UNIVERSIDAD METROPOLITANA DE EDUCACIÓN, CIENCIA Y TECNOLOGÍA

Decreto Ejecutivo 575 del 21 de julio de 2004

Acreditada mediante Resolución N°15 del 31 de octubre de 2012

TECNOLOGIA CONSTRUCCIÓN Y MEDIO AMBIENTE

Maestría en sistemas computacionales con énfasis. En redes y comunicaciones ECTF-75-2011

TRABAJO DE GRADO COMO OPCIÓN AL TÍTULO DE MAGISTER

Realizado en Bogotá Colombia

CONSTRUCCIÓN DE UNA METODOLOGÍA PARA LA EVALUACIÓN DE PRODUCTOS DE SOFTWARE DE LA UNIVERSIDAD DE CUNDINAMARCA

Autor: CESAR YESID BARAHONA RODRIGUEZ

Tutor: ALEX ORLANDO NIÑO CENDALES

Panamá, agosto ,2017

UNIVERSIDAD METROPOLITANA DE EDUCACIÓN, CIENCIA Y TECNOLOGÍA

Decreto Ejecutivo 575 del 21 de julio de 2004

Acreditada mediante Resolución N°15 del 31 de octubre de 2012

TECNOLOGIA CONSTRUCCIÓN Y MEDIO AMBIENTE

Maestría en sistemas computacionales con énfasis. En redes y comunicaciones ECTF-75-2011

CONSTRUCCIÓN DE UNA METODOLOGÍA PARA LA EVALUACIÓN DE PRODUCTOS DE SOFTWARE DE LA UNIVERSIDAD DE CUNDINAMARCA

Trabajo presentado como requisito para optar al grado de:

Magister en sistemas computacionales con énfasis. En redes y comunicaciones

Autor: CESAR YESID BARAHONA RODRIGUEZ

Tutor: ALEX ORLANDO NIÑO CENDALES

Panamá, agosto ,2017

Dedicatoria

A mi padre, por enseñarme cómo afrontar la vida y al motor de mi vida

Sophie L.

Agradecimientos

Agradecido con Dios por la vida y hacer posible que sea un profesional y orgullo para mi familia y mi país, gracias también a ellos que con su incondicional apoyo hacen que pueda cumplir mis metas y sueños de los cuales ellos se sienten participes y felices.

A las docentes de la Universidad Metropolitana de Educación y Tecnología UMECIT, el aprendizaje y amistad que se forja cada día, a mi institución la Universidad de Cundinamarca por brindar el apoyo para iniciar y terminar esta maestría y por último mi familia por aceptar este gran sacrificio.

Tabla de contenido

Lista de Gráficas

Lista de Tablas

2. RESUMEN

Según el informe dado por Software Engineering Institute (SEI) Colombia se encuentra en el primer lugar a nivel Latinoamérica en producción de software de nivel cinco destacándose de esta manera en el desarrollo de software de calidad.(MINTIC, 2015).

Por lo tanto, la Universidad de Cundinamarca se centra en que sus desarrolladores se enfoquen en proyectos informáticos orientados a tecnología web los cuales deben ser diseñados y construidos con estándares internacionales de calidad, de este modo se propone esta metodología que abarca dicha evaluación partiendo desde el modelado, hasta las pruebas funcionales. En el planteamiento de esta se tuvo en cuenta la más reciente actualización de evaluación de calidad de software la ISO/IEC/IEEE 29119 la cual hace una recopilación de los criterios de calidad presentes en la ISO 9126 que está orientada a la funcionalidad y la satisfacción del cliente; en la ISO 25000 se realiza un compendio de la 9126 y 14598 dirigida directamente al evaluador. Basado en estas características dicho estándar hace una valoración de la calidad encaminada al testeo.

Como seguimiento a esta actividad se estudian distintas pruebas que se recomiendan realizar al software. También se tiene en cuenta el modelado UMl 2.0, con el fin de construir una metodología íntegra.

Se busca que los evaluadores cuenten con una guía y un grupo de herramientas que faciliten la valoración de los distintos aplicativos.

De esta manera los estudiantes, desarrolladores, arquitectos de software, directores de proyectos se enfoquen en generar criterios enfocados a la calidad de software durante todo el ciclo de vida del producto tecnológico.

3. ABSTRACT

According to the report given by the Software Engineering Institute (SEI), Colombia is in the first place in Latin America in level five software production, emphasizing in this way the development of quality software (MINTIC, 2015).

Therefore, the University of Cundinamarca focuses on its developers focusing on web-technology-oriented computer projects which must be designed and built with international quality standards, thus proposing this methodology that covers such evaluation starting from modeling , To functional tests. This approach took into account the most recent software quality assessment update ISO / IEC / IEEE 29119 which compiles the quality criteria present in ISO 9126 that is oriented towards functionality and satisfaction the client's; In ISO 25000 a compendium of the 9126 and 14598 is made addressed directly to the evaluator. Based on these characteristics, this standard evaluates the quality of the test.

As a follow-up to this activity, several tests that are recommended to the software are studied. UMl 2.0 modeling is also taken into account in order to construct an integral methodology.

It is sought that the evaluators have a guide and a group of tools that facilitate the valuation of the different applications.

In this way students, developers, software architects, project managers focus on generating criteria focused on software quality throughout the technological product life cycleobstante, en cada caso habrá que añadir las páginas iniciales correspondientes y ajustarse a lo especificado en la correspondiente normativa sobre encuadernación, y secciones o capítulos que debe incluir, así como su extensión.

4. INTRODUCCIN

El testeo de software normalmente se considera como el único proceso para hallar defectos en un producto software, pero esta consideración es errónea debido a que el testeo va relacionado con ciertos atributos como escalabilidad, mantenibilidad, usabilidad, compatibilidad y capacidad del producto software.

Por otra parte, el desarrollo de productos software se ha enfrentado a grandes y constantes cambios en los últimos años, por consiguiente existe la necesidad de desarrollar aplicaciones con un alto nivel de calidad, pero existe un gran desconocimiento acerca de este proceso, una forma de realizar el proceso de calidad es a través del uso de estándares de calidad.

En este sentido el presente proyecto propone la metodología que permite evaluar el nivel de calidad de aplicaciones web desarrolladas en la Universidad de Cundinamarca, basada en la norma ISO/IEC/IEEE 29119 1-2-3, esta metodología tiene en cuenta también la evaluación de modelado basándose en el framework UML 2.0.

Otra tarea prioritaria es la posibilidad de automatizar el proceso de pruebas que realiza el equipo de desarrollo, hasta el momento a través del uso de herramientas libres que permiten facilitar el proceso de evaluación de calidad de una aplicación web, aún no existe una herramienta que evalúe el modelado de una aplicación por lo tanto se plantean una serie de criterios para tener en cuenta para realizar la evaluación de modelado de la aplicación.

Se plantea la presente metodología con el fin de complementar la evaluación de modelado de aplicaciones web y la evaluación de pruebas las cuales teniendo en cuenta la norma ISO/IEC/IEEE 29119 parte 2 se dividen en: gestión de pruebas, pruebas funcionales y pruebas de carga y rendimiento.

A través de la realización de encuestas fue posible establecer los valores para cada una de las partes que pertenecen al proceso de evaluación de calidad de un producto software.

Sobre la base de las ideas expuestas la metodología para la evaluación y análisis de productos software de la Universidad de Cundinamarca apoya el proceso de creación y evaluación de un producto software para que pueda ser competitivo desde su diseño hasta su desarrollo y ejecución

1. CONTEXTUALIZACIÓN DEL PROBLEMA

1.1. Planteamiento del problema

En la Universidad de Cundinamarca en la Facultad de Ingeniería en el programa de sistemas extensión Facatativá, el perfil profesional está orientado "en analizar, diseñar y desarrollar aplicaciones informáticas en actividades académicas, comerciales, industriales y agro-Industriales, de auditoria para controlar el procesamiento de información para garantizar la veracidad de la información, de tal manera que esta sea, veraz, completa y oportuna.("OCUPACIONAL," n.d.) Para lograr estas competencias se forma y especializa a la comunidad estudiantil en técnicas y procesos tecnológicos orientados a la construcción de software.

Por otro lado, se ha evidenciado que el desarrollo de estas aplicaciones en la actualidad no se está aplicando un método para evaluar estos desarrollos informáticos, se ha definido metodologías de investigación y de desarrollo informático, pero el proceso de evaluar la aplicación informática no se tiene concebida.

Por lo tanto, este proyecto se enfocará en la construcción de una metodología con la cual se evalúe la calidad de las aplicaciones web creadas en el programa de ingeniería de sistemas en la Universidad de Cundinamarca. La cual le servirá de apoyo en el proceso de creación de un

software competitivo basada en las normas internacionales para el desarrollo de aplicaciones web especialmente en la ISO/IEC/IEEE 29119.

1.2. Objetivos de la Investigación

Objetivo general:

Construir una metodología que permita la evaluación de métricas de calidad en proyectos de desarrollo de software en la Universidad de Cundinamarca basados en la norma ISO/IEC/IEEE 29119

Objetivos específicos:

- Estudiar la información referente a la norma ISO/IEC/IEEE 29119
- Plantear una metodología acorde a los parámetros establecidos en la norma valorando el software a través de herramientas de testeo.
- Construir un modelo de medición para la evaluación de productos informáticos orientados a métricas de calidad de software definidos en la metodología propuesta.
- Definir y evaluar un escenario de prueba para la aplicación de la metodología propuesta, para estimar la calidad de los productos software respecto a las normativas y estándares

1.3. Justificación e Impacto

Con este proyecto se busca establecer una metodología con la cual se evalúe la calidad de los productos de software desarrollados por el programa de ingeniería de sistemas en la Universidad de Cundinamarca. La cual le servirá de apoyo en el proceso de creación de un software competitivo. Por otro lado, la metodología estará basada en las normas internacionales para el desarrollo de aplicaciones web especialmente en la ISO/IEC/IEEE 29119 y en cada una de sus partes dedicada a cada etapa del ciclo de vida del producto informático.

Se definirá una guía y herramientas que faciliten la valoración de los distintos proyectos de desarrollo en aplicaciones web. La Universidad de Cundinamarca al usar esta metodología hará que se mejore de manera notable el análisis por parte de los ingenieros de desarrollo en el momento de dedicarse a diseñar, desarrollar e implementar un software.

Al definir parámetros de calidad aplicados a un producto informático se generará un valor agregado y competitivo, en la fabricación de software generados al interior de la Universidad para el mercado interno y externo de la organización

2. MARCO TEÓRICO

En este capítulo se definirá los conceptos generales sobre calidad de software, sus antecedentes históricos, brindando de esta manera un enfoque general acerca del proceso de diseño y desarrollo de un producto software, posteriormente se establecerá el aspecto legal y finalmente los antecedentes referentes al desarrollo de la metodología para evaluar la calidad de software.

2.1. Antecedentes de la investigación históricos e Investigativos

EL desarrollo de software es una de las practicas con mayor auge e impacto en el mundo, en la creación de un producto de software es importante involucrar el concepto de calidad de software, que es el conjunto de cualidades medibles y específicas que dependen del tipo de software que se va a construir, para determinar su utilidad.(SENA, 2011)

Desarrollo Histórico de modelos de Calidad de Software

Los modelos de calidad de software más característicos son:

- **Modelo de McCall**: Definido en 1977 por Jim Mc Call, donde se identifican 11 factores de calidad relacionados con 23 criterios característicos del software, bajo la mirada de 3 perspectivas: la revisión del producto, la operación del producto y la transición del producto.

- **Modelo de Boehm:** Este modelo fue planteado en 1978, la característica principal fue la definición de la calidad como una "utilidad general", se centró en una estructura jerárquica cada una de las cuales contribuye a la calidad total.

- **Modelo de Evans y Marciniak**: se creó en 1987 es una alternativa al modelo de McCall, que define doce factores que se agrupan en tres categorías: diseño, rendimiento y adaptación.

- **Modelo Deutsch y Willis:** Fue definido en 1988 como una variante del modelo McCall, tiene quince factores agrupados en cuatro categorías: funcionamiento, rendimiento, cambio y gestión.

- **Modelo de calidad ISO 9126**: Fue planteado en 1991 como un estándar internacional para la medición de la calidad del software. Se derivó del modelo de McCall. El modelo de calidad ISO 9126-1

tiene dos partes principales que consisten en atributos de calidad externa e interna y atributos de calidad de uso.

- **Modelo Dromey:** En 1992 Dromey genera un marco de trabajo para evaluar los requisitos o requerimientos, el diseño y la implementación. El marco consta de tres elementos: El modelo de calidad de requisitos, el modelo de calidad de diseño y por último el modelo de calidad de implementación.

- **Modelo FURPS+:** Robert Grady de Hewlett Packard en 1992 planteo este modelo en donde las características de los productos de software las definió en: Requisitos Funcionales y los Requisitos No Funcionales **(F Functional Requirements y Non Functional Requirements)**, la usabilidad **(U Usability),** la fiabilidad **(R Reliability),** el rendimiento **(P Performance)** y por ultimo el soporte **(S Supportability),** a esto hace referencia la sigla **FURPS+.**

- **Modelo SEI:** En 1995 el Instituto de Ingeniería de Software (SEI), formulo un informe sobre los Atributos de Calidad (Informe Técnico CMU / SEI-95-TR-021) donde se centró en los atributos de rendimiento, fiabilidad y seguridad

- **Modelo ISO 25000 (SQuaRE) :** En el año de 2011 se generó este estandar derivado de la ISO/IEC 9126:1991, **SQuaRE** (The Systems and software Quality Requirements and Evaluation), este modelo

está definido para sistemas informáticos como a productos de software.(BOUKOUCHI, MARZAK, BENLAHMER, & MOUTACHAOUIK, 2013).

2.2. Antecedentes Investigativos

En el desarrollo sobre modelos de calidad de software y metodologías de evaluación de software se ha definido, la investigación" Adapting Software Quality Models: Practical Challenges, Approach, and First Empirical Results" realizada por M. Klas, C. Lampasona y J. Munch, sobre los modelos de calidad aclaran que se han propuesto muchos modelos para apoyar a las partes interesadas en el manejo de la calidad del software. Sin embargo, en la mayoría de los casos, los modelos de calidad no encajan perfectamente en el contexto de la aplicación de destino de aclaran en la mayoría de los casos, los modelos de calidad no encajan perfectamente en el contexto de la aplicación de destino; por lo tanto utilizan un enfoque basado en herramientas para la adaptación eficiente de modelos de calidad y como resultado son considerablemente más consistentes y adecuadamente el modelo adaptado que los obtenidos siguiendo un enfoque ad hoc.(Kläs & Lampasona, 2011)

En el trabajo de Syahrul Fahmy, Nurul Haslinda, Wan Roslina y Ziti Fariha en el año 2012; de la Universidad College en Malasia , indagan sobre un modelo apropiado para evaluar la calidad del software en un e-Book, donde realizan una comparación para identificar las mejores características de

calidad que se deben utilizar en la evaluación del e-Book. Se propone una extensión del modelo de evaluación ISO 9126, categorizando cinco características de calidad para e-Book .(Fahmy, Haslinda, Roslina, & Fariha, 2012)

Un importante trabajo realizado por Inderpal Singh en el 2013 donde expone diferentes modelos en referencia a la calidad estos modelos incluyen diferentes factores en relación con los requisitos funcionales y no funcionales, la fiabilidad, la medición de la calidad del software; en donde se da gran relevancia a las fases de planificación, los requisito, la implementación de codificación.(Singh, 2013).

En el año 2014 José P. Miguel , David Mauricio y Glen Rodríguez en su trabajo " A Review of Software Quality Models for the Evaluation of Software Products" aclaran la importancia de medir y evaluar la calidad de un producto de software que se evidencia como un factor crítico en muchas empresas. Además la existencia de muchos modelos muy generalizados dan como consecuencia la dificultad de aplicarlos a casos específicos, por lo tanto la utilización de modelos de calidad a la medida permiten obtener características de calidad apropiadas a un producto de software específico.(Miguel, Mauricio, & Rodríguez, 2014)

2.3. La industria enfocada a pruebas y calidad de software

En la actualidad existen empresas dedicadas a utilizar diferentes técnicas que permiten evaluar los criterios referentes a calidad de software, ayudando a cumplir con los objetivos y funcionalidad de este, entre las empresas dedicadas a realizar pruebas de software se encuentran:

- **Sogeti**: Es una compañía dedicada a brindar soluciones informáticas, dedicada principalmente al testing y calidad de software, ofreciendo su servicio a través de 12000 profesionales en 15 países de Europa, Estados Unidos e India ,Sogeti es el creador de las metodologías estándar del mercado: TMap® (Test Management Approach) y TPI® (Test Process Improvement). Además, es autor de numerosas publicaciones sobre Calidad de Software, entre ellas el informe anual "World Quality Report", y creador del evento Testing & Tools Day”(“GRUPO SOGETI,” n.d.)

- **Applause**: Esta empresa tiene en cuenta todo el ciclo de vida del software, utilizando herramientas para el análisis y evaluación del software. Con funcionalidad, usabilidad, la localización, la carga y las pruebas de seguridad, la comunidad global Applause de testeadores profesionales le ayuda en cada paso del camino para que pueda gestionar la calidad de sus aplicaciones desde el concepto hasta el

lanzamiento.(“Software Testing Tools & Services - Applause,” n.d.)

- **QActions:** Empresa dedicada a brindar servicios de consultoría, control y aseguramiento de software esta empresa ofrece distintos tipos de prueba como son las pruebas funcionales, pruebas técnicas, pruebas de rendimiento, pruebas de volumen y pruebas de compatibilidad. Esta empresa en particular ofrece distintos test pertenecientes a las pruebas funcionales como son test de integración, test del sistema, pruebas de regresión, test de usabilidad y pruebas de aceptación de usuarios.(“QActions,” n.d.)

- **SQS (Software Quality System):** Esta empresa europea dedicada a la consultoría de software donde la empresa busca asegurar la calidad de los desarrollos de los clientes. Esta empresa busca la satisfacción del cliente al 100% donde lo guían en caso de que sea necesaria una reducción de costos en el desarrollo, también los guía para que sus productos se adapten a algún tipo de normativa dirigida a la calidad, esta organización .(“SQS S.A. - Servicios de Consultoría de Calidad de Software y Testing,” n.d.)

- **MTP software quality assurance:** consultora especializada en servicios de ingeniería y calidad de software, tiene certificación TMMi4 para los procesos de testing y calidad de software. TMMi (Test Maturity Model integrated) es el estándar internacional

promovido por la TMMi Foundation, que mide y mejora los procesos y actividades relacionados con el diseño y ejecución de servicios de Testing y Calidad del Software.(“Home,” n.d.).

En el contexto local , en Colombia existen varias empresas de dicadas a la industria de las pruebas y calidad de software , entre ellas tenemos :

- **Finding:** Empresa Colombiana fundada en el 2006 con el apoyo de la universidad de Buenaventura. Ofrece distintos tipos de servicios como son: Análisis, training (formacion de clientes y desarrolladores), testing “Permite validar, verificar y conocer con claridad si su sistema de información (software) cumple con la necesidad actual de su compañía, sin arriesgar presupuestos, tiempos y alcance de sus proyectos informáticos.”(“Finding |,” n.d.)

- **Greensqa:** Empresa 100% Colombiana establecida en el 2002, desde entonces adquiriendo experiencia en la evaluación de calidad de software aplicados a distintos sectores “Servicios Públicos, Financiero, Salud, Solidario, Educativo y Comercial, así mismo se han finalizado exitosamente proyectos de implementación de Sistemas de Gestión de Calidad con base en la norma ISO9001 y el modelo CMMI-SVC 1.3 en organizaciones del sector IT” Por tanto se encuentran actualizados en los distintos procesos de software.(“Inicio - GreenSQA,” n.d.)

- **Choucair Testing S.A.:** Compañía Colombiana creada en 1999, ofrece servicios de pruebas de software para reducir los riesgos, a través de la investigación aplicando metodologías certificadas.(“Inicio - Choucair Testing,” n.d.)

2.4. Bases teóricas y legales

En este apartado se definirá conceptos sobre calidad de software, testeo de software, además como se involucra este proceso en el ciclo de vida de un producto de software.

2.4.1 Software

El software son programas informáticos, procedimientos, datos y documentación asociada al funcionamiento de un sistema informático. (IEEE, 1990) estos elementos son necesarios para asegurar la calidad del proceso de desarrollo del software.

2.4.2 Calidad de software

La calidad de un software hace referencia a un conjunto de cualidades que lo caracteriza, se puede decir que un software de calidad cuenta con eficiencia, flexibilidad, corrección, confiabilidad, mantenibilidad, portabilidad, usabilidad, seguridad e integridad. Según el Instituto de ingenieros eléctrico y electrónicos (IEEE), la calidad de software es:

- El grado en que un sistema, componente o proceso satisface los requisitos o requerimientos.

- El grado en que un sistema, componente o proceso cumple con las necesidades o expectativas de los usuarios.(Galin, 2004)

2.4.3 Aseguramiento de la calidad de software

Es un proceso sistemático o un patrón unificado que permite brindar acciones necesarias para brindar la suficiente confianza en que un artículo o producto cumple con requerimientos técnicos, también es el conjunto de actividades destinadas a evaluar el proceso mediante el cual los productos se desarrollan o fabrican, en este caso el producto es el software.

2.4.4 Factores de la calidad de software

Los modelos clásicos de Calidad de software como el de Mccall, define 11 factores , los cuales son enfocados en tres factores: los factores de operación del producto, factores de revisión del producto y los factores de transición del producto, en la Gráfica # 1 se especifica este modelo.

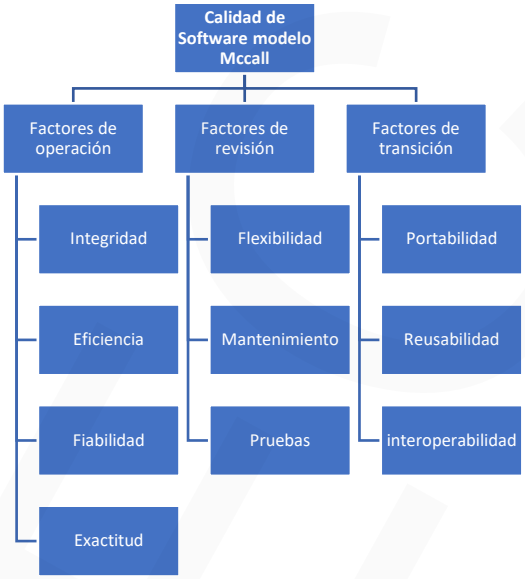
En la literatura existen diversos modelos, pero como referencia conceptual se toma el modelo de Mccall como modelo base(Galin, 2004)

2.4.5 Pruebas del Software

Las prueba del software es el proceso que se ejecuta en un programa o sistema informático con la intensión de encontrar errores, este proceso permite analizar una software para detectar las diferencias entre las condiciones existentes y las requeridas (Es decir, errores) (IEEE, 1990). las pruebas se clasifican en:

Pruebas Funcionales o Caja negra: Identifica los errores debido al mal funcionamiento del software de acuerdo a las salidas del sistema, se centran en los requisitos funcionales del sistema.

