SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE APIZACO

"FRAMEWORK PARA PRUEBAS DE SOFTWARE ALINEADO A LOS ESTÁNDARES DE CALIDAD CMMI Y MOPROSOFT"

PRESENTA:

LIC. CYNDI GONZÁLEZ ÁGUILA

T E S I S

PARA OBTENER EL GRADO DE:

MAESTRA EN SISTEMAS COMPUTACIONALES

DIRECTORES DE TESIS:

M. EN C. MARÍA JANAÍ SÁNCHEZ HERNÁNDEZ

M. EN C. MARÍA GUADALUPE MEDINA BARRERA

APIZACO, TLAXCALA

AGOSTO 2018

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico de Apizaco

Apizaco, Tlax., 01 de Agosto de 2018

No. de Oficio: DEPI/259/18

ASUNTO: **Se Autoriza Impresión de Tesis de Grado.**

LIC. CYNDI GONZÁLEZ ÁGUILA
CANDIDATA AL GRADO DE MAESTRA
EN SISTEMAS COMPUTACIONALES
No. de Control: **M08370662**
P R E S E N T E.

Por este medio me permito informar a usted, que por aprobación de la Comisión Revisora asignada para valorar el trabajo, mediante la Opción: **I Tesis de Grado por Proyecto de Investigación**, de la **Maestría en Sistemas Computacionales**, que presenta con el tema: "**FRAMEWORK PARA PRUEBAS DE SOFTWARE ALINEADO A LOS ESTÁNDARES DE CALIDAD CMMI Y MIPROSOFT**" y conforme a lo establecido en el Procedimiento para la Obtención del Grado de Maestría en el Instituto Tecnológico, la División de Estudios de Posgrado e Investigación a mi cargo le emite la:

AUTORIZACIÓN DE IMPRESIÓN

Debiendo entregar un ejemplar del mismo debidamente encuadernado y seis copias en CD en formato PDF, para presentar su Acto de Recepción Profesional a la brevedad.

Sin otro particular por el momento, le envío un cordial saludo.

A T E N T A M E N T E
EXCELENCIA EN EDUCACIÓN TECNOLÓGICA®
PENSAR PARA SERVIR, SERVIR PARA TRIUNFAR®

DR. JOSÉ FEDERICO CASCO VÁSQUEZ
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN.

SECRETARÍA DE EDUCACIÓN PÚBLICA
TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE APIZACO
DIVISIÓN DE ESTUDIO DE POSGRADO E INVESTIGACIÓN

JFCV/MJSH*mebr.

C.p. Expediente.

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO
Instituto Tecnológico de Apizaco

Apizaco, Tlax., 27 de Junio de 2018

ASUNTO: **Aprobación del trabajo de Tesis de Maestría.**

DR. JOSÉ FEDERICO CASCO VÁSQUEZ
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN.
P R E S E N T E.

Por este medio se le informa a usted, que los integrantes de la **Comisión Revisora** para el trabajo de tesis de maestría que presenta la **LIC. CYNDI GONZÁLEZ ÁGUILA**, con número de control **M08370662,** candidata al grado de **Maestra en Sistemas Computacionales** y egresada del **Instituto Tecnológico de Apizaco,** cuyo tema es "**FRAMEWORK PARA PRUEBAS DE SOFTWARE ALINEADO A LOS ESTÁNDARES DE CALIDAD CMMI Y MIPROSOFT**", fue:

A P R O B A D O

Lo anterior, al valorar el trabajo profesional presentado por la candidata y constatar que las observaciones que con anterioridad se le marcaron así como correcciones sugeridas para su mejora ya han sido realizadas.

Por lo que se avala se continúe con los trámites pertinentes para su titulación.

Sin otro particular por el momento, le envió un cordial saludo.

LA COMISIÓN REVISORA

M.C. MARÍA JANAÍ SÁNCHEZ HERNÁNDEZ

M.C. MARÍA GUADALUPE MEDINA BARRERA

M.C. JOSÉ JUAN HERNÁNDEZ MORA

M.C. JUAN RAMOS RAMOS

C. p.- Interesada.

Instituto Tecnológico de Apizaco
Carretera Apizaco-Tzompantepec, Esq. con Av. Instituto Tecnológico S/N
Conurbado Apizaco-Tzompantepec, Tlaxcala, Méx.
C.P. 90300, Apizaco, Tlax. Tels. 01241 4172010, Ext. 146, 246
e-mail: depi@apizaco.tecnm.mx, **www.itapizaco.edu.mx**

Dedicatoria

Mi tesis la dedico con todo mi cariño:

A mis hijos Estela Yamilet, Axel y Álvaro por darme fuerza para seguir adelante, a mi amado esposo por confiar en mí y sobre todo por apoyarme en los momentos más difíciles.

A mis suegros por la confianza y apoyo brindado.

A mis padres y hermana por creer en mí, y por compartir momentos agradables y momentos tristes que me hicieron crecer y valorar muchas cosas.

Agradecimientos

Son muchas las personas que han contribuido a este trabajo. Antes que nada quiero agradecer al doctor Alberto Portilla Flores ya que fue mi asesor externo, agradezco el conocimiento, tiempo y apoyo brindado para la realización de este proyecto.

Agradezco a la empresa Miracle Business Network por el apoyo e información necesaria para hacer posible este proyecto.

A mis directores de tesis la M. en C. María Janaí Sánchez Hernández y María Guadalupe Medina Barrera por el conocimiento y apoyo brindado para la conclusión de este proyecto.

Al Conacyt por el apoyo económico brindado durante el periodo de vida de este proyecto.

Resumen

En este proyecto de tesis se realizó un estudio para desarrollar una propuesta que consta de un *framework* alineado a dos estándares de calidad, CMMI y MOPROSOFT, para realizar pruebas de software, con la finalidad de mejorar la calidad de los productos en una empresa. Se diseñaron formatos para un mejor control de las pruebas a ejecutar, así como de los resultados obtenidos. La ejecución de pruebas se realizó por medio de dos metodologías de desarrollo agiles (Scrum) y clásica (Cascada), estas para que se puedan adaptar a las necesidades de cada proyecto.

Se utilizaron herramientas open source JUnit y Selenium para la semi-automatización de las pruebas. Las pruebas que se ejecutaron mediante las herramientas fueron tres: pruebas unitarias, pruebas de integración y pruebas funcionales, aplicadas a dos proyectos diferentes en cuanto a complejidad, tamaño y necesidades del proyecto, además de las metodologías antes mencionadas.

El proceso de ejecución de pruebas de acuerdo a lo propuesto en el *framework* resulto una guía útil para realizar la ejecución de los diferentes tipos de pruebas, al mismo tiempo de que se aumentó la calidad del sistema. Con la aplicación del *framework* se reducen los defectos del software antes de la liberación, proporcionando al cliente una mejor calidad y pronta satisfacción, además de no incrementar los costos en los proyectos.

Abstract

In this thesis project a study was carried out to develop a proposal that consists of one *framework* aligned to two quality standards, CMMI and MOPROSOFT, for testing of software, with the aim of improving the quality of the products in a company, were designed formats for a better control of the tests to run as well as the results obtained, the test execution was performed by means of two development methodologies agile(Scrum) and classic(Waterfall) these to be able to adapt to the needs of each project.

We used open source tools JUnit and Selenium to the semi-automation of the tests, the tests that were run using the tools were three: unit tests, integration tests and functional tests, applied to two different projects in terms of complexity, size and needs of the project, in addition to the methodologies mentioned above.

The process of execution of tests according to what is proposed in both frameworks turned out to be a useful guide for performing the execution of the different types of tests, at the same time increase the quality of the system, with the application of the frameworks is to reduce software defects before release, providing the customer a better quality and prompt satisfaction, in addition to not increase the cost of the projects.

Indice

Tablas

Figuras

Abreviaturas y acrónimos

CRUD	Crear, Leer, Actualizar y Borrar(Create, Read, Update and Delete)
CMMI	Integración de modelos de madurez de capacidades (Capability Maturity Model Integration)
DMS	Desarrollo y Mantenimiento de Software
ISO	Organización Internacional de Normalización(International Organization for Standardization)
MOPROSOFT	Modelo de Procesos para la Industria del Software
MBN	Miracle Business Network
PIM	Modelos Independientes de la Plataforma tecnológica
PSM	Modelos específicos de la plataforma
SEI	Software Engineering Institute
SDO	Organizaciones de Desarrollo de Software
SIU	Consorcio integrado por 39 Universidades Nacionales Públicas que desarrolla soluciones informáticas y brinda servicios para el Sistema Universitario Nacional
SPDEF	Marco de Implementación y Evaluación de Procesos de Software
SWPM	Manual de práctica de cableado estándar (Standard Wiring Practice Manual)
TI	tecnología de la información

Capítulo 1

Introducción

1 Introducción

En este Capítulo se presenta los preliminares del trabajo de investigación, ya que se definen el contexto del problema, planteamiento del problema y la propuesta de solución. También presentamos las investigaciones y /o modelos existentes de pruebas de software para realizar el aseguramiento de la calidad del software y un cuadro comparativo de los framework ya existentes, con el fin de conocer las técnicas y estrategias ya probadas para poder ser utilizados como referencia para la realización del framework para pruebas de software.

1.1 Contexto del problema

Pruebas de Software

Son técnicas aplicadas en el desarrollo del software cuyo objetivo es mostrar información de los errores que se encuentren en el software, con la finalidad de mejorar la calidad de los productos y/o servicios antes que sean entregados al usuario final.

Las pruebas de software verifican que se cumpla con las especificaciones planteadas por el cliente o analista, eliminando los errores que se hayan cometido en cualquier etapa del desarrollo del software. Las pruebas son básicamente un conjunto actividades que se deben realizar para asegurar la calidad de los componentes del software.

Actualmente es necesario empezar a realizar pruebas desde etapas tempranas, ya que entre más tarde comiencen a realizarse las pruebas hay más posibilidad de encontrar

errores además de que con esto se incrementan tanto los costos como el tiempo en el desarrollo.

Las pruebas de software son una parte importante del desarrollo del software, pero también muy costosa.
Pueden llegar a representar entre el 30 y 50% del costo total del desarrollo del software (Kuhn, Wallace, & Gallo Jr, 2004)
Generalmente se comienzan probando las partes más pequeñas para continuar con las más grandes. Además de los distintos tipos de pruebas también existen estándares de calidad que ayudan a tener una estabilidad y confiabilidad a lo largo del desarrollo del software.

Miracle Business Network (MBN)

El trabajo de investigación se realizo en la empresa Miracle Business Network S.A. de C.V. (MBN), que es una empresa desarrolladora de software que se encuentra ubicada en el estado de Tlaxcala. La empresa se ha caracterizado por adoptar la calidad como una forma de vida, ajustándose a distintas dinámicas y metodologías como 5S's, comunicación efectiva, liderazgo, trabajo en equipo, entre otras.

Con base en el análisis de distintas normas relacionadas con empresas de TI, desde inicios del año 2008 se toma como estrategia de largo plazo en MBN adoptar la norma MoProSoft-059-NYCE-2005 como una herramienta estratégica para dar estructura a la organización.

Después de permanecer tres años en el nivel 1 de la norma Moprosoft se decide alcanzar el siguiente nivel de calidad, por lo que, en el año 2012, se accede al servicio de consultoría para la implementación del modelo de calidad MoProSoft para nivel 2. Así, a inicios del año 2013 se recibe la certificación del Nivel 2 de MoProSoft, siendo la única compañía del estado que ostenta dicha certificación y una de las 4 de la región Puebla-Tlaxcala.

En enero de 2015, MBN es evaluada de manera satisfactoria en SCAMPI-A por evaluadores del Software Engineering Institute (SEI) para la obtención del Nivel de Madurez 2 del modelo de CMMI-Dev v.1.3. CMMI (Capability Maturity Model Integration). Es un modelo para la mejora de procesos que proporciona a las organizaciones los elementos esenciales para implantar y mantener procesos eficaces que ayuden a mejorar su rendimiento. La evaluación en el nivel de Madurez 2 indica que se están llevando a cabo sus procesos en un nivel "gestionado".

En este nivel los proyectos se realizan conforme a una gestión disciplinada, se establecen políticas organizativas, se definen los procesos, se planifican, monitorean y ejecutan de acuerdo a su descripción. El obtener este nivel de madurez es un logro significativo para Miracle Business Network, ya que los procesos implantados permitirán reducir costos y tiempos en el desarrollo de proyectos bajo un enfoque de calidad buscando la satisfacción de los clientes. Como parte de la experiencia formada a lo largo de más de una década, la empresa MBN a la fecha cuenta con un equipo altamente calificado para realizar cualquier desarrollo, ajuste o customización en el cual se logre una transformación del servicio de manera que el cliente (customer) participe activamente para lograr el cumplimiento de los objetivos establecidos en el proyecto desarrollado por la empresa. Por ejemplo, respecto al área de Fábrica de software se han construido unidades de programación como:

- Desarrollo de software OnSite (en el sitio)
- Desarrollo de software en fábrica
- Actualización y mantenimiento de aplicaciones
- Servicio de consultoría tecnológica. (MBN, 2016)

1.2 Planteamiento del problema

Existe una gran demanda creciente de servicios y productos de software, es por eso que la competitividad de la organización MBN depende de incrementar calidad y reducir tanto costos como plazos de entrega de productos y servicios.

De acuerdo a la globalización en la que estamos inmersos, es necesario realizar pruebas para que se pueda evitar el costo que conlleva detectar errores de manera tardía esto debido a la complejidad del software que hace que este sea susceptible de errores, causando una pérdida considerable tanto en tiempo como a nivel económico.

Sin embargo, a pesar de la importancia y beneficios que se obtienen con el testing, en MBN no existe un Framework que reúna los procesos vigentes que permitan detectar errores en el software, motivo por el cual en este proyecto se pretende diseñar un Framework que permita mejorar la calidad de un proceso de desarrollo de software alineado a dos estándares de calidad, ya que en MBN realizan buenas prácticas tomando como guía dos metodologías, CMMI y MOPROSOFT, teniendo como referencia una mejora continua en cuanto a la calidad, al mismo tiempo que se ahorran tanto costos como tiempo invertidos, a su vez, se genera una pronta satisfacción y aceptación por parte del cliente, reduciendo la probabilidad de encontrar defectos en el software. De esta manera se genera un vínculo de identificación plena entre el consumidor y la empresa.

Objetivos

Objetivo general:

Proponer un framework para realizar pruebas unitarias, de integración y funcionales de software que permitan de manera metódica realizar este proceso apegado a estándares de calidad (CMMI y Moprosoft).

Objetivos específicos:

- Revisar literatura de testing.
- Revisar los modelos CMMI y Moprosoft.
- Proponer una estructura integral para el framework.
- Realizar pruebas del framework.

Alcances:

Propiciar para MBN un testing que reduzca los defectos en la liberación del software evitando de esta manera los costos y las pérdidas de servicio.

Limitaciones:

- Son necesarias herramientas externas para la automatización de las pruebas (JUnit y Selenium)
- Solo se realizarán pruebas unitarias, integración y funcionales.
- El framework se desarrollará con base a las metodologías de calidad CMMI y Moprosoft.

1.3 Propuesta de solución

Proponer un framework para pruebas de software con el objetivo de mejorar la calidad del software, ya que son las características esperadas al entregarse a un cliente. El framework se desarrollará con base en dos modelos CMMI y Moprosoft, ya que ayudan a tener una idea más clara de cómo debe ser planteado el entorno de calidad en la empresa, ya que actualmente MBN, lleva a cabo la aplicación de dos metodologías de desarrollo ágil y clásicas para poder satisfacer las necesidades del cliente en cuanto al costo y tiempo.

1.4 Pregunta de investigación

¿Se puede diseñar un framework alineado a dos estándares de calidad CMMI y Moprosoft, para la aplicación con dos metodologías de desarrollo de calidad ágil y clásica que permita reducir los defectos del software antes de su liberación?

1.5 Trabajos relacionados

Agile Practices Adoption in CMMI Organizations: A Systematic Literature Review (Marco Palomino, 2016)

En los últimos años, la adopción de marcos de trabajo y metodologías ágiles en las Organizaciones de Desarrollo de Software (SDO) ha crecido considerablemente. El objetivo de este estudio es identificar y analizar las prácticas ágiles más utilizados que se utilizan en combinación con el CMMI dentro de SDO. Este estudio ha identificado prácticas relacionadas reunión diaria y la gestión de cartera de productos de framework Scrum como prácticas ágiles más comunes utilizados en combinación con CMMI. Además, se pudo identificar que existen beneficios específicos de la aplicación de prácticas de ambos enfoques.

CMMI (Capability Maturity Model Integration), que es un modelo que agrupa las mejores prácticas en el desarrollo y las actividades de mantenimiento; es un modelo de proceso que ha sido adoptado por muchos SDO, las prácticas y la adopción de procesos a partir de un cierto nivel es relativamente un reto en las empresas pequeñas, por eso la importancia de identificar prácticas ágiles que en Concordancia con el CMMI puede ayudar en la mejora del proceso de desarrollo de software.

Existen varios estudios relacionados con prácticas ágiles y CMMI y cómo estos diferentes enfoques trabajan juntos en primer lugar, existen investigaciones de prácticas ágiles cómo podría contribuir a obtener niveles de madurez de CMMI. En segundo lugar, existen estudios de caso, que describen las consecuencias de la aplicación de prácticas ágiles dentro de una organización con cultura de CMMI.

El más reciente en esta investigación son: (i) verificar si el tamaño del equipo afecta el uso combinado de Agile y CMMI, (ii) analizar los estudios y las investigaciones publicadas hasta 2016, que amplía el ámbito de la investigación anterior (iii) analizar si cualquier metodología ágil puede utilizarse en contextos de organizaciones de CMMI.

Además de las preguntas de la investigación, también se realizó un examen de los 52 estudios seleccionados a fin de analizar la publicación años, canales para la publicación de investigación y el tipo de todos los estudios identificados al final del proceso de selección.

A partir del análisis de los 52 estudios primarios, podríamos identificar que ambos enfoques Agile y CMMI no son opuestos el uno al otro. De hecho, ambas culturas comparten criterios y prácticas similares. A partir de la premisa anterior, podemos afirmar que existe un nivel de compatibilidad entre Agile y CMMI enfoques que se ve corroborado en los estudios. Donde pudimos determinar que las prácticas de diferentes culturas, como ágiles o CMMI, pueden complementarse mutuamente con el fin de mejorar los procesos actuales.

Análisis de todos los 52 estudios primarios que se apliquen prácticas ágiles en el CMMI Contextos porque esa combinación permite a las organizaciones:

- Reducir el desperdicio del tiempo dentro del equipo
- Reducir el tiempo de entrega de los productos
- Aumentar la productividad del equipo
- Mejorar la competitividad de las organizaciones y de la calidad del producto
- Incluyen la flexibilidad y agilidad en los procesos de las organizaciones de CMMI
- Mejorar la comunicación con los interesados mediante prácticas ágiles

El resultado de la investigación concluyó que la integración de enfoques Agiles y CMMI son considerados por la industria de software como directrices con principios opuestos y, en algunas circunstancias, incompatibles; sin embargo, han encontrado en investigaciones recientes que ambos comparten los mismos objetivos y que pueden confluir para contribuir provechosamente a las organizaciones.

Por otro lado, las organizaciones con Agile directrices se benefician de CMMI porque incorporar las buenas prácticas que añadir formalidad en la infraestructura organizativa. Por otro lado, podríamos comprobar que hay estudios que indican que es posible obtener la certificación en los primeros niveles de madurez de CMMI, mediante el uso de prácticas ágiles. Además, el uso de diversas prácticas de diferentes metodologías ágiles permite a las empresas aplicar para aumentar aún más los niveles de madurez de CMMI.

La principal amenaza identificada es el sesgo de selección, porque los resultados de la investigación están condicionados.

La adecuada selección de los estudios primarios. La omisión de alguno de los estudios que pueden contribuir a la investigación es una de las amenazas más importantes a tener en cuenta.

An analysis on the significance of ticket analytics and defect analysis from software quality perspective (Christa & Suma, An analysis on the significance of ticket analytics and defect analysis from software quality perspective, 2016)

Este documento se centra en la importancia de mantener la calidad de los productos de software y también lo distinguen de los defectos asociados con un sistema de software.
Este proyecta la importancia de identificar defectos en el software, así como manejar el incidente relacionado entradas y resolverlo cuando son vistos desde la perspectiva de la calidad.
Para conseguir un producto de alta calidad etiqueta, los defectos en el sistema deben estar bien dirigida.

Alrededor del 40 al 50 por ciento del total de los gastos asignados a un proyecto se gasta en las pruebas y la fase de integración.

Se añade a la productividad, el esfuerzo, así como el costo asociado con el tratamiento de un defecto. El costo de encontrar y corregir un defecto antes del lanzamiento es muy inferior en comparación a encarar una post-producción defecto. El coste y el esfuerzo de resolver un defecto varían según la fase en la que se detecta. En realidad, el costo aumenta si el defecto está identificado en una fase avanzada.
Las pruebas de software, así como las técnicas que comprueban que el software cumple las expectativas que el cliente espera, juega un papel importante en la detección de defectos. Incluso entonces los defectos surgen después de la entrega del producto.
El costo de arreglar el defecto en un sistema post-parto es tres veces más costosa que la fijación durante la fase de desarrollo.

Los problemas no planificados, puede llevar a reducir la calidad de los servicios. Las industrias en esta investigación son CMMI Nivel 5 y también la certificación ISO (especifica que los requisitos para los sistemas de gestión de la calidad).

El estudio de caso se basa en los datos obtenidos de una empresa madura, que se adhieren a la prevención de defectos diferentes, así como la calidad de las actividades relacionadas con el mantenimiento.

Sobre la base de la categoría trivial y la gravedad, los defectos se clasifican como urgente, mediana y baja.

Este documento tiene el propósito de proyectar el aspecto importante de la calidad de un sistema de software, es decir, análisis de los defectos.

La calidad de cualquier sistema de software también está determinada por la calidad del servicio prestado a los incidentes inesperados que ocurren en él.

Influencia de la gestion de la calidad en los resultados de innovación a través de la gestión del conocimiento (García-Fernández, 2016)

El objetivo de este artículo es analizar la influencia de la gestión de la calidad en la innovación a través de la gestión del conocimiento. La metodología utilizada es a través del estudio de datos primarios (entrevistas en profundidad -entrevista personal y cuestionario de observación directa) y secundarios documentos internos y externos) de 5 empresas de servicios.

Los resultados muestran que la gestión de la calidad de impacta positivamente y la innovación a través de la gestión del conocimiento que puede ser un elemento mediador de este modo las empresas analizadas tienen un mayor grado de gestión de la calidad esto es que se desarrollan en mayor medida a las prácticas de gestión de calidad liderazgo planificación de la calidad gestión personal gestión de procesos información y análisis enfoque en el cliente gestión de proveedores y diseño de producto obteniendo mayores resultados e Innovación de procesos y de producto y medio de las prácticas de gestión del conocimiento creación almacenamiento y transferencia y aplicación y uso del

conocimiento la principal contribución de esta investigación es aportar información sobre el papel mediador de la gestión del conocimiento es la relación entre la gestión de calidad y la innovación. Asimismo, para las empresas de menor nivel de gestión de calidad de los resultados sirven como ejemplo para mejorar sus niveles innovación tomando como referencia las prácticas de la empresa de mayor nivel de gestión de la calidad.

Communication between Developers and Testers in Distributed Contino usAgile Testing (Cruzes, Moe, & Dybå, 2016)

Desarrolladores y probadores de software tienen que trabajar para alcanzar los objetivos de los proyectos de desarrollo de software

Durante la prueba en tales equipos, desarrolladores y evaluadores deben coordinar y comunicarse con frecuencia.

Sin embargo, la coordinación se ve afectada por la disponibilidad del proyecto de información, que se distribuye entre los diferentes miembros del proyecto y las estructuras organizativas.

En este artículo se investigó como la comunicación entre los desarrolladores y probadores de software en equipos de dos empresas de desarrolladoras de software realizan continuas pruebas con metodología ágil en una configuración distribuida.

Se describen cuatro prácticas de comunicación utilizados por el equipo: entrega a través del sistema de seguimiento de incidencias, reuniones formales, escrito de comunicación y la coordinación por ajuste mutuo.

Han encontrado que la pronta participación de los evaluadores es muy importante para el éxito de la entrega entre los desarrolladores y probadores de software.

La comunicación entre los desarrolladores y probadores no es suficientemente eficaz a través de una comunicación escrita y que cambia dependiendo del tipo de tareas y la experiencia de los evaluadores

Una serie de tendencias recientes se han centrado en la necesidad de transformar a las organizaciones ofrecer software continuamente entre el desarrollo y las pruebas es necesaria para garantizar que los errores se detectan y solucionan tan pronto como sea posible.

Para desarrollar software de alta calidad en equipos distribuidos, es imprescindible utilizar métodos y herramientas de pruebas de software de forma eficaz y eficiente.

Las actuales prácticas de pruebas de software están lejos de ser satisfactorias, como se indica en diversos estudios sobre software testing métodos y herramientas para el ensayo y aseguramiento de la calidad en el global desarrollo de software.

A framework for software process deployment and evaluation (Iván Ruiz-Rube, 2017)

Con el objetivo de minimizar el tiempo requerido para adaptar las herramientas al inicio de cada nuevo proyecto y reducir la complejidad de la construcción de mecanismos para la evaluación automatizada, se ha elaborado el Marco de Implementación y Evaluación de Procesos de Software (SPDEF).

El marco propuesto se basa en la aplicación de técnicas bien conocidas en Ingeniería de Software, tales como la ingeniería impulsada por modelos y la integración de información a través de datos abiertos vinculados. Comprende un método sistemático para el despliegue y la evaluación, una serie de modelos y relaciones entre modelos, y algunas herramientas de software.

El despliegue automatizado de la metodología OpenUP se prueba a través de la aplicación del framework SPDEF y herramientas de soporte para permitir la evaluación de calidad automatizada de proyectos de desarrollo o mantenimiento de software.

El modelado de procesos se incluye en la definición de proceso organizacional área de proceso de madurez de la capacidad Modelo de Integración (CMMI), mientras que en ISO / IEC 12207, que se incluye en el Proceso de Mejora grupo.

El marco propuesto para el despliegue y la evaluación de proceso de software está abierto a extensiones adicionales.
Este marco incluye dos modelos de nivel PIM: SWPM para la definición de los productos de trabajo, y SPCM para el control del proyecto. Ambos modelos permiten el enriquecimiento de los elementos de los modelos de procesos de software SPEM para su posterior implementación en herramientas de apoyo.

Sin embargo, hay otros aspectos del proceso de software, que sólo son tratadas vagamente en la norma SPEM, tales como gestión de la configuración o gestión de personas. Diseño de modelos PIM y PSM para estos aspectos y el posterior despliegue en las herramientas adecuadas, así como la definición de las relaciones entre los distintos PIM, se proponen como futuras líneas de investigación. Por otra parte, los modelos PSM no son compatibles con todas las capacidades ofrecidas por los diferentes tipos de herramientas, sino que se utiliza sólo los más comúnmente. En el futuro, estos modelos se pueden ampliar con otras características.

Hacia un Modelo de Madurez para apoyar el Desarrollo de Software Dirigido por Modelos (Valenzuela & Pavlich-Mariscal, 2014)

En este artículo se analiza el escenario actual en cuanto a los productos de software, se presentan una demanda creciente de productos y servicios de software con mayor exigencia en calidad y plazos de entrega. La práctica e investigación de la ingeniería de software en mejora de procesos, dieron origen a modelos de madurez y capacidad, dirigidos a incrementar la calidad y la productividad del desarrollo de software. Esto se denomina Desarrollo Dirigido por Modelos (MDD).

La idea principal de esta investigación es la propuesta de modelo de madurez y capacidad, realizando a: un modelo ágil y específico para MDD, aplicable en pequeñas organizaciones de desarrollo de software.

Se propone como métrica para evaluar y calificar los procesos, a través de metas definidas que para cumplirse requieren seguir un conjunto de buenas prácticas.
El modelo propuesto parte de los grandes estándares y se conjuga en dos partes principales:

1) Un modelo de procesos de referencia con niveles incrementales de madurez y capacidad;
2) Un método de evaluación y diagnóstico.

La estrategia de adaptación hacia pequeñas organizaciones es aligerar el modelo. Se reduce el número de elementos dentro de los dos modelos base y se adecuan y complementan los elementos seleccionados, orientándolos hacia una forma más ágil y pertinente con MDD.

La aplicación del modelo fue hecha de forma parcial, a pesar que inicialmente hubo motivación y apoyo por parte de la dirección del área de informática, las presiones y urgencias de las entregas y de los compromisos dio una baja prioridad a las acciones del plan de mejora de procesos. De todas maneras, aunque pequeños, hubo avances en ciertos grupos de trabajo en metas específicas.

El modelo es aplicable a la gran mayoría de pequeñas y muy pequeñas empresas de desarrollo de software, incluyendo las que cuentan con experiencia en metodologías ágiles orientadas al código, pues MDD comprende también modelos textuales, tal como los lenguajes específicos de dominio DSL.

1.6 Análisis de trabajos relacionados

Una serie de tendencias recientes se han centrado en la necesidad de transformar a las organizaciones, las pruebas son necesarias para garantizar que los errores se detecten y

solucionen tan pronto como sea posible. Por lo cual es imprescindible utilizar metodologías de calidad y herramientas de pruebas de software que puedan ayudar a disminuir los errores en la liberación del software. Ya que los diversos estudios realizados acerca del costo del testing indican que alrededor del 40 al 50 por ciento del total de los gastos asignados a un proyecto se gasta en las pruebas y la fase de integración (Christa & Suma, An analysis on the significance of ticket analytics and defect analysis from software quality perspective, 2016)., esto sin mencionar que entre más tiempo tarden en detectarse las fallas en el software el costo se incrementara aún más.

Para apoyar a la empresa MBN para que sus productos o desarrollos cuenten con los estándares de calidad que los clientes necesitan, se propone la realización de un framework para desarrollar pruebas unitarias, de integración y funcionalidad soportado de herramientas open source tales como JUnit y Selenium.

En la ***Tabla 1. Análisis comparativo de frameworks***, se presenta una comparación entre los frameworks ya existentes.

Tabla 1. Análisis comparativo de frameworks

Características	**Framework**	
	A framework for software process deployment and evaluation	Hacia un Modelo de Madurez para apoyar el Desarrollo de Software Dirigido por Modelos
• Estándares de calidad	• CMMI • ISO/IEC 12207	• CMMI • ISO/IEC 12207
• Métodos de desarrollo	• OpenUP	• Agiles
• Diseñado para PYMES	• PYMES	• PYMES
• Muestra proceso	• Muestra proceso	• Muestra proceso

1.7 Conclusión

Las pruebas de software son una herramienta indispensable en el desarrollo del software, ya que a través de ellas se identifican errores en etapas tempranas del ciclo de vida de software, lo cual beneficia a la calidad del producto terminado y se logra la satisfacción del cliente. Existen diversos frameworks, pero no se adecuan a los estándares que la empresa MBN maneja, por lo cual es importante la propuesta de un framework que se adapten al CMMI y MoProSoft y que ayude a disminuir los defectos en los productos de software.

1.8 Estructura de la tesis

La presente tesis se organiza de la siguiente manera: en el Capítulo 1 Introducción se presenta los preliminares de nuestro trabajo como primer punto el Contexto del problema el cual contiene una breve introducción a la empresa MBN, el planteamiento del problema que contiene los objetivos, alcances y limitaciones del problema, como tercer punto presentamos la propuesta de solución, como cuarto punto la pregunta de investigación, como quinto punto los trabajos relacionados y como sexto punto un análisis de los trabajos relacionados y finalmente el punto siete una conclusión sobre los puntos anteriores.

En el Capítulo 2 Pruebas de software se presenta el marco teórico, conceptos de los tipos de prueba, modelos de calidad, de modelos de desarrollo y de herramientas que apoyaran a la optimización de las pruebas, y finalmente una conclusión sobre los puntos anteriores.

En el Capítulo 3 Framework para pruebas de software se presenta la propuesta de del framework para pruebas de software el primero es aplicando metodología ágil(Scrum) y el segundo aplicando metodología clásica(Cascada) y, así como los formatos que deberán ser llenados una vez seleccionado el tipo de framework de acuerdo a las necesidades del sistema.

En el Capítulo 4 Aplicación y resultados del Framework para pruebas de software se presenta los resultados de la aplicación del framework en dos metodologías de desarrollo

ágil(Scrum) y clásica(cascada), así como algunos de los formatos utilizados para la ejecución de las pruebas (unitarias, de integración y funcionales)

En el Capítulo 5 Conclusiones se presenta las conclusiones finales del análisis realizado en el apartado anterior Capitulo 4 a las que se llegó una vez que se aplicaron las pruebas mediante las dos aplicaciones propuestas, así como el trabajo futuro que permitirá darle continuidad a este proyecto.

Capítulo 2

Pruebas de software

2 Pruebas de software

En este Capítulo se presenta algunos conceptos importantes sobre frameworks (marco de trabajo), pruebas de software, se describen los modelos de calidad en específico CMMI y Moprosoft, además se hace un análisis de las herramientas que serán utilizadas para la optimización de los diferentes tipos de pruebas a ejecutar.

2.1 Framework

Es un esquema (un esqueleto, un patrón) para el desarrollo y/o la implementación de una aplicación (Sánchez, 2006)

En (Gamma et al, 1995) se cita la definición de Christopher Alexander sobre patrones: "cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, así como la solución a ese problema, de tal modo que se pueda aplicar esta solución un millón de veces, sin hacer lo mismo dos veces". (Campo, 2009)

Un Patrón de Diseño (design pattern) es una solución repetible a un problema recurrente en el diseño de software. Esta solución no es un diseño terminado que puede traducirse directamente a código, sino más bien una descripción sobre cómo resolver el problema, la cual puede ser utilizada en diversas situaciones. Los patrones de diseño reflejan todo el rediseño y recodificación que los desarrolladores han ido haciendo a medida que intentaban conseguir mayor reutilización y flexibilidad en su software. (Campo, 2009)

2.2 Pruebas de software

Las pruebas de software intentan demostrar que un programa hace lo que se intenta que haga, así como descubrir defectos en el programa antes de usarlo. El proceso de prueba tiene dos metas distintas:

1. Demostrar al desarrollador y al cliente que el software cumple con los requerimientos. Para el software personalizado, esto significa que en el documento de requerimientos debe haber, por lo menos, una prueba por cada requerimiento. Para los productos de software genérico, esto quiere decir que tiene que haber pruebas para todas las características del sistema, junto con combinaciones de dichas características que se incorporarán en la liberación del producto.

2. Encontrar situaciones donde el comportamiento del software sea incorrecto, indeseable o no esté de acuerdo con su especificación. Tales situaciones son consecuencia de defectos del software. La prueba de defectos tiene la finalidad de erradicar el comportamiento indeseable del sistema, como caídas del sistema, interacciones indeseadas con otros sistemas, cálculos incorrectos y corrupción de datos.

La primera meta conduce a la prueba de validación; en ella, se espera que el sistema se desempeñe de manera correcta mediante un conjunto dado de casos de prueba, que refleje el uso previsto del sistema. La segunda meta se orienta a pruebas de defectos, donde los casos de prueba se diseñan para presentar los defectos. (Sommerville, 2011)

La Prueba o Testeo de Software, es un procedimiento llevado a cabo para identificar posibles fallos de implementación, calidad, o usabilidad de un programa. Básicamente es una fase en el desarrollo de software cuyo objetivo es probar las funcionalidades de la aplicación construida. La prueba de software es un proceso que corre en paralelo al proceso de desarrollo de software, y que se realiza por el convencimiento de que todo sistema debe ser inspeccionado o probado con el objetivo de establecer si el nivel de calidad requerido es alcanzado. Es un elemento que a menudo se refiere como

verificación y validación. En (Pressman, 2005) se plantea que "la verificación se refiere al conjunto de actividades que aseguran que el software implementa correctamente una función específica. La validación se refiere a un conjunto diferente de actividades que aseguran que el software construido se ajusta a los requisitos del cliente". (Argota Vega, 2007)

2.2.1 Evolución de las pruebas de software

La separación del proceso de "debugging" del "Testing", fue una idea introducida por Glenford J. Myers in 1979. Myers enfocaba las pruebas como algo destructivo ("una prueba exitosa es aquella que descubre un Bug. Aquí se refleja claramente el deseo de la comunidad de ingenieros de software de separar las actividades fundamentales del desarrollo, debugging y verificación). Dave Gelperin y William C. Hetzel en 1988 clasificaron la evolución de las fases y objetivos de las pruebas en las siguientes etapas:

- Antes -1956 - Orientadas a la depuración
- 1957–1978 - Orientadas a la demostración
- 1979–1982 - Orientadas a la destrucción
- 1983–1987 - Orientadas a la evaluación
- 1988–2000 - Orientadas a la prevención (De Freitas Vieira, Benavidez Torres, & Irumbe, 2012)

2.2.2 Ciclo de vida de las pruebas

La ejecución de pruebas de un sistema involucra las etapas que a continuación se detallan en la ***Figura 1. Ciclo de vida de las pruebas***:

- Planificación de pruebas.
- Diseño y construcción de los Casos de Prueba.
- Preparación del ambiente de pruebas y generación de datos de prueba.

