan errores ortográficos.

3.4 Validación de expertos. Método Delphi.

Para validar la estrategia propuesta se utilizó el método delphi. Se seleccionó este método ya que presenta ventajas como:

- La información disponible está siempre más contrastada que aquella de la que dispone el participante mejor capacitado, es decir, que la del experto más preparado en el tema.
- El número de factores que es considerado por un grupo es mayor que el que podría ser tenido en cuenta por una sola persona. Cada experto podrá aportar a la discusión general la idea que tiene sobre el tema debatido desde su área de conocimiento.

El método propuesto posibilita un aprovechamiento del debate en grupo, eliminando a la vez las interacciones sociales indeseables que existen dentro de todo grupo de trabajo. Como características fundamentales está presente el anonimato, el cual permite que ningún experto conozca la identidad de los demás que componen el grupo de debate. Otra de las características que posee es la interacción y retroalimentación controlada, la cual se basa en la presentación del mismo cuestionario varias veces.

Al presentar los resultados obtenidos en un cuestionario anterior se logra que los expertos conozcan los distintos puntos de vista y de esta forma modifiquen su opinión si los argumentos que ya han sido presentados les parecen más apropiados que los suyos. La respuesta del grupo en forma estadística es otra característica que está presente en este método ya que la información que se presenta a los expertos contiene todas las opiniones que indicando el grado de acuerdo que se ha

obtenido. La encuesta que se realizó para que cada experto le de respuesta luego de revisar el trabajo previamente realizado consta de 5 interrogantes, las cuales se muestran a continuación.

1. ¿Considera usted que las técnicas de prueba aplicadas al aplicativo SIGUCI fueron las correctas?
2. ¿Considera usted que la estrategia de pruebas propuesta garantizará que el aplicativo SIGUCI pueda sea liberado?
3. ¿Considera que la investigación se pudiera utilizar para la realización del proceso de pruebas en todos los aplicativos SIG incluyendo la plataforma GENESIG?
4. En un rango de 1 a 5 valore la calidad de la investigación.
5. En un rango de 1 a 5 valore la facilidad de comprensión de la investigación.

Para darle respuesta a las preguntas anteriormente expuestas se seleccionaron 7 expertos. Primeramente se tuvo en cuenta años de experiencia como profesor o profesora, idoneidad en el sector educacional, prestigio entre el colectivo de profesores y estudiantes, disposición para participar en la validación, capacidad de análisis y pensamiento lógico, espíritu colectivista y autocrítico, además de valores como la honestidad, responsabilidad, competencia y seriedad.

Elegir los expertos atendiendo a las características mencionadas propicia obtener resultados con calidad, haciendo que las opiniones brindadas sean confiables y válidas para el objetivo propuesto. La encuesta fue enviada a los expertos a través de correo electrónico estableciéndoles un plazo de 5 días para darle respuesta a la misma. Conjuntamente con la encuesta fue entregado el documento que respalda la investigación, así como los entregables definidos en la misma para que puedan realizar analizar adecuadamente lo estudiado. Se siguieron tres etapas para llevar a cabo este proceso. A continuación se muestra un diagrama donde se explica la metodología utilizada.