Gráfica # 1Modelo Mccall



Pruebas estructurales o caja blanca: Estas pruebas se enfocan en el comportamiento interno del sistema, con lla finalidad de encontrar fallas en la codificación.

Cuando se realiza el proceso de aseguramiento de calidad por costos de implementación se sugiere que se realice solo una clase de pruebas, actualmente en la industria se centran la realización de pruebas funcionales por su bajo costo.

2.4.6 Aspectos Normativos

Dentro de este estudio para la creación de la metodología de evaluación de productos de software se centra en estándar **ISO/IEC/IEEE 29119**, Este estándar especifica definiciones y conceptos en las pruebas de software; por lo tanto se definen:

- Procesos de pruebas
- Documentación de pruebas
- Técnicas de ensayo

Este estándar se divide en 5 elementos principalmente:

- ISO / IEC 29119-1: Conceptos y definiciones (publicado en septiembre de 2013)
- ISO / IEC 29119-2: Procesos de prueba (publicado en septiembre de 2013)
- ISO / IEC 29119-3: Documentación de prueba (publicada en septiembre de 2013)
- ISO / IEC 29119-4: Técnicas de prueba (publicación a finales de 2014)
- ISO / IEC 29119-5: Pruebas dirigidas (publicación en 2015)

2.4.7 Definición de pruebas de la norma ISO/IEC/IEEE 29119

En el estudio de la norma ISO/IEC/IEEE 29119, esta realiza una descripción detallada de cada prueba que se debe realizar a un producto de software, por lo tanto, se realiza un cuadro resumen de las pruebas a las que se deben someter un software, además se realiza un análisis comparativo sobre tres criterios de comparación: dificultad, calidad y prioridad , con el fin de identificar las pruebas que son aplicables a nuestro caso de estudio.

Tabla 1 Descripción de Pruebas a un producto de software

Tipo de prueba	**Evaluación**	**Dificultad 1-5**	**Calidad 1-5**	**Prioridad 1-5**	**Descripción**
· Prueba de accesibilidad	No	5	5	2	los usuarios que se tienen en cuenta tienen discapacidades que afectan a su manera de utilizarlo
· Prueba de caja negra	No	2	2	2	Se limitan a que el tester pruebe con “datos” de entrada y estudie

Tipo de prueba	Evaluación	Dificultad 1-5	Calidad 1-5	Prioridad 1-5	Descripción
					como salen, sin preocuparse de lo que ocurre en el interior.
· Prueba de capacidad	No	2	3	1	hasta dónde se puede llegar cargando el sistema antes de que sea inutilizable
· Prueba de compatibilidad	Si	3	5	4	Mostrar una calidad adecuada en el software y así verificar que funcionará con normalidad en todos los navegadores o se podrá instalar en todos los sistemas

Tipo de prueba	Evaluación	Dificultad 1-5	Calidad 1-5	Prioridad 1-5	Descripción
					operativos que veas necesario.
· Prueba de resistencia	Si	4	5	5	ejecuta un sistema de tal manera que requiera una frecuencia o un volumen anormal de recursos
· Prueba exploratoria	Si	5	5	5	Podría decirse que el testing exploratorio se completa con cinco elementos: un reporte de errores sobre un producto, realizado por un tester en particular en un tiempo determinado y con una misión a

Tipo de prueba	Evaluación	Dificultad 1-5	Calidad 1-5	Prioridad 1-5	Descripción
					cumplir. * Exploración del producto. * Diseño de pruebas. * Ejecución de pruebas. * Heurísticas. * resultados revisables.
· Prueba de instalabilidad	Si	4	4	3	pruebas realizadas para evaluar si un elemento de prueba o conjunto de elementos de prueba se pueden instalar como se requiere en todos los entornos especificados

Tipo de prueba	Evaluación	Dificultad 1-5	Calidad 1-5	Prioridad 1-5	Descripción
· Prueba de carga	Si	4	5	5	Eficiencia en el desempeño realizado para evaluar el comportamiento de un elemento de prueba en las condiciones previstas de carga variable, generalmente entre las condiciones previstas de uso bajo, típico, y el pico.
· Prueba de mantenibilidad	No	5	4	1	La capacidad de un producto software para ser modificado. Las modificaciones pueden incluir correcciones,

Tipo de prueba	Evaluación	Dificultad 1-5	Calidad 1-5	Prioridad 1-5	Descripción
					mejoras o adaptación del software a cambios en el entorno, en los requerimientos o en las especificaciones funcionales.
· Prueba organizacional	Si	3	4	3	Documento nivel ejecutivo- que describe los propósitos, objetivos y alcance global de la prueba dentro de un proyecto, y que expresan por qué se realiza la prueba y lo que se espera lograr.

Tipo de prueba	Evaluación	Dificultad 1-5	Calidad 1-5	Prioridad 1-5	Descripción
· Prueba de rendimiento	Si	4	5	4	Determinar lo rápido que realiza una tarea un sistema en condiciones particulares de trabajo. También puede servir para validar y verificar otros atributos de la calidad del sistema, tales como la escalabilidad, fiabilidad y uso de los recursos.
· Prueba de portabilidad	No	5	2	1	Para evaluar la facilidad con la que un elemento de prueba puede ser transferido de un entorno de

Tipo de prueba	Evaluación	Dificultad 1-5	Calidad 1-5	Prioridad 1-5	Descripción
					hardware o software a otro.
· Prueba de procedimiento	Si	3	3	3	Si las instrucciones de procedimiento para interactuar con un elemento de prueba o el uso de sus productos cumplen con los requisitos del usuario y apoyar a los fines de su utilización.
· Prueba de regresión	Si	5	4	4	tipo de pruebas de software que intentan descubrir errores (bugs), carencias de funcionalidad, o divergencias funcionales con

Tipo de prueba	Evaluación	Dificultad 1-5	Calidad 1-5	Prioridad 1-5	Descripción
					respecto al comportamiento esperado del software, causados por la realización de un cambio en el programa.
· Prueba de fiabilidad	No	4	3	2	Para evaluar la capacidad de un producto de ensayo para llevar a cabo sus funciones requeridas, incluyendo la evaluación de la frecuencia con que se producen fallos, cuando se utiliza en condiciones establecidas por

Tipo de prueba	Evaluación	Dificultad 1-5	Calidad 1-5	Prioridad 1-5	Descripción
					un período de tiempo especificado. Prevención de fallos. Tolerancia a fallos
· Prueba basada en riesgo	Si	3	5	5	Gestión, selección, priorización, y el uso de las actividades de prueba y recursos se basan conscientemente sobre los tipos y niveles de riesgo analizados.
· Prueba de escenario	Si	2	4	4	Permite establecer cruces por medio de Matrices de Casos versus

Tipo de prueba	Evaluación	Dificultad 1-5	Calidad 1-5	Prioridad 1-5	Descripción
					Escenarios que ayudan a aumentar la cobertura de las pruebas; es decir, identificar todos los casos de prueba, cuya ejecución certificarán la calidad del sistema.
· Prueba con guion	No	3	1	1	Este término se aplica normalmente a las pruebas ejecutadas manualmente, en lugar de la ejecución de un script automatizado.

Tipo de prueba	Evaluación	Dificultad 1-5	Calidad 1-5	Prioridad 1-5	Descripción
· Prueba de seguridad	Si	5	5	5	Evaluar el grado en que un elemento de prueba, y los datos asociados y la información están protegidos para que las personas o sistemas no autorizados no puedan utilizar, leer, o modificarlos, y autorizó a personas o sistemas no se les niega el acceso a ellos.
· Prueba basada en especificación	No	2	2	2	sinónimos de pruebas basadas en las especificaciones

Tipo de prueba	Evaluación	Dificultad 1-5	Calidad 1-5	Prioridad 1-5	Descripción
					incluyen caja-negra o pruebas de caja cerrada.
· Prueba estática	Si	3	3	3	Tienen por objetivo evaluar la conformidad de un producto con respecto a especificaciones, planes y asegurar la integridad técnica y conceptual de los cambios.
· Prueba de estrés	Si	5	5	5	eficiencia en el desempeño pruebas realizadas para evaluar el comportamiento de un elemento de prueba bajo

Tipo de prueba	Evaluación	Dificultad 1-5	Calidad 1-5	Prioridad 1-5	Descripción
					condiciones de carga por encima de los requisitos previstos o especificados de capacidad
· Prueba estructural	Si	4	4	4	Se denomina cajas blancas a un tipo de pruebas de software que se realiza sobre las funciones internas de un módulo. Las pruebas de caja blanca están dirigidas a las funciones internas.
· Prueba de condición	SI	4	5	4	Esta prueba se asegura de que cada condición

Tipo de prueba	Evaluación	Dificultad 1-5	Calidad 1-5	Prioridad 1-5	Descripción
					del programa no contenga errores.
· Prueba de cobertura	NO	1	2	1	porcentaje que expresa el grado en el que el código fuente ha sido testeado
· Prueba de informe de preparación de datos	SI	4	5	5	Evaluación de los datos creados o seleccionados para satisfacer los requisitos de entrada para la ejecución de uno o más casos de prueba.
· Prueba de proceso de diseño e implementación	SI	3	5	5	Documento que describe el estado de cada requerimiento de datos de prueba

Tipo de prueba	Evaluación	Dificultad 1-5	Calidad 1-5	Prioridad 1-5	Descripción
· Prueba especificación de diseño	NO	1	2	1	Documento que especifica las características para ser probado y sus condiciones de prueba correspondientes
· Prueba de entorno	SI	5	4	5	Instalaciones, hardware, software, firmware, los procedimientos y la documentación destinados o utilizados para realizar pruebas de software
· Prueba de informe	NO	3	1	2	Documento que describe el cumplimiento de cada requisito

Tipo de prueba	Evaluación	Dificultad 1-5	Calidad 1-5	Prioridad 1-5	Descripción
ambiental disposición					entorno de prueba
· Prueba puesta en marcha de procesos	SI	3	4	5	Proceso de prueba dinámica para establecer y mantener un entorno de prueba requerido
· Prueba de ejecución	Si	5	4	5	Proceso de prueba dinámica de la ejecución de los procedimientos de prueba creados en el proceso de diseño de la prueba y la aplicación en el entorno de prueba preparada, y el

Tipo de prueba	Evaluación	Dificultad 1-5	Calidad 1-5	Prioridad 1-5	Descripción
					registro de los resultados
· Prueba de gestión	Si	4	5	5	Planificación, programación, la estimación, seguimiento, información, control y realización de los activos de prueba
· Prueba de proceso de planificación	Si	4	4	4	Proceso de gestión de prueba utilizado para completar la planificación de pruebas y desarrollar planes de prueba
· Prueba de requisito	Si	3	4	4	buscar discrepancias entre los

Tipo de prueba	Evaluación	Dificultad 1-5	Calidad 1-5	Prioridad 1-5	Descripción
					requerimientos y la ejecución del software.
· Prueba de resultado	Si	4	5	4	si el resultado real observado como elemento de prueba de salida se corresponde con el resultado esperado o si se observaron desviaciones
· Prueba de especificaciones	Si	4	4	4	La documentación completa del diseño de la prueba, casos de prueba y procedimientos de prueba o un elemento de

Tipo de prueba	Evaluación	Dificultad 1-5	Calidad 1-5	Prioridad 1-5	Descripción
					prueba específica
· Prueba de informe de estado	No	2	2	1	proporciona información sobre el estado de la prueba que se realiza en un período de referencia especificado
· Prueba de artículos	Si	3	3	3	Artículos prueba puede incluir cosas tales como la documentación, los guiones, los insumos, los resultados esperados, archivos, bases de datos, el medio ambiente

Tipo de prueba	Evaluación	Dificultad 1-5	Calidad 1-5	Prioridad 1-5	Descripción
· Pruebas de volumen	No	4	3	1	Pruebas de eficiencia en el desempeño realizado para evaluar la capacidad del producto de ensayo para procesar volúmenes de datos (por lo general en o cerca de la capacidad máxima especificada) especifica en términos de capacidad de procesamiento, capacidad de almacenamiento, o ambos.

Tipo de prueba	Evaluación	Dificultad 1-5	Calidad 1-5	Prioridad 1-5	Descripción
· Prueba de verificación y validación	Si	5	5	5	Proceso de revisión que verifica que el sistema de software producido cumple con las especificaciones y que logra su cometido. Es normalmente una parte del proceso de pruebas de software de un proyecto, que también utiliza técnicas tales como evaluaciones, inspecciones y tutoriales.

Tipo de prueba	Evaluación	Dificultad 1-5	Calidad 1-5	Prioridad 1-5	Descripción
· Prueba genérica	No	4	2	1	El software tiene un ciclo de vida esperado de su concepción inicial hasta su eventual retiro. Las pruebas de software se lleva a cabo en el contexto más amplio del desarrollo y mantenimiento de software, esta cláusula se describe un ciclo de vida de desarrollo de software de ejemplo

Debido a la gran cantidad de pruebas a realizar resulta complicado y lento desarrollarlas de forma manual, es por esto que se deben tener

herramientas que automaticen y faciliten dicha labor al comité de evaluación de la Universidad de Cundinamarca; Atendiendo a estas consideraciones resultaría un tanto tedioso contar con 25 herramientas diferentes que permitan realizar la evaluación, considerando estas apreciaciones la norma ISO/IEC/IEEE 29119 parte 2 se refiere al proceso de pruebas donde se especifica un modelo de pruebas de procesos generales para el ciclo de vida de un proyecto software establece 3 capas de cobertura:

1) Gestión de pruebas (gestionar fase de pruebas)

2) Pruebas funcionales (procesos dinámicos)

3) Pruebas de carga y rendimiento

Por ello se hace necesario clasificar las 25 pruebas en estas 3 capas de cobertura de la siguiente manera según la Tabla 2 :

Tabla 2 Elección de tipos de pruebas según capa de cobertura

Tipo de prueba	**Pruebas**
Prueba de compatibilidad	Gestión de pruebas
Prueba exploratoria	Gestión de pruebas
Prueba de instalabilidad	Gestión de pruebas

Prueba organizacional	Gestión de pruebas
Prueba de escenario	Gestión de pruebas
Prueba de condición	Gestión de pruebas
Prueba de informe de preparación de datos	Gestión de pruebas
Prueba de proceso de diseño e implementación	Gestión de pruebas
Prueba de entorno	Gestión de pruebas
· Prueba de gestión	Gestión de pruebas
· Prueba de proceso de planificación	Gestión de pruebas
· Prueba de requisito	Gestión de pruebas
· Prueba de especificaciones	Gestión de pruebas
· Prueba de artículos	Gestión de pruebas
Prueba de procedimiento	funcional
Prueba de regresión	funcional

Prueba basada en riesgo	funcional
Prueba estructural	Funcional
Prueba puesta en marcha de procesos	funcional
· Prueba de resultado	funcional
· Prueba de verificación y validación	funcional
Prueba de carga	carga y rendimiento
Prueba de resistencia	carga y rendimiento
Prueba de rendimiento	carga y rendimiento
Prueba de estrés	carga y rendimiento

2.4.8 Lenguaje unificado de modelado UML

El modelado es el diseño de aplicaciones de software antes de la codificación. El modelado es una parte esencial en los proyectos de software. Un modelo desempeña el papel análogo en el desarrollo de

software como son los planos en la construcción de un edificio. Utilizando un modelo, permite asegurar la funcionalidad, la escalabilidad, robustez y seguridad antes de la implementación; por lo tanto el modelado es la única manera de visualizar su diseño y comprobarlo en función de los requisitos antes de que los ingenieros de desarrollo empiecen a codificar.

Para realizar el modelado existen el marco de trabajo UML 2.0, donde se definen 3 tipos de diagramas, divididos en tres categorías:

- Diagramas de estructura estática
- Diagramas de comportamiento
- Diagramas de interacción

Los diagramas de estructura incluyen:

- El diagrama de Clases
- El diagrama de Objetos
- El diagrama de Componentes
- El Diagrama de Estructura Compuesta
- El Diagrama de Paquetes
- El Diagrama de Despliegue.

Los diagramas de Comportamiento incluyen:

- El Diagrama de Casos de Uso
- Diagrama de actividad
- Diagrama de máquina de estado.

Diagramas de interacción lo conforman:

- El Diagrama de Secuencias
- Diagrama de Comunicación
- Diagrama de Tiempo
- Diagrama de Interacción (The Object Management Group, 2004)

2.4.9 Elementos para realizar el modelado UML

Según el estándar UML el modelado se compone de una seria de diagramas y categorías o dimensiones , en la siguiente tabla se realiza un resumen sobre los componentes que debe incluir cada diagrama en el framework UML 2.0.

Tabla 3 Componentes de los diagramas UML

DIAGRAMA	COMPONENTES	OPCIONAL	USO DE COMPONENTES
DIAGRAMA DE ACTIVIDADES Los diagramas de actividades muestran el flujo de trabajo desde el punto de inicio hasta el punto final detallando muchas de las rutas de decisiones que existen en el progreso de eventos contenidos en la actividad	Actividad	NO	es la especificación de una secuencia parametrizada de comportamiento
	Acción	NO	Una acción representa un solo paso dentro de una actividad.
	Restricciones de acción	SI	Son adjuntas a una acción se pueden presentar antes de realizar una acción o posteriormente.
	Flujo de control	NO	Muestra el flujo de control de una acción sobre otra.
	Nodo inicial	NO	Describe el comienzo de cualquier actividad.
	Nodo final		
	Nodo final de actividad	NO	El nodo final de actividad se describe como un círculo

DIAGRAMA	COMPONENTES	OPCIONAL	USO DE COMPONENTES
			con un punto dentro del mismo.
	Nodo final de flujo	NO	El nodo final de flujo se describe como un círculo con una cruz dentro del mismo.
	Flujo de objetos	SI	Un flujo de objeto es la ruta a lo largo de la cual pueden pasar objetos o datos
	Nodos de decisión y combinación	SI	Los flujos de control que provienen de un nodo de decisión tendrán condiciones de guarda que permitirán el control para fluir si la condición de guarda se realiza
	Nodos de bifurcación y unión	SI	Estos indican el comienzo y final de hilos actuales de control

DIAGRAMA	COMPONENTES	OPCIONAL	USO DE COMPONENTES
	Región de expansión	SI	Es una región de actividad estructurada que se ejecuta muchas veces.
	Gestores de excepción	SI	Son de uso exclusivo cuando para la realización de una actividad se tengan restricciones u observaciones.
	Región de actividad interrumpible	SI	Rodea un grupo de acciones que se pueden interrumpir.
	Partición	SI	las particiones se usan para separar acciones dentro de una actividad
DIAGRAMA DE CASOS DE USO	Actores	NO	Los actores representan los roles que pueden incluir usuarios humanos, un hardware externo u otros sistemas. (Los actores pueden generalizar otros actores.)

DIAGRAMA	COMPONENTES	OPCIONAL	USO DE COMPONENTES
Son un medio de comunicación con los usuarios y otros interesados acerca de lo que se piensa hacer del sistema.	Casos de uso	NO	- Notación para usar un caso de uso es una línea de conexión con una punta de flecha opcional mostrando la dirección del control. - Pueden contener la funcionalidad de otro caso de uso. - Se pueden incluir por uno o más casos de uso. - Se puede usar para extender el comportamiento de otro. - El punto al cual un caso de uso extendido se agrega se puede definir por medio de un punto de extensión. - Un Caso de Uso normalmente incluye: • Nombre y Descripción • Requisitos • Restricciones • Escenarios

DIAGRAMA	COMPONENTES	OPCIONAL	USO DE COMPONENTES
			• Diagramas de escenarios • Información adicional.
	Inclusión de casos de uso	SI	Los casos de uso pueden contener la funcionalidad de otro caso de uso como parte de su proceso normal. Los casos de uso se pueden incluir por uno o más casos de uso, ayudando a reducir el nivel de duplicación de la funcionalidad.
	Casos de uso extendidos	SI	Se pude usar para extender el comportamiento de otro. (En caso de que un usuario necesite permiso de otro para el acceso a ese caso de uso).

DIAGRAMA	COMPONENTES	OPCIONAL	USO DE COMPONENTES
	Puntos de extensión	SI	Se utiliza un punto de extensión en caso de que se cumplan algunas condiciones específicas para ejecutar el caso de uso relacionado.
	Límite del sistema	SI	Usualmente se usa para mostrar casos de uso dentro del sistema y actores fuera del sistema.
DIAGRAMA DE SECUENCIA	Línea de vida	NO	Una línea de vida representa un participante individual en un diagrama de secuencia. Se puede presentar un actor, una clase, una entidad, o otra línea de vida.
	Mensajes	SI	Deben ser mostrados como flechas.

DIAGRAMA	COMPONENTES	OPCIONAL	USO DE COMPONENTES
			- Síncronos: denotados por una flecha con punta oscura - Asíncronos: denotados por una flecha con un punta en línea. - Llamadas o señales: denotado por una flecha punteada
	Ocurrencia de ejecución	SI	Denota la ocurrencia de ejecución o activación de un foco de control.
	Mensajes self	SI	Puede representar una llamada recursiva de una operación, o un método llamando a otro método perteneciente al mismo objeto.
	Mensajes perdidos y encontrados	SI	Los mensajes perdidos son aquellos que han sido enviados pero que no han

DIAGRAMA	COMPONENTES	OPCIONAL	USO DE COMPONENTES
			llegado al destino esperado, o que han llegado aún.
	Inicio y final de línea de vida	NO	Una línea de vida se puede crear o destruir durante la escala de tiempo representada por un diagrama de secuencia.
	Restricciones de tiempo y duración	SI	Al configurar una restricción de duración para un mensaje, el mensaje se mostrará como una línea inclinada.
	Fragmentos combinados	SI	Fragmento combinado es una o más secuencias de procesos incluidas en un marco y ejecutadas bajo circunstancias nombradas específicas.
	Puerto	SI	Conexión entre un fragmento y un mensaje.

DIAGRAMA	COMPONENTES	OPCIONAL	USO DE COMPONENTES
	Descomposición en parte	SI	Permite mensajes de entre e intra objetos para que se muestren en el mismo diagrama.
	Continuaciones / Invariantes de Estado	SI	Se puede encontrar extendida por una o más líneas de vida en distintos objetos.
DIAGRAMA DE CLASE El diagrama de Clase muestra los bloques de construcción de cualquier sistema orientado a objetos.	Clases	NO	Es un elemento que define los atributos y comportamientos que un objeto podrá generar.
	Notación de clase	SI	Se usan para denotar el nombre de la clase, atributos y operaciones.
	Interfaces	NO	Si se usa una interfaz, se garantiza que las clases soporten un comportamiento requerido, que permite que el sistema trate los elementos no

DIAGRAMA	COMPONENTES	OPCIONAL	USO DE COMPONENTES
			relacionados en la misma manera. Se puede presentar en forma de circulo o rectangular con anotaciones.
	Tablas	SI	Una tabla es una clase estereotipada.
	Asociaciones	SI	Una asociación implica que dos elementos del modelo tienen una relación, usualmente implementada como una variable de instancia de una clase
	Generalizaciones	SI	Una generalización se usa para indicar herencia.
	Agregaciones	SI	Las agregaciones se usan para describir elementos que están compuestos de componentes más pequeños.