- Ejecución de las pruebas.
- Registro de errores encontrados.
- Registración de resultados obtenidos.
- Depuración de los errores.
- Informe de los resultados obtenidos.

En el marco del proceso definido para el área, las etapas mencionadas forman el ciclo de vida de las pruebas el cual se integrará al proceso de desarrollo de la aplicación:

- Durante la fase de Elaboración se define el Plan de Pruebas.
- Durante la fase de Elaboración se definen los Casos de Prueba.
- Durante la fase de Construcción, se construyen los Casos de Prueba y se genere las bases de datos de prueba.
- Posteriormente se ejecuten todos los scripts y programas de pruebas anteriormente desarrollados y se realice un adecuado seguimiento

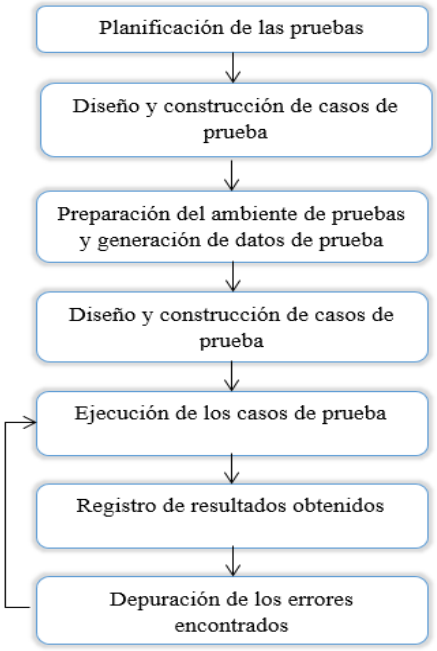


Figura 1.Ciclo de vida de las pruebas

2.2.3 Proceso de prueba

El proceso de prueba se compone de los siguientes pasos:

1. El proceso comienza con la generación del plan de pruebas por parte del líder del proyecto en la fase de elaboración. Dicho plan se generará una vez aprobado el plan de proyecto para el área al cual pertenece el sistema a probar.
2. Luego en la fase de construcción, el analista de sistemas del proyecto diseñará las pruebas basándose en la documentación y conocimiento del software a probar, esto dará como resultado tantos documentos de casos de prueba como testers sean designados. Esta actividad dependerá de la situación:

- Si el sistema está en producción y se prueba una nueva versión o actualización del mismo, en el caso en que exista, se utilizará el documento general de casos de prueba
- En la prueba del sistema o aplicación existirá un documento de casos de prueba por tester, en éste el analista de sistemas completará el ambiente de prueba a utilizar, cuáles serán los datos de entrada y los resultados esperados

- Si se está probando un sistema por primera vez, a parte de los documentos de casos de prueba que se generen por tester, se deberá crear un documento general de casos de prueba que reúna todos los eventos de prueba, para reutilizar en futuras pruebas.

- Si se está probando una nueva operación, se adicionarán tantos casos de prueba como sean necesarios al final del documento general de casos de prueba, donde se especificará la situación por la cual se generaron dichos casos.

- Si se está probando (un script ya sea de consulta, actualización, procedimiento almacenado, etc.) se deberá generar un documento de caso de prueba que contemple la funcionalidad e impacto del mismo en el sistema.

1. Una vez que el documento de casos de prueba fue aprobado, inicialmente por el líder del proyecto, éste pasará al tester. El tester previamente verificará que el ambiente de prueba se cumpla y ejecutará los casos con los datos de prueba proporcionados, documentando los resultados obtenidos y conclusiones inferidas en el mismo documento de casos de prueba.

2. A partir de los resultados obtenidos, y en el caso de que existieran errores el analista de sistemas compilará los errores registrados en los distintos documentos de casos de prueba en un único reporte de errores, en éste se dejará constancia de todos los defectos, errores y fallos detectados, este artefacto es de características técnicas, dirigido principalmente al implementador con el objetivo de que se corrijan los errores. El reporte de errores estará asociado a varios documentos de casos de prueba, será actualizado por el implementador en los casos en que corresponda.

- En los sistemas de código cerrado el reporte de errores también será generado por el Analista de Sistemas con el objetivo de enviar al SIU o a la entidad que corresponda y si existiera una solución (proporcionada por el SIU, etc.) será registrada por él en el mismo documento.

- En los sistemas o aplicaciones de código abierto, a partir de la detección se realizará la depuración (localización y corrección de defectos si correspondiere).

1. La depuración estará a cargo del Implementador, el cual puede o no corregir los errores. Es su responsabilidad completar la acción tomada en el documento reporte de errores, columna acción tomada.

2. Si se corrige un error, se debe volver a probar el software para comprobar que el problema este resuelto y cumple con la funcionalidad requerida, en este caso el Reporte de errores completo y el producto a evaluar volverá al analista de sistemas quien definirá un nuevo documento de casos de prueba, en el que se menciona si se trata de una ejecución o re-ejecución, este va al Tester quien en el caso en que

persistan los errores (los mismos o nuevos) los registrará en el mismo reporte de errores inicial, a partir de acá el proceso de repite para todas las situaciones en las que se continúen encontrando errores.

3. Una vez finalizadas las pruebas y depurados los errores encontrados el líder del proyecto generará el informe de resultados obtenidos el cual va dirigido, a la gerencia colaborativa, gerencia general y cliente según corresponda, el mismo contiene un resumen de los resultados de las pruebas ejecutadas, aporta una evaluación del software basada en dichos resultados y una descripción del análisis de errores encontrados el cual servirá para realizar predicciones de la fiabilidad del software, detectar las causas más habituales de error y mejorar los procesos de desarrollo y prueba.

4. Una vez revisado volverá al líder de proyecto para que este lo almacene en el directorio correspondiente del espacio de trabajo. (Díaz Vivar, Miranda, & Molina)

2.2.4 Defectos del software y sus causas

A los defectos del software se les conoce comúnmente como bug (bicho), y corresponde a un error, imperfecto o falla de una aplicación para computador que puede causar un resultado no deseado o incumplimiento de un requerimiento, ver ***Figura 2. Defectos del software.***

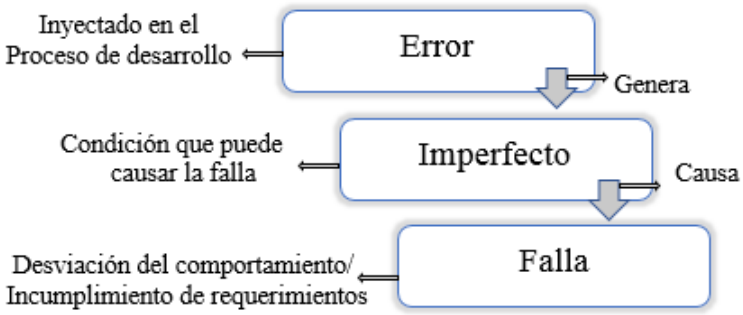


Figura 2. Defectos del software

Un defecto es mucho más que un error de la aplicación, realmente un defecto abarca todo el proceso productivo del desarrollo de software y los artefactos que este produce, es decir que los defectos se inyectan en las diferentes etapas del ciclo de vida del desarrollo de aplicaciones y no solo en la fase de desarrollo. Un defecto puede ser ingresado por ejemplo en un manual de usuario que especifica una funcionalidad no disponible en la aplicación, esto finalmente sigue siendo un defecto de la aplicación si la miramos no solo como el programa que usa el cliente sino como todo un paquete que incluye varios componentes, entre ellos la documentación. La siguiente lista enumera algunas de las causas que generan defectos en las aplicaciones:

- Problemas de comunicación Cliente - Proveedor
- Definición incorrecta o ausencia de los requerimientos
- Desviación deliberada de los requerimientos (por presión del tiempo, avances sin autorización)
- Errores de Diseño
- Errores de codificación
- Incumplimiento de estándares
- Pruebas insuficientes
- Errores de documentación

La mayoría de los defectos se producen en etapas tempranas del proceso de desarrollo, por lo que en estas etapas debe existir una estructura robusta de control y no solo en la verificación y validación del producto final. Por ejemplo, en la fase de Requerimientos se generan el 56% o más de los errores que se inyectan durante todo el ciclo de vida de desarrollo ***Tabla 2. Defectos ingresados por etapa***. Sin embargo, en la mayoría de las organizaciones no existen actividades de testing en esta etapa. Las actividades de calidad en etapas tempranas se ven recompensadas en grandes beneficios para las empresas de desarrollo de software, ya que los costos de corregir defectos son más altos entre más tarde son detectados. (Coronel , 2013)

Tabla 2. Defectos ingresados por etapa

FASE	% DEFECTOS
Requerimiento	56%
Diseño	27%
Codificación	7%
Otros	10%

2.2.5 Tipos de pruebas

Pruebas: unitarias e integración

- Unitarias: permite verificar la funcionalidad y estructura de cada componente individualmente del sistema una vez que ha sido codificado.

- Integración: permite verificar el correcto ensamblaje entre los distintos módulos que componen el sistema desarrollado.

Pruebas de sistema: estas pruebas buscan diferencias entre la solución desarrollada y los requerimientos, con el fin de identificar errores que se puedan generar entre la especificación funcional y el diseño del sistema. Las pruebas de sistema pueden ser:

- Carga: valida aquellos volúmenes de datos máximos especificados en los requerimientos no Funcionales.

- Volumen: esta prueba somete el software a grandes cantidades de datos para determinar si se alcanzan límites que causen la falla del software.

- Estrés: valida aquellos volúmenes de datos máximos que resiste el sistema antes de comenzar con errores.

- Robustez: valida si el sistema se mantiene estable y consistente después de circunstancias adversas.

- Concurrencia: valida la capacidad del sistema de atender múltiples solicitudes de parte de los usuarios que acceden a un mismo recurso.

- Interfaz de usuario: permite verificar que la navegación a través de los elementos que se están probando, reflejen las funciones del negocio y los requerimientos funcionales.

- Recuperación a fallas: estas pruebas aseguran que el que el software pueda recuperarse a fallas de hardware, software o mal funcionamiento de la red sin pérdida de datos o de integridad de los datos.

- Rendimiento: permite validar si la aplicación cumple los criterios de tiempos de respuesta establecidos.

- Seguridad: verifica el cumplimiento de las políticas de seguridad acordadas para el sistema.

- Integridad de las bases de datos: consiste en asegurar que los métodos y procesos de acceso a la base de datos funcionan correctamente y sin corromper datos.

- Interoperabilidad: esta prueba permite verificar todos los artefactos de la solución desarrollada, su arquitectura base, los protocolos de la solución, las interfaces y los módulos del sistema, funcionando en forma conjunta.

- Desempeño: este tipo de prueba es un aspecto fundamental en una aplicación, ya que, si ésta no responde en el debido tiempo, se pueden perder clientes, o dañar la imagen ante los usuarios.

- Configuración: establece y mantiene la integridad de los productos de software a través del ciclo de vida del proceso del mismo.

Prueba funcional: es un proceso para procurar encontrar discrepancias entre el programa y la especificación funcional. Los tipos de pruebas funcionales pueden ser:

- Caja Negra: estas pruebas permiten obtener conjuntos de condiciones de entrada que ejecutan todos los requisitos funcionales de un programa.

- Ciclo de Negocio: esta prueba tiene por objeto garantizar que el proceso de negocio esta adecuadamente soportado por el software desarrollado y que éste dispone de la funcionalidad adecuada para ejecutar todas las tareas incorporadas en el proceso de negocio.

- Usabilidad: esta prueba permite encontrar problemas de factores humanos, o usabilidad.

- Instalación: esta prueba permite verificar la instalación y desinstalación de la aplicación en diferentes entornos de hardware y software

Prueba aceptación: es la prueba final basada en las especificaciones del usuario o basada en el uso del programa por el usuario final luego de un periodo de tiempo

Prueba de regresión: valida que el sistema mantenga su correcta funcionalidad debido a la incorporación de un ajuste, corrección o nuevo requerimiento. Es una prueba funcional y técnica que valida que el sistema siga funcionando perfectamente después de que las correcciones sean aplicadas. (Ministerio de comunicaciones, 2008)

Descripción de la Prueba:

- Particionar los módulos en pruebas en unidades lógicas fáciles de probar.
- Por cada unidad hay que definir los casos de prueba (pruebas de caja blanca).

- Para esto los casos de prueba deben diseñarse de forma tal que se recorran todos los caminos de ejecución posibles dentro del código bajo prueba; por lo tanto, el diseñador debe construirlos con acceso al código fuente de la unidad a probar.
- Los aspectos a considerar son los siguientes: Rutinas de excepción, Rutinas de error, Manejo de parámetros, Validaciones, Valores válidos, Valores límites, Rangos, Mensajes posibles.

Técnica:

- Comparar el resultado esperado con el resultado obtenido.
- Si existen errores, reportarlos. (Abad Londoño, 2005)

Para que una prueba unitaria sea "buena" se deben cumplir los siguientes requisitos:

- Que sea automatizable: no se debe de requerir de intervención manual. Esto es especialmente útil para integración continúa.
- Que sea completa: deben cubrir la mayor cantidad de código.
- Que sea reutilizable: deben poder ejecutarse tantas veces como sea necesario. También es útil para integración continua.
- Que sea Independiente: La ejecución de una prueba no debe afectar a la ejecución de otra.
- Que sea profesional: las pruebas deben ser escrita con la misma profesionalidad que se escribe el código, por lo que deben estar documentadas y bien diseñadas.
- Aunque estos requisitos no tienen que ser cumplidos completamente, se recomienda seguirlos o de lo contrario las pruebas pierden parte de su objetivo. El objetivo de las pruebas unitarias es aislar cada parte del programa y mostrar que las partes individuales son correctas. (Cristian, 2003)

2.2.6 Riesgos

La gestión de riesgos conlleva una valoración de los riesgos más comunes del proyecto, definiendo un plan de acción por si su ocurrencia causara problemas y monitorizando los proyectos para obtener avisos cuando se entre en una zona de riesgo, de modo que se tomen las medidas oportunas.

El cálculo del riesgo viene dado por la combinación de los resultados del análisis del impacto del negocio y de la valoración de la probabilidad de fallo. Esta combinación permite categorizar las funciones de negocio en función del riesgo como se ve en la ***Tabla 3. Cálculo de las categorías de riesgos:*** (Inteco, 2008)

Tabla 3. Calculo de las categorías de riesgos

CRITERIO	RESULTADOS		
	Poco probable	Probable	Muy probable
Impacto alto	Riesgo medio	Riesgo alto	Riesgo alto
Impacto medio	Riesgo bajo	Riesgo medio	Riesgo medio
Impacto bajo	Riesgo bajo	Riesgo bajo	Riesgo medio

2.3 Modelos de calidad

En la actualidad existe una constante demanda de la industria por un mejor y más barato software, que debe entregarse en plazos cada vez más cortos. Por consiguiente, numerosas compañías de software han dirigido la atención hacia la mejora de procesos de software como una forma de aumentar la calidad de su software, reducir sus costos o acelerar los procesos de desarrollo. La mejora de procesos significa comprender los procesos existentes y cambiarlos para incrementar la calidad del producto o reducir los costos y el tiempo de desarrollo. (Sommerville, 2011)

2.3.1 CMMI

El objetivo global del CMMI es proporcionar un marco que comparta las mejores prácticas y aproximaciones de mejora de procesos de manera coherente, pero lo suficientemente

Flexible como para satisfacer las necesidades en constante evolución de la comunidad.

Áreas de proceso

1. Análisis causal y resolución (CAR).
2. Gestión de configuración (CM).
3. Análisis de decisiones y resolución (DAR).
4. Gestión integrada del proyecto + IPPD (IPM + IPPD)
5. Medición y análisis (MA).
6. Innovación y despliegue en la organización (OID).
7. Definición de procesos de la organización + IPPD (OPD + IPPD)
8. Enfoque en procesos de la organización (OPF).
9. Rendimiento del proceso de la organización (OPP).
10. Formación organizativa (OT).
11. Integración de producto (PI).
12. Monitorización y control del proyecto (PMC).
13. Planificación de proyecto (PP).
14. Aseguramiento de la calidad de proceso y de producto (PPQA).
15. Gestión cuantitativa de proyecto (QPM).
16. Desarrollo de requerimientos (RD).
17. Gestión de requerimientos (REQM).
18. Gestión de riesgos (RSKM).
19. Gestión de acuerdos con proveedores (SAM).
20. Solución técnica (TS).
21. Validación (VAL).
22. Verificación (VER).

Los niveles se utilizan en CMMI para describir un camino evolutivo recomendado para una organización que quiera mejorar los procesos que utiliza para desarrollar y mantener sus productos y servicios.
Estos niveles son un medio de predecir los resultados generales del siguiente proyecto que se acometa. Existen cinco niveles de madurez, numerados de 1 a 5.

Nivel de madurez 1: Inicial

En el nivel de madurez 1, los procesos son generalmente ad-hoc y caóticos. La organización generalmente no proporciona un entorno estable para dar soporte a los procesos.
El éxito en estas organizaciones depende de la competencia y heroicidad del personal de la organización y no del uso de procesos probados.

A pesar de este caos, las organizaciones de nivel de madurez 1 a menudo producen productos y servicios que funcionan; sin embargo, frecuentemente exceden sus presupuestos y no cumplen sus calendarios.

Las organizaciones de nivel de madurez 1 se caracterizan por una tendencia a comprometerse en exceso, a abandonar los procesos en tiempos de crisis y a una incapacidad para repetir sus éxitos.

Nivel de madurez 2: Gestionado
En el nivel de madurez 2, los proyectos de la organización han asegurado que los procesos se planifican y realizan de acuerdo a políticas; los proyectos emplean personal con habilidad que dispone de recursos adecuados para producir resultados controlados; involucran a las partes interesadas relevantes; se monitorizan, controlan y revisan; y se evalúan en cuanto a su adherencia a sus descripciones de proceso.

La disciplina de proceso reflejada por el nivel de madurez 2 ayuda a asegurar que las prácticas existentes se mantienen durante tiempos de estrés. Cuando estas prácticas están en su lugar, los proyectos se realizan y gestionan de acuerdo a sus planes documentados.
En el nivel de madurez 2, el estado de los productos de trabajo y la entrega de los servicios son visibles a la dirección en puntos definidos.

Se establecen compromisos entre las partes interesadas relevantes y se revisan, según sea necesario. Los productos de trabajo se controlan de forma apropiada.
Los productos de trabajo y servicios satisfacen sus descripciones de proceso especificadas, estándares y procedimientos.

Nivel de madurez 3: Definido

En el nivel de madurez 3, los procesos son bien caracterizados y comprendidos, y se describen en estándares, procedimientos, herramientas y métodos. El conjunto de procesos estándar de la organización, que es la base del nivel de madurez 3, se establece y mejora a lo largo del tiempo.

Estos procesos estándar se usan para establecer la consistencia en toda la organización.
Los proyectos establecen sus procesos definidos adaptando el conjunto de procesos estándar de la organización de acuerdo a las guías de adaptación (consultar el glosario para una definición de "conjunto de procesos estándar de la organización").
Una distinción crítica entre los niveles de madurez 2 y 3 es el alcance de los estándares, descripciones de proceso y procedimientos.

En el nivel de madurez 2, los estándares, descripciones de proceso y procedimientos pueden ser bastante diferentes en cada instancia específica de un proceso.
En el nivel de madurez 3, los estándares, descripciones de proceso y procedimientos para un proyecto se adaptan para adecuarse a un proyecto particular o unidad organizativa a partir del conjunto de procesos estándar de la organización y, por tanto, son más consistentes, exceptuando las diferencias permitidas por las guías de adaptación.

Otra distinción crítica es que en el nivel de madurez 3, los procesos normalmente se describen más rigurosamente que en el nivel de madurez 2. Un proceso definido establece claramente el propósito, entradas, criterios de entrada, actividades, roles, medidas, etapas de verificación, salidas y criterios de salida. En el nivel de madurez 3, los procesos se gestionan más proactivamente utilizando una comprensión de las interrelaciones de las actividades del proceso y las medidas detalladas del proceso, sus productos de trabajo y sus servicios.

En el nivel de madurez 3, la organización debe madurar más las áreas de proceso de nivel de madurez 2. Para lograr el nivel de madurez 3, se aplican las prácticas genéricas asociadas con la meta genérica 3 que no fueron tratadas en el nivel de madurez 2.

Nivel de madurez 4: Gestionado cuantitativamente

En el nivel de madurez 4, la organización y los proyectos establecen objetivos cuantitativos en cuanto al rendimiento de calidad y del proceso, y los utilizan como criterios en la gestión de los procesos.

Los objetivos cuantitativos se basan en las necesidades del cliente, usuarios finales, organización e implementadores del proceso.

El rendimiento de calidad y del proceso se comprende en términos estadísticos y se gestiona durante la vida de los procesos.

Para los subprocesos seleccionados, se recogen y analizan estadísticamente medidas detalladas de rendimiento del proceso.

Las medidas de rendimiento de calidad y del proceso se incorporan en el repositorio de medición de la organización para dar soporte a la toma de decisiones basada en hechos. Se identifican las causas especiales de variación y, donde sea apropiado, se corrigen las fuentes de las causas especiales para prevenir sus futuras ocurrencias.

Una distinción crítica entre los niveles de madurez 3 y 4 es la predictibilidad del rendimiento del proceso.

En el nivel de madurez 4, el rendimiento de los procesos se controla utilizando técnicas estadísticas y otras técnicas cuantitativas, y es predecible cuantitativamente.

En el nivel de madurez 3, los procesos normalmente sólo son predecibles

Nivel de madurez 5: En optimización

En el nivel de madurez 5, una organización mejora continuamente sus procesos basándose en una comprensión cuantitativa de las causas comunes de variación inherentes a los procesos.

El nivel de madurez 5 se centra en mejorar continuamente el rendimiento de procesos mediante mejoras incrementales e innovadoras de proceso y tecnológicas. Los objetivos cuantitativos de mejora de procesos para una organización se establecen, se revisan continuamente para reflejar el cambio a los objetivos del negocio, y se utilizan como criterios para gestionar la mejora de procesos.

Los efectos de las mejoras de procesos desplegadas se miden y evalúan frente a los objetivos cuantitativos de mejora de procesos.
Tanto los procesos definidos como el conjunto de procesos estándar de la organización son objeto de las actividades de mejora cuantitativa.
Una distinción crítica entre los niveles de madurez 4 y 5 es el tipo de variación del proceso tratado. En el nivel de madurez 4, la organización se preocupa por tratar las causas especiales de variación del proceso y por proporcionan predictibilidad estadística de los resultados.
Aunque los procesos pueden producir resultados predecibles, los resultados pueden ser insuficientes para alcanzar los objetivos establecidos.

En el nivel de madurez 5, la organización se interesa en tratar las causas comunes de variación del proceso y en cambiar el proceso (para cambiar la media de rendimiento del proceso o reducir la variación inherente del proceso experimentada) para mejorar el rendimiento del proceso y para alcanzar sus objetivos cuantitativos de mejora de procesos establecidos. (Beth Chrissis , Konrad , & Shrum, 2009)

2.3.2 Moprosoft

Es un modelo de procesos para la industria de software nacional de México promovido por la Secretaría de Economía, que fomenta la estandarización de la operación a través de la incorporación de las mejores prácticas en gestión e ingeniería de software. La adopción del modelo permite elevar la madurez de los procesos para las organizaciones que desarrollan y/o mantienen software para ofrecer servicios con calidad y alcanzar niveles internacionales de competitividad. (Ocampo Acosta & Correa Tapasco, 2011)

Estructura de Moprosoft

El modelo pretende apoyar a las organizaciones en la estandarización de sus prácticas, en la evaluación de su efectividad y en la integración de la mejora continua. Sintetiza las mejores prácticas en un conjunto pequeño de procesos que abarcan las responsabilidades asociadas a la estructura de una organización que son: la Alta Dirección, Gestión y Operación.

Moprosoft es un modelo integrado donde las salidas de un proceso están claramente dirigidas como entradas a otros; las prácticas de planeación, seguimiento y evaluación se incluyeron en todos los procesos de gestión y administración; por su parte los objetivos, los indicadores, las mediciones y las metas cuantitativas fueron incorporados de manera congruente y práctica en todos los procesos; las verificaciones, validaciones y pruebas están incluidas de manera explícita dentro de las actividades de los procesos; y existe una base de conocimientos que resguarda todos los documentos y productos generados. Veamos a continuación el propósito de los procesos de Moprosoft:

Categoría:

- Alta Dirección

Proceso:

- Gestión de Negocio

Propósito:

Establecer la razón de ser de la organización, sus objetivos y las condiciones para lograrlos, para lo cual es necesario considerar las necesidades de los clientes, así como evaluar los resultados para poder proponer cambios que permitan la mejora continua. Adicionalmente habilita a la organización para responder a un ambiente de cambio y a sus miembros para trabajar en función de los objetivos establecidos.

Categoría:

- Gestión

Proceso:

- Gestión de Procesos

Establecer los procesos de la organización, en función de los procesos requeridos identificados en el Plan Estratégico. Así como definir, planificar e implantar las actividades de mejora en los mismos

Categoría:

- Gestión

Proceso:

- Gestión de Proyectos

Asegurar que los proyectos contribuyan al cumplimiento de los objetivos y estrategias de la organización

Categoría:

- Gestión

Proceso:

- Gestión de Recursos

Conseguir y dotar a la organización de los recursos humanos, infraestructura, ambiente de trabajo y proveedores, así como crear y mantener la Base de Conocimiento de la organización. La finalidad es apoyar el cumplimiento de los objetivos del Plan Estratégico de la organización. Las actividades de este proceso se apoyan en tres subprocesos:

- Recursos humanos y ambiente de trabajo.
- Bienes, servicios e infraestructura.
- Conocimiento de la organización

Categoría:

- Operación

Proceso:

- Administración de Proyectos Específicos

Establecer y llevar a cabo sistemáticamente las actividades que permitan cumplir con los objetivos de un proyecto en tiempo y costo esperados.

Categoría:

- Operación

Proceso:

- Desarrollo y Mantenimiento de Software

Realización sistemática de las actividades de análisis, diseño, construcción, integración y pruebas de productos de software nuevos o modificados cumpliendo con los requerimientos especificados (Ventura Miranda & Peñaloza Baez, 2008).

2.4 Modelos de desarrollo

Ingeniería de software es la aplicación de enfoques sistemáticos y disciplinados al desarrollo de software, para esto se han creado modelos y metodologías para la correcta utilización del tiempo y recursos que una empresa o entidad disponen.

Los modelos de desarrollo de software ofrecen un marco de trabajo usado para controlar el proceso de desarrollo de sistemas de información, estos marcos de trabajo consisten en una filosofía de desarrollo de programas la cual debe de contar con las herramientas necesarias para la asistencia del proceso de desarrollo. (Pablo, s.f.)

2.4.1 Cascada

Este es el más básico de todos los modelos y ha servido como bloque de construcción para los demás paradigmas de ciclo de vida. Está basado en el ciclo convencional de una ingeniería y su visión es muy simple: el desarrollo de software se debe realizar siguiendo una secuencia de fases. Cada etapa tiene un conjunto de metas bien definidas y las

actividades dentro de cada una contribuyen a la satisfacción de metas de esa fase o quizás a una subsecuencia de metas de la misma. El arquetipo del ciclo de vida abarca las siguientes actividades:

1. **Ingeniería y Análisis del Sistema**: Debido a que el software es siempre parte de un sistema mayor, el trabajo comienza estableciendo los requisitos de todos los elementos del sistema y luego asignando algún subconjunto de estos requisitos al software.

2. **Análisis de los requisitos del software**: el proceso de recopilación de los requisitos se centra e intensifica especialmente en el software. El ingeniero de software debe comprender el ámbito de la información del software, así como la función, el rendimiento y las interfaces requeridas.

3. **Diseño**: el diseño del software se enfoca en cuatro atributos distintos del programa; la estructura de los datos, la arquitectura del software, el detalle procedimental y la caracterización de la interfaz. El proceso de diseño traduce los requisitos en una representación del software con la calidad requerida antes de que comience la codificación.

4. **Codificación**: el diseño debe traducirse en una forma legible para la máquina. Si el diseño se realiza de una manera detallada, la codificación puede realizarse mecánicamente.

5. **Prueba**: una vez que se ha generado el código comienza la prueba del programa. La prueba se centra en la lógica interna del software y en las funciones externas, realizando pruebas que aseguren que la entrada definida produce los resultados que realmente se requieren.

6. **Mantenimiento**: el software sufrirá cambios después de que se entrega al cliente. Los cambios ocurrirán debidos a que se haya encontrado errores, a que el software deba adaptarse a cambios del entorno externo (sistema operativo o dispositivos periféricos) o a que el cliente requiera ampliaciones funcionales o del rendimiento.

En el modelo vemos una ventaja evidente y radica en su sencillez, ya que sigue los pasos intuitivos necesarios a la hora de desarrollar el software. Pero el modelo se aplica en un contexto, así que debemos atender también a él y saber que:

- Los proyectos reales raramente siguen el flujo secuencial que propone el modelo. Siempre hay iteraciones y se crean problemas en la aplicación del paradigma.
- Normalmente, al principio, es difícil para el cliente establecer todos los requisitos explícitamente. El ciclo de vida clásico lo requiere y tiene dificultades en acomodar posibles incertidumbres que pueden existir al comienzo de muchos productos.
- El cliente debe tener paciencia. Hasta llegar a las etapas finales del proyecto no estará disponible una versión operativa del programa. Un error importante que no pueda ser detectado hasta que el programa esté funcionando, puede ser desastroso. (Ble, 2010-2013)

2.4.2 Scrum

Scrum es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos.

En Scrum se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales.

Scrum también se utiliza para resolver situaciones en que no se está entregando al cliente lo que necesita, cuando las entregas se alargan demasiado, los costes se disparan o la calidad no es aceptable, cuando se necesita capacidad de reacción ante la competencia,

cuando la moral de los equipos es baja y la rotación alta, cuando es necesario identificar y solucionar ineficiencias sistemáticamente o cuando se quiere trabajar utilizando un proceso especializado en el desarrollo de producto. (proyectos agiles.org, s.f.)

2.5 Herramientas para hacer pruebas de software

Actualmente el número de herramientas para pruebas de software disponibles, tanto de manera gratuita (open source) o herramientas comerciales (de pago) es muy amplio. A continuación, se mencionan algunas características de dos importantes herramientas open source que serán utilizadas para la optimización de las pruebas antes ya mencionadas.

2.5.1 JUnit

JUnit es un *framework* orientado a la realización de pruebas unitarias en Java. La idea en JUnit es crear una clase de prueba por cada clase que queramos probar. Esta clase de prueba se encarga de:

1. Instanciar clase a probar
2. Preparar el caso de prueba (fixture)
3. Invocar métodos sobre clase a probar
4. Verificar que los métodos entregan una salida adecuada con respecto a una entrada particular.

Una clase de pruebas hereda siempre de la clase Test Case y contiene un conjunto de métodos que corresponden a las pruebas que se van a realizar. En general, por cada método de la clase que queramos probar, vamos a tener un método en la clase de prueba. El método de la clase de prueba tiene el mismo nombre que el de la clase probada, pero con el prefijo test. También hay que notar que se recomienda que las clases de prueba se pongan en package con el mismo nombre pero que inicie por test (Cervantes, 2005)

2.5.2 Selenium

Selenium IDE es un plugin de Firefox que pertenece al juego de herramientas SeleniumHQ, permite realizar juegos de pruebas sobre aplicaciones web.

Para ello realiza la grabación de la acción seleccionada (navegación por una página) en un "script", el cual se puede editar y parametrizar para adaptarse a los diferentes casos, y lo que es más importante su ejecución se puede repetir tantas veces como se quiera.

El principal objetivo de este plugin es crear pruebas funcionales, aunque no se puede pasar por alto que este tipo de herramientas permiten automatizar tareas que requieren un cierto "procesamiento" mental básico:

- Rellenar formularios (autenticación o cualquier otro tipo)
- Navegación web
- Acciones de gestión (CRUD de comentarios blog / correos / noticias / etc.)

Esta herramienta permite al desarrollador web ahorrarse mucho esfuerzo (mucho esfuerzo = muchas horas) cada vez que se resuelve alguna incidencia o se genera una versión nueva. Para ello permite automatizar la realización de las pruebas ya sean o bien pruebas específicas (una acción en particular) o bien juegos de pruebas (un conjunto de acciones).

Características:

- Facilidad de registro y ejecución de los test.
- Referencia a objetos DOM en base al ID, nombre o a través de XPath.
- Autocompletado para todos los comandos.
- Las acciones pueden ser ejecutadas paso a paso.
- Herramientas de depuración y puntos de ruptura (breakpoints).
- Los test pueden ser almacenados como HTML y scripts Ruby, entre otros formatos.
- Soporte para Selenium user-extensions.js.
- Ejecución en varios navegadores.
- Uso de diferentes API's en diferentes lenguajes (PHP, Ruby, JAVA, JavaScript, etc.). (Madrid, 2008)

2.6 Conclusión

Existen varios tipos de pruebas de software que son parte clave en la calidad de un producto ya que cuidan la construcción a lo largo del ciclo de vida del proyecto.

Las pruebas de software verifican que se cumplan los requerimientos planteados por parte del cliente, detectan errores para que sean corregidos, evitan perdidas tanto económicas como de tiempo.

Actualmente las pruebas de software pueden ser apoyadas por diversas metodologías y herramientas diseñadas para mejorar la calidad del producto, por lo que es de suma importancia analizar las características de estas para poder identificar cual es la metodología y herramienta más óptima con el fin de facilitar el proceso y la optimización de pruebas.

Capítulo 3

Framework para pruebas de software

3 *Framework* para pruebas de software

En este Capítulo se desarrolla un *Framework* para pruebas de software, tomando como base fundamental dos estándares de calidad CMMI y MOPROSOFT, y dos metodologías de desarrollo clásicas y agiles.

De acuerdo a los analizado en los capítulos anteriores deben realizar pruebas en cualquier sistema una vez que se empiezan a plantear los requerimientos, desde las primeras fases de desarrollo empiezan a surgir los errores.
Es importante empezar a detectarlos y a corregirlos, entre más avanzado este el desarrollo de un sistema y no se cuente con un plan de pruebas este sistema tiende a ser de baja calidad, ya que muchas de las veces no se cumplen los requerimientos especificados por el cliente, o en su defecto tienden a ser un fracaso, por lo cual el cliente queda insatisfecho provocando que haya pérdidas de tiempo, recurso y prestigio para la empresa, entre otras.

El *framework* es una estructura que dará soporte para realizar pruebas de software, cada uno de sus componentes son alineados a los estándares de calidad CMMI y Moprosoft. Sin embargo, aprender a hacer pruebas es difícil debido a que, primero se conoce muy poco acerca de los conceptos de pruebas, y segundo, hacer pruebas es un proceso complejo y aún más abstracto que la programación misma (Velázquez, s.f.)

