Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ciencias de la Computación e Informática

Informe de Proyecto de Graduación para optar por el grado académico de
Licenciatura en Computación e Informática

Prototipo de plataforma basada en herramientas de código abierto para apoyar las tareas de verificación y validación de proyectos de software en el Centro de Informática de la Universidad de Costa Rica.

por

Guillermo Esteban Murillo Goussen – A43703

Ciudad Universitaria Rodrigo Facio,
San Pedro, San José, Costa Rica
octubre de 2016

Este proyecto de graduación ha sido aceptado por el Tribunal Examinador como requisito parcial para optar al grado académico de Licenciatura en Computación e Informática.

Miembros del Tribunal Examinador:

_____

M.Sc. Gabriela Salazar Bermúdez
Directora del Proyecto

_____

Dr. Vladimir Lara Villagrán
Asesor

_____

Mag. Luis Quesada Quirós
Asesor

_____

M.Sc. Alan Calderón Castro
Lector

_____

Dr. Carlos Vargas Castillo
Director de la Escuela de
Ciencias de la Computación e Informática

DEDICATORIA

¡Viva Cristo Rey!

¡Viva María Santísima!

AGRADECIMIENTO

Esta investigación fue realizada gracias al aporte y contribución de muchas personas a las que quisiera extender mi más sincero reconocimiento.

A mi directora, Gabriela Salazar Bermúdez, acompañamiento constante, por su buena disposición en todo momento y a lo largo de este largo proceso. Quisiera agradecerle todo su apoyo y sobre todo, la confianza que tuvo en mí para desarrollar este proyecto.

A Vladimir Lara Villagrán, porque sin sus acciones y recomendaciones, siempre tan pertinentes e inteligentes, no hubiese podido conseguir los frutos de esta investigación. Gracias por todas las manifestaciones de apoyo que tuvo para conmigo.

Al profesor Allan Berrocal Rojas por toda su colaboración, por sus consejos en torno a la actitud que debía tomar para lograr los objetivos en una institución como la Universidad de Costa Rica y por su actitud siempre servicial, para colaborar sin importar las distancias y lo complicado de los trámites.

A Luis Quesada Quirós por estar dispuesto a colaborar con el proyecto en uno de los momentos más complicados que atravesé en su realización. Le estaré siempre muy agradecido.

Al Centro de Informática, en la persona de Alonso Alvarado por toda su ayuda y apoyo durante la realización de la investigación. Le agradezco sobretodo, haber sacado tiempo para reunirse tantas veces conmigo, hacer el esfuerzo por entender mis propuestas y

A mi esposa, por ser mi más grande apoyo, por estar conmigo días y noches, en mis crisis y triunfos, por celebrar detalles tan simples como la correcta instalación de una herramienta. Te amo con todo mi corazón.

A mis padres y suegros por sus palabras de aliento, por brindarme fuerzas siempre que los busqué para compartir el estado del proyecto.

Y a mis amigos, por su preocupación y el cariño que me otorgaron durante el proyecto.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS ..... **X**

