HERRAMIENTA COMPUTACIONAL PARA LA DOCUMENTACIÓN DE PRUEBAS DE SOFTWARE INTEGRADA AL SOFTWARE QUIS 2.0.

EDGAR FERNANDO VEGA MORALES

JUAN LUIS AGUILAR GARCÍA

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE INGENIERIAS FISICO-MECANICAS

ESCUELA DE INGENIERIA DE SISTEMAS E INFORMÁTICA

BUCARAMANGA

2013

HERRAMIENTA COMPUTACIONAL PARA LA DOCUMENTACIÓN DE PRUEBAS DE SOFTWARE INTEGRADA AL SOFTWARE QUIS 2.0.

EDGAR FERNANDO VEGA MORALES

JUAN LUIS AGUILAR GARCÍA

Trabajo de Grado para optar por el título de Ingeniero de Sistemas

Director

LUIS CARLOS GOMEZ FLOREZ

MSc. Informática y Ciencias de la Computación

Codirector

Nelson Enrique León Martínez

MSc. Informática y Ciencias de la Computación

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE INGENIERIAS FISICO-MECANICAS

ESCUELA DE INGENIERIA DE SISTEMAS E INFORMÁTICA

BUCARAMANGA

2013

AGRADECIMIENTOS

Al profesor Luis Carlos Gómez Flores por sus siempre valiosas enseñanzas y sus invaluables consejos.

Al ingeniero Nelson Enrique León Martínez por su guía, apoyo, confianza, tiempo y paciencia.

A los compañeros que nos acompañaron en nuestro paso por la universidad, gracias a ellos nuestro paso por ella fue mucho más agradable e enriquecedora.

A los compañeros del grupo de investigación en Ingeniería Biomédica y del grupo de investigación en Sistemas y Tecnologías de la información, por su apoyo en el desarrollo del último objetivo de este proyecto.

A la Universidad Industrial de Santander a través de la Vicerrectoría de Investigación y Extensión por el apoyo económico al proyecto.

TABLA DE CONTENIDO

LISTA DE TABLAS

Pág.

LISTA DE FIGURAS

Pág.

RESUMEN

TITULO

IMPLEMENTACIÓN DE UNA HERRAMIENTA COMPUTACIONAL PARA LA DOCUMENTACIÓN DE PRUEBAS DE SOFTWARE, INTEGRADA AL SISTEMA PARA LA EVALUACION DE CALIDAD SOFTWARE DERIVADO DE ACTIVIDADES DE INVESTIGACION QUIS 2.0.*

AUTORES.

Juan Luis Aguilar García, Edgar Fernando Vega Morales.

PALABRAS CLAVE

Calidad, Documentación, IEEE 829, Pruebas, Software

DESCRIPCIÓN

Las pruebas software toman un lugar importante durante el desarrollo de aplicaciones informáticas ya que gracias a estas se pueden evitar un sin número de defectos en las aplicaciones finales, los cuales hacen que estas no puedan, en el peor de los casos, ser usadas. Sin embargo, no es suficiente solo la realización de las pruebas, Es igual de importante documentarlas, ya que permite obtener resultados tangibles de las pruebas realizadas y así mismo llevar un control de lo que se ha probado hasta el momento, para que al momento de presentarse un fallo, se pueda establecer un plan de acción.

Este proyecto tiene como objetivo principal crear una herramienta computacional que permita documentar las pruebas de software, soportada en normas internacionales elaboradas para tal fin. Esta herramienta será incorporada en la aplicación Quality UIS (QUIS), cuyo fin es apoyar la metodología de evaluación de la calidad en los grupos de investigación de la Universidad Industrial de Santander que desarrollan software como resultado de sus investigaciones. El proyecto generó como resultado una herramienta computacional para la documentación de pruebas software, basada en la norma IEE 829 e incorporada al software para la evaluación de la calidad QUIS.

El resultado fue obtenido gracias a una amplia investigación sobre la documentación de pruebas, herramientas que existen en el mercado relacionadas con pruebas software, la revisión del estado actual de QUIS con el propósito de hacer un mantenimiento correctivo de los problemas presentes en el momento y la implementación del módulo de documentación de pruebas software a la herramienta. Para finalizar se presenta una ilustración del uso de la herramienta llevada a cabo en varios enfoques de investigación.

* Trabajo de grado.

** Facultad de ingenierías Fisicomécanicas. Escuela de ingeniería de sistemas e informática. Director: Luis Carlos Gómez Florez. Codirector: Nelson Enrique León Martínez

ABSTRACT

TITLE

IMPLEMENTATION OF A COMPUTER TOOL FOR THE SOFTWARE TESTING DOCUMENTATION. INTEGRATED TO THE SYSTEM FOR ASSESSMENT OF QUALITY SOFTWARE DERIVATE OF RESEARCH ACTIVITIES QUIS 2.0.*

AUTHORS

Juan Luis Aguilar García, Edgar Fernando Vega Morales.

KEY WORDS

Quality, Documentation, IEE 829, Software Testing

DESCRIPTION

The software testing takes a very important place in the application development. And thanks to that it can prevent an uncountable numbers of defects in the final applications. Those defects make the applications useless in the worse of the cases. However it's not enough to only do testing. It's equally of important to document. Because that can get concrete results of the tests and you keep track of what has been test so far. So when there is a bug. You can establish an action plan.

This Project has as a main objective created a computer tool to the software testing documentation. This tool had support on international standards for development that goal. The tool will be incorporate in the application Quality UIS (QUIS). The object of QUIS is support the methodology of quality assessment on the research groups of the Universidad Industrial de Santander to develop software as a result of their research. The project produces as a result one computer tool for the software testing documentation. It has support en the standard IEE 829 and adds to the software for the quality assessment QUIS.

This result was obtain thanks to a extend research of the testing documentation, tools available on the market relating to software testing, a review of the current state of QUIS. With the purpose of make a corrective maintenance of the problems present at the time, and the implementation of the testing documentation module on the tool. To conclude it shows the illustration of the use of the tool performs in several research approaches.

* Degree Work.

** Faculty of Physical-Mechanics engineering. School of systems and computing engineering. Director: Luis Carlos Gómez Florez. Co-director: Nelson Enrique León Martínez.

INTRODUCCIÓN

Si se tiene en cuenta la alta competitividad que hay hoy en día, donde el usuario puede escoger entre un variado y amplio número de productos orientados a satisfacer la misma necesidad(es), no se puede pasar por alto un factor tan importante como lo es la calidad, la cual está relacionada con la satisfacción del usuario. En el campo del desarrollo del software, la calidad juega un papel importantísimo, ya que si un software no posee un grado de calidad aceptable, difícilmente llegara a ser comercializado, y si lo fuere, no tardaría mucho en ser dejado en el olvido a causa de sus defectos, los cuales generan inconformidad y desuso por parte de los usuarios. Dentro de las actividades propias de los grupos de investigación se encuentra el desarrollo de software. Este software está orientado a satisfacer necesidades en el interior de los grupos o de una organización ajena al mismo, por lo tanto es indispensable la calidad en los productos software allí realizados, para que el usuario quede satisfecho y a su vez se cumpla de la mejor manera el propósito para el cual fue desarrollado.

QUIS 2.0 (Quality UIS), es un software desarrollado en el grupo de investigación en Sistemas y Tecnologías de la Información (STI-UIS), orientado a apoyar los procesos de desarrollo de software de calidad. En sus primeras versiones evaluaba tres aspectos: calidad del proyecto, calidad del proceso y calidad del producto. No obstante, si se desea abarcar de forma más precisa la calidad de un producto software, no se puede dejar a un lado las pruebas de software y su respectiva documentación como evidencia tangible de las actividades de prueba realizadas.

Si se tiene en cuenta que las pruebas de software constituyen un proceso vital para la calidad de los productos software ya que ayudan a encontrar los errores presentes en él y a su vez evitaran a futuro: perdidas económicas, de personal, de información, etc. Se puede entender el gran esfuerzo que hacen hoy día las

organizaciones por llevar a cabo este tipo de prácticas destinando para ello entre el 30 y 40 por ciento del esfuerzo de un proyecto. [1]

En la realización de pruebas hay una actividad muy importante, que es la documentación de estas. Prácticamente todos los desarrolladores podrían afirmar que efectúan pruebas durante el desarrollo de sus aplicaciones ya sea haciendo uso de las técnicas y métodos existentes o simplemente al ejecutar la aplicación y verificar su funcionamiento, pero estas pruebas no pueden quedar a la deriva, es necesario poseer una evidencia tangible de las actividades de pruebas realizadas y los resultados obtenidos. Por lo tanto, se podría decir que la documentación es inherente a las pruebas, porque sin ella no se podría constatar lo que se ha hecho y no se puede valorar la calidad del software probado.

La problemática planteada anteriormente, hace necesario el desarrollo de una nueva versión del software QUIS la cual posea un módulo para la documentación de pruebas por medio del cual se pueda complementar la herramienta y a su vez determinar la calidad de los productos software desarrollados dentro de los grupos de investigación con evidencias tangibles.

A continuación se presenta el trabajo realizado para la implementación del módulo de pruebas, comenzando por la identificación, a partir de las normas y estándares internacionales, el estado del arte que soporta la realización y documentación de las pruebas software, producto del cual se pudo establecer la norma IEEE 829 como guía a seguir para el planteamiento del modelo propuesto. Luego se realizó una búsqueda y estudio de algunas herramientas computacionales que tuvieran funciones similares a la herramienta que se quería realizar, para tomar algunas funcionalidades que pudieran ser de gran importancia incluir. Después de esta investigación, se procedió a avaluar la herramienta QUIS 2.0 para detectar y corregir el mayor número de defectos posibles. Posteriormente se procedió a implementar el módulo de pruebas, basado en el modelo propuesto a partir de la norma IEEE 829, que fue incluido

[1] Pressman Roger. Ingeniería del software-Un enfoque práctico. Quinta Edición. México. McGraw-Hill 2002. p.281.

dentro de la nueva versión de QUIS. Por último, se realizó una ilustración del uso de QUIS 3.0, mediante la evaluación de calidad realizada a aplicaciones de diferentes grupos de investigación.

1 GENERALIDADES.

1.1 CONTEXTUALIZACIÓN DEL PROYECTO

El presente proyecto se desarrolló en el interior del Grupo de Investigación en Sistemas y Tecnologías de la información STI de la Universidad Industrial de Santander. Este proyecto está dirigido a los grupos de investigación de esta institución, más específicamente a los que desarrollan software como producto de sus trabajos de investigación y consiste en la implementación de un software basado en normas internacionales elaboradas para la documentación de pruebas software.

1.1.1 Grupos de investigación

En la Universidad Industrial de Santander existen alrededor de 113 grupos de investigación, de los cuales varios producen software y la totalidad de ellos usan productos como estos en sus investigaciones. Generalmente la estructura de un grupo de investigación que desarrolla software está conformada por:

Director: es la persona encargada de liderar el grupo.

Jefes de líneas o directores de línea: responden ante el director y se encargan de buscar personas para los proyectos a desarrollar y a su vez acompañan a los desarrolladores durante el proceso.

Desarrolladores: son personas en su mayoría estudiantes de pregrado, de últimos semestres, los cuales se involucran en el proyecto, que les sirve como requisito de grado.

¿Cómo se dan los desarrollos de software en un grupo de investigación?

Los proyectos nacen de una necesidad bien sea propia del grupo de sus investigadores o de una entidad externa que incluya dentro de sus requerimientos el desarrollo de una aplicación computacional y que se enmarque dentro de una de las líneas de investigación. El jefe de línea busca personal para su desarrollo, este les plantea el problema y con esta idea se

empieza a elaborar el plan para la realización del proyecto propuesto. En la mayoría de casos sucede que el desarrollador define la metodología, el lenguaje, la estructura para su plan de trabajo, es decir hace su desarrollo dependiendo de su habilidad y sin tener en cuenta estudios de viabilidad de desarrollo y la compatibilidad de herramientas para tal fin, generando en algunas ocasiones inconvenientes tales como incumplimiento en los cronogramas por cambios debido a la falta de planeación, problemas de compatibilidad entre herramientas, entro otros.

Otro inconveniente que se presenta es que a la hora de determinar el proyecto no se hace una evaluación exhaustiva de calidad al software desarrollado, lo cual conlleva a que el proyecto desarrollado no se utilice como se planificó y termine siendo otro producto más archivado en la biblioteca.

1.2 PLANTEAMIENTO Y DEFINICIÓN DEL PROBLEMA

¿De qué manera se puede incorporar la documentación de pruebas de software en la evaluación de la calidad de los desarrollos realizados en el interior de los grupos de investigación?

En el grupo de investigación en Sistemas y Tecnologías de la Información (STI) se desarrolló una herramienta computacional para la evaluación de la calidad del software llamada QUIS (Quality UIS), que en sus primeras versiones evaluaba tres aspectos: calidad del proyecto, calidad del proceso y calidad del producto. Esta herramienta pretende ayudar a los grupos de investigación en el proceso de desarrollar software de calidad, ya que dentro de los grupos se realizan varios proyectos y con diferentes enfoques, pero no se tiene un mecanismo adecuado de comprobación de la calidad software.

Uno de los factores más importantes que impide que el software desarrollado dentro de los grupos de investigación sea utilizado, son los defectos presentes en la aplicación, los cuales generan inconformidad y desconfianza para su posible uso. Partiendo de esta situación, se ha generado una gran preocupación por resolver este inconveniente, el cual está directamente relacionado con la calidad del software desarrollado y más específicamente con

la falta de realización de pruebas de software y su documentación, lo cual no se tenía en cuenta en la evaluación realizada por QUIS.

Para la realización de este proyecto se ha centrado la atención en el factor mencionado anteriormente. Puesto que existe una herramienta computacional enfocada a la evaluación de la calidad de los productos software, se hace indispensable disminuir los defectos presentes en las aplicaciones que se desarrollan continuamente, haciendo uso de las pruebas de software, las cuales quedarán debidamente documentadas gracias a la nueva versión de QUIS, que incluirá un módulo destinado para ello.

En conclusión, la documentación de las pruebas de software es un componente de gran importancia que debe tener la herramienta QUIS, ya que es necesario dejar evidencia de las pruebas realizadas al software en evaluación.

1.3 OBJETIVOS

1.3.1 General

IMPLEMENTAR UNA HERRAMIENTA COMPUTACIONAL PARA LA DOCUMENTACIÓN DE PRUEBAS DE SOFTWARE, INTEGRADA AL SISTEMA PARA LA EVALUACIÓN DE CALIDAD SOFTWARE DERIVADO DE ACTIVIDADES DE INVESTIGACIÓN QUIS 2.0.

1.3.2 Específicos.

- Identificar, a partir de las normas y los estándares internacionales más relevantes, el estado del arte que soporte la realización y documentación de pruebas de software.
- Realizar una revisión de las herramientas computacionales existentes en el mercado que cumplan una función similar al software que se desea realizar.
- Hacer una evaluación del estado actual de la herramienta QUIS versión 2.0, que determine los defectos presentes en la aplicación para su corrección.
- Implementar en el software de evaluación de la calidad QUIS el módulo para la documentación de pruebas software.
- Ilustrar el uso de la herramienta QUIS versión 3.0 en la documentación de pruebas de software desarrollados en el interior de grupos de investigación.

2. CONCEPTOS BÁSICOS, NORMAS Y ESTÁNDARES PARA LA DOCUMENTACIÓN DE PRUEBAS SOFTWARE.

2.1 FUNDAMENTOS BÁSICOS DE LAS PRUEBAS DE SOFTWARE

2.1.1. Las pruebas de software.

Las pruebas de software son procesos realizados concurrentemente a través de las etapas del desarrollo de software que utiliza y mantiene el testware y cuyo objetivo es apoyar la disminución del riesgo de aparición de fallas y faltas en operación.[2]

Las pruebas de software también pueden ser vistas como la ejecución de un programa con la intención de descubrir un error, para lo cual se encuentran diferentes estrategias y técnicas experimentales que ayudarán a hacer esta práctica más eficiente.

Las pruebas constituyen un proceso muy importante para la calidad del producto software ya que ayudan a encontrar los errores presentes dentro del mismo, lo cual evitará a futuro: perdidas económicas, de personal, de información, etc., por lo cual las organizaciones desarrolladoras de software están destinando entre el 30 y 40 por ciento del esfuerzo de un proyecto a la realización de las pruebas. [3]

En este proceso existe una interesante analogía para los ingenieros del software ya que durante la fase de desarrollo el ingeniero tiene una actitud constructiva y cuando llega a la etapa de las pruebas tiene que cambiarla por una destructiva, dado que las pruebas solo serán consideradas exitosas si se encuentra un error.

[2] www.acis.org.co. Recuperado el 27 de diciembre de 2012, de http://www.acis.org.co/fileadmin/Base_de_Conocimiento/XXVII_Salon_Informatica/MariaClaraChoucair-PruebasDeSoftware.pdf.

[3] Pressman Roger. Ingeniería del software-Un enfoque práctico. Op.cit. p. 281

2.1.2 Objetivos de las pruebas

En el marco de las pruebas de software se encuentran muchos objetivos dentro de los cuales se pueden resaltar los siguientes:[4] [5]

1) Para mostrar al desarrollador y al cliente que el software satisface sus requerimientos.
2) Para descubrir errores en el software en el que el comportamiento de este es incorrecto o, no deseable o no cumple su especificación.
3) Lograr confianza acerca del nivel de calidad.
4) Proveer información
5) Prevenir defectos.

2.1.3 Principio de las pruebas

En las pruebas de software se deben tener en cuenta ciertos principios fundamentales para poder determinar casos de prueba efectivos. Algunos de estos principios se encuentran en el libro de Pressman[6] sugeridos por Davis[7], y son los siguientes:

- A todas las pruebas se les debería hacer un seguimiento hasta los requisitos del cliente.
- Las pruebas deberían planificarse mucho antes de que empiecen.
- El principio de Pareto es aplicable a la prueba del software.
- Las pruebas deberían empezar por lo pequeño y progresar hacia lo grande.
- No son posibles las pruebas exhaustivas.
- Para ser más eficaces las pruebas deberían ser realizadas por un equipo independiente.

[4] Sommerville Ian. Ingeniería del software. Séptima Edición. Madrid. Pearson Educación 2005. p. 492.
[5] www.acis.org.co. Op.cit.
[6] Pressman Roger. Ingeniería del software-Un enfoque práctico. Op. cit. p. 282 – 283.
[7] Davis, A., 201 Principles of Software Development, McGraw-Hill, 1995.

También puede tener en cuenta los siete principios que sugiere el libro Foundations of Software:[8]

- Las pruebas evidencian la presencia de defectos.
- El testing en forma exhaustiva es imposible.
- Probar en fases tempranas.
- Agrupamiento de los defectos.
- Paradoja "del pesticida": "La cual está definida, como el proceso de repetir una y otra vez el mismo conjunto de pruebas, con lo que eventualmente con este conjunto no se podrán encontrar ningún nuevo defecto en el software". [9]
- Las pruebas son dependientes del contexto.
- Falacia de la ausencia de errores.