DIAGRAMA	COMPONENTES	OPCIONAL	USO DE COMPONENTES
	Clase asociación	SI	Una clase asociación es una estructura que permite una conexión de asociación para tener conexiones y atributos.
	Dependencias	SI	Una dependencia se usa para modelar un alto rango de relaciones dependientes entre elementos del modelo.
	Trazado	SI	La relación de trazado es una especialización de una dependencia, vinculando elementos del modelo o conjuntos de elementos que representan la misma idea a través de los modelos.
	Relaciones	NO	Se usa para expresar trazabilidad e integridad en el modelo.
	Anidamientos	SI	Un anidamiento es un conector que muestra que

DIAGRAMA	COMPONENTES	OPCIONAL	USO DE COMPONENTES
			el elemento fuente se anida dentro del elemento destino.
DIAGRAMA DE ESTRUCTURA COMPUESTA Un diagrama de estructura compuesta es un diagrama que muestra la estructura interna de un clasificador, incluyendo sus puntos de interacción a otras partes del sistema.	Parte	SI	Una parte es un elemento que representa un conjunto de una o más instancias que pertenecen a una instancia del clasificador contenida
	Puerto	SI	Un Puerto es un elemento escrito que representa una parte visible externa de una instancia del clasificador contenido.
	Interfaces	SI	Una interfaz es similar a una clase pero con un número de restricciones. Todas las operaciones de la interfaz son públicas y abstractas, y no proveen ninguna implementación predeterminada. Todos los atributos de la interfaz

DIAGRAMA	COMPONENTES	OPCIONAL	USO DE COMPONENTES
			deben ser constantes. Sin embargo, mientras que una clase puede solo heredar de una sola súper-clase, puede implementar interfaces múltiples.
	Delegar	SI	Un conector delegar se usa para definir los trabajos internos de los puertos e interfaces externas del componente.
	Colaboración	SI	Una colaboración define un conjunto de roles cooperativos usados colectivamente para ilustrar una funcionalidad especifica.
	Enlace de roles	SI	Un conector enlace de roles se dibuja desde una colaboración a un clasificador que completa el rol

DIAGRAMA	COMPONENTES	OPCIONAL	USO DE COMPONENTES
	Representa	SI	Un conector representa se puede dibujar desde una colaboración a un clasificador para mostrar que una colaboración se usa en el clasificador.
	Ocurrencia	SI	Un conector ocurrencia se puede dibujar desde una colaboración a un clasificador para mostrar que la colaboración representa (sic) el clasificador.
DIAGRAMA DE MÁQUINA DE ESTADO	Estado	NO	Se denotan con su nombre dentro del mismo.
	Estado inicia y final	NO	Indicador del estado.
	Transiciones	SI	Una transición puede tener un disparador, una guarda y un efecto, como a continuación.

DIAGRAMA	COMPONENTES	OPCIONAL	USO DE COMPONENTES
	Acciones de estado	SI	Define las acciones que ocurren en los eventos, o acciones que siempre ocurren.
	Transacciones recursivas	SI	Un estado puede tener una transición que retorna a sí misma.
	Estados compuestos	SI	Se relaciona de componentes grandes a sus subcomponentes.
	Punto de entrada	SI	Se puede presentar dos tipos de punto inicial en el cual se puede tener un inicio norma o por otro lado con condiciones o simplemente desde otro proceso.
	Punto de salida	SI	Se pueden tener puntos de salida nombrados lo que nos permite tener varios puntos de salida.

DIAGRAMA	COMPONENTES	OPCIONAL	USO DE COMPONENTES
	Pseudo estado elección	SI	Un pseudo estado se muestra como un diamante con una transición llegando y dos o más transiciones saliendo.
	Pseudo estado unión	SI	Los pseudo estados unión se usan para unir transiciones múltiples
	Pseudo estado terminar	SI	Ingresar un pseudo terminar indica que la línea de vida de la máquina de estado ha terminado.
	Estado historial	SI	Un estado historial se usa para recordar el estado anterior de una máquina de estado cuando fue interrumpida.
	Regiones recientes	SI	Un estado se puede dividir en regiones conteniendo sub estados que existen y se ejecutan concurrentemente.

DIAGRAMA	COMPONENTES	OPCIONAL	USO DE COMPONENTES
DIAGRAMA DE DESPLIEGUE Un Diagrama de Despliegue modela la arquitectura en tiempo de ejecución de un sistema	Nodo	NO	Un Nodo es un elemento de hardware o software.
	Instancia de nodo	SI	Se puede distinguir desde un nodo por el hecho de que su nombre esta subrayado y tiene dos puntos antes del tipo de nodo base.
	Estereotipo de nodo	SI	Un número de estereotipos estándar se proveen para los nodos.
	Artefacto	SI	Un artefacto es un producto del proceso de desarrollo de software, que puede incluir los modelos del proceso.
	Asociación	SI	En el contexto del diagrama de despliegue, una asociación representa una ruta de comunicación entre los nodos.

DIAGRAMA	COMPONENTES	OPCIONAL	USO DE COMPONENTES
	Nodo como contenedor	SI	Un nodo puede contener otros elementos, como componentes o artefactos.
Diagrama de componentes Los Diagramas de Componentes ilustran las piezas del software, controladores embebidos, etc. que conformarán un sistema.	Representando componentes	SI	Los componentes se representan como un clasificador rectangular con la clave «componente», opcionalmente el componente se puede mostrar como un rectángulo con un icono de componente en la esquina derecha arriba
	Interfaces requeridas	SI (Únicamente obligatoria en caso de que un componente lo requiera)	Esto permite que un componente provea los servicios que otro componente requiere.

DIAGRAMA	COMPONENTES	OPCIONAL	USO DE COMPONENTES
	Componentes con puertos	SI	Usar puertos con Diagramas de Componentes permite que se especifique un servicio o comportamiento a su entorno así como también un servicio o comportamiento que un componente requiere. Los puertos pueden especificar entradas, salidas así como también operar bidireccionalmente.

3. MARCO METODOLÓGICO

3.1. Naturaleza de la investigación

En este capítulo se analiza la forma como se realizó el proyecto, por lo tanto se definen los objetivos del mismo y la razón del desarrollo del proyecto, como primer elemento a analizar se define el tipo de estudio, el cual es un enfoque cuantitativo, como lo señala Hernández, Fernández y Baptista. (1991) un enfoque cuantitativo usa la recolección de datos para probar hipótesis, con base en la medición numérica y el análisis estadístico, para establecer patrones de comportamiento y probar hipótesis; en este proyecto se analiza las razones de construir una metodología que permita la evaluación de métricas de calidad en proyectos de desarrollo de software en la Universidad de Cundinamarca basados en la norma ISO/IEC/IEEE 29119

En el desarrollo del proyecto se utilizó un método teórico, el cual permite la construcción y desarrollo de la teoría científica para analizar de una forma más amplia los postulados antes mencionados, se cumplió por medio de análisis y síntesis, lo que culmino en una investigación documental, por otro lado, los resultados de la investigación se desarrollaron de una forma cuantitativa

Por otro lado, la secuencia que se realizó en la investigación fue la siguiente:

- Análisis bibliográfico
- Definición de la tendencia tecnológica orientada al desarrollo de software
- Comparación y análisis de las tendencias relevantes
- Creación de los parámetros para la implementación de la metodología
- Evaluación de la metodología en escenarios de prueba

3.2. Hipótesis

A lo largo de este estudio, donde se involucra la creación de software involucrando criterios de calidad de software se plantea la siguiente hipótesis:

- La creación de una metodología permite medir la calidad de software en los productos informáticos desarrollados en la Universidad de Cundinamarca.

Para analizar y explorar esta hipótesis se producen una serie de variables que identifican criterios para crear una metodología de evaluación y permitir cuantificar la calidad de un software , la variables que su utilizan son:

Criterios de evaluación de calidad de software

Tipos de pruebas de calidad de software

3.3. Población y Muestra

Para el análisis de esta propuesta se plantea como población de estudio los docentes, estudiantes y grupo de desarrollo de la Universidad de Cundinamarca en la extensión de Facatativá del programa académico ingeniería de sistemas.

La muestra que se utilizara de la población es no probabilística, ya que la característica de la investigación implica recoger y documentar las necesidades de la aplicación de una metodología, por lo tanto se realizaron encuestas a 42 personas que involucran a la totalidad de equipo de desarrollo del programa de ingeniería de sistemas, los docentes y un grupo de estudiantes pertenecientes al equipo de apoyo de desarrollo de software.

Se define la metodología con la cual la información fue recogida, el análisis que se empleó dentro de esta investigación de donde se plantea la encuesta titulada proceso de evaluación de calidad de software con el objetivo de determinar el conocimiento de estudiantes, docentes y equipo de desarrollo sobre el modo de evaluar aplicativos web, igualmente se tuvo en cuenta la percepción de calidad de software de cada participante.

De dicho cuestionario se puede obtener la siguiente ficha técnica:

Tabla 4Ficha técnica de la encuesta

Elemento	Descripción
Universo	**Estudiantes y docentes de la Universidad de Cundinamarca.**
Ámbito del estudio	**Universidad de Cundinamarca** **Facultad de Ingeniería** **Programa: Ingeniería de sistemas**
Diseño del cuestionario	**Formulario publicado en encuestasonline.com para los docentes y para los estudiantes sobre Drive forms. La creación de estas encuestas basadas en el estudio previo del estándar ISO/IEC/IEEE 29119.**
Tamaño muestral	**42 encuestas aplicadas.**
Procedimiento de muestreo	**Diseñado el investigador auxiliares y al asesor del**

Elemento	Descripción
	proyecto
Realización del trabajo de campo	**Universidad de Cundinamarca extensión Facatativá.**
Fecha de realización	**27 de agosto de 2016**
Grabación de los datos	**Herramienta gratuita de Drive.**
Análisis y elaboración del informe	**Basado en el libro de Rafael Romero Villafranca y Luis Rosa Zunica Ramajo titulado métodos estadísticos en ingeniería.**
Tabulaciones en el anexo	**A cargo del proponente.**

Los objetivos de este estudio se concretan de la siguiente manera:

- Identificar los conocimientos de estudiantes y docentes acerca de la evaluación de calidad de software y de la forma en que se realizan estos procesos dentro de la universidad.

- Determinar la prioridad que los participantes le dan a los diagramas de UML y a las pruebas que se usan en el aplicativo.

Finalmente, este estudio tiene como fin brindar soporte a la investigación previa dirigida a los estándares internacionales ISO/IEC/IEEE 29119 con el fin de sincronizar la Universidad con el exterior. Basados en la encuesta obtener una evaluación porcentual para cada parte de la valoración de calidad.

3.4. Técnicas e instrumentos de recolección de datos

Dentro de la investigación, la técnica para la realización del estudio está orientado a encuestas y una análisis documental, la literatura nos brinda diferentes métodos para la generación de los instrumentos, para la realización de la encuentras se utiliza los cuestionarios, que consisten en una serie de preguntas respecto a una o más variables a medir . En los cuestionarios se diseñaron preguntas cerradas donde se tiene opciones de respuestas delimitadas.

Por otro lado se utilizara análisis documental donde permite estudiar analizar información de manera objetiva, sistemática y cuantitativa.

3.5. Validez y confiabilidad

En el desarrollo metodológico se realizó una validez de documental sobre la literatura relacionada con los modelos de calidad de software desde sus inicios analizando la evolución técnica e histórica de los modelos planteados, hasta concluir con la utilización de normas técnicas internacionales que permiten asociar características medibles en un software, que imprimen intrínsecamente criterios de calidad de software, además la confiabilidad , por otro lao la confiabilidad se realiza por medio de medidas de estabilidad donde es mismo instrumento se realizó más de dos veces y se analizó la correlación para que sea altamente positiva.

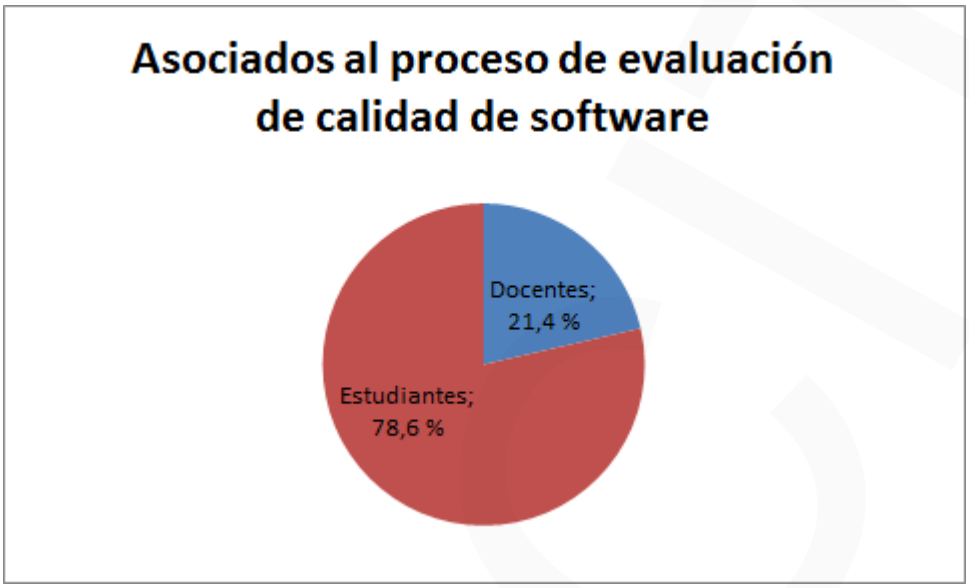
4. ANÁLISIS DE RESULTADOS

En este apartado se analizarán y examinaran los datos tomados para la creación de la metodología de evaluación de productos de software, se tomarán análisis estadísticos descriptivos.

4.1. Perfil de los encuestados

Las personas que se tuvieron en cuenta para realizar la encuesta hacen parte de la Universidad de Cundinamarca, se dividieron en dos grandes grupos son docentes y el grupo de desarrollo del programa de ingeniería de sistemas de la universidad de Cundinamarca extensión Facatativá , el otro grupo es los estudiantes que integran el grupo de desarrollo del programa de ingeniería de sistemas estudiantes de Ingeniería de sistemas, en total se realizaron 42 encuestas de las cuales 9 son docentes es decir el 21.4 % de la totalidad de encuestados, y 33 estudiantes es decir el 78.6 %.

Gráfica # 2 Perfil de Encuestado



4.2. Calidad de software en la Universidad de Cundinamarca

En este punto se analizan ciertas definiciones de calidad en la que cada asociado a la encuesta determinaba su punto de vista, así mismo a partir del estándar ISO/IEC/IEEE 29119 se establecieron el tipo de pruebas que se realizarán y cada asociado según criterio eligió un porcentaje para determinar la realización de cada prueba.

La calidad de software dentro del estándar ISO/IEC/IEEE 29119 está definida como la posibilidad de disminuir el riesgo de que un producto sea defectuoso en algún aspecto específico de su función.

En la Gráfica Gráfica # 4 se responde a la pregunta, ¿Para usted la calidad de software es? ,se plantea tres conceptos sobre calidad de software, los cuales son:

- El conjunto de propiedades que tiene un producto software generando ventaja competitiva y cumpliendo con las necesidades de los usuarios.
- El cumplimiento de parámetros de seguridad durante todo el ciclo de vida del software.
- Probabilidad de operación libre de fallas de un producto software en un entorno determinado y durante un tiempo específico.

Gráfica # 3 Percepción sobre el concepto de calidad de software

Gráfica # 4 Percepción concepto de Calidad de software

Cifra	Numera de	Frecuencia	Pregunta

porcentual	respuestas	relativa	
59.5%	25	0.59	El conjunto de propiedades que tiene un producto software generando ventaja competitiva y cumpliendo con las necesidades de los usuarios.
14.3%	6	0.14	El cumplimiento de parámetros de seguridad durante todo el ciclo de vida del software.
26.2%	11	0.26	Probabilidad de operación libre de fallas de un producto software en un entorno determinado y durante un tiempo específico

En este tipo de pregunta se manejan variables aleatorias simples, las cuales hacen referencia a las cualidades, por lo tanto, se caracterizan por el pensamiento de cada uno de los encuestados en relación con la calidad de software. Con anterioridad se planteó el concepto de calidad para la metodología que se desea implantar en la universidad de Cundinamarca

con el fin de ponerla a nivel con otros productores de software a nivel internacional.

Por tanto, con esta consulta se llega a la conclusión que los estudiantes y docentes tomados como muestra, para este estudio tienen conocimiento y claridad sobre la concepción de calidad de software.

4.3. Procesos para de evaluar un producto software

En el desarrollo de un producto informático, se debe conocer el proceso de como se debería evaluar dicho producto, por lo tanto es necesario conocer la apreciación de sobre la realización del proceso; para esto se realiza la siguiente pregunta: ¿Sabe usted qué procesos se tienen en cuenta en el momento de evaluar un producto software en la Universidad de Cundinamarca?, en la Gráfica # 5 se visualiza los resultados.

Gráfica # 5 Conoce el proceso de evaluación de un producto de software.

Considerando que todos los encuestados son parte de la Universidad Cundinamarca y que se encuentran directamente relacionados ya que son docentes o estudiantes de semestres avanzados el porcentaje de personas

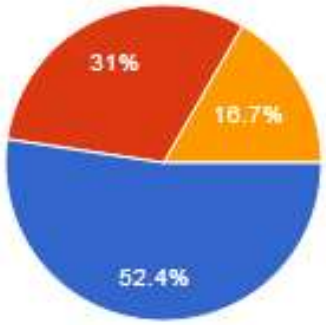
que no conocen el proceso de evaluación de la calidad de un software el muy alto 71,4%. A razón de esto se podrían tener distintas observaciones:

1. Que se tiene en cuenta dicha información únicamente cuando se va a presentar proyectos de desarrollo específicos.
2. Los docentes que no se encuentran vinculados al proceso de evaluación no se informan sobre el tema.
3. A los estudiantes no se les inculca desde el principio la necesidad de desarrollar software competente y de calidad.

4.4. Impacto del modelado de un sistema informático frente a la calidad de software

Dentro del modelo de evaluación de software, el modelado de un software es un criterio a tener en cuenta, por lo tanto es necesario conocer la importancia del modelado y sobre la revisión bibliográfica que se enfoca sobre UML cuales modelos son importantes al momento de iniciar el diseño de una solución informática; para considerar este elemento se plantea la siguiente pregunta. En el momento de evaluar la calidad de software. ¿Qué prioridad cree que tiene el modelado?

Gráfica # 6 Prioridad del Modelado en la Evaluación de Calidad de software

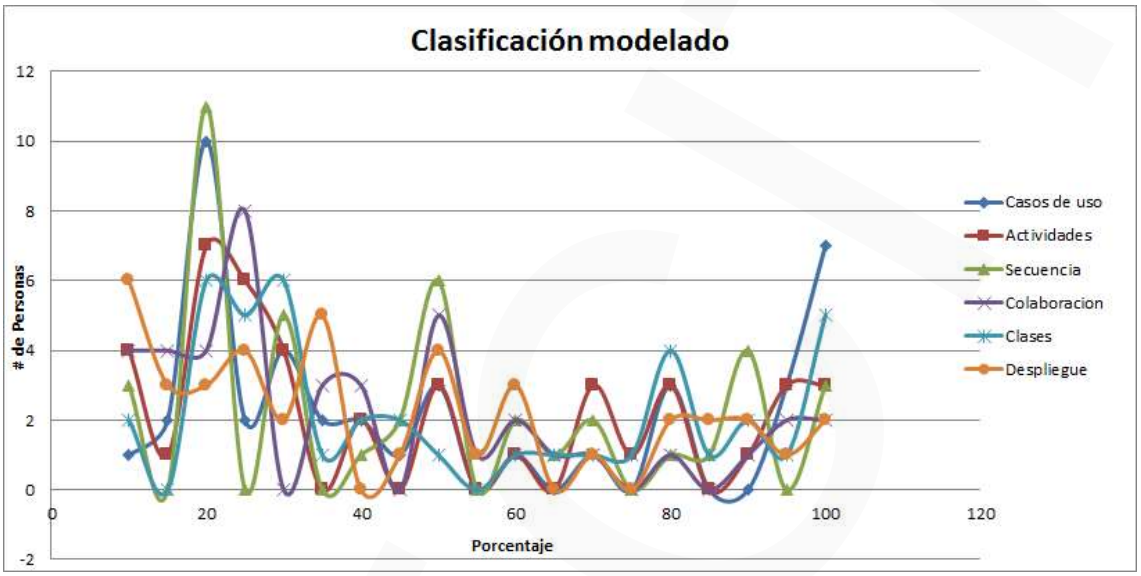


Alta	22	52.4%
Media	13	31%
Baja	7	16.7%

El modelado juega un papel importante durante el ciclo de vida de un software y esto se puede ratificar con las respuestas de los encuestados la mayor parte de ellos concuerda que se le debe dar una prioridad alta.

Otro criterio a analizar según el estándar de modelado UML existen diversos tipos de diagramas, donde se busca encontrar la importancia de cada esquema del estándar UML.

Gráfica # 7 Clasificación de modelos para UML



De acuerdo con los resultados obtenidos para cada diagrama tenido en cuenta en el modelado, es necesario determinar la media, mediana y la moda de la distribución de dichos diagramas para realizar posteriormente la asignación adecuada y determinar el porcentaje que se le asignará a cada diagrama, se obtuvieron los siguientes resultados:

Tabla 5 Estadística Descriptiva modelos UML

Diagramas	Media	Mediana	Moda
Casos de uso	48,21	42,5	20
Actividades	54,28	45	20

Secuencia	57,3	60	20
Colaboración	54	55	25
Clases	57,35	60	20/30=25
Despliegue	54,06	52,5	10

Los diagramas UML se clasifican en tres tipos que son:

- Los diagramas de comportamiento permiten exhibir las reacciones de un sistema.
- los diagramas de interacción son un conjunto de diagramas de comportamiento que permiten enfatizar las interacciones entre objetos.
- los diagramas de estructura los cuales muestran elementos que actúan de forma independiente al tiempo.

Respecto a esto se desea conocer:

A partir de la prioridad del modelado. ¿Cuál cree usted que es la relación principal para el desarrollo de estos diagramas?