ÍNDICE DE TABLAS ..... **XII**

ÍNDICE DE EXTRACTOS DE CÓDIGO ..... **XIII**

ÍNDICE DE ABREVIATURAS ..... **XIV**

RESUMEN ..... **XV**

CAPÍTULO I: INTRODUCCIÓN **1**

1.1 JUSTIFICACIÓN 1

1.2 OBJETIVOS 5

1.2.1 Objetivo General *5*

1.2.2 Objetivos Específicos *5*

1.3 DELIMITACIÓN DEL PROBLEMA 6

CAPÍTULO II: MARCO TEÓRICO **8**

2.1 ASEGURAMIENTO DE LA CALIDAD DEL SOFTWARE 8

2.2 VERIFICACIÓN Y VALIDACIÓN DE SOFTWARE 10

2.3 PRUEBAS DE SOFTWARE 12

2.3.1 Técnicas de Pruebas de Software *13*

2.3.2 Niveles de Pruebas de Software *14*

2.4 REQUERIMIENTOS FUNCIONALES Y NO FUNCIONALES 16

2.5 HERRAMIENTAS PARA LA ADMINISTRACIÓN DE PROYECTOS 18

2.6 HERRAMIENTAS PARA EL CONTROL DE VERSIONES 20

2.7 BUSINESS PROCESS MANAGEMENT 22

2.8 ARQUITECTURA DE SOFTWARE 27

2.9 ESTÁNDARES INTERNACIONALES DE CALIDAD 29

2.9.1 Estándares ISO *30*

2.9.2 Estándares IEEE *32*

CAPÍTULO III: METODOLOGÍA **34**

3.1. ANÁLISIS DE LA SITUACIÓN ACTUAL EN EL CENTRO DE INFORMÁTICA 34
3.2 IDENTIFICACIÓN DE LOS ESTÁNDARES INTERNACIONALES APLICABLES A LOS CINCO PROBLEMAS MÁS IMPORTANTES DEL CI EN MATERIA DE V&V DE SOFTWARE 35
3.3 ADAPTACIÓN DE LOS ESTÁNDARES SELECCIONADOS CONSIDERANDO LAS CARACTERÍSTICAS PARTICULARES DEL CI 36
3.4 IDENTIFICACIÓN DE LAS HERRAMIENTAS DE SOFTWARE 36
3.5 DISEÑO Y CONSTRUCCIÓN DEL PROTOTIPO DE PLATAFORMA 37
3.6 EVALUACIÓN DEL PROTOTIPO DE PLATAFORMA 37

CAPÍTULO IV: RESULTADOS 39

4.1 OBJETIVO ESPECÍFICO I: IDENTIFICAR Y SELECCIONAR LOS ESTÁNDARES APLICABLES A LOS CINCO PROBLEMAS MÁS IMPORTANTES DEL CI, EN MATERIA DE V&V DE SOFTWARE 39
4.1.1 Situación Actual del Centro de Informática 39
4.1.2 Selección de los Cinco Principales Problemas 43
4.1.2.1 Problema #1: No hay automatización en las pruebas de software 44
4.1.2.2 Problema #2: El proceso de pruebas no se ejecuta sistemáticamente igual 45
4.1.2.3 Problema #3: No se generan reportes estadísticos sobre el proceso de pruebas 45
4.1.2.4 Problema #4: El registro de los criterios de aceptación es bastante irregular 46
4.1.2.5 Problema #5: El seguimiento y control de los proyectos es dificultoso 47
4.1.3 Selección de los Estándares Internacionales 47
4.2 OBJETIVO ESPECÍFICO II: CON BASE EN LOS ESTÁNDARES DE CALIDAD SELECCIONADOS, PROPONER SOLUCIONES A LOS PROBLEMAS ESCOGIDOS 49
4.2.1 Solución Propuesta al Problema #1 49
4.2.2 Solución Propuesta al Problema #2 50
4.2.2.1 Diseño del Nuevo Proceso de Pruebas 50
4.2.2.2 Documentación del Proceso de Pruebas 57
4.2.3 Solución Propuesta al Problema #3 64
4.2.4 Solución Propuesta al Problema #4 65
4.2.5 Solución Propuesta al Problema #5 66
4.3 OBJETIVO ESPECÍFICO: IDENTIFICAR LAS HERRAMIENTAS DE SOFTWARE APROPIADAS PARA CONSTRUIR UN PROTOTIPO DE PLATAFORMA QUE LE PERMITA AL CI SOLUCIONAR LOS PROBLEMAS ESCOGIDOS 68
4.3.1 Processmaker 68
4.3.2 Redmine 69