2.2 ESTRATEGIAS DE LAS PRUEBAS.

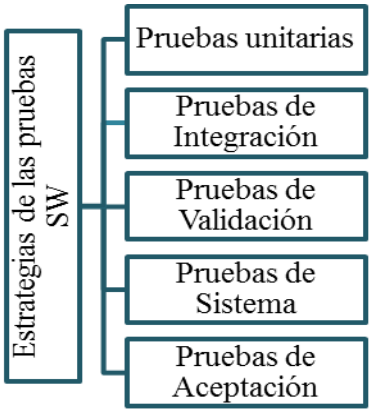
Como Pressman señala[10] "*una estrategia de prueba del software integra las técnicas de diseño de casos de prueba en una serie de pasos bien planificados que dan como resultado una correcta construcción del software. La estrategia proporciona un mapa que describe los pasos que hay que llevar a cabo como parte de la prueba, cuando se deben planificar y realizar esos pasos, y cuanto esfuerzo, tiempo y recursos se van a requerir. Por lo tanto, cualquier estrategia de prueba debe incorporar la planificación de la prueba, el diseño de casos de prueba, la ejecución de las pruebas y la agrupación y evaluación de los datos resultantes.*"

Las estrategias de las pruebas son muy importantes, ya que si un software se prueba sin tenerlas en cuenta, seguramente se desperdiciaría tiempo, se dedicaría un esfuerzo innecesario y lo que es peor puede que no se detecte

[8] Graham Dorothy, Van Veenendaal Erik, Evans Isabel, Black Rex. Foundations of software testing. Edición actualizada por la fundación ISTQB año 2007. p. 22.
[9] Ibíd. p. 22.
[10] Pressman Roger. Ingeniería del software-Un enfoque práctico. Op. cit. p. 305.

ningún error. Las principales estrategias de software se pueden ver en la siguiente figura:

Figura 1. Estrategias de las pruebas de software.



Fuente: Autores

2.2.1 Pruebas unitarias.

Esta prueba es el primer paso a seguir dentro de las estrategias y se centra básicamente en la unidad más pequeña de desarrollo, el módulo. Todos los caminos se prueban para descubrir errores dentro de los límites del módulo (métodos, funciones, consultas para reportes etc.).

Algunas consideraciones que se deben tener en cuenta al momento de hacer pruebas de unitarias son las descritas por Pressman:[11]

- La interfaz del módulo se prueba para asegurarse de que la información fluye apropiadamente hacia adentro y fuera de la unidad de programa sujeta a prueba.
- Examinar todas las estructuras de datos locales para garantizar que todos los temporales mantengan su integridad durante todos los pasos de la ejecución del algoritmo.
- Se reconocen todos los caminos independientes para asegurarse de que se hallan recorrido por lo menos una vez.
- Se prueba en los límites para asegurarse de que el módulo funciona bien en los mismos.

[11] Ibíd. p. 310.

- Se prueban todos los caminos de manejo de errores.

2.2.2 Pruebas de integración

Una prueba de integridad consiste básicamente en unir todos los módulos que han sido probados o no probados por pruebas unitarias y forman un esquema más completo o interfaz, ya que así se hayan probado todos los módulos con pruebas unitarias esto no garantiza que cuando sean unidos no presentes errores, porque algunos pueden estar trabajando de forma adversa.[12]

La prueba de integración es una técnica que permite construir la arquitectura del software mientras que al mismo tiempo se prueba para descubrir errores asociados principalmente a la interfaz.[13]

Dentro de esta estrategia de prueba se recomienda probar los módulos de forma incremental, de tal forma que el siguiente módulo que se debe probar se combina con el conjunto de módulos que ya han sido probados. Algunas estrategias de pruebas de integración incremental son: prueba de regresión, integración ascendente e integración decadente.

2.2.3 Pruebas de validación

Cuando el producto software esta ensamblado correctamente y en su totalidad se hace necesario hacer una prueba más, la prueba de validación en la cual se prueban todos los módulos en conjunto para así determinar si se están cumpliendo con todos los requerimientos. En esta práctica cabe resaltar que hay dos tipos: la prueba alfa y la prueba beta.

En la prueba alfa el software se utiliza en un entorno natural mientras el desarrollador está al tanto para registrar los errores y problemas de uso, esta pruebas son realizadas en entornos controlados. Mientras que las pruebas beta se aplican en un lugar de trabajo de los usuarios finales y a diferencia de la

[12] Sommerville Ian. Ingeniería del software. Op.cit. p. 295 – 297.
[13] Pressman Roger. Ingeniería del software-Un enfoque práctico. Op.cit. p. 312

alfa, por lo general el desarrollador no está, por lo tanto el usuario es el que registra los errores que el aprecia en la aplicación.[14]

Existen algunos criterios para este tipo de pruebas:[15]

- La validación del software se logra mediante una serie de pruebas que demuestren que se cumplen los requisitos.
- Un plan de prueba delinea la clase de pruebas que se aplicaran y un procedimiento de prueba define los casos de prueba específicos.
- Después de que se ha dirigido cada caso de prueba de validación, existirá dos condiciones posibles: 1) la característica de funcionamiento o desempeño cumple con la especificación. 2) se descubre una desviación de la especificación y se crea una lista de deficiencias.

2.2.4 Pruebas de sistema

Las pruebas del sistemas son aquellas que se hacen cuando ya se tienen todos los módulos conectados entre sí, ya que a pesar de que puedan estar funcionando bien individualmente, puede que en conjunto estén ocasionando muchos errores que no se hayan podido detectar. En el marco de las pruebas de sistemas se pueden destacar cuatro tipos:[16]

- *Prueba de recuperación:* esta prueba lo que busca es hacer que el sistema falle de varias maneras y verifica que su recuperación se realice apropiadamente.
- *Prueba de seguridad:* se centra en la comprobación del funcionamiento apropiado de los mecanismos de control al momento de sufrir interrupciones inapropiadas.
- *Prueba de resistencia:* ejecuta el sistema de tal forma que los recursos que demande del mismo en cantidad, volumen o frecuencia, sean anormales, para así poder tener una idea de cuánto resiste el sistema.

[14] Ibíd. p. 316
[15] Ibíd. p. 316.
[16] Ibíd. p. 317 -318.

- *Prueba de desempeño:* está diseñada principalmente para probar el desempeño del software en tiempo de ejecución dentro de contexto de un sistema integrado.

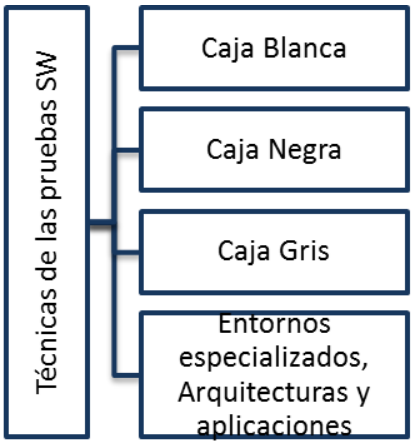
2.2.5 Pruebas de aceptación.

Están son las últimas pruebas que se hacen y consiste en que el usuario interactúe con el software dentro de su propio entorno de trabajo, para que así él pueda dar su visto bueno de acuerdo con la satisfacción de los requerimientos establecidos previamente.

2.3 TÉCNICAS DE LAS PRUEBAS

Las técnicas de las pruebas de software no son más que un conjunto de procedimientos a seguir para obtener los mejores resultados al momento de realizar las pruebas de software. Dentro de estas técnicas se pueden destacar las de caja blanca, negra, gris y las pruebas de entornos especializados.[17] En el siguiente grafico se puede apreciar las principales técnicas de las pruebas:

Figura 2. Técnicas de pruebas.



Fuente: Autores

2.3.1 Pruebas de caja blanca

Las pruebas de caja blanca buscan un examen del detalle procedimental. En esta técnica se prueban las rutas lógicas, los bucles y condiciones, y el estado

[17] Ibíd. p. 285.

del programa en diferentes puntos para asegurarse de que la operación interna del software se ajuste a las especificaciones. Existen dos tipos de pruebas de caja blanca: pruebas de la ruta básica y pruebas de la estructura de control.[18]

2.3.2 Pruebas de caja negra

Las pruebas de caja negra dejan a un lado la parte lógica del sistema y se centra en la interfaz, en los resultados de las entradas, es decir, se centra en los requisitos funcionales del software. Es a su vez un enfoque complementario de las pruebas de caja blanca. Dentro de esta técnica se puede encontrar algunos tipos de prueba como: métodos gráficos, partición equivalente, análisis de valores límite, prueba de tabla ortogonal.[19]

2.3.3 Pruebas de caja gris

Es una combinación de las pruebas de caja blanca y negra, pero su diferencia radica en que las pruebas son diseñadas por personas que conocen la estructura interna del software , por lo tanto lo entienden bien y pueden deducir en que partes se pueden presentar mayor números de errores.

2.3.4 Pruebas de entornos especializados, arquitecturas y aplicaciones.

Debido a la creciente complejidad del software se ha visto la necesidad de tener un enfoque de pruebas especializado dentro del cual se encuentran tipos de pruebas como: pruebas de interfaces graficas de usuarios, arquitectura cliente servidor, sistemas de tiempo real, documentación y facilidades de ayuda, etc.

2.4 DOCUMENTACIÓN DE LAS PRUEBAS

2.4.1 ¿Qué es?

La documentación de las pruebas de software es en esencia el hecho de dejar evidencia de las mismas y sus resultados, por medio de documentos que a su

[18] Ibíd. p. 286 – 294.
[19] Ibíd. p. 294 – 299.

vez permitan llevar un control de las pruebas realizadas para evitar que se repitan, facilitando la preparación de nuevas pruebas.

2.4.2 Importancia de la documentación

La documentación de las pruebas de software es muy importante, ya que:

- Evita la repetición de pruebas y facilita la preparación de nuevas pruebas a partir de otras anteriores.
- Por medio de esta forma de proceder, se puede dejar evidencia de las mismas y sus resultados.
- Conservar la historia, facilita la utilización por parte del usuario, garantiza la permanencia.

2.4.3 Normas y estándares para la documentación de las pruebas de software.

Las normas y estándares más relevantes en la documentación y pruebas de software son:

ISO / IEC 29119 software testing: Es una norma dedicada exclusivamente a la pruebas de software y abarca: definiciones y vocabulario, proceso de prueba, documentación de prueba, técnicas de ensayo y un modelo de proceso de evaluación para los procesos de pruebas de software. Esta norma reemplazara algunos estándares internacionales como: IEEE 829 Documentación de prueba, IEEE 1008 Unidad de Pruebas, BS 7925-1 Vocabulario de términos en Pruebas de Software, BS 7925-2 Componente de pruebas de software estándar.[20]

Estándares británicos BS 7925: Estándar británico para pruebas de componentes de software, describe técnicas para el diseño y la medición de casos de prueba, trata la ejecución y análisis de los resultados, características a seleccionar para determinar, comparar y mejorar la calidad de la prueba**.**

[20] ISO/IEC/IEEE 29119 Software Testing Standard, Recuperado el día 25 de noviembre de 2012, de http://www.softwaretestingstandard.org/

IEEE 829 Documentación de prueba: este estándar describe los documentos que se pueden producir durante el proceso de prueba de software. Las etapas y los documentos que define la norma son las siguientes: Planeación de la pruebas, Especificación del diseño de prueba, Especificación de los Casos de Prueba, Especificación del Procedimiento, Reporte de avance de los ciclos probados, Registro de la prueba, Reporte de incidentes, Sumario de la prueba.[21]

IEEE 1008 Unidad de Pruebas: su principal objetivo es especificar un método para la realización de pruebas unitarias, describe los conceptos de ingeniería de software y pruebas de los supuestos en que está basado el enfoque del estándar.[22]

2.5 IEEE 829 DOCUMENTACIÓN DE PRUEBA

Al momento de querer efectuar las actividades de pruebas se encentran varias normas y estándares que ofrecen pautas importantes para la realización de las pruebas y su documentación, las cuales fueron nombradas en el numeral 2.4.3. No obstante, dentro de estas normas se destaca la IEE 829 ya que esta norma está dispuesta principalmente para las actividades de documentación de pruebas, la cual permite obtener de forma ordenada y precisa todos los datos que arrojan.

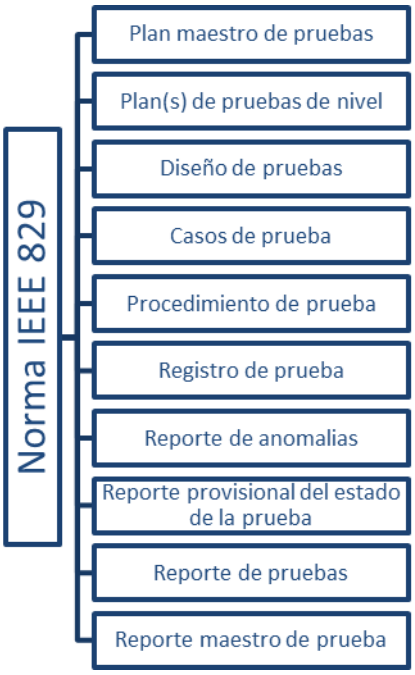
La norma IEEE 829 establece 10 documentos importantes (ver figura 3) que deben ser generados para hacer una correcta y completa documentación de pruebas, que son: Plan maestro de pruebas, Plan(s) de pruebas, Diseño de pruebas, Casos de prueba, Procedimiento de la prueba, Registro de prueba, Reporte de anomalías, Reporte provisional del estado de la prueba, Reporte de pruebas, Reporte maestro de pruebas.

[21] IEEE 1008- 1987. IEEE Standard for Software Unit Testing. E-ISBN: 0-7381-0400-0. INSPEC Accession Number: 2894459
Identificador digital: 10.1109/IEEESTD.1986.81001.

[22] IEEE 829-2008. IEEE Standard for Software Test Documentation. Software Engineering Technical Committee of the IEEE Computer Society.
E-ISBN: 978-0-7381-5746-7. Print ISBN: 978-0-7381-5747-4. INSPEC Accession Number: 10219290. Digital Object Identifier: 10.1109/IEEESTD.2008.4578383.

Por otra parte, la norma ofrece dos formas diferentes de hacer la documentación de las pruebas que se planean realizar, ya que todos los software no poseen la misma complejidad, las pruebas no tienen el mismo nivel de exhaustividad o no siempre se cuenta con un grupo de trabajo lo suficientemente adecuado para desarrollar a cabalidad todas la actividades de documentación registradas en la norma.

Figura 3.Documentos norma IEEE 829.



Fuente: Autores

La primera forma de hacer la documentación según la norma es estableciendo un nivel de integridad para el software a ser probado y de acuerdo con el nivel de integridad que se haya seleccionado esta norma sugiere los documentos que se deben realizar o gestionar durante las pruebas para que la documentación sea completa y precisa. En la tabla 1 se aprecia lo dicho anteriormente.

Tabla 1. Documentación de pruebas de acuerdo a los niveles de integridad

Nivel de integridad	Documentación de pruebas
1Insignificante	Plan(s) de pruebas, Diseño de pruebas, Casos de prueba, Procedimiento de la prueba, Registro de prueba, Reporte de anomalías, Reporte de pruebas.
2 Marginal	Plan(s) de pruebas, Diseño de pruebas, Casos de prueba, Procedimiento de la prueba, Registro de prueba, Reporte de anomalías, Reporte provisional del estado de la prueba, Reporte de pruebas.
3 Crítico	Plan maestro de pruebas, Plan(s) de pruebas, Diseño de pruebas, Casos de prueba, Procedimiento de la prueba, Registro de prueba, Reporte de anomalías, Reporte provisional del estado de la prueba, Reporte de pruebas, Reporte maestro de pruebas.
4 Catastrófico	Plan maestro de pruebas, Plan(s) de pruebas, Diseño de pruebas, Casos de prueba, Procedimiento de la prueba, registro de prueba, Reporte de anomalías, Reporte provisional del estado de la prueba, Reporte de pruebas, Reporte maestro de pruebas.

Fuente: Autores

La segunda forma puede ser llamada como la de libre decisión, es decir, los documentos que el probador o el equipo de pruebas opten por realizar de los diez sugeridos por la norma. Es decisión propia de cada uno y se tendrán en cuenta factores como la complejidad del software, el nivel de exigencia de las pruebas, el equipo de pruebas, etc. Además podrá optar por unir uno o varios documentos e incluir algún otro si fuere necesario. En conclusión, la norma ofrece la facilidad de ser adaptable por el usuario a la forma en la que él se sienta cómodo.

En la tabla 2 se aprecia los documentos sugeridos por la norma y la información que podrían contener cada uno de ellos.

Tabla 2. Documentos de prueba y su información requerida.

Documentos	Información requerida
Plan maestro de pruebas	Identificador de documento, alcance, referencias, vista general del sistema y características generales, resumen de la prueba, procesos de prueba, documentación de prueba, requisitos de administración de prueba, requerimientos del reporte de prueba, glosario, cambio de procedimiento e historia en el documento.
Planes de pruebas	Identificador, alcance, referencias, nivel en la secuencia general, clases de prueba y condiciones generales de prueba, ítems y sus identificadores, matriz de trazabilidad, características a probar, características que no se prueban, enfoque, matriz de prueba, ítems de criterio de paso/fallo, entregables de las pruebas, actividades y tareas planificadas, ambiente/infraestructura, responsabilidades y autoridad, interfaces, recursos y su ubicación, horarios estimaciones y costos, riesgos y su contingencia, procedimientos de aseguramiento de calidad, métricas, cobertura de la prueba, glosario, cambio de procedimiento e historia en el documento.
Diseño de pruebas	Identificador, alcance, referencias, características a probar, cambios en el enfoque, identificador de prueba, características que aprueban/fallan los criterios, entregables de la prueba, glosario, cambio de procedimiento e historia en el documento.
Casos de prueba	Identificador, enfoque, referencias, contexto, notación para la descripción, identificador del caso de prueba, objetivo, entradas, salidas, hardware, software, requisitos con procedimiento especial, dependencia entre casos, glosario, cambio de procedimiento e historia en el documento.
Procedimiento de la prueba	Identificador, enfoque, referencias, relación con otros procedimientos, entradas, salidas, requisitos especiales, descripción de los pasos para tener en cuenta por cada persona, glosario, cambio de procedimiento e historia en el documento.
Registro de la prueba	Identificador, alcance, referencias, descripción, entrada de actividades y eventos, descripción de la ejecución, resultados del procedimiento, información del entorno, eventos anómalos, identificador de reporte de anormalidad, glosario.
Reporte de anomalías	Identificador, alcance, referencias, resumen, fecha de descubrimiento de la anormalidad, contexto, descripción de la anormalidad, impacto, evaluación de la urgencia en el arreglo, descripción de las acciones de corrección, estado de la anormalidad, conclusiones y recomendaciones, cambio de procedimiento e historia en el documento.
Reporte	Identificador, alcance, referencias, resumen del estado de la

provisional del estado de la prueba	prueba, cambios desde el plan, métricas del estado de la prueba, cambio de procedimiento e historia en el documento.
Reporte de pruebas	Identificador, alcance, referencias, resumen de los resultados, detalles de los resultados de la prueba, justificación de las decisiones, conclusiones y recomendaciones, glosario, cambio de procedimiento e historia en el documento.
Reporte maestro de pruebas	Identificador, alcance, referencias, listado de todos los resultados de la prueba, justificación de las decisiones, conclusiones y recomendaciones, glosario, cambio de procedimiento e historia en el documento.