Gráfica # 8 Prioridad diagramas de Comportamiento

Gráfica # 9Prioridad diagramas de Interración

Gráfica # 10 Prioridad diagrama de Estructura

4.5. Pruebas dedicadas a la evaluación de la calidad de software

En el estándar ISO/IEC/IEEE/29119 propone una cierta cantidad de pruebas de las cuales se han seleccionado las de mayor importancia.

Gráfica # 11 Importancia de Pruebas de Software

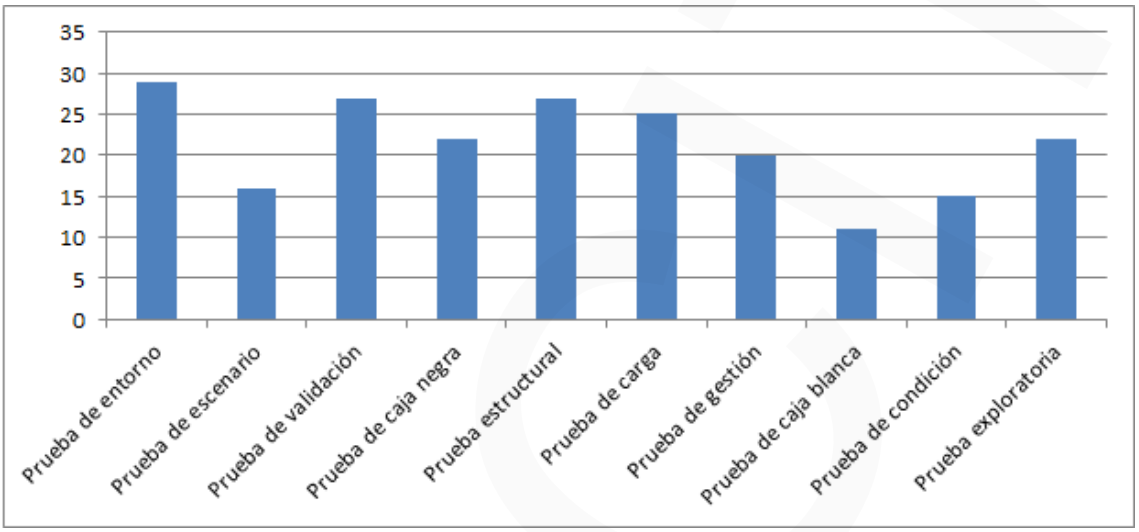


Tabla 6 importancia de las pruebas de Software

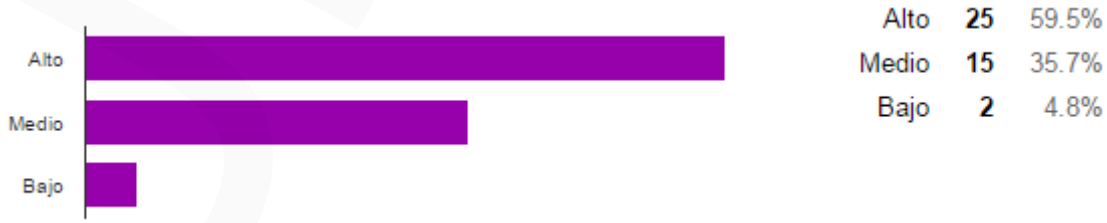
Tipo de Prueba	**Peso relativo**	**Porcentaje de importancia**
Prueba de entorno	**29**	69%
Prueba de escenario	**16**	38.1%
Prueba de validación	**27**	64.3%
Prueba de caja negra	**22**	52.4%
Prueba estructural	**27**	64.3%
Prueba de carga	**25**	59.5%
Prueba de gestión	**20**	47.6%

Prueba de caja blanca	**11**	26.2%
Prueba de condición	**15**	35.7%
Prueba exploratoria	**22**	52.4%

Considerando la norma ISO/IEC/IEEE 29119-2 y la clasificación de las pruebas para evaluar la calidad de software. Existen 3 capas de cobertura las cuales son: gestión de pruebas, pruebas funcionales y pruebas de carga y rendimiento. De acuerdo con estos criterios se analiza el nivel de prioridad en cada capa.

Gráfica # 12 prioridad de la gestión de pruebas

Gráfica # 13 prioridad de las pruebas funcionales

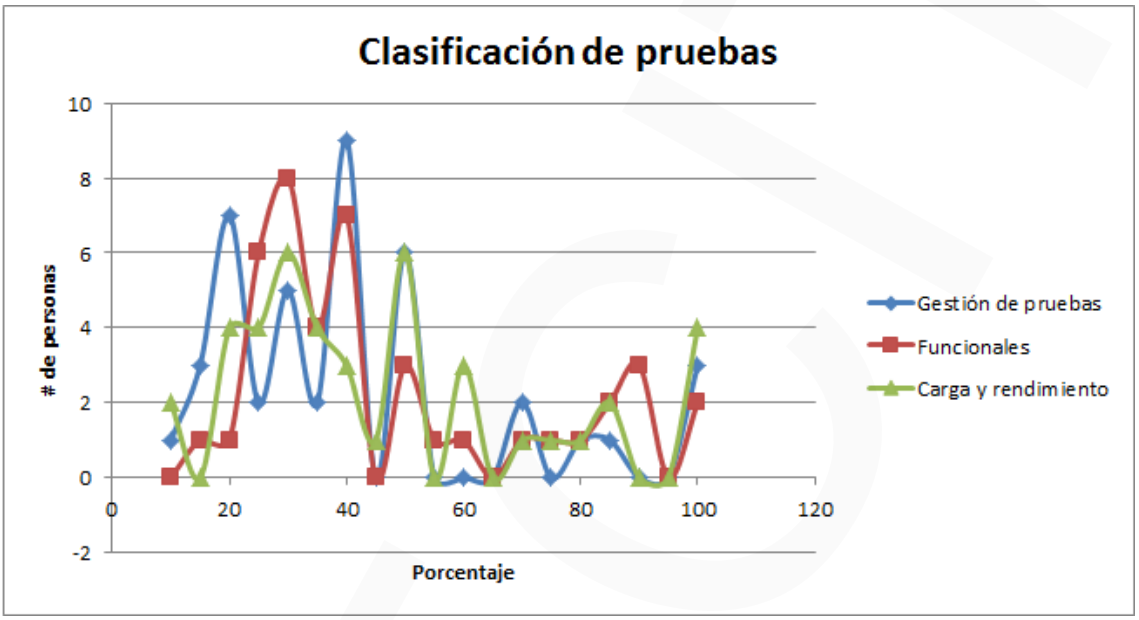


Gráfica # 14 prioridad de las pruebas de carga y estrés

En las variables aleatorias que se dieron para representar la prioridad de cada uno de los módulos de evaluación de software, a todos se les dio una importancia alta, por lo tanto, se deduce que lo ideal para nuestros encuestados es satisfacer las necesidades del dueño del producto informático.

En la Gráfica # 15 se define el porcentaje de prioridad de cada una de las capas de las pruebas .

Gráfica # 16clasificación de las capas de pruebas



En la Tabla 7 se realiza el análisis descriptivo de la clasificación de las pruebas

Tabla 7 análisis estadístico de las pruebas

Tipo de prueba	Media	Mediana	Moda
Gestión de pruebas	**46,66**	**37,5**	**40**
Pruebas funcionales	**55,33**	**55**	**30**
Pruebas de carga y rendimiento	**51,78**	**47,5**	**30/50=40**

4.6. Estudio estadístico para las encuestas

Para normalizar los datos obtenidos, se analizaron las medias de tendencia central en un estudio estadístico descriptivo , asegurando intervalos de confianza del 90% , como primera medida se normalizaron los valores de los diagramas del modelado UML ,los medidas de tendencia central usadas son: la media, mediana, moda, desviación estándar, el tamaño de la muestra para posteriormente hallar el intervalo de confianza, estos valores se encuentran especificados en la Tabla 8

Tabla 8 estudio estadístico para el modelado enfocado a los diagramas

diagrama	Media	mediana	Moda	Desviación	muestra	Z1=0,05 /90 %	Raíz	Interv. 1 punto	Interv. 2 punto
casos de uso	16.07	42.5	20	28.01	14	1.76	3.74	29.2544	2.888
Actividades	18.1	55	20	87.1	14	1.76	3.74	59.0979	-22.9
secuencia	17.74	60	20	26.72	13	1.77	3.60	30.8614	4.615
colaboración	19.29	55	25	28.53	15	1.75	3.87	32.2001	6.371
clases	23.21	60	25	27.18	17	1.74	4.12	34.6806	11.75
despliegue	20.6	52.5	10	29.06	16	1.75	4	33.2788	7.912

Para determinar el nivel de impacto de cada dimensión del modelado se analiza de la Tabla 9

Tabla 9 prioridad de las dimensiones del modelado

Prioridad	Comportamiento	Interacción	Estructura
Alta	**57.1**	**45.2**	**47.6**
Media	**28.6**	**50**	**26.2**
Baja	**14.3**	**4.8**	**26.2**

Para la normalización de estos valores se utilizó un criterio de ajuste, sobre promedios, lo que dio como resultado, las prioridad de cada dimensión de del modelado UML son :

Tabla 10 porcentaje final para la clasificación de diagramas

Dimensión	Porcentaje
Comportamiento	**28,50%**
Interacción	**10,50%**
Estructura	**11,00%**
Total	**50%**

Se realizó el mismo proceso para dar valor a cada uno de los diagramas que componen al diagrama de comportamiento, obteniendo los siguientes resultados:

Diagramas de Comportamiento → 28,50%

Diagrama de actividades → 13,44 %

Diagrama de casos de uso → 15,06%

Diagramas de interacción 10,50%

Diagrama de clases →5,46%

Diagrama de despliegue → 5,04%

Diagrama de Estructura 11%

Diagrama de secuencia → 5,50%

Diagrama de colaboración → 5,50%

Por otro lado, para normalizar los valores sobre el tipo de pruebas que se deben hacer a un producto de software se llevó a cabo el mismo proceso para determinar el porcentaje de cada uno de los tres tipos de pruebas existentes considerando nuevamente que no se debe sobrepasar el 100%

total, fue necesario nuevamente tener en cuenta la media la mediana, la moda, el tamaño de la muestra, la desviación estándar y de esta manera determinar el intervalo de confianza, de la siguiente manera:

Tabla 11 estudio estadístico de pruebas

Tipo de prueba	**Media**	**mediana**	**Moda**	**Desviación**	**muestra**	**Z1=0,05 /90 %**	**Raíz**	**Interv. 1 punto**	**Interv. 2 punto**
Gestión de pruebas	13.33	42.5	20	28.01	14	1.76	3.74	26.516	0.15
Pruebas funcionales	19.76	55	30	87.1	14	1.76	3.74	60.764	-21.2
Pruebas de carga y rendimiento	17.26	60	0	26.72	13	1.77	3.60	30.385	4.139

Para determinar el nivel de impacto de cada fase de pruebas se normalizaron los datos obtenidos y se definió sus porcentajes, como muestra la siguiente tabla:

Tabla 12 porcentajes fases de pruebas

Fase de Pruebas	**Porcentaje**
Gestión	**10,50%**
Funcional	**30%**
Carga	**9,50%**
	50%

Finalmente para determinar la evaluación total del producto software y sus porcentajes de una manera adecuada, el modelado equivale al 50 % de la evaluación total del producto y la evaluación de los 3 tipos de pruebas equivale al otro 50% del 100% total de la evaluación de calidad que se llevara a cabo, en la Tabla 13 se plantea el modelo # 1.

Tabla 13 Modelo #1 de Medición para Calidad de Software

ITEMS A EVALUAR		**PORCENTAJES**
MODELADO		
Clasificación	**Diagramas**	
comportamiento	Actividades	13.44%
	Casos de uso	15.06%
interacción	Clases	5.46%
	Despliegue	5.04%
Estructura	Secuencia	5.50%
	Colaboración	5.50%

PRUEBAS SOBRE EL SISTEMA		
Clasificación	**Herramienta**	
gestión de pruebas	QABook	10.50%
pruebas funcionales	JMeter	30%
pruebas de carga y estrés	LoadUI	9.50%
TOTAL		**100%**

A partir de una presentación realizada ante el comité de evaluación surgen ciertas recomendaciones respecto al manejo de porcentajes, a pesar de que la distribución de porcentajes se realizó a partir de las respuestas obtenidas a través de la aplicación de la encuesta a docentes y estudiantes de ingeniería de sistemas, se sugiere que la evaluación de modelado y la evaluación en cuanto a ejecución de pruebas no correspondan al mismo valor porcentual (50%) es decir se llega a la conclusión que la evaluación de modelado al tratarse de una evaluación de tipo subjetivo debe tener un menor valor respecto al valor total de la evaluación de calidad. A partir de estas evidencias se plantean dos modelos de medición diferentes, con la posibilidad de reajustar estos valores para lo cual sería necesario plantear una nueva encuesta para asignar los valores correspondientes según las

nuevas respuestas obtenidas. La primera opción que se plantea se le asigna un valor de 40% al modelado y 60% a la evaluación de pruebas (Modelo # 2). En la segunda opción se le asigna un valor de 30% al modelado y 70% a la evaluación de pruebas (Modelo # 3). Las posibilidades de reajuste de porcentajes se presentan a continuación:

Tabla 14 Modelo #2 de Medición para Calidad de Software

ITEMS A EVALUAR		**PORCENTAJES**
MODELADO		
Clasificación	**Diagramas**	
comportamiento	Actividades	11%
	Casos de uso	12%
interacción	Clases	4.58%
	Despliegue	4.42%
Estructura	Secuencia	4%
	Colaboración	4%
PRUEBAS SOBRE EL SISTEMA		

Clasificación	Herramienta	
gestión de pruebas	QABook	13,52%
pruebas funcionales	JMeter	34%
pruebas de carga y estrés	LoadUI	12,48%
TOTAL		**100%**

Tabla 15 Modelo #3 de Medición para Calidad de Software

ITEMS A EVALUAR		PORCENTAJES
MODELADO		
Clasificación	**Diagramas**	
comportamiento	Actividades	7.40%
	Casos de uso	9.60%
interacción	Clases	3.66%

	Despliegue	3.34%
Estructura	Secuencia	3%
	Colaboración	3%
PRUEBAS SOBRE EL SISTEMA		
Clasificación	**Herramienta**	
gestión de pruebas	QABook	18%
pruebas funcionales	JMeter	36%
pruebas de carga y estrés	LoadUI	16%
TOTAL		**100%**

5. CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones

Para el proceso de Calidad de software que se debe tener en cualquier ciclo de vida de un producto informático surge la idea de plantear la metodología encargada de evaluar la calidad de productos software desarrollados por de la Universidad de Cundinamarca. Como se había mencionado anteriormente no existe una metodología que consolide todos los aspectos a contemplar en el desarrollo de software la cual permita un procedimiento adecuado en cuanto al manejo y consolidación de calidad de software, en muchas ocasiones se debe a implicaciones comerciales o simplemente por desconocimiento del tema.

El análisis previo de los procesos permitió definir las posibles dificultades presentes en el proceso de evaluación de software debido a que el aspecto de calidad es desconocido para algunos evaluadores, además la necesidad de realizar aplicaciones en poco tiempo con características no solamente

funcionales sino también hacer énfasis en la calidad del producto software mediante el uso de estándares internacionales los cuales ayudan a regular estos aspectos y así plantear los entornos adecuados para su evaluación. En este caso es necesario considerar que la calidad de un producto software no sólo hace referencia a la ejecución de la aplicación sino que se ve reflejada durante el ciclo de vida del software, tomando como base la norma ISO/IEC/IEEE 29119 fue posible proponer y desarrollar la metodología para evaluar la calidad de productos software de la Universidad de Cundinamarca.

Mediante el análisis estadístico del proyecto es posible afirmar que tanto docentes como estudiantes desconocen el procedimiento que se efectúa para la evaluación de productos software, también se pudo determinar el valor correspondiente a cada tipo de diagrama y cada tipo de prueba que se realiza a la aplicación web a evaluar; al analizar los resultados obtenidos se determinó 50 % para evaluación de modelado y 50 % para evaluación de pruebas de software.

Para llevar a cabo el proceso de evaluación de modelado se tiene en cuenta que cumpla con los requerimientos mínimos de acuerdo con el framework UML 2.0 y de acuerdo a los criterios planteados en la presente metodología; aunque no es posible garantizar esta evaluación mediante el uso de una herramienta que automatice este proceso es posible utilizar esta

metodología como guía para complementar la evaluación de modelado de una aplicación web.

Por otra parte, el proceso de evaluación de los tres tipos de pruebas resultó un poco más sencillo debido a que se contó con herramientas libres las cuales permitieron sistematizar este proceso que realizado manualmente resultaba un poco tedioso y lento. Los resultados de los informes de cada una de las tres herramientas utilizadas para la gestión de pruebas QABook, para las pruebas funcionales JMeter y para las pruebas de carga y rendimiento LoadUI presentaron una serie de errores derivados de cada una de las evaluaciones.

Como seguimiento de esta actividad la metodología para evaluar la calidad de productos software de la Universidad de Cundinamarca fue aplicada al proyecto “Modulo web para la gestión de usuarios del sistema de información de autoevaluación de la Universidad de Cundinamarca” (módulo administrador), en esta aplicación se tuvieron en cuenta los requerimientos establecidos previamente por el desarrollador especificando el tipo de requerimiento, su nivel de prioridad y si se cumplía o no con el desarrollo de cada requerimiento, en este caso se contó con 54 requerimientos de tipo funcional de los cuales 8 resultaron defectuosos en el momento de ejecutar los 30 casos de prueba que fueron tenidos en cuenta para la gestión de pruebas.

En las pruebas funcionales fue posible determinar tres variaciones distintas de entorno de evaluación lo cual quiere decir diferentes usuarios en un tiempo determinado por el evaluador y de esta manera obtener el tiempo de respuesta de la aplicación, al realizar los promedios de fallos obtenidos a través de la herramienta JMeter se concluye que este módulo tiene una vulnerabilidad del 0.4 % haciendo referencia a 11 advertencias respecto al proceso de ejecución de la prueba funcional.

En la prueba de carga y rendimiento se tuvo en cuenta el nivel de respuesta de la aplicación de acuerdo con la máxima cantidad de usuarios que pueden ser atendidos por segundo.

En cuanto a la evaluación de modelado se determinó si los diagramas cumplían con los elementos mínimos, en este parte el proyecto “Modulo web para la gestión de usuarios del sistema de información de autoevaluación de la Universidad de Cundinamarca” cuenta solamente con tres de los seis diagramas que se tienen en cuenta para realizar la evaluación, es decir este proyecto no cuenta con diagramas de interacción como son diagrama de secuencia y diagrama de colaboración, y uno de los diagramas de comportamiento que es el diagrama de actividad, como se había mencionado no se cuenta con una herramienta que realice este proceso por lo tanto esta parte del proceso va más relacionada con el criterio del evaluador.

A partir de estas evidencias es posible manifestar que la metodología para evaluar la calidad de software de la Universidad de Cundinamarca cumple con los parámetros establecidos en la norma ISO/IEC/IEEE 29119 partes 1-2-3, permitiendo valorar en una escala de 1 a 100% el nivel de calidad manejado por el producto software que se acoge a esta evaluación, determinando también que el uso de la metodología facilita de cierta manera el proceso de testeo de software que viene realizando al interior del equipo de desarrollo y a su vez agiliza este proceso a través del uso de herramientas libres.

De acuerdo a la respuestas obtenidas del proceso de evaluación realizado al proyecto “Modulo web para la gestión de usuarios del sistema de información de autoevaluación de la Universidad de Cundinamarca” (módulo administrador), ha sido posible observar que la evaluación de calidad se encuentra dividida entre las pruebas de ejecución y el modelado de la aplicación cada una tiene asignado un 50% del total de la valoración de calidad, estos valores fueron especificados a partir de las respuestas obtenidas de las encuestas aplicadas.

En la metodología de calidad de software esta se divide en dos elementos , el primero en la generación y evaluación del modelado, el modelo al ser el diseño técnico previo a la generación de cualquier solución informática, por lo tanto para hacer el seguimiento y evaluación del modelado , en este

proyecto se tomó como guía el modelo UML 2.0 , para un segundo momento la metodología plantea la evaluación de las pruebas que se deben aplicar a un software, las pruebas según estándares internacionales se deben enfocar en pruebas funcionales, carga y rendimiento, para la evaluación de estos parámetros se utiliza la ayuda de software especializado con JMeter y LoadUI, con la utilización y evaluación de estos criterios, la metodología permite la evaluación de productos de software web en la Universidad de Cundinamarca.

.

5.2. Recomendaciones

Para un trabajo futuro se recomienda ajustar los porcentajes del modelado y las pruebas en general, para que se le dé mayor prioridad a la ejecución de pruebas: Gestión de pruebas, pruebas funcionales y pruebas de carga y rendimiento. Ya que como se determinó la evaluación de modelado a pesar de tener una ponderación del 50 % se trata de una evaluación más subjetiva debido a que el resultado de esta parte de la evaluación es obtenida por criterio del evaluador a pesar de estar establecidos ciertos parámetros para efectuarla tratándose de una evaluación realizada de forma manual.

Se pretender realizar una nueva versión de la metodología para la evaluación y análisis de productos software de la Universidad de

Cundinamarca, se trata del diseño de una plataforma que permita integrar las tres herramientas libres que se vienen utilizando para realizar la gestión de pruebas, las pruebas funcionales y las pruebas de carga y rendimiento teniendo en cuenta los escenarios de prueba que se manejarán y las diferentes fases de ejecución.

Para complementar esta labor se propone el desarrollo de un modelo matemático que respalde la fidelidad y exactitud del proyecto a evaluar en un tiempo determinado y según los criterios de evaluación que se han planteado en la presente metodología, logrando mitigar de cierta manera los errores que se presentan como resultado del testeo de software.

Otra consideración importante que se puede tener en cuenta para un trabajo futuro de la metodología de evaluación de calidad de software en la Universidad de Cundinamarca hace referencia a la seguridad informática con la que debe cumplir cualquier producto software y que respalde el manejo adecuado de la información siguiendo las normalizaciones existentes para esta investigación, además de expandir esta evaluación a aplicaciones móviles y así lograr mayor auge en el desarrollo de productos software en cualquier tipo de plataforma.

6. PROPUESTA DE SOLUCIÓN AL PROBLEMA

7.1. Denominación de la propuesta

Metodología para la evaluación de la calidad de software de productos de software para la Universidad de Cundinamarca.

7.2. Descripción

Se requiere definir una método que defina y guie el proceso de la evaluación de software, lo que permite es la generación parámetros de calidad de software, la metodología se centra en la generación de una rúbrica de evaluación para los elementos que se evaluaran y medirán.

7.3. Fundamentación

La calidad de software nos permite asegurar que una solución informática soluciona las necesidad el usuario final o cliente, por lo tanto tener criterios de calidad permite la reproducción de elementos que a lo largo del ciclo de vida de un software se generan indicadores orientadores sobre el cumplimiento de la misión del software, la metodología nos permite evaluar que los requerimientos de un software se cumple y satisfacen las necesidades del cliente.