4.3.3 Selenium *70*
4.3.4 Git *71*
4.4 OBJETIVO ESPECÍFICO: DISEÑAR UN PROTOTIPO DE PLATAFORMA COMPUTACIONAL QUE IMPLEMENTE LAS SOLUCIONES ENCONTRADAS PARA LOS PROBLEMAS IDENTIFICADOS EN EL CI 74
4.4.1 Configuración de Processmaker *74*
4.4.1.1 Roles y Usuarios 75
4.4.1.2 Modelado de Procesos 76
4.4.1.3 Formularios dinámicos 82
4.4.1.4 Exportación de documentos 84
4.4.2 Configuración de Redmine *85*
4.4.2.1 Roles y Usuarios 85
4.4.2.2 Peticiones y Campos Personalizados 85
4.4.3 Integración Processmaker-Redmine *87*
4.4.4 Integración Redmine-Git *90*
4.5 OBJETIVO ESPECÍFICO: EVALUAR LA APLICABILIDAD DEL PROTOTIPO DE PLATAFORMA IMPLEMENTADO, COMO APOYO AL PERSONAL DEL CI EN LAS TAREAS DE V&V DE LOS PROYECTOS DE SOFTWARE 92
CAPÍTULO V: CONCLUSIONES 95
5.1 CONCLUSIONES 95
5.2 RECOMENDACIONES 98
5.3 TRABAJO FUTURO 99
REFERENCIAS 101
ANEXOS 105
ANEXO A – CUESTIONARIO ANÁLISIS DE LA SITUACIÓN ACTUAL DEL CI 105
ANEXO B – DUDAS SOBRE EL CUESTIONARIO DE DIAGNÓSTICO 106
ANEXO C – RESUMEN SOBRE LAS PRUEBAS DE PORTABILIDAD 107
ANEXO D – REGISTRO DEL PROYECTO DE PRUEBAS 110
ANEXO E – GUÍA PARA LA REVISIÓN DE REQUERIMIENTOS 111
ANEXO F – EVALUACIÓN DE CASO DE USO 115
ANEXO G – EVALUACIÓN DE HISTORIA DE USUARIO 116
ANEXO H – RESULTADO DE LAS PRUEBAS EXPLORATORIAS 118

ANEXO I – PRUEBAS DE ACEPTACIÓN 119
ANEXO J – PLAN DE PRUEBAS..... 120
ANEXO K – DISEÑO DE PRUEBAS 122
ANEXO L – REPORTE DE INCIDENTE 123
ANEXO M – REPORTE DE FINALIZACIÓN DE LAS PRUEBAS 124
ANEXO N – ESPECIFICACIÓN DE REQUERIMIENTOS DE USUARIO 126
ANEXO Ñ – REGISTRO DE HISTORIA DE USUARIO..... 128
ANEXO O – MANUAL DE USUARIO 129
ANEXO P – CONSECUCIÓN DE LOS OBJETIVOS DE INVESTIGACIÓN 153