Fuente: Autores

Por último la norma IEE 829 ofrece ocho anexos que son de utilidad para realizar las pruebas entre los cuales se encuentran: bibliografía, ejemplo de esquemas de niveles de integridad, tareas de prueba, ejemplos de documentación, etc.

3. ESTUDIO COMPARATIVO DE HERRAMIENTAS COMPUTACIONALES PARA LA REALIZACION Y DOCUMENTACION DE PRUEBAS DE SOFTWARE

Durante la búsqueda de herramientas se encontraron alrededor de 30 aplicaciones, de las cuales se escogieron 8, debido a que eran las herramientas más relevantes y de fácil acceso. Las elegidas fueron las siguientes:

3.1 JMETER

Apache JMeter puede ser utilizado para probar el rendimiento de los recursos, tanto estáticos como dinámicos (archivos, Servlets, scripts de Perl, objetos Java, bases de datos y consultas, servidores FTP y mucho más). Se puede utilizar para simular una carga pesada en un servidor, probar la resistencia de red o de un objeto o para analizar el rendimiento general bajo diferentes tipos de carga. Se puede utilizar para hacer un análisis gráfico de rendimiento o para probar el comportamiento de un servidor/script/objeto bajo una concurrente carga pesada.[23] Dentro de sus características principales se tienen:

- Se puede cargar y probar el rendimiento de muchos tipos diferentes de servidor:
    - Web - HTTP, HTTPS.
    - SOAP.
    - Base de datos a través de JDBC.
    - LDAP.
    - JMS.
    - Correo - SMTP (S), POP3 (S) e IMAP (S).
    - Comandos nativos o scripts de Shell.
- Portabilidad completa y pureza del 100% Java.
- Framework Completamente multithreading que permite el muestreo simultáneo de muchos procesos y muestreo de las diferentes funciones de grupos procesos separados.

[23] Apache JMETER, Recuperado el 25 de marzo de 2013, de http://jmeter.apache.org/

- Cuidadoso diseño de la GUI que permite un funcionamiento más rápido.
- Almacenamiento y análisis / repetición sin conexión de los resultados de las pruebas.
- Altamente Extensible.

3.2 SMARTESOFT-SMARTESCRIPT

Una potente, robusta y comprobada herramienta automatizada de pruebas funcionales, SmarteScript es fácil de aprender; cuenta con una interfaz amigable con el usuario, no se necesita habilidades de programación, incluso para las funciones más avanzadas del software. Como el código del software bajo prueba puede cambiar, actualiza automáticamente todos los scripts de prueba afectados, por lo que se puede ahorrar tiempo en las pruebas de regresión.[24] Cuenta con las siguientes características principales:

- Flexible y rápido.
- Fácil de usar e implementar.
- Flexibilidad para probar datos variables.
- Mantenimiento rápido.
- Un mundo perfecto: Prueba rápida y confiable.

3.3 TESTMASTER

Testmaster es una herramienta de automatización de pruebas, registro de caso de prueba e informes, similar a cualquier programa comercial para dirigir las pruebas.

La herramienta fue concebida gracias a que su creador tuvo una mala experiencia desarrollando pruebas software, donde no se contaba con la suficiente organización.[25] A continuación se presenta las principales características de Testmaster.

[24] SmarteScript Functional Test Tool, Recuperado el 25 de marzo de 2013, de http://www.smartesoft.com/products_smartescript.php.
[25] Testmaster, Recuperado el 25 de marzo de 2013, de http://testmaster.sourceforge.net/

- Totalmente basado en la web
- Gestión jerárquica de bancos de pruebas (Departamento, Proyecto, banco de pruebas)
- Registro de Testcase (incluyendo el estado (pasa / falla / retención, etc.), notas probador, ids trouble ticket y cambiar el historial).
- Interfaz administrativa.
- Plantillas personalizables web.
- Prueba de interfaz de automatización

3.4 SONAR

Sonar es una plataforma para evaluar código fuente. Sonar, que es software libre usa diversas herramientas de análisis estático de código fuente como Checkstyle, PMD o FindBugs para obtener métricas que pueden ayudar a mejorar la calidad del código de un programa.[26] Se presentan algunas características de Sonar:

- Ofrece informes de código duplicado, normas de codificación, pruebas unitarias, cobertura de código, código complejo, errores potenciales, comentarios, diseño y arquitectura.
- Idioma principal compatible Java. Aunque otros idiomas son compatibles con las extensiones.
- Se integra con Maven, Ant y herramientas de integración continua (Atlassian Bamboo, Jenkins, Hudson).
- Extensible con el uso de plugins.
- Diseño y Arquitectura de las dimensiones.

[26] Sonar | Marco de Desarrollo de la Junta de Andalucía, Recuperado el 25 de marzo de 2013, de http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/372

3.5 SELENIUM

Selenium es un set de herramientas que permiten desarrollar scripts para pruebas automatizadas, para aplicaciones Web en diversos lenguajes como Java, Ruby, Python, Perl, .Net o PHP.

Selenium IDE es un plugin para Firefox que permite grabar y ejecutar scripts directamente desde nuestro navegador. Las pruebas que permite realizar Selenium son principalmente pruebas unitarias y de integración.[27] Sus principales características son:

- Una amplia Barra de Herramientas.
- Lenguajes de programación: permite hacer pruebas de diferentes lenguajes de programación web, tales como Java, Ruby, Python, Perl, .Net o PHP.
- Casos o conjuntos de casos de prueba.
- Modo recording: Con lo que el programa guarda automáticamente las acciones que hace el usuario en la página a la que se le va a hacer la prueba.
- Muestra el progreso de la prueba.
- Instrucciones: Gran cantidad de instrucciones con su respectiva documentación, la cual ilustra su funcionalidad y los parámetros o valores que esta requiere.

3.6 WATIR

Watir es un conjunto de herramientas de código abierto que controla navegadores y ayuda a automatizar partes del testing funcional en aplicaciones web. Watir está implementado en Ruby, pero esto no significa que solo pueda manejar aplicaciones web escritas en Ruby. No importa que sistema operativo (Windows, Linux o Mac), lenguaje de programación (C#, Java, PHP, Ruby…) o que framework utilice la aplicación web a probar, seguramente se puede probar

[27] Selenium - Herramienta para automatizar pruebas de software ~ Programando con Café Recuperado el 3 de abril de 2013, de http://www.programandoconcafe.com/2011/10/selenium-herramienta-para-automatizar.html

con Watir.[28] Como de costumbre a continuación se presentan las principales características:

- Para poder utilizar Watir se debe tener instalado Ruby, el kit de desarrollo en Ruby y la versión más reciente de RubyGems.
- Utilizando Watir se tiene acceso a todas las características de Ruby, por lo que se tiene una gran colección de comandos y funcionalidades para utilizar en los scripts de pruebas.
- Con Watir se puede hacer prácticamente cualquier cosa que un usuario hace con un navegador (abrir una página, clic en un link, botón, radio botón, etc.).

3.7 RATIONAL FUNTIONAL TESTER

IBM Rational Functional Tester es una herramienta para la realización de pruebas funcionales y de regresión automatizadas. Este software proporciona funciones de pruebas automatizadas para pruebas funcionales, de regresión, de GUI y basadas en los datos. Rational Functional Tester da soporte a diversas aplicaciones, como aplicaciones basadas en web, .Net, Java, Siebel, SAP, basadas en emulador de terminal, PowerBuilder, Ajax, Adobe Flex, Dojo Toolkit, GEF, documentos Adobe PDF, zSeries, iSeries y pSeries.[29] Posteriormente se presentan las principales características de esta herramienta:

- De gran ayuda para las pruebas dinámicas, contiene múltiples puntos de verificación y además cuenta con el uso de expresiones regulares para el mejor funcionamiento de las pruebas.
- La característica de registrar las acciones del usuario en un script de prueba, con lo que se facilita la codificación de las pruebas automáticas, generando un script para pruebas posteriores.

[28] Watir.com | Web Application Testing in Ruby Recuperado el 3 de abril de 2013, de http://watir.com/
[29] IBM - Rational Functional Tester - Colombia Recuperado el 3 de abril de 2013, de http://www-03.ibm.com/software/products/co/es/functional/

- Mientras se está ejecutando el recording de la prueba, el programa se minimiza a una pequeña interfaz para sacar un mejor provecho del espacio en pantalla del equipo.

3.8 TESTLINK

Test link forma un repositorio de requisitos y casos de prueba, y los relaciona con los productos, las plataformas y el personal. Se asignan pruebas al personal y se registran los resultados. Todo esto para que con una gran variedad de informes se logre proporcionar información sobre lo que se ha hecho y lo que aún queda por hacer.[30] Finalmente se presentan las características principales de esta herramienta:

- Soporte para diferentes productos.
- Creación de diferentes roles con distintos permisos para los integrantes del equipo de testing.
- Creación ilimitada de carpetas en forma de árbol (llamadas testsuites) para una mejor organización y agrupamiento de los casos de prueba.
- Versionado de casos de prueba.
- Ejecución de los casos de prueba por versión del software bajo prueba.
- Creación de test plans para la ejecución.
- Asignación de casos (para ejecución) a los usuarios, de modo que en un mismo test plan se puede tener varios testers trabajando y ellos sólo ven sus casos de prueba asignados.

3.9 OTRAS CARACTERÍSTICAS

Por otro lado para esta revisión también se tuvo en cuenta algunos aspectos importantes de las herramientas como son: la plataforma, el manual de usuario, forma de adquisición y registro de pruebas. Lo cual se puede ver en la tabla 3.

[30] TestLink Home Recuperado el 3 de abril de 2013, de http://testlink.sourceforge.net/docs/testLink.php

Tabla 3. Características generales de las herramientas.

Nombre Herramienta	Plataforma	Manual	Es Pago	Registro de pruebas
JMeter	Independiente.	Si	Libre y de código abierto.	Ofrece reportes, gráficas de resultados, todos los datos de la prueba y tablas estadísticas como el promedio de tiempo, la desviación, numero de errores etc.
smartesoft-smartescript	Independiente	Si	Paga	La información proporcionada por el proveedor del software, no define que registro de pruebas hace.
Testmaster	Linux	Si	Libre	Muestra reportes, deja ver el estado de las pruebas y se pueden tomar mediciones secuencialmente. Guarda casos de prueba, que posteriormente pueden ser editados y consultados por el usuario.
Sonar	Independiente	Si	Libre	No maneja registro de pruebas y solo analiza código fuente.
Selenium	Independiente	Si	Libre	Se guardan casos de prueba o si el usuario lo prefiere conjuntos de casos de pruebas software. Para este caso particular un caso de prueba es una serie de comandos realizados por el usuario con el objetivo de comprobar el funcionamiento correcto de una página web.
Watir	Independiente	Si	Libre	No recolecta ninguna información, aunque se puede guardar un script con el test a probar, este script se guarda con extensión .rb.
Rational Funtional Tester	Windows, Linux	Si	Pago (tiene versión de prueba)	Se genera un reporte al final de la prueba, con información del test, tales como warnings, failures, succes and error messages. Con gráficos y dibujos detallados de cada uno de estos.
TestLink	Windows, Linux	Si	Libre	Test link recolecta variada información de la realización de las pruebas, entre ellas está la versión del programa que se evalúa, el estatus de la prueba si se pasó, fallo, no compila o está bloqueada, la fecha de inicio, fecha de finalización, quien ejecuto la prueba, quien la asignó y los requisitos del sistema con una descripción de si se cumplieron o no, estos entre otros temas tratados más a profundidad.

Fuente: autores

3.10 Ventajas y desventajas identificadas.

Ventajas.

En forma general puede decir que estas herramientas tienen la ventaja de ofrecer muchas funcionalidades para la realización de pruebas, abarcando ampliamente sus áreas de aplicación, ya sean entornos web o para código fuente. Además tienen una estructura correcta para el buen diseño y realización de pruebas permitiéndole al usuario crear varios casos de prueba o paquetes de casos de prueba.

Desventajas.

La principal desventaja encontrada fue el hecho de que estas herramientas han sido diseñadas más específicamente para realizar pruebas, más no para documentarlas. Aunque algunas de ellas documenten alguna parte de las pruebas como los casos de pruebas, no es suficiente con solo documentar este nivel de pruebas. Por lo tanto se hace más razonable la necesidad de una herramienta específica para esta labor.

3.11 Funcionalidades que se tomaron en cuenta para la realización del software.

Las herramientas que hay actualmente en el mercado están orientadas a la realización de pruebas automatizadas, por esta razón se dificulto la labor de encontrar alguna funcionalidad que pudiera servir al momento de desarrollar el software propuesto, excepto por el buen manejo de los casos de prueba con el que cuentan algunas de ellas, lo cual podría ser acoplado, dependiendo de la flexibilidad de la norma.

4. HERRAMIENTA COMPUTACIONAL PARA LA EVALUACIÓN DE LA CALIDAD SOFTWARE QUIS: REVISIÓN DE FUNCIONALIDAD.

En este capítulo se describirá brevemente el estado y las características, en la cual se encuentra la herramienta para la evaluación de la calidad (Quality UIS – QUIS) en su versión 2.0. Posteriormente se presentarán un resumen de lo encontrado y lo corregido en la herramienta.

4.1 ¿Qué es QUIS 2.0?

QUIS 2.0 es una aplicación computacional, desarrollada con el propósito de ayudar en la gestión, evaluación y documentación de la calidad en productos software derivados de actividades de investigación universitaria.

Esta desarrollada en Visual Basic .NET y contiene una pequeña base de datos en Access para facilitar el manejo de la información de los modelos de calidad.

4.2 Estructura funcional de QUIS 2.0.

QUIS 2.0 está compuesto de varios formularios, los cuales permiten administrar la información generada a partir de la gestión y evaluación de calidad software a partir de tres enfoques planteados por el sistema: como Proyecto, como Proceso y como Producto.

4.3 Caracterización del desarrollo de QUIS 2.0

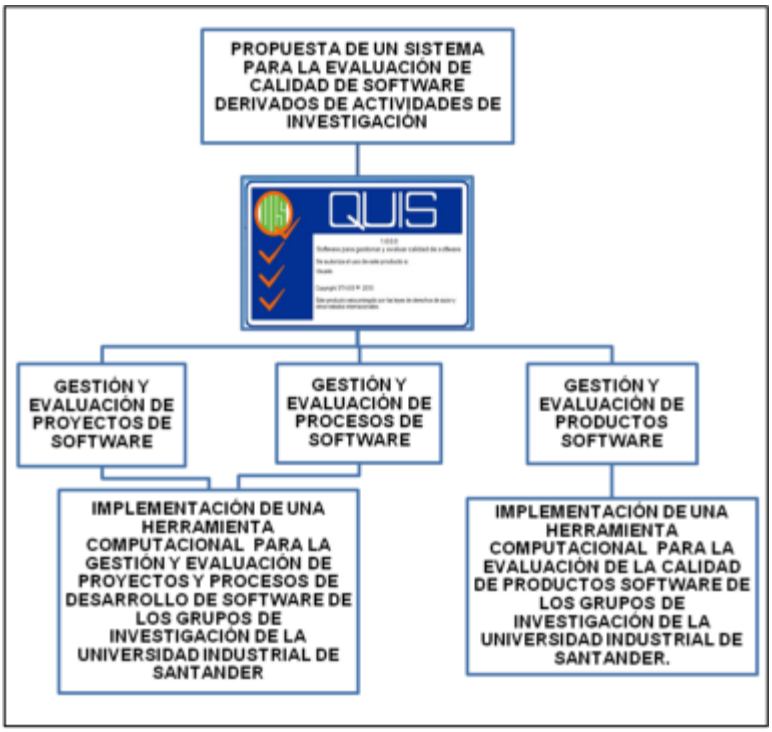
QUIS como aplicación, surge como resultado de una investigación desarrollada por el grupo de investigación en Sistemas y Tecnologías de la información (STI) de la UIS. De la cual hacían parte un proyecto de maestría y dos de pregrado. En la figura 4 se relacionan los trabajos de grado con los módulos desarrollados para la herramienta:

- Implementación de una herramienta computacional para la gestión y evaluación de proyectos y procesos de desarrollo de software de los grupos de investigación de la Universidad Industrial de Santander. [31]

[31] Pimentel, Jorge Iván. Implementación de una herramienta computacional para la gestión y evaluación de proyectos y procesos de desarrollo de software de los grupos de investigación

- Implementación de una herramienta computacional para la evaluación de la calidad de productos software de los grupos de investigación de la Universidad Industrial de Santander. [32]
- Propuesta de un sistema para la evaluación de calidad de software derivado de actividades de investigación.[33]

Figura 4. Proyectos asociados a la implementación de QUIS 2.0.



Fuente: [34]

---

de la Universidad Industrial de Santander. Bucaramanga: 2011. Presentada en la Universidad Industrial de Santander para obtención del título de Ingeniero de sistemas.

[32] Pinto, Nelson. Implementación de una herramienta computacional para la evaluación de la calidad de productos software de los grupos de investigación de la Universidad Industrial de Santander. Bucaramanga: 2011. Presentada en la Universidad Industrial de Santander para obtención del título de Ingeniero de sistemas.

[33] LEÓN, Nelson. Propuesta de un sistema para la evaluación de calidad de software derivado de actividades de investigación. Bucaramanga: 2011. Presentada en la Universidad Industrial de Santander para obtención del título de Magister en Ingeniería área de Informática y Ciencias de la Computación.

[34] Ibíd.p.69

4.4 Visión general del estado actual de la herramienta e identificación de errores.

Se desarrolló una revisión con el fin de descubrir el estado en el que se encontraba la aplicación QUIS 2.0. Para dicha revisión se tuvo en cuenta el manual de usuario, el instalador, el ejecutable y el código fuente del programa. Se distribuyó a cada uno de los integrantes del proyecto, un módulo del programa a probar y así poder detectar los posibles defectos que pudiesen existir en la aplicación. Cabe aclarar que dichos módulos se rotaban entre los integrantes con el fin de que cada integrante tuviese la posibilidad de probar el software en su totalidad.

De esta revisión se pudo encontrar un número considerable de defectos, por lo que se decidió hacer una clasificación en tres categorías, tal y como se presenta a continuación:

- ***Funcionales:*** Son aquellos defectos que comprometen el objetivo o función de alguna de las partes o del programa en general. Eje: En el módulo de ejecución de la evaluación se guardaban parámetros con valores por encima de los límites especificados.
- ***Lógicos:*** Son aquellos defectos en el código fuente que no afectan la funcionalidad, pero que de algún modo entorpecen el uso del software. Eje: En el módulo Modelo de calidad, en la opción Editar al querer actualizar y darle la opción NO, muestra un mensaje, que indica que no se le ha asociado ninguna característica al modelo, aun cuando se hayan seleccionado varias características.
- ***Visual:*** Son aquellos defectos pertenecientes a la parte gráfica del programa, tales como algún error ortográfico, dificultades al entender la funcionalidad del software, defectos en el manual de usuario, entre otros. Eje: Error de tipo ortográfico cuando se saca el reporte del módulo ver evaluación en la tabla de responsables de la evaluación.