Como se mencionó anteriormente MBN tiene diversas unidades estratégicas de negocios, de acuerdo a la norma en la cual está certificada Moprosoft entre las cuáles se encuentra DMS - "Fábrica de Software" que desarrolla proyectos de Software a la medida para diversos clientes. Bajo CMMI Dev 2 los proyectos de desarrollo de software se realizan conforme a una gestión disciplinada a través de políticas organizacionales, y en base a procesos que planifican, monitorean y ejecutan el desarrollo de proyectos de software.

Es necesario realizar para el área de desarrollo del software, un proceso de pruebas de software de manera metódica para: i) alinearse a Moprosoft y CMMI Dev 2, ii) ejecutar de manera eficiente los proyectos de desarrollo manteniendo los márgenes de operación planeados en la estimación de cada proyecto, y iii) lograr la satisfacción de los clientes en los proyectos desarrollados.

Para lograr estos objetivos se consideró la definición de un proceso detallado para el área de pruebas de software (propuesta de framework para pruebas de software bajo los estándares de calidad CMMI y Moprosoft tomando como base dos metodologías de desarrollo (clásicas y agiles)), el desarrollo de plantillas para la documentación y control de las pruebas y el uso de herramientas de software que ayuden al proceso de ejecución de dicho proceso.

3.1 *Framework* aplicando metodologías clásicas

Como se mencionó en la sección anterior el proceso de ejecución para pruebas será tomando como base metodologías clásicas, esto debido a que nos proporciona un orden de desarrollo en el proyecto, además de que sus objetivos están claramente definidos y muy constantes.

Los elementos del Plan de pruebas aplicando metodologías clásicas contendrá los siguientes elementos como se muestra en la Figura ***3. Framework aplicando metodologías clásicas:***

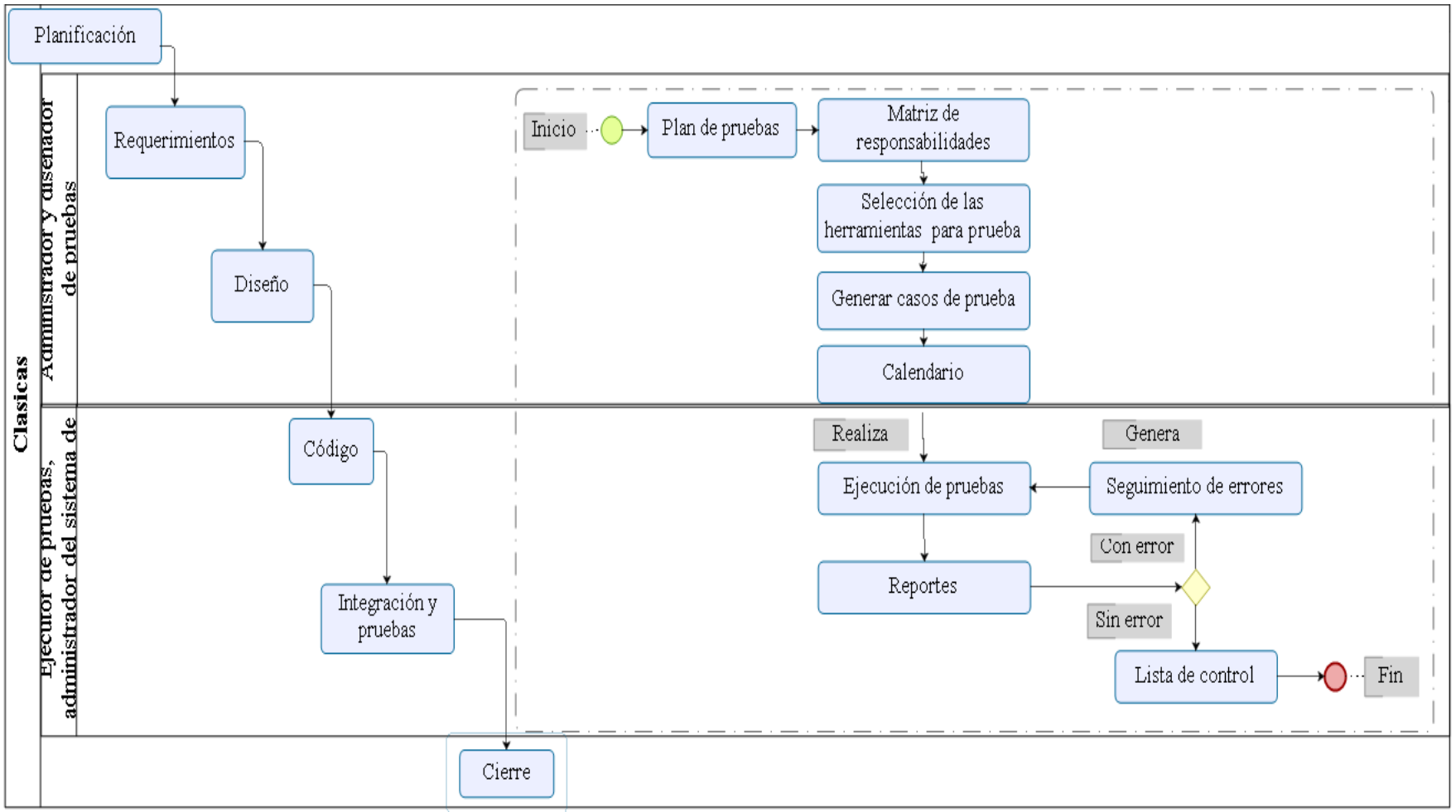


Figura 3. Framework aplicando metodologías clásicas

3.1.1 Matriz de responsabilidades

Este formato tiene como finalidad comunicar a cada integrante del equipo de pruebas las responsabilidades y actividades que deben realizar de acuerdo a su rol asignado.
Especifica el recurso humano(nombre), rol y responsabilidades que se deben realizar en el proyecto de pruebas.

En este componente del plan de pruebas se realiza una selección del personal adecuado como se muestra en la ***Figura 4. Formato Matriz de Responsabilidades***, para la ejecución de pruebas, se realiza una asignación de las actividades a realizar según sea el rol de cada persona.

1. Logotipo de la empresa
2. Nombre de la empresa
3. Nombre del proyecto
4. Número del recurso humano
5. Nombre y apellido del responsable
6. Asignación de un responsable para cada acción definida
7. Asignación de tareas o actividades a un rol para cumplir determinado objetivo

1	**MATRIZ DE RESPONSABILIDADES**		
Nombre de la empresa: 2	**Nombre del proyecto: “ 3 ”**		
Recursos humanos		**Rol**	**Responsabilidades**
Número	**Nombre**		
4	5	6	7

Figura 4. Formato Matriz de Responsabilidades

3.2 Selección de herramientas

Con el fin de automatizar la calidad del software, es necesario utilizar herramientas de pruebas de software, si bien no reemplazan las pruebas manuales, se agilizan los tiempos de desarrollo del producto, por lo cual se realizaron tanto análisis como selección de herramientas ver ***Figura 5. Formato Selección de Herramientas para pruebas.***

1. Logotipo de la empresa
2. Nombre de la empresa
3. Nombre del proyecto
4. Nombre del software para realizar las pruebas automatizadas
5. Número del tipo de prueba a ejecutar
6. Tipo de Licencia
7. Lenguaje
8. Requisitos
9. Características
10. Desventajas

1	**SELECCIÓN DE HERRAMIENTAS PARA PRUEBAS**					
Nombre de la empresa: 2	**Nombre del proyecto: “** 3 **”**					
Nombre de la herramienta	**Tipo de prueba**	**Licencia**	**Lenguaje**	**Requisitos**	**Características**	**Desventajas**
4	5	6	7	8	9	10

Figura 5. Formato Selección de Herramientas para Pruebas

3.2.1 Caso de prueba

El propósito fundamental de este caso de prueba es validar que efectivamente los requerimientos son satisfactorios por lo cual es importante que haya un caso de prueba para verificar el cumplimiento de cada requerimiento ver ***Figura 6. Formato Caso de Prueba***

1. Logotipo de la empresa
2. Nombre de la empresa
3. Nombre del tipo de prueba
4. Nombre del proyecto
5. Código o identificador de la prueba
6. Fecha en la que se realiza la prueba
7. Nombre ejecutor de pruebas
8. Estado de la prueba que ya sea en proceso, o que ya se realizó y aún no ha sido aprobada
9. Estado en el cual ya fue aceptado o aprobado el resultado de a prueba
10. Nombre del caso de uso de la prueba
11. Descripción de lo que se probara (escenarios)
12. Condición(es) para que se ejecute el caso de prueba
13. Se especifica cada valor o dato de entrada que se requiere para la ejecución del caso de prueba
14. Aciertos y errores de las pruebas realizadas
15. Descripción del resultado que se espera de la ejecución de los casos de prueba con los datos de entrada
16. Casos en los que puede llegar a fallar la prueba
17. Observaciones relacionadas con la ejecución de los casos de prueba realizados y los resultados obtenidos

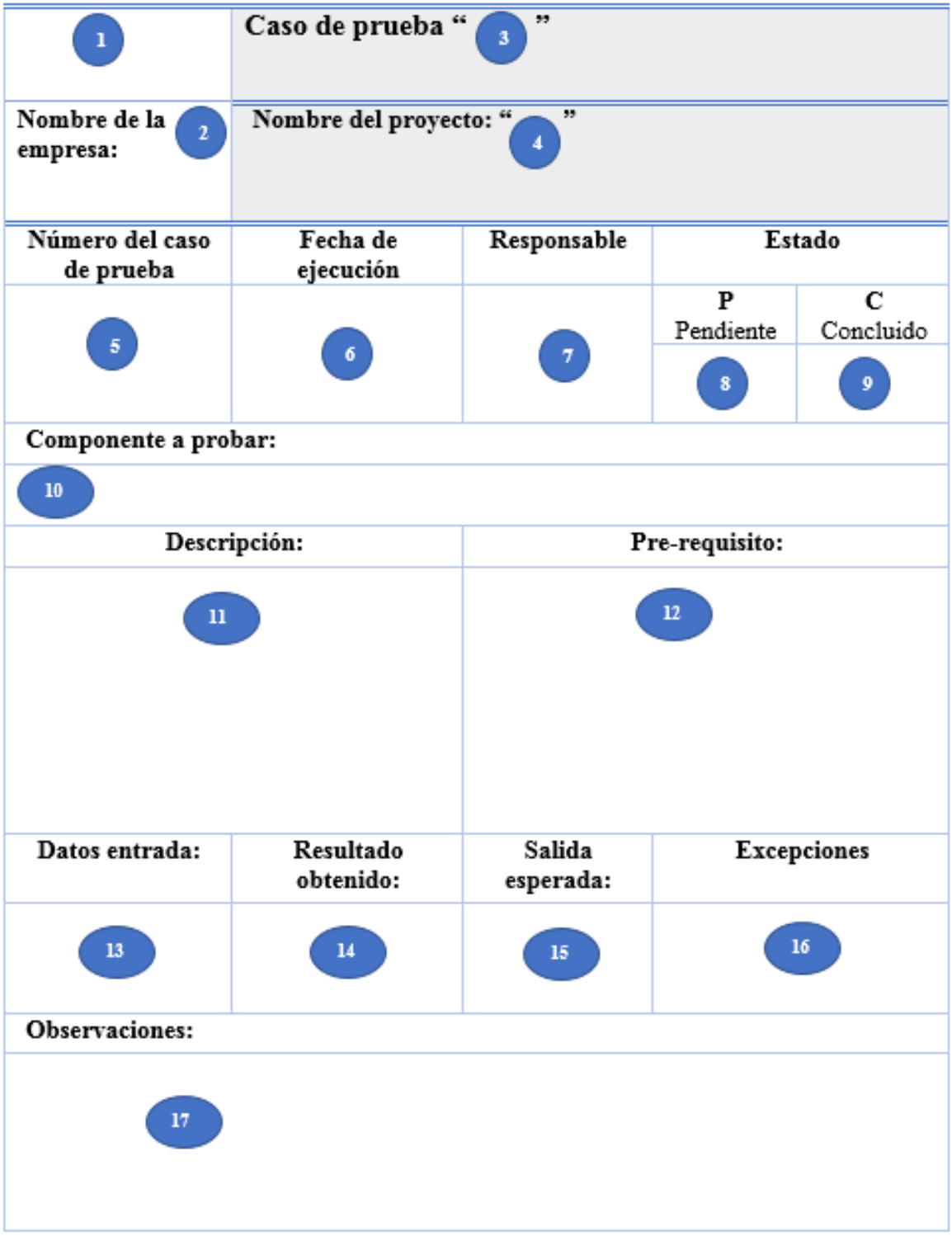


Figura 6. Formato Caso de Prueba

3.2.2 Calendario

El objetivo de este es contar con un orden de ejecución de pruebas. las fechas de inicio y fin en las que se realizan, el nombre del responsable de ejecución de pruebas y el módulo probado o componente probado como se muestra en la ***Figura 7. Formato Calendario***

1. Logotipo de la empresa
2. Nombre de la empresa
3. Nombre del proyecto
4. Nombre del caso de uso de la prueba
5. Nombre ejecutor de pruebas
6. Fecha de inicio en la que se realiza la prueba
7. Fecha de finalización de la prueba

1	**CALENDARIO**		
Nombre de la empresa: 2	**Nombre del proyecto: “ 3 ”**		
Modulo o componentes a probar	**Nombre del responsable de ejecución de prueba**	**Tiempo programado de ejecución de prueba**	
		Inicio	**Fin**
4	5	6	7

Figura 7. Calendario

3.2.3 Reporte de error prueba

Una vez realizadas las pruebas es necesario que se realice un reporte donde se informe los errores detectados, así como su prioridad en solución, el objetivo de los reportes de errores es documentar cualquier situación presentada en la ejecución de las pruebas que requiera de una investigación posterior, los elementos para cada reporte de error y que debe ser documentado son los siguientes:

Este formato permite comunicar al equipo de pruebas de software, cual es la situación o estado actual de la ejecución de pruebas, errores detectados, así como su prioridad en solución, los casos exitosos, etc. El objetivo de los reportes ***Figura 8. Reporte de Error de Pruebas*** es documentar cualquier situación presentada en la ejecución de las pruebas que requiera de una investigación posterior.

1. Logotipo de la empresa
2. Nombre de la empresa
3. Nombre del proyecto
4. Especificar el tipo de prueba realizada
5. Identificador o código del caso de prueba
6. Fecha en la que se realiza la prueba
7. Aciertos y errores de las pruebas realizadas
8. Descripción breve del error
9. Nombre ejecutor de pruebas
10. Importancia del error encontrado. La aplicación funciona, pero el problema tiene un alto impacto
11. Importancia del error encontrado. El problema tiene poco impacto en el sistema
12. Importancia del error encontrado. El problema tiene muy poco impacto en el sistema
13. El producto que se está probando en caso de tener otras versiones

1	REPORTE DE ERROR PRUEBA: “3”						
Nombre de la empresa: 2	Nombre del proyecto: “4”						
Número del caso de prueba	Fecha de ejecución de prueba	Resultado de la prueba	Descripción del error	Nombre del responsable de ejecución de prueba	Prioridad			Versión
					Alta	Media	Baja	
5	6	7	8	9	10	11	12	13

Figura 8. Formato Reporte de Error Prueba

3.2.4 Lista de control

Documento que tiene por objetivo validar el proceso de pruebas a través de preguntas que se realizaran en cuanto al proceso de pruebas, para asegurar que realmente se realizó un seguimiento correcto del procedimiento en la ejecución de pruebas.

De esta manera se asegura que el producto final cumpla o satisfaga los requerimientos por parte del cliente. La persona encargada de realizar esta lista de control será el Evaluador de pruebas. La encuesta se aplicará al final de todo el proceso de acuerdo a la estructura de las metodologías clásicas.

Para cada una de las pruebas se tendrá en cuenta:
evaluación por medio de la siguiente tabla o lista de control, que será diferente para cada tipo de prueba ver ***Figura 9. Formato Lista de Control***:

1. Logotipo de la empresa
2. Nombre de la empresa
3. Nombre del proyecto
4. Especificar el tipo de prueba realizada
5. Especifica la fecha en que se ejecuta la evaluación del proceso de pruebas
6. Numero o identificador de las preguntas
7. Pregunta para saber cómo fue el procedimiento en cuanto a la ejecución de pruebas, las preguntas pueden ser las siguientes:

¿Se realizaron las pruebas unitarias con un software?
¿Cuál es el porcentaje de cobertura del sistema, una vez ejecutadas las pruebas?
¿El funcionamiento de la prueba unitaria cumple con el diseño establecido?
¿Existe un manejo de errores adecuado al finalizar las pruebas?
¿Se cumple con el procedimiento de ejecución de la prueba?
¿Se generan los reportes de errores correspondientes al finalizar la ejecución de pruebas?

¿Se le da el seguimiento correspondiente a cada reporte de error?

En caso de no ser necesarias omitir alguna, en caso contrario el evaluador podrá realizar las preguntas de acuerdo a las necesidades del proyecto

8. Seleccionar esta casilla con una X en caso de ser positiva la respuesta del paso 7.
9. Seleccionar esta casilla con una X en caso de ser negativa la respuesta del paso 7.
10. Observaciones relacionadas con la ejecución de los casos de prueba realizados y los resultados obtenidos
11. Nombre y firma del evaluador de pruebas.

1	**LISTA DE CONTROL**			
Nombre de la empresa: 2	**Nombre del proyecto: “** 3 **”**			
	Nombre tipo de prueba: “ 4 **”**			
	Fecha evaluación: 5			
Código	**Elemento a verificar**	**Si**	**No**	**Observaciones**
6	7	8	9	10

11

Nombre y firma del evaluador

Figura 9. Formato Lista de Control

3.2.5 Lecciones aprendidas

Una vez que el proyecto ha finalizado es importante realizar un registro de los sucesos inesperados que hayan provocado algún contra tiempo a lo largo del desarrollo del proyecto, como se muestra en la ***Figura 10. Formato Lecciones Aprendidas*** para de esta manera contar con un buen nivel de comprensión de los propios errores, muy necesario para proyectos futuros.

En este formato se bebe realizar una retroalimentación del proceso de pruebas para realizar una mejora sobre el mismo especificando el incidente o el tipo de lección ya sea bueno o malo, y una descripción de la misma para poder tenerlas disponibles para su uso adecuado del proceso.

1. Logotipo de la empresa
2. Nombre de la empresa
3. Nombre del proyecto del cual se registrará la lección
4. Fecha de inicio y finalización del proyecto
5. Seleccionar con una X esta casilla en caso de que el tipo de lección aprendida haya contribuido en el éxito del proyecto en cualquiera de sus fases
6. Seleccionar con una X esta casilla en caso de que el tipo de lección aprendida haya sido negativa o haya afectado éxito del proyecto en cualquiera de sus fases
7. Especificar las características de la lección aprendida

1	**LECCIONES APRENDIDAS**		
Nombre de la empresa: 2	**Nombre del proyecto: “ 3 ”**		
Fecha del Proyecto	**Tipo de lección**		**Descripción de la lección**
	Buena	**Mala**	
4	5	6	7

Figura 10. Formato Lecciones Aprendidas

3.3 *Framework* aplicando metodologías agiles

Actualmente para ser competitivos es necesario la agilidad y flexibilidad a la hora de la creación de los proyectos, es por eso que al hacer uso de Scrum se obtienen varios beneficios entre ellos una mejora constante del producto, ya que se logra mejor comunicación entre el equipo de desarrollo y el cliente, reducción en tiempos, al usar Scrum el trabajo del equipo de desarrollo tiene un enfoque más preventivo que reactivo.

Algunos de los riesgos que se quieren evitar con el uso de Scrum son: cuando se realiza una modificación la entrega planificada por parte de los desarrolladores, se reduce el tiempo para realizar pruebas, generando que haya un producto de menor calidad ya que no se pueden ejecutar todas las pruebas correspondientes, o simplemente el equipo de pruebas de software no realizan las actividades correspondientes de acuerdo a lo especificado.

A continuación, se describen los elementos de este modelo para la ejecución de pruebas ver ***Figura 11. framework aplicando metodologías agiles***

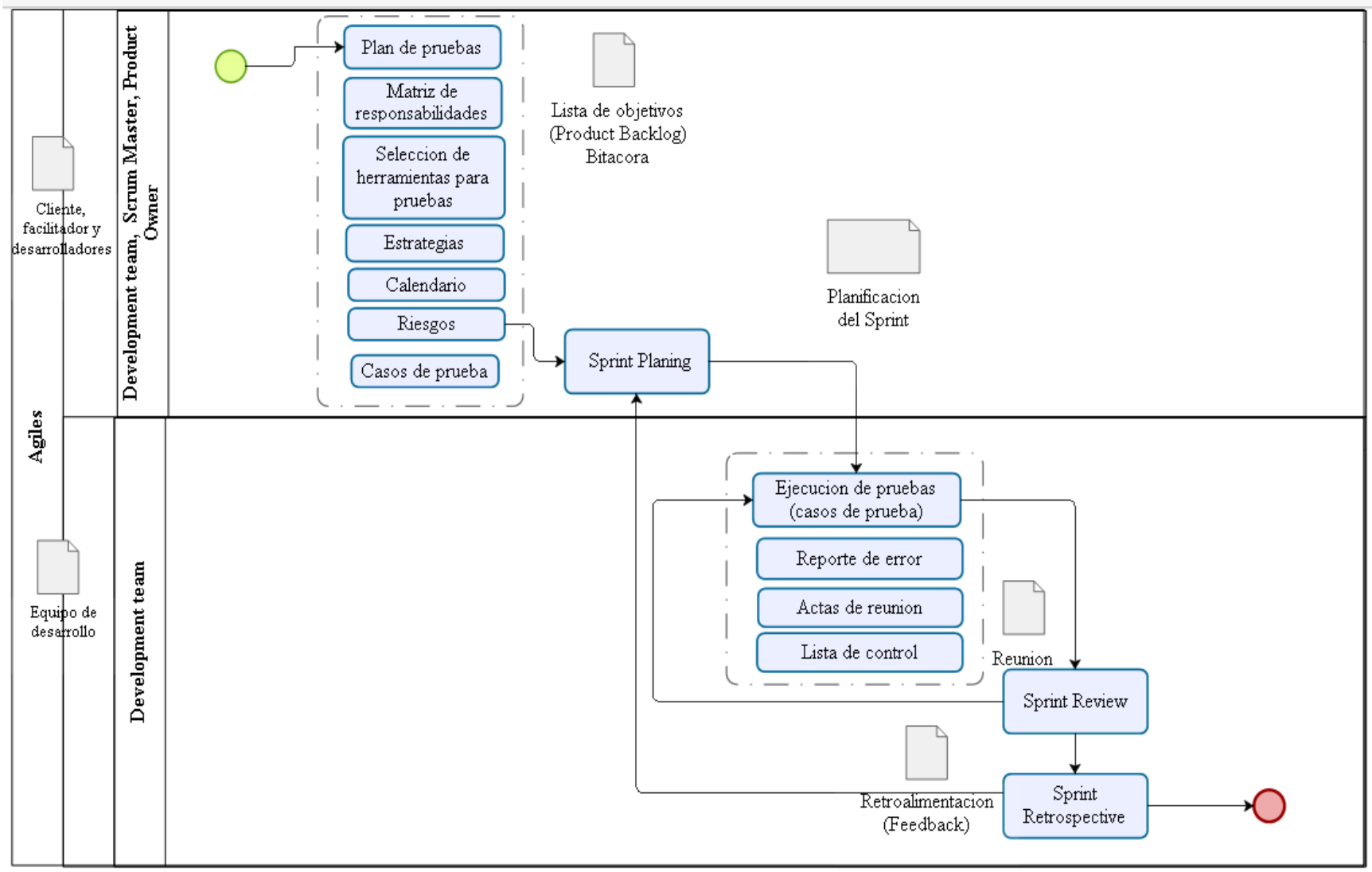


Figura 11. Framework aplicando metodologías agiles

En esta se realiza una la lista ordenada de las actividades necesarias para el producto a desarrollar.

3.3.1 Matriz de responsabilidades

Se realiza una selección del personal adecuado ***Figura 12. Formato Matriz de Responsabilidades***, todos los integrantes del equipo de pruebas tienen que ser parte del equipo de desarrollo, por lo que deben trabajar juntos en todo el desarrollo del proyecto. Los roles son simples, pero son fundamentales para que todo funciona correctamente, cada rol tiene un objetivo, dentro de los roles que se deben tener están los siguientes:

- Dueño del producto (Product Owner): se encarga verifica los intereses del producto final.
- Scrum master: eje fundamental del equipo ya que coordina y facilita los requerimientos del ciclo de vida del proyecto
- Equipo de desarrollo (Developmen team): se encargan de implementar los objetivos de un Sprint.

1. Logotipo de la empresa
2. Nombre de la empresa
3. Nombre del proyecto
4. Número del recurso humano
5. Nombre y apellido del responsable
6. Asignación de un responsable para cada acción definida
7. Asignación de tareas o actividades a un rol para cumplir determinado objetivo

1	**MATRIZ DE RESPONSABILIDADES**		
Nombre de la empresa: 2	**Nombre del proyecto: “** 3 **”**		
Recursos humanos		**Rol**	**Responsabilidades**
Número	**Nombre**		
4	5	6	7

Figura 12. Formato Matriz de Responsabilidades

3.3.2 Estrategias:

3.3.2.1 Recursos

a) Recursos de sistema:

Indica si se realizaran pruebas automatizadas con alguna herramienta(software), características del software, además incluye el recurso humano necesario para la ejecución de pruebas.

Las pruebas se realizarán en un ambiente controlado por el Administrador del sistema de pruebas, en este caso serán necesarias dos herramientas JUnit para pruebas unitarias.

b) Recurso humano:

El recurso humano que debe estar disponible para la ejecución de acuerdo a los distintos tipos de pruebas y a los objetivos.

En la ***Figura 13. Formato Niveles de Prueba*** se muestran los recursos, tanto humanos como de sistemas para poder ejecutar las pruebas.

1. Logotipo de la empresa
2. Nombre de la empresa
3. Nombre del proyecto
4. Nombre de la prueba a ejecutarse
5. Nombre de la persona que ejecutara las pruebas en el software
6. Finalidad de realizar la ejecución de pruebas
7. Nombre del software que se ocupara para la ejecución de las pruebas

1	**Niveles de prueba**		
Nombre de la empresa: 2	**Nombre del proyecto: “ 3 ”**		
Tipo de prueba	**Participantes**	**Objetivo**	**Herramienta**
4	5	6	7

Figura 13. Formato Niveles de Prueba

3.3.2.2 Riesgos

Formato que especifica los riesgos y contingencias que hay en el desarrollo de software. Cada uno de los elementos del plan de prueba contaran con algunos entregables para validación y verificación de la ejecución de pruebas ver ***Figura 14. Formato Riesgos.***

Sin importar la precisión con la que se haya preparado el proyecto a desarrollar, siempre se produce algún contratiempo que pueden afectar negativamente el desarrollo del mismo por lo cual surge la necesidad de analizar el tipo de riesgos y el impacto que produce en el proyecto, por lo cual es necesario elaborar un plan de contingencia, con el cual lo que se pretende es dejar un margen para imprevistos previsibles y añadir cierta holgura a la planificación del proyecto.

1. Logotipo de la empresa
2. Nombre de la empresa
3. Nombre del proyecto
4. Identificador del riesgo
5. Nombre del riesgo
6. El problema tiene un alto impacto que pusiera afectar tanto al equipo como al sistema
7. El problema tiene un impacto medio que pusiera afectar tanto al equipo como al sistema
8. El problema tiene un bajo impacto que pusiera afectar tanto al equipo como al sistema
9. Solución al riesgo

1	**RIESGOS**				
Nombre de la empresa: 2	**Nombre del proyecto: “ 3 ”**				
Identificador del riesgo	**Riesgo**	**Impacto**			**Plan de contingencia**
		Alto	**Medio**	**Bajo**	
4	5	6	7	8	9

Figura 14. Formato Riesgos

3.3.2.3 Selección de las herramientas

Con el fin de automatizar la calidad del software, es necesario utilizar herramientas de pruebas de software, si bien no reemplazan las pruebas manuales, se agilizan los tiempos de desarrollo del producto, por lo cual se realizaron tanto análisis como selección de herramientas ver ***Figura 15. Formato Selección de Herramientas.***

1. Logotipo de la empresa
2. Nombre de la empresa
3. Nombre del proyecto
4. Nombre del software para realizar las pruebas automatizadas
5. Número del tipo de prueba a ejecutar

1	**SELECCIÓN DE HERRAMIENTAS PARA PRUEBAS**					
Nombre de la empresa: 2	**Nombre del proyecto: “** 3 **”**					
Nombre de la herramienta	**Tipo de prueba**	**Licencia**	**Lenguaje**	**Requisitos**	**Características**	**Desventajas**
4	5	6	7	8	9	10

Figura 15. Formato Selección de Herramientas

3.3.2.4 Actas de reunión

Documento en cual se hará constar los temas que se tratan y los acuerdos que se han llevado a cabo en la reunión, esto para dejar constancia de que se le dará un seguimiento y validez a cada incidencia que se detecte en la ejecución de las pruebas.

En cada reunión realizada es importante registrar los temas tratados y los acuerdos a los que se llegan en cada una de las reuniones para que tengan validez formal, además de que de esta manera se podrá generar un seguimiento de las reuniones.

El formato de actas de reunión deberá contener los siguientes elementos ver ***Figura 16. Formato Actas de Reunión***

1. Logotipo de la empresa
2. Nombre de la empresa
3. Nombre del proyecto
4. Numero de revisión
5. Numero de Sprint
6. Reunión diaria
7. Lugar de la reunión
8. Acta de reunión numero
9. Fecha en la que se lleva a cabo la reunión
10. Nombre de las personas que asisten a la reunión
11. Rol de las personas que asisten a la reunión
12. Firma de las personas que asisten a la reunión
13. Temas tratados en la reunión
14. Compromisos de la reunión
15. Fecha máxima para entrega de lo acordado en la reunión
16. Si cumplió tanto en tiempo y forma con lo establecido en la reunión
17. Anotar compromisos nuevos o que hayan quedado pendientes para dar solución
18. Lugar donde se llevará a cabo la próxima reunión

19. Fecha en la que se llevara a cabo la próxima reunión
20. Hora en la que se llevara a cabo la siguiente reunión

1	**ACTAS DE REUNIÓN**		
Nombre de la empresa: 2	**Nombre del proyecto: “ 3 ”**		
Reunión diaria:	4	**Acta de reunión #:**	7
Revisión (Numero de Sprint):	5	**Fecha:**	8
Lugar:	6		
Asistentes(nombres):	**Cargo:**	**Firma:**	
9	10	11	
Temas tratados			
12			
Compromisos de esta reunión	**Fecha máxima de cumplimiento**	**Cumplido**	
		Si	**No**
13	14	15	16
Agenda			
17			
Próxima Reunión:			
Lugar: 18	**Fecha:** 19	**Hora:** 20	

Figura 16. Formato Actas de Reunión

3.3.2.5 Calendario

El formato servirá como apoyo para poder tener un control del tiempo en el que se realizara cada actividad y de esta manera poder mantener el tiempo programado del sistema.

Establece el módulo o componente a probar, el responsable ejecutor de la prueba y el tiempo programado (fecha inicio y fecha de finalización) ver ***Figura 17. Formato Calendario***

1. Logotipo de la empresa
2. Nombre de la empresa
3. Nombre del proyecto
4. Número del Sprint
5. Nombre del caso de uso de la prueba
6. Nombre ejecutor de pruebas
7. Fecha de inicio en la que se realiza la prueba
8. Fecha de finalización de la prueba

1	**CALENDARIO**		
Nombre de la empresa: 2	**Nombre del proyecto: “** 3 **”**	**Número de Sprint:**	4

Modulo o componentes a probar	**Nombre del responsable de ejecución de prueba**	**Tiempo programado de ejecución de prueba**	
		Inicio	**Fin**
5	6	7	8

Figura 17. Formato Calendario

3.3.2.6 Casos de prueba

El propósito fundamental del formato caso de prueba es validar que efectivamente los requerimientos son satisfactorios por lo cual es importante que haya un caso de prueba para verificar el cumplimiento de cada requerimiento ver ***Figura 18. Formato Casos de Prueba***

1. Logotipo de la empresa
2. Nombre de la empresa
3. Nombre del tipo de prueba
4. Nombre del proyecto
5. Número del Sprint
6. Código o identificador de la prueba
7. Fecha en la que se realiza la prueba
8. Nombre ejecutor de pruebas
9. Estado de la prueba que ya sea en proceso, o que ya se realizó y aún no ha sido aprobada
10. Estado en el cual ya fue aceptado o aprobado el resultado de a prueba
11. Nombre del caso de uso de la prueba
12. Descripción de lo que se probara (escenarios)
13. Condición(es) para que se ejecute el caso de prueba
14. Se especifica cada valor o dato de entrada que se requiere para la ejecución del caso de prueba
15. Aciertos y errores de las pruebas realizadas
16. Descripción del resultado que se espera de la ejecución de los casos de prueba con los datos de entrada
17. Casos en los que puede llegar a fallar la prueba
18. Observaciones relacionadas con la ejecución de los casos de prueba realizados y los resultados obtenidos

1	Caso de prueba “ 3 ”			
Nombre de la empresa: 2	Nombre del proyecto: “ 4 ”		Número de Sprint:	5
Número del caso de prueba	Fecha de ejecución	Responsable	Estado	
6	7	8	P Pendiente 9	C Concluido 10
Componente a probar:				
11				
Descripción:		Pre-requisito:		
12		13		
Datos entrada:	Resultado obtenido:	Salida esperada:	Excepciones	
14	15	16	17	
Observaciones:				
18				

Figura 18. Formato Caso de Prueba

3.3.2.7 Reporte de errores

Este formato permite comunicar al equipo de pruebas de software, cual es la situación o estado actual de la ejecución de pruebas, errores detectados, así como su prioridad en solución, los casos exitosos, etc. El objetivo de los reportes de errores es documentar cualquier situación presentada en la ejecución de las pruebas que requiera de una investigación posterior ver ***Figura 19. Formato Reporte de errores***

1. Logotipo de la empresa
2. Nombre de la empresa
3. Nombre del tipo de prueba
4. Nombre del proyecto
5. Número del sprint
6. Identificador o código del caso de prueba
7. Fecha en la que se realiza la prueba
8. Aciertos y errores de las pruebas realizadas
9. Descripción breve del error
10. Nombre ejecutor de pruebas
11. Importancia del error encontrado. La aplicación funciona, pero el problema tiene un alto impacto
12. Importancia del error encontrado. El problema tiene poco impacto en el sistema
13. Importancia del error encontrado. El problema tiene muy poco impacto en el sistema
14. El producto que se está probando en caso de tener otras versiones

1	**REPORTE DE ERROR PRUEBA: “ 3 ”**						
Nombre de la empresa: 2	**Nombre del proyecto: “ 4 ”**				**Número de Sprint:**	5	
Número del caso de prueba	**Fecha de ejecución de prueba**	**Resultado de la prueba**	**Descripción del error**	**Nombre del responsable de ejecución de prueba**	**Prioridad**			**Versión**
					Alta	**Media**	**Baja**	
6	7	8	9	10	11	12	13	14

Figura 19. Formato Reporte de Error

3.3.2.8 Lista de control

Documento que tiene por objetivo validar el proceso de pruebas a través de preguntas que se realizaran en cuanto al proceso de pruebas, para asegurar que realmente se realizó un seguimiento correcto del procedimiento en la ejecución de pruebas.

De esta manera se asegura que el producto final cumpla o satisfaga los requerimientos por parte del cliente. La persona encargada de realizar esta lista de control será el Evaluador de pruebas. La encuesta se aplicará al final de todo el proceso de acuerdo a la estructura de las metodologías clásicas.

Para cada una de las pruebas se tendrá en cuenta:
evaluación por medio de la siguiente tabla o lista de control, que será diferente para cada tipo de prueba ver ***Figura 20. Formato Lista de Control***:

1. Logotipo de la empresa
2. Nombre de la empresa
3. Nombre del proyecto
4. Especificar el tipo de prueba realizada
5. Número del sprint
6. Especifica la fecha en que se ejecuta la evaluación del proceso de pruebas
7. Número o identificador de las preguntas
8. Pregunta para saber cómo fue el procedimiento en cuanto a la ejecución de pruebas, las preguntas pueden ser las siguientes:

¿Se realizaron las pruebas unitarias con un software?
¿Cuál es el porcentaje de cobertura del sistema, una vez ejecutadas las pruebas?
¿El funcionamiento de la prueba unitaria cumple con el diseño establecido?
¿Existe un manejo de errores adecuado al finalizar las pruebas?
¿Se cumple con el procedimiento de ejecución de la prueba?