ÍNDICE DE FIGURAS

FIGURA 1: ACTIVIDADES DE V&V A LO LARGO DEL CICLO DE VIDA DEL SOFTWARE 10

FIGURA 2: EJEMPLO DE PROCESO DE PRUEBAS DE SOFTWARE 13

FIGURA 3: DIAGRAMA DEL PROCESO PARA SOLICITAR PRODUCTOS A BODEGA, USANDO NOTACIÓN BPMN 23

FIGURA 4: VISTA FUNCIONAL DEL SISTEMA "TIENDA EN LÍNEA". BASADO EN [33] 29

FIGURA 5: FLUJO BÁSICO DE ACTIVIDADES PARA LA OBTENCIÓN DE REQUERIMIENTOS EN EL CI 40

FIGURA 6: DIAGRAMA DE ACTIVIDADES EN EL PROCEDIMIENTO DE VALIDACIÓN DE REQUERIMIENTOS EN EL CI 42

FIGURA 7: MODELO POR CAPAS BASADO EN EL DEFINIDO POR EL ESTÁNDAR ISO/IEC/IEEE 29119 53

FIGURA 8: MODELO DE PROCESO PARA LAS ACTIVIDADES DE V&V DE SOFTWARE EN EL CENTRO DE INFORMÁTICA 55

FIGURA 9: VISTA FUNCIONAL DEL PROTOTIPO DE PLATAFORMA PROPUESTO 72

FIGURA 10: VISTA DE INFORMACIÓN DEL PROTOTIPO DE PLATAFORMA PROPUESTO 73

FIGURA 11: GRUPOS DE USUARIO DEFINIDOS EN PROCESSMAKER 75

FIGURA 12: USUARIOS DE PROCESSMAKER Y SU ROL ASOCIADO 75

FIGURA 13: PROCESOS DISEÑADOS EN PROCESSMAKER 76

FIGURA 14: MACRO-PROCESO "PRUEBAS DE SOFTWARE U. CALIDAD" 77

FIGURA 15: SUB-PROCESO "VERIFICACIÓN DE REQUISITOS" 78

FIGURA 16: SUB-PROCESO "PRUEBAS EXPLORATORIAS" 78

FIGURA 17: SUB-PROCESO "PRUEBAS FUNCIONALES" 79

FIGURA 18: SUB-PROCESO PRUEBAS DINÁMICAS 79

FIGURA 19: SUB-PROCESO PRUEBAS DE ACEPTACIÓN 80

FIGURA 20: VARIABLES COMPARTIDAS ENTRE EL PROCESO GENERAL Y EL SUB-PROCESO VERIFICACIÓN DE REQUISITOS 81

FIGURA 21: PERMISOS DE PROCESO PARA EL GRUPO LÍDER 81

FIGURA 22: FORMULARIO Y DISPARADORES ASIGNADOS A LA TAREA FINALIZACIÓN 82

FIGURA 23: DOCUMENTO DE SALIDA UTILIZADO PARA EXPORTAR EL RESULTADO DE LAS PRUEBAS EXPLORATORIAS 84

FIGURA 24: PERFILES DEFINIDOS EN REDMINE 85

FIGURA 25: REPORTE DE INCIDENTE EN REDMINE 86

FIGURA 26: INCLUSIÓN DE LA BASE DE DATOS DE REDMINE EN PROCESSMAKER 87

FIGURA 27: ENLACE DE GRILLA DE INCIDENCIAS CON LA BASE DE DATOS DE REDMINE 88

FIGURA 28: DEFINICIÓN DE LAS CARACTERÍSTICAS DEL REPOSITORIO VERSIONADO 90

FIGURA 29: REPOSITORIO GIT, CON IDENTIFICADOR "LIC" INTEGRADO EN REDMINE 91

ÍNDICE DE TABLAS

TABLA 1: EJEMPLOS DE REQUERIMIENTOS FUNCIONALES Y NO FUNCIONALES 18

TABLA 2: POSIBLE EVALUACIÓN DEL PROYECTO DE GRADUACIÓN 38

TABLA 3: ESTÁNDARES DE CALIDAD PARA EL DESARROLLO Y MANTENIMIENTO DE SOFTWARE EXAMINADOS 48

TABLA 4: PROPUESTA DE PLANTILLAS PARA DOCUMENTAR EL PROCESO DE PRUEBAS FUNCIONALES EN EL CI 59

TABLA 5: ADAPTACIÓN DEL PLAN DE PRUEBAS CON RESPECTO AL ESTÁNDAR IEEE 29119 60

TABLA 6: ADAPTACIÓN DEL DISEÑO DE PRUEBAS CON RESPECTO AL ESTÁNDAR IEEE 29119 61

TABLA 7: ADAPTACIÓN DEL REPORTE DE INCIDENTE CON RESPECTO AL ESTÁNDAR IEEE 29119 63

TABLA 8: ADAPTACIÓN DEL REPORTE DE FINALIZACIÓN CON RESPECTO AL ESTÁNDAR IEEE 29119 63

TABLA 9: RESUMEN DE SOLUCIONES PROPUESTAS 67

ÍNDICE DE EXTRACTOS DE CÓDIGO

CÓDIGO 1: OBTENCIÓN DE DATOS DE USUARIO, LA FECHA Y LA HORA DEL SISTEMA: 83

CÓDIGO 2: UTILIZACIÓN DE MÉTODOS JAVASCRIPT PREDEFINIDOS 83

CÓDIGO 3: PROGRAMACIÓN DE SUB-RUTINA JAVASCRIPT PARA EL CÁLCULO DE LA COBERTURA DE PRUEBAS 84

CÓDIGO 4: DISPARADOR PARA EL LLENADO DE LA TABLA "NOCONFORMIDADES" 89

ÍNDICE DE ABREVIATURAS

La siguiente lista contiene las abreviaturas y acrónimos de uso más frecuente a lo largo del documento:

BPM:	*Business Process Management*
CI:	Centro de Informática
CU:	Caso de Uso
IEEE:	Instituto de Ingeniería Eléctrica y Electrónica; en inglés *Institute of Electrical and Electronics Engineers*
IEC:	Comisión Electrotécnica Internacional; en inglés *International Electrotechnical Commission*
ISO:	Organización Internacional de Normalización; en inglés International *Organization for Standardization*
SCM:	Sistema de Control de Código Fuente; en inglés *Source Code Management*
SWEBOK:	Cuerpo de Conocimiento en Ingeniería de Software; en inglés *Software Engineering Body of Knowledge*
UCMC:	Unidad de Calidad y Mejora Continua
UCR:	Universidad de Costa Rica
V&V:	Verificación y validación