En total en la revisión de la aplicación se encontraron 86 defectos, los cuales se clasificaron de la siguiente forma:

Tabla 4. Clasificación defectos encontrados

Clasificación	Cantidad
Funcionales	32
Lógicos	29
Visuales	25
TOTAL	**86**

Fuente: Autores

4.5 Registro de las solicitudes de cambio identificadas.

Los defectos encontrados durante la evaluación hecha a la herramienta QUIS 2.0 fueron registrados gracias a la misma. Cada vez que se encontraba un defecto se registraba como una solicitud de cambio en un proyecto creado en QUIS llamado QUIS 2.0, por medio de un formulario llamado solicitudes de modificación, con el cual se pudo llevar un control de los defectos encontrados. Además de esto, este registro fue de gran ayuda al momento de la corrección de los mismos ya que a cada una de las solicitudes registradas anteriormente se le asignaba una respuesta después de corregida.

4.6 Corrección de errores que el grupo de trabajo ha considerado pertinentes.

Los autores realizaron un extenso trabajo en la corrección de los defectos encontrados en la aplicación. Después de registrados gracias a la herramienta, fueron clasificados según el impacto que tenían en el buen funcionamiento de la misma. Se clasificaron como defectos de bajo, medio y alto impacto. Después de clasificados se procedió a resolverlos en su totalidad, comenzando por los de alto impacto y terminando con los de bajo impacto.

Cabe resaltar que gracias al uso de la herramienta se pudo llevar un control adecuado de los defectos corregidos y de los nuevos que surgieron durante la corrección de los anteriores, que a su vez también fueron corregidos. Gracias a este registro también fue registrada información adicional como: las causas, la

fecha de detección, la persona encargada de corregirlo, una solución parcial para el mismo, entre otros.

5. DISEÑO E IMPLEMENTACIÓN DE LA HERRAMIENTA COMPUTACIONAL PARA LA DOCUMENTACIÓN DE PRUEBAS SOFTWARE.

Posterior a la revisión de las normas más importantes, de las herramientas computacionales que cumplen una función similar a la herramienta que se quiere implementar y del estado del software QUIS en su versión 2.0, en este capítulo se muestra el diseño del sistema de información que facilitara la documentación de las pruebas software de desarrollos derivados de actividades de grupos de investigación universitarios. En la segunda sección del mismo se describe brevemente las características más importantes de la herramienta computacional que se basa en el sistema propuesto, la cual se denomina QUIS 3.0.

5.1 DISEÑO DE UN SISTEMA PARA LA DOCUMENTACIÓN DE PRUEBAS SOFTWARE.

5.1.1 Límites del sistema.

El diseño del sistema se inicia con la definición de sus límites. Para el caso de estudio, el sistema general se denomina *Estructura investigativa* dentro del cual se enmarcan diferentes componente que identifican la forma como se relacionan las diferentes entidades que participan en el desarrollo y realización de pruebas software en un proyecto de investigación. Dentro de estas entidades se encuentran:

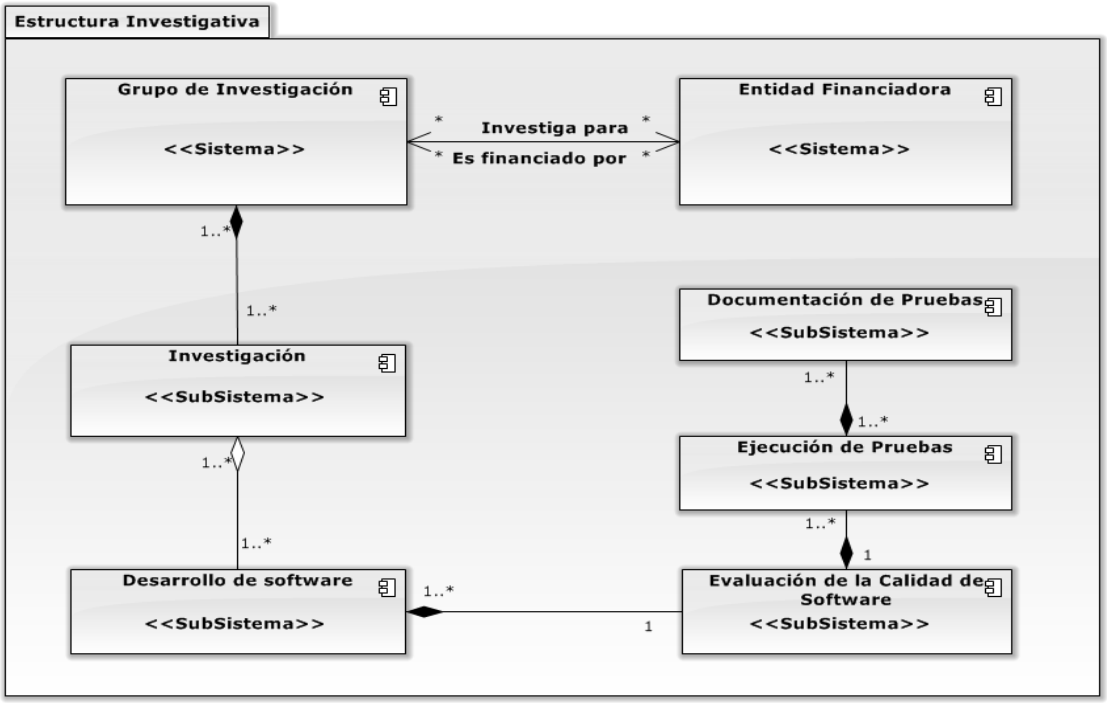
- **Grupo de investigación**: Es aquella organización encargada de llevar a cabo proyectos de investigación, produciendo resultados de conocimiento tangibles y verificables, a partir de la utilización de recursos de personal, infraestructura, tiempo y dinero asignados. Algunas veces los proyectos son planteados desde el interior del grupo para solucionar problemáticas identificadas a partir de la experiencia de sus integrantes o de la literatura y otras veces surgen de necesidades planteadas por entidades externas.

- **Entidad financiadora:** Entidad externa a los grupos de investigación que aporta algunos recursos para desarrollar las investigaciones y que se ve beneficiada con los resultados de esta.
- **Investigación:** Es el conjunto de actividades realizadas con el fin de obtener nuevo conocimiento que en algunas ocasiones es aplicado para la solución de una problemática planteada. Son unidades estructurales que componen el trabajo en los grupos de investigación y cuentan con roles y recursos propios, se desarrollan actividades y se obtienen productos. Una investigación puede llevarse a cabo por uno o más grupos de investigación.
- **Desarrollo software:** Es una de las actividades que pueden ser incluidas dentro del desarrollo de una investigación y de cuya ejecución se obtienen productos que servirán de apoyo para obtener otros resultados dentro de una investigación.
- **Evaluación de la calidad de software:** Es un conjunto de actividades tendientes a evaluar si un desarrollo informático cumple con las necesidades expresadas en la investigación que necesitan de un soporte informático para ser satisfechas.
- **Ejecución de las pruebas:** Es un conjunto de actividades cuyo objetivo es comprobar el correcto funcionamiento del software. Las pruebas software son consideradas parte importante del aseguramiento de la calidad software y en prácticamente todos los desarrollos se realizan pruebas.
- **Documentación de las pruebas**: Es la actividad de proporcionar información a posteriori, donde se deja una constancia del proceso de realización de las pruebas especificando que se hizo y sus resultados.

Sobre este último componente se centrara la estructura del sistema.[35]

[35] N. del A.: La estructura basada en los componentes anteriores del sistema se puede encontrar en. Ibíd.p.58-67

Figura 5. Estructura del sistema

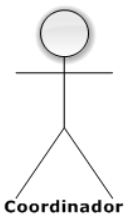
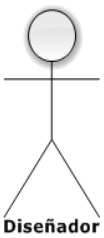


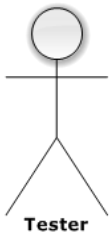
Fuente: Autores

5.1.2 Roles del sistema.

Una vez definidos los límites del sistema, corresponde definir los roles que se desempeñan en el sistema, las funciones que cumplen cada uno y por quien es desempeñado. Esto se resume en la tabla 5.

Tabla 5. Roles del sistema

Rol	Funciones	Desempeñado por
	• Dirigir y organizar la adecuada documentación de las pruebas. • Aprobar los resultados intermedios y finales de las pruebas.	• Director de grupo • Jefe de línea
	• Planear, estructurar y crear las pruebas.	• Auxiliar de investigación • Jefe de línea

	• Ejecutar las pruebas. • Documentar los resultados de las pruebas.	• Auxiliar de investigación • Personal externo

Fuente: Autores

5.1.3 Estructura general del sistema de documentación de pruebas.

La estructura general del sistema se puede dividir en cuatro partes:

Planificación: Comprende lo relacionado con los procesos de planeación de las pruebas, concluyendo con la generación de los siguientes documentos:

- Plan maestro de pruebas
- Plan de pruebas de nivel

Desarrollo: Es el conjunto de actividades que permiten diseñar y producir las pruebas, por medio de la cual se establece la forma en la que se van a realizar las pruebas, se crean los casos de pruebas y los procedimientos necesarios. Los documentos que deben generarse al final son:

- Casos de prueba
- Procedimiento de prueba

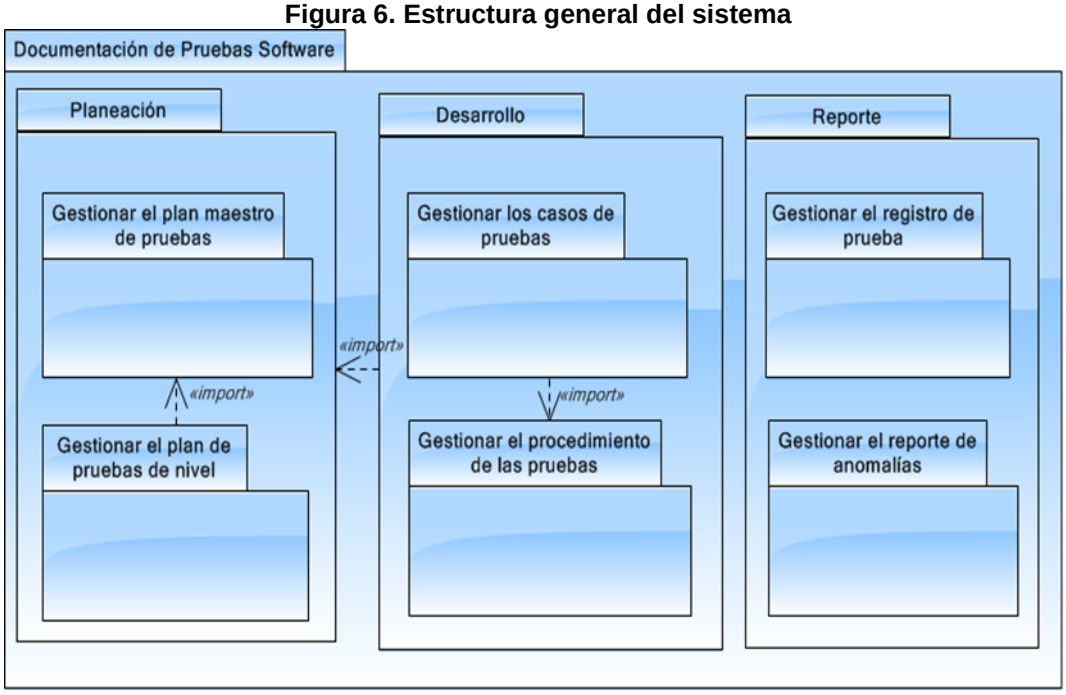
Reporte: Comprende las actividades de ejecución de las pruebas. Como los registros de actividades y anomalías presentadas durante el proceso. Los documentos que deben generarse son:

- Registro de la prueba
- Reporte de anomalías de la prueba

Reporte general: Corresponde a las actividades necesarias para generar los reportes de las pruebas que ya se han desarrollado durante el proceso de prueba. Permite la generación de reportes de todos los documentos anteriores.

La estructura general se puede observar en la figura 6.

Figura 6. Estructura general del sistema



Fuente: Autores

5.1.4 Actividades del sistema para la documentación de las pruebas software.

Las actividades principales para la documentación de pruebas de software se pueden dividir en cuatro bloques que conforman el sistema, como se observa en la siguiente tabla.

Tabla 6. Bloques para la documentación de las pruebas software.

Bloque.	Descripción.
A.1 Planear la pruebas	Su propósito es proporcionar una planificación de la prueba y de los documentos de gestión de la misma.
A.2 Desarrollo de las pruebas	Es el conjunto de actividades que permiten diseñar y producir las pruebas, por medio de la cual se establece la forma en la que se van a realizar las pruebas, se crean los casos de pruebas y los procedimientos necesarios.
A.3 Reporte de las pruebas	Comprende las actividades de ejecución de las pruebas. Como los registros de actividades y anomalías presentadas durante el proceso.
A.4 Generar los reportes de las pruebas	Corresponde a las actividades necesarias para generar los reportes de las pruebas

	que ya se han desarrollado durante el proceso de prueba.

Fuente: Autores

De igual forma el Bloque *Planear las pruebas* se divide en dos actividades:

Tabla 7. Actividades del bloque *Planear las pruebas.*

Actividad.	Descripción.
A.1.1 Gestionar el Plan maestro de pruebas	El plan maestro de pruebas comprende la selección de los elementos que constituyen el esfuerzo de las pruebas, estableciendo los objetivos para cada parte, establecer la división de trabajo (tiempo y recursos), el alcance, las interacciones entre las partes y la identificación de los riesgos.
A.1.2 Gestionar el Plan de pruebas de nivel	Se debe especificar por cada plan de pruebas de nivel los ítems a ser probados, las características y las tareas de prueba a implementar, el personal responsable para cada tarea.

Fuente: Autores

La actividad *Gestionar el Plan maestro de pruebas* consta de las siguientes subactividades:

Tabla 8. Subactividades de la actividad *Gestionar el Plan maestro de pruebas.*

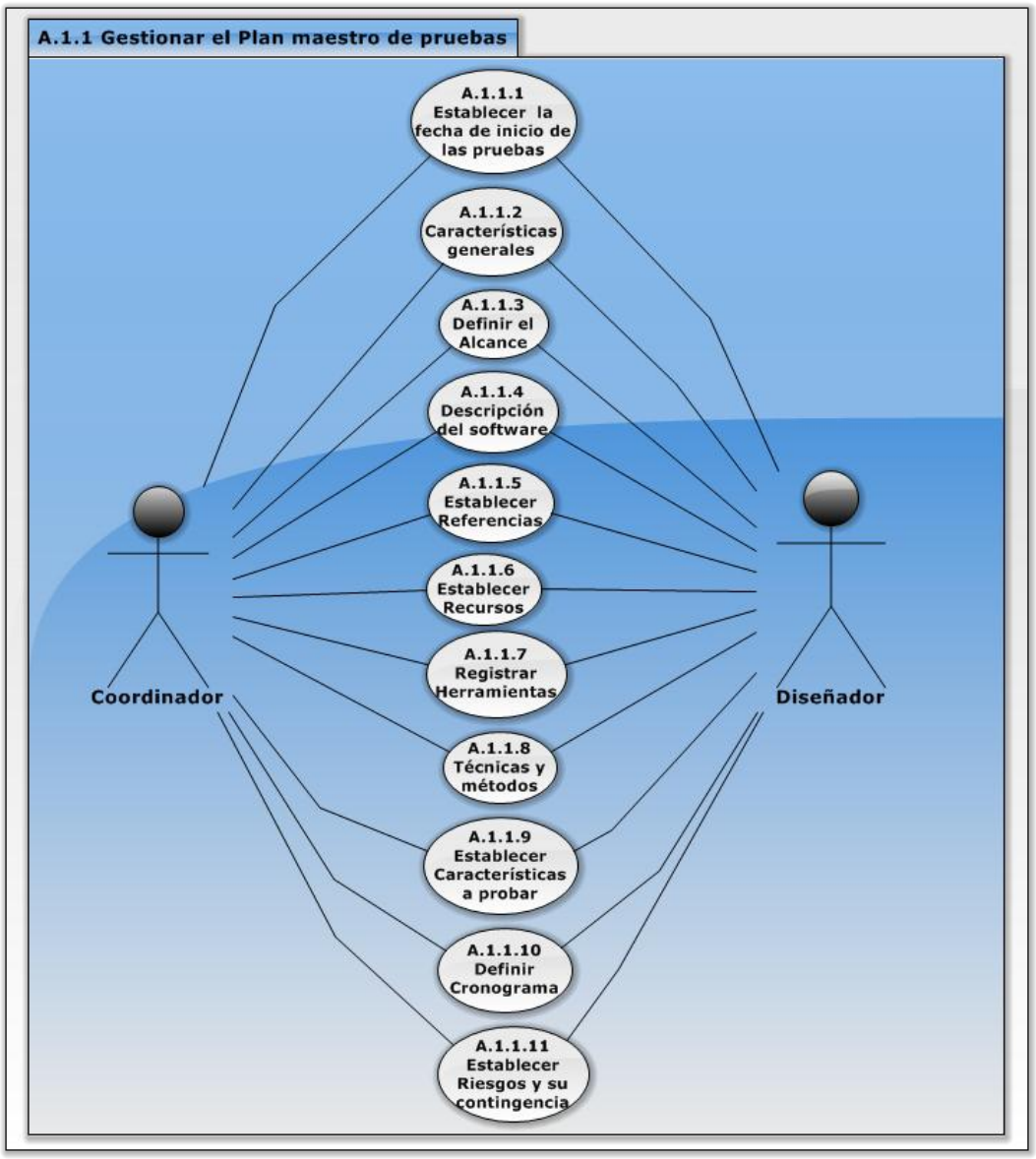
Actividad.	Descripción.
A.1.1.1 Establecer la fecha de inicio de las pruebas	Se establece la fecha de inicio de las pruebas, la cual no debe ser menor que la de inicio del proyecto ni mayor que la de la finalización del mismo.
A.1.1.2 Describir el nombre, versión y estado del software a ser evaluado	Se deben describir las características generales del software como su nombre, versión y estado (inicio , desarrollo, terminado)
A.1.1.3 Definir el Alcance	Describir el producto software o los ítems del sistema y característica que se probaran en los diferentes niveles de pruebas. Indicar el tipo de software.
A.1.1.4 Descripción del software	Realizar una descripción de la misión del software que se está probando(o referencia a esa información) y describe las características generales(o referencia a esa información).
A.1.1.5 Establecer Referencias	Realizar una lista de todos los documentos referenciados. separados en dos: referencias externas y referencias

	internas
A.1.1.6 Establecer Recursos	Describir los recursos de las pruebas, incluyendo personal y herramientas para la gestión de las pruebas.
A.1.1.7 Registrar Herramientas	Describir las herramientas software a utilizar para la realización de las pruebas. Incluir información acerca de la adquisición, la plataforma y una descripción general, para cada herramienta.
A.1.1.8 identificar técnicas y métodos	Describir técnicas y métodos usados en el proceso de prueba. Describe las técnicas a usar para identificar y capturar el testware (documentación) reutilizable.
A.1.1.9 Establecer Características a probar	Identificar todas las características del producto software o del sistema que se van a probar, además de las combinaciones entre estas. Estas se importan del módulo de evaluación de calidad del producto.
A.1.1.10 Definir Cronograma	Definir los eventos de prueba. Estimar el tiempo requerido para cada tarea de test. Especificar el cronograma para cada test task y evento de test. Para cada recurso del test (eje: instalaciones, herramientas y personal, especificar el periodo de uso.
A.1.1.11 Establecer Riesgo(s) y su contingencia	Identificar los problemas de riesgos que puedan afectar negativamente la correcta finalización de las actividades de prueba planeadas. Especifica el impacto potencial de cada riesgo, con planes de contingencia que mitigan o evitar el riesgo.