¿Se generan los reportes de errores correspondientes al finalizar la ejecución de pruebas?
¿Se le da el seguimiento correspondiente a cada reporte de error?
En caso de no ser necesarias omitir alguna, en caso contrario el evaluador podrá realizar las preguntas de acuerdo a las necesidades del proyecto

9. Seleccionar esta casilla con una X en caso de ser positiva la respuesta del paso 7.
10. Seleccionar esta casilla con una X en caso de ser negativa la respuesta del paso 7.
11. Observaciones relacionadas con la ejecución de los casos de prueba realizados y los resultados obtenidos
12. Nombre y firma del evaluador de pruebas.

1	**LISTA DE CONTROL**			
Nombre de la empresa: 2	**Nombre del proyecto: “** 3 **”**		**Número de Sprint:**	5
	Nombre tipo de prueba: “ 4 **”**			
		Fecha evaluación: 6		
Código	**Elemento a verificar**	**Si**	**No**	**Observaciones**
7	8	9	10	11

11

Nombre y firma del evaluador

Figura 20. Formato Lista de Control

3.3.2.9 Lecciones aprendidas

Una vez que el proyecto ha finalizado es importante realizar un registro de los sucesos inesperados que hayan provocado algún contra tiempo a lo largo del desarrollo del proyecto, para de esta manera contar con un buen nivel de comprensión de los propios errores, muy necesario para proyectos futuros.

En este formato se bebe realizar una retroalimentación del proceso de pruebas para realizar una mejora sobre el mismo especificando el incidente o el tipo de lección ya sea bueno o malo, y una descripción de la misma para poder tenerlas disponibles para su uso adecuado del proceso ver ***Figura 21. Formato Lecciones Aprendidas***

1. Logotipo de la empresa
2. Nombre de la empresa
3. Nombre del proyecto del cual se registrará la lección
4. Número del sprint
5. Fecha de inicio y finalización del proyecto
6. Seleccionar con una X esta casilla en caso de que el tipo de lección aprendida haya contribuido en el éxito del proyecto en cualquiera de sus fases
7. Seleccionar con una X esta casilla en caso de que el tipo de lección aprendida haya sido negativa o haya afectado éxito del proyecto en cualquiera de sus fases
8. Especificar las características de la lección aprendida

1	**LECCIONES APRENDIDAS**		
Nombre de la empresa: 2	**Nombre del proyecto: “ 3 ”**	**Número de Sprint:**	4

Fecha del Proyecto	**Tipo de lección**		**Descripción de la lección**
	Buena	**Mala**	
5	6	7	8

Figura 21. Formato Lecciones Aprendidas

3.4 Conclusión

Cada componente del plan de prueba tiene diferentes enfoques, objetivos, diferentes estrategias y herramientas de acuerdo a los tipos de pruebas a ejecutar, un equipo del cual dependerá que cumplan sus roles y actividades para tener al final de todas las pruebas un producto de calidad, por lo que es importante que se tenga un plan de pruebas coherente organizado y bien estructurado para poder detectar los errores en el software.

En cuanto a las metodologías que se toman como apoyo para la ejecución de pruebas serán dos metodologías SCRUM y Cascada, con la finalidad de que sean adaptables a las necesidades de cada proyecto.

De esta manera se podrá tener un testing integrado para el proceso de desarrollo del software, dando como resultado una mejor calidad para entregar a los clientes sus productos en tiempo y forma.

Capítulo 4

Aplicación y resultados del framework para pruebas de software

4 Aplicación y resultados del *Framework* para pruebas de software

En este Capítulo se presenta la aplicación del f*ramework* mediante las dos metodologías de desarrollo ágil (Scrum) y clásica (Cascada), se muestran algunos formatos de los resultados obtenidos de los dos sistemas probados.

4.1 *Framework* aplicando la metodología ágil (Scrum):

La aplicación del *framework* con la metodología ágil(Scrum) se realizó con el sistema "Concrete Pro Design" vinculado a la tesis titulada "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto", desarrollado por el Ing. Alfredo Xochitemol Cruz ***ver Anexo A*** .El sistema presenta el siguiente problema: No se cuenta con una aplicación, para el diseño de mezclas de concreto, por lo que genera errores en los cálculos de materiales como son: cemento, agua, entre otros.

El sistema "Concrete Pro Design" contara con siete Sprints: 1) interfaz, 2) tablas y funciones matemáticas, un sprint por cada método 3) ACI, 4) Walker, 5) Fuller y 6) Bolomey, y 7) el método comparativo.

A continuación, se muestra el proceso de ejecución de pruebas unitarias aplicando la metodología ágil ya que el sistema cuenta con cuatro métodos (ACI, Walker, Fuller y Bolomey), por lo cual se realizan iteraciones continuas con el cliente para realizar mejoras en el sistema.

4.1.1 Sprint 1 Interfaz

1. Matriz de responsabilidades

En la ***Figura 22. Formato Matriz de Responsabilidades Metodología Scrum Sprint 1*** se definió el nombre, el rol y las responsabilidades que tendrán que realizar cada integrante del equipo de pruebas.

Los roles definidos en el proyecto son tres: Scrum master, Development team y el product owner.

Logo	**MATRIZ DE RESPONSABILIDADES**		
Nombre de la empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"		
Recursos humanos		**Rol**	**Responsabilidades**
Número	**Nombre**		
1	María Janai Sáchez Hernández	Scrum Master	• Proporcionar atención especial al funcionamiento correcto de las tareas principales del sistema. • Administrar los informes de incidencia • Asegurar que cada miembro del equipo de pruebas este realizando las actividades correspondientes • Verificar la lista de control • Genera las listas de control • Evalúa si las pruebas se realizaron de manera correcta
1	Cyndi Gonzalez Águila	Development team	• Implementar los casos de la prueba a ejecutar. • Realizar plan de pruebas • Diseña,genera y llena los casos de prueba • Selecciona las herramientas para la automatización de pruebas requeridas • Realizar las pruebas • Ejecuta las pruebas • Generar los reportes • Asegurar el ambiente de prueba • Administrar el manejo de pruebas del sistema. • Administrar el calendario de pruebas • Controlar el acceso de los integrantes del equipo de pruebas a los sistemas de prueba • Verifica que las pruebas se hayan realizado de manera correcta
1	Miguel Ángel Daza Merino	Product Owner	• Define los objetivos del producto o proyecto • Participar en reuniones en colaboración con el equipo • Representante en los resultados del proyecto

Figura 22. Formato Matriz de Responsabilidades Metodología Scrum Sprint 1

2. Estrategias

En la ***Figura 23. Formato Niveles de Prueba Metodología Scrum Sprint 1*** se definió que el tipo de prueba a realizarse será unitario(código) ya que por las operaciones matemáticas que se tendrán que realizar necesitamos asegurar que realizan los cálculos exactos, según sea el caso.

Logo	**NIVELES DE PRUEBA**			
Nombre de la empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"			
Tipo de prueba	**Participantes**	**Objetivo**	**Herramienta**	
Unitaria	Tester	Encontrar errores en los datos, lógica, etc.	JUnit	NetBeans

Figura 23. Formato Niveles de Prueba Metodología Scrum Sprint 1

3. Riesgos

Los posibles riesgos encontrados que pueden afectar el proceso de pruebas ver ***Figura 24. Formato de Riesgos Metodología Scrum Sprint 1***, se clasificaron de acuerdo al tipo de impacto y además se realizó un plan de contingencia para evitar que ocurran.

Logo	**RIESGOS**		
Nombre de la empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"		
Identificador del riesgo	**Riesgo**	**Impacto**	**Plan de contingencia**
R01	Reducción del alcance del sistema o cambios frecuentes en la definición de los objetivos	Medio	El equipo de testing tendrá que trabajar de la mano con los de requerimientos con la finalidad de estar informados acerca de los cambios que hayan durante el levantamiento de requerimientos
R02	Falta de control automático del código fuente	Alto	Los casos de prueba deberán tener el seguimiento que se indica en el framework para metodología agil(Scrum)para evitar que haya nuevas versiones del sistema.
R04	Reducción de la fase de pruebas	Alto	Los casos de prueba deberán ser diseñados desde la fase de requerimientos para asi evitar la perdida de tiempo en diseñarlos una ves que ya esten realizando la codificación del sistema, evitando asi que se realicen pruebas superficiales que al final afecten o pongan en riesgo la calidad del producto
R05	Errores a la hora de hacer la ejecución de pruebas de la aplicación	Bajo	Las pruebas realizadas(casos de prueba) se realizaran de acuerdo a los casos de uso, mismos que proceden de los requerimientos dados por el cliente

Figura 24. Formato de Riesgos Metodología Scrum Sprint 1

4. Selección de las herramientas

El método utilizado para la selección de herramientas fue el siguiente: primero se realizó una investigación documental acerca del tema de pruebas de software a fin de aclarar aspectos y generar un reporte técnico para MBN que ayude a diseminar el conocimiento entre los integrantes de DMS-MBN; segundo, se investigó en el área cuales son las herramientas de software más usadas, se definieron los elementos de comparación entre ellos y se realizaron pruebas de uso de cada herramienta a fin de valorarlas; finalmente se discutió internamente los resultados de la valoración de herramientas y se definió cuáles cubren de mejor manera las necesidades de DMS.

El análisis se realizó con respecto a las siguientes características:

- Tipo de prueba: se refiere al tipo de prueba que se puede realizar con la herramienta.
- Licencia: se refiere al tipo de términos y cláusulas que el usuario debe cumplir para usar el software o herramienta
- Lenguaje: estructura, contenido y uso que posee un programa de computadora o software.
- Requisitos: condición necesaria para el uso del software.
- Sistema operativo: conjunto de órdenes y programas para procesos de una computadora.
- Interfaz gráfica: interfaz de usuario representa información y acciones disponibles del sistema.
- Reportes: muestran los resultados de la ejecución de pruebas en distintos formatos.
- Herramientas complementarias: software que complementa y amplia con nuevas funcionalidades.
- Desventajas: limitación de funciones en el software.
- Funcionamiento: serie de instrucciones y datos para aprovechar los recursos con los que cuenta un software.

En nuestro caso la herramienta que mejor se adapta a MBN es JUnit por las siguientes razones: i) cubre las pruebas unitarias y de integración además permite realizar pruebas de regresión para asegurarnos que un cambio o la incorporación de una nueva funcionalidad no haya causado otros problemas, las cuales se definieron como las más importantes para la fábrica de software de MBN, ii) la licencia de la herramienta es abierta, lo cual permite utilizarla sin incrementar el costo de la ejecución de proyectos en MBN, y iii) se adapta de manera adecuada a proyectos desarrollados en Java, los cuales son el tipo de proyectos que se desarrollan principalmente en MBN.

En la ***Figura 25. Formato Selección de Herramientas Metodología Scrum Sprint 1*** se muestra la herramienta que más se adapta a las necesidades del sistema a testear.

Logo	**SELECCIÓN DE HERRAMIENTAS**					
Nombre de la empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"					
Nombre de la herramienta	**Tipo de prueba**	**Licencia**	**Lenguaje**	**Requisitos**	**Características**	**Desventajas**
• JUnit	• Unitarias	• Open Source	• Java	• NetBeans • Eclipse	• Runners que pueden ser en modo texto, gráfico (AWT o Swing) o como tarea en Ant.	• Carencia de una interfaz completamente gráfica

Figura 25. Formato Selección de Herramientas Metodología Scrum Sprint 1

5. Actas de reunión

En la ***Figura 26. Formato Actas de Reunión Metodología Scrum Sprint 1*** se muestran los temas tratados acerca de lo que se realizaron en el proceso de ejecución de pruebas para el sprint 1, fue necesaria solo una reunión ya que solo se trató el tema de revisión de prototipo para validar que se cumple con los requerimientos por parte del cliente a través de una lista de comprobación arquitectura prototipo (Cheklist).

Logo	**ACTAS DE REUNIÓN**			
Nombre de la empresa:	**Nombre del proyecto: "**Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"			
Reunión 1	**Revisión (Numero de Sprint):** 1	**Acta de reunión #:** 001		
Lugar: Instituto Tecnológico de Apizaco		**Fecha:** 03/noviembre/2017		
Asistentes(nombres):	**Cargo:**	**Firma:**		
1.Alfredo Xochitemol Cruz	Desarrollador			
2.Cyndi González Águila	Tester			
3. María Janaí Sánchez Hernández	Scrum Master			
Temas tratados				
1	Revisión de requerimientos funcionales			
2	Arquitectura del prototipo			
Compromisos de esta reunión		**Fecha máxima de cumplimiento**	**Cumplido**	
			Si	**No**
2	Implementación de la interfaz	06/Noviembre/2017	X	
Agenda				
Próxima Reunión:				
Lugar: Instituto Tecnológico de Apizaco	**Fecha**:07/Noviembre /2017	**Hora:** 2:00 pm		

Figura 26. Formato Actas de Reunión Metodología Scrum Sprint 1

6. Calendario

En la ***Figura 27. Formato Calendario Metodología Scrum Sprint 1***, se realizó la planeación de los módulos en los que se aplicaron las pruebas, en este caso solo se realizó la planificación de revisión de arquitectura del prototipo e implementación de la interfaz, que son parte del primer sprint que se realizó con el cliente para validar los requerimientos, y de esta manera empezar a desarrollar el sistema.

Logo	**CALENDARIO**		
Nombre de la empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"	**Número de Sprint:**	1

Modulo o componentes a probar	**Nombre del responsable de ejecución de prueba**	**Tiempo programado de ejecución de prueba**	
		Inicio	**Fin**
Arquitectura del prototipo	Cyndi González Águila	01/Noviembre/2017	01/Noviembre/2017
Implementacion de interfaz	Cyndi González Águila	02/Noviembre/2017	05/Noviembre/2017

Figura 27. Formato Calendario Metodología Scrum Sprint 1

7. Lista de comprobación arquitectura prototipo (Cheklist)

En la ***Figura 28. Formato Lista de Comprobación Arquitectura Prototipo Metodología Scrum Sprint 1*** se realizó con la finalidad de brindar apoyo al Tester, mediante una serie de preguntas para realizar la inspección de los diferentes componentes que debe incluir el sistema.

Dado que se realiza la evaluación de la arquitectura del prototipo, el Tester de esta manera encontró defectos típicos que suelen aparecer en los requisitos dados por el cliente.

Logo	**LISTA DE COMPROBACIÓN ARQUITECTURA PROTOTIPO**		
Nombre de la empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"	**Número de Sprint:**	1

Pregunta	**Nombre del responsable de ejecución de prueba**	**Cumple**	
		Si	**No**
¿El diseño cumple con los requerimientos especificados por el cliente?	Cyndi González Águila		X
¿Se especifican las interfaces necesarias para el sistema?	Cyndi González Águila	X	
¿Existen contradicciones en la especificación de los requerimientos?	Cyndi González Águila		X
¿Cubre el prototipo todos los requerimientos funcionales?	Cyndi González Águila		X

Fecha de aprobación: 05/Noviembre/2017

Figura 28. Formato Lista de Comprobación Arquitectura Prototipo Metodología Scrum Sprint 1

8. Lecciones aprendidas

Las lecciones aprendidas se realizaron con la finalidad de tener una conclusión una vez que se han realizado las actividades correspondientes en el proceso de ejecución de pruebas Sprint 1. En este formato se incluyeron las causas(errores) por las cuales alguna actividad no se haya logrado de acuerdo a lo planeado, además de la descripción de mejora en la lección de error para poder tener así una mejora en los siguientes procesos de pruebas, se muestran en la ***Figura 29. Formato Lecciones Aprendidas Metodología Scrum Sprint 1***

Logo	**LECCIONES APRENDIDAS**		
Nombre de la empresa:	**Nombre del proyecto** " Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto "		
Fecha del Proyecto	**Tipo de lección**		**Descripción de lección**
	Buena	**Mala**	
01/Noviembre/2017 al 07/Noviembre/2017	X		Al realizar reuniones tanto con el cliente como con el equipo de desarrollo facilito la realización del formato lista de comprobación arquitectura prototipo (checklist)

Figura 29. Formato Lecciones Aprendidas Metodología Scrum Sprint 1

4.1.2 Sprint 2 Tablas y Funciones matemáticas

1. Actas de reunión

En este documento se realizó un registro de los temas tratados acerca las tablas como son el inicio de sesión, crear, abrir guardar proyecto, así como el cálculo de la desviación estándar que serán aplicados para los cuatro métodos (ACI, Walker, Fuller y Bolomey). Además de la implementación del primer método. Ver ***Figura 30. Formato Actas de Reunión Metodología Scrum Sprint 2***

Logo	**ACTAS DE REUNIÓN**				
Nombre de la empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"				
Reunión 3	**Revisión (Número de Sprint):** 2		**Acta de reunión #:** 003		
Lugar: Instituto Tecnológico de Apizaco			**Fecha:** 27/Noviembre/2017		
Asistentes(nombres):	**Cargo:**	**Firma:**			
1.Alfredo Xochitemol Cruz	Desarrollador				
2.Cyndi González Águila	Tester				
3. María Janaí Sánchez Hernández	Scrum Master				
Temas tratados					
1	Tablas (inicio de sesión, crear proyecto, abrir proyecto, ingresar datos y guardar proyecto) y funciones matemáticas (desviación estándar)				
Compromisos de esta reunión		**Fecha máxima de cumplimiento**	**Cumplido**		
			Si	**No**	
1	Implementación método ACI	13/Diciembre/2017	X		
Agenda					
Próxima Reunión:					
Lugar: Instituto Tecnológico de Apizaco	**Fecha**: 15/Diciembre/2017		**Hora:** 2:00 pm		

Figura 30. Formato Actas de Reunión Metodología Scrum Sprint 2

2. Calendario

En este documento se registró un orden cronológico de los módulos a testear de acuerdo a las fechas de desarrollo, las pruebas se empezaron a ejecutar a partir del 8 al 27 de noviembre del 2017 ver ***Figura 31. Formato Calendario Metodología Scrum Sprint 2***

Logo	**CALENDARIO**		
Nombre de la empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"	**Número de Sprint:**	2

Modulo o componentes a probar	**Nombre del responsable de ejecución de prueba**	**Tiempo programado de ejecución de prueba**	
		Inicio	**Fin**
Inicio de sesión	Cyndi Gonzalez Águila	08/noviembre/2017	09/noviembre/2017
Crear proyecto	Cyndi Gonzalez Águila	10/noviembre/2017	14/noviembre/2017
Abrir proyecto	Cyndi Gonzalez Águila	15/noviembre/2017	21/noviembre/2017
Guardar proyecto	Cyndi Gonzalez Águila	21/noviembre/2017	23/noviembre/2017
Calculo de la desviación estándar	Cyndi Gonzalez Águila	23/noviembre/2017	27/noviembre/2017

Figura 31. Formato Calendario Metodología Scrum Sprint 2

3. Casos de prueba

Este formato se realizó con la finalidad de determinar si se cumple con los requerimientos dados por el cliente, el tipo de prueba a ejecutar será unitaria, en la ***Figura 32. Formato Caso de Prueba Unitaria1 Metodología Scrum Sprint 2,*** se muestra el módulo a probar inicio de sesión

Logo	**Caso de prueba** "Prueba Unitaria"			
Nombre empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"			
Número del caso de prueba	**Fecha de ejecución**	**Responsable**	**Estado**	
PU_01	08/Noviembre /2017 al 09/Noviembre/2017	Cyndi González Águila	**P** Pendiente	**C** Concluido
				C
Componente a probar:				
Inicio de sesión				
Descripción:		**Pre-requisito:**		
Seleccionar usuario, dar clic sobre el botón usuario para iniciar sesión		Dar clic sobre el botón usuario para iniciar sesión		
Datos entrada:	**Resultado obtenido:**	**Salida esperada:**	**Excepciones**	
Dar clic en usuario	Muestra la pantalla de inicio	Mostrar la pantalla de inicio	No conexión con la base de datos	
Observaciones:				

Figura 32. Formato Caso de Prueba Unitaria1 Metodología Scrum Sprint 2

En la ***Figura 33. Formato Caso de Prueba Unitaria2 Metodología Scrum Sprint 2*** el módulo a probar es el de crear nuevo proyecto

<table>
<tr><td>Logo.</td><td colspan="4">Caso de prueba “Prueba Unitaria”</td></tr>
<tr><td>Nombre empresa</td><td colspan="4">Nombre del proyecto: “Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto”</td></tr>
<tr><td>Número del caso de prueba</td><td>Fecha de ejecución</td><td>Responsable</td><td colspan="2">Estado</td></tr>
<tr><td rowspan="2">PU_02</td><td rowspan="2">10/Noviembre/2017 al 14/Noviembre/2017</td><td rowspan="2">Cyndi González Águila</td><td>P Pendiente</td><td>C Concluido</td></tr>
<tr><td></td><td>C</td></tr>
<tr><td colspan="5">Componente a probar:</td></tr>
<tr><td colspan="5">Crear nuevo proyecto</td></tr>
<tr><td colspan="2">Descripción:</td><td colspan="3">Pre-requisito:</td></tr>
<tr><td colspan="2">• Seleccionar nuevo proyecto en el menú archivo, introducir nombre del proyecto y dar clic en el botón guardar</td><td colspan="3">• Haber iniciado sesión
• Tener conexión con la base de datos</td></tr>
<tr><td>Datos entrada:</td><td>Resultado obtenido:</td><td>Salida esperada:</td><td colspan="2">Excepciones</td></tr>
<tr><td>• Introducir nombre</td><td>• Creación de nuevo proyecto</td><td>• Creación de nuevo proyecto</td><td colspan="2">• Campo nombre vacío</td></tr>
<tr><td colspan="5">Observaciones:</td></tr>
<tr><td colspan="5"></td></tr>
</table>

Figura 33. Formato Caso de Prueba Unitaria2 Metodología Scrum Sprint 2

En la ***Figura 34. Formato Caso de Prueba Unitaria3 Metodología Scrum Sprint 2*** el módulo a probar es el de abrir proyecto

Logo	**Caso de prueba** "Prueba Unitaria"			
Nombre empresa	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"			
Número del caso de prueba	**Fecha de ejecución**	**Responsable**	**Estado**	
PU_03	15/Noviembre/2017 al 21/Noviembre/2017	Cyndi González Águila	**P** Pendiente	**C** Concluido C
Componente a probar:				
Abrir proyecto				
Descripción:		**Pre-requisito:**		
• Abrir proyecto, selección de ruta, selección del proyecto y dar clic en botón open		• Haber iniciado sesión • Tener proyectos guardados en la base de datos		
Datos entrada:	**Resultado obtenido:**	**Salida esperada:**	**Excepciones**	
• Ruta de donde se encuentra el proyecto	• Abrir proyecto	• Abrir proyecto	• No conexión con la base de datos • No tener creado ningún proyecto	
Observaciones:				

Figura 34. Formato Caso de Prueba Unitaria3 Metodología Scrum Sprint 2

En la ***Figura 35. Formato Caso de Prueba Unitaria4 Metodología Scrum Sprint 2*** el módulo a probar es el de guardar proyecto

<table>
<tr><td>Logo</td><td colspan="4">Caso de prueba “Prueba Unitaria”</td></tr>
<tr><td>Nombre empresa</td><td colspan="4">Nombre del proyecto: “Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto”</td></tr>
<tr><td>Número del caso de prueba</td><td>Fecha de ejecución</td><td>Responsable</td><td colspan="2">Estado</td></tr>
<tr><td rowspan="2">PU_04</td><td rowspan="2">21/Noviembre/2017
al
23/Noviembre/2017</td><td rowspan="2">Cyndi González Águila</td><td>P
Pendiente</td><td>C
Concluido</td></tr>
<tr><td></td><td>C</td></tr>
<tr><td colspan="5">Componente a probar:</td></tr>
<tr><td colspan="5">Guardar proyecto</td></tr>
<tr><td colspan="2">Descripción:</td><td colspan="3">Pre-requisito:</td></tr>
<tr><td colspan="2">• Seleccionar en el menú archivo la opción de guardar, ingresar ruta de destino y dar clic en el botón guardar</td><td colspan="3">• Haber iniciado sesión</td></tr>
<tr><td>Datos entrada:</td><td>Resultado obtenido:</td><td>Salida esperada:</td><td colspan="2">Excepciones</td></tr>
<tr><td>• Ruta de donde se desea guardar el proyecto</td><td>• Guardar proyecto</td><td>• Guardar proyecto</td><td colspan="2">• No conexión con la base de datos
• No agregar ruta
• No dar clic en el botón guardar</td></tr>
<tr><td colspan="5">Observaciones:</td></tr>
<tr><td colspan="5"></td></tr>
</table>

Figura 35. Formato Caso de Prueba Unitaria4 Metodología Scrum Sprint 2

En la ***Figura 36. Formato Caso de Prueba Unitaria5 Metodología Scrum Sprint 2*** el módulo a probar es el cálculo y eliminación de la desviación estándar

Logo	**Caso de prueba** "Prueba Unitaria"			
Nombre empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"			
Número del caso de prueba	**Fecha de ejecución**	**Responsable**	**Estado**	
PU_05	23/Noviembre 2017 al 27/Noviembre 2017	Cyndi González Águila	**P** Pendiente	**C** Concluido
				C
Componente a probar:				
• Calculo de la desviación estándar • Eliminar calculo de la desviacion estándar				
Descripción:		**Pre-requisito:**		
• Seleccionar en el menú la opción de Herramientas		**Calcular**: • Haber iniciado sesión • Ingresar un numero entero o con decimal en resistencia • Dar click en agregar **Eliminar**: • Dar click en eliminar **Eliminar todo**: • Dar click en eliminar todo		
Datos entrada:	**Resultado obtenido:**	**Salida esperada:**	**Excepciones**	
Calcular: • Ingresar un numero en resistencia **Eliminar**: • Seleccionar el registro a eliminar **Eliminar todo**: • Dar click en eliminar todo	**Calcular**: • Calculo de la desviacion estándar **Eliminar**: • Eliminacion del calculo de desviacion estándar **Eliminar todo**: • Eliminacion de todos los calculos de desviacion estándar	**Calcular**: • Calculo de la desviacion estándar **Eliminar**: • Eliminacion del calculo de desviacion estándar **Eliminar todo**: • Eliminacion de todos los calculos de desviacion estándar	**Calcular**: • No conexión con la base de datos • No haber ingresado un numero en resistencia • No dar clic en el botón agregar **Eliminar**: • Eliminacion de calculos de desviacion estándar	
Observaciones:				

Figura 36. Formato Caso de Prueba Unitaria5 Metodología Scrum Sprint 2

4. Lista de Control de Pruebas

Con la ***Figura 37. Formato Lista de Control Pruebas Metodología Scrum Sprint 2*** se realizó con la finalidad de verificar que se siguieron los pasos del Framework aplicando metodologías agiles, que las actividades establecidas en el Formato Matriz de Responsabilidades Metodología Scrum Sprint 1, con el llenado de los formatos indicados, además de la utilización de un software o herramienta para la automatización de las pruebas. Una vez asegurado que el módulo cumplió los requerimientos dados por parte del cliente el Evaluador firmo este documento, asegurando que el proceso para la ejecución de pruebas se realizó de manera satisfactoria

Logo	**LISTA DE CONTROL PRUEBAS UNITARIAS**			
Nombre de la empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"			
	Nombre tipo de prueba: "Unitaria**"**			
	Fecha evaluación:			
Código	**Elemento a verificar**	**Si**	**No**	**Observaciones**
PU1	¿Se realizaron las pruebas unitarias con un software?	**X**		
PU2	¿Cuál es el porcentaje de cobertura del sistema, una vez ejecutadas las pruebas?	**X**		
PU3	¿El funcionamiento de la prueba unitaria cumple con el diseño establecido?	**X**		
PU4	¿Existe un manejo de errores adecuado al finalizar las pruebas?	**X**		
PU5	¿Se cumple con el procedimiento de ejecución de la prueba?	**X**		
PU6	¿Se generan los reportes de errores correspondientes al finalizar la ejecución de pruebas?	**X**		
PU7	¿Se le da el seguimiento correspondiente a cada reporte de error?	**X**		
	María Janai Sánchez Hernández **Nombre y firma del evaluador**			

Figura 37. Formato Lista de Control Pruebas Metodología Scrum Sprint 2

5. Lecciones aprendidas

El tipo de lección en el Sprint 2 es de acierto ya que la comunicación entre equipo de desarrollo y el de pruebas facilito el cumplimiento del tiempo planeados ver ***Figura 38. Formato Lecciones Aprendidas Metodología Scrum Sprint 2***

Logo	**LECCIONES APRENDIDAS**		
Nombre de la empresa:	**Nombre del proyecto** " Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto "		
Fecha del Proyecto	**Tipo de lección**		**Descripción de lección**
	Buena	**Mala**	
08/Noviembre/2017 al 27/Noviembre/2017		X	Tener una mejor comunicación y mantenerse informados de los cambios que surgen en el proyecto

Figura 38.Formato Lecciones Aprendidas Metodología Scrum Sprint 2

1.1.1 Sprint 3 Método ACI

1. Actas de reunión

En este documento se registró los temas tratados en la reunión, se realizó la entrega del método ACI y el compromiso que se realizo fue la implementación del método Walker, ver ***Figura 39. Formato Actas de Reunión Metodología Scrum Sprint 3***

Logo	**ACTAS DE REUNION**				
Nombre de la empresa:	**Nombre del proyecto: "**Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"				
Reunión	4	**Revisión (Número de Sprint):**	3	**Acta de reunión #:**	004
Lugar:	Instituto Tecnológico de Apizaco			**Fecha:**	15/Diciembre/2017
Asistentes (nombres):		**Cargo:**		**Firma:**	
1.Alfredo Xochitemol Cruz		Desarrollador			
2.Cyndi González Águila		Tester			
3.María Janaí Sánchez Hernández		Scrum Master			
Temas tratados					
1	Método ACI				
Compromisos de esta reunión		**Fecha máxima de cumplimiento**	**Cumplido**		
			Si		**No**
	Implementación método Walker	27/Diciembre/2017	X		
Agenda					
Próxima Reunión:					
Lugar: Instituto Tecnológico de Apizaco		**Fecha**: 29/Diciembre/2017	**Hora:** 2:00 pm		

Figura 39. Formato Actas de Reunión Metodología Scrum Sprint 3

2. Calendario

Este documento contiene el registro de los módulos a probar, así como el tiempo programado de ejecución de la prueba, ver ***Figura 40. Formato Calendario Metodología Scrum Sprint 3***

Logo	**CALENDARIO**		
Nombre del proyecto: "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"		**Numero de Sprint:**	"2"
Modulo o componentes a probar	**Nombre del responsable de ejecución de prueba**	**Tiempo programado de ejecución de prueba**	
		Inicio	**Fin**
Determinación de la resistencia promedio con la compresión f´cr	Cyndi González Águila	28/Noviembre/2017	28/Noviembre/2017
Selección del asentamiento	Cyndi González Águila	29/Noviembre/2017	30/Noviembre/2017
Selección del volumen unitario del agua	Cyndi González Águila	01/Diciembre/2017	04/Diciembre/2017
Selección del volumen del aire	Cyndi González Águila	05/Diciembre/2017	05/Diciembre/2017
Determinación de la relación agua/cemento	Cyndi González Águila	06/Diciembre/2017	06/Diciembre/2017
Determinar el factor cemento	Cyndi González Águila	07/Diciembre/2017	07/Diciembre/2017
Determinar contenido del agregado grueso	Cyndi González Águila	08/Diciembre/2017	08/Diciembre/2017
Calculo de los volúmenes absolutos	Cyndi González Águila	11/Diciembre/2017	11/Diciembre/2017
Calculo del contenido del agregado	Cyndi González Águila	12/Diciembre/2017	12/Diciembre/2017
Valores de diseño	Cyndi González Águila	13/Diciembre/2017	13/Diciembre/2017
Corrección por humedad del agregado	Cyndi González Águila	14/Diciembre/2017	14/Diciembre/2017
Proporción en peso	Cyndi González Águila	15/Diciembre/2017	15/Diciembre/2017
Equivalencia en peso por tanda de bulto	Cyndi González Águila	18/Diciembre/2017	18/Diciembre/2017

Figura 40. Formato Calendario Metodología Scrum Sprint 2

3. Casos de prueba

En la ***Figura 41. Formato Caso de Prueba Unitaria2 Metodología Scrum Sprint 3*** el módulo a probar es el de determinación de la resistencia promedio con la con la compresión f´cr

Logo	**Caso de prueba** "Prueba Unitaria"			
Nombre empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"			
Numero del caso de prueba	**Fecha de ejecución**	**Responsable**	**Estado**	
PU_01	28/Noviembre /2017	Cyndi González Aguila	**P** Pendiente	**C** Concluido C
Componente a probar:				
Determinación de la resistencia promedio con la compresión f´cr				
Descripción:		**Pre-requisito:**		
Determinación de la resistencia promedio con la compresión f´cr: **a)mediante desviacion estándar:** • Calcular desviacion estandar **b)sin registro de numero anteriores** • Seleccionar la opción de Definir valor propio de coreccion • agregar un numero en la opción Valor a sumar **c)mediante el control de calidad de la obra** • Seleccion el control en la tabla proporcionada • Seleccionar la opción de Definir valor propio de coreccion o Agregar un numero en la opción Valor de coreccion		Haber dado click en el mantt en la opción de métodos y seleccionar ACI		
Datos entrada:	**Resultado obtenido:**	**Salida esperada:**	**Excepciones**	
a)mediante desviacion estándar Seleccionar uno de los 30 ensayos consecutivos de las obras anteriores **b)sin registro de numero anteriores:** Agregar un numero entero o decimal en la opción Valor de coreccion **c)mediante el control de calidad de la obra** Agregar un numero entero o decimal en la opción Valor de coreccion	Determinación de la resistencia promedio con la compresión f´cr: **a)mediante desviacion estándar** **b)sin registro de numero anteriores:** **c)mediante el control de calidad de la obra**	Determinación de la resistencia promedio con la compresión f´cr: **a)mediante desviacion estándar** **b)sin registro de numero anteriores:** **c)mediante el control de calidad de la obra**	• No conexión con la base de datos • Sin registro de 30 ensayos consecutivos de obras anteriores	
Observaciones:				

Figura 41. Formato Caso de Prueba Unitaria1 Metodología Scrum Sprint 3

En la ***Figura 42. Formato Caso de Prueba Unitaria2 Metodología Scrum Sprint 3*** el módulo a probar es el de selección del asentamiento

Logo.	**Caso de prueba** "Prueba Unitaria"		
Nombre empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"		
Número del caso de prueba	**Fecha de ejecución**	**Responsable**	**Estado**
PU_02	29/Noviembre /2017 al 30/Noviembre/2017	Cyndi González Águila	**P** Pendiente / **C** Concluido / C
Componente a probar:			
Selección del asentamiento			
Descripción:		**Pre-requisito:**	
Selección del asentamiento (numero de pulgada)del 1 al 10		• Calcular el paso 1: la resistencia promedio a la compresión f´cr, y dar click en el botón siguiente paso • Seleccionar el tipo de asentamiento	
Datos entrada:	**Resultado obtenido:**	**Salida esperada:**	**Excepciones**
Selección del asentamiento (numero de pulgada)del 1 al 10	Selección del asentamiento	Selección del asentamiento	• No conexión con la base de datos • No haber realizado el paso anterior
Observaciones:			

Figura 42. Formato Caso de Prueba Unitaria2 Metodología Scrum Sprint 3

En la ***Figura 43. Formato Caso de Prueba Unitaria3 Metodología Scrum Sprint 3*** el módulo a probar es el de selección del volumen unitario del agua

Logo.	**Caso de prueba** "Prueba Unitaria"			
Nombre empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"			
Número del caso de prueba	**Fecha de ejecución**	**Responsable**	**Estado**	
PU_03	01/Diciembre/2017 al 04/Diciembre/2017	Cyndi González Águila	**P** Pendiente	**C** Concluido
				C
Componente a probar:				
Selección del volumen unitario del agua				
Descripción:		**Pre-requisito:**		
Calculo del volumen unitario del agua		• Calcular el paso 2: Selección del asentamiento • Seleccionar la casilla Definir volumen de agua		
Datos entrada:	**Resultado obtenido:**	**Salida esperada:**	**Excepciones**	
Introducir un numero entero o decimal en Definir volumen de agua	Calculo del volumen unitario del agua	Calculo del volumen unitario del agua	• No conexión con la base de datos • No haber realizado el paso anterior	
Observaciones:				

Figura 43. Formato Caso de Prueba Unitaria3 Metodología Scrum Sprint 3

En la ***Figura 44. Formato Caso de Prueba Unitaria4 Metodología Scrum Sprint 3*** el módulo a probar es el de selección del volumen de aire

<table>
<tr><td>Logo.</td><td colspan="4">Caso de prueba “Prueba Unitaria”</td></tr>
<tr><td>Nombre empresa:</td><td colspan="4">Nombre del proyecto: “Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto”</td></tr>
<tr><td>Número del caso de prueba</td><td>Fecha de ejecución</td><td>Responsable</td><td colspan="2">Estado</td></tr>
<tr><td rowspan="2">PU_04</td><td rowspan="2">05/Diciembre/2017</td><td rowspan="2">Cyndi González Águila</td><td>P
Pendiente</td><td>C
Concluido</td></tr>
<tr><td></td><td>C</td></tr>
<tr><td colspan="5">Componente a probar:</td></tr>
<tr><td colspan="5">Selección del volumen de aire</td></tr>
<tr><td colspan="2">Descripción:</td><td colspan="3">Pre-requisito:</td></tr>
<tr><td colspan="2">Calculo del volumen de aire atrapado</td><td colspan="3">• Calcular el paso 3: Selección del volumen unitario del agua
• Seleccionar la casilla Definir contenido de aire atrapado</td></tr>
<tr><td>Datos entrada:</td><td>Resultado obtenido:</td><td>Salida esperada:</td><td colspan="2">Excepciones</td></tr>
<tr><td>Seleccionar un numero ya sea con decimal o entero</td><td>Calculo del volumen de aire atrapado</td><td>Calculo del volumen de aire atrapado</td><td colspan="2">• No conexión con la base de datos
• No haber realizado el paso anterior</td></tr>
<tr><td colspan="5">Observaciones:</td></tr>
<tr><td colspan="5"></td></tr>
</table>

Figura 44. Formato Caso de Prueba Unitaria4 Metodología Scrum Sprint 3

En la ***Figura 45. Formato Caso de Prueba Unitaria5 Metodología Scrum Sprint 3*** el módulo a probar es el de determinación de la relación agua/cemento, el resultado obtenido de esta prueba fue de error ya que la casilla determinar la relación agua/cemento no tenía ninguna acción por lo cual no se pudo obtener un resultado, este error se notificó al Scrum Master para que de inmediato se le hiciera entrega al desarrollador del reporte con el registro del resultado obtenido para que se realizara la corrección del mismo. Y una vez realizada la corrección de realizo nuevamente la prueba para asegurar que el error fue corregido.