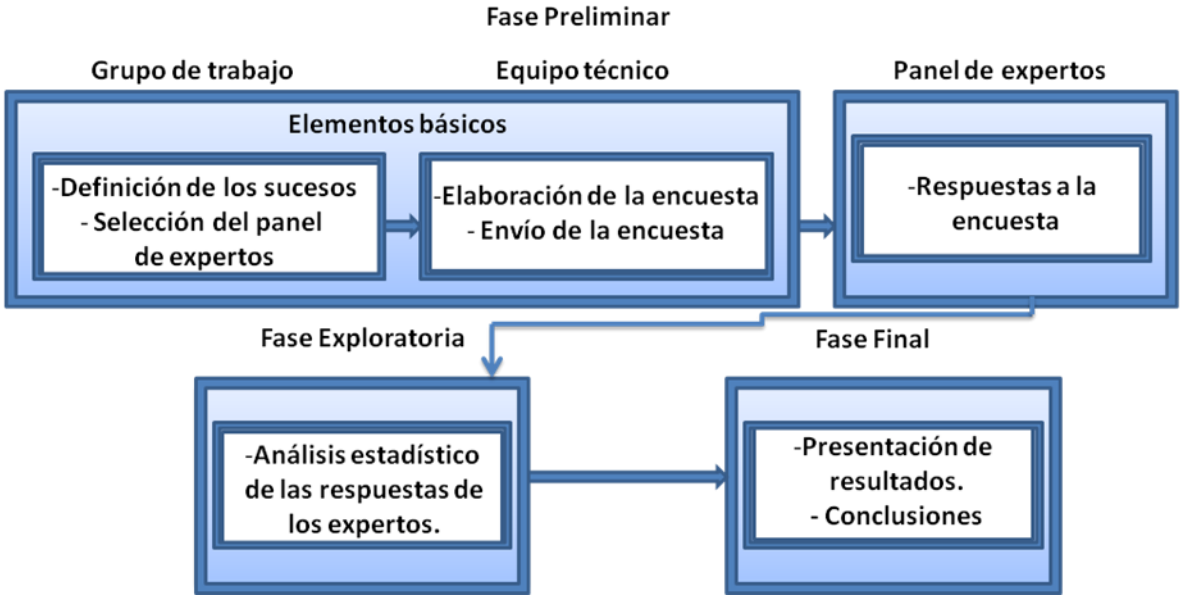


Figura 9 Diagrama de la metodología del método Delphi

Las respuestas obtenidas en el criterio de los expertos permitieron arribar a los siguientes resultados:

Encuesta	Expertos						
Preguntas	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	No	Si	Si	No	Si	Si	Si
2	Si	Si	Si	Si	Si	Si	Si
3	Si	Si	Si	Si	Si	Si	Si
4	5	5	5	4	5	5	5
5	4	5	5	4	5	5	5
%	**84**	**100**	**100**	**76.92**	**100**	**100**	**100**

Tabla 19 Resultados de la encuesta

Como se evidencia en la tabla anterior, los expertos en sentido general consideran que la presente investigación se realizó con la calidad requerida. Se evidencia que las técnicas y los métodos utilizados para la realización del proceso de pruebas fueron las correctas. Dos expertos consideran que se deberían de haber realizado también pruebas de caja blanca. Plantean que éstas son necesarias para comprobar la estructura procedimental y verificar el código implementado en el sistema.

CONCLUSIONES

Los fundamentos teóricos permitieron realizar una descripción explícita y elocuente de los artefactos, actividades y todos los aspectos relacionados con el flujo de trabajo de pruebas como parte del desarrollo de software.

El diseño de la estrategia de pruebas propuesta para la plataforma GENESIG y sus aplicativos logró una mayor organización en las actividades llevadas a cabo durante el proceso de la ejecución de las pruebas.

La aplicación de la estrategia de pruebas permitió verificar la correcta implementación de los requisitos planteados por el cliente, encontrando errores significativos que no fueron detectados por los desarrolladores y que además empañan la buena calidad y estructura tanto del documento como de la aplicación.

RECOMENDACIONES

A partir de la realización de una estrategia de pruebas para la plataforma GENESIG y sus aplicativos y las experiencias adquiridas se recomienda:

Insertar un equipo de aseguradores de calidad a los grupos de desarrollo que aún no cuentan con este, para que se realicen las pruebas necesarias desde los primeros momentos que se comienza a desarrollar el software y durante todo el ciclo de vida, tal y como lo establecen las metodologías de desarrollo.

Incentivar a los grupos de calidad de las diferentes facultades al estudio y preparación sobre las normas de calidad, técnicas y herramientas de pruebas existentes, para asegurar en un futuro que los servicios y productos de la UCI estén a la altura del mercado mundial.

Realizarle al aplicativo SIGUCI las pruebas de seguridad que no han sido realizadas y que propone la guía de pruebas de OWASP.

Referencias Bibliográficas

1. 21, Programa Capacity. Generalidades del SIG. s.l : GeoTecnologías SA, 2002.

2. Cueva, Juan Manuel Lovelle. Calidad del Software. España : s.n., 2009.

3. Social, Fondo. Calidad y Formación: Binomio inseparable. Europa : s.n.

4. Pressman, Roger S. Ingeniería de Software: Un enfoque Práctico. 2005.

5. 8402, ISO. (UNE 66-001-92).

6. Booch, Ivar Jacobson James Rumbaugh Grady. El Proceso Unificado de Desarrollo de Software. Madrid : s.n., 2000. 84-7829-0362.

7. Fernández, Tony Moisés Amarán. Sistema de Gestión de Datos Geológicos. Módulo Inventario de Petróleo y Gas. La Habana : s.n., 2010.

8. Arles Amado Tamayo Rosales, Yosvany Núñez Figueiras. Pruebas de Liberación a la Plataforma de Cálculo Distribuido T-arenal. La Habana : s.n., 2009.