Fuente: Autores

En la figura 7 se muestra un gráfico resumen de las subactividades y roles involucrados en la actividad *Gestionar el Plan maestro de pruebas.*

Figura 7. Subactividades de la *Gestión del Plan maestro de pruebas.*



Fuente: Autores

La actividad *Gestionar el Plan de pruebas de nivel* consta de las siguientes subactividades:

Tabla 9. Subactividades de la actividad *Gestionar el Plan de pruebas de nivel.*

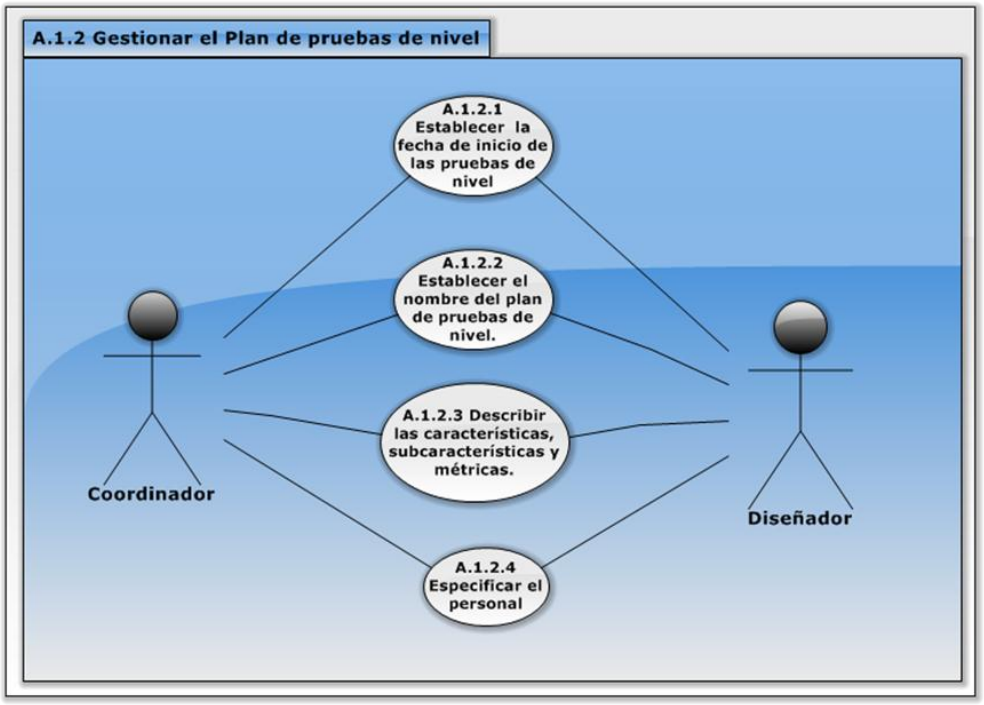
Actividad.	Descripción.
A.1.2.1 Establecer la fecha de inicio de las pruebas de nivel	Establecer la fecha de inicio de las pruebas de nivel, la cual no debe ser menor que la de inicio de la pruebas del plan maestro de pruebas ni mayor que la de la finalización del proyecto.
A.1.2.2 Establecer	Establecer el nombre del producto software que se quiere

el nombre del plan de pruebas de nivel.	evaluar (código fuente, clases, módulos, etc.)
A.1.2.3 Describir las características, subcaracterísticas y métricas.	Identificar todas las características y subcaracteristicas que se quieren evaluar para cada producto. Las métricas que se van a usar, incluyendo su fórmula y parámetros.
A.1.2.4 Especificar el personal	Realizar una lista del personal encargado de llevar a cabo cada plan de nivel.

Fuente: Autores.

En la figura 8, se puede apreciar las subactividades y roles de la actividad *Gestionar el Plan de pruebas de nivel.*

Figura 8. Subactividades de la *Gestión del Plan de pruebas de nivel.*



Fuente: Autores

De forma similar el grupo de *Desarrollo de las pruebas* consta de las siguientes actividades:

Tabla 10. Actividades Grupo de desarrollo

Actividad.	Descripción.
A.2.1 Gestionar los casos de prueba	Definir (con un apropiado nivel de detalle) la información necesaria en lo que respecta a las entradas, salidas y entorno del software que se está

	probando.
A.2.2 Gestionar el procedimiento de la prueba	Especificar los pasos para ejecutar una serie de casos de prueba o generalmente los pasos usados para evaluar un grupo de características.

Fuente: Autores

Así mismo la actividad *Gestionar los casos de prueba* consta de las siguientes subactividades:

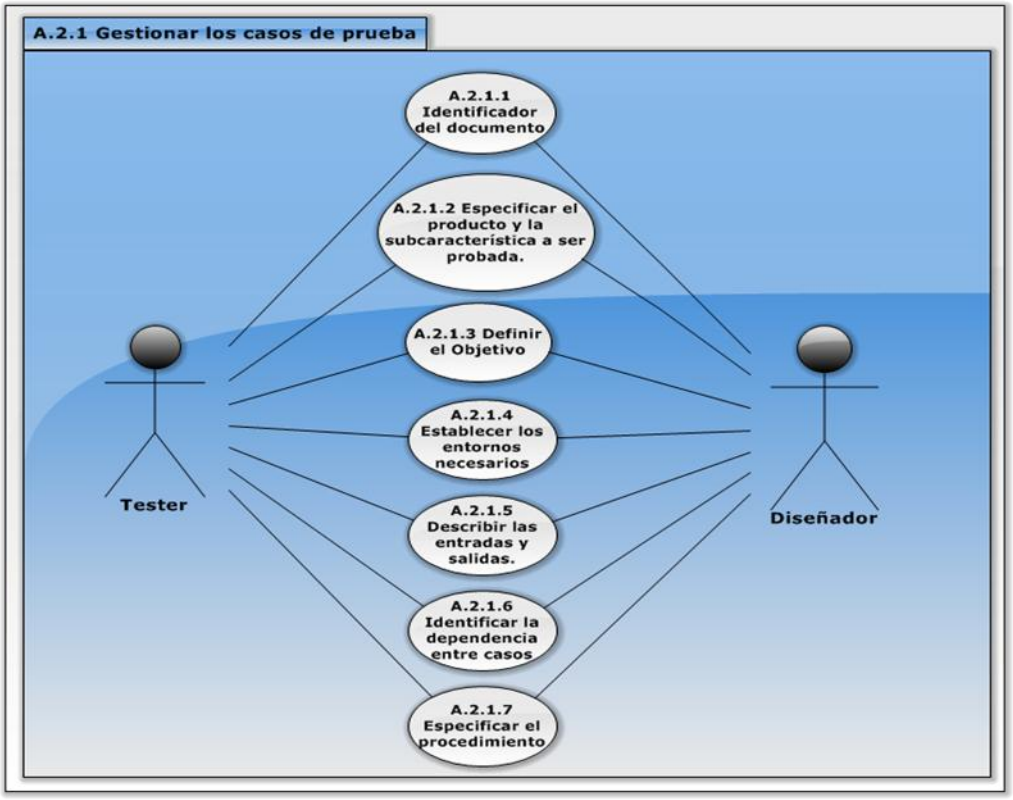
Tabla 11. Subactividades de la actividad *Gestionar los casos de prueba.*

Actividad.	Descripción.
A.2.1.1 Identificador del documento	Crear un identificador único necesario para que cada test case, se pueda distinguir de los demás.
A.2.1.2 Especificar el producto y la subcaracterística a ser probada.	Especificar el producto (plan de prueba de nivel) y la subcaracterística para la cual se está estableciendo el caso de prueba.
A.2.1.3 Definir el Objetivo	Identificar y describir brevemente el enfoque especial u objetivo para cada caso de prueba. Este es más detallado.
A.2.1.4 Establecer los entornos necesarios	Describir el entorno de prueba necesario para la ejecución del test y la documentación de resultados. Características y configuración de hardware requerida para el caso de prueba. La configuración de software requerida para ejecutar el caso de prueba, Puede incluir sistemas operativos, compiladores, simuladores y herramientas de prueba de software, además de cualquier otro requisito que no haya sido incluido (necesidades específicas en las instalaciones, personal especialmente capacitado u medio ambiente de otro fabricante).
A.2.1.5 Describir las entradas y salidas.	Especificar cada entrada requerida para ejecutar el caso de prueba. Algunas entradas se especificaran por valor (con tolerancias cuando sea el caso), mientras que otros, tales como tablas de constantes o archivos transaccionales, se especificaran por nombre. Identificar todas las bases de datos apropiadas, archivos, mensajes de terminal, espacio en memoria y valores transmitidos por el sistema operativo. Especificar todas las salidas y su comportamiento esperado (eje: tiempo de respuesta) para los ítems de prueba.
A.2.1.6 Identificar la dependencia entre casos	Seleccionar los identificadores de los test cases que se deben ejecutar antes del caso de prueba que se está gestionando.
A.2.1.7 Especificar el procedimiento.	Describir el procedimiento específico para el caso de prueba, cuando sea diferente al procedimiento general.

Fuente: Autores

A continuación se presenta gráficamente subactividades y roles presentes en la actividad *Gestión de los casos de prueba.*

Figura 9. Subactividades de la *Gestión de los casos de prueba.*



Fuente: Autores

Por otra parte la actividad *Gestionar el procedimiento de la prueba* consta delas siguientes subactividades:

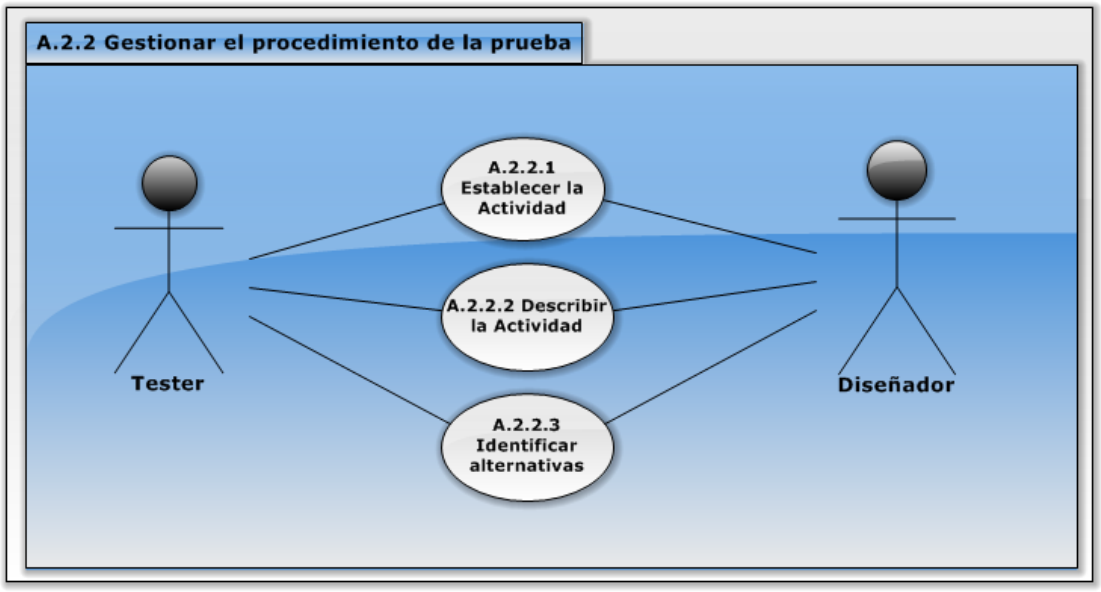
Tabla 12. Subactividades de la actividad *Gestionar el procedimiento de la prueba.*

Actividad.	Descripción.
A.2.2.1 Establecer la Actividad	Identificar y enunciar las actividades generales necesarias para llevar a cabo cualquier caso de prueba.
A.2.2.2 Describir la Actividad	Hacer una breve descripción de cada actividad, donde se especifique claramente los instrumentos necesarios para llevarla a cabo y su importancia dentro de la correcta ejecución del caso de prueba.
A.2.2.3 Identificar alternativas	Identificar posibles acciones a tomar cuando no sea posible llevar a cabo la actividad descrita.

Fuente: Autores

A continuación se presenta gráficamente subactividades y roles presentes en la actividad *Gestión del procedimiento de las pruebas.*

Figura 10. Subactividades de la actividad *Gestión del procedimiento de las pruebas.*



Fuente: Autores

Del bloque *Reporte de las pruebas* consta delas siguientes actividades:

Tabla 13. Actividades del bloque *Reporte de las pruebas.*

Actividad.	Descripción.
A.3.1 Gestionar el registro de pruebas	Proveer un registro cronológico acerca de la realización de la prueba, registrando la medición de los parámetros y la fecha de su medida.
A.3.2 Gestionar el reporte de anomalías	Documentar cualquier evento que ocurra en la ejecución de la prueba y requiera investigación. Estos pueden llamarse un problema, incidente de prueba, defecto, problema, error o anormalidad.

Fuente: Autores

De igual manera la actividad *Gestionar el registro de prueba* consta de las siguientes subactividades:

Tabla 14. Subactividades de la actividad *Gestionar el registro de prueba.*

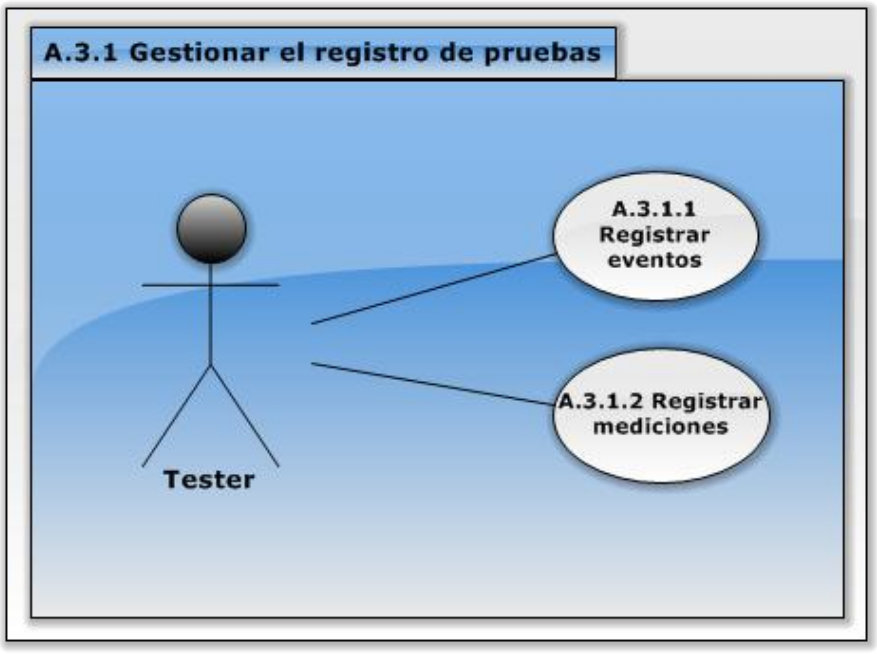
Actividad.	Descripción.
A.3.1.1 Registrar eventos	Registrar actividades/eventos para cada detalle relevante, incluyendo el inicio y final de la actividad, con la fecha respectiva.

A.3.1.2 Registrar mediciones	Especificar la medición de los parámetros teniendo en cuenta el producto, característica, subcaracteristica, parámetro, valor y la fecha en la que se hizo la medición.

Fuente: Autores

A continuación se presenta gráficamente subactividades y roles presentes en la actividad *Gestión del registro de prueba.*

Figura 11. Subactividades de la actividad *Gestión del registro de prueba.*



Fuente: Autores

La actividad *Gestionar el reporte de anomalías* consta de las siguientes subactividades:

Tabla 15. Subactividades de la actividad *Gestionar el reporte de anomalías.*

Actividad.	Descripción.
A.3.2.1 Gestionar la anomalía	Registrar toda la información pertinente, para la adecuada documentación de las anomalías detectadas durante la ejecución de los casos de prueba.
A.3.2.2 Gestionar la respuesta al reporte de anomalía	Especificar la forma en que solucionó la anomalía, incluyendo al responsable, verificación de funcionalidad, observaciones, medios de entrega de la solución, etc.

Fuente: Autores

La subactividad *Gestionar la anomalía* consta de los siguientes pasos:

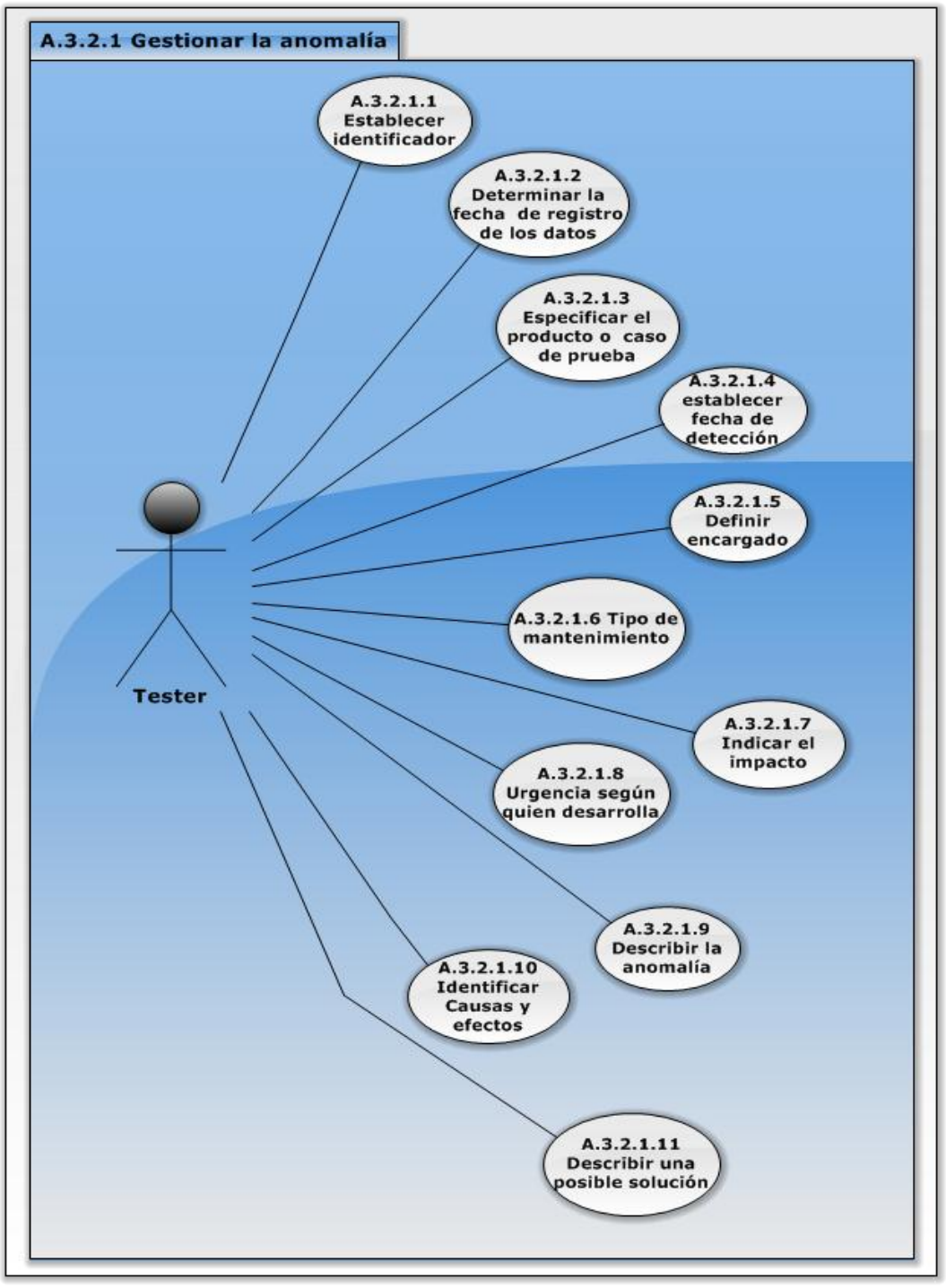
Tabla 16. Pasos de la subactividad *Gestionar la anomalía.*

Actividad.	Descripción.
A.3.2.1.1 Establecer identificador	Describir el identificador único necesario para que cada test case se pueda distinguir de los demás.
A.3.2.1.2 Determinar la fecha de registro de los datos.	Especificar la fecha en la cual se comenzó a registrar la información de la anomalía.
A.3.2.1.3 Especificar el caso de prueba.	Especificar el caso de prueba del cual proviene la anomalía encontrada.
A.3.2.1.4 Establecer fecha de detección	Indicar la fecha en la cual fue detectada la anomalía.
A.3.2.1.5 Definir Encargado	Indicar la persona que realizo la solicitud.
A.3.2.1.6 Tipo de mantenimiento	Especificar si el mantenimiento es de corrección, optimización o eliminación.
A.3.2.1.7 Indicar el impacto	Indicar el impacto de la anomalía, el cual puede ser: ninguno, bajo, medio, alto y muy alto.
A.3.2.1.8 Urgencia según quien desarrolla.	Indicar la urgencia de solución, que puede ser: alta, media y baja.
A.3.2.1.9 Describir la anomalía	Especificar de forma clara y concisa en que consiste la anomalía.
A.3.2.1.10 Identificar Causas y efectos.	Indicar de forma precisa que está causando la anomalía y como se ve afectado el funcionamiento del producto, además de determinar los efectos que generan dentro del sistema.
A.3.2.1.11 Describir una posible solución	Sugerir una o varias soluciones para este inconveniente, las cueles serán evaluadas y algunas se llevaran a cabo.