7.4. Objetivos de la propuesta

7.4.1 Objetivo general

Desarrollar de una metodología para la evaluación e productos de software para la Universidad de Cundinamarca.

7.4.2 Objetivos específicos

- Generar el proceso para la medición de productos de software.
- Desarrollar las herramientas para la gestión pruebas, pruebas funcionales y pruebas de carga y estrés basados en la norma ISO/IEC/IEEE 29119
- Realizar un plan de pruebas para la evaluación de los proyectos
- Generar herramientas para analizar resultados de la metodología frente a los requerimientos de las soluciones informáticas desarrolladas

7.5. Metas

Con este proyecto se busca desarrollar una herramienta basada en la norma ISO/IEC/IEEE 29119, para evaluar la calidad de las aplicaciones web creadas al interior de la Universidad de Cundinamarca.

El grupo de desarrollo contara con una herramienta completa, eficaz, y de fácil manejo para mejorar y llevar de una forma organizada la valoración de

los distintos proyectos de desarrollo de aplicaciones web. Esta herramienta facilitará la evaluación de los parámetros de calidad de los productos software de la Universidad de Cundinamarca.

La herramienta estará en capacidad de realizar pruebas de calidad basadas en la norma ISO/IEC/IEEE 29119, llevando a cabo una ponderación por cada criterio de evaluación de manera dinámica, para obtener un resultado acorde a las condiciones de evaluación previamente establecidas por el administrador.

7.6. Beneficiarios

Esta propuesta involucra al equipo de desarrollo de la Universidad de Cundinamarca extensión Facatativá y a los usuarios finales de las soluciones informáticas desarrolladas por la Universidad.

7.7. Productos

Con la utilización de esta propuesta se generarán:

- Lineamientos para la evaluación de producto de software.
- Una herramienta informática que permite el control seguimiento de las evaluaciones de los productos de software.
- Un sistema informático que permite la evaluación del modelado en el framework UML 2.0

- Un sistema informático que permite la evaluación y creación de pruebas funcionales, gestión de pruebas y pruebas de rendimiento

7.8. Localización

Esta propuesta se desarrollara en la ciudad de Facatativá Colombia en la Universidad de Cundinamarca extensión Facatativá

7.9. Metodología

Considerando las necesidades encontradas acerca de la evaluación de productos software desarrollados por el equipo de desarrollo de la Universidad de Cundinamarca y falta de estándares que sustenten y acompañen el proceso de creación del mismo desde el diseño hasta el desarrollo y ejecución de la aplicación. Al realizar esta propuesta es posible dar un soporte a la valoración de los proyectos implementando los estándares de calidad de software basadas en la norma ISO/IEC/IEEE 29119

Para el desarrollo de la herramienta se aplicara el modelo de desarrollo MVC ya que permite manejar de una manera más organizada todo el proyecto y a su vez permitirá un desarrollo ágil de la aplicación, en la elaboración de esta aplicación se implementara la metodología Scrum, la cual permite avances en bloques temporales cortos, de esta manera habrá mayor compromiso, mejores resultados al trabajar en equipo.

En la siguiente etapa se realizará el modelado y documentación del aplicativo, luego se construirá la página principal, de registro e ingreso de usuarios y paginas principales por cada rol.

Posteriormente se realizará el módulo de modelado y documentación, el módulo de pruebas (gestión de pruebas, carga y estrés, funcionales) y el modulo administrativo para la definición de porcentajes de evaluación, de acuerdo con esto se realizaran informes y se mostraran los resultados obtenidos de la evaluación, por último se realizaran escenarios de pruebas.

Por cada etapa se debe realizar una retroalimentación en la cual se pueda corregir y mejorar las posibles fallas que se vayan presentando en el desarrollo del aplicativo y de esta manera facilitar el análisis y la consecución de los objetivos previamente establecidos.

7.10. Cronograma

Número	Actividad	Tiempo
1	Aprobación del proyecto por parte de la Universidad de Cundinamarca..	5 semanas
2	Recolección de información concerniente a los antecedentes del proyecto.	6 semanas
3	Realizar el levantamiento de requerimientos.	4 semanas

4	Diseño del modelado de la plataforma.	3 semanas
5	Entrega y realimentación de modelado	1 semana
6	Desarrollo de módulos de la herramienta.	30 semanas
7	Entrega y retroalimentación de módulos.	4 semanas
8	Realización de escenarios de pruebas de cada uno de los módulos por los cuales se integran la aplicación.	3 semanas
9	Realización del informe	4 semanas
10	Pruebas y resultados	3 semana
11	Entrega final del aplicativo.	1 semana
12	Sustentación final del proyecto.	1 semana

7.11. Recursos

Nombre	**Función en el proyecto**	**Tipo de vinculación**	**Dedicación Horas/ Semana**
Cesar Barahona	Investigador principal	Docente	ocho horas
Andres Monroy	Estudiante pregrado investigador auxiliar	Estudiante	ocho horas

John Acosta	Estudiante pregrado investigador auxiliar	Estudiante	ocho horas
Yeison Ruiz	Estudiante pregrado investigador auxiliar	Estudiante	ocho horas

7.12. Presupuesto

	Solicitado en efectivo a UDEC	**Contrapartida en especie**		
Rubros		**UDEC**	**Otras Entidades**	**Total**
PERSONAL	$59’000.000			$59’000.000
EQUIPOS	$6’000.000			$6’000.000
MATERIALES E INSUMOS	$500.000			$500.000
SERVICIOS TECNOLOGICOS	$960.000			$960.000
VIAJES	$600.000			$600.000
OTROS	$500.000			$500.000
TOTALES	**$ 67’560.000**			**$ 67’560.000**

7.13. Aplicación de la metodología propuesta

Para realizar la metodología de evaluación de software inicialmente se analizará el modelado como elemento primordial para la evaluación, en la siguiente tabla indica los elementos que deben incluir los diagramas en el modelado.

Tabla 16 Evaluación del modelado

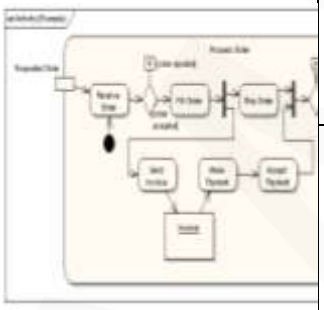
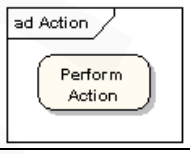
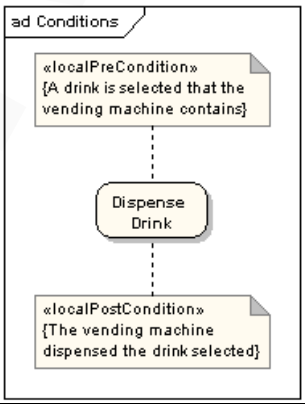
DIAGRAMA	COMPONENENTES	OPCIONAL	USO DE COMPONENTES
DIAGRAMA DE ACTIVIDADES  Los diagramas de actividades muestran el flujo de trabajo desde el punto de inicio hasta el punto final detallando muchas de las rutas de decisiones que existen en el progreso de eventos contenidos en la actividad	Actividad 	NO	es la especificación de una secuencia parametrizada de comportamiento
	Acción 	NO	Una acción representa un solo paso dentro de una actividad
	Restricciones de acción 	SI	Son adjuntas a una acción se pueden presentar antes de realizar una acción o posteriormente.

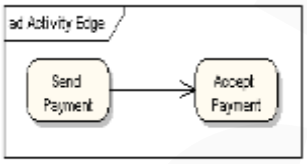
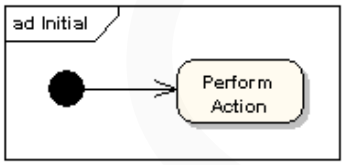
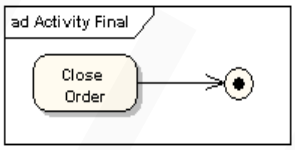
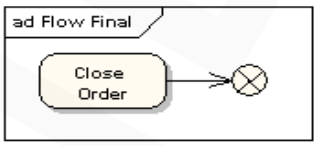
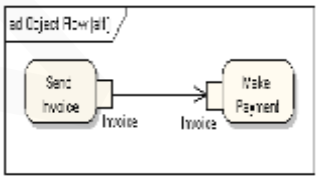
DIAGRAMA	COMPONENENTES	OPCIONAL	USO DE COMPONENTES
	Flujo de control 	NO	Muestra el flujo de control de una acción sobre otra.
	Nodo inicial 	NO	Describe el comienzo de cualquier actividad.
		Nodo final	
	Nodo final de actividad 	NO	El nodo final de actividad se describe como un círculo con un punto dentro del mismo.
	Nodo final de flujo 	NO	El nodo final de flujo se describe como un círculo con una cruz dentro del mismo.
	Flujo de objetos 	SI	Un flujo de objeto es la ruta a lo largo de la cual pueden pasar objetos o datos
	Nodos de decisión y combinación	SI	Los flujos de control que provienen de un nodo de decisión tendrán condiciones de guarda que permitirán el control para

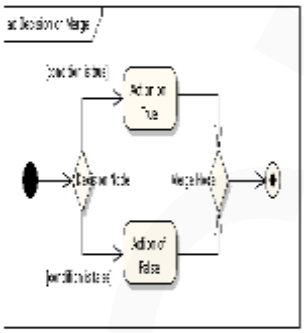
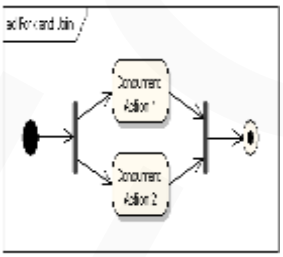
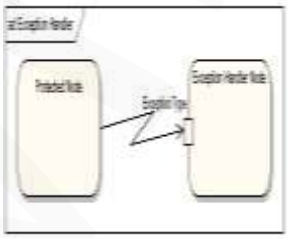
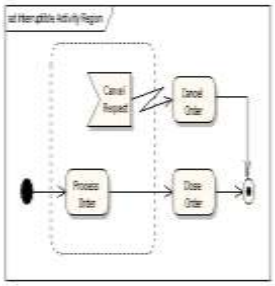
DIAGRAMA	COMPONENENTES	OPCIONAL	USO DE COMPONENTES
			fluir si la condición de guarda se realiza
	Nodos de bifurcación y unión 	SI	Estos indican el comienzo y final de hilos actuales de control
	Región de expansión	SI	Es una región de actividad estructurada que se ejecuta muchas veces.
	Gestores de excepción 	SI	Son de uso exclusivo cuando para la realización de una actividad se tengan restricciones u observaciones.
	Región de actividad interrumpible 	SI	Rodea un grupo de acciones que se pueden interrumpir.

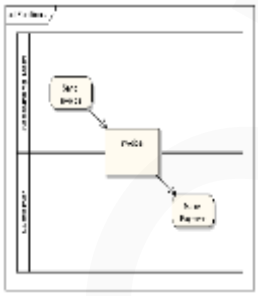
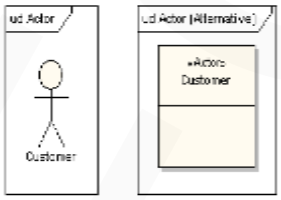
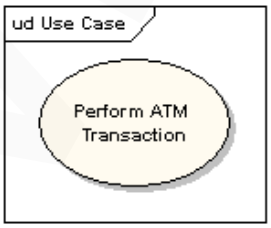
DIAGRAMA	COMPONENENTES	OPCIONAL	USO DE COMPONENTES
	Partición 	SI	las particiones se usan para separar acciones dentro de una actividad
DIAGRAMA DE CASOS DE USO  Son un medio de comunicación con los usuarios y otros interesados acerca de lo que se piensa hacer del sistema.	Actores 	NO	Los actores representan los roles que pueden incluir usuarios humanos, un hardware externo u otros sistemas. (Los actores pueden generalizar otros actores.)
	Casos de uso 	NO	- Notación para usar un caso de uso es una línea de conexión con una punta de flecha opcional mostrando la dirección del control. - Pueden contener la funcionalidad de otro caso de uso. - Se pueden incluir por uno o más casos de uso. - Se puede usar para extender el comportamiento de otro.

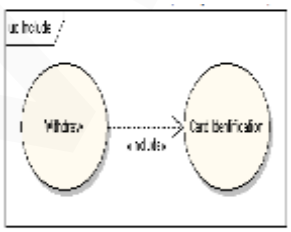
DIAGRAMA	COMPONENENTES	OPCIONAL	USO DE COMPONENTES
			- El punto al cual un caso de uso extendido se agrega se puede definir por medio de un punto de extensión. - Un Caso de Uso normalmente incluye: • Nombre y Descripción • Requisitos • Restricciones • Escenarios • Diagramas de escenarios • Información adicional.
	Inclusión de casos de uso 	SI	Los casos de uso pueden contener la funcionalidad de otro caso de uso como parte de su proceso normal. Los casos de uso se pueden incluir por uno o más casos de uso, ayudando a reducir el nivel de duplicación de la funcionalidad.

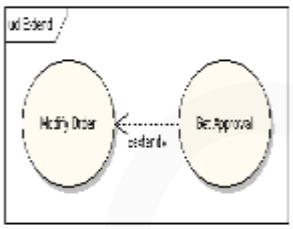
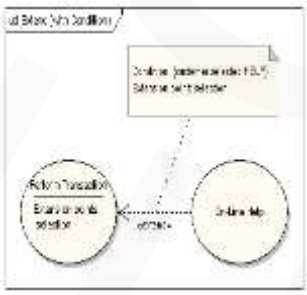
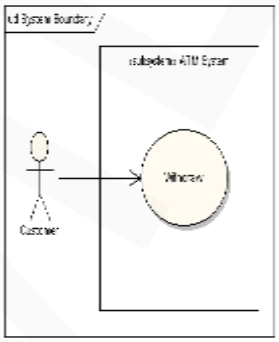
DIAGRAMA	COMPONENENTES	OPCIONAL	USO DE COMPONENTES
	Casos de uso extendidos 	SI	Se pude usar para extender el comportamiento de otro. (En caso de que un usuario necesite permiso de otro para el acceso a ese caso de uso).
	Puntos de extensión 	SI	Se utiliza un punto de extensión en caso de que se cumplan algunas condiciones específicas para ejecutar el caso de uso relacionado.
	Límite del sistema 	SI	Usualmente se usa para mostrar casos de uso dentro del sistema y actores fuera del sistema.

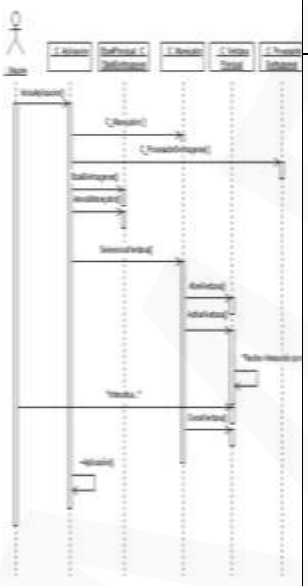
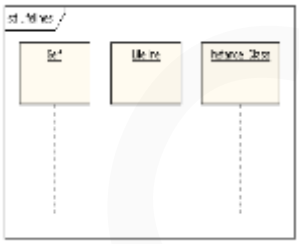
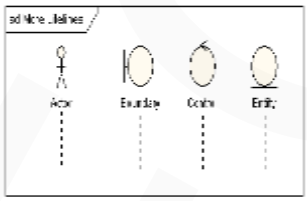
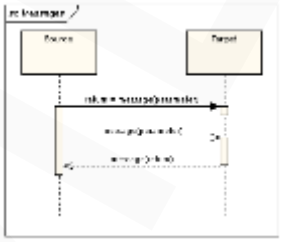
DIAGRAMA	COMPONENENTES	OPCIONAL	USO DE COMPONENTES
Diagrama de secuencia 	Línea de vida  	NO	Una línea de vida representa un participante individual en un diagrama de secuencia. Se puede presentar un actor, una clase, una entidad, o otra línea de vida.
	Mensajes 	SI	Deben ser mostrados como flechas. - Síncronos: denotados por una flecha con punta oscura - Asíncronos: denotados por una flecha con un punta en línea. - Llamadas o señales: denotado por una flecha punteada
	Ocurrencia de ejecución	SI	Denota la ocurrencia de ejecución o activación de un foco de control.
	Mensajes self	SI	Puede representar una llamada recursiva de una operación,

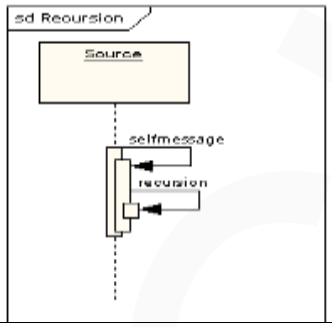
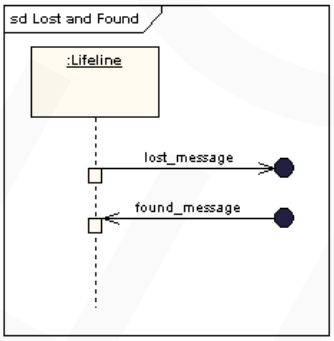
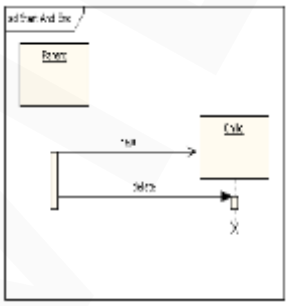
DIAGRAMA	COMPONENENTES	OPCIONAL	USO DE COMPONENTES
			o un método llamando a otro método perteneciente al mismo objeto.
	Mensajes perdidos y encontrados 	SI	Los mensajes perdidos son aquellos que han sido enviados pero que no han llegado al destino esperado, o que han llegado aún.
	Inicio y final de línea de vida 	NO	Una línea de vida se puede crear o destruir durante la escala de tiempo representada por un diagrama de secuencia.
	Restricciones de tiempo y duración	SI	Al configurar una restricción de duración para un mensaje, el mensaje se mostrará como una línea inclinada.

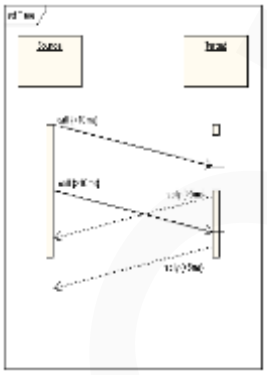
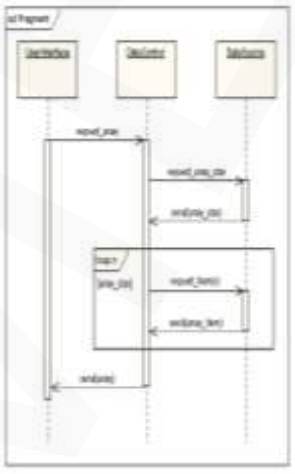
DIAGRAMA	COMPONENENTES	OPCIONAL	USO DE COMPONENTES
			
	Fragmentos combinados 	SI	Fragmento combinado es una o más secuencias de procesos incluidas en un marco y ejecutadas bajo circunstancias nombradas específicas.
	Puerto	SI	Conexión entre un fragmento y un mensaje.

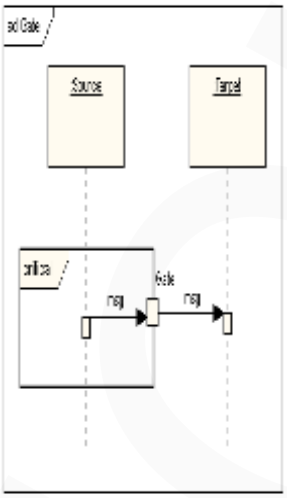
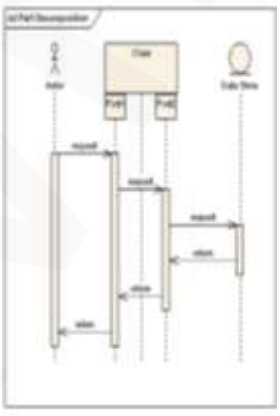
DIAGRAMA	COMPONENENTES	OPCIONAL	USO DE COMPONENTES
			
	Descomposición en parte 	SI	Permite mensajes de entre e intra objetos para que se muestren en el mismo diagrama.
	Continuaciones / Invariantes de Estado	SI	Se puede encontrar extendida por una o más líneas de vida en distintos objetos.

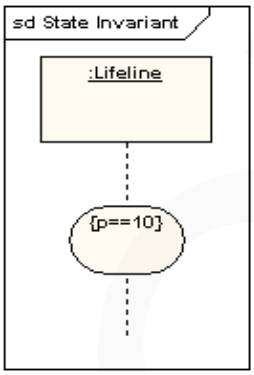
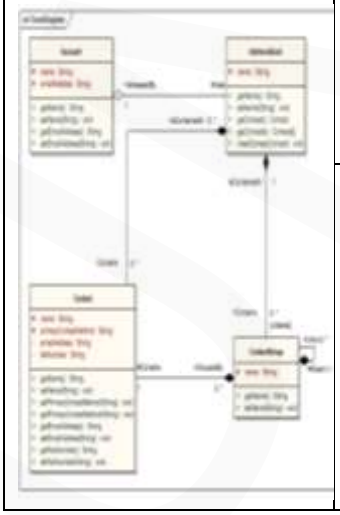
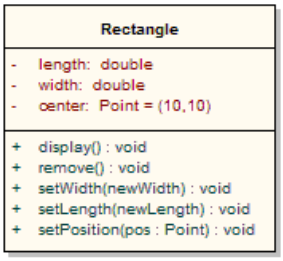
DIAGRAMA	COMPONENENTES	OPCIONAL	USO DE COMPONENTES
			
Diagrama de clase 	Clases	NO	Es un elemento que define los atributos y comportamientos que un objeto podrá generar.
	Notacion de clase 	SI	Se usan para denotar el nombre de la clase, atributos y operaciones.