9. López, Yisel Tornés. Diiseño y aplliicaciión de pruebas all producto PRIMICIA,, Pllatafforma de Telleviisiión Informativa. La Habana : s.n., 2009.

10. SistemaDeAseguramientoDeCalidad. . SistemaDeAseguramientoDeCalidad. [Online] 11 12, 2010. http://www.mitecnologico.com..

11. Johanna Rojas Rojas, Emilio José Barrios. Investigación sobre estado del arte en diseño y aplicación de pruebas de software. [Online] 2007. [Cited: 11 25, 2010.] http://www.udistrital.edu.co.

12. Bichachi, Dra. Diana Susana. 2002. Universidad del Salvador. [En línea] 2002. [Citado el: 3 de 4 de 2010.]
http://www.salvador.edu.ar/vrid/iiefgs/tr_check_list.pdf.

13. Anónimo. (2010). Bases de datos Espaciales: Sistemas GIS.

BIBLIOGRAFÍA

1. Capacity, Programa. Generalidades del SIG. s.l. : GeoTecnologías SA, 2002. 21.

2. Cueva, Juan Manuel Lovelle. Calidad del Software. España : s.n., 2009.

3. Social, Fondo. Calidad y Formación: Binomio inseparable. Europa : s.n.

4. Pressman, Roger. Ingeniería de Software: Un enfoque Práctico. 2005.

5. 66-001-92), (UNE. ISO 8402.

6. Booch, Ivar Jacobson James Rumbaugh Grady. El Proceso Unificado de Desarrollo de Software. Madrid : 84-7829-0362, 2000.

7. Fernández, Tony Moisés Amarán. Sistema de Gestión de Datos Geológicos. Módulo Inventario de Petróleo y Gas. La Habana : s.n., 2010.

8. rles Amado Tamayo Rosales, Yosvany Núñez Figueiras. Pruebas de Liberación a la Plataforma de Cálculo Distribuido T-arenal. La Habana : s.n., 2009.

9. López, Yisel Tornés. Diiseño y aplliicaciión de pruebas all producto PRIMICIA. Pllatafforma de Telleviisiión Informativa. La Habana : s.n., 2009.

10. SistemaDeAseguramientoDeCalidad. SistemaDeAseguramientoDeCalidad. [En línea] 12 de 11 de 2010. http://www.mitecnologico.com.

11. Johanna Rojas Rojas, Emilio José Barrios. Investigación sobre estado del arte en diseño y aplicación de pruebas de software. [En línea] 25 de 11 de 2010. http://www.udistrital.edu.co.

12. Bichachi, Dra. Diana Susana. [En línea]

13. [En línea] http://www.salvador.edu.ar.

14. Acuña, César Javier. Kybele. Grupo de Investigación. 2011.

15. Educación. [En línea] [Citado el: 20 de enero de 2011.] http://www.salvador.edu.ar.

16. ConexionIT. Comunidad de Profesionales de IT. [En línea] 22 de enero de 2008. http://www.conexionit.com/blog/metodologias/que-es-rup.html..

17. Fernández, José Almendros. [En línea] Molinux, 27 de febrero de 2009. https://forja.molinux.info/frs/download.php/203/Spela_PRU_PlandePruebas_v3.0.doc..

18. Myers, Glenford J. The Art of the Software Testing. 1979.

19. Aspiazu, G. C. Ingeniería del Software. Principios y Conceptos. La Paz. Bolivia : s.n., 2002.

20. Berzosa, L. R. Evitando el síndrome de "caja negra" en proyectos de desarrollo software externalizados. Madrid. España: Expo:QA : s.n., 2008.

21. Cosín, J. D. Técnicas cuantitativas para la gestión en la ingeniería del software. s.l. : Netbiblo, 2007.

22. Díaz, J. G. Introducción al Proceso de Pruebas. Departamento de Lenguajes y Sistemas Informáticos. Sevilla : Universidad de Sevilla.

23. Jornadas Profesionales de Calidad y Testing de Software. [En línea] Expo:QA, 2005. http://www.expoqa.com.

24. Pérez, P. G. Principios básicos del desarrollo seguro. División de Seguridad Lógica de Germinus. [En línea] 2009.

25. Anónimo. Bases de datos Espaciales: Sistemas GIS. 2010.

ANEXOS

Anexo 1.Lista de recomendaciones de Pruebas exploratorias.