Logo.	**Caso de prueba** "Prueba Unitaria"			
Nombre empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"			
Número del caso de prueba	**Fecha de ejecución**	**Responsable**	**Estado**	
PU_05	06/Diciembre/2017	Cyndi González Águila	**P** Pendiente	**C** Concluido
				C
Componente a probar:				
Determinación de la relación agua/cemento				
Descripción:		**Pre-requisito:**		
Calculo de la relación agua/cemento Seleccionar la casilla determinar la relación agua/cemento Seleccionar un numero decimal o entero		• Calcular el paso 4: Selección del volumen del aire • Seleccionar la casilla:Determinar la relación agua/cemento		
Datos entrada:	**Resultado obtenido:**	**Salida esperada:**	**Excepciones**	
Seleccionar un numero ya dea con decimal o entero	No tiene acción la casilla de Determinar la relación agua/cemento	Calculo de la relación agua/cemento	• No conexión con la base de datos • No haber realizado el paso anterior	
Observaciones:				

Figura 45. Formato Caso de Prueba Unitaria5 Metodología Scrum Sprint 3

En la ***Figura 46. Formato Caso de Prueba Unitaria5_1 Metodología Scrum Sprint 3*** el módulo a probar es el de determinación de la resistencia promedio con la con la compresión f´cr, en este documento se muestra que se realizó nuevamente la prueba y que el resultado obtenido es el que realmente se esperaba

Logo	**Caso de prueba** "Prueba Unitaria"			
Nombre empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"			
Número del caso de prueba	**Fecha de ejecución**	**Responsable**	**Estado**	
PU_05_01	06/Diciembre/2017	Cyndi González Águila	**P** Pendiente	**C** Concluido
				C
Componente a probar:				
Determinación de la relación agua/cemento				
Descripción:		**Pre-requisito:**		
Calculo de la relación agua/cemento Seleccionar la casilla determinar la relación agua/cemento Seleccionar un numero decimal o entero		• Calcular el paso 4: Selección del volumen del aire • Seleccionar la casilla:Determinar la relación agua/cemento		
Datos entrada:	**Resultado obtenido:**	**Salida esperada:**	**Excepciones**	
Seleccionar un numero ya dea con decimal o entero	Calculo de la relación agua/cemento	Calculo de la relación agua/cemento	• No conexión con la base de datos • No haber realizado el paso anterior	
Observaciones:				

Figura 46. Formato Caso de Prueba Unitaria5_1Metodologia Scrum Sprint 3

En la ***Figura 47. Formato Caso de Prueba Unitaria6 Metodología Scrum Sprint 3*** el módulo a probar es el de determinar el factor cemento

Logo	**Caso de prueba** "Prueba Unitaria"			
Nombre empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"			
Número del caso de prueba	**Fecha de ejecución**	**Responsable**	**Estado**	
PU_06	07/Diciiembre /2017	Cyndi González Águila	**P** Pendiente	**C** Concluido C
Componente a probar:				
Determinar el factor cemento				
Descripción:		**Pre-requisito:**		
• Determinar la cantidad de cemento • Introducir el tamaño del bulto		• Calcular el paso 5: Determinación de la relación agua/cemento		
Datos entrada:	**Resultado obtenido:**	**Salida esperada:**	**Excepciones**	
Introducir el tamaño del bulto del cemento	Calculo del factor cemento	Calculo del factor cemento	• No conexión con la base de datos • No haber realizado el paso anterior	
Observaciones:				

Figura 47. Formato Caso de Prueba Unitaria6 Metodología Scrum Sprint 3

En la ***Figura 48. Formato Caso de Prueba Unitaria7 Metodología Scrum Sprint 3*** el módulo a probar es el de determinar contenido del agregado grueso

<table>
<tr><td>Logo</td><td colspan="4">Caso de prueba “Prueba Unitaria”</td></tr>
<tr><td>Nombre empresa:</td><td colspan="4">Nombre del proyecto: “Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto”</td></tr>
<tr><td>Número del caso de prueba</td><td>Fecha de ejecución</td><td>Responsable</td><td colspan="2">Estado</td></tr>
<tr><td rowspan="2">PU_07</td><td rowspan="2">08/Diciembre /2017</td><td rowspan="2">Cyndi González Águila</td><td>P
Pendiente</td><td>C
Concluido</td></tr>
<tr><td></td><td>C</td></tr>
<tr><td colspan="5">Componente a probar:</td></tr>
<tr><td colspan="5">Determinar contenido del agregado grueso</td></tr>
<tr><td colspan="2">Descripción:</td><td colspan="3">Pre-requisito:</td></tr>
<tr><td colspan="2">Calcular el contenido del agregado grueso
Habilitar la casilla Determinar volumen del AG
Seleccionar un numero decimal o entero (del 0 al 1)</td><td colspan="3">• Calcular el paso 6: Determinar el factor cemento</td></tr>
<tr><td>Datos entrada:</td><td>Resultado obtenido:</td><td>Salida esperada:</td><td colspan="2">Excepciones</td></tr>
<tr><td>Seleccionar un numero decimal o entero (del 0.01 al 1)</td><td>Calculo del contenido del agregado grueso</td><td>Calculo del contenido del agregado grueso</td><td colspan="2">• No conexión con la base de datos
• No haber realizado el paso anterior</td></tr>
<tr><td colspan="5">Observaciones:</td></tr>
<tr><td colspan="5"></td></tr>
</table>

Figura 48. Formato Caso de Prueba Unitaria7 Metodología Scrum Sprint 3

En la ***Figura 49. Formato Caso de Prueba Unitaria4 Metodología Scrum Sprint 3*** el módulo a probar es el de selección del volumen de aire

<table>
<tr><td>  </td><td colspan="4">Caso de prueba “Prueba Unitaria”</td></tr>
<tr><td>Nombre empresa:</td><td colspan="4">Nombre del proyecto: “Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto”</td></tr>
<tr><td>Número del caso de prueba</td><td>Fecha de ejecución</td><td>Responsable</td><td colspan="2">Estado</td></tr>
<tr><td rowspan="2">PU_08</td><td rowspan="2">11/Diciembre /2017</td><td rowspan="2">Cyndi González Águila</td><td>P
Pendiente</td><td>C
Concluido</td></tr>
<tr><td></td><td>C</td></tr>
<tr><td colspan="5">Componente a probar:</td></tr>
<tr><td colspan="5">Calculo de los volúmenes absolutos</td></tr>
<tr><td colspan="2">Descripción:</td><td colspan="3">Pre-requisito:</td></tr>
<tr><td colspan="2">Calculo de los volúmenes absolutos
Mostrar en una tabla el calculo de la suma de los volúmenes absolutos del agua, cemento, aire y agregado grueso</td><td colspan="3">• Calcular el paso 7: Determinar contenido del agregado grueso</td></tr>
<tr><td>Datos entrada:</td><td>Resultado obtenido:</td><td>Salida esperada:</td><td colspan="2">Excepciones</td></tr>
<tr><td>Suma de los volúmenes absolutos del agua, cemento, aire y agregado grueso</td><td>Mostrar los valores en una tabla de la suma de los volúmenes absolutos del agua, cemento, aire y agregado grueso</td><td>Mostrar los valores en una tabla de la suma de los volúmenes absolutos del agua, cemento, aire y agregado grueso</td><td colspan="2">• No conexión con la base de datos
• No haber realizado el paso anterior</td></tr>
<tr><td colspan="5">Observaciones:</td></tr>
<tr><td colspan="5"></td></tr>
</table>

Figura 49. Formato Caso de Prueba Unitaria8 Metodología Scrum Sprint 3

En la ***Figura 50. Formato Caso de Prueba Unitaria9 Metodología Scrum Sprint 3*** el módulo a probar es el de selección del cálculo del agregado fino

Logo	**Caso de prueba** "Prueba Unitaria"		
Nombre empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"		
Número del caso de prueba	**Fecha de ejecución**	**Responsable**	**Estado**
PU_09	12/Diciembre /2017	Cyndi González Águila	**P** Pendiente / **C** Concluido / C
Componente a probar:			
Calculo del contenido del agregado fino			
Descripción:		**Pre-requisito:**	
Calculo del contenido del agregado fino Mostrar el contenido del agregado		• Calcular el paso 8: Calculo de los volúmenes absolutos	
Datos entrada:	**Resultado obtenido:**	**Salida esperada:**	**Excepciones**
La unidad entre el resultado del paso 8:Calculo de los volúmenes absolutos	Mostrar los valores del calculo del contenido del agregado fino	Mostrar los valores del calculo del contenido del agregado fino	• No conexión con la base de datos • No haber realizado el paso anterior
Observaciones:			

Figura 50. Formato Caso de Prueba Unitaria9 Metodología Scrum Sprint 3

4. Reporte de errores

En la ***Figura 51. Formato Reporte de Error Prueba Metodología Scrum Sprint 3*** se describen cada uno de los errores encontrados en el módulo ACI, en los cuales se aplicaron nuevamente las pruebas una vez que el desarrollador realizo las correcciones con lo que se validó el funcionamiento del mismo.

Logo.	**REPORTE DE ERROR PRUEBA: “ Unitaria ”**							
Nombre de la empresa:	**Nombre del proyecto: “** Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto **”**				**Número de Sprint:**			**3**
Número del caso de prueba	**Fecha de ejecución de prueba**	**Resultado de la prueba**	**Descripción del error**	**Nombre del responsable de ejecución de prueba**	**Prioridad**			**Versión**
					Alta	**Media**	**Baja**	
PU_05	06/Diciembre/ 2017	No se puede realizar el Calculo de la relación agua/cemento	No tiene acción la casilla de Determinar la relación agua/cemento	Cyndi González Águila		X		1

Figura 51.Formato Reporte de Error Prueba Metodología Scrum Sprint 3

5. Lista de control de pruebas

Se realizo con la finalidad de verificar que se siguieron los pasos del Framework aplicando metodologías agiles, de esta manera podemos verificar que el proceso se realizó como se establece en el framework aplicando la metodología ágil ver **Figura *52. Formato Lista de Control Pruebas Metodología Scrum Sprint 3***

Logo	**LISTA DE CONTROL PRUEBAS UNITARIAS**			
Nombre de la empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"			
	Nombre tipo de prueba: "Unitaria**"**			
	Fecha evaluación:			
Código	**Elemento a verificar**	**Si**	**No**	**Observaciones**
PU1	¿Se realizaron las pruebas unitarias con un software?	**X**		
PU2	¿Cuál es el porcentaje de cobertura del sistema, una vez ejecutadas las pruebas?	**X**		
PU3	¿El funcionamiento de la prueba unitaria cumple con el diseño establecido?	**X**		
PU4	¿Existe un manejo de errores adecuado al finalizar las pruebas?	**X**		
PU5	¿Se cumple con el procedimiento de ejecución de la prueba?	**X**		
PU6	¿Se generan los reportes de errores correspondientes al finalizar la ejecución de pruebas?	**X**		
PU7	¿Se le da el seguimiento correspondiente a cada reporte de error?	**X**		

María Janai Sánchez Hernández

Nombre y firma del evaluador

Figura 52. Formato Lista de Control Pruebas Metodología Scrum Sprint 3

6. Lecciones aprendidas

En este formato se incluyeron las causas por las cuales se dificulto el proceso de ejecución de pruebas del sprint como se muestran en la ***Figura 53. Formato Lecciones Aprendidas Metodología Scrum Sprint 3***

Logo	**LECCIONES APRENDIDAS**		
Nombre de la empresa:	**Nombre del proyecto** " Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto "		
Fecha del Proyecto	**Tipo de lección**		**Descripción de lección**
	Buena	**Mala**	
28/Noviembre/2017 al 18/Diciembre/2017	X		Al realizar el reporte de errores facilita al desarrollador la corrección de los mismos

Figura 53.Formato Lecciones Aprendidas Metodología Scrum Sprint 3

4.1.3 Sprint 4 Método Walker

1. Actas de reunión

En este documento se registró los temas tratados en la reunión, se realizó la entrega del método Walker y el compromiso que se realizo fue la implementación del método Fuller, ver ***Figura 54. Formato Actas de Reunión Metodología Scrum Sprint 4***

Logo	**ACTAS DE REUNIÓN**			
Nombre de la empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"			
Reunión 5	**Revisión (Número de Sprint):**	4	**Acta de reunión #:**	005
Lugar:	Instituto Tecnológico de Apizaco		**Fecha:**	29/Diciembre/2017
Asistentes(nombres):	**Cargo:**	**Firma:**		
1.Alfredo Xochitemol Cruz	Desarrollador			
2.Cyndi Gonzalez Águila	Tester			
3. María Janaí Sánchez Hernández	Scrum Master			
Temas tratados				
1 Método Walker				
Compromisos de esta reunión	**Fecha máxima de cumplimiento**	**Cumplido**		
		Si	**No**	
1 Implementación método Fuller	15/Enero/2018	X		
Agenda				
Próxima Reunión:				
Lugar: Instituto Tecnológico de Apizaco	**Fecha**: 17/Enero/2018	**Hora:** 2:00 pm		

Figura 54. Formato Actas de Reunión Metodología Scrum Sprint 4

2. Calendario

Este documento se hizo una planeación de cada uno de los componentes a probar que integran la funcionalidad del método Walker, ver ***Figura 55. Formato Calendario Metodología Scrum Sprint 4***

Logo.	**CALENDARIO**		
Nombre de la empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"	**Número de Sprint:**	4

Modulo o componentes a probar	**Nombre del responsable de ejecución de prueba**	**Tiempo programado de ejecución de prueba**	
		Inicio	**Fin**
Calcular la suma de los volúmenes absolutos de la pasta	Cyndi González Águila	19/Diciembre/2017	20/Diciembre/2017
Determinar el volumen absoluto de los agregados	Cyndi González Águila	21/Diciembre/2017	22/Diciembre/2017
Determinar el porcentaje del agregado fino	Cyndi González Águila	23/Diciembre/2017	26/Diciembre/2017
Definir los volúmenes absolutos del agregado	Cyndi González Águila	27/Diciembre/2017	28/Diciembre/2017
Cálculo del contenido del agregado fino	Cyndi González Águila	29/Diciembre/2017	02/Enero/2018

Figura 55. Formato Calendario Metodología Scrum Sprint 4

3. Casos de prueba

En la ***Figura 56. Formato Caso de Prueba Unitaria1 Metodología Scrum Sprint 4*** el módulo a probar es el de calcular la suma de los volúmenes absolutos de la pasta

<table>
<tr><td>Logo</td><td colspan="4">Caso de prueba “Prueba Unitaria”</td></tr>
<tr><td>Nombre empresa:</td><td colspan="4">Nombre del proyecto: “Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto”</td></tr>
<tr><td>Número del caso de prueba</td><td>Fecha de ejecución</td><td>Responsable</td><td colspan="2">Estado</td></tr>
<tr><td rowspan="2">PU_01</td><td rowspan="2">19/Diciembre /2017
al
20/Diciembre/2017</td><td rowspan="2">Cyndi González Águila</td><td>P
Pendiente</td><td>C
Concluido</td></tr>
<tr><td></td><td>C</td></tr>
<tr><td colspan="5">Componente a probar:</td></tr>
<tr><td colspan="5">Calcular la suma de los volúmenes absolutos de la pasta</td></tr>
<tr><td colspan="2">Descripción:</td><td colspan="3">Pre-requisito:</td></tr>
<tr><td colspan="2">Calcular la suma de los volúmenes absolutos de la pasta
Mostrar el resultado en volumen de la pasta</td><td colspan="3">• Calcular el paso 6: Determinar el factor cemento</td></tr>
<tr><td>Datos entrada:</td><td>Resultado obtenido:</td><td>Salida esperada:</td><td colspan="2">Excepciones</td></tr>
<tr><td>Realizar la suma de los valores de cemento, agua y aire, de los pasos anteriores</td><td>Calcular la suma de los volúmenes absolutos de la pasta</td><td>Calcular la suma de los volúmenes absolutos de la pasta</td><td colspan="2">• No conexión con la base de datos
• No haber realizado el paso anterior</td></tr>
<tr><td colspan="5">Observaciones:</td></tr>
<tr><td colspan="5"></td></tr>
</table>

Figura 56.Formato Caso de Prueba Unitaria1 Metodología Scrum Sprint 4

En la ***Figura 57. Formato Caso de Prueba Unitaria2 Metodología Scrum Sprint 4*** el módulo a probar es el de determinar el volumen absoluto de los agregados

Logo.	**Caso de prueba** "Prueba Unitaria"			
Nombre empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"			
Número del caso de prueba	**Fecha de ejecución**	**Responsable**	**Estado**	
PU_02	21/Diciembre /2017 al 22/Diciembre/2017	Cyndi González Águila	**P** Pendiente	**C** Concluido C
Componente a probar:				
Determinar el volumen absoluto de los agregados				
Descripción:		**Pre-requisito:**		
Calcular el volumen absoluto de los agregados Mostrar el resultado en volumen de los agregados		Calcular el paso 7: Calcular el volumen absoluto de la pasta		
Datos entrada:	**Resultado obtenido:**	**Salida esperada:**	**Excepciones**	
Realizar la resta de el columen absoluto de los agregados que es igual a 1 menos el volumen absoluto de la pasta calculado en el paso 7	Calcular el volumen absoluto de los agregados	Calcular el volumen absoluto de los agregados	• No conexión con la base de datos • No haber realizado el paso anterior	
Observaciones:				

Figura 57. Formato Caso de Prueba Unitaria2 Metodología Scrum Sprint 4

En la ***Figura 58. Formato Caso de Prueba Unitaria3 Metodología Scrum Sprint 4*** el módulo a probar es el de determinar el porcentaje del agregado fino, el resultado obtenido presenta un error en la conexión por lo cual no se puede calcular la operación el módulo correspondiente

Logo	**Caso de prueba** "Prueba Unitaria"			
Nombre empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"			
Número del caso de prueba	**Fecha de ejecución**	**Responsable**	**Estado**	
PU_03	23/Diciembre /2017 al 26/Diciembre/2017	Cyndi González Águila	**P** Pendiente P	**C** Concluido
Componente a probar:				
Determinar el porcentaje del agregado fino				
Descripción:		**Pre-requisito:**		
Calcular el porcentaje del agregado fino		Calcular el paso 7: Calcular el volumen absoluto de la pasta		
Datos entrada:	**Resultado obtenido:**	**Salida esperada:**	**Excepciones**	
Seleccionar alguno de los valores deados de las siguientes tablas:Perfil del agregado , el tamaño máximo nominal, el modulo de fineza y el factor cemento	No se tiene conexión con la bd para Calcular el porcentaje del agregado fino	Calcular el porcentaje del agregado fino	• No conexión con la base de datos • No haber realizado el paso anterior	
Observaciones:				

Figura 58. Formato Caso de Prueba Unitaria3 Metodología Scrum Sprint 4

En la ***Figura 59. Formato Caso de Prueba Unitaria3 Metodología Scrum Sprint 4*** el módulo a probar es el de determinar el porcentaje del agregado fino, se realizó nuevamente la prueba para verificar que haya sido corregida por el desarrollador, el resultado obtenido fue la salida esperada

Logo.	**Caso de prueba** "Prueba Unitaria"			
Nombre empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"			
Número del caso de prueba	**Fecha de ejecución**	**Responsable**	**Estado**	
PU_03_01	26/Diciembre/2017	Cyndi González Águila	**P** Pendiente	**C** Concluido
				C
Componente a probar:				
Determinar el porcentaje del agregado fino				
Descripción:		**Pre-requisito:**		
Calcular el porcentaje del agregado fino		Calcular el paso 7: Calcular el volumen absoluto de la pasta		
Datos entrada:	**Resultado obtenido:**	**Salida esperada:**	**Excepciones**	
Seleccionar alguno de los valores deados de las siguientes tablas:Perfil del agregado , el tamaño máximo nominal, el modulo de fineza y el factor cemento	Calcular el porcentaje del agregado fino	Calcular el porcentaje del agregado fino	• No conexión con la base de datos • No haber realizado el paso anterior	
Observaciones:				

Figura 59. Formato Caso de Prueba Unitaria3_1 Metodología Scrum Sprint 4

En la ***Figura 60. Formato Caso de Prueba Unitaria4 Metodología Scrum Sprint 4*** el módulo a probar es el de determinar los volúmenes absolutos del agregado, el resultado obtenido presenta un error en la conexión por lo cual no se puede calcular la operación el módulo correspondiente

Logo	**Caso de prueba** "Prueba Unitaria"			
Nombre empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"			
Número del caso de prueba	**Fecha de ejecución**	**Responsable**	**Estado**	
PU_04	27/Diciembre /2017 al 28/Diciembre/2017	Cyndi González Águila	**P** Pendiente	**C** Concluido
				C
Componente a probar:				
Determinar los volúmenes absolutos del agregado				
Descripción:		**Pre-requisito:**		
Calcular el volumen absoluto de los agregados a) Se determinara el volumen absoluto del agregado fino , multiplicando el porcentaje del agregado fino por el volumen total de los agregados b) Se determinara el volumen absoluto del agregado grueso , restando al volumen total d elos agregados el volumen del agregado fino		Calcular el paso 7: Calcular el volumen absoluto de la pasta		
Datos entrada:	**Resultado obtenido:**	**Salida esperada:**	**Excepciones**	
Resultado del calculo volumen total de los agregados y el resultado del agregado fino	No se tiene conexión con la bd para Calculo del volumen absoluto de los agregados	Determinar los volúmenes absolutos del agregdo	• No conexión con la base de datos • No haber realizado el paso anterior	
Observaciones:				

Figura 60.Formato Caso de Prueba Unitaria4 Metodologia Scrum Sprint 4

En la ***Figura 61. Formato Caso de Prueba Unitaria4_1 Metodología Scrum Sprint 4*** el módulo a probar es el de determinar los volúmenes absolutos del agregado

Logo.	**Caso de prueba** "Prueba Unitaria"		
Nombre empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"		
Número del caso de prueba	**Fecha de ejecución**	**Responsable**	**Estado**
PU_04_01	28/Diciembre/2017	Cyndi González Águila	**P** Pendiente / **C** Concluido C
Componente a probar:			
Determinar los volúmenes absolutos del agregado			
Descripción:		**Pre-requisito:**	
Calcular el volumen absoluto de los agregados a) Se determinara el volumen absoluto del agregado fino , multiplicando el porcentaje del agregado fino por el volumen total de los agregados b) Se determinara el volumen absoluto del agregado grueso , restando al volumen total d elos agregados el volumen del agregado fino		Calcular el paso 7: Calcular el volumen absoluto de la pasta	
Datos entrada:	**Resultado obtenido:**	**Salida esperada:**	**Excepciones**
Resultado del calculo volumen total de los agregados y el resultado del agregado fino	No se tiene conexión con la bd para Calculo del volumen absoluto de los agregados	Determinar los volúmenes absolutos del agregdo	• No conexión con la base de datos • No haber realizado el paso anterior
Observaciones:			

Figura 61. Formato Caso de Prueba Unitaria4_1 Metodología Scrum Sprint 4

En la ***Figura 62. Formato Caso de Prueba Unitaria5 Metodología Scrum Sprint 4*** el módulo a probar es el de cálculo del contenido del agregado fino, el resultado obtenido presenta un error en la conexión por lo cual no se puede calcular la operación el módulo correspondiente

Logo	**Caso de prueba** "Prueba Unitaria"			
Nombre empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"			
Número del caso de prueba	**Fecha de ejecución**	**Responsable**	**Estado**	
PU_05	29/Diciembre /2017 al 02/Enero/2018	Cyndi González Águila	**P** Pendiente	**C** Concluido C
Componente a probar:				
Calculo del contenido del agregado fino				
Descripción:		**Pre-requisito:**		
Calcular el volumen absoluto de los agregados. Se multiplica el volumen absoluto de los agregados fino y grueso por su peso especifico respectivamente a) Se determina el peso seco del agregado fino, multiplicando su peso especifico por su volumen absoluto y luego por mil b) Se determina el peso seco del agregado grueso , multiplicando su peso especifico por su volumen absoluto y luego por mil		Calcular el paso 7: Calcular el volumen absoluto de la pasta		
Datos entrada:	**Resultado obtenido:**	**Salida esperada:**	**Excepciones**	
Resultado de volúmenes absolutos fino y grueso	No se tiene conexión con la bd para Calculo del contenido del agregado fino	Calculo del contenido del agregado fino	• No conexión con la base de datos • No haber realizado el paso anterior	
Observaciones:				

Figura 62.Formato Caso de Prueba Unitaria5 Metodología Scrum Sprint 4

En la ***Figura 63. Formato Caso de Prueba Unitaria5_1 Metodología Scrum Sprint 4*** el módulo a probar es el de calcular el contenido del agregado fino

Logo	**Caso de prueba** "Prueba Unitaria"			
Nombre empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"			
Número del caso de prueba	**Fecha de ejecución**	**Responsable**	**Estado**	
PU_05_01	06/Diciembre/2017	Cyndi González Águila	**P** Pendiente	**C** Concluido
				C
Componente a probar:				
Determinación de la relación agua/cemento				
Descripción:		**Pre-requisito:**		
Calculo de la relación agua/cemento Seleccionar la casilla determinar la relación agua/cemento Seleccionar un numero decimal o entero		• Calcular el paso 4: Selección del volumen del aire • Seleccionar la casilla:Determinar la relación agua/cemento		
Datos entrada:	**Resultado obtenido:**	**Salida esperada:**	**Excepciones**	
Seleccionar un numero ya dea con decimal o entero	Calculo de la relación agua/cemento	Calculo de la relación agua/cemento	• No conexión con la base de datos • No haber realizado el paso anterior	
Observaciones:				

Figura 63. Formato Caso de Prueba Unitaria5_1 Metodología Scrum Sprint 4

4. Reporte de errores

En la ***Figura 64. Formato Reporte de Error Prueba Metodología Scrum Sprint 4*** se describen cada uno de los errores encontrados en el módulo Walker, en los cuales se aplicaron nuevamente las pruebas una vez que el desarrollador realizo las correcciones con lo que se validó el funcionamiento del mismo.

Logo	**REPORTE DE ERROR PRUEBA: " Unitarias "**							
Nombre de la empresa:	**Nombre del proyecto: "** Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto **"**					**Número de Sprint:**		**4**
Número del caso de prueba	**Fecha de ejecución de prueba**	**Resultado de la prueba**	**Descripción del error**	**Nombre del responsable de ejecución de prueba**	**Prioridad**			**Versión**
					Alta	**Media**	**Baja**	
PU_03	23/Diciembre/2017 Al 26/Diciembre/2017	No se puede calcular el porcentaje del agregado fino	No se tiene conexión con la BD para calcular el porcentaje del agregado fino	Cyndi González Águila		X		1
PU_04	27/Diciembre/2017 Al 28/Diciembre/2017	No se puede determinarr los volúmenes absolutos del agregado	No se tiene conexión con la BD para determinar los volúmenes absolutos del agregado	Cyndi González Águila		X		1
PU_05	29/Diciembre/2017 Al 02/Enero/2018	No se puede determinar calculo del contenido del agregado fino	No se tiene conexión con la BD para determinar calculo del contenido del agregado fino	Cyndi González Águila		X		1

Figura 64. Formato Reporte de Error Prueba Metodología Scrum Sprint 4

5. Lista de control de pruebas

Se realizo con la finalidad de verificar que se siguieron los pasos del Framework en el sprint 4 que corresponde al método Walker aplicando metodologías agiles ver ***Figura 65. Formato Lista de Control Pruebas Metodología Scrum Sprint 4***

Logo	**LISTA DE CONTROL PRUEBAS UNITARIAS**			
Nombre de la empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"			
	Nombre tipo de prueba: "Unitaria**"**			
	Fecha evaluación:			
Código	**Elemento a verificar**	**Si**	**No**	**Observaciones**
PU1	¿Se realizaron las pruebas unitarias con un software?	**X**		
PU2	¿Cuál es el porcentaje de cobertura del sistema, una vez ejecutadas las pruebas?	**X**		
PU3	¿El funcionamiento de la prueba unitaria cumple con el diseño establecido?	**X**		
PU4	¿Existe un manejo de errores adecuado al finalizar las pruebas?	**X**		
PU5	¿Se cumple con el procedimiento de ejecución de la prueba?	**X**		
PU6	¿Se generan los reportes de errores correspondientes al finalizar la ejecución de pruebas?	**X**		
PU7	¿Se le da el seguimiento correspondiente a cada reporte de error?	**X**		
	María Janai Sánchez Hernández **Nombre y firma del evaluador**			

Figura 65. Formato Lista de Control Pruebas Metodología Scrum Sprint 4

6. Lecciones aprendidas

Se muestran en la ***Figura 66. Formato Lecciones Aprendidas Metodología Scrum Sprint 4*** la lección que aportaron de manera positiva en el sprint 4 al realizar pruebas en el desarrollo del método Walker

Logo	**LECCIONES APRENDIDAS**		
Nombre de la empresa:	**Nombre del proyecto** " Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto "		
Fecha del Proyecto	**Tipo de lección**		**Descripción de lección**
	Buena	**Mala**	
19/Diciembre/2017 al 02/Enero/2018	X		El contar con el formato calendario y el formato actas de reunión aumenta las posibilidades de que se tenga un mejor control en la planeación de tiempos y actividades para poder mostrar al cliente los avances en tiempo y forma y sobre todo con una mejor calidad

Figura 66.Formato Lecciones Aprendidas Metodología Scrum Sprint 4

4.1.4 Sprint 5 Método Fuller

1. Actas de reunión

Documento en el que se registró la entrega del método Fuller y el compromiso que se realizo fue la implementación del método Bolomey, ver ***Figura 67. Formato Actas de Reunión Metodología Scrum Sprint 5***

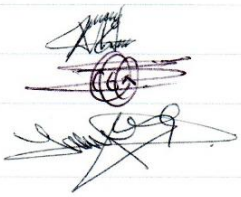
Logo	**ACTAS DE REUNIÓN**		
Nombre de la empresa:	**Nombre del proyecto: "**Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto**"**		
Reunión 6	**Revisión (Numero de Sprint):** 5	**Acta de reunión #:**	006
Lugar: Instituto Tecnológico de Apizaco		**Fecha:** 17/Enero/2018	
Asistentes(nombres):	**Cargo:**	**Firma:**	
1.Alfredo Xochitemol Cruz	Desarrollador		
2.Cyndi Gonzalez Águila	Tester		
3.María Janaí Sánchez Hernández	Scrum Master		
Temas tratados			
1 Método Fuller			
Compromisos de esta reunión	**Fecha máxima de cumplimiento**	**Cumplido**	
		Si	**No**
1 Implementación método Bolomey	23/Enero/2018	X	
Agenda			
Próxima Reunión:			
Lugar: Instituto Tecnológico de Apizaco	**Fecha**: 25/Enero/2018	**Hora:** 2:00 pm	

Figura 67. Formato Actas de Reunión Metodología Scrum Sprint 5

2. Calendario

Documento en el que se registró el tiempo programado de ejecución de las pruebas a realizar, así como su fecha de inicio y fin de cada uno de los componentes a probar que integran la funcionalidad del método Fuller, ver ***Figura 68. Formato Calendario Metodología Scrum Sprint 5***

Logo	**CALENDARIO**		
Nombre de la empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"	**Número de Sprint:**	5

Modulo o componentes a probar	**Nombre del responsable de ejecución de prueba**	**Tiempo programado de ejecución de prueba**	
		Inicio	**Fin**
Selección de la cantidad de agua	Cyndi Gonzalez Águila	03/Enero/2018	04/Enero/2018
Determinar el volumen del cemento	Cyndi Gonzalez Águila	05/Enero/2018	08/Enero/2018
Determinar la parábola de Füller	Cyndi Gonzalez Águila	09/Enero/2018	11/Enero/2018
Determinar el peso de los agregados	Cyndi Gonzalez Águila	12/Enero/2018	15/Enero/2018
Porporción en peso por kilo de cemento	Cyndi Gonzalez Águila	16/Enero/2018	17/Enero/2018

Figura 68. Formato Calendario Metodología Scrum Sprint 5

3. Casos de prueba

En la ***Figura 69. Formato Caso de Prueba Unitaria1 Metodología Scrum Sprint 5*** el módulo a probar es el de selección de la cantidad del agua

Logo	**Caso de prueba** "Prueba Unitaria"			
Nombre empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"			
Número del caso de prueba	**Fecha de ejecución**	**Responsable**	**Estado**	
PU_01	03/Enero/2018 al 04/Enero/2018	Cyndi González Águila	**P** Pendiente	**C** Concluido
				C
Componente a probar:				
Selección de la cantidad del agua				
Descripción:		**Pre-requisito:**		
Calcular la cantidda de agua Seleccionar casilla volumen de agua Insertar un numero entero		Calcular el paso 2: Selección del acentamiento •		
Datos entrada:	**Resultado obtenido:**	**Salida esperada:**	**Excepciones**	
Insertar un numero entero	Calculo del contenido de agua por metro cubico de concreto	Calculo del contenido de agua por metro cubico de concreto	• No conexión con la base de datos • No haber realizado el paso anterior • Insertar un numero decimal, solo calcula numero enteros	
Observaciones:				