Fuente: Autores

A continuación se presenta gráficamente pasos y roles presentes en la subactividad *Gestión de la anomalía.*

Figura 12. Pasos de la subactividad *Gestión de la anomalía.*



Fuente: Autores

La subactividad *Gestionar la respuesta al reporte de anomalía* consta de los siguientes pasos:

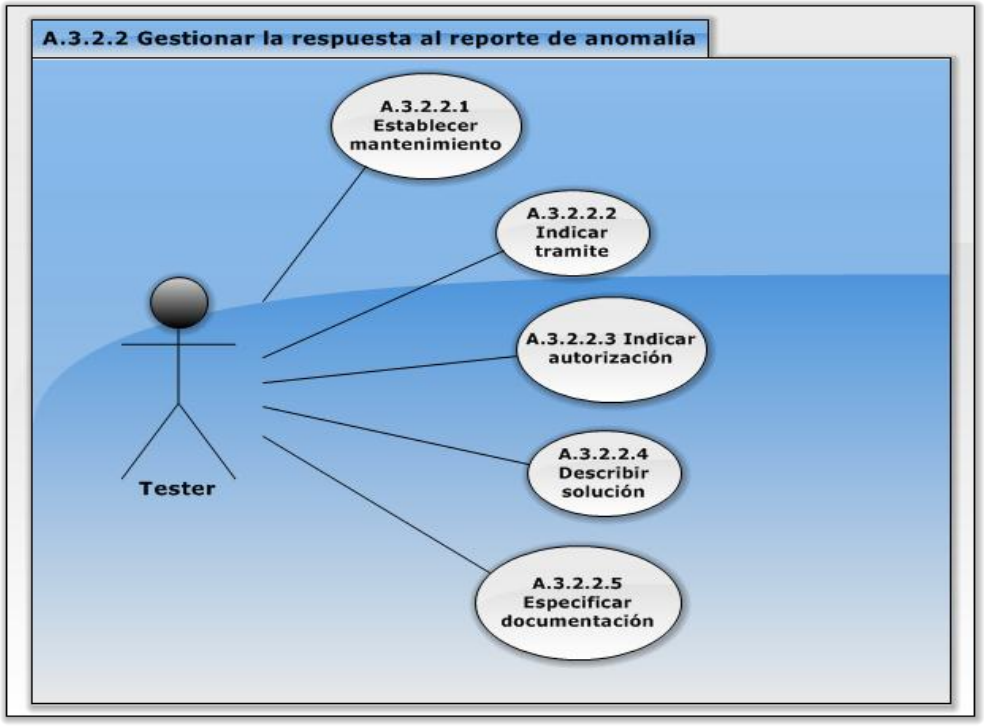
Tabla 17. Pasos de la subactividad *Gestionar la respuesta al reporte de anomalía.*

Actividad.	Descripción.
A.3.2.2.1 Establecer mantenimiento	Establecer el encargado del mantenimiento, verificación de falla o nueva solicitud (si o no), prioridad de desarrollo (alta, media, bajo) y por último se registran algunas observaciones.
A.3.2.2.2 Indicar tramite	Diligenciar el estado del mantenimiento. Si fue aprobado, aplazado o rechazado, con su respectiva fecha.
A.3.2.2.3 Indicar autorización	Indicar que el mantenimiento de esta anomalía ha sido autorizado o no y por quien.
A.3.2.2.4 Describir solución	Gestionar la entrega de la solución de la anomalía, en la cual debe especificar la fecha, el medio de entrega y el responsable de la recepción.
A.3.2.2.5 Especificar documentación	Especificar toda la documentación referente a la solución de la anomalía.

Fuente: Autores

A continuación se presenta gráficamente pasos y roles presentes en la subactividad *Gestión de la respuesta al reporte de anomalía.*

Figura 13. Pasos de la subactividad *Gestión de la respuesta al reporte de anomalía*.



Fuente: Autores

El bloque *Generar los reportes de las pruebas* consta de las siguientes actividades:

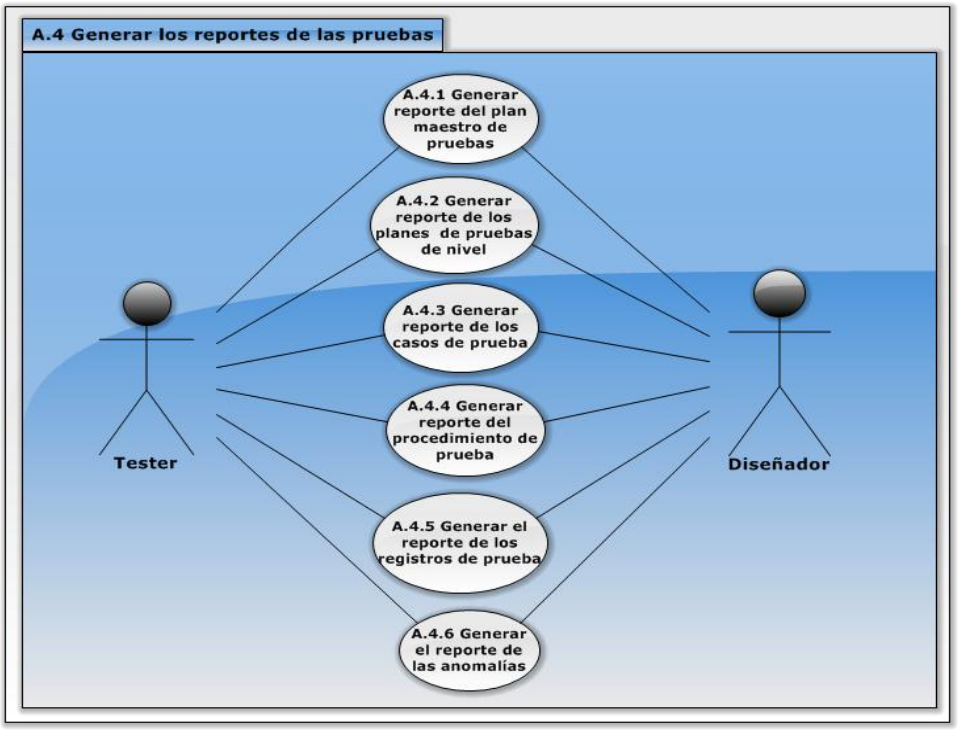
Tabla 18. Actividades del bloque *Generar los reportes de las pruebas.*

Actividad.	Descripción.
A.4.1 Generar reporte del plan maestro de pruebas	Crear un documento con la información presente en el plan maestro de pruebas
A.4.2 Generar reporte de los planes de pruebas de nivel	Crear un documento con la información presente en el plan de pruebas de nivel. Un documento por cada Plan de nivel.
A.4.3 Generar reporte de los casos de prueba	Crear un documento para cada caso de prueba diseñado durante las pruebas.
A.4.4 Generar reporte del procedimiento de prueba	Crear un documento con la información del o los procedimientos que se plantearon para la realización de los casos de prueba.
A.4.5 Generar el reporte de los registros de prueba	Crear un documento con la información de los registros llevados a cabo durante las pruebas.
A.4.6 Generar el reporte de las anomalías	Crear un documento con la información de cada anomalía registrada durante la ejecución de los casos de prueba.

Fuente: Autores

A continuación se presenta gráficamente actividades y roles presentes en el bloque *Gestión de los reportes de las pruebas.*

Figura 14. Actividades del bloque *Gestión de los reportes de las pruebas.*



Fuente: Autores

5.2 HERRAMIENTA COMPUTACIONAL PARA LA DOCUMENTACIÓN DE PRUEBAS DE SOFTWARE.

5.2.1 ¿Qué es QUIS 3.0?

QUIS es una herramienta computacional, cuyo propósito es apoyar el aseguramiento de la calidad de desarrollos software derivados de actividades de investigación. Con este propósito en mente, se desarrolló la versión 3.0 que incorpora la funcionalidad de documentar pruebas de software, además de reparar una amplia variedad de defectos que poseía la versión anterior.

Cabe destacar que QUIS 3.0 fue desarrollada en Visual Basic .Net y contiene una pequeña base de datos en Access para facilitar el manejo de la información de los modelos de calidad.

5.2.2 Estructura funcional de QUIS 3.0

QUIS 3.0 está compuesto por formularios que permiten administrar la información generada a partir de la gestión, evaluación y documentación de calidad software desde cuatro enfoques planteados. Estos formularios se pueden agrupar en:

- Formularios para la gestión y evaluación del proyecto.
- Formularios para la gestión y evaluación de procesos.
- Formularios para la evaluación de productos.
- Formularios para la documentación de las pruebas software.

La lista completa de los formularios están en los proyectos de Pimentel[36] y Pinto[37], en este proyecto solo se describirán la lista de formularios implementados para la documentación de las pruebas software, tal como se observa en la tabla 19:

Tabla 19. Formularios para la documentación de las pruebas software.

Documentación de las pruebas software	
Formularios. *Funcionalidad*	**Información Administrada**
Plan maestro de prueba: Generalidades. *Se diligencia con las generalidades del proyecto software a documentar.*	• Fecha • Nombre del proyecto • Estado del proyecto • Versión actual • Alcance de las pruebas • Descripción del software • Documentos referenciados en el proyecto
Plan maestro de prueba: Recursos. *Se gestionan los recursos del proyecto, personal y herramientas necesarias para el desarrollo del proyecto.*	• Personal (Tabla): Nombre, Responsabilidad • Herramientas computacionales (Tabla): Nombre
Plan maestro de prueba: Herramientas. *Documenta*	• Herramientas de prueba (Tabla): Nombre, Plataforma, Forma de Adquisición, Descripción

[36] Pimentel, Jorge Iván. Implementación de una herramienta computacional para la gestión y evaluación de proyectos y procesos de desarrollo de software de los grupos de investigación de la Universidad Industrial de Santander. Op. Cit. p. 103 – 136.
[37] Pinto, Nelson. Implementación de una herramienta computacional para la evaluación de la calidad de productos software de los grupos de investigación de la Universidad Industrial de Santander. Op. cit. p. 89 – 107.

las herramientas encargadas de realizar las pruebas	
Plan maestro de prueba: Técnicas y métodos. *Si se utilizaron técnicas y/o métodos de pruebas se registran en esta parte.*	• Técnicas y métodos de prueba (Tabla): Nombre, Descripción
Plan maestro de prueba: Característica a probar.	• Característica a probar (Tabla): Producto, Característica, Subcaracterística
Plan maestro de prueba: Cronograma. *Cronograma de las pruebas*	• Fecha de Inicio • Fecha de Finalización • (Tabla): Actividad, Responsable, Fecha de inicio, Fecha de finalización
Plan maestro de prueba: Riesgos. *Riesgos asociados a las pruebas y/o al proyecto.*	• Riesgos (Tabla): Nombre, Causas, Plan de contingencia, Nivel de riesgo
Editar plan de pruebas de nivel *Se divide el proyecto, en cada uno de los productos (partes del programa) que se van a probar.*	• Nombre del producto a probar • Fecha • Evaluación (Tabla): Características, Subcaracterísticas, Métricas, Formula, Parámetros • Personal (Tabla): Nombre
Nuevo Caso de Prueba: Generalidades del caso. *Se documentan los nuevos casos de prueba que se planeen hacer en el proyecto.*	• Id del caso • Producto a probar • Subcaracterística a probar • Nombre del caso • Objetivo • Descripción entorno de prueba • Entradas y salidas (Tabla): Entradas, Salidas • Casos de prueba anteriores (Tabla): Selección, Id caso, Objetivo caso de prueba
Nuevo Caso de Prueba: Procedimiento caso. *Se escoge si el procedimiento del caso de prueba será el mismo general o será más especifico*	• Descripción del procedimiento (Selección entre Trabajar con el procedimiento general de pruebas o crear procedimiento específico para el caso) • (Si se escoge crear procedimiento específico para el caso) Tabla : Actividad, Descripción actividad
Selección caso de prueba. *Se selecciona el caso de prueba que se quiere modificar*	• Id del caso de prueba
Editar caso de prueba: Generalidades del caso. *Se modifican las generalidades de algún caso de prueba existente.*	• Id del caso • Producto a probar • Subcaracterística a probar • Nombre del caso • Objetivo • Descripción entorno de prueba

	• Entradas y salidas (Tabla): Entradas, Salidas • Casos de prueba anteriores (Tabla): Selección, Id caso, Objetivo caso de prueba
Editar caso de prueba: Procedimiento caso. *Se modifican el procedimiento de algún caso de prueba existente.*	• Descripción del procedimiento (Selección entre Trabajar con el procedimiento general de pruebas o crear procedimiento específico para el caso) • (Si se escoge crear procedimiento específico para el caso) Tabla : Actividad, Descripción actividad
Procedimiento general de pruebas. *Se especifica el procedimiento general que se utilizara en las pruebas*	• Descripción del procedimiento (Tabla): Actividad, Descripción actividad
Nuevo reporte de anomalías. *Se diligencian las anomalías encontradas durante la realización de las pruebas y se documenta el caso de prueba donde se encontró la anomalía.*	• Id de anomalía • Fecha de reporte • Encargado reportar anomalía • Nombre del caso de prueba asociado • Fecha detección • Tipo de mantenimiento • Impacto • Prioridad de desarrollo • Descripción • Causas • Efectos • Posible solución
Selección reporte de anomalías. *Se selecciona el reporte de anomalía que se quiere modificar*	• Id reporte de anomalía
Editar reporte de anomalías. *Se modifican algún reporte de Anomalía existente.*	• Id de anomalía • Fecha de reporte • Encargado reportar anomalía • Nombre del caso de prueba asociado • Fecha detección • Tipo de mantenimiento • Impacto • Prioridad de desarrollo • Descripción • Causas • Efectos • Posible solución
Responder reporte de anomalías: Generalidades de la anomalía. *Se identifica y se muestran las generalidades del reporte a responder.*	• Id de anomalía • Fecha de reporte • Encargado reportar anomalía • Nombre del caso de prueba asociado • Fecha detección • Tipo de mantenimiento • Impacto • Prioridad de desarrollo

	• Descripción • Causas • Efectos • Posible solución
Responder reporte de anomalías: Mantenimiento de software. *Se especifican variables involucradas en el proceso de corrección de la anomalía.*	• Encargado Mantenimiento • Verificación de falla o nueva funcionalidad • Observaciones • Tramites (Tabla): Fecha y Estado • Autorización (Tabla): Nombre
Responder reporte de anomalías: Solución de la anomalía. Se deja la constancia del encargado de la corrección y de que entrega.	• Fecha de entrega • Medio de entrega • Responsable • Documentación (Tabla): Descripción, Nombre documento
Registro de las pruebas: Eventos. *Registro de los eventos significativos ocurridos durante las pruebas*	• Eventos (Tabla): Fecha, Descripción
Registro de las pruebas: Medición. *Se realiza una evaluación de los productos a probar.*	• Producto • Evaluación (Tabla): Característica, Subcaracterística, Métrica, Parámetro, Fecha de Evaluación, Valor del parámetro, Valor de la métrica
Ver evaluación. *Se muestran los resultados de las evaluaciones hechas a cada uno de los productos.*	• Evaluación • Fecha de evaluación • Evaluación (Tabla): Producto, Características, Subcaracterísticas, Métricas, Valor permitido, Valor Medido

Fuente: Autores

5.2.3 Caracterización del desarrollo de QUIS 3.0

QUIS 3.0 surge de la necesidad de gestionar por completo los aspectos necesarios de evaluación de la calidad en la herramienta QUIS, también es el resultado de una exhaustiva revisión del estado de la aplicación para así lograr corregir la mayor cantidad de defectos posibles. Este proyecto contó con dos estudiantes de pregrado que participaron como desarrolladores de la aplicación, que gracias a la orientación de los directores lograron crear un modelo de documentación de pruebas software basado en la norma IEE 829; para después implementar el modelo trabajando bajo el paradigma orientado a

objetos, y desarrollando la aplicación siguiendo la metodología ágil de Extreme Programing (XP).

Este proyecto surge dentro del desarrollo de una investigación desarrollada por el grupo de investigación en Sistemas y Tecnologías de la Información de la UIS. Para este proyecto se contó con la ayuda económica suministrada por la Universidad Industrial de Santander gracias al financiamiento del proyecto de investigación "MODELO PARA LA DOCUMENTACIÓN DE PRUEBAS DE SOFTWARE INCORPORADO EN LA HERRAMIENTA DE EVALUACIÓN DE CALIDAD DE SOFTWARE DERIVADO DE ACTIVIDADES DE INVESTIGACIÓN". Este auxilio permitió la compra de equipos y software que facilitaron el desarrollo, además de ofrecer un auxilio económico a los desarrolladores del proyecto.

6. ILUSTRACIÓN DEL USO DEL SISTEMA PROPUESTO

En este último capítulo se ilustrara el uso del sistema propuesto e implementado en la herramienta QUIS, para la valoración de aspectos de calidad y documentación de pruebas software en varios contextos de investigación diferentes.