No.	Criterio de evaluación	Elemento	Recomendaciones	IMP	ED	EST
1	Aplicación SIGUCI	Aplicación.	La aplicación no se muestra en el navegador: Internet explorer. Es factible que la aplicación se muestre en los navegadores habituales como: Explorer, opera, Netscape.	A	Pruebas exploratorias.	Pendiente.
2	Aplicación SIGUCI.	Interfaces	Se deben aprovechar los espacios en algunas interfaces, para esto se puede reducir el tamaño de las mismas .	A	Pruebas exploratorias.	Pendiente.
3	Aplicación SIGUCI.	Interfaces/ Buscar personas.	Deben aparecer todos los criterios de búsqueda para encontrar un usuario.	A	Pruebas exploratorias.	Pendiente.
4	Aplicación SIGUCI.	Aplicación	Debe identificar al usuario en cualquier parte de la aplicación mientras el mismo se encuentre en el sistema.	A	Pruebas exploratorias.	Pendiente.

No.	Criterio de evaluación	Elemento	Recomendaciones	IMP	ED	EST
5	Aplicación SIGUCI.	Aplicación	Eliminar el error que muestra en ocasiones cuando se abre la aplicación.	A	Pruebas exploratorias.	Pendiente.
6	Aplicación SIGUCI	Contenido	Revisar la ortografía en el contenido de la aplicación y arreglar los errores que presenta.	A	Pruebas exploratorias.	Pendiente.
7	Aplicación SIGUCI	Aplicación	Implementar correctamente las funcionalidades: “Mover mapa”, “Mapa temático”, “Anterior”, “Siguiente”	A	Pruebas exploratorias.	Pendiente.

Anexo 2. No conformidades detectadas durante las pruebas funcionales.

Elemento	No	No Conformidad	Aspecto Correspondiente	Etapa de Detección	Significativa	No Significativa	Recomendación	Estado NC	Respuesta del Equipo de Desarrollo
Aplicación	1	No se muestran correctamente todos los datos del edificio, en algunas descripciones faltan imágenes.	Ventana Buscar edificio/opción "ver detalles"	Segunda iteración de pruebas de caja negra.	• Funcionalidad	-	Mostrar todos los datos correspondientes a los apartamentos y las personas del edificio seleccionado.	PD	
Aplicación	2	El sistema no cancela la búsqueda de personas.	Ventana Buscar edificio/ opción "cerrar ventana"	Segunda iteración de pruebas de caja negra.	• Funcionalidad	-	Debe cerrarse la búsqueda que se ha llevado a cabo hasta el momento.	PD	

Aplicación	3	Algunos errores que se muestran para especificar los valores que el usuario debe entrar son descritos en inglés.	Opciones: “Buscar persona”, “Buscar edificio”	Segunda iteración de pruebas de caja negra	-	X	Especificar los errores en español que sean entendibles para el usuario.	PD	
Aplicación	4	El sistema en ocasiones no busca la persona que desea el usuario cuando el criterio de búsqueda es “usuario” o “solapín” siendo los datos correctos y además muestra un mensaje de aviso.	Funcionalidad: Buscar persona	Segunda iteración de pruebas de caja negra	Funcionalidad	-	Verificar que con todos los criterios de búsqueda se realice correctamente la búsqueda de una persona.	PD	

Aplicación	5	La interfaz correspondiente a esta sección no se muestra en la aplicación, por lo que no se puede probar ningún escenario de esta sección.	Barra de herramientas/ opción “mapa temático”	Segunda iteración de pruebas de caja negra.	Funcionalidad	-	Implementar correctamente esta funcionalidad del sistema.	PD	
Aplicación	6	No se puede seleccionar la opción de “Gráficas dinámicas”, ya que no se muestra en la aplicación la interfaz correspondiente a esta funcionalidad, por lo que no se puede probar ninguno de los escenarios de esta sección.	Barra de herramientas/ opción “mapa temático”	Segunda iteración de pruebas de caja negra.	Funcionalidad	-	Implementar correctamente esta funcionalidad del sistema.	PD	

Aplicación	7	No se puede seleccionar la opción de "Símbolo Proporcional", ya que no se muestra en la aplicación la interfaz correspondiente a esta funcionalidad, por lo que no se puede probar ninguno de los escenarios de esta sección.	Barra de herramientas/ opción "mapa temático"	Segunda iteración de pruebas de caja negra.	Funcionalidad	-	Implementar correctamente esta funcionalidad del sistema.	PD	

Anexo 3. Lista de recomendaciones de la revisión a la documentación del aplicativo SIGUCI.