Figura 69. Formato Caso de Prueba Unitaria1 Metodología Scrum Sprint 5

En la ***Figura 70. Formato Caso de Prueba Unitaria2 Metodología Scrum Sprint 5*** el módulo a probar es el de determinar el volumen del cemento

Logo	**Caso de prueba** "Prueba Unitaria"			
Nombre empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"			
Número del caso de prueba	**Fecha de ejecución**	**Responsable**	**Estado**	
PUF_02	05/Enero/2018 al 08/Enero/2018	Cyndi González Águila	**P** Pendiente	**C** Concluido C
Componente a probar:				
Determinar el volumen del cemento				
Descripción:		**Pre-requisito:**		
Mostrar el volumen del cemento obtenido de una división del volumen u itario del agua entre la relación agua/cemento		Calcular el paso 5 : Determinar la relación agua/cemento		
Datos entrada:	**Resultado obtenido:**	**Salida esperada:**	**Excepciones**	
Obtener en resultado del paso 3 y dividirlo entre el paso 5	Calculo del volumen del cemento	Calculo del volumen del cemento	• No conexión con la base de datos • No haber realizado el paso anterior	
Observaciones:				

Figura 70. Formato Caso de Prueba Unitaria2 Metodología Scrum Sprint 5

En la ***Figura 71. Formato Caso de Prueba Unitaria3 Metodología Scrum Sprint 5*** el módulo a probar es el de determinar la parábola de Fuller

Logo	**Caso de prueba** "Prueba Unitaria"			
Nombre empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"			
Número del caso de prueba	**Fecha de ejecución**	**Responsable**	**Estado**	
PUF_03	09/Enero/2018 al 11/Enero/2018	Cyndi González Aguila	**P** Pendiente	**C** Concluido
				C
Componente a probar:				
Determinar la parábola de Füller				
Descripción:		**Pre-requisito:**		
Calcular la parábola de Fuller Graficar la curva granulométrica a) agregado fino Insertar un valor de 0 a 100 en cada una de las celdas(si una celda queda vacia no se considera para el calculo) b) agregado grueso Insertar un valor de 0 a 100 en cada una de las celdas(si una celda queda vacia no se considera para el calculo) Seleccionar el numero deseado en la malla de referencia		Calcular el paso 8 : Determinar el volumen de los agregados		
Datos entrada:	**Resultado obtenido:**	**Salida esperada:**	**Excepciones**	
Agregar 15 datos del 0 al 100 en la tabla de agregado fino Agregar 15 datos del 0 al 100 en la tabla de agregado grueso Seleccionar el agregado ideal Selección del numero deseado para la malla de referencia	Calculo de la parábola de Fuller	Calculo de la parábola de Fuller	• No conexión con la base de datos • No haber realizado el paso anterior	
Observaciones:				

Figura 71. Formato Caso de Prueba Unitaria3 Metodología Scrum Sprint 5

En la ***Figura 72. Formato Caso de Prueba Unitaria4 Metodología Scrum Sprint 5*** el módulo a probar es el de determinar el peso de los agregados

Logo	**Caso de prueba** "Prueba Unitaria"			
Nombre empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"			
Número del caso de prueba	**Fecha de ejecución**	**Responsable**	**Estado**	
PU_04	12/Enero/2018 al 15/Enero/2018	Cyndi González Águila	**P** Pendiente	**C** Concluido C
Componente a probar:				
Determinar el peso de los agregados				
Descripción:		**Pre-requisito:**		
Calcular el volumen del cemento Se dibuja la parábola de Fuller 1. Introducir el agregado fino 2. Introducir el agregado grueso 3. Malla de referencia En la misma grafica se dibujan las curvas granulométricas se los agregados Se determina en la malla 1. Agregado fino 2. Agregado grueso 3. Agregado ideal Se calculan los valores de α y β Se calcula el volumen del agregado fino y gruesopor metro cubito		Calcular el paso : Calcular el volumen absoluto de la pasta		
Datos entrada:	**Resultado obtenido:**	**Salida esperada:**	**Excepciones**	
Introducir el agregado fino Introducir el agregado grueso Malla de referencia	Calculo del volumen del cemento	Calculo del volumen del cemento	• No conexión con la base de datos • No haber realizado el paso anterior	
Observaciones:				

Figura 72.Formato Caso de Prueba Unitaria4 Metodología Scrum Sprint 5

En la ***Figura 73. Formato Caso de Prueba Unitaria5 Metodología Scrum Sprint 5*** el módulo a probar es el de proporción en peso por kilo de cemento

Logo.	**Caso de prueba** "Prueba Unitaria"		
Nombre empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"		
Número del caso de prueba	**Fecha de ejecución**	**Responsable**	**Estado**
PU_05	16/Enero/2018 al 17/Enero/2018	Cyndi González Aguila	**P** Pendiente / **C** Concluido: C
Componente a probar:			
Proporción en peso por kilo de cemento			
Descripción:		**Pre-requisito:**	
Mostrar el resultado e de peso por kilo de cemento en una tabla		Calcular el paso 9: valores de diseño	
Datos entrada:	**Resultado obtenido:**	**Salida esperada:**	**Excepciones**
Cemento Agregado fino Agregado grueso Agua Obtenidos en los pasos anteriores	El calculo de la proporción en peso por kilo de cemento	El calculo de la proporción en peso por kilo de cemento	• No conexión con la base de datos • No haber realizado el paso anterior
Observaciones:			

Figura 73.Formato Caso de Prueba Unitaria5 Metodología Scrum Sprint 5

4. Lista de control de pruebas

Al realizar los elementos a verificar se comprueba que el proceso se realizó como se establece en el framework aplicando la metodología ágil ver **Figura *74. Formato Lista de Control Pruebas Metodología Scrum Sprint 5***

Logo	**LISTA DE CONTROL PRUEBAS UNITARIAS**			
Nombre de la empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"			
	Nombre tipo de prueba: "Unitaria**"**			
	Fecha evaluación:			
Código	**Elemento a verificar**	**Si**	**No**	**Observaciones**
PU1	¿Se realizaron las pruebas unitarias con un software?	**X**		
PU2	¿Cuál es el porcentaje de cobertura del sistema, una vez ejecutadas las pruebas?	**X**		
PU3	¿El funcionamiento de la prueba unitaria cumple con el diseño establecido?	**X**		
PU4	¿Existe un manejo de errores adecuado al finalizar las pruebas?	**X**		
PU5	¿Se cumple con el procedimiento de ejecución de la prueba?	**X**		
PU6	¿Se generan los reportes de errores correspondientes al finalizar la ejecución de pruebas?	**X**		
PU7	¿Se le da el seguimiento correspondiente a cada reporte de error?	**X**		
	María Janai Sánchez Hernández **Nombre y firma del evaluador**			

Figura 74. Formato Lista de Control Pruebas Metodología Scrum Sprint 5

5. Lecciones aprendidas

En este formato se incluyeron las causas por las cuales se dificulto el proceso de ejecución de pruebas del sprint 3 como se muestran en la ***Figura 75. Formato Lecciones Aprendidas Metodología Scrum Sprint 5***

Logo	**LECCIONES APRENDIDAS**		
Nombre de la empresa:	**Nombre del proyecto** " Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto "		
Fecha del Proyecto	**Tipo de lección**		**Descripción de lección**
	Buena	**Mala**	
03/Enero/2018 al 17/Enero/2018	X		El reutilizar código de los métodos anteriores evito que hubiera errores en las pruebas ejecutadas

Figura 75. Formato Lecciones Aprendidas Metodología Scrum Sprint 5

4.1.5 Sprint 6 Método Bolomey

1. Actas de reunión

En este documento se registró los temas tratados en la reunión, se realizó la entrega del método Bolomey y el compromiso que se realizo fue la implementación del método comparativo, ver ***Figura 76. Formato Actas de Reunión Metodología Scrum Sprint 6***

Logo	**ACTAS DE REUNIÓN**		
Nombre de la empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"		
Reunión 7	**Revisión (Numero de Sprint):** 6	**Acta de reunión #:** 007	
Lugar: Instituto Tecnológico de Apizaco		**Fecha:** 25/Enero/2018	
Asistentes(nombres):	**Cargo:**	**Firma:**	
1.Alfredo Xochitemol Cruz	Desarrollador		
2.Cyndi González Águila	Tester		
3. María Janaí Sánchez Hernández	Scrum Master		
Temas tratados			
1 Método Bolomey			
Compromisos de esta reunión	**Fecha máxima de cumplimiento**	**Cumplido**	
		Si	**No**
1 Implementación método Comparativo	30/Enero/2018	X	
Agenda			
Próxima Reunión:			
Lugar: Instituto Tecnológico de Apizaco	**Fecha**: 31/Enero/2018	**Hora:** 2:00 pm	

Figura 76.Formato Caso de Prueba1 Metodología Scrum Sprint 6

2. Calendario

En este documento se registró un orden cronológico de los módulos a testear de acuerdo a las fechas de desarrollo, las pruebas se empezaron a ejecutar a partir del 18 al 24 de noviembre del 2018. Ver ***Figura 77. Formato Calendario Metodología Scrum Sprint 6***

Logo	**CALENDARIO**		
Nombre de la empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"	**Número de Sprint:**	1

Modulo o componentes a probar	**Nombre del responsable de ejecución de prueba**	**Tiempo programado de ejecución de prueba**	
		Inicio	**Fin**
Determinar el factor cemento y volumen del cemento	Cyndi Gonzalez Águila	18/Enero/2018	19/Enero/2018
Determinar los valores de la curva de Bolomey	Cyndi Gonzalez Águila	22/Enero/2018	23/Enero/2018
Proporcion en peso por kilo de cemento	Cyndi Gonzalez Águila	24/Enero/2018	25Enero/2018

Figura 77. Formato Calendario Metodología Scrum Sprint 6

3. Casos de prueba

En la ***Figura 78. Formato Caso de Prueba Unitaria1 Metodología Scrum Sprint 6*** el módulo a probar es el de determinar el factor cemento y volumen del cemento

Logo.	**Caso de prueba** "Prueba Unitaria"			
Nombre empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"			
Número del caso de prueba	**Fecha de ejecución**	**Responsable**	**Estado**	
PU_01	18/Enero /2018 al 19/Enero /2018	Cyndi González Águila	**P** Pendiente	**C** Concluido
				C
Componente a probar:				
Determinar el factor cemento y volumen del cemento				
Descripción:		**Pre-requisito:**		
Calcular el volumen del cemento Mostrar el resultado de la división del volumen unitario de agua entre la relación agua/cemento		Calcular el paso 5: Determinar la relación, agua cemento		
Datos entrada:	**Resultado obtenido:**	**Salida esperada:**	**Excepciones**	
Obtener los resultados de: Volumen unitario del agua y la relación agua/cemento	Calculo del volumen del cemento	Calculo del volumen del cemento	• No conexión con la base de datos • No haber realizado el paso anterior	
Observaciones:				

Figura 78. Formato Caso de Prueba Unitaria1 Metodología Scrum Sprint 6

En la ***Figura 79. Formato Caso de Prueba Unitaria2 Metodología Scrum Sprint 6*** el módulo a probar es el de determinar los valores de la curva de Bolomey

Logo.	**Caso de prueba** "Prueba Unitaria"		
Nombre empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"		
Número del caso de prueba	**Fecha de ejecución**	**Responsable**	**Estado**
PU_02	22/Enero/2018 al 23/Enero/2018	Cyndi González Águila	**P** Pendiente / **C** Concluido C
Componente a probar:			
Determinar los valores de la curva de Bolomey			
Descripción:		**Pre-requisito:**	
Calcular el volumen del cemento Mostrar el resultado de la división del volumen unitario de agua entre la relación agua/cemento		Calcular el paso 5: Determinar la relación, agua cemento	
Datos entrada:	**Resultado obtenido:**	**Salida esperada:**	**Excepciones**
Obtener los resultados de: Volumen unitario del agua y la relación agua/cemento	Calculo del volumen del cemento	Calculo del volumen del cemento	• No conexión con la base de datos • No haber realizado el paso anterior
Observaciones:			

Figura 79. Formato Caso de Prueba Unitaria2 Metodología Scrum Sprint 6

4. Lista de control de pruebas

Se realizo con la finalidad de verificar que se siguieron los pasos del Framework en el sprint 6 que corresponde al método Bolomey aplicando metodologías agiles, de esta manera podemos verificar que el proceso se realizó como se establece en el framework aplicando la metodología ágil ver **Figura *80. Formato Lista de Control Pruebas Metodología Scrum Sprint 6***

Logo.	**LISTA DE CONTROL PRUEBAS**			
Nombre de la empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"			
	Nombre tipo de prueba: "Unitarias**"**			
	Fecha evaluación:			
Código	**Elemento a verificar**	**Si**	**No**	**Observaciones**
PU1	¿Se realizaron las pruebas unitarias con un software?	**X**		
PU2	¿Se cumple con el porcentaje de cobertura del sistema, una vez ejecutadas las pruebas?	**X**		
PU3	¿El funcionamiento de la prueba unitaria cumple con el diseño establecido?	**X**		
PU4	¿Existe un manejo de errores adecuado al finalizar las pruebas?	**X**		
PU5	¿Se cumple con el procedimiento de ejecución de la prueba?	**X**		
PU6	¿Se generan los reportes de errores correspondientes al finalizar la ejecución de pruebas?	**X**		
PU7	¿Se le da el seguimiento correspondiente a cada reporte de error?	**X**		
	María Janai Sáchez Hernández **Nombre y firma del evaluador**			

Figura 80. Formato Lista de Control Pruebas Metodología Scrum Sprint 6

5. Lecciones aprendidas

Se muestran en la ***Figura 81. Formato Lecciones Aprendidas Metodología Scrum Sprint 6*** que fue muy útil la planeación de algunos riesgos que pudieran presentarse en el periodo de ejecución de pruebas.

.

Logo	**LECCIONES APRENDIDAS**		
Nombre de la empresa:	**Nombre del proyecto** " Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto "		
Fecha del Proyecto	**Tipo de lección**		**Descripción de lección**
	Buena	**Mala**	
18/Enero/2018 al 25/Enero/2018	X		El tener un plan de riesgos facilita al equipo de pruebas cuando algo dificulta el avance en las actividades de ejecución de pruebas

Figura 81.Formato Lecciones Aprendidas Metodología Scrum Sprint 6

4.1.6 Sprint 7 Método comparativo

1. **Actas de reunión**

En este documento se registró los temas tratados en la reunión, se realizó la entrega del método comparativo ver ***Figura 82. Formato Actas de Reunión Metodología Scrum Sprint 7***

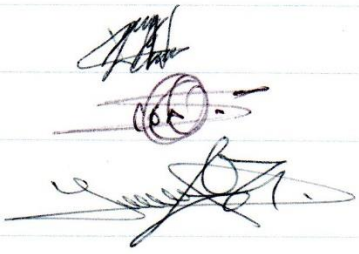
Logo	**ACTAS DE REUNIÓN**				
Nombre de la empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"				
Reunión	8	**Revisión (Numero de Sprint):**	7	**Acta de reunión #:**	008
Lugar:	Instituto Tecnológico de Apizaco		**Fecha:**	31/Enero/2018	
Asistentes(nombres):	**Cargo:**	**Firma:**			
1.Alfredo Xochitemol Cruz	Desarrollador				
2.Cyndi Gonzalez Águila	Tester				
3. María Janaí Sánchez Hernández	Scrum Master				
Temas tratados					
1	Método Comparativo				
Compromisos de esta reunión		**Fecha máxima de cumplimiento**	**Cumplido**		
			Si	**No**	
1			X		
Agenda					
Próxima Reunión:					
Lugar:	**Fecha**:	**Hora:**			

Figura 82.Formato Actas de Reunión Metodología Scrum Sprint 7

2. Calendario

En este documento se registró un orden cronológico de los módulos a testear de acuerdo a las fechas de desarrollo, las pruebas se empezaron a ejecutar a partir del 26 de enero al 02 de febrero del 2018. Ver ***Figura 83. Formato Calendario Metodología Scrum Sprint 7***

Logo	**CALENDARIO**		
Nombre de la empresa:	**Nombre del proyecto:** " Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto "	**Número de Sprint:**	7

Modulo o componentes a probar	**Nombre del responsable de ejecución de prueba**	**Tiempo programado de ejecución de prueba**	
		Inicio	**Fin**
Obtener los valores de los métodos(ACI,Walker, Fuller y Bolomey)	Cyndi Gonzalez Águila	26/Enero/2018	02/Febrero/2018

Figura 83. Formato Calendario Metodología Scrum Sprint 7

3. Casos de prueba

En la ***Figura 84. Formato Caso de Prueba Unitaria1 Metodología Scrum Sprint 7*** el módulo a probar es el de determinar los valores obtenidos por métodos, en el que se realizaron tanto pruebas unitarias como de integración, ya que en este módulo se muestran los resultados de los métodos calculados.

<table>
<tr><td>Logo.</td><td colspan="4">Caso de prueba “Prueba Unitaria”</td></tr>
<tr><td>Nombre empresa:</td><td colspan="4">Nombre del proyecto: “Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto”</td></tr>
<tr><td>Número del caso de prueba</td><td>Fecha de ejecución</td><td>Responsable</td><td colspan="2">Estado</td></tr>
<tr><td rowspan="2">PUF_01</td><td rowspan="2">26/Enero/2018
al
02/Febrero/2018</td><td rowspan="2">Cyndi González Águila</td><td>P
Pendiente</td><td>C
Concluido</td></tr>
<tr><td></td><td>C</td></tr>
<tr><td colspan="5">Componente a probar:</td></tr>
<tr><td colspan="5">Valores obtenidos por metodo</td></tr>
<tr><td colspan="2">Descripción:</td><td colspan="3">Pre-requisito:</td></tr>
<tr><td colspan="2">Obtener los valores de los 4 metodos ACI, Walker, Fuller y Bolomey,(el modulo compartarivo se podrá realizar con el calculo de uno, o mas métodos según se dese)</td><td colspan="3">Obtener el resultado calculado de los métodos ACI, Walker, Fuller y Bolomey (el modulo compartarivo se podrá realizar con el calculo de uno, o mas métodos según se dese)</td></tr>
<tr><td>Datos entrada:</td><td>Resultado obtenido:</td><td>Salida esperada:</td><td colspan="2">Excepciones</td></tr>
<tr><td>Obtener el resultado de los cualquiera de los métodos ACI, Walker, Fuller y Bolomey</td><td>Valores obtenidos por metodo</td><td>Valores obtenidos por metodo</td><td colspan="2">• No conexión con la base de datos
• No</td></tr>
<tr><td colspan="5">Observaciones:</td></tr>
<tr><td colspan="5"></td></tr>
</table>

Figura 84.Formato Caso de Prueba Unitaria1 Metodología Scrum Sprint 7

4. Lista de control de pruebas

Se realizo con la finalidad de verificar que se siguieron los pasos del Framework en el sprint 7 que corresponde al método comparativo, de esta manera podemos verificar que el proceso se realizó como se establece en el framework aplicando la metodología ágil ver **Figura *85. Formato Lista de Control Pruebas Metodología Scrum Sprint 7***

Logo	**LISTA DE CONTROL PRUEBAS UNITARIAS**			
Nombre de la empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto"			
	Nombre tipo de prueba: "Unitaria**"**			
	Fecha evaluación:			
Código	**Elemento a verificar**	**Si**	**No**	**Observaciones**
PU1	¿Se realizaron las pruebas unitarias con un software?	**X**		
PU2	¿Cuál es el porcentaje de cobertura del sistema, una vez ejecutadas las pruebas?	**X**		
PU3	¿El funcionamiento de la prueba unitaria cumple con el diseño establecido?	**X**		
PU4	¿Existe un manejo de errores adecuado al finalizar las pruebas?	**X**		
PU5	¿Se cumple con el procedimiento de ejecución de la prueba?	**X**		
PU6	¿Se generan los reportes de errores correspondientes al finalizar la ejecución de pruebas?	**X**		
PU7	¿Se le da el seguimiento correspondiente a cada reporte de error?	**X**		

María Janai Sánchez Hernández

Nombre y firma del evaluador

Figura 85.Formato Lista de Control Pruebas Metodología Scrum Sprint 7

5. Lecciones aprendidas

En este formato se incluyeron las causas por las cuales se dificulto el proceso de ejecución de pruebas del sprint 7 como se muestran en la ***Figura 86. Formato Lecciones Aprendidas Metodología Scrum Sprint 7***

Logo	**LECCIONES APRENDIDAS**		
Nombre de la empresa:	**Nombre del proyecto** " Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto "		
Fecha del Proyecto	**Tipo de lección**		**Descripción de lección**
	Buena	**Mala**	
26/Enero/2018 al 02/Febrero/2018	X		La comunicación fue una parte escencial para que se concluyera en tiempo y forma con la ejecución de pruebas

Figura 86.Formato Lecciones Aprendidas Metodología Scrum Sprint 7

4.1.7 Pruebas Ejecución de pruebas unitarias con la herramienta JUnit

En las siguientes Imágenes se muestran algunas pantallas de las pruebas automatizadas que se realizaron con la herramienta JUnit, para verificar que el sistema funciona correctamente. En la ***Figura 87. Prueba Unitaria Modulo Inicio de sesión*** *se* muestra parte de la ejecución de pruebas unitarias en el módulo inicio de sesión del Sprint 2 Tablas y Funciones matemáticas

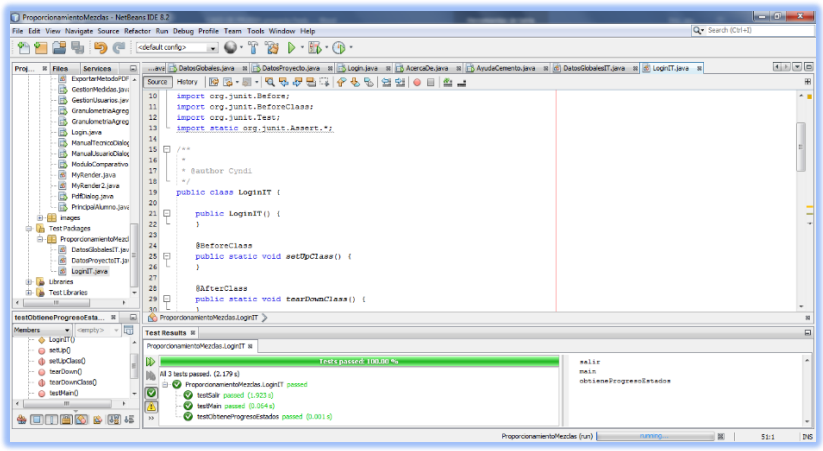


Figura 87. Prueba Unitaria Modulo Inicio de sesión

En la ***Figura 88. Prueba Unitaria Modulo Ingresar datos*** *se* muestra parte de la ejecución de pruebas unitarias en el módulo inicio de sesión del sprint 2 Tablas y Funciones matemáticas

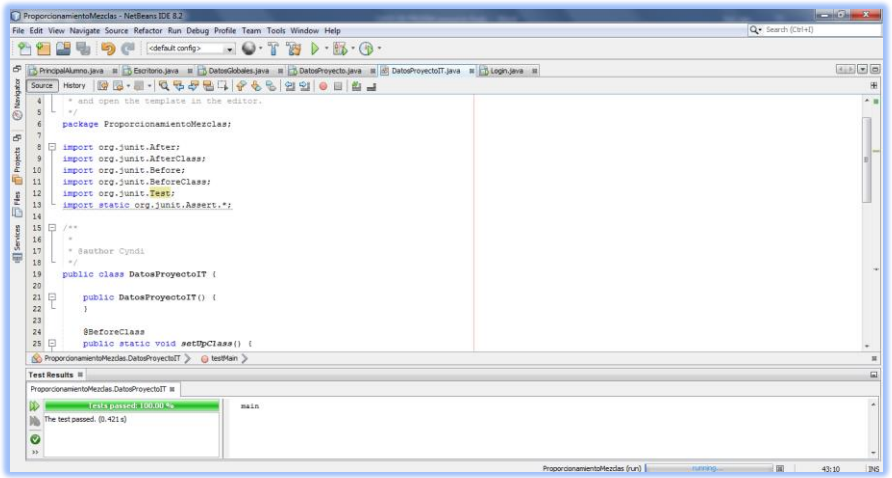


Figura 88. Prueba Unitaria Modulo Ingresar datos

En la ***Figura 89. Prueba Unitaria Modulo datos globales*** se muestra parte de la ejecución de pruebas unitarias en el módulo inicio de sesión del sprint 2 Tablas y Funciones matemáticas

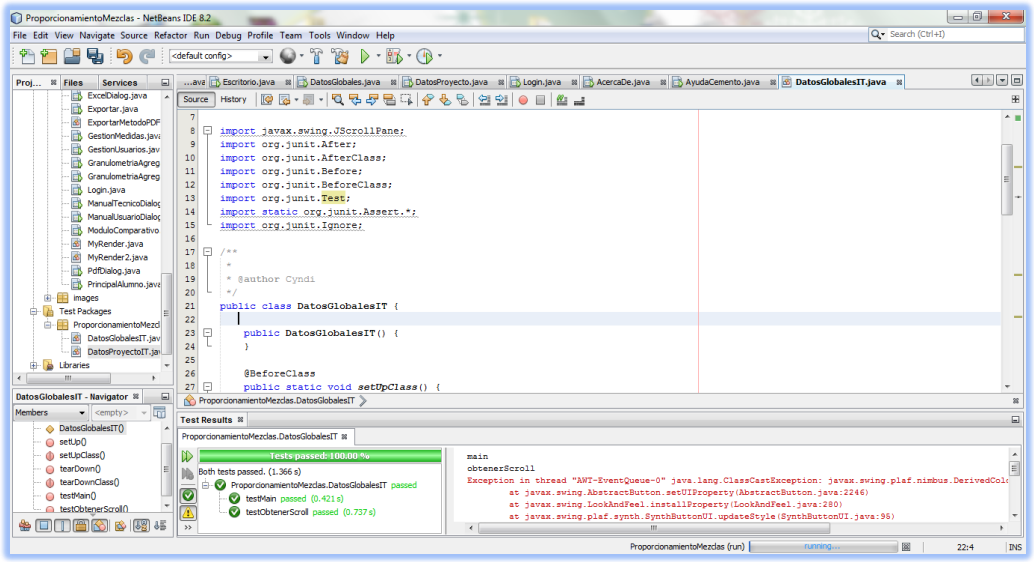


Figura 89. Prueba Unitaria Modulo datos globales se

En la ***Figura 90. Prueba Unitaria Método ACI y Figura 91. Prueba Unitaria Método ACI continuación*** se muestra parte de la ejecución de pruebas unitarias en el módulo inicio de sesión del sprint 3 Método ACI

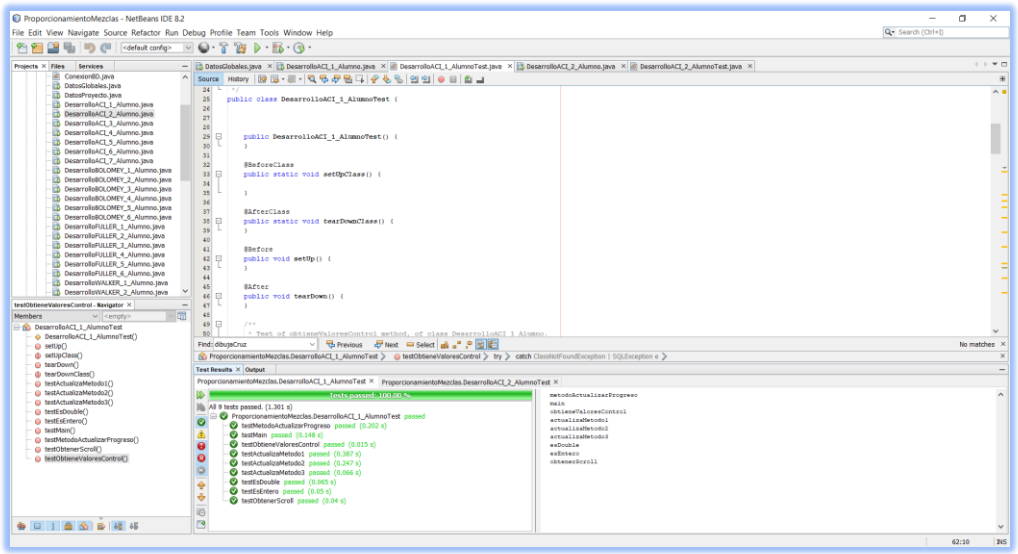


Figura 90. Prueba Unitaria Método ACI

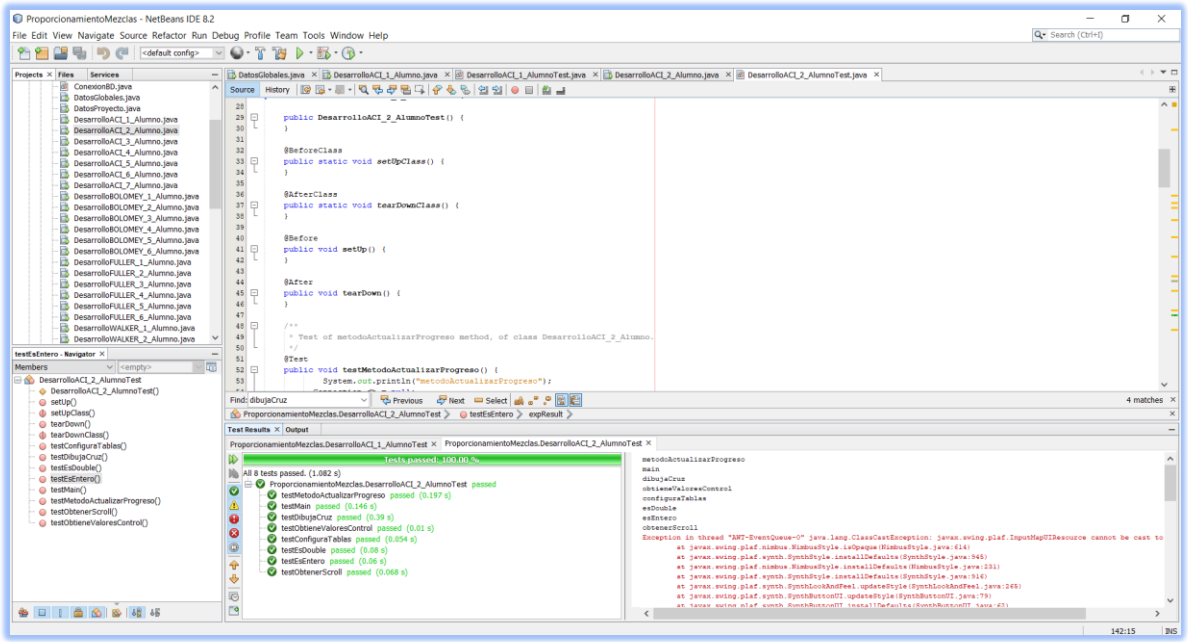


Figura 91. Prueba Unitaria Método ACI continuación

En la ***Figura 92. Prueba Unitaria Método Walker*** se muestra parte de la ejecución de pruebas unitarias en el método Walker Sprint 4

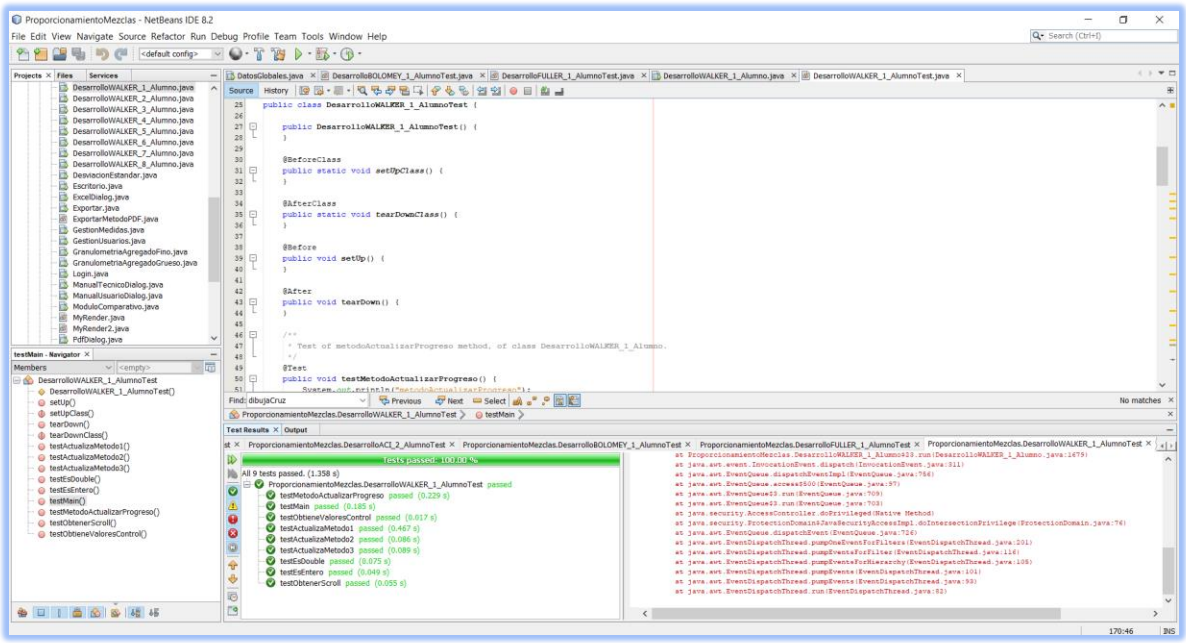


Figura 92. Prueba Unitaria Método Walker

En la ***Figura 70. Prueba Unitaria Método Fuller*** se muestra parte de la ejecución de pruebas unitarias en el método Fuller Sprint 5

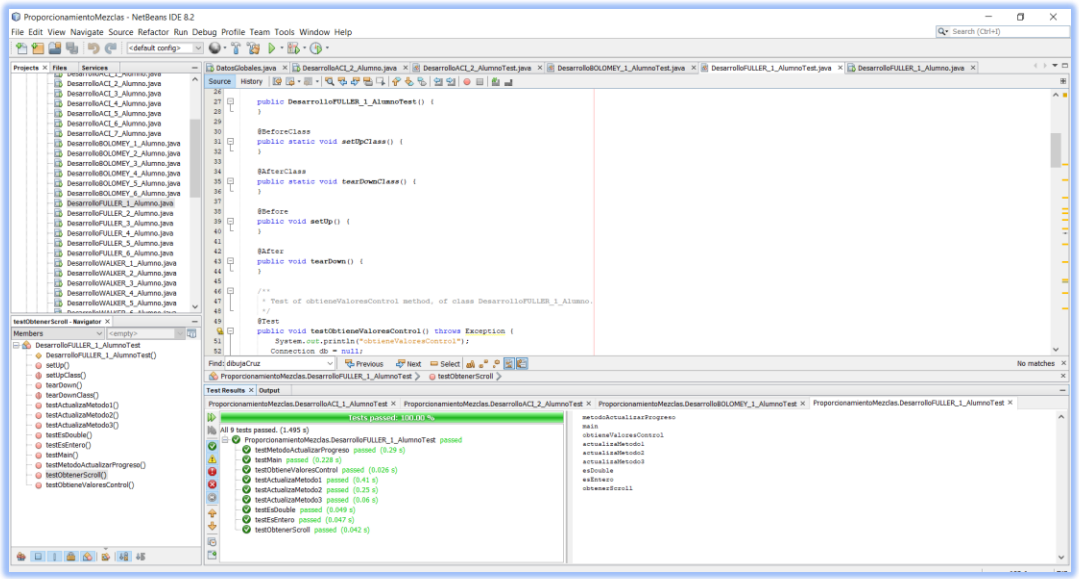


Figura 93.Prueba Unitaria Método Fuller

En la ***Figura 94. Prueba Unitaria Método Bolomey*** se muestra parte de la ejecución de pruebas unitarias en el método Bolomey Sprint 6

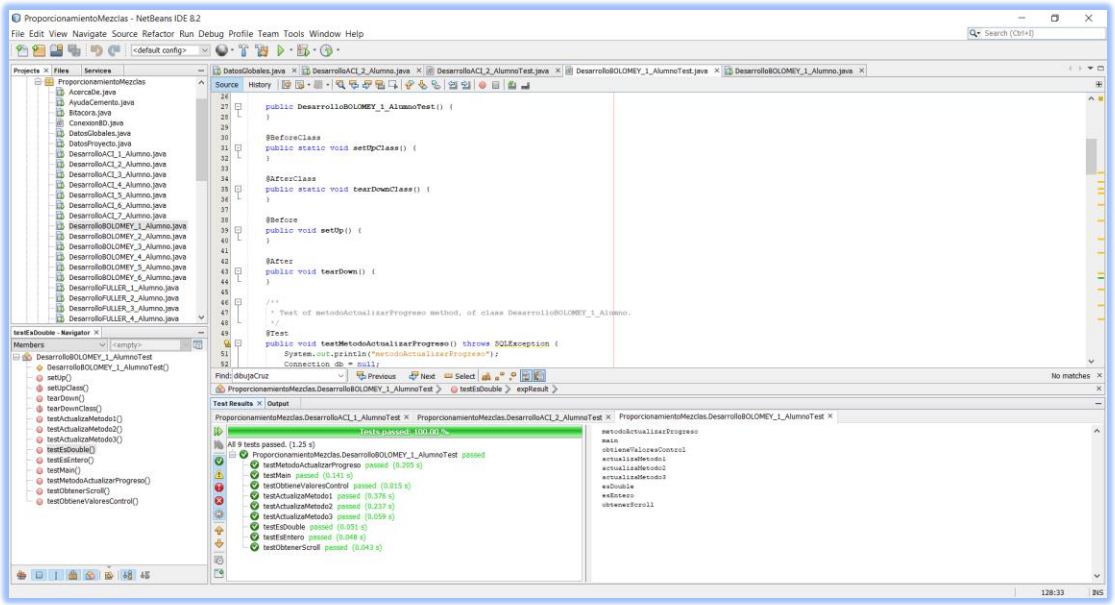


Figura 94. Prueba Unitaria Método Bolomey

En la ***Figura 95. Prueba Unitaria Método Bolomey*** se muestra parte de la ejecución de pruebas unitarias en el método Bolomey Sprint 6

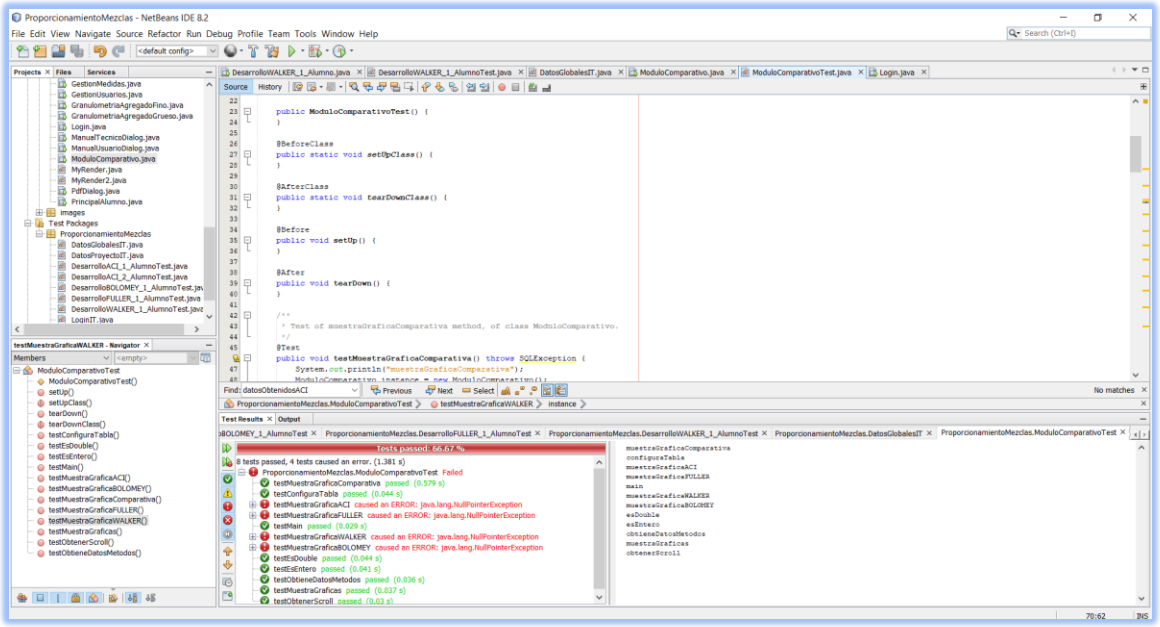


Figura 95. Prueba Unitaria Modulo Comparativo

4.2 *Framework* aplicando la metodología clásica (Cascada):

El sistema "estado29 Informacion responsable dia a dia", brinda informacion de los ultimos acontecimientos ocurridos en el estado de Tlaxcala, asi como noticias nacionales, entretenimiento y política. A continuación, se muestra el proceso de ejecución de pruebas funcionales aplicando la metodología clásica.