RESUMEN

Cita bibliográfica

Murillo G., Guillermo E. (2016). *Prototipo de plataforma basada en herramientas de código abierto para apoyar las tareas de verificación y validación de proyectos de software en el Centro de Informática de la Universidad de Costa Rica*. Informe de Proyecto de Graduación para optar por el grado de Licenciatura en Ciencias de la Computación e Informática. Ciudad Universitaria Rodrigo Facio Brenes, Universidad de Costa Rica.

Directora del proyecto

M. Sc. Gabriela Salazar Bermúdez

Palabras clave

Verificación y validación de software, pruebas de software, herramientas de código abierto, IEEE 1012, IEEE 830, ISO/IEC/IEEE 29119, BPM, Processmaker, Redmine, Git, Selenium

Resumen

El presente trabajo de investigación fue realizado en la Unidad de Calidad y Mejora Continua del Centro de Informática de la Universidad de Costa Rica, con el objetivo de construir, un prototipo de plataforma computacional para el apoyo de sus de actividades de verificación y validación de software. Dicho prototipo, debía contar con la particularidad de estar integrado únicamente por herramientas de código abierto.

Para lograrlo, fue necesario realizar un análisis sobre la manera en que se desarrollan actualmente las actividades de verificación y validación de software en el Centro de Informática. De dicho análisis se extrajeron 5 problemas y se eligieron 3 estándares internacionales para hacerles frente. Los estándares elegidos fueron: IEEE-1012 Verificación y Validación de Software y Sistemas, IEEE-830 Práctica Recomendada para la Especificación de Requerimientos de Software, ISO/IEC/IEEE 29119 Pruebas de Software.

Con el apoyo de las normas internacionales, se propusieron soluciones a los problemas seleccionados y de forma paralela se seleccionaron las herramientas de código abierto que integrarían el prototipo de plataforma. Se eligieron 4 herramientas: Processmaker, Redimine, Git y Selenium. Éstas fueron configuradas e integradas de modo que juntas implementan las soluciones propuestas.

Finalmente se evaluó la aplicabilidad del prototipo desarrollado en el Centro de Informática y se determinó que con el uso de la plataforma, es posible mejorar la forma en que se ejecutan algunas tareas de verificación y validación, puntualmente: la verificación de los requerimientos de usuario, las pruebas exploratorias, las pruebas funcionales y las pruebas de aceptación.

Capítulo I: INTRODUCCIÓN

1.1 JUSTIFICACIÓN

La instancia encargada de liderar los procesos técnicos y estratégicos de las tecnologías de información y comunicación en la Universidad de Costa Rica es el Centro de Informática (CI) [1]. Esta oficina tiene como política brindar productos y servicios de calidad, a través de la innovación tecnológica, el compromiso del personal y su desarrollo constante, todo ello mediante procesos de mejora continua [2]. Esta política afecta transversalmente todos los objetivos estratégicos del CI, siendo de mayor relevancia para la presente investigación los siguientes [1]:

- Desarrollar productos y servicios que faciliten el acceso universal y la utilización masiva de Tecnologías de la Información.
- Gestionar oportunamente el aprovisionamiento de recursos para la implementación y mantenimiento de productos y servicios de Tecnologías de la Información.

De acuerdo a estos objetivos y por el apoyo solicitado a la Escuela de Ciencias de la Computación e Informática, para desarrollar el proyecto "Definición de una metodología de calidad de software para la Universidad de Costa Rica (UCR)", se infiere que el CI posee el interés de generar y adquirir productos de software de calidad. Por tanto, se realizó un análisis preliminar (con la información obtenida en entrevistas) sobre la forma en que se llevan a cabo las actividades de verificación y validación (V&V) de software en el CI y se identificaron las siguientes situaciones:

1. Algunas de las soluciones de software desarrolladas o adquiridas por el Centro de Informática presentan problemas de seguridad (por ejemplo E-matricula), y otras colapsan a diario producto del consumo exagerado de recursos (por ejemplo el Sistema de Gestión de Documentos, llamado SISDOC) [3].
2. Los proyectos de desarrollo sufren de extensiones excesivas de tiempo y dinero; y los entregables no satisfacen totalmente las necesidades del usuario.