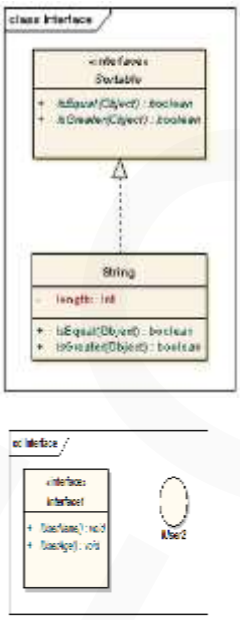
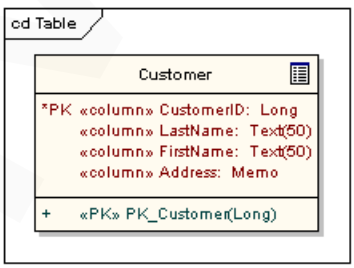
DIAGRAMA	COMPONENENTES	OPCIONAL	USO DE COMPONENTES
El diagrama de Clase muestra los bloques de construcción de cualquier sistema orientado a objetos.	Interfaces 	NO	Si se usa una interfaz, se garantiza que las clases soporten un comportamiento requerido, que permite que el sistema trate los elementos no relacionados en la misma manera. Se puede presentar en forma de circulo o rectangular con anotaciones.
	Tablas 	SI	Una tabla es una clase estereotipada.
	Asociaciones	SI	Una asociación implica que dos elementos del modelo tienen una relación,

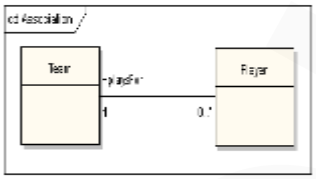
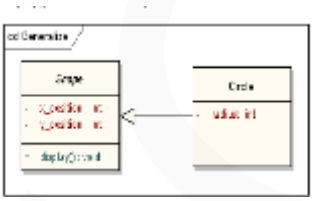
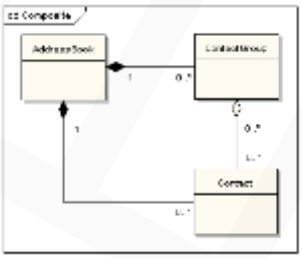
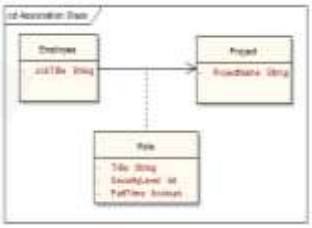
DIAGRAMA	COMPONENENTES	OPCIONAL	USO DE COMPONENTES
			usualmente implementada como una variable de instancia de una clase
	Generalizaciones 	SI	Una generalización se usa para indicar herencia.
	Agregaciones 	SI	Las agregaciones se usan para describir elementos que están compuestos de componentes más pequeños.
	Clase asociación 	SI	Una clase asociación es una estructura que permite una conexión de asociación para tener conexiones y atributos.
	Dependencias	SI	Una dependencia

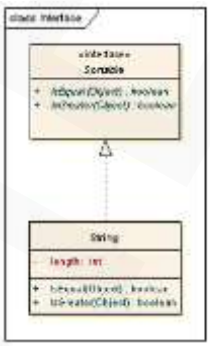
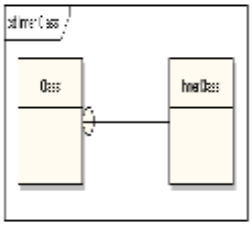
DIAGRAMA	COMPONENENTES	OPCIONAL	USO DE COMPONENTES
			se usa para modelar un alto rango de relaciones dependientes entre elementos del modelo.
	Trazado	SI	La relación de trazado es una especialización de una dependencia, vinculando elementos del modelo o conjuntos de elementos que representan la misma idea a través de los modelos.
	Relaciones 	NO	Se usa para expresar trazabilidad e integridad en el modelo.
	Anidamientos 	SI	Un anidamiento es un conector que muestra que el elemento fuente se anida dentro del elemento destino.

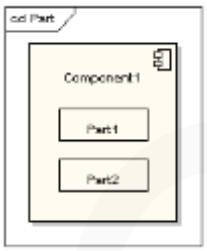
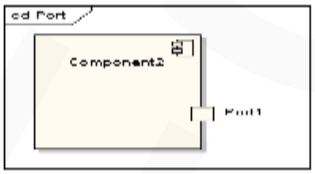
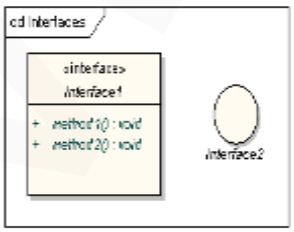
DIAGRAMA	COMPONENENTES	OPCIONAL	USO DE COMPONENTES
Diagrama de estructura compuesta Un diagrama de estructura compuesta es un diagrama que muestra la estructura interna de un clasificador, incluyendo sus puntos de interacción a otras partes del sistema.	Parte  	SI	Una parte es un elemento que representa un conjunto de una o más instancias que pertenecen a una instancia del clasificador contenida
	Puerto  	SI	Un Puerto es un elemento escrito que representa una parte visible externa de una instancia del clasificador contenido.
	Interfaces  	SI	Una interfaz es similar a una clase pero con un número de restricciones. Todas las operaciones de la interfaz son públicas y abstractas, y no proveen ninguna implementación predeterminada. Todos los atributos de la interfaz deben ser constantes. Sin embargo, mientras que una clase puede solo heredar de una sola súper-clase, puede implementar interfaces múltiples.

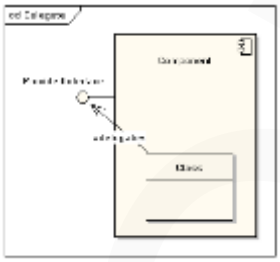
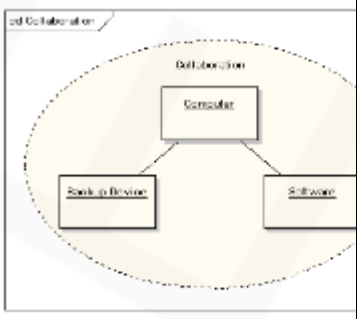
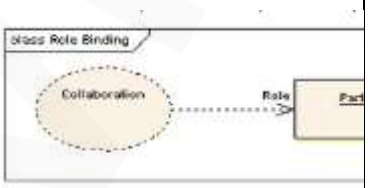
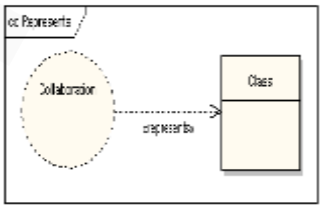
DIAGRAMA	COMPONENENTES	OPCIONAL	USO DE COMPONENTES
	Delegar 	SI	Un conector delegar se usa para definir los trabajos internos de los puertos e interfaces externas del componente.
	Colaboración 	SI	Una colaboración define un conjunto de roles cooperativos usados colectivamente para ilustrar una funcionalidad especifica.
	Enlace de roles 	SI	Un conector enlace de roles se dibuja desde una colaboración a un clasificador que completa el rol
	Representa 	SI	Un conector representa se puede dibujar desde una colaboración a un clasificador para mostrar que una colaboración se usa en el clasificador.

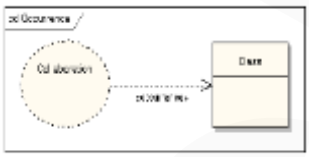
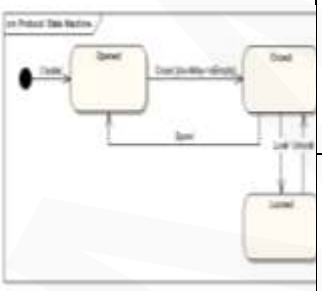
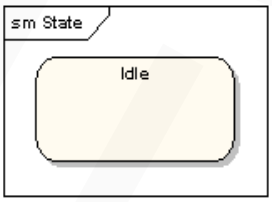
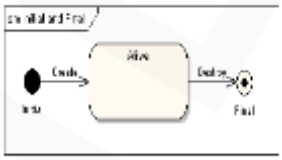
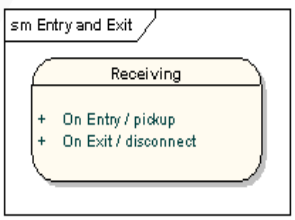
DIAGRAMA	COMPONENENTES	OPCIONAL	USO DE COMPONENTES
	Ocurrencia 	SI	Un conector ocurrencia se puede dibujar desde una colaboración a un clasificador para mostrar que la colaboración representa (sic) el clasificador.
Diagrama de máquina de estado 	Estados 	NO	Se denotan con su nombre dentro del mismo.
	Estado inicia y final 	NO	Indicador del estado.
	Transiciones 	SI	Una transición puede tener un disparador, una guarda y un efecto, como a continuación.
	Acciones de estado 	SI	Define las acciones que ocurren en los eventos, o acciones que siempre ocurren.

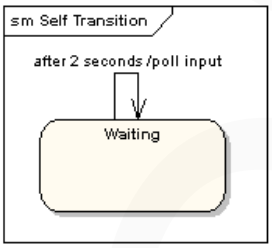
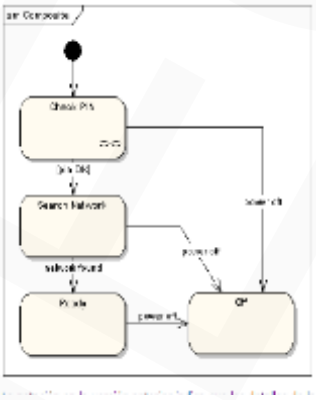
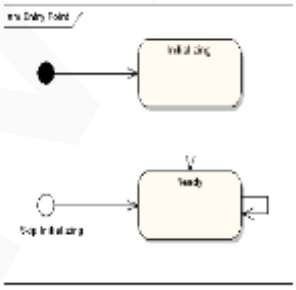
DIAGRAMA	COMPONENENTES	OPCIONAL	USO DE COMPONENTES
	Transacciones recursivas 	SI	Un estado puede tener una transición que retorna a sí misma.
	Estados compuestos 	SI	Se relaciona de componentes grandes a sus subcomponentes.
	Punto de entrada 	SI	Se puede presentar dos tipos de punto inicial en el cual se puede tener un inicio norma o por otro lado con condiciones o simplemente desde otro proceso.
	Punto de salida	SI	Se pueden tener puntos de salida nombrados lo que nos permite tener

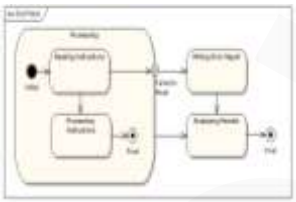
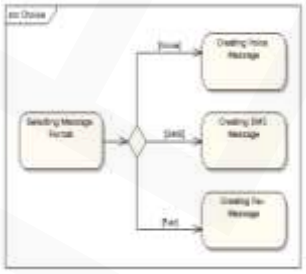
DIAGRAMA	COMPONENENTES	OPCIONAL	USO DE COMPONENTES
			varios puntos de salida.
	Pseudo estado elección 	SI	Un pseudo estado se muestra como un diamante con una transición llegando y dos o más transiciones saliendo.
	Pseudo estado unión 	SI	Los pseudo estados unión se usan para unir transiciones múltiples
	Pseudo estado terminar	SI	Ingresar un pseudo terminar indica que la línea de vida de la

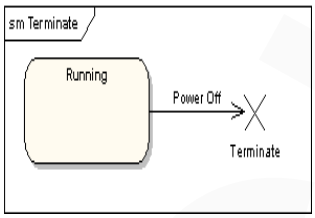
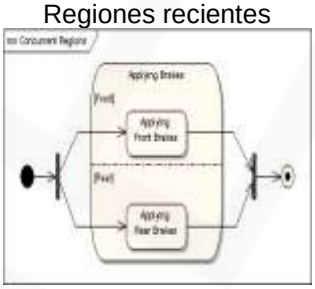
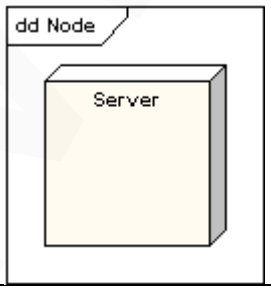
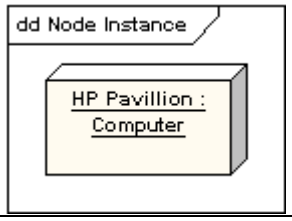
DIAGRAMA	COMPONENENTES	OPCIONAL	USO DE COMPONENTES
			máquina de estado ha terminado.
	Estado historial 	SI	Un estado historial se usa para recordar el estado anterior de una máquina de estado cuando fue interrumpida.
	Regiones recientes 	SI	Un estado se puede dividir en regiones conteniendo sub estados que existen y se ejecutan concurrentemente.
Diagrama de despliegue Un Diagrama de Despliegue modela la arquitectura en tiempo de ejecución de un sistema	Nodo 	NO	Un Nodo es un elemento de hardware o software.
	Instancia de nodo 	SI	Se puede distinguir desde un nodo por el hecho de que su nombre esta subrayado y tiene dos puntos antes

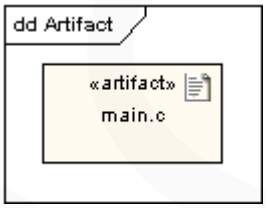
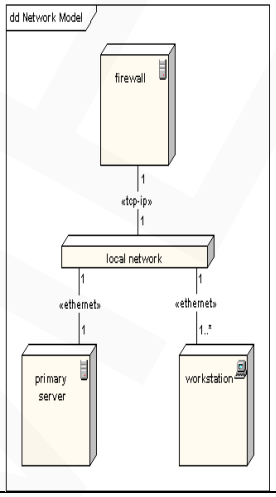
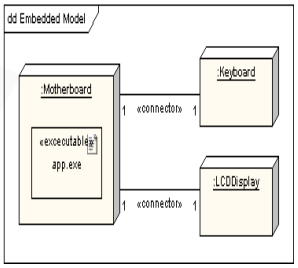
DIAGRAMA	COMPONENENTES	OPCIONAL	USO DE COMPONENTES
			del tipo de nodo base.
	Estereotipo de nodo 	SI	Un número de estereotipos estándar se proveen para los nodos.
	Artefacto 	SI	Un artefacto es un producto del proceso de desarrollo de software, que puede incluir los modelos del proceso.
	Asociación 	SI	En el contexto del diagrama de despliegue, una asociación representa una ruta de comunicación entre los nodos.
	Nodo como contenedor 	SI	Un nodo puede contener otros elementos, como componentes o artefactos.

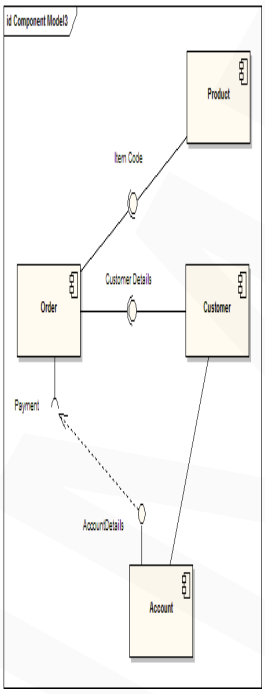
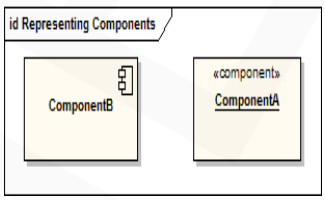
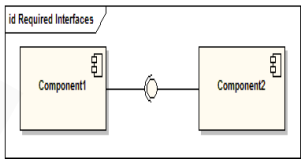
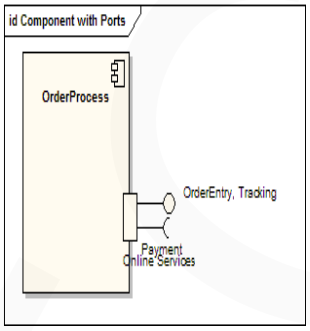
DIAGRAMA	COMPONENENTES	OPCIONAL	USO DE COMPONENTES
Diagrama de componentes  Los Diagramas de Componentes ilustran las	Representando componentes  	SI	Los componentes se representan como un clasificador rectangular con la clave «componente», opcionalmente el componente se puede mostrar como un rectángulo con un icono de componente en la esquina derecha arriba
	Interfaces requeridas  	SI (Únicamente obligatoria en caso de que un componente lo requiera)	Esto permite que un componente provea los servicios que otro componente requiere.

DIAGRAMA	COMPONENENTES	OPCIONAL	USO DE COMPONENTES
piezas del software, controladores embebidos, etc. que conformarán un sistema.	Componentes con puertos 	SI	Usar puertos con Diagramas de Componentes permite que se especifique un servicio o comportamiento a su entorno así como también un servicio o comportamiento que un componente requiere. Los puertos pueden especificar entradas, salidas así como también operar bidireccionalmente.

Para la realización de la anterior tabla se tuvo en cuenta la documentación brindada por UML 2.0 en su página principal, también se tuvieron en cuenta los diferentes documentos que brinda la empresa sparx systems para el uso de su herramienta estrella Enterprise Architect la cual han adaptado y mejorado para que los distintos tipos de diagrama se puedan estandarizar con UML en su última actualización.

Basados en esto es posible encontrar que cada uno de los componentes que se proponen para el desarrollo de los distintos diagramas tienen un uso excepcional y que en algunos casos es indispensable que se encuentren

en los diagramas, un ejemplo claro se puede encontrar en un diagrama de actividades en el cual no se presentan actividades el componente como tal con sus distintas especificaciones. Por tanto se ha enunciado en la tabla los componentes que se deben encontrar dentro de cada diagrama del modelado también se puede encontrar en que momento o en qué situación se debe usar cada componente con el fin de hacer más visible al evaluador de si el diagrama se encuentra bien estructurado. Se tiene claro que con el manejo de la metodología scrum los evaluadores tendrán el espacio de evaluar únicamente el modelado por tanto se está en la completa capacidad de tener en cuenta cada diagrama y sus distintos componentes y evaluar que se estén usando en el momento indicado.

Con el anterior estudio de lenguaje universal de modelado se ha generado la tabla anterior pero también se ha llegado a que como consecuente de este se debe tener muy en cuenta que si se tiene claro el modelado de un software se tiene claro que es lo que se espera en el desarrollo del mismo por tanto para las posteriores evaluaciones de calidad se debe tener siempre presente el modelado del proyecto.

Existen diversas herramientas que permiten automatizar el proceso de evaluación de software y que a su vez brindan informes con los resultados obtenidos una vez realizada la evaluación paso a paso, para la clasificación de dichas herramientas se tuvieron en cuenta criterios como la descripción de la misma, si cuentan con plataforma de escritorio es decir si es posible

instalarlo en el equipo en el que se van a realizar las pruebas o si solamente permiten el uso online, si es necesario utilizar bases de datos y si se trata de herramienta libres o propietarias, dentro de las herramientas dedicadas a este proceso es posible encontrar las siguientes

Tabla 17Definición de Herraminetas para pruebas

Descripción	**Herramienta**	**Plataforma escritorio**	**Plataforma web**	**Bases de datos**	**Tipo de herramienta**
Es una herramienta administradora de casos de prueba, permite realizar pruebas de software y reportes de defectos encontrados; Testopia permite a los usuarios acceder a una herramienta y utiliza permisos de grupo Bugzilla para limitar el acceso a la modificación de objetos de prueba.	Bugzilla Testopia	No	Si (Mozilla)	MySQL 5.0 o Postgres 8.1.x	Libre
Es una herramienta de código abierto, es adaptable incluye pruebas web/GUI. Se admiten los principales idiomas Java, Python, o C#.	FitNesse	No	Si (Chrome, Mozilla)	No	Libre
Herramienta basada en web, permite seguimiento de ciclos de pruebas, seguir revisiones de código, administrar y asignar recursos Track, informes en formato PDF.	qaManager	No	Si (Explorer, chrome, mozilla)	No	Libre

Descripción	Herramienta	Plataforma escritorio	Plataforma web	Bases de datos	Tipo de herramienta
Permite gestión de pruebas, creación, administración y edición de: requerimientos, casos de prueba , ejecución de pruebas; es posible utilizarla como aplicación de escritorio, versión web o Microsoft SharePoint (interfaz de usuario), compatible con Windows 7 y Windows 8.	qaBook	Si	Si (Chrome, Explorer, mozilla)	MySQL	Libre
Herramienta basada en la web de código abierto diseñado para gestionar los requisitos, pruebas, resultados de pruebas y defectos de todo el ciclo de vida de la aplicación. La herramienta proporciona un enfoque estructurado para las pruebas de software y aumenta la visibilidad del proceso de pruebas mediante la creación de un repositorio común para todos los activos de la prueba, incluyendo los requisitos, casos de prueba, planes de prueba, y los resultados de las pruebas.	RTH (Requisitos y pruebas Hub)	No	Si	No	Libre
Ofrece características para la creación y ejecución de las pruebas. Salomé-TMF utiliza el concepto de pruebas definidas en la norma ISO9646. Las pruebas pueden ser manual o automática, las pruebas se organizan en las campañas y se ejecutan con diferentes conjuntos de datos en diferentes entornos. Para hacer la ejecución de pruebas	Salome-tmf	No	Si	MySQL	Libre

Descripción	Herramienta	Plataforma escritorio	Plataforma web	Bases de datos	Tipo de herramienta
totalmente automático, Salomé-TMF ha integrado un lenguaje de script basado en Java, como uno de varios plugins que se extienden las funcionalidades Salome-TMF.					
Ha sido diseñado para el intercambio de proyectos de prueba entre las partes interesadas. Además, puede estar relacionado con Squash TA para lanzar y generar informes sobre las campañas de pruebas automatizadas.	Squash TM	No	Si (Mozilla, chrome, Explorer, safari)	NO	Libre
Es una aplicación escrita en PHP que permite a sus usuarios componer y gestionar planes de pruebas de control de calidad. Cada caso de prueba se puede asignar a un requisito funcional (documentado en otros lugares), se puede asignar un tipo (por ejemplo, la regresión, funcionales, etc.).le permite seleccionar y elegir qué prueba caso tendrá que ejecutar para cualquier instancia determinada del plan de pruebas. Luego seguir el paso y el ritmo de cada instancia de plan de pruebas.	Testitool	No	Si	No	Propietario
XStudio es una solución ALM modular y abierta que permite gestionar el ciclo de vida completo de sus productos . Studio es una solución ALM modular y abierta que permite	Xqual Studio	Si	Si	MySQL	Libre

Descripción	Herramienta	Plataforma escritorio	Plataforma web	Bases de datos	Tipo de herramienta
gestionar el ciclo de vida completo de sus productos . XQual Estudio (XStudio) es un 100% gráfica y modular en el diseño de aplicaciones de gestión de activos que maneja el ciclo de vida completo de sus proyectos de principio a fin: usuarios, requisitos, especificaciones, documentos, proyectos, pruebas, planes de pruebas, informes de pruebas y campañas de prueba.					
Es un entorno de pruebas de software para aplicaciones basadas en la web. Selenium provee una herramienta de grabar/reproducir para crear pruebas sin usar un lenguaje de scripting para pruebas (Selenium IDE). Incluye también un lenguaje específico de dominio para pruebas (Selanese) para escribir pruebas en un amplio número de lenguajes de programación populares incluyendo Java, C#, Ruby, Groovy, Perl, Ph p y Python.	Selenium	Si	Si(Mozilla, chrome, explorer, safari)	No	Libre
Comenzó en diciembre de 2005 para hacer el mismo tipo de pruebas de aplicaciones Web posible que los lenguajes .Net. Desde entonces WatiN se ha convertido en una herramienta fácil de usar, cuentan marco rico y estable. WatiN está desarrollado en C # y su	WatiN	Si	Si(Explorer, mozilla)	No	Libre