6.1 GESTIÓN DE LA CALIDAD DEL PROYECTO HSLABCC

La primera de las herramientas que se utilizaron para ilustrar el uso del sistema planteado es una herramienta del Grupo de Investigación en Sistemas y Tecnologías de la Información STI de la UIS, denominada HSLABCC – Sistema de Gestión de información para laboratorios orientado a la Cloud Computing. Esta herramienta es una actualización orientada a la Cloud de la herramienta que anteriormente se manejaba. Los autores de esta herramienta decidieron únicamente evaluar y probar el código fuente. A continuación se presenta una tabla con las generalidades del proyecto

Tabla 20. Ficha Técnica de Evaluación de Calidad y Documentación de Pruebas del software HSLABCC

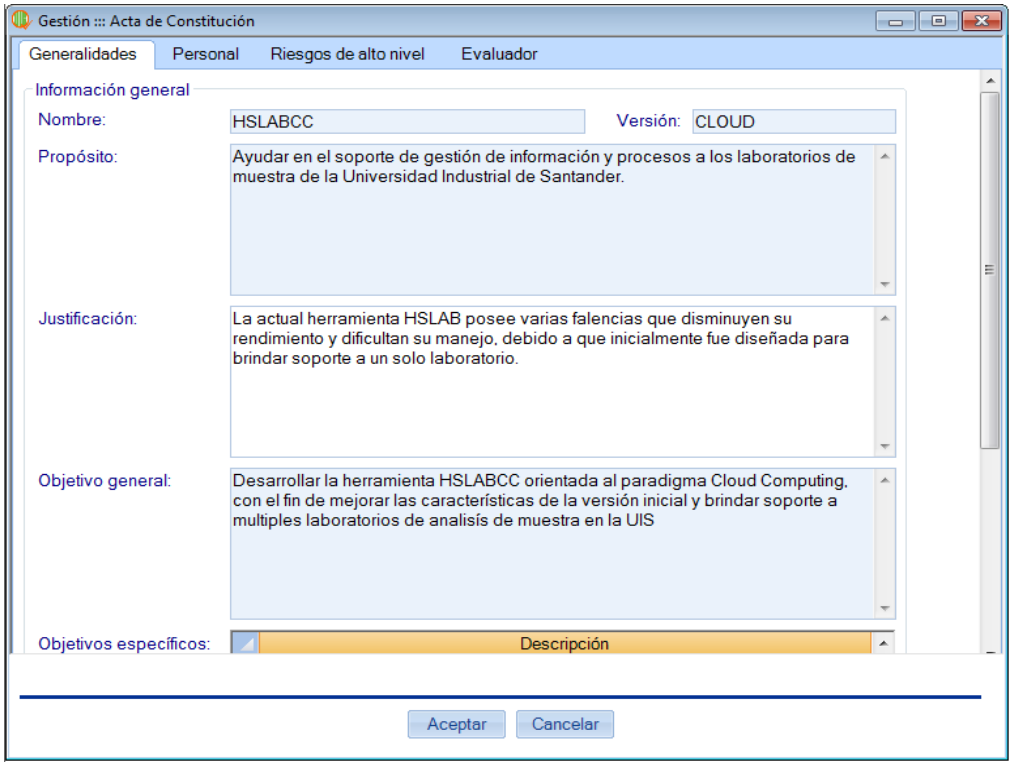
Ficha Técnica de Evaluación de Calidad y Documentación de Pruebas	
Nombre del software	HSLABCC – Sistema de Gestión de Información para laboratorios orientado a la Cloud Computing
Breve descripción de su funcionalidad	Herramienta computacional que ayuda en el soporte de gestión de información y procesos a los laboratorios de muestra de la Universidad Industrial de Santander.
Grupo propietario	Grupo de Investigación en Sistemas y Tecnologías de la Información UIS
Aspectos de calidad evaluados y/o gestionados	Gestión de la calidad de producto y documentación de pruebas software
Actividades llevadas a cabo	Las relacionadas con las actividades de A.1 Planear la pruebas, A.3 Reporte de las pruebas, A.4 Generar los reportes de las pruebas, además de la gestión en el módulo de producto.
Participantes	• Wilson Andrés Cardona Rada estudiante y desarrollador de la herramienta. • Karenth Andrea Martínez Saboya estudiante y desarrollador de la herramienta. • Edgar Fernando Vega Morales estudiante y auxiliar

	gestor de calidad
Resultados y/o comentarios	Esta evaluación se realizó en principio con la versión 2.0 de la herramienta QUIS, una vez completada la herramienta 3.0 se procedió a completar la evaluación.

Fuente: Autores

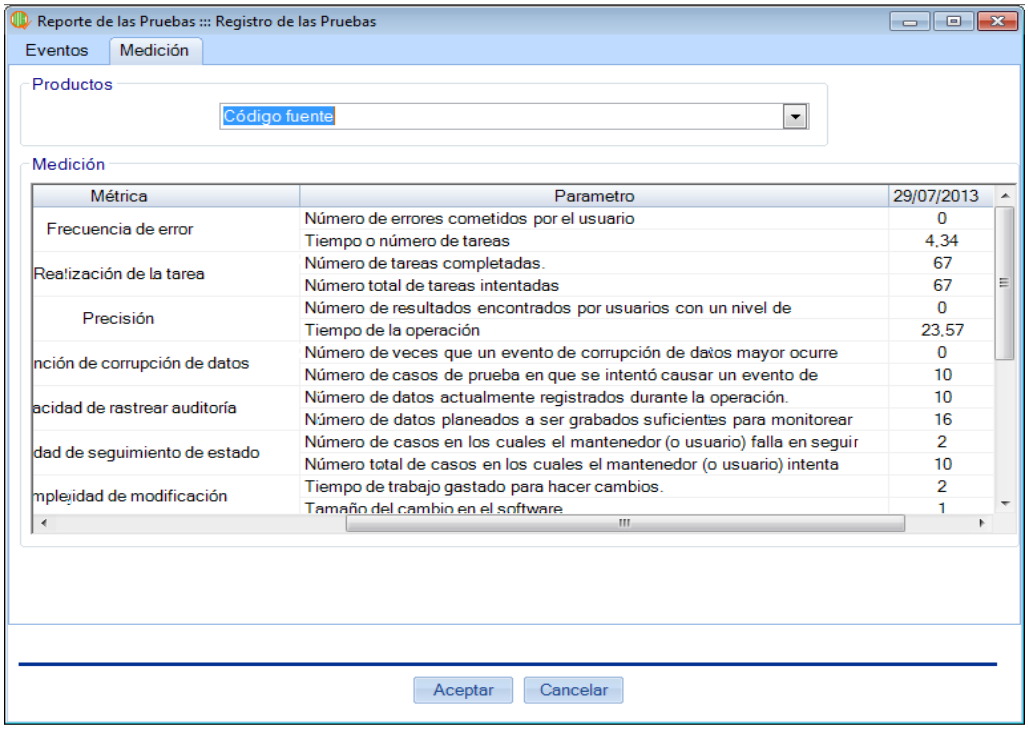
A continuación se muestras algunos formularios utilizados en la evaluación y documentación de la herramienta HSLABCC

Figura 15. Generalidades del acta de constitución del proyecto HSLABCC



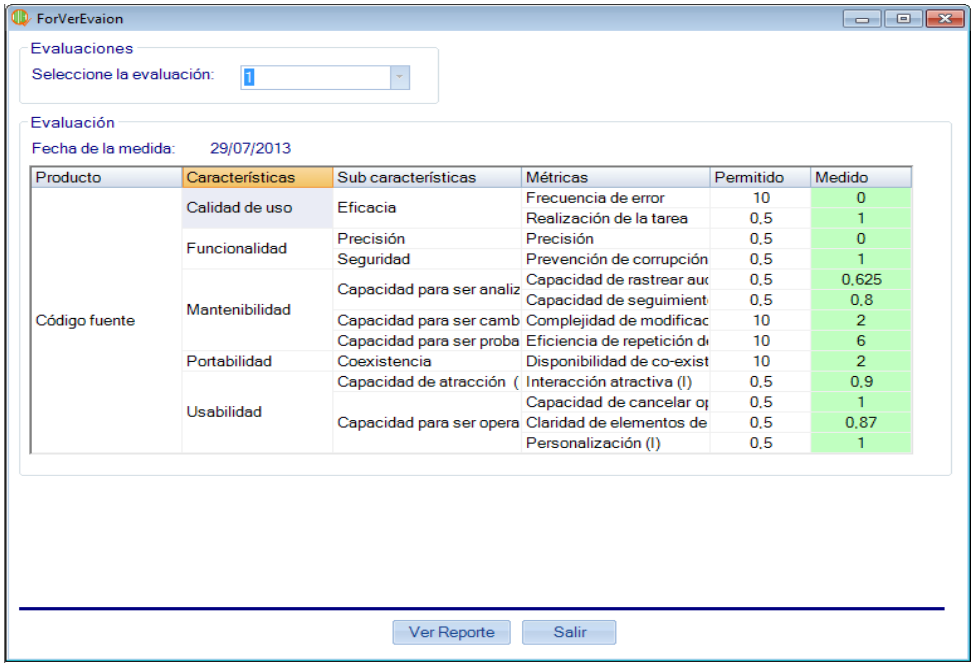
Fuente: Autores

Figura 16. Características a evaluar en la herramienta HSLABCC



Fuente: Autores

Figura 17. Reporte de evaluación de características de la herramienta HSLABCC



Fuente: Autores

6.2 GESTIÓN DE LA CALIDAD DEL PROYECTO LEUKOSORTER

La segunda de las herramientas que se utilizaron para ilustrar el uso del sistema planteado es una herramienta del Grupo de Investigación en Ingeniería Biomédica GIIB de la UIS, denominada LEUKORTER – Sistema Clasificador de Glóbulos Blancos. Los autores de esa herramienta decidieron evaluar y probar el ejecutable de la aplicación. A continuación se presenta una tabla con las generalidades del proyecto

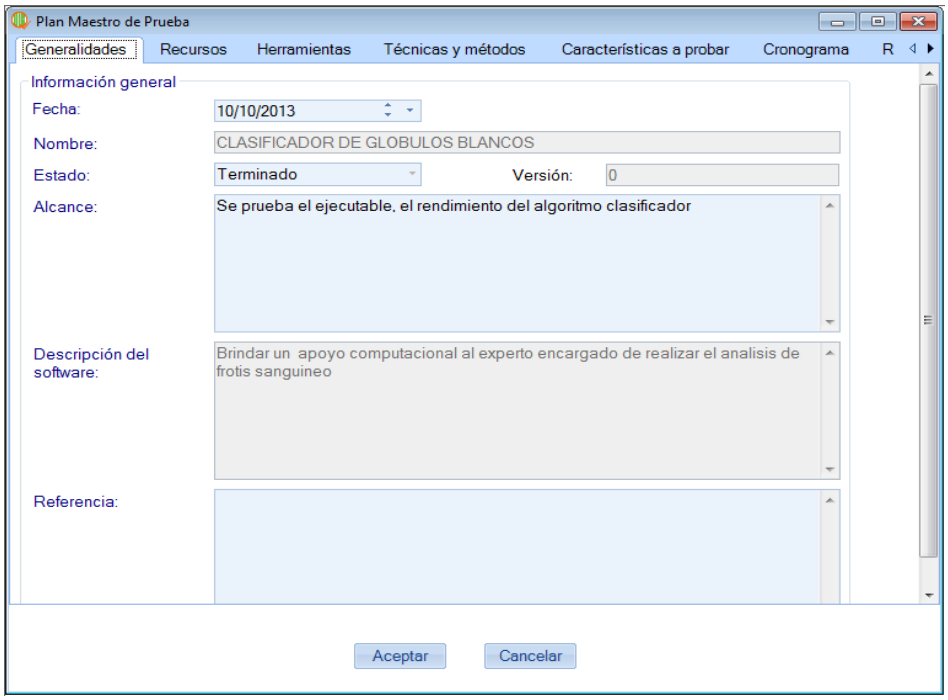
Tabla 21. Ficha Técnica de Evaluación de Calidad y Documentación de Pruebas del software LEUKOSORTER

Ficha Técnica de Evaluación de Calidad y Documentación de Pruebas	
Nombre del software	LEUKOSORTER – Sistema Clasificador de Glóbulos Blancos
Breve descripción de su funcionalidad	Es una herramienta que pretende brindar un apoyo computacional al experto encargado de realizar el análisis de frotis sanguíneo
Grupo propietario	Grupo de Investigación en Ingeniería Biomédica GIIB de la UIS
Aspectos de calidad evaluados y/o gestionados	Gestión de la calidad de producto y documentación de pruebas software
Actividades llevadas a cabo	Las relacionadas con las actividades de A.1 Planear la pruebas, A.3 Reporte de las pruebas, A.4 Generar los reportes de las pruebas, además de la gestión en el módulo de producto.
Participantes	• Juan Carlos Pico Sarmiento estudiante y desarrollador de la herramienta. • Johnatan Naranjo Acevedo estudiante y desarrollador de la herramienta. • Edgar Fernando Vega Morales estudiante y auxiliar gestor de calidad
Resultados y/o comentarios	Fueron evaluados y documentados pocos aspectos, ya que los autores de la herramienta LEUKOSORTER solo querían evaluar aspectos muy específicos.

Fuente: Autores

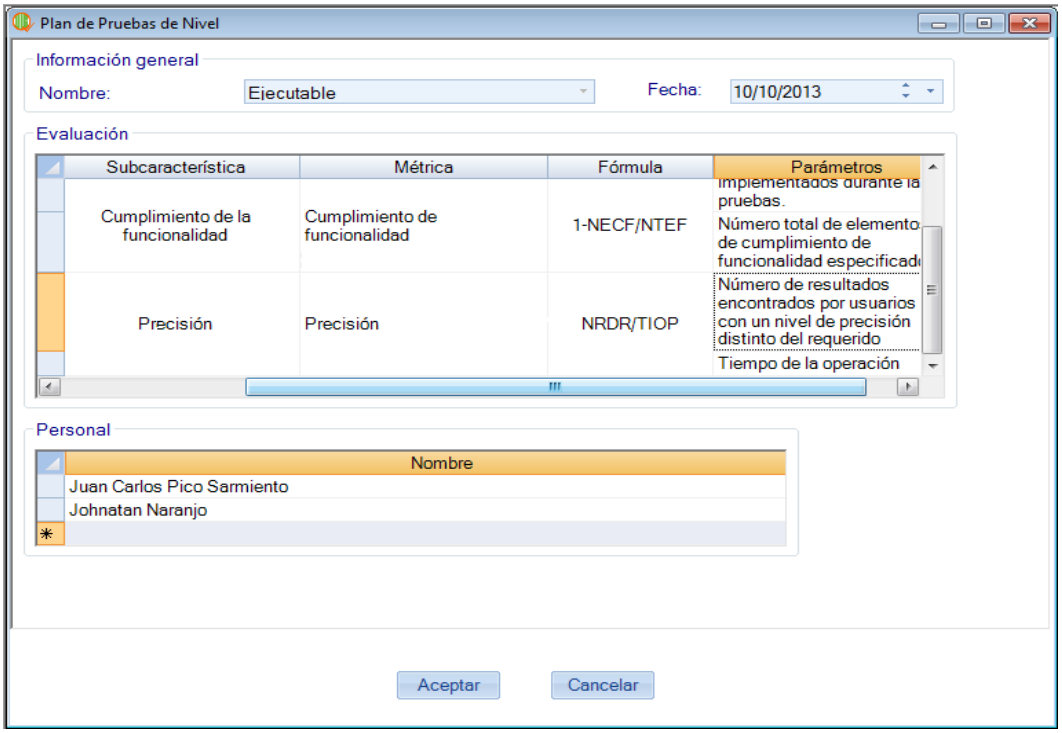
En seguida se muestra algunos formularios diligenciados en la evaluación y documentación de la herramienta LEUKOSORTER.

Figura 18. Generalidades Plan maestro de pruebas de la herramienta LEUKOSORTER



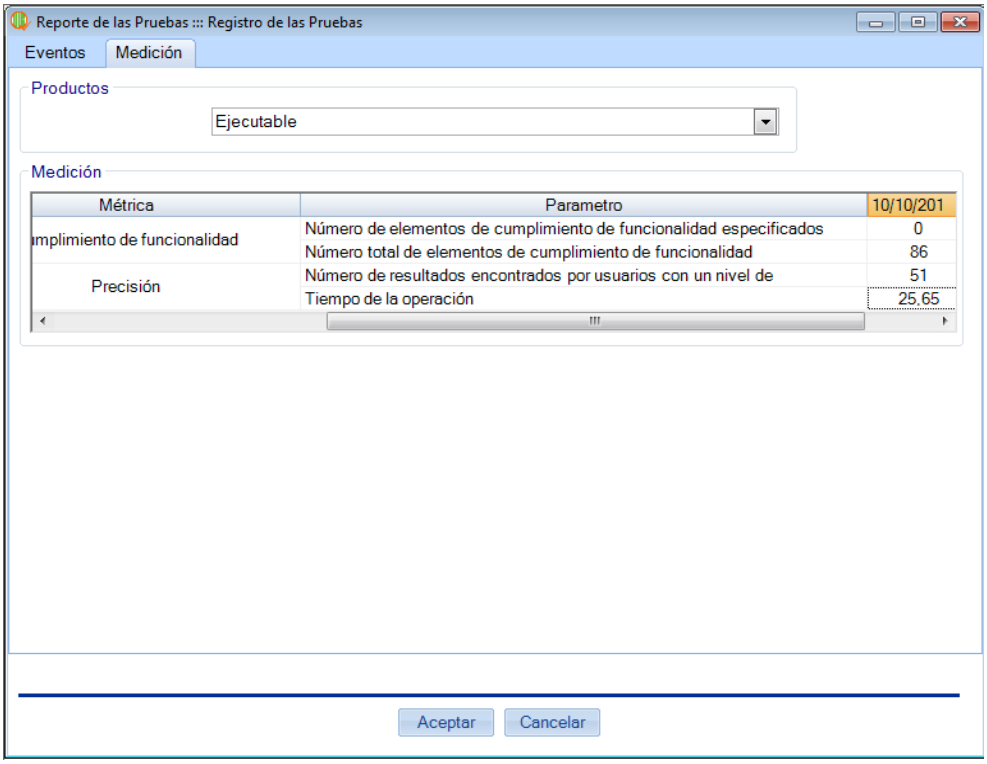
Fuente: Autores

Figura 19. Características a evaluar en la herramienta LEUKOSORTER



Fuente: Autores

Figura 20. Medición de las características evaluadas en la herramienta LEUKOSORTER



Fuente: Autores

6.3 GESTIÓN DE LA CALIDAD DEL PROYECTO QUIS VERSIÓN 3.0

La tercera de las herramientas que se utilizaron para ilustrar el uso del sistema planteado es la misma herramienta para la gestión y evaluación de la calidad software derivado de actividades de investigación QUIS en su versión 3.0 del grupo de investigación en Sistemas y Tecnologías de la Información STI de la UIS. Los autores de esta herramienta decidieron utilizar todos los módulos del software QUIS en la evaluación para así tratar de utilizar la mayor cantidad de formularios posibles.