3. Los desarrolladores se quejan del poco involucramiento del usuario, lo que se traduce en dificultades para traducir en requerimientos sus necesidades. Por su parte, los usuarios se quejan de que el producto recibido no cumple con sus expectativas. Todo deriva en la solicitud de cambios frecuentemente y el tiempo de espera para poder utilizar el sistema se alarga considerablemente.
4. No se automatiza ninguna de las actividades del proceso de pruebas, todas ellas son ejecutadas manualmente. El desarrollador puede llegar a dedicar entre un 30% y un 40% del tiempo total del proyecto a las pruebas de software. A su vez, el *encargado de calidad* puede invertir hasta 20 horas a la semana en tareas ligadas a las pruebas de software (lo equivalente a 1/2 Tiempo Completo en la universidad).
5. Se utilizan algunas plantillas para la documentación de: casos de uso, plan de pruebas, hoja de chequeo; sin embargo, no se utilizan en forma regular.
6. No se llevan ni métricas ni estadísticas de las pruebas.
7. El líder técnico del proyecto revisa aleatoriamente el código fuente para validar prácticas de programación.
8. No se sigue ningún estándar internacional para las actividades de V&V de software.
9. Para la evaluación de los requerimientos de usuario sólo se utilizan los casos de uso definidos. Así mismo, se está introduciendo la metodología ágil SCRUM, específicamente el desarrollo de iteraciones cortas con entregables "funcionales" y la generación de Historias de Usuario.

Con el Trabajo Final de Graduación propuesto, se desea apoyar al CI en las situaciones enumeradas, de la siguiente manera:

1. Como lo dice Al Endres en [4], la calidad no se da por accidente, sino que es producto de la planeación y a esta planeación pertenecen tareas clave como: desarrollar un proceso, optimizarlo y sistematizarlo. La presente investigación propone apoyar dos procesos en particular en el Centro de Informática: la verificación de requerimientos y la ejecución de pruebas funcionales de software. El objetivo del primer proceso es garantizar que los requisitos sean redactados de manera clara, concreta y concisa, para poder extraer de

ellos los criterios de aceptación y utilizar éstos en el diseño de los casos de prueba; y ojalá especificados en conformidad con un estándar. El objetivo del segundo es apoyar las pruebas funcionales para mejorar la manera en que se determina si el software desarrollado funciona de acuerdo a las especificaciones funcionales del cliente.

2. La plataforma propuesta implementará el seguimiento de estándares internacionales tanto sobre la ejecución de tareas de V&V de software como sobre su documentación. Las empresas requieren de más apoyo para la adopción de los estándares existentes. Los estándares ofrecen un conjunto de procesos valiosos para que las empresas puedan normalizar sus actividades y asegurar productos de software de mayor calidad [5].
3. En el CI todas las pruebas se ejecutan manualmente. Esto resulta un trabajo intensivo y propenso a errores. El prototipo de plataforma dispondrá de herramientas para la automatización de las pruebas, reduciendo no solo el tiempo dedicado a su ejecución sino también la probabilidad de cometer errores en el proceso.
4. El prototipo de plataforma busca solventar las necesidades básicas en cuanto a herramientas y automatización para ejecutar las actividades de V&V en el CI de forma eficaz y eficiente. Las compañías desarrolladoras de software se enfrentan a serios desafíos al probar sus productos, cada vez más grandes y complejos. Para hacerles frente, además de contratar gente capacitada, se les debe ayudar a conseguir las herramientas necesarias [6].
5. El prototipo de plataforma integrará herramientas de código abierto y de ser necesario, software desarrollado por el postulante, compartido en acceso abierto al concluir la investigación. Con esto los costos de implementación serán casi o totalmente nulos. Las empresas ven la importancia de la automatización para reducir los costos y tiempos. Además buscan que el costo de las licencias de los productos de software no sea una barrera para comenzar el proceso de automatización [7].
6. Con el prototipo de plataforma se busca simplificar, agilizar y normalizar las actividades de V&V de manera que no representen para el CI una sobrecarga de trabajo o una actividad engorrosa y burocrática. El tiempo de retraso de un proyecto es directamente

proporcional a la disminución en el tiempo dedicado a las actividades de V&V, específicamente a las pruebas de software [6].