1. Matriz de Roles y Responsabilidades

En la ***Figura 96. Formato Matriz de Roles y Responsabilidades Metodología Clásica*** se registró el rol del responsable, así como las actividades que deben realizar, de acuerdo a la metodología clásica fueron necesarios 5 roles (Administrador de pruebas, Diseñador de pruebas, Ejecutores de pruebas, Administrador del sistema de pruebas y un Evaluador de pruebas)

Logo.	**MATRIZ DE ROLES Y ACTIVIDADES** **Nombre del proyecto:** "Desarrollo de una aplicación "		
Recursos humanos		**Rol**	**Responsabilidades**
Numero	**Nombre**		
1	Cyndi Gonzalez Aguila	Administrador de pruebas	Proporcionar atención especial al funcionamiento correcto de las tareas principales del sistema. Administrar los informes de incidencia Asegurar que cada miembro del equipo de pruebas este realizando las actividades correspondientes Verificar la lista de control
1	Cyndi Gonzalez Aguila	Diseñador de pruebas	Implementar los casos de la prueba a ejecutar. Realizar plan de pruebas Diseña y genera los casos de prueba Selecciona las herramientas para la automatización de pruebas requeridas Generar una lista de control
1	Cyndi Gonzalez Aguila	Ejecutores de pruebas	Realizar las pruebas Ejecuta las pruebas Generar los reportes Realiza el llenado de los reportes
1	Cyndi González Aguila	Administrador del sistema de pruebas	Asegurar el ambiente de prueba Administrar el manejo de pruebas del sistema. Administrar el calendario de pruebas Controlar el acceso de los integrantes del equipo de pruebas a los sistemas de prueba
1	María Janai Sanchez Hernández	Evaluador de pruebas	Verifica que las pruebas se hayan realizado de manera correcta Evalúa si las pruebas se realizaron de manera correcta

Figura 96. Formato Matriz de Roles y Actividades Metodología Clásica

2. Selección de las herramientas

En nuestro caso la herramienta que mejor se adapta a la empresa MBN es Selenium por las siguientes razones: i) cubre las pruebas funcionales , las cuales se definieron como una de las más importantes para la fábrica de software de MBN, ii) la licencia de la herramienta es abierta, lo cual permite utilizarla sin incrementar el costo de la ejecución de proyectos en MBN, y iii) se adapta de manera adecuada a proyectos desarrollados, los cuales son el tipo de proyectos que se desarrollan principalmente en MBN, ver ***Figura 97 . Formato Matriz de Roles y Responsabilidades Metodología Clásica***

Logo	**SELECCIÓN DE HERRAMIENTAS**					
Nombre de la empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación Web"					
Nombre de la herramienta	**Tipo de prueba**	**Licencia**	**Lenguaje**	**Requisitos**	**Características**	**Desventajas**
• Selenium	• Funcionales	• Open Source	• Java	• Mozilla Firefox	• Los scrips grabados pueden ser exportados en distintos lenguajes	• Las pruebas tienen que ser locales, no se pueden lanzar en remoto

Figura 97. Formato Matriz de Roles y Responsabilidades Metodología Clásica

3. Calendario

En la ***Figura 98. Formato Calendario Metodología Clásica***, se realizó la planeación de los módulos en los que se aplicaron las pruebas funcionales para tener un mejor control en los tiempos establecidos.

Logo	**CALENDARIO**		
Nombre de la empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación Web "		
Modulo o componentes a probar	**Nombre del responsable de ejecución de prueba**	**Tiempo programado de ejecución de prueba**	
		Inicio	**Fin**
Alta de noticia	Cyndi González Águila	19/Febrero/2018	19/Febrero/2018
Listado de noticias	Cyndi González Águila	20/Febrero/2018	21/Febrero/2018
Listado de politica	Cyndi González Águila	22/Febrero/2018	22/Febrero/2018
Listado de entretenimiento	Cyndi González Águila	23/Febrero/2018	23/Febrero/2018
Columna de opinión	Cyndi González Águila	26/Febrero/2018	28/Febrero/2018

Figura 98. Formato Calendario Metodología Clásica

4. Casos de pruebas funcionales

Es una parte fundamental del proceso de pruebas para poder contar con una unidad de medida, ya que se registran los diferentes escenarios con los que cuenta la aplicación web, ver ***Figura 99. Formato Caso de Prueba Funcional1 Metodología Clásica***

Logo.	**Caso de prueba** "Prueba Funcional"			
Nombre empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación Web"			
Número del caso de prueba	**Fecha de ejecución**	**Responsable**	**Estado**	
PF_01	19/Febrero/2018	Cyndi González Águila	**P** Pendiente	**C** Concluido
				C
Componente a probar:				
Alta de noticia				
Descripción:		**Pre-requisito:**		
Se permitirá dar de alta una noticia en el sistema		Usuario registrado		
Datos entrada:	**Resultado obtenido:**	**Salida esperada:**	**Excepciones**	
Noticia	Noticia publicada	Noticia publicada	El sistema muestra mensaje de error informando que no existe dicho mail El sistema muestra mensaje de error informando que la contraseña no es correcta	
Observaciones:				

Figura 99.Formato Casos de Prueba Funcional1 Metodología Clásica

Con este formato se validó que la aplicación funciona correctamente de acuerdo a los casos de uso realizados, ver ***Figura 100. Caso de Prueba Funcional2 Metodología Clásica***

Logo	**Caso de prueba** "Prueba Funcional"			
Nombre empresa:	**Nombre del proyecto:** "Desarrollo de una aplicación Web"			
Número del caso de prueba	**Fecha de ejecución**	**Responsable**	**Estado**	
PF_02	20/Febrero/2018 al 21/Febrero/2018	Cyndi González Águila	**P** Pendiente	**C** Concluido
				C
Componente a probar:				
Listado de noticias				
Descripción:		**Pre-requisito:**		
Se permitirá mostrar un listado de noticias mas recientes del estado en el sistema		Seleccion de una noticia		
Datos entrada:	**Resultado obtenido:**	**Salida esperada:**	**Excepciones**	
Seleccion de una noticia	Ver una noticia seleccionada, mostrando el título, descripción y las fotos ó el video si corresponde.	Ver una noticia seleccionada, mostrando el título, descripción y las fotos ó el video si corresponde.	No se debe exceder el tamaño maximo de video en MB.	
Observaciones:				

Figura 100. Formato Caso de Prueba Funcional2 Metodología Clásica

Se valido que el módulo columna de opinión funcionara, para asegurar la calidad de la aplicación se realizó el llenado de este formato ver ***Figura 101. Formato Caso de Prueba Funcional3 Metodología Clásica***

<table>
<tr><td>Logo</td><td colspan="4">Caso de prueba “Prueba Funcional”</td></tr>
<tr><td>Nombre empresa:</td><td colspan="4">Nombre del proyecto: “Desarrollo de una aplicación Web”</td></tr>
<tr><td>Número del caso de prueba</td><td>Fecha de ejecución</td><td>Responsable</td><td colspan="2">Estado</td></tr>
<tr><td rowspan="2">PF_03</td><td rowspan="2">20/Febrero/2018 al 21/Febrero/2018</td><td rowspan="2">Cyndi González Águila</td><td>P Pendiente</td><td>C Concluido</td></tr>
<tr><td></td><td>C</td></tr>
<tr><td colspan="5">Componente a probar:</td></tr>
<tr><td colspan="5">Columna de opinión</td></tr>
<tr><td colspan="2">Descripción:</td><td colspan="3">Pre-requisito:</td></tr>
<tr><td colspan="2">Se permitirá puntuar una noticia</td><td colspan="3">• El usuario debe proporcionar un email
• El usuario debe proporcionar un nombre</td></tr>
<tr><td>Datos entrada:</td><td>Resultado obtenido:</td><td>Salida esperada:</td><td colspan="2">Excepciones</td></tr>
<tr><td>Insertar un comentario</td><td>Comentar una noticia</td><td>Comentar una noticia</td><td colspan="2">No ingresar los campos obligatorios(marcados con *)</td></tr>
<tr><td colspan="5">Observaciones:</td></tr>
<tr><td colspan="5"></td></tr>
</table>

Figura 101. Formato Casos de Prueba Funcional3 Metodología Clásica

5. Lecciones aprendidas

Las lecciones aprendidas que se realizaron con la finalidad de tener una conclusión una vez que se han realizado las actividades correspondientes en el proceso de ejecución de pruebas funcionales En este formato se deben incluir las causas o posibles razones por las cuales alguna actividad no se haya logrado de acuerdo a lo planeado, o de lo contrario, un hecho significativo que ayudo a la culminación de dicha actividad o proyecto , para poder tener así una mejora en los siguientes sistemas a desarrollar.

Logo	**LECCIONES APRENDIDAS**		
Nombre de la empresa:	**Nombre del proyecto** "Desarrollo de una aplicación Web"		
Fecha del Proyecto	**Tipo de lección**		**Descripción de lección**
	Buena	**Mala**	
19Febrero/2018 al 28/Febrero/2018	X		El apoyo de algunos compañeros facilito la etapa de pruebas ya que nos proporcionan herramientas e información para la ejecución de las mismas

Figura 102. Formato Lecciones Aprendidas Metodología Clásica

4.2.1 Ejecución de pruebas funcionales con la herramienta Selenium

Una vez que fueron realizadas las acciones (alta de noticia, listado de noticias, listado de política, listado de entretenimiento columna de opinión) y tras haber grabado el script, se generó un código con esta estructura HTML es el lenguaje de grabación por defecto, ver ***Figura 103. Pruebas Funcionales*** y ***Figura 104. Prueba Funcional***

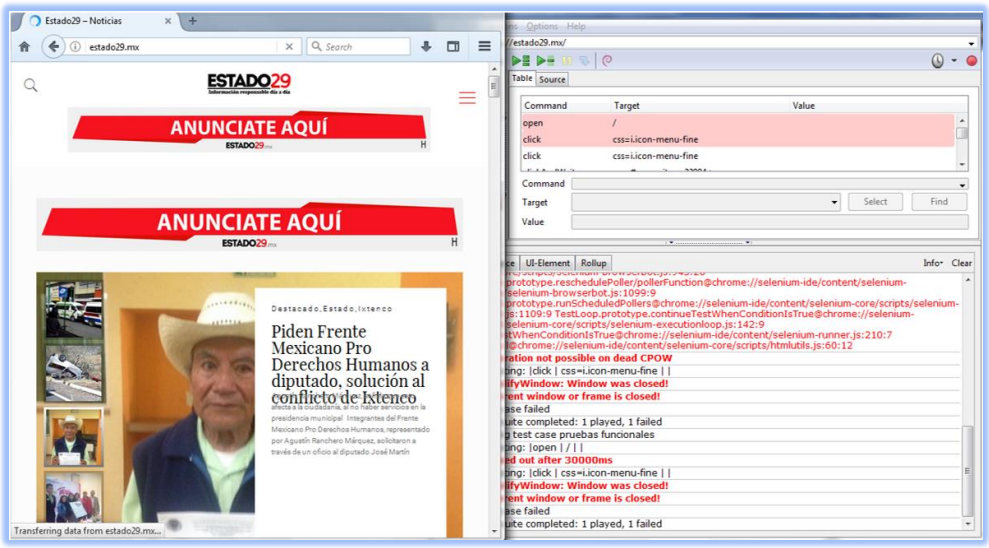


Figura 103. Pruebas Funcionales

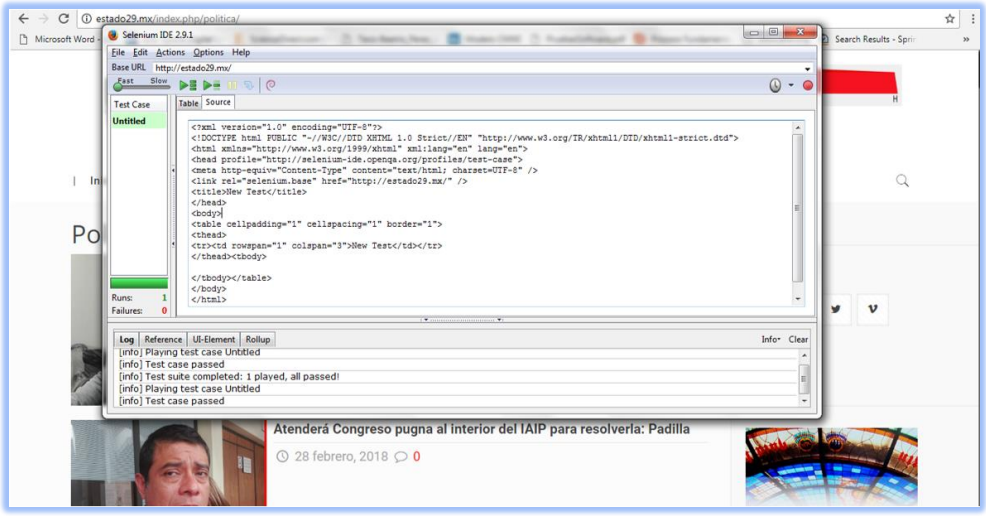


Figura 104. Prueba Funcional

En las siguientes ***Figura 105. Prueba Funcional Acción1, Figura 106. Prueba Funcional Acción2, Figura 107. Prueba Funcional Acción3 y Figura 108. Prueba Funcional Acción4*** se muestra parte del panel de código que se genera una en cada una de las acciones o escenarios indicados en los casos de pruebas

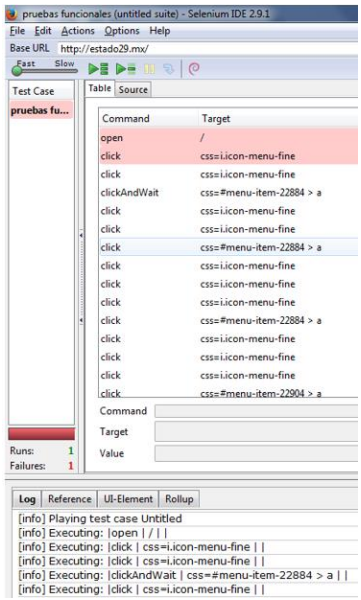


Figura 105.Prueba Funcional Acción1

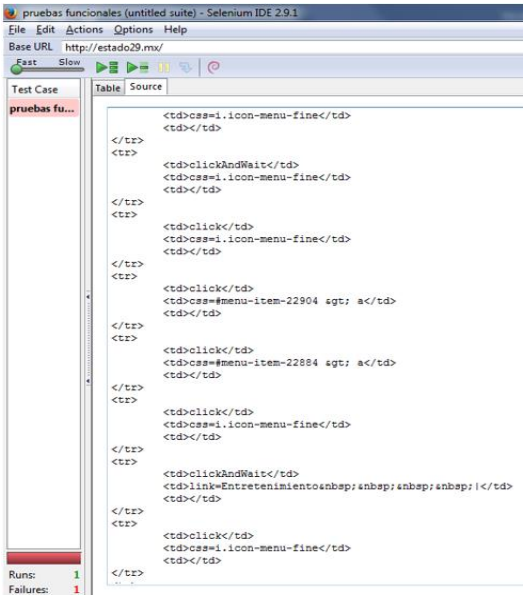


Figura 106.Prueba Funcional Acción2

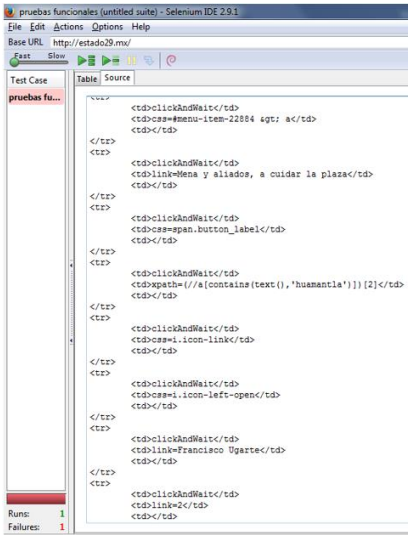


Figura 107. Prueba Funcional Acción3

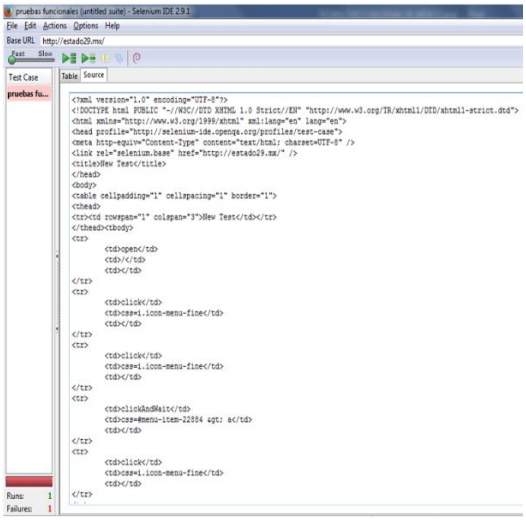
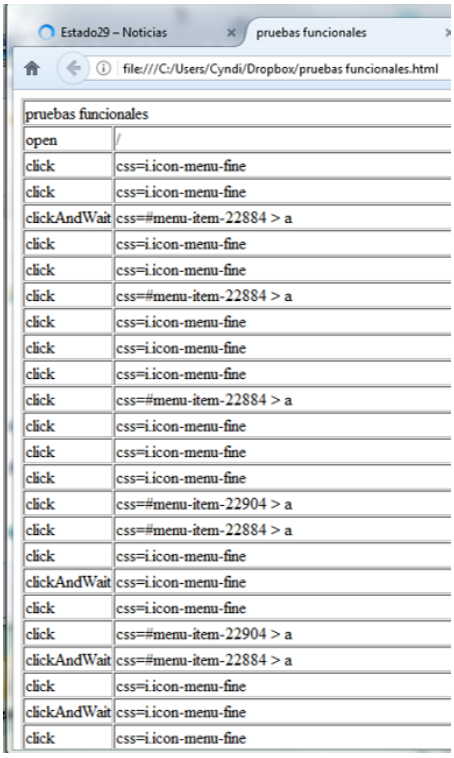


Figura 108. Prueba Funcional Acción4

En la ***Figura 109. Reporte Caso de Prueba Funcional*** se muestra parte del test realizado en la aplicación web estado29, que se almacenaron como HTML



pruebas funcionales	
open	/
click	css=i.icon-menu-fine
click	css=i.icon-menu-fine
clickAndWait	css=#menu-item-22884 > a
click	css=i.icon-menu-fine
click	css=i.icon-menu-fine
click	css=#menu-item-22884 > a
click	css=i.icon-menu-fine
click	css=i.icon-menu-fine
click	css=i.icon-menu-fine
click	css=#menu-item-22884 > a
click	css=i.icon-menu-fine
click	css=i.icon-menu-fine
click	css=i.icon-menu-fine
click	css=#menu-item-22904 > a
click	css=#menu-item-22884 > a
click	css=i.icon-menu-fine
clickAndWait	css=i.icon-menu-fine
click	css=i.icon-menu-fine
click	css=#menu-item-22904 > a
clickAndWait	css=#menu-item-22884 > a
click	css=i.icon-menu-fine
clickAndWait	css=i.icon-menu-fine
click	css=i.icon-menu-fine

Figura 109. Reporte Caso de Prueba Funcional

4.3 Conclusión

Las metodologías agiles nos brindan una mejora continua en el desarrollo del sistema, por medio de la colaboración del cliente y las iteraciones cortas, además de que represento un cambio drástico en el tiempo de desarrollo.

Al aplicar el plan de pruebas con la metodología ágil se obtuvo un aporte significativo ya que permitieron realizar proceso de pruebas organizado, teniendo un seguimiento y control de cada prueba.

La colaboración fue esencial para poder realizar el proceso de manera exitosa, se obtuvo una buena comunicación entre el equipo de desarrollo y el equipo de testing, obteniendo una mejor calidad en el sistema, además de la entrega del sistema se realizó en tiempo y forma.

En cuanto a la aplicación de la metodología clásica, el proceso de pruebas además de que se realizó al final de la codificación, pudimos observar que los requerimientos fueron muy constantes sin importar que el tamaño o complejidad del sistema fueron simples, permitiendo tener una estructura ordenada, aunque una de las desventajas que encontramos es que los errores se encontraron y corrigieron una vez finalizada la codificación por lo cual implico más tiempo en la corrección de los mismos.

Capítulo 5

Conclusiones

5 Conclusiones

En este Capítulo se presentan las conclusiones obtenidas de la aplicación del *framework* para metodologías agiles y metodologías clásicas. Además, se presentan los trabajos que se pueden seguir realizando tomando como referencia lo presentado anteriormente.

En este trabajo se realizó un estudio para desarrollar una propuesta de *framework* alineado a dos estándares de calidad CMMI y MOPROSOFT para realizar pruebas de software para mejorar la calidad de los productos en una empresa.

En cuanto a CMMI los proyectos de desarrollo de software se realizaron conforme a una gestión disciplinada a través de políticas organizacionales, y en base a procesos que planifican, monitorean y ejecutan el desarrollo de proyectos de software y Moprosoft, entre las cuáles se encuentra DMS - "Fábrica de Software" que desarrolla proyectos de Software a la medida para diversos clientes. Se tuvo un proceso de mejora en la ejecución de las pruebas, ya que los formatos diseñados y utilizados se presentaron como una herramienta de apoyo y soporte a cada una de las tareas o actividades asignadas a los diferentes roles, dependiendo de las necesidades encontradas en los sistemas a probar.

Para lograr la mejorara en la calidad se consideró la definición de un proceso detallado (framework) para el área de pruebas de software y el uso de herramientas de software que ayuden al proceso de ejecución de dicho proceso. El método utilizado fue el siguiente:

1. Se realizó una investigación documental acerca del tema de pruebas de software a fin de aclarar aspectos que ayuden a diseminar el conocimiento entre los integrantes de DMS-MBN. ***Ver sección 2.2 Pruebas de software***
2. Se realizó una investigación de cuáles son las herramientas de software más usadas, se definieron los elementos de comparación entre ellos como por ejemplo el tipo de licencia, funcionamiento, desventajas, entre otros y se realizaron pruebas de uso de cada herramienta a fin de valorarlas. Es importante mencionar que las herramientas comparadas fueron open source, esto con la finalidad de no incrementar los costos en los sistemas desarrollados. ***Ver anexo F. Articulo***
3. Finalmente, se discutieron internamente los resultados de la valoración de herramientas y se definió cuáles cubren de mejor manera las necesidades de DMS.

Las herramientas seleccionadas para la automatización de las pruebas fueron dos, JUnit y Selenium, ya que cubren pruebas unitarias y de integración, además de su licencia que como ya se menciono es abierta y se adapta de acuerdo al proyecto desarrollado en Java.

Con JUnit se obtuvieron reportes de cada una de los métodos probados, ya sea que el método de la clase pasó exitosamente la prueba o en caso contrario un fallo en el método correspondiente a probar, como se mostró en el Capítulo 4. Esta herramienta se utilizó en la ejecución de dos tipos de pruebas (unitarias e integración), bajo un marco de desarrollo ágil(Scrum) utilizados en el sistema "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto". ***Ver sección 4.1.7 Pruebas ejecución de pruebas unitarias con la herramienta JUnit***

Otra de las herramientas seleccionadas fue Selenium, ya que es una herramienta que permite el uso de diferentes API´s en diferentes lenguajes por ejemplo Java, permite realizar una grabación de las pruebas funcionales, con este servicio se consigue obtener una batería de pruebas automatizadas que podrán ser utilizadas cuando sea necesario repetir las pruebas, y su licencia es de código abierto.

Selenium permitió realizar pruebas funcionales en el sistema “estado29 Información responsable día a día”,ya que es una aplicación basada en pruebas web. Al realizar la ejecucion de las pruebas se obtuvo como resultado una grabacion de las pruebas, y una vez encontrados los errores en el sistema permitio realizar la edición de manera manual para la corrección de los mismos de una forma más fácil. ***Ver sección 4.2.1 framework aplicando metodología clásica***

El proceso de ejecución de pruebas de acuerdo a lo propuesto en ambas aplicaciones del framework resulto una guía útil para realizar la ejecución de los diferentes tipos de pruebas, al mismo tiempo se aumentó la calidad del sistema, esto al encontrar errores en el mismo, por lo cual se puede concluir que con el diseño del framework alineado a dos estándares de calidad CMMI y Moprosoft aplicado a dos metodologías de desarrollo ágil(Scrum) y clásica(Cascada) se reducen los defectos del software antes de la liberación, proporcionando al cliente una mejor calidad y pronta satisfacción.

5.1 Trabajo futuro

1. Se propone la ejecución de los diferentes tipos de pruebas dependiendo las necesidades de cada sistema para que se tenga una mejor cobertura y calidad del sistema, ya que en este caso solo se realizaron pruebas unitarias(código), de integración y funcionales.
2. Realizar la ejecución de las pruebas con otras herramientas que sean open source (Mockito, Cactus, JMeter, Soapui) por mencionar algunas, para mantener los costos del sistema, ya que en este caso solo se utilizaron dos herramientas de software libre, JUnit y Selenium.
3. Implementar un plan de capacitación para el equipo de Tester en cuanto al uso de las diferentes herramientas a utilizar, para que a mediano plazo se pueda brindar una creciente calidad en el sistema desarrollado.

Bibliografía

Kuhn, D., Wallace, D., & Gallo Jr, A. (2004). Software fault interactions and implications for software testing. *IEEE*.

Abad Londoño, J. H. (06 de abril de 2005). Obtenido de http://ing-sw.blogspot.mx/2005/04/tipos-de-pruebas-de-software.html

Argota Vega, I. E. (2007). *Ilustrados*. Obtenido de http://www.ilustrados.com/tema/12008/Estudio-frameworks-para-desarrollo-pruebas-software.html

Beth Chrissis , M., Konrad , M., & Shrum, S. (06 de 05 de 2009). Obtenido de http://www.sei.cmu.edu/library/assets/cmmi-dev-v12-spanish.pdf

Ble, C. (2010-2013). *LIBROSWEB*. Obtenido de http://librosweb.es/libro/tdd/capitulo_1/modelo_en_cascada.html

Campo, G. D. (2009). Obtenido de http://www.ucasal.edu.ar/htm/ingenieria/cuadernos/archivos/4-p101-Campo.pdf

Cervantes, H. (2005). *Proceso unificado*. Obtenido de http://www.humbertocervantes.net/cursos/rup/junit.html

Christa, S., & Suma, V. (2016). An analysis on the significance of ticket analytics and defect analysis from software quality perspective. *International Conference on Inventive Computation Technologies (ICICT)*, 1 - 5.

Coronel , G. (25 de febrero de 2013). *Desarrollo Web*. Obtenido de http://desarrollandowebapps.blogspot.mx/2013/02/defectos-del-software-y-sus-causas.html

Cristian. (01 de marzo de 2003). *Los Muchachos Del Bonsai*. Obtenido de http://losmuchachosdelbonsai.blogspot.mx/2013/03/pruebas-unitarias-y-junit.html

Cruzes, D. S., Moe, N. B., & Dybå, T. (2016). Communication between Developers and Testers in Distributed Continuous Agile Testing. *IEEE 11th International Conference on Global Software Engineering* , 59 - 68.

De Freitas Vieira, C., Benavidez Torres, J. F., & Irumbe, É. C. (12 de DICIEMBRE de 2012). *Calidad, Métricas del Producto y Proceso de Pruebas de Software*. Obtenido de http://uptaprocesodepruebasycalidadymetricas.blogspot.mx/2012/12/pruebas-de-software.html

Díaz Vivar, M. R., Miranda, M. G., & Molina, S. E. (s.f.). Obtenido de http://www.unpa.edu.ar/sites/default/files/descargas/Administracion_y_Apoyo/Materiales/2016/T205/GESTION%20GENERAL/PRUEBAS/PAS_PLAN%20DE%20GESTION%20DE%20PRUEBA_V1.0.pdf

García-Fernández, M. (2016). Influencia de la gestion de la calidad en los resultados de innovación a través de la gestión del conocimiento.

ingenieria de software. (2005). Obtenido de http://www.humbertocervantes.net/cursos/ingsoft/practica5/practica5.html

Inteco, I. N. (junio de 2008).

Iván Ruiz-Rube, J. M. (2017). A framework for software process deployment and evaluation. *ScienceDirect*.

Madrid, V. J. (2008). *AdictosAlTrabajo.com*. Obtenido de https://www.adictosaltrabajo.com/tutoriales/selenium-ide/

Marco Palomino, A. D. (2016). Agile Practices Adoption in CMMI Organizations: A.

Ministerio de comunicaciones, R. d. (Diciembre de 2008). Obtenido de www.contraloriabogota.gov.co/.../ANEXO%2013%20MODELO%20PLAN%20DE%...

Ocampo Acosta , A., & Correa Tapasco, L. M. (2011). Obtenido de http://repositorio.utp.edu.co/dspace/bitstream/handle/11059/2470/0057565O15i.pdf;jsessionid=7B5A2B5194344AFF045DA68301F8C404?sequence=1

Pablo, J. (s.f.). *el Conspirador, un blog de volcanes, exploraciones y tecnología*. Obtenido de https://www.elconspirador.com/2013/08/19/modelos-de-desarrollo-de-software/

proyectos agiles.org. (s.f.). Obtenido de https://proyectosagiles.org/que-es-scrum/

Sánchez, J. (29 de Septiembre de 2006). *jordisan.net*. Obtenido de http://jordisan.net/blog/2006/que-es-un-framework/

Sommerville, I. (2011). *Ingenieria del Software , Novena edicion.* Mexico: Pearson Educación.

Valenzuela, J., & Pavlich-Mariscal, J. (2014). Hacia un Modelo de madurez para apoyar el Desarrollo de software Dirigido por Modelos. *IEEE*.

Velázquez, E. M. (s.f.). *El camino a un mejor programador*. Recuperado el 26 de Junio de 2017, de https://emanchado.github.io/camino-mejor-programador/html/ch06.html

Ventura Miranda, M. T., & Peñaloza Baez, M. (27 de Noviembre de 2008). *computo academico unam*. Obtenido de http://www.enterate.unam.mx/Articulos/2006/marzo/moprosoft.htm

Anexos

A. Carta de no inconveniencia para realizar pruebas unitarias y de integración

Carta de no inconveniencia

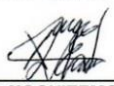
Apizaco Tlaxcala a 28 de Mayo del 2018

Asunto: No Inconveniencia

Por medio de la presente me permito informar a usted que no existe ningún inconveniente para que la C. Cyndi González Águila alumna de Maestría en Sistemas Computacionales del Instituto Tecnológico de Apizaco con número de control M08370662, haga uso del sistema vinculado a la tesis titulada "Desarrollo de una aplicación didáctica para el diseño y evaluación de mezclas de concreto", para la realización de pruebas unitarias y de integración con el framework alineado a los estándares de calidad CMMI y Moprosoft de la misma manera solicito se me haga entrega de los resultados obtenidos de dichas pruebas.