Descripción	Herramienta	Plataforma escritorio	Plataforma web	Bases de datos	Tipo de herramienta
objetivo es lograr que una manera fácil de automatizar sus pruebas.					
Herramienta para pruebas automatizadas de aplicaciones web de una manera muy eficaz. contiene una secuencia de comandos de utilidad capaz de crear un nuevo proyecto con algunos ejemplos listos para funcionar.	Canoo WebTest	Si	Si	No	Libre
Es una herramienta de prueba gratuita construida como un plug-in para el Eclipse IDE. Proporciona funciones para grabar una sesión de cliente, ajustarlo de acuerdo a diversos parámetros y reproducirla posteriormente lo general con el fin de garantizar la no regresión del comportamiento de la aplicación (con capacidades de pruebas de estrés que se añade en una etapa posterior).	Solex	Si	Si	No	Libre
Es una aplicación Java independiente que puede poner a prueba su sitio web desde la perspectiva del usuario. Es increíblemente sencillo y ligero, pero puede ser utilizado para la prueba poderosa de scripts utilizando	ITP	Si	Si	No	Propietario

Descripción	Herramienta	Plataforma escritorio	Plataforma web	Bases de datos	Tipo de herramienta
bloques de construcción para crear grandes ejecuciones de prueba.					
Es una herramienta de prueba de automatización web de código abierto que utiliza Watir como la biblioteca para conducir páginas web.Usted no tiene que descargar / instalar Watir separado o saber nada de Watir. Watir es la biblioteca que utiliza WET y se instala de forma automática. puede realizar todas las operaciones necesarias para las aplicaciones web de pruebas - como hacer clic automáticamente un enlace, la introducción de texto en un campo de texto, hacer clic en un botón, etc. WET le permite realizar diversos controles como parte del proceso de prueba mediante el uso de puntos de control.	WET	Si	Si	No	Libre
Es una herramienta gratuita para pruebas automatizadas de aplicaciones web y servicios web. Puede ser utilizado para probar los componentes individuales del sistema que tienen interfaces HTTP (JSP, ASP, CGI, PHP, AJAX, Servlets, formularios HTML, XML / Servicios web SOAP, REST, etc.), y se pueden utilizar como un instrumento de	WebInject	Si	Si	No	Libre

Descripción	Herramienta	Plataforma escritorio	Plataforma web	Bases de datos	Tipo de herramienta
prueba para crear un conjunto de [nivel HTTP] automatizado pruebas funcionales, de aceptación y de regresión.					
Es un probador web funcional y de carga, escrito en Python, cuyos casos el uso principal son: Prueba de funcionamiento de proyectos web, y por lo tanto las pruebas de regresión también. * Las pruebas de rendimiento: al cargar la aplicación web y el monitoreo de los servidores que le ayuda a localizar los cuellos de botella, dando un informe detallado de la medición del desempeño.	Funkload	Si	Si	No	Libre
Fast Web Performance Test Tool (Herramienta rápida web de prueba de rendimiento). Herramienta para probar aplicaciones web,para aplicaciones asp.net, jsp, php.Se puede grabar las solicitudes normales y AJAX.	Fwptt	Si	Si(IE, Firefos, opera)	MySQL	Libre
Plataforma integrada incluye la posibilidad de cargar API de prueba para asegurarse de que puede hacer frente a cualquier número de usuarios simultáneos. Reutilice sus pruebas API funcionales construido en el # 1 herramienta de prueba de API, SoapUI NG para acelerar las pruebas y reducir el tiempo	LoadUI	Si	Si	No	Libre

Descripción	Herramienta	Plataforma escritorio	Plataforma web	Bases de datos	Tipo de herramienta
que le lleva a desplegar servicios web REST alto rendimiento y SOAP.					
Aplicación Java puro 100% diseñado para cargar comportamiento funcional de prueba y medir el rendimiento. Originalmente fue diseñado para probar las aplicaciones web, pero desde entonces se ha expandido a otras funciones de prueba.	Jmeter	Si	Si	MySQL	Libre
Permite crear y organizar casos de prueba y configurar comandos con el fin de obtener y comprobar resultados de pruebas de disco. Esta plataforma funciona solo con el sistema operativo Linux permitiendo realizar pruebas de escritura libre.	Autotestnet	Si	No	No	Libre
Maneja diferentes formatos que se pueden usar para pruebas de software por medio de su código abierto en JavaScript para crear datos característicos.	Data generator	Si	Si	MySQL	Libre hasta 100 pruebas

Descripción	Herramienta	Plataforma escritorio	Plataforma web	Bases de datos	Tipo de herramienta
Incremental Scenario Testing Tool (Prueba escenario incremental) permite realizar casos de prueba teniendo en cuenta pruebas anteriores, se basa en experiencias de probadores de software, IST permite sesiones de pruebas flexibles, adaptables y eficientes que se adaptan perfectamente para el desarrollo de software ágil	IST	Si	Si	No	Libre
Es una herramienta que permite controlar la calidad, y de esta manera permitir mejor flujo de trabajo, visibilidad y tiempo de respuesta, uno de los objetivos de Litmus es servir como repositorio de casos de prueba, con capacidades de gestión, repositorio de resultados de pruebas, interfaz de consulta para visualización.	Litmus	No	Si	No	Libre
Implementa lenguaje de dominio específico, permite ejecutar múltiples modos de pruebas. Dos tipos de propiedades: - solo fuente: Comando de entrada y salida esperada (1 solo archivo). -Experimentación controlada: múltiples pruebas pueden ser generadas a partir de la misma fuente con diferentes resultados esperados.	MTS: Multi-Tester	No	Si	No	Libre

Descripción	Herramienta	Plataforma escritorio	Plataforma web	Bases de datos	Tipo de herramienta
Requirements and Tests Management Repository. Esta es una herramienta de software de prueba que le permite: • gestionar los requisitos de software a lo largo de su ciclo de vida • describir los escenarios y casos de prueba para asegurar la validación de estos requisitos • Campañas de ejecución específica de las pruebas • Siga todas las fallas y los errores se reúnen durante las pruebas: A través de gestor de fallos internos A través rastreador externa error (Bugzilla, predicador, redmine ... etc) La solución incluye una versión del sistema de gestión de proyectos, para los requisitos, para escenarios y casos de prueba que realiza un seguimiento de los cambios de software y pueden cubrir fácilmente todas las pruebas de regresión.	RTMR	No	Si	Si (Harmonized system database online)	Libre
Director de pruebas, soporta características como la configuración de plan de prueba , actualizaciones (crear/editar) los resultados de las pruebas para establecer los informes.	Radi	Si	No	No	Libre

Descripción	Herramienta	Plataforma escritorio	Plataforma web	Bases de datos	Tipo de herramienta
Es una herramienta para la gestión de pruebas de software en proyectos de software ágiles. Permite realizar pruebas ágiles, gestión de pruebas, informes, y usabilidad.	Tarántula	No	Si	No	Libre
Es una aplicación que permite crear, ejecutar, e informar sobre planes de prueba. Los planes de prueba se componen de casos de prueba y de procedimientos paso a paso.	Tesly	Si	Si	Si (MySQL, PostgresSQL)	Libre
Es una alternativa para plan de pruebas, es práctico, esta aplicación permite visualizar de una manera fácil el riesgo del proyecto. Permite crear ACC (Atributos componentes capacidades), y así reemplazar un plan de pruebas convencional.	Test Analytic	No	Si	No	Libre
Test environment toolkit. Es un sistema de Gestión de ejecución encargado de administrar y presentar informes, y secuencia de pruebas.	TET	Si	No	Si	Libre
Permite controlar la ejecución de pruebas y comparación de resultados reales con resultados previstos. Es posible automatizar algunas tareas repetitivas, pero necesarias en el proceso de pruebas existentes o añadir	TestAutomation	Si	No	No	Libre

Descripción	Herramienta	Plataforma escritorio	Plataforma web	Bases de datos	Tipo de herramienta
pruebas adicionales, dificiles de realizar de forma manual.					
Herramienta de gestión de casos de prueba basado en web, para realizar seguimiento de casos de prueba, hace el proceso de prueba eficiente y precisa.	TestCube	Si	Si	No	Libre
Herramienta que facilita el aseguramiento de calidad del software. Ofrece soporte para casos de prueba, planes de prueba, proyectos de prueba, gestión de usuarios, informes y estadísticas.	TestLink	Si	Si	Si(Mysql, PostgresSQL)	Libre
Es una herramienta de registro, presentación de informes y automatización de prueba, basado en web, gestión jerárquica de pruebas, importación de casos de prueba , interfaz de automatización de pruebas.	TestMaster	Si	No	Si (postgresSQL)	Libre
es una infraestructura que proporciona la solución en la que diferentes herramientas pueden conectar a sí mismo y hacer su trabajo de una manera unificada. Por lo tanto proporcionar una plataforma común para el	Utility Automation Vibz	Si	No	No	Libre

Descripción	Herramienta	Plataforma escritorio	Plataforma web	Bases de datos	Tipo de herramienta
ingeniero de automatización de hacer su trabajo.					
Herramienta de gestión de pruebas diseñado por testers para testers, administra casos de prueba, ejecución de pruebas durante todo el ciclo de vida del proyecto.	Vienna 2	Si	Si	No	Libre
Herramienta para gestión de prueba, casos de prueba simplemente en el código, prueba robustez, eficacia y eficiencia se evalúan en un informe limpio.	TestMP	Si	No	Si (MySQL)	Libre

Como se ha mencionado anteriormente en el estándar ISO/IEC/IEEE 29119 parte 2 se especifican los 3 tipos de pruebas en los que se clasifican para la realización de la evaluación, a continuación se presenta esta clasificación de acuerdo a las 25 pruebas que se realizarán a través del uso de esta metodología:

Es necesario identificar de qué se trata cada tipo de prueba:

* Gestión de Pruebas: Hace referencia al diseño de casos de prueba a su planificación, programación, y control de las mismas, permite definir el tipo de requerimientos que manejará la aplicación a evaluar, así mismo el nivel de prioridad de cada requerimiento, va relacionado con la documentación del proyecto.

* Pruebas Funcionales: Permite establecer y mantener el entorno de prueba, así mismo permite garantizar si el producto software cumple con las especificaciones, evaluaciones e inspecciones, y la comparación entre el resultado esperado y el resultado real.

* Pruebas de carga y rendimiento: Permite evaluar el comportamiento del producto software bajo condiciones de carga que sobrepasan los requisitos preestablecidos.

La metodología que permitire evaluar la calidad de productos software elaborados por el equipo de desarrollo de la Universidad de Cundinamarca, tendrá en cuenta desde el diseño de diagramas de acuerdo al lenguaje de modelado Universal (UML 2.0), aunque no es posible automatizar esta evaluación debido a que no existe una herramienta que realice dicho proceso, para lograr una evaluación adecuada se realizarán los diagramas de: casos de uso, diagrama de actividades, diagrama de clase, diagrama de secuencia, diagrama de despliegue y diagrama de colaboración; se

plantean una serie de elementos a tener en cuenta para garantizar un diseño adecuado de acuerdo a los criterios planteados.

Con referencia a la evaluación del software sobre pruebas automatizadas en la norma ISO/IEC/IEEE 29119 parte 1 se definen 41 tipos de prueba diferentes en el momento de realizar evaluación a un software, de las cuales para plantear esta metodología se eligieron 25 teniendo en cuenta ciertos criterios que se presentan en la tabla de pruebas a realizar sobre el software que posteriormente se clasifican en tres grupos definidos en el estándar en la segunda parte y se puede observar esta clasificación en la investigación realizada sobre las distintas pruebas.

Se identifican las pruebas y se suplen con herramientas más adecuadas y con el reconocimiento suficiente. Que se asignaron de la siguiente manera:

Gestión de pruebas: QaBook

Pruebas funcionales: JMeter

Pruebas de carga y estrés: LoadUI

7.14. Escenarios de prueba de la metodología

Para probar la metodología que se propone se realiza pruebas al producto de software utilizado en la Universidad de Cundinamarca llamado Sistema

de información de Autoevaluación (SIA), el documento de informe generado por la herramienta QABook para el proceso de testo del proyecto de autoevaluación módulo administrador de la Universidad de Cundinamarca, el cual muestra los criterios de éxito para ejecutar el proyecto, los resultados de las pruebas y los ambientes (entornos) y los casos de prueba utilizados para su ejecución.

Los indicadores clave son los encargados de proporcionar un resumen de la evaluación realizada y los resultados obtenidos, para llevar a cabo la prueba se tuvieron en cuenta:

54 requerimientos

1 ambiente de prueba

30 casos de prueba

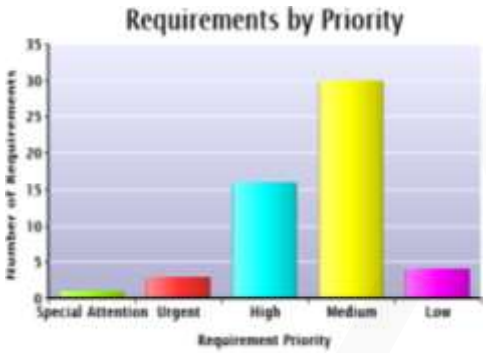
8 defectos encontrados

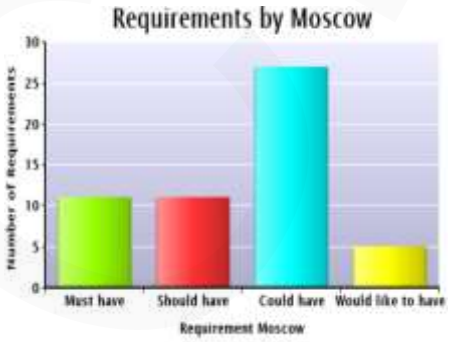
El estado actual del proyecto autoevaluación es “tolerante a defectos”, por lo que el número de defectos no sobrepasa los niveles de tolerancia.

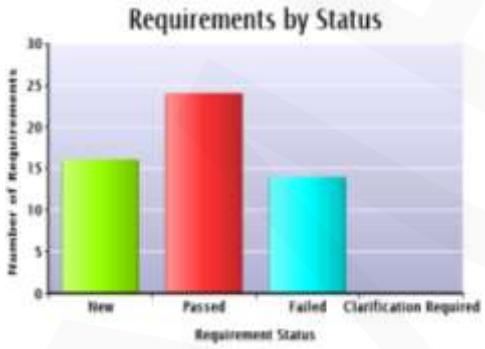
Los criterios de éxito del proyecto se definen por caso de prueba Rate Pass y Defecto Tolerancia. Prueba Caso Pass Rate es el número de casos de prueba como un total que debe pasar y el número y severidad de los defectos que se permiten para las pruebas para ser marcado como completado y exitosa.

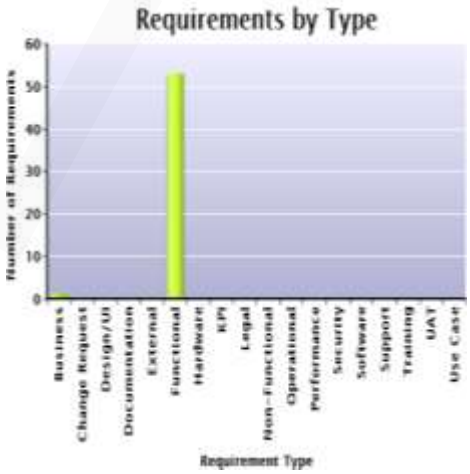
La evaluación se definió desde el 12 de febrero de 2017 hasta el 14 de marzo de 2017.

A continuación, se muestra gráficamente los requerimientos especificados para realizar la prueba:

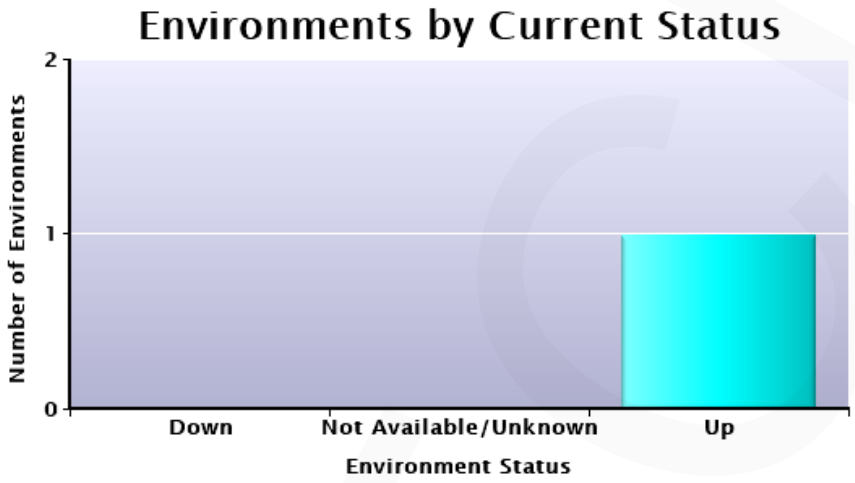


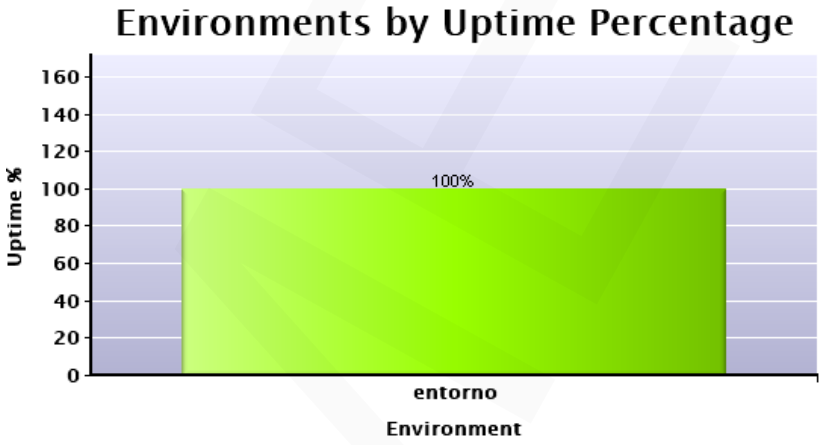




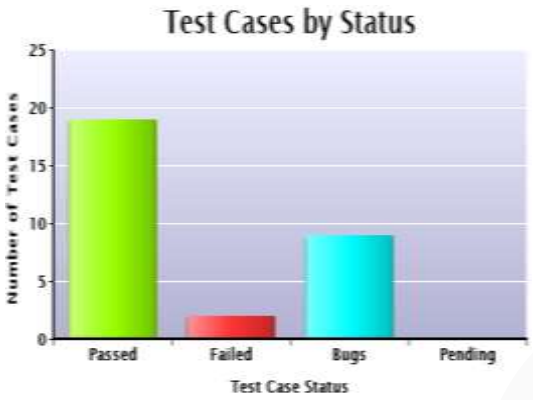


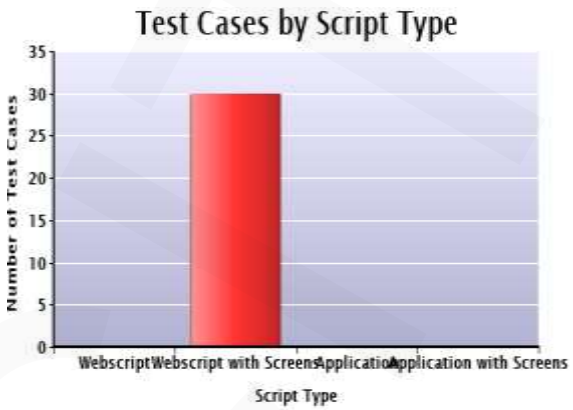
Entornos de prueba



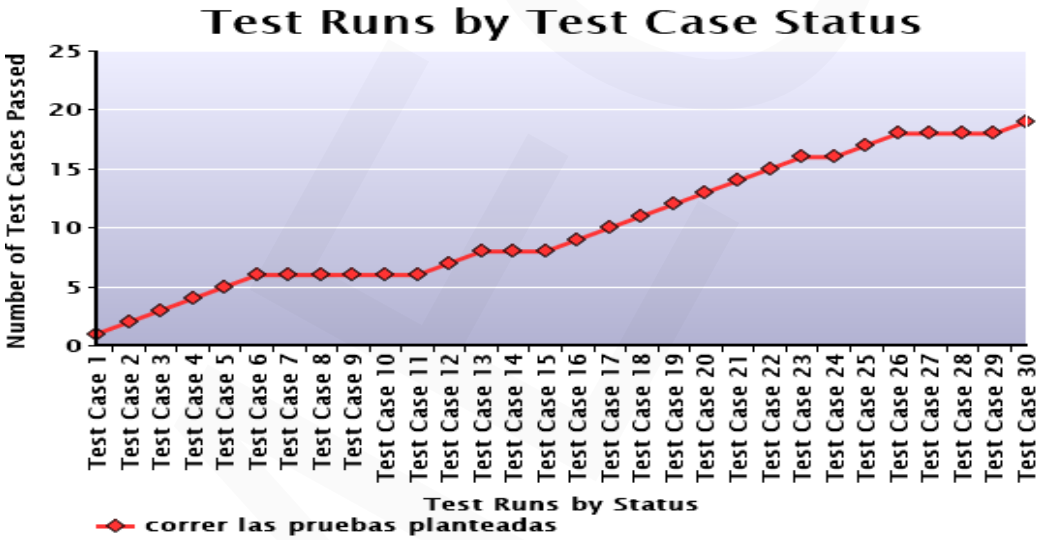


Gráficos de casos de prueba

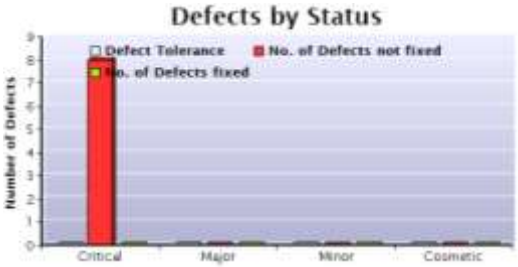


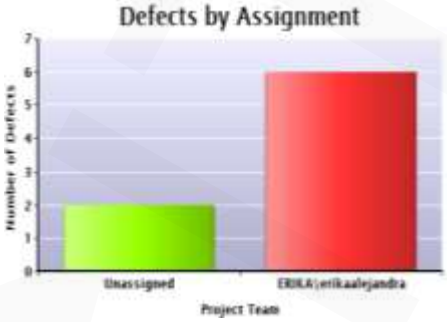


Prueba de funcionamiento



Gráficas defectos





Los requerimientos fueron definidos de la siguiente manera:

Requirements

Agregar módulos
Reference: 0001
Se pueden agregar diferentes módulos los que sean necesarios para el proceso de autoevaluación.

Requirement Status: Passed
Requirement Moscow: Should have
Requirement Priority: Medium
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra{
Approved: No

Test Case Association

- modulo se agrega de forma correcta

Modificar modulos
Reference: 0002
Se podrá actualizar o modificar el contenido de dicho modulo si se llega a necesitar.