7. La plataforma propuesta implementará la metodología *Business Process Management* (BPM) pues ésta engloba todas las actividades del ciclo de vida de un proceso de negocio (descubrimiento, diseño, simulación, ejecución, interacción, monitoreo, análisis, optimización). Además la plataforma estará conformada por herramientas distintas y la metodología BPM provee estándares de interoperabilidad para lograr la integración de ambientes heterogéneos, haciendo convivir las aplicaciones existentes con los nuevos desarrollos [7]. Con la inclusión de estas funcionalidades se espera construir una herramienta capaz de gestionar efectivamente los procesos de calidad del CI.
8. Según el Plan Estratégico Institucional 2013-2017, la Universidad de Costa Rica tiene como estrategia el desarrollar un Sistema de Gestión de Calidad de los procesos administrativos y define como meta para el mismo, elaborar e implementar herramientas para el mejoramiento continuo de los procesos, las cuales incluyan la desconcentración de los procesos que lo ameriten y la simplificación de trámites [8]. El CI no es ajeno a lo trazado por el Plan Estratégico Institucional, por ello la plataforma espera servirle de apoyo en el mejoramiento continuo del proceso de pruebas, sistematizándolo y simplificando trámites relativos a la evaluación de la calidad del software desarrollado y adquirido por la institución.

1.2 OBJETIVOS

1.2.1 Objetivo General

- Diseñar un prototipo de plataforma basada en herramientas de código abierto, que apoye al personal del CI de la Universidad de Costa Rica en las tareas de V&V de los proyectos de software.

1.2.2 Objetivos Específicos

1. Identificar y seleccionar los estándares de calidad aplicables a los cinco problemas más importantes del CI en materia de V&V de software.
2. Con base en los estándares de calidad seleccionados, proponer soluciones a los problemas escogidos.
3. Identificar las herramientas de software apropiadas para construir un prototipo de plataforma que le permita al CI solucionar los problemas escogidos.
4. Diseñar un prototipo de plataforma computacional que implemente las soluciones encontradas para los problemas identificados en el CI.
5. Evaluar la aplicabilidad del prototipo de plataforma implementado, como apoyo al personal del CI en las tareas de V&V de los proyectos de software.

1.3 DELIMITACIÓN DEL PROBLEMA

El prototipo de plataforma desarrollado al concluir el trabajo de investigación, permitirá gestionar y ejecutar actividades de V&V a proyectos de software desarrollados por el CI, específicamente: verificación de requerimientos a los proyectos y pruebas funcionales de software.

Mediante la plataforma se podrá evaluar la calidad de la especificación de los requerimientos de software del proyecto según las prácticas recomendadas. El personal encargado podrá determinar si los requerimientos del sistema son completos, consistentes, verificables, modificables, inequívocos, correctos, priorizables y trazables.

En cuanto a las pruebas funcionales, la plataforma acompañará el proceso completo. Todo proceso de pruebas atraviesa una serie de etapas, requiere de entradas específicas y genera productos propios. La plataforma participará de todas estas etapas y permitirá la creación y administración de todos los documentos que formen parte del proceso.

A continuación se describen las etapas y productos que manejará la plataforma:

- **Presentación**: posibilitará la elaboración de la s**olicitud de servicio** por parte del personal de desarrollo, mediante la cual se inicia el proceso de pruebas. En esta solicitud se describe el sistema que se evaluará, el tipo de pruebas que requiere y los entregables que se adjunten.
- **Planificación**: permitirá la especificación del **plan de pruebas**. En este documento se define el alcance de las pruebas, lo que se probará y lo que no, los criterios de aceptación o rechazo y los recursos disponibles.
- **Diseño**: contará con espacio para diseñar las pruebas necesarias para validar el correcto funcionamiento del entregable de software. En este espacio también se especificarán los datos de entrada necesarios para ser ejecutados. Al concluir esta etapa, se contará con los **casos de prueba** y los **scripts de pruebas**.
- **Ejecución**: permitirá definir el ambiente necesario para las pruebas de acuerdo a las condiciones planificadas en cuanto a: hardware, software, espacio de laboratorio y recursos humanos. Luego mediante algunas de las herramientas que conforman la plataforma se podrán ejecutar las pruebas solicitadas en la etapa de Presentación.