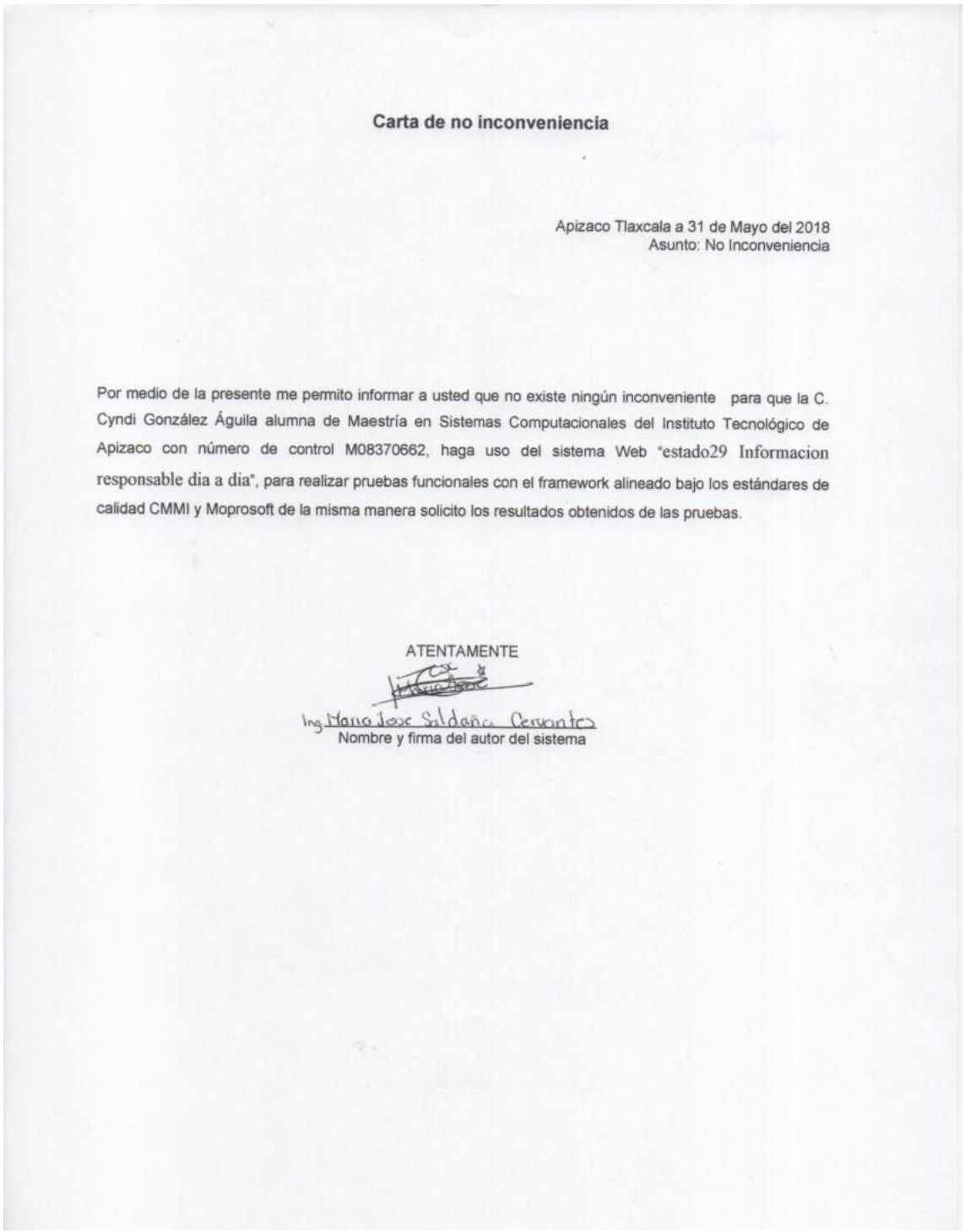
Sin más por el momento, agradezco la atención que se brinda a la presente.

ATENTAMENTE



ALFREDO XOCHITEMOL CRUZ
Autor del Software

B. Carta de no inconveniencia para realizar pruebas funcionales



Carta de no inconveniencia

Apizaco Tlaxcala a 31 de Mayo del 2018
Asunto: No Inconveniencia

Por medio de la presente me permito informar a usted que no existe ningún inconveniente para que la C. Cyndi González Águila alumna de Maestría en Sistemas Computacionales del Instituto Tecnológico de Apizaco con número de control M08370662, haga uso del sistema Web "estado29 Informacion responsable dia a dia", para realizar pruebas funcionales con el framework alineado bajo los estándares de calidad CMMI y Moprosoft de la misma manera solicito los resultados obtenidos de las pruebas.

ATENTAMENTE

Ing. Maria Jose Saldaña Cervantes
Nombre y firma del autor del sistema

C. Carta de liberación de estancia

MBN Miracle Business Network S.A de C.V.
Hacemos de la tecnología, su mejor aliado de negocio

Asunto: CARTA DE LIBERACIÓN

Tlaxcala, Tlax., 11 de septiembre del 2017

Mtro. Felipe Pascual Rosario Aguirre
DIRECTOR DEL INSTITUTO TECNOLÓGICO DE APIZACO
Con atención a:
Dr. José Federico Casco Vázquez
JEFE DEL DEPARTAMENTO DE DIVISÓN DE ESTUDIOS
DE POSGRADO E INVESTIGACIÓN
P R E S E N T E

Me permito comunicar a usted, que la **Lic. Cyndi González Aguila**, estudiante de la Maestría en Sistemas Computacionales en el Tecnológico Nacional de México, Instituto Tecnológico de Apizaco, con número de matrícula **M08370662,**ha concluido una **Estancia Técnica**, realizando el proyecto de **"Framework para pruebas de software alineado a los estándares de calidad CMMI y MOPROSOFT".,** durante el período comprendido del 01 de Febrero del 2017 al 17 de Agosto del 2017, los días Lunes y Jueves con un horario de 9:00 a 18:00 horas, teniendo como jefes inmediatos al Drhp. Alberto Portilla Flores e Ing. Uriel Diaz Austria.

Sin más por el momento, se extiende la presente carta de liberación para los efectos correspondientes.

MBN
Miracle Business Network, S.A de C.V.
Hacemos de la tecnología, su mejor aliado de negocio.

11 SEP 2017

R.F.C.MBN060316RI9
Calle 37 No. 216 B La Loma Xicohténcatl
Tlaxcala, Tlax. C.P. 90070 Tel. 01 246 416 45 08

ATENTAMENTE

Lic. Vidalia Flores Mojica

Responsable de Recursos Humanos y Ambiente de Trabajo.

Calle 37 # 216-B
La Loma Xicohténcatl, Tlaxcala, C.P. 90062

www.mbn.com.mx
Tel: 01 (246) 41 64508

V11.0, CARTADELIBERACION-PRACTICAS/ESTADIAS-RHAT

Pág. 1 de 1

D. Carta de satisfacción de estancia

MBN Miracle Business Network S.A de C.V.
Hacemos de la tecnología, su mejor aliado de negocio

Tlaxcala, Tlax., 11 de septiembre del 2017
Asunto: Constancia de satisfacción.

Mtro. Felipe Pascual Rosario Aguirre
DIRECTOR DEL INSTITUTO TECNOLÓGICO DE APIZACO
Con atención a:
AT'N: Dr. José Federico Casco Vázquez
JEFE DEL DEPARTAMENTO DE DIVISÓN DE ESTUDIOS
DE POSGRADO E INVESTIGACIÓN
P R E S E N T E:

Sirva la presente para enviarle un cordial saludo y notificarle que posterior a la recepción del proyecto de tesis de la **Lic. Cyndi González Aguila**, alumna de la Maestría en Sistemas Computacionales, con número de matrícula **M08370662,** de la institución que usted destacadamente dirige, se incluyó con el proyecto que lleva como título:

"Framework para pruebas de software alineado a los estándares de calidad CMMI y MOPROSOFT".

Siendo este desarrollado bajo la dirección de la M. en C. Maria Guadalupe Medina Barrera catedrático de la citada maestría.

En virtud de que sea cubierto satisfactoriamente los objetivos establecidos del citado proyecto.

Tenemos a bien dar constancia de que dicho proyecto de tesis cubre y satisface las expectativas planteadas al inicio de este proyecto.

Sin más por el momento, se extiende la presente carta de satisfación para los efectos correspondientes.

MBN
Miracle Business Network, S.A de C.V.
Hacemos de la tecnología, su mejor aliado de negocio.

ATENTAMENTE

11 SEP 2017

R.F.C.MBN060316RI9
Calle 37 No. 216 B La Loma Xicohténcatl
Tlaxcala. Tlax. C.P. 90070 Tel. 01 246 41[illegible]

Lic. Vidalia Flores Mojica

Responsable de Recursos Humanos y Ambiente de Trabajo.

Calle 37 # 216-B
La Loma Xicohténcatl, Tlaxcala, C.P. 90062

www.mbn.com.mx
Tel: 01 (246) 41 64508

V11.0, CARTA DE SATISFACCIÓN-RHAT

Pág. 1 de 1

E. Reconocimiento de publicación Articulo

F. Articulo

Bits & Software Magazine (2017) 1(1): 1-10.
ISSN: En trámite

1er CONGRESO DE INGENIERÍA EN SISTEMAS COMPUTACIONALES Y APLICACIONES (CISCA 2017)

Análisis Comparativo de Herramientas para hacer Pruebas de Software en una PYME de TI

C. González-Águila, A. Portilla-Flores y M. G. Medina-Barrera

Resumen. En este artículo se presenta un análisis de herramientas para la ejecución de pruebas de software. Nosotros consideramos una prueba de software a el proceso que demuestra que un programa hace lo que se intenta que haga, así como descubre los posibles defectos de dicho programa antes de usarlo. Las pruebas de software son muy importantes para el desarrollo exitoso del proyecto de desarrollo de software, sin embargo, se les presta poca atención o bien se implementan de manera artesanal y poco metódica. En nuestro trabajo se toma en cuenta tres tipos de pruebas: pruebas unitarias, pruebas de integración y pruebas funcionales. Nuestra aportación es un análisis de características sobre herramientas de software libre, con la finalidad de hacer una selección de la herramienta que más se adapte a las necesidades de una empresa PYME, en nuestro caso MBN, y que sea la base para la implementación de un área de pruebas.

Palabras clave: (Pruebas de software, Herramientas para pruebas y Modelos de calidad)

Abstract. In this paper an analysis of tools for the execution of tests of software is presented. We consider a test of software as the process that demonstrates when a program does what it tries to do, as well as it enables to discover defects before using such a program. Software test is very important for the successful development of a software development project; however, little attention gas given they to this process, therefore it is implemented as an ad-hoc process without methodical support. In this paper it is considered three types of tests: unit tests, integration tests and functional tests. Our contribution is an analysis of features related to free test software tools with the purpose of chossing a tool well suited to the needs of a SME company (in our case MBN) which will be the basis for the implementation of a testing area.

Key words: (Software testing, Tools for testing and Quality models)

I. INTRODUCCIÓN

Las pruebas de software verifican que el sistema desarrollado cumpla con las especificaciones planteadas por el cliente o analista en la fase de especificación de requerimientos, eliminando los errores que se hayan cometido en cualquier etapa del desarrollo del software. Sin embargo, no solo se limitan a medir la funcionalidad del producto final, sino que se enfocan en medir la calidad de diversos componentes del producto y en diferentes etapas.

Cyndi González Águila
Maestría en Sistemas Computacionales
Tecnológico Nacional de México
Instituto Tecnológico de Apizaco
gonzalez.cindy@mbn.com.mx

Alberto Portilla Flores
Centro de Innovación y Desarrollo de Talento
MBN-Miracle Business Network S.A.
portilla.alberto@mbn-corp.com.mx

María Guadalupe Medina Barrera
Tecnológico Nacional de México
Instituto Tecnológico de Apizaco
medinabm0184@itapizaco.edu.mx

Actualmente es necesario empezar a realizar pruebas desde etapas tempranas, ya que entre más tarde comiencen a realizarse las pruebas hay más posibilidad de encontrar errores. Es importante acotar que, las pruebas de software son una parte importante del desarrollo del software, pero también muy costosa, ya que pueden llegar a representar entre el 30 y 50% del costo total del desarrollo del software (Kuhn, Wallace, & Gallo Jr, 2004). Sin embargo, aprender a hacer pruebas es difícil debido a que, primero se conoce muy poco acerca de los conceptos de pruebas, y segundo, hacer pruebas es un proceso complejo y aún más abstracto que la programación misma (Velázquez, s.f.).

Nuestro trabajo está asociado a la empresa Miracle Business Network (MBN), una PYME de TI localizada en Tlaxcala con clientes en todo el País (MBN, 2016). MBN tiene diversas unidades estratégicas de negocios, de acuerdo a la norma en la cual está certificada Moprosoft (MoProsSoft, 2005), entre las cuáles se encuentra DMS - "Fábrica de Software" que desarrolla proyectos de Software a la medida para diversos clientes. MBN también esta evaluada de manera satisfactoria por el Software Engineering Institute (SEI, 2011) (MBN, 2016) en el

Bits & Software Magazine (2017) 1(1): 1-10.
ISSN: En trámite

Nivel de Madurez 2 del modelo de CMMI-Dev v.1.3. (Capability Maturity Model Integration for Development). Bajo CMMI Dev 2 los proyectos de desarrollo de software se realizan conforme a una gestión disciplinada a través de políticas organizacionales, y en base a procesos que planifican, monitorean y ejecutan el desarrollo de proyectos de software. Actualmente MBN enfrenta una gran demanda de productos de software por parte de sus clientes, por lo que es necesario contar con un proceso efectivo de pruebas para reducir los riesgos en el desarrollo de proyectos, que genere un vínculo de identificación plena entre el consumidor y la empresa, y en consecuencia una satisfacción y aceptación total por parte del cliente. Por lo tanto, MBN plantea como prioridad la institucionalización del proceso de área de pruebas de software de manera metódica para: i) alinearse a Moprosoft y CMMI Dev 2, ii) ejecutar de manera eficiente los proyectos de desarrollo manteniendo los márgenes de operación planeados en la estimación de cada proyecto, y iii) lograr la satisfacción de los clientes en los proyectos desarrollados. Para lograr estos objetivos MBN considera la definición de un proceso detallado para el área de pruebas de software y el uso de herramientas de software que ayuden al proceso de ejecución de dicho proceso, en este artículo presentamos el estudio de dichas herramientas. El método utilizado fue el siguiente: primero se realizó una investigación documental acerca del tema de pruebas de software a fin de aclarar aspectos y generar un reporte técnico para MBN que ayude a diseminar el conocimiento entre los integrantes de DMS-MBN; segundo, se investigó en el área cuales son las herramientas de software más usadas, se definieron los elementos de comparación entre ellos y se realizaron pruebas de uso de cada herramienta a fin de valorarlas; finalmente se discutió internamente los resultados de la valoración de herramientas y se definió cuáles cubren de mejor manera las necesidades de DMS.

El resto del artículo está organizado como sigue, en la Sección II presentamos los conceptos asociados a las pruebas de software, en la Sección III introducimos características de herramientas de software libre para la ejecución de pruebas de software, en la Sección IV se hace mención de los trabajos existentes relacionados a este articulo, y por último en la Sección 5 se presentan conclusiones y trabajos futuros.

II. PRUEBAS DE SOFTWARE

El proceso de prueba tiene dos objetivos (Sommerville, 2011):

a) Demostrar al desarrollador y al cliente que el software cumple con los requerimientos.
b) Encontrar situaciones donde el comportamiento del software sea incorrecto, indeseable o no esté de acuerdo con su especificación.

En la literatura de ingeniería de software se identifican varios tipos de pruebas (Comunicacion, 2008):

- Pruebas unitarias y de integración:
  - Unitaria: permite verificar la funcionalidad y estructura de cada componente individualmente del sistema una vez que ha sido codificado.
  - Integración: permite verificar el correcto ensamblaje entre los distintos módulos que componen el sistema desarrollado.
- Pruebas de sistema: buscan diferencias entre la solución desarrollada y los requerimientos, con el fin de identificar errores que se puedan generar entre la especificación funcional y el diseño del sistema. Las pruebas de sistema incluyen:
  - Carga: valida aquellos volúmenes de datos máximos especificados en los requerimientos no funcionales.
  - Volumen: somete el software a grandes cantidades de datos para determinar si se alcanzan límites que causen la falla del software.
  - Estrés: valida aquellos volúmenes de datos máximos que resiste el sistema antes de comenzar con errores.
  - Robustez: valida si el sistema se mantiene estable y consistente después de circunstancias adversas.
  - Concurrencia: valida la capacidad del sistema de atender múltiples solicitudes de parte de los usuarios que acceden a un mismo recurso.
  - Interfaz de usuario: permite verificar que la navegación a través de los elementos que se están probando, reflejen las funciones del negocio y los requerimientos funcionales.
  - Recuperación a fallas: aseguran que el que el software pueda recuperarse a fallas de hardware, software o mal funcionamiento de

Bits & Software Magazine (2017) 1(1): 1-10.
ISSN: En trámite

la red sin pérdida de datos o de integridad de los datos.
  - Seguridad: verifica el cumplimiento de las políticas de seguridad acordadas para el sistema.
  - Integridad de las bases de datos: aseguran que los métodos y procesos de acceso a la base de datos funcionan correctamente y sin corromper datos.
  - Interoperabilidad: permite verificar todos los artefactos de la solución desarrollada, su arquitectura base, los protocolos de la solución, las interfaces y los módulos del sistema, funcionando en forma conjunta.
  - Desempeño: aspecto fundamental en una aplicación, ya que, si ésta no responde en el debido tiempo, se pueden perder clientes, o dañar la imagen ante los usuarios.
  - Configuración: establece y mantiene la integridad de los productos de software a través del ciclo de vida del proceso del mismo.
- Pruebas funcionales: son un proceso para procurar encontrar discrepancias entre el programa y la especificación funcional, las pruebas funcionales pueden ser:
  - Caja Negra: permiten obtener conjuntos de condiciones de entrada que ejecutan todos los requisitos funcionales de un programa.
  - Ciclo de Negocio: garantizan que el proceso de negocio esta adecuadamente soportado por el software desarrollado y que éste dispone de la funcionalidad adecuada para ejecutar todas las tareas incorporadas en el proceso de negocio.
  - Usabilidad: permite encontrar problemas de factores humanos, o usabilidad.
  - Instalación: verifican la instalación y desinstalación de la aplicación en diferentes entornos de hardware y software.
- Pruebas de regresión: se valida que el sistema mantenga su correcta funcionalidad debido a la incorporación de un ajuste, corrección o nuevo requerimiento. Es una prueba funcional y técnica que valida que el sistema siga funcionando perfectamente después de que las correcciones sean aplicadas.
- Pruebas de aceptación: es la prueba final basada en las especificaciones del usuario o basada en el uso del programa por el usuario final luego de un periodo de tiempo.

La Tabla 1 muestra los tipos de pruebas con respecto a las siguientes características:

- Tipo de prueba:
  - Funcionales: comprueba que los sistemas desarrollados cumplan con las funciones específicas para los cuales han sido creados.
  - No funcionales: permiten conocer los riesgos relacionados con el mal desempeño de las aplicaciones en los entornos de producción.
- Técnicas de prueba:
  - Caja blanca: se realizan sobre la estructura interna(código) del sistema.
  - Caja negra: se realizan sobre la interfaz del sistema sin importar la estructura interna.
- Encargado de ejecutar pruebas: función de una persona en el proyecto.
- Objetivos de las pruebas:
  - Encontrar defectos: detectar la presencia de errores en desarrollo en el sistema.
  - Proporcionar mayor nivel de calidad: conformidad de las especificaciones en los requisitos del sistema.

III. HERRAMIENTAS PARA HACER PRUEBAS DE SOFTWARE

Las herramientas que analizamos son las siguientes:

- Junit: es un framework de código abierto desarrollado especialmente para crear, ejecutar y hacer reportes de estado de conjuntos de prueba unitaria automatizados hechos en lenguaje Java (De Seta, s.f.).
- Mockito: es una librería Java para la creación de Mock Object muy usados para el testeo unitario en Test Driven Development, basado en EasyMock. EasyMock y Mockito puede hacer exactamente las mismas cosas, pero Mockito tiene un API más natural y práctico de usar (De Seta, s.f.).
- Cactus: es un simple framework de pruebas para prueba unitaria de código Java (Servlets, EJB, TagLib, etc), intenta simplificar la escritura de pruebas del lado del servidor. Usa JUnit y lo extiende. Las pruebas son ejecutadas dentro del container (De Seta, s.f.).

Bits & Software Magazine (2017) 1(1): 1-10.
ISSN: En trámite

- EasyMock: proporciona Mock Object para interfaces durante la prueba unitaria mediante la generación de estas sobre la marcha utilizando mecanismo de proxy de Java. Debido al estilo único de registro de expectativas en EasyMock, la mayoría de refactor's no afectará a los objetos mock. Así EasyMock es perfecto para Test Driven Development. Es software de código abierto disponible bajo los términos de la licencia MIT (De Seta, s.f.).
- Selenium: compuesto por dos herramientas: Selenium IDE y SeleniumWebDriver. La primera, un plugin de Firefox que te genera un entorno de desarrollo y que permite crear casos de prueba para aplicaciones web. La segunda, Selenium WebDriver, ejecuta las pruebas (Garzas, 2012).
- Soapui: es una aplicación muy versátil que nos permite probar, simular y generar código de servicios web de forma ágil, partiendo del contrato de los mismos en formato WSDL y con vínculo SOAP sobre HTTP. soapUI tiene dos distribuciones: soapUI freeware (GNU LGPL y opensource java) y soapUIPro (comercial), en versión de escritorio, online y plugin para varios IDE. (Puebla, s.f.).
- JMeter: aplicación de escritorio en Java y dentro del proyecto Jakarta. Esta herramienta permite realizar pruebas funcionales (y de rendimiento) para aplicaciones web (Garzas, 2012).
- Testlink: permite crear y gestionar casos de prueba, organizarlos en planes de pruebas, realizar un seguimiento de los resultados, establecer trazabilidad con los requisitos, generar informes etc. Se integra con otros sistemas de seguimiento de "bugs" y "ticketing" como Bugzilla, Mantis, etc. Licencia: GPL (Garzas, 2012).

Tabla 1. Tipos de pruebas de software

Nombre de la prueba	Tipo de prueba		Técnicas de prueba		Encargado de ejecutar pruebas			Objetivos de pruebas	
	Funcionales	No Funcionales	Caja blanca	Caja negra	Desarrollador	Tester	Cliente	Encontrar defectos	Proporcionar mayor nivel de calidad
*Unitaria	X		X		X			X	
*Integración	X		X		X			X	
*Sistema	X			X		X		X	
*Carga		X		X		X		X	
*Volumen		X		X		X		X	
*Estrés		X		X		X		X	
*Robustez		X		X		X		X	
*Concurrencia		X		X		X		X	
*Interfaz de usuario		X		X		X		X	
*Recuperación a fallos		X		X		X		X	
*Seguridad		X		X		X		X	
*Integridad de base de datos		X		X		X		X	
*Interoperabilidad		X		X		X		X	
*Desempeño		X		X		X		X	
*Configuración		X		X		X		X	
		X		X		X		X	
*Funcionales	X			X		X		X	
*Ciclo de negocio	X			X		X		X	
*Usabilidad	X			X		X		X	
*Instalación	X			X		X		X	
*Pruebas de regresión	X			X		X		X	
*Pruebas de aceptación	X			X			X		X

El análisis se realizó con respecto a las siguientes características:

- Tipo de prueba: se refiere al tipo de prueba que se puede realizar con la herramienta (ver Sección II).
- Licencia: se refiere al tipo de términos y cláusulas que el usuario debe cumplir para usar el software o herramienta [11]:
  - Apache: el usuario puede modificarlo, y distribuir versiones modificadas de ese software, pero debe conservar el copyright y el disclaimer, al crear nuevas piezas de software, los desarrolladores deben incluir dos archivos en el directorio principal de los paquetes de software redistribuidos: una

Bits & Software Magazine (2017) 1(1): 1-10.
ISSN: En trámite

copia de la licencia y un documento de texto que incluya los avisos obligatorios del software presente en la distribución.
  - GPL: el desarrollador conserva los derechos de autor, permite su libre distribución, modificación y uso, en el caso de que el software se modifique, el nuevo software que se desarrolle como resultado queda con la misma licencia, y debe estar disponible y accesible, para copias ilimitadas a cualquier persona que lo solicite. De cara al usuario final, el software es totalmente gratuito, pudiendo pagar únicamente por gastos de copiado y distribución.
- Lenguaje: estructura, contenido y uso que posee un programa de computadora o software.
- Requisitos: condición necesaria para el uso del software.
- Sistema operativo: conjunto de órdenes y programas para procesos de una computadora.
- Interfaz gráfica: interfaz de usuario representa información y acciones disponibles del sistema.
- Reportes: muestran los resultados de la ejecución de pruebas en distintos formatos.
- Herramientas complementarias: software que complementa y amplia con nuevas funcionalidades.
- Desventajas: limitación de funciones en el software.
- Funcionamiento: serie de instrucciones y datos para aprovechar los recursos con los que cuenta un software.

Tabla 2. Herramientas para hacer pruebas de software

La Tabla 2 muestra los nombres de las herramientas analizadas. A partir de la Tabla 1 y 2 se puede deducir, por ejemplo, que Cactus es un software de código abierto, su lenguaje es Java, y funciona en sistemas operativos Windows y Linux, pero no posee una interfaz gráfica, el nivel de prueba que soporta es unitaria, las cuales son de tipo funcional y deben ser ejecutadas por el desarrollador para encontrar los defectos que se hayan generado en el desarrollo del sistema. De manera similar Soapui es un software para pruebas funcionales y de carga el tipo de técnica utilizada para la ejecución de sus pruebas es de caja negra y las pruebas deben ser ejecutadas por el desarrollador para encontrar los defectos que se hayan generado en el desarrollo del sistema, software de código abierto, su lenguaje es Java, y es multiplataforma, posee una interfaz gráfica, se puede complementar con más herramientas por ejemplo JUnit, genera reportes con formato HTML y cuenta con la función de probar, simular y generar código de servicios web.

En nuestro caso la herramienta que mejor se adapta a MBN es JUnit por las siguientes razones: i) cubre las pruebas unitarias y de integración además permite realizar pruebas de regresión para asegurarnos que un cambio o la incorporación de una nueva funcionalidad no haya causado otros problemas, las cuales se definieron como las más importantes para la fábrica de software de MBN, ii) la licencia de la herramienta es abierta, lo cual permite utilizarla sin incrementar el costo de la ejecución de proyectos en MBN, y iii) se adapta de manera adecuada a proyectos desarrollados en Java, los cuales son el tipo de proyectos que se desarrollan principalmente en MBN. Otra de las herramientas que se adapta a MBN en cuanto a pruebas funcionales es Selenium ya que i) es una herramienta que permite el uso de diferentes API´s en diferentes lenguajes por ejemplo Java, ii) permite realizar una grabación de las pruebas funcionales durante la generación de pruebas de regresión, con este servicio se consigue obtener una batería de pruebas automatizadas que podrán ser utilizadas cuando sea necesario repetir las pruebas, y iii) su licencia es de código abierto.

IV. TRABAJOS RELACIONADOS

Algunos trabajos han abordado problemáticas similares a las que planteamos en este artículo. En (Sakti, Pesant, & Guéhéneuc, 2016) JTExpert at the Fourth Unit Testing Tool Competition JTExpert se analizan las características de la herramienta para pruebas de software llamada JTExpert y se muestran los resultados de la comparación de herramientas que participan en el concurso de pruebas unitarias celebrado como parte del Taller Internacional sobre pruebas de software basadas en búsquedas en Austin, Texas, en donde JTExpert ocupó el tercer lugar, es una herramienta que genera automáticamente una suite de pruebas completa. Toma como entradas un código fuente de Java y sus dependencias y produce automáticamente una suite de casos de prueba en formato JUnit. En (Bauersfeld, Vos, Lakhotia, Poulding, & Condori, 2013) Unit Testing Tool Competition se realiza una comparativa de herramientas para pruebas de software, describe la metodología y los resultados de

Bits & Software Magazine (2017) 1(1): 1-10.
ISSN: En trámite

Tabla 2. Herramientas para hacer pruebas de software.

Nombre de la herramienta	JUnit	Mockito	Cactus	EasyMock	Selenium	Soapui	JMeter	Testlink
Tipo de prueba	Unitarias Regresión Integración	Unitarias	Unitarias	Unitarias	Funcionales	Funcionales Carga	Funcionales Rendimiento	Funcionales
Licencia	Open Source	Open Source	Open Source	Open Source	Apache 2.0	GNU LGPL	Open Source	Open Source
Lenguaje	Java	Java	Java	Java	Java	Java	Java	PHP MySQL
Requisitos	NetBeans Eclipse	Maven	Eclipse Maven	Proxy de Java	Mozilla Firefox	Internet Explorer	Mozilla Firefox	Mozilla Firefox
Sistema Operativo	Multiplataforma	Multiplataforma	Windows Linux	Multiplataforma	Windows Linux OSX	Multiplataforma	Multiplataforma	Windows Linux
Interfaz gráfica	No	No	No	No	Si	Si	Si	Si
Reportes	Feedback Formato XML	Feedback	Feedback	Feedback Formato HTML	Formato XML	Formato HTML	Formatos (CVS, jtl)	Formato XML, CSV
Herramientas complementarias	Mockrunner, EasyMock, JMock, DbUnit	JUnit	JUnit	JUnit o TestNG	Selenium WebDriver, Selenium IDE	Maven, Hudson, Bamboo, JUnit, ANT	Firebug, HTTP Fox, A SoapUI, A Badboy	Bug Tracking systems: Bugzilla, Mantis, Jira, TrackPlus, Eventum, Trac, Seapine, Redmine
Desventajas	Carencia de una interfaz completamente gráfica	Verifica sólo las interacciones que se le pidió que verificar	Extensión de JUnit	Solo se integra con java	Las pruebas tienen que ser locales; no se pueden lanzar en remoto	El realizar programas clientes de prueba se vuelve tedioso.	Se necesitan herramientas adicionales durante el desarrollo de su test plan	Si no se cuenta con los datos suficientes para el estudio del sistema la ejecución es complicada
Funcionamiento	Runners que pueden ser en modo texto, gráfico (AWT o Swing) o como tarea en Ant	Crea objetos mock, estos objetos simulan parte del comportamiento de una clase	Pruebas del lado del servidor	Utiliza mecanismos de Proxy de Java	Los scripts grabados, pueden ser exportados en distintos lenguajes	Prueba, simula y genera código de servicios web.	Analiza el rendimiento general de la aplicación	Crea y gestiona casos de prueba

la 4ta edición de herramientas de prueba de unidad de Java, estas herramientas incluyen una seria de mejoras de infraestructura, métricas de efectividad de pruebas y una evaluación de las herramientas de generación de pruebas. Para ello, se realiza una competencia definiendo primero un benchmark de clases Java, que son seleccionadas a partir de proyectos de código abierto, y se genera una fórmula de puntuación que tiene en cuenta la efectividad (es decir, la cobertura de código y la capacidad de localización de fallos) de las pruebas generadas. En (Panichella, Meshesha Kifetew, & Tonella, 2015) Results for EvoSuite -- MOSA at the Third Unit Testing Tool Competition presenta los resultados obtenidos por Evo Suite-MOSA en la tercera prueba de unidad de pruebas en la SBST'15. Entre los seis participantes, Evo Suite-MOSA se ubicó tercero,es una herramienta de generación de datos de prueba de unidad que emplea un nuevo algoritmo de optimización de objetivos múltiples. Se implementó extendiendo la herramienta de generación de datos de prueba de Evo Suite. La ejecución de las pruebas impacta en los riesgos del producto de software, es por eso que en este trabajo se realiza un análisis de algunas herramientas existentes sobre pruebas de software, ya en los trabajos que se mencionaron anteriormente no se realiza una comparativa de las herramientas para las pruebas de software que puedan apoyar a la optimización de pruebas.

V. CONCLUSIONES

Los sistemas de software hoy en día son parte importante en nuestras actividades diarias, pero es necesario entender que los sistemas son desarrollados por seres humanos, por lo cual se generan defectos en cualquier etapa de desarrollo del software. Es por esto que es necesario realizar pruebas de software, que ayuden a detectar errores para darle solución a los mismos para mejorar la calidad del producto. En este trabajo presentamos los conceptos básicos para entender el entorno de pruebas en un proyecto de desarrollo de software, así como un análisis de herramientas para el desarrollo de pruebas que puede ser útil a la hora de seleccionar una herramienta de apoyo que pueda ser la base para realizar pruebas exitosas de manera semi-automática. Como trabajo futuro se está definiendo la propuesta de un framework alineado a los estándares de calidad CMMI y MoProSoft para mejorar la calidad de los productos de software realizados en MBN.

Bits & Software Magazine (2017) 1(1): 1-10.
ISSN: En trámite

VI. REFERENCIAS

1. (s.f.). Recuperado el 21 de Julio de 2017, de https://bbvaopen4u.com/es/actualidad/las-5-licencias-de-software-libre-mas-importantes-que-todo-desarrollador-debe-conocer
2. Panichella, A., Meshesha Kifetew, F., & Tonella, P. (2015). Results for EvoSuite-MOSA at the Third Unit Testing Tool Competition. IEEE/ACM 8th International Workshop on Search-Based Software Testing.
3. Sakti, A., Pesant, G., & Guéhéneuc, Y.-G. (2016). JTExpert at the Fourth Unit Testing Tool Competition. 2016 9th International Workshop on Search-Based Software Testing.
4. Bauersfeld, S., Vos, T., Lakhotia, K., Poulding, S., & Condori, N. (2013). Unit Testing Tool Competition. IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops.
5. Comunicacion, I. N. (2008). Guia de mejores practicas de calidad de producto. 188.
6. De Seta, L. (s.f.). Dos Ideas. Recuperado el 21 de Julio de 2017, de https://dosideas.com/wiki/JUnit
7. Garzas, J. (28 de Marzo de 2012). javiergarzas.com. Recuperado el 21 de Julio de 2017, de http://www.javiergarzas.com/2012/03/herramientas-para-pruebas-software.html
8. Kuhn, D. R., Wallace, D. R., & Gallo Jr, A. M. (2004). Software fault interactions and implications for software testing. IEEE.
9. MBN. (2016). EncuadreCMMiDev2-MBN.
10. MoProsSoft. (2005). Norma Mexicana NMX-I-059/01-NYE-2005, NYCE.
11. Puebla, I. G. (s.f.). AdictosAlTrabajo.com. Recuperado el 21 de Julio de 2017, de https://www.adictosaltrabajo.com/tutoriales/introduccion-soap-ui/
12. SEI. (January de 2011). Software Engineering Institute,Carnegie Mellon, Introduction to CMMI for Development Version 1.3.
13. Sommerville, I. (2011). Ingenieria de Software. Mexico: PEARSON Educación,.
14. Velázquez, E. M. (s.f.). El camino a un mejor programador. Recuperado el 26 de Junio de 2017, de https://emanchado.github.io/camino-mejor-programador/html/ch06.html

VII. BIOGRAFÍA

Lic. Cyndi González Águila alumna del Instituto Tecnológico de Apizaco en la Maestría de Sistemas Computacionales.

Dr. Alberto Portilla Flores es responsable del Centro de Innovación y Desarrollo de Talento (CIDT) en MBN y profesor en la Facultad de Ciencias Básicas, Ingeniería y Tecnología de la Universidad Autónoma de Tlaxcala, Doctor en Informática por Universidad de Grenoble, Francia, Doctor en Ciencias de la Computación Cum Laude por la Fundación Universidad de las Américas-Puebla, México y PosDoc obtenido en el French Mexican Laboratory of Informatics and Automatic Control (LAFMIA UMI-3175), Evaluador Acreditado de los Comités de Acreditación de Evaluadores del Sistema Nacional de Evaluación Científica y Tecnológica (SINECYT-CONACyT. Ha publicado artículos arbitrados a nivel nacional e internacional en el área de cómputo en la nube, sistemas transaccionales, cómputo orientado a servicios e ingeniería de software.

M. en C. María Guadalupe Medina Barrera es docente de tiempo completo de la Maestría en Sistemas Computacionales del Instituto Tecnológico de Apizaco. Área de adscripción: División de Estudios de Posgrado e Investigación. Maestra en Sistemas Computacionales por el Centro Nacional de Investigación y Desarrollo Tecnológico (cenidet). Profesora con Reconocimiento al Perfil Deseable PRODEP y miembro del cuerpo académico PRODEP "Sistemas de Información". Evaluadora del Consejo Nacional de Acreditación en Informática y Computación A.C. (conaic).