Tabla 22. Ficha Técnica de Evaluación de Calidad y Documentación de Pruebas del software QUIS 3.0

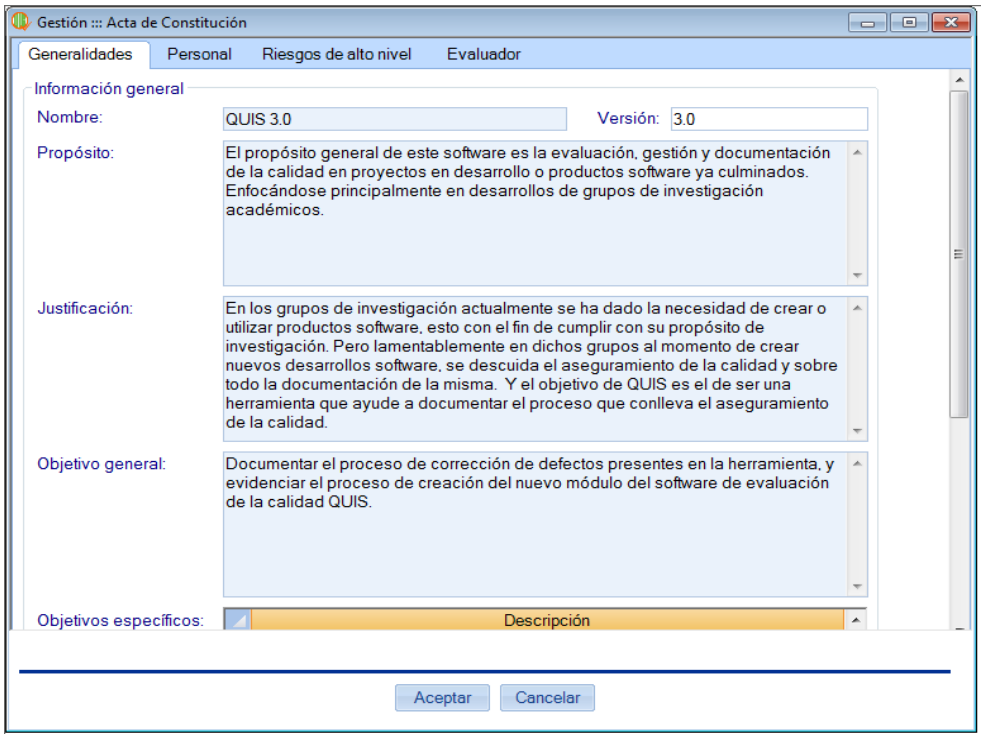
Ficha Técnica de Evaluación de Calidad y Documentación de Pruebas	
Nombre del software	QUIS – Sistema de Gestión y evaluación de la calidad de software derivado de actividades de investigación
Breve descripción de su funcionalidad	El propósito general de este software es la evaluación, gestión y documentación de la calidad

	en proyectos en desarrollo o productos software ya culminados. Enfocándose principalmente en desarrollos de grupos de investigación académicos.
Grupo propietario	Grupo de Investigación en Sistemas y Tecnologías de la Información UIS
Aspectos de calidad evaluados y/o gestionados	La totalidad de los aspectos de calidad contenidos en la aplicación QUIS
Actividades llevadas a cabo	Las relacionadas con las actividades de A.1 Planear la pruebas, A.2 Desarrollo de las pruebas, A.3 Reporte de las pruebas, A.4 Generar los reportes de las pruebas, además de la gestión en los módulos de proyecto, proceso y producto.
Participantes	• Ing. Nelson Enrique León Martínez • Juan Luis Aguilar García estudiante y desarrollador de la herramienta. • Edgar Fernando Vega Morales estudiante y auxiliar gestor de calidad
Resultados y/o comentarios	

Fuente: Autores

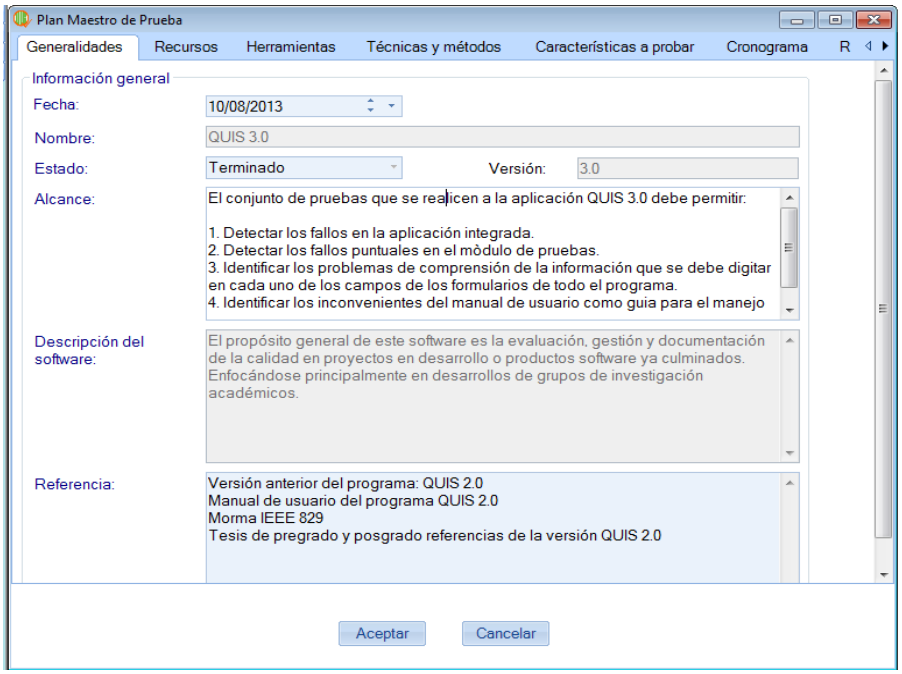
A continuación se muestras algunos formularios utilizados en la evaluación y documentación de la herramienta QUIS.

Figura 21. Generalidades del acta de constitución del proyecto QUIS 3.0



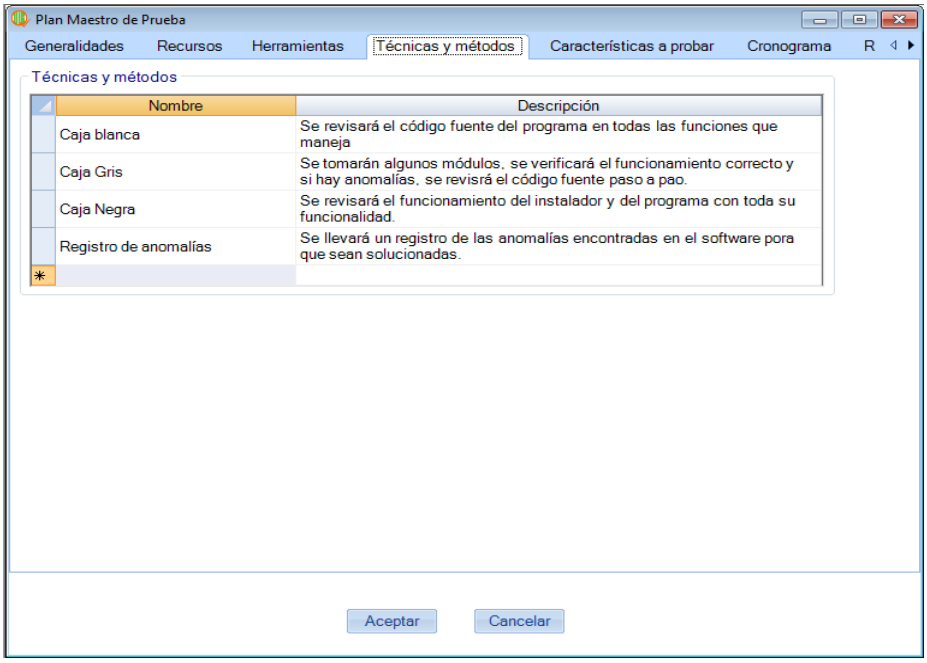
Fuente: Autores

Figura 22. Generalidades plan maestro de prueba de la herramienta QUIS 3.0



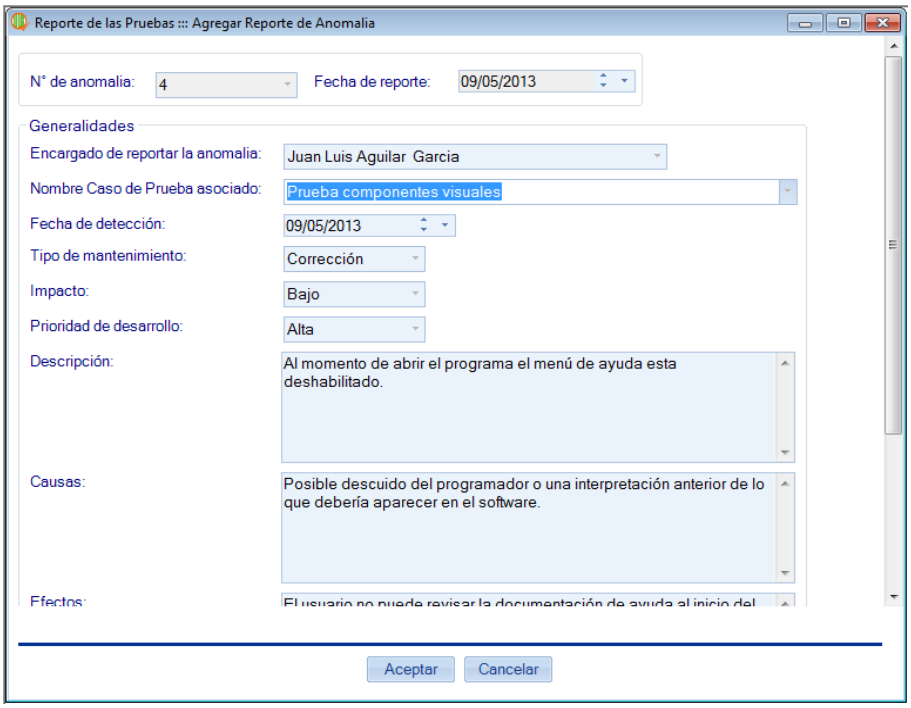
Fuente: Autores

Figura 23. Técnicas y métodos de prueba diligenciados plan maestro de pruebas de la herramienta QUIS 3.0



Fuente: Autores

Figura 24. Ejemplo reporte de anomalías encontradas en la herramienta QUIS 3.0



Fuente: Autores

CONCLUSIONES Y RECOMENDACIONES

El proceso para desarrollar el modelo de documentación de pruebas software, que se implementó en la herramienta para la evaluación de la calidad QUIS, se realizó mediante una serie de revisiones, la implementación del modelo, y una ilustración del uso de la herramienta con el propósito de aportar diferentes contextos sobre los cuales se podría desempeñar.

La primera de las revisiones que se realizaron fue sobre las normas más relevantes para la documentación de pruebas software, esto con el fin de realizar el modelo que posteriormente se implementaría en la herramienta. Se llegó a la conclusión de que dicho modelo debía estar basado en la norma IEEE 829, ya que esta norma era la que mejor se adaptaba al desarrollo que se quería realizar, además de mostrarse flexible a la hora de su implementación. Posteriormente se hizo un reconocimiento de las herramientas computacionales que tuvieran una función similar a la documentación de pruebas software, con lo que se determinó que existen en el mercado un variado conjunto de herramientas que le permiten al usuario realizar pruebas software, aunque la documentación que estás realizan es muy limitada, restringiéndose solamente a documentar los casos de pruebas realizados. La última de las revisiones que se realizaron, fue una comprobación del estado de la herramienta con el fin de identificar la mayor cantidad de defectos posibles, documentándolos y posteriormente corrigiéndolos. Esto sirvió además para que el personal del proyecto de familiarizara con el software y con la metodología de programación.

Con una versión más estable del software se procedió a la implementación del modelo en el lenguaje de programación Visual Basic .NET, siguiendo el paradigma orientado a objetos y utilizando la metodología ágil de Extreme Programing (XP). Finalmente, con el objetivo de probar y dejar evidencia del uso de la aplicación, se procedió a ilustrar el uso de la herramienta sobre software desarrollados en diferentes grupos de investigación, evidenciándose problemas comunes como: cambio de personal, desconocimiento del lenguaje

de programación, incorrecta gestión del tiempo del proyecto, desaprovechamiento de los recursos del proyecto, entre todos.

Es de importancia destacar, ya que puede llegar a generar dudas en algún usuario, que este proyecto tiene como objetivo implementar un modelo para la documentación de pruebas software. Se pueden gestionar los recursos disponibles para la realización de las pruebas, planear las pruebas, documentar las técnicas y métodos de realización de pruebas, registrar las anomalías, documentar los eventos significativos y los resultados de las pruebas, pero no es el objetivo de esta herramienta la realización de las pruebas software, ya que como se pudo demostrar anteriormente, existe un gran número de herramientas computacionales que sirven como soporte para realizar las pruebas.

Es posible afirmar que se implementó una herramienta con la que cualquier grupo de investigación que desarrolle software puede realizar una completa gestión y evaluación de la calidad de sus aplicaciones, y que por ahora no es necesario el desarrollo de módulos adicionales, ya que si se agregan más funcionalidades, se corre el riesgo de perder uno de los principios de la herramienta, la cual debe ser una ayuda y no una carga adicional a la persona que quiere evaluar la calidad de su proyecto. Lo que sí se considera necesario y obligatorio es hacer una revisión continúa de la herramienta para optimizar el código fuente y su funcionalidad.

Por último los autores invitan a los grupos de investigación de la Universidad Industrial de Santander, a masificar el uso de herramientas como la que se desarrolló, ya que el desarrollo software en estos grupos está expuesto a una serie de problemas que pueden derivar en el fracaso del proyecto y con el soporte brindado, se puede mitigar este riesgo.

GLOSARIO

Calidad: propiedad o conjunto de propiedades inherentes a algo, que permite juzgar su valor. Características propias del software aquellas que tu quiera controlar y asegurar, el software es un producto inmaterial que no se fabrica, tampoco se degrada físicamente, si no que se desarrolla; el software puede tener errores, incidencias pero no son similares a lo que cualquier equipo de carácter físico.

Documentación: la documentación es la acción de plasmar información en algo tangible para un fin determinado. Lo cual es muy importante en las pruebas software ya que gracias a la documentación se puede dejar evidencia tangible del trabajo realizado.

IEEE 829: este estándar describe los documentos que se pueden producir durante el proceso de prueba de software. Las etapas y los documentos que define la norma son las siguientes: Planeación de la pruebas, Especificación del diseño de prueba, Especificación de los Casos de Prueba, Especificación del Procedimiento, Reporte de avance de los ciclos probados, Registro de la prueba, Reporte de incidentes, Sumario de la prueba.

Pruebas software: son actividades de prueba que se realizan durante el proceso de desarrollo de software, siguiendo unas técnicas y estrategias establecidas, con el fin de minimizar la futura aparición de defectos en el producto final.

Testware: el testware proviene de la unión de las palabras “test” y “software”. El resultado de las actividades pruebas, es lo que se conoce como testware.

Niveles de integridad: es el grado en el que un software cumple con ciertas características del sistemas de software o partes del mismo como: la complejidad, rendimiento, fiabilidad, etc. Este valor simbólico representa el grado de cumplimiento dentro de un esquema de niveles de integridad.

BIBLIOGRAFIA.

- www.acis.org.co. Recuperado el 27 de diciembre de 2012, de http://www.acis.org.co/fileadmin/Base_de_Conocimiento/XXVII_Salon_Informatica/MariaClaraChoucair-PruebasDeSoftware.pdf

- Sommerville Ian. Ingeniería del software. Séptima Edición. Madrid. Pearson Educación 2005.

- Pressman Roger. Ingeniería del software-Un enfoque práctico. Quinta Edición. México. McGraw-Hill 2002.

- Davis, A., 201 Principles of Software Development, McGraw-Hill, 1995.

- Graham Dorothy, Van Veenendaal Erik, Evans Isabel, Black Rex. Foundations of software testing. Edición actualizada por la fundación ISTQB año 2007.

- IEEE 829-2008. IEEE Standard for Software Test Documentation. Software Engineering Technical Committee of the IEEE Computer Society.
  E-ISBN: 978-0-7381-5746-7. Print ISBN: 978-0-7381-5747-4. INSPEC Accession Number: 10219290. Digital Object Identifier: 10.1109/IEEESTD.2008.4578383.

- IEEE 1008- 1987. IEEE Standard for Software Unit Testing. E-ISBN: 0-7381-0400-0. INSPEC Accession Number: 2894459
  Identificador digital: 10.1109/IEEESTD.1986.81001.

- Recuperado el día 25 de noviembre de 2012, de http://www.softwaretestingstandard.org/

- es.scribd.com. Recuperado el 1 de diciembre de 2012, de http://es.scribd.com/doc/76765984/46/DOCUMENTO-DE-ESPECIFICACION-DE-DISENO-DE-PRUEBAS.

- Marcos Esperanza. Investigación en Ingeniería del Software vs. Desarrollo Software. Recuperado el 28 de mayo de 2013, de http://gidis.inf.pucp.edu.pe/recursos/InvIngSWvsDS.pdf

- Sampieri Roberto, Fernández Carlos, Baptista Pilar. Metodología de la investigación. Quinta edición. México. McGraw-Hill 1996.

- Joyales Luis. Programación orientada a objetos. Primera edición. Madrid. McGraw-Hill 1996.

- Pimentel, Jorge Iván. Implementación de una herramienta computacional para la gestión y evaluación de proyectos y procesos de desarrollo de software de los grupos de investigación de la Universidad Industrial de Santander. Bucaramanga: 2011. Presentada en la Universidad Industrial de Santander para obtención del título de Ingeniero de sistemas.

- Pinto, Nelson. Implementación de una herramienta computacional para la evaluación de la calidad de productos software de los grupos de investigación de la Universidad Industrial de Santander. Bucaramanga: 2011. Presentada en la Universidad Industrial de Santander para obtención del título de Ingeniero de sistemas.

- León, Nelson Enrique. Propuesta de un sistema para la evaluación de calidad de software derivado de actividades de investigación. Bucaramanga: 2011. Presentada en la Universidad Industrial de Santander para obtención del título de Magister en Ingeniería área de Informática y Ciencias de la Computación.

- Apache JMETER, recuperado el 25 de marzo de 2013, de http://jmeter.apache.org/

- SmarteScript Functional Test Tool, recuperado el 25 de marzo de 2013, de http://www.smartesoft.com/products_smartescript.php.

- Testmaster, recuperado el 25 de marzo de 2013, de http://testmaster.sourceforge.net/

- Sonar | Marco de Desarrollo de la Junta de Andalucía, recuperado el 25 de marzo de 2013, de http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/372

- Selenium - Herramienta para automatizar pruebas de software ~ Programando con Café, recuperado el 3 de abril de 2013, de http://www.programandoconcafe.com/2011/10/selenium-herramienta-para-automatizar.html

- Watir.com | Web Application Testing in Ruby, recuperado el 3 de abril de 2013, de http://watir.com/

- IBM - Rational Functional Tester - Colombia, recuperado el 3 de abril de 2013, de http://www-03.ibm.com/software/products/co/es/functional/

- TestLink Home, recuperado el 3 de abril de 2013, de http://testlink.sourceforge.net/docs/testLink.php