No.	Criterio de evaluación	Elemento	Recomendaciones	IMP	ED	EST
1	Adherencia a producto	Diario de Actividades v1.0 (2. Gestión de Proyecto/2.1 Plan de Proyecto) No existe el documento	Crear el documento	A	R	PD
2	Adherencia a producto	PILA DE SPRINT (SPRINT BACKLOG) v1.0 (2. Gestión de Proyecto/2.1 Plan de Proyecto) No existe el documento	Crear el documento	A	R	PD

3	Adherencia a producto	REGISTRO DE TAREAS DE INGIENERIA v1.0 (2. Gestión de Proyecto/2.1 Plan de Proyecto) No existe el documento	Crear el documento	A	R	PD
4	Adherencia a producto	PLAN DE CAPACITACIÓN v2.0 (2. Gestión de Proyecto/2.3 Recursos) No se identifica correctamente el nombre del Proyecto en la primera hoja, aparece: Proyecto GeneSIG Plataforma Soberana GeneSIG	Identificar correctamente el nombre del Proyecto	M	R	PD
5	Adherencia a Producto	MINUTA DE REUNIÓN v2.0 (2. Gestión de Proyecto/2.7 Reuniones) Minuta 16 de septiembre Se repite del punto 2.	Eliminar la repetición del punto 2.	A	R	PD
6	Adherencia a Producto	MINUTA DE REUNIÓN v2.0 (2. Gestión de Proyecto/2.7 Reuniones) Minuta 16 de septiembre Faltan los acuerdos de trabajos tomados	Poner los acuerdos de la minuta.	A	R	PD
7	Adherencia a Producto	MINUTA DE REUNIÓN v2.0 (2. Gestión de Proyecto/2.7 Reuniones) Minuta 7 de Octubre Existe una falta ortográfica en la inquietud 8, segundo párrafo.	Revisar la ortografía de la inquietud 8.	A	R	PD
8	Adherencia a Producto	ESPECIFICACIÓN DE REQUISITOS v2.0 (1. Ingeniería/1.1 Requisitos)	Especificar los requisitos que afectan la usabilidad, fiabilidad, eficacia y soporte	M	R	PD

			que puedan afectar el sistema.			
9	Adherencia a Producto	ROLES Y RESPONSABILIDADES v2.0 (2. Gestión de Proyecto/2.3 Recursos) Uno de los roles que se especifican está mal definido, el de Calidad	Definir bien el rol de Calidad.	M	R	PD
10	Adherencia a Producto	ARQUITECTURA DE INFORMACIÓN v2.0 (1. Ingeniería/1.2 Arquitectura y Diseño) No existe el documento	Crear el documento	A	R	PD
11	Adherencia a Producto	MODELO DE DISEÑO v2.0 (1. Ingeniería/1.2 Arquitectura y Diseño) No existe el documento	Crear el documento	A	R	PD
12	Adherencia a Producto	PLAN DE DESARROLLO DE SOFTWARE v2.0 (2. Gestión de Proyecto/2.1 Plan de Proyecto) En la Introducción no se describen correctamente los objetivos	Describir correctamente los objetivos de la introducción del documento	A	R	PD
13	Adherencia a Producto	PLAN DE DESARROLLO DE SOFTWARE v2.0 (2. Gestión de Proyecto/2.1 Plan de Proyecto) No se especifica en Interfaces externas los Datos de los contactos	Especificar los datos de contacto de las Interfaces externas	A	R	PD
14	Adherencia a Producto	PLAN DE ASEGURAMIENTO DE CALIDAD v2.0 (3. Soporte/3.1 Aseguramiento de la Calidad) No existe el documento	Crear el documento	A	R	PD
15	Adherencia a Producto	GLOSARIO DE TÉRMINOS v1.0 (3. Soporte/3.1 No existe el documento	Crear el documento	A	R	PD

16	Adherencia a Producto	PLAN DE MEDICIONES v1.0 (3. Soporte/3.1 Aseguramiento de la Calidad) No existe el documento	Crear el documento	A	R	PD
17	Adherencia al producto.	ARQUITECTURA DE INFORMACIÓN v2.0 (1. Ingeniería/1.2 Arquitectura y Diseño) No existe el documento	Crear el documento	A	R	PD
18	Adherencia al producto	DOCUMENTO DE ARQUITECTURA DE SOFTWARE v2.0 (1. Ingeniería/1.2 Arquitectura y Diseño) No existe el documento	Crear el documento	A	R	PD