Requirement Status: Passed

Requirement Moscow: Could have
Requirement Priority: Low
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra
Approved: No

Test Case Association

- es posible modificar modulo

Inhabilitar modulos
Reference: 0003
Deshabilitar los factores que ya no son necesarios mas no serán eliminados por motivos de tener un control de los factores anteriores.

Requirement Status: Passed
Requirement Moscow: Would like to have
Requirement Priority: Medium
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra
Approved: No

Test Case Association

- cambiar estado modulo

Ver modulos
Reference: 0004
Poder ver los modulos actuales que cuenta el programa de autoevaluación.

Requirement Status: Passed
Requirement Moscow: Must have
Requirement Priority: Medium
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra
Approved: No

Test Case Association

- observar el modulo

Agregar grupo
Reference: 0005
Se puede agregar diferentes grupos a los modulos correspondientes para el proceso de autoevaluación.

Requirement Status: Passed
Requirement Moscow: Could have
Requirement Priority: Medium
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra
Approved: No

Test Case Association

- insertar grupo

Modificar grupos
Reference: 0006
Poder modificar o actualizar el contenido de las grupos si así lo requiere el usuario.
Requirement Status: Passed
Requirement Moscow: Should have
Requirement Priority: Medium
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra
Approved: No

Test Case Association

- editar grupo

Inhabilitar grupo
Reference: 0007
Deshabilitar los grupos mas no serán eliminadas pues se quiete tener un historial de las características anteriores.

Requirement Status: Passed
Requirement Moscow: Could have
Requirement Priority: Medium
Requirement Type: Functional
Component: componente funcionalidad
Assigned To: ERIKA\erikaalejandra
Approved: No
Test Case Association

- editar grupo

Ver grupo
Reference: 0008
Poder ver las diferentes grupos para saber cuáles cuenta el programa de autoevaluación

Requirement Status: Failed
Requirement Moscow: Must have
Requirement Priority: Medium
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra
Approved: No

Test Case Association

- observar grupo

Agregar subgrupo
Reference: 0009
Poder agregar los diferentes subgrupos a los grupos del programa de autoevaluación.

Requirement Status: Failed
Requirement Moscow: Could have
Requirement Priority: High
Requirement Type: Functional
Component: componente funcionalidad

Assigned To:
Approved: No

Test Case Association

- insertar subrupo

Modificar subgrupo
Reference: 0010
Poder actualizar o modificar el subgrupo si se llega a solicitar sistema nacional de acreditación.

Requirement Status: Failed
Requirement Moscow: Must have
Requirement Priority: Medium
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra
Approved: No

Test Case Association

- editar subgrupo

Inhabilitar subgrupo
Reference: 0011
Se podrá deshabilitar los diferentes subgrupos mas no serán eliminados por motivos de tener un control de los aspectos anteriores.

Requirement Status: Failed
Requirement Moscow: Could have
Requirement Priority: High
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra
Approved: No

Test Case Association

- editar subgrupo

Ver subgrupos
Reference: 0012
Poder ver los diferentes subgrupos del programa de autoevaluación.

Requirement Status: Failed
Requirement Moscow: Could have
Requirement Priority: Medium
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra
Approved: No

Test Case Association

- observar subgrupo

Agregar actividad
Reference: 0013
Se podrá insertar todas las actividad necesarias enlazándolas a grupos y subgrupos

Requirement Status: Failed
Requirement Moscow: Could have
Requirement Priority: High
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra
Approved: No

Test Case Association

- insertar actividad

Modificaractividad
Reference: 0014
En caso de que alguna actividad haya quedado mal digitada se podrá modificar o en caso de que se quiera actualizar la actividad.

Requirement Status: Passed
Requirement Moscow: Would like to have
Requirement Priority: Low
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra
Approved: No

Test Case Association

- editar actividad

Inhabilitar actividad
Reference: 0015
Por motivos de querer que se tenga presente las actividades anteriores no se eliminaran pero estarán suspendidas.

Requirement Status: Passed
Requirement Moscow: Could have
Requirement Priority: Medium
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra
Approved: No

Test Case Association

- editar actividad

Ver actividad
Reference: 0016
El usuario podrá ver el enlace creado entre los grupos a las actividades para saber dónde están conectadas.

Requirement Status: Passed
Requirement Moscow: Could have
Requirement Priority: Medium
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra
Approved: No

Test Case Association

- observar actividad

Insertar Proceso
Reference: 0017
Se podrá crear los diferentes procesos que se pueden realizar en las diferentes fechas y que estarán presentes en los diferentes programas de la institución.

Requirement Status: New
Requirement Moscow: Would like to have
Requirement Priority: Medium
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra
Approved: No

Modificar Proceso
Reference: 0018
Se podrá modificar el contenido de los procesos que están en uso actualmente en caso de que sea requerido.

Requirement Status: New
Requirement Moscow: Would like to have
Requirement Priority: Medium
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra

Approved: No

Inhabilitar Proceso
Reference: 0019
Como en un futuro se quiere tener un registro de los procesos realizados en la institución se tendrán inhabilitados para tener los registros.

Requirement Status: New
Requirement Moscow: Could have
Requirement Priority: Low
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra
Approved: No

Ver Proceso
Reference: 0020
Los usuarios podrán ver los diferentes procesos que se han inscritos desde los que están en vigencia y los que ya se han realizado.

Requirement Status: New
Requirement Moscow: Must have
Requirement Priority: High
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra
Approved: No

Insertar Estado
Reference: 0021
Se podrá crear los diferentes estados de los procesos desde en ejecución, finalizados, etc.

Requirement Status: New
Requirement Moscow: Must have
Requirement Priority: Special Attention
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra
Approved: No

Modificar Estado
Reference: 0022
En caso de que el usuarios necesite modificar un estados o corregir su contenido podrá hacerlo si es requerido.

Requirement Status: New
Requirement Moscow: Must have
Requirement Priority: Urgent
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra
Approved: No

Inhabilitar Estado
Reference: 0023
El estado no desaparecerá por motivo de que en todo el aplicativo se quiere tener un historial de todos los procesos que se han realizado y se necesita saber qué estado tenían o tienen actualmente.

Requirement Status: New
Requirement Moscow: Could have
Requirement Priority: Medium
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra

Approved: No

Ver Estado
Reference: 0024
Se podrá observar los diferentes estados que se han creado y cuales se están usando actualmente para tener una clara visión de qué estado se asignara a los procesos.

Requirement Status: New
Requirement Moscow: Should have
Requirement Priority: High
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra
Approved: No

Insertar Grupo Interés
Reference: 0025
En este grupo de interés se insertara los diferentes grupos como son docentes, estudiantes, etc.

Requirement Status: New
Requirement Moscow: Could have
Requirement Priority: Medium
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra
Approved: No

Modificar Grupo Interés
Reference: 0026
En caso de una mala digitación se podrá modificar su contenido.

Requirement Status: New
Requirement Moscow: Should have
Requirement Priority: Medium
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra
Approved: No

Inhabilitar Grupo Interés
Reference: 0027
Como estos grupos estarán presentes en los procesos pero se quiere tener una evidencia de que grupos estuvieron no se eliminaran sino se dejaran suspendidos.

Requirement Status: New
Requirement Moscow: Must have
Requirement Priority: High
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra
Approved: No

Ver Grupo Interés
Reference: 0028
Se podrá observar los diferentes grupos de interés que cuenta el aplicativo para saber con cuales cuenta y cuales están habilitados.

Requirement Status: New
Requirement Moscow: Could have
Requirement Priority: Medium
Requirement Type: Functional
Component: (none)

Assigned To: ERIKA\erikaalejandra

Approved: No

Agregar Usuarios
Reference: 0029
Permite agregar los diferentes usuarios que puede entrar al módulo del súper administrador y el módulo del CNA.

Requirement Status: Failed
Requirement Moscow: Should have
Requirement Priority: Medium
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra
Approved: No

Test Case Association

- usuario agregar

Modificar Usuarios
Reference: 0030
Permite actualizar o modificar datos de los usuarios registrados.

Requirement Status: Failed
Requirement Moscow: Must have
Requirement Priority: Medium
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra
Approved: No

Test Case Association

- editar usuario

Inhabilitar Usuarios
Reference: 0031
Se podrán deshabilitar los usuarios mas no serán eliminados.

Requirement Status: Failed
Requirement Moscow: Could have
Requirement Priority: Low
Requirement Type: Business
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra
Approved: No

Test Case Association

- editar usuario

Ver Usuarios
Reference: 0032
Se podrá ver los usuarios que están registrados con sus respectivo rol y actividades.

Requirement Status: Passed
Requirement Moscow: Could have
Requirement Priority: High
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra
Approved: No

Test Case Association

- observar usuario

Agregar Rol
Reference: 0033
Creación de los diferentes roles donde estarán las actividades para los usuarios.

Requirement Status: Passed
Requirement Moscow: Should have
Requirement Priority: High
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra
Approved: No

Test Case Association

- crear rol

Modificar Rol
Reference: 0034
Se podrá actualizar o modificar las actividades o el contenido de dicho rol.

Requirement Status: Passed
Requirement Moscow: Could have
Requirement Priority: Medium
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra

Approved: No

Test Case Association

- editar rol

Inhabilitar Rol
Reference: 0035
Se deshabilitara el rol mas no se eliminara por motivos de saber que actividades tenia dicho rol.

Requirement Status: Passed
Requirement Moscow: Could have
Requirement Priority: Medium
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra
Approved: No

Test Case Association

- editar rol

Ver Roles
Reference: 0036
Se podrá ver los diferentes roles con la relación de las actividades que tiene encargado.

Requirement Status: Passed
Requirement Moscow: Could have
Requirement Priority: Medium
Requirement Type: Functional

Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra
Approved: No

Test Case Association

- observar rol

Adicionar un Rol a un Usuario
Reference: 0037
Se podrá dar un rol a cada usuario para así poder realizar ciertas actividades.

Requirement Status: New
Requirement Moscow: Must have
Requirement Priority: Urgent
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra
Approved: No

Agregar Actividades
Reference: 0038
Se crearan las diferentes actividades que podrán hacer los usuarios.

Requirement Status: Passed
Requirement Moscow: Could have
Requirement Priority: High
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra
Approved: No

Test Case Association

- crear actividad

Modificar Actividades
Reference: 0039
Se podrá actualizar o modificar el contenido de las actividades.

Requirement Status: New
Requirement Moscow: Could have
Requirement Priority: Medium
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra
Approved: No

Inhabilitar Actividad
Reference: 0040
Se deshabilitaran las actividades mas no serán eliminadas ya que podrán ser usadas en el futuro.

Requirement Status: New
Requirement Moscow: Could have
Requirement Priority: High
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra
Approved: No

Ver Actividades
Reference: 0041
Se podrán observar las diferentes actividades con la relación que tienen con los diferentes aspectos.

Requirement Status: New
Requirement Moscow: Should have
Requirement Priority: High
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra
Approved: No

Adicionar Actividades a los Roles
Reference: 0042
Se podrá agregar o quitar las actividades de los diferentes roles habilitados.

Requirement Status: Passed
Requirement Moscow: Could have
Requirement Priority: Medium
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra
Approved: No

Test Case Association

- agregar actividad rol

Crear Programa
Reference: 0043
Se creara el programa necesario para la facultad y la sede donde se le

asignaran la facultad y la sede

Requirement Status: Failed
Requirement Moscow: Must have
Requirement Priority: Urgent
Requirement Type: Functional
Component: (none)

Assigned To: ERIKA\erikaalejandra
Approved: No

Test Case Association

- insertar programa

Modificar Programa
Reference: 0044
Se podrá modificar su contenido o actualizarlo si así es requerido.

Requirement Status: Failed
Requirement Moscow: Could have
Requirement Priority: High
Requirement Type: Functional
Component: (none)

Assigned To: ERIKA\erikaalejandra
Approved: No

Test Case Association

- editar programa

Inhabilitar Programa
Reference: 0045
El programa se deshabilitara para que esté presente para los usuarios.

Requirement Status: Failed
Requirement Moscow: Could have
Requirement Priority: Medium
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra
Approved: No

Test Case Association

- editar programa

Ver programa
Reference: 0046
Se podrá ver los diferentes programas que hay disponibles en la base de datos.

Requirement Status: Passed
Requirement Moscow: Could have
Requirement Priority: Medium
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra
Approved: No

Test Case Association

- observar programa

Creación de sede
Reference: 0047
Se creara la sede necesaria para los programas con sus datos necesarios.

Requirement Status: Passed
Requirement Moscow: Could have
Requirement Priority: High
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra
Approved: No

Test Case Association

- insertar sede

Modificación de sede
Reference: 0048
En caso de que algún dato o contenido de la sede este mal o se quiera actualizar.

Requirement Status: Passed
Requirement Moscow: Would like to have
Requirement Priority: Medium
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra
Approved: No

Test Case Association

- editar sede

Inhabilitación de sede
Reference: 0049
Se deshabilitara la sede para tenerla en la base de datos si es necesario volverla a activar en el futuro.

Requirement Status: Passed
Requirement Moscow: Should have
Requirement Priority: Medium
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra
Approved: No

Test Case Association

- editar sede

Ver sede
Reference: 0050
Se observaran las sedes que tiene el aplicativo disponible.

Requirement Status: Failed
Requirement Moscow: Must have
Requirement Priority: High
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra
Approved: No

Test Case Association

- observar sede

Creación de facultad
Reference: 0051
Se creara la facultad para que más adelante sea relacionado con el programa donde se relacionara con la sede.

Requirement Status: Passed
Requirement Moscow: Should have
Requirement Priority: Medium
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra
Approved: No

Test Case Association

- insertar facultad

Modificación de facultad
Reference: 0052
En caso de una actualización del contenido o modificación de los datos

Requirement Status: Passed
Requirement Moscow: Should have
Requirement Priority: Medium
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra
Approved: No

Test Case Association

- editar facultad

Inhabilitación de facultad
Reference: 0053
Se deshabilitara la facultada pues no se quiere eliminar de la base de datos para un futuro ser utilizada.

Requirement Status: Passed
Requirement Moscow: Could have
Requirement Priority: High
Requirement Type: Functional
Component: (none)

Assigned To: ERIKA\erikaalejandra
Approved: No

Test Case Association

- editar facultad

Ver facultad
Reference: 0054
Se observaran las facultades disponibles que tiene la base de datos.

Requirement Status: Failed
Requirement Moscow: Should have
Requirement Priority: High
Requirement Type: Functional
Component: componente funcionalidad

Assigned To: ERIKA\erikaalejandra

Approved: No

Test Case Association

- observar facultad

7.4.3 Prueba funcional realizada con la herramienta JMeter

Para la ejecución de esta prueba se realizaron tres ambientes distintos en los cuales se efectúan sobre el software directamente montado en un servidor local lo que le da una respuesta rápida en casos en los cuales se mide también la capacidad de un servidor externo se encuentran más fallos de los que se dan si solo se prueba el aplicativo web como se realizó en este caso.

El primer ambiente que se creó se contaba con 3 usuarios en ejecución simultánea durante 60 segundos y se realiza con una repetición de 2 veces.

Como resultado de esta prueba obtenemos que en este caso como lo ejecutamos sobre local host el tiempo de conexión siempre será cero. Para esta prueba nos encontramos con 11 advertencias identificadas principalmente en la carga de archivos.

Se toman 1895 muestras en la cual la última tuvo un tiempo de carga de 6060 segundos y un promedio en la carga de todas las pruebas de 558

milisegundos. La diversidad en el tiempo de los procesos depende de las ejecuciones internas de cada proceso.

Posterior se ejecuta una prueba en la cual aumente el número de usuarios o ejecuciones simultáneas sobre el mismo tiempo para observar el comportamiento del software.

Tomaremos de la siguiente manera 60 usuarios simultáneos, en 60 segundos y dos repeticiones continuas la única variable que se ve alterada es el número de usuarios. Durante esta prueba se hace muy notorio el esfuerzo del equipo para una simulación tan pesada y se toma un tiempo largo durante la realización de esta prueba.

Se han encontrado 56 fallos sobre el software al ejecutar esta prueba exponiéndolo a una gran cantidad de usuarios simultáneos en corto tiempo.

De los cuales se puede notar que el tiempo de respuesta se hace más grande al tener mayor número de usuarios simultáneos. Algunos de los fallos se presentan sobre cargar un archivo y registros de distintos datos.

Para esta prueba se toman 20542 muestras que en un promedio cada una uso 16226 ms en obtener una respuesta aclarando que no se conecta con un servidor por tanto no se agrega tiempo de conexión.

La prueba se inicia el 29 de septiembre de 2015 a las 22:26 y culmina a las 23:47 por tanto se tomó 1 hora y 21 minutos en ejecutarse en su totalidad.

Se realiza la última ejecución con una cantidad de 60 usuarios simultáneos en 60 minutos en 2 repeticiones.

Se da inicio a la prueba a las 14:48:41 el 1 de octubre de 2015 y se da por terminada a las 17:02:31 durante la práctica se tomaron 19196 muestras de las cuales se encontraron 79 fallos.

Análisis de resultados:

Después de las ejecuciones en distintos escenarios de prueba tenemos que el software tiene una vulnerabilidad de 0.4%.

Esto se obtiene de:

La primera prueba con 3 usuarios en 60 segundos:

1895 muestras→100%

11 fallos→0.58%

La segunda con 60 usuarios en 60 segundos:

20542 muestras→100%

56 fallos→0.27%

La tercera con 60 usuarios en 3600 segundos:

19196 muestras→100%

76 fallos→0.41%

Sacando un promedio de los fallos durante cada ejecución se obtiene el porcentaje de falencias del software.

Ahora para el tiempo de respuesta obtenemos que durante la primera prueba la media de dichas ejecuciones es de 558 milisegundos, en el segundo escenario de prueba se tuvo una media de 16226 milisegundos y en la tercera ejecución se encontró un tiempo de respuesta promedio de 2804 milisegundos.

7.4.4 Prueba de carga y estrés aplicada con el software LoadUI

Para ejecutar una prueba de carga y estrés lo que se busca es conocer cuál es el máximo soporte de la aplicación puesta sobre un servidor para la realización de esta no se tuvo en cuenta el alcance del servidor puesto que se ejecutó sobre local host lo que se esperaba encontrar cual era la máxima capacidad del software para responder a una gran masa de usuarios. En algunos se realizan estas pruebas con un número definido de clientes para evitar poner al límite el aplicativo web esto se realiza por lo general cuando el comprador del software exige un número de usuarios mínimo.

Esta herramienta como ya lo he dicho anteriormente permite conocer el número de usuarios máximo sobre la aplicación en un segundo. Lo que se hace es poner la cantidad máxima que son 10000 usuarios por segundo y el software nos arrojara el número de usuarios atendidos correctamente o

respuestas y el número de fallos sobre los clientes ya definidos. Es necesario dejar que el software se regule y asimile la cantidad de usuarios en ejecución para así obtener resultados correctos por tanto en este caso se dejó un tiempo de 10 minutos.

Al llegar al tiempo definido se puede observar que se responden 2800 peticiones a usuarios correctamente por segundo y se presentan 7200 respuestas fallidas. En un tiempo menor antes de que el software se estabilice teníamos una respuesta por segundo de 1300 por tanto es necesario dejarlo correr el tiempo mínimo de 10 minutos o más si se hace muy notable la variabilidad de respuestas.

8. BIBLIOGRAFÍA

BOUKOUCHI, Y., MARZAK, A., BENLAHMER, H., & MOUTACHAOUIK, H. (2013). Comparative Study of Software Quality Models. *International Journal of Computer Science Issues*, *10*(6), 309. https://doi.org/IJCSI-10-6-1-309-314

Fahmy, S., Haslinda, N., Roslina, W., & Fariha, Z. (2012). Evaluating the Quality of Software in e-Book Using the ISO 9126 Model. *International Journal of Control and Automation*, *5*(2), 115–122.

Finding |. (n.d.). Retrieved June 15, 2017, from http://findingtc.com/

Galin, D. (2004). *Software quality assurance: from theory to implementation*. *Software Engineering, IEEE …*. Retrieved from http://books.google.com/books?hl=en&lr=&id=p5jDETUc2K8C&oi=fnd&pg=PR17&dq=Software+Quality+Assurance:+From+Theory+to+Implementation&ots=OJJtBcHL2B&sig=fnD_Z9APND4DEQs3Uak19mKk7

R0

GRUPO SOGETI. (n.d.). Retrieved June 15, 2017, from https://www.sogeti.es/sobre-sogeti/grupo-sogeti/

Home. (n.d.). Retrieved June 15, 2017, from http://www.mtp.es/

IEEE. (1990). IEEE_SoftwareEngGlossary.pdf, 84. Retrieved from www.mit.jyu.fi/ope/kurssit/.../IEEE_SoftwareEngGlossary.pdf

Inicio - Choucair Testing. (n.d.). Retrieved June 15, 2017, from https://www.choucairtesting.com/

Inicio - GreenSQA. (n.d.). Retrieved June 15, 2017, from http://greensqa.com/

Kläs, M., & Lampasona, C. (2011). Adapting Software Quality Models : Practical Challenges , Approach , and First Empirical Results. In *EUROMICRO Conference on Software Engineering and Advanced Applications* (p. 341–348.). https://doi.org/10.1109/SEAA.2011.6

Miguel, J. P., Mauricio, D., & Rodríguez, G. (2014). A Review of Software Quality Models for the Evaluation of Software Products, *5*(6), 31–53.

MINTIC. (2015). Guía para realizar el Análisis de Impacto de Negocios BIA Guía Técnica, (11). Retrieved from http://www.mintic.gov.co/gestionti/615/w3-article-5482.html

OCUPACIONAL. (n.d.). Retrieved June 15, 2017, from http://www.ingenieria.unicundi.edu.co/index.php/ingenieria-de-sistemas/perfil/ocupacional

QActions. (n.d.). Retrieved June 15, 2017, from http://www.qactions.com/

SENA. (2011). Formación en ambientes virtuales de aprendizaje. Retrieved June 15, 2017, from https://senaintro.blackboard.com/bbcswebdav/institution/semillas/228106_2_VIRTUAL-2015/contenido/oaaps/oaap10/aa2/oa_calidad/oa.pdf

Singh, I. (2013). Different Software Quality Model. *International Journal on Recent and Innovation Trends in Computing and Communication*, *1*(5), 438–412.

Software Testing Tools & Services - Applause. (n.d.). Retrieved June 15, 2017, from https://www.applause.com/es/

SQS S.A. - Servicios de Consultoría de Calidad de Software y Testing. (n.d.). Retrieved June 15, 2017, from http://www.sqs.es/

The Object Management Group. (2004). *Unified Modeling Language: Superstructure Version 2.0*. *OMG document formal/05-07-04*. https://doi.org/10.1109/VLHCC.2005.65