- **Gestión de Incidencias**: al concluir la etapa de Ejecución, se contará con un **resultado de pruebas**. La plataforma permitirá registrar los errores identificados y construirá un **reporte** especificando las pruebas satisfactorias, las que generaron inconformidades y un conjunto de métricas del proceso de pruebas.

Por otro lado, la plataforma permitirá darle seguimiento a los procesos de V&V desde cualquier etapa del proceso y en cualquier momento. Esto es, conocer cuáles actividades ya se han ejecutado, qué tareas se encuentran pendientes, los responsables de las actividades y cuánto tiempo tienen para completarlas.

Las herramientas que conformen la plataforma serán accedidas únicamente con previa autenticación, ya que de ella dependerá la asignación del rol del usuario (privilegios y capacidades).

Los documentos generados por la plataforma tendrán como base estándares internacionales sobre documentación de pruebas de software (por ejemplo el IEEE 829.1998), adaptados a las necesidades del CI.

CAPÍTULO II: MARCO TEÓRICO

En esta sección se describen los principales conceptos teóricos involucrados en la solución del problema de investigación propuesto. La sección se compone de los siguientes apartados: Aseguramiento de la Calidad del Software, Verificación y Validación de Software, Pruebas de Software, Requerimientos Funcionales y No funcionales, Herramientas para la Administración de Proyectos, Herramientas para el Control de Versiones, *Business Process Management*, Arquitectura de Software y Estándares Internacionales de Calidad.

2.1 ASEGURAMIENTO DE LA CALIDAD DEL SOFTWARE

La plataforma propuesta pretende acompañar y reforzar los procesos de pruebas de software llevados a cabo en el CI, tanto en los sistemas desarrollados como los adquiridos. En el contexto de la ingeniería de software, las pruebas pertenecen al grupo de actividades dedicadas al aseguramiento de la calidad, específicamente del producto desarrollado, de ahí la pertinencia del tema para la investigación.

Lograr un producto o un servicio de calidad es una preocupación creciente en todas las empresas, tomando en cuenta que la variedad y cantidad de ofertas en un mismo nicho de mercado ha convertido a la calidad en un factor determinante y diferenciador al elegir entre un bien (o servicio) y otro [9]. Sin embargo, definir si un producto es de *calidad* ofrece tantas dificultades como distintas definiciones se encuentran para el adjetivo (o sustantivo) calidad.

Según Harvey y Green, calidad significa diferentes cosas para diferentes personas y es relativa a los procesos o a los resultados [10]. Pudiera definirse como un fenómeno excepcional (excelencia, elitismo, perfección), como ajustarse a un estándar o simplemente cumplir con el propósito inicial. Calidad puede ser también lograr las metas institucionales eficientemente o satisfacer las necesidades (explícitas e implícitas) de un cliente.

En la industria, una de las definiciones de uso más generalizado es la ofrecida por el estándar ISO 9000: “calidad es el grado en el que un conjunto de características inherentes cumple con los requisitos [11]". Sin embargo en la industria del software el cumplimiento de requisitos no es tan sencillo de determinar, aun cuando el desarrollo de una aplicación se encuentra íntimamente ligado al concepto *satisfacción de requerimientos*.

El objetivo final de la programación es solucionar un problema de la vida real. La especificación de dicho problema se da en términos de requerimientos de usuario, es decir, las características esperadas en la solución a base de software. Si la calidad puede determinarse por satisfacer o no el “propósito inicial”, desde la perspectiva del desarrollador la calidad es un atributo binario: se tiene o no [12]. Pero, como indican Harvey y Green, “la calidad depende del ojo con que se mire [10]”. Si el usuario es quien juzga la *facilidad de uso* (la relativa facilidad con la que el usuario interactúa con la aplicación) o la *portabilidad* del sistema (capacidad de un sistema de ejecutarse a través de una amplia gama de arquitecturas de hardware), nos topamos con que su cumplimiento no es discreto y la subjetividad puede desembocar en largas discusiones desarrollador-cliente.

Buscando reducir en cierto grado estas discrepancias, se definió que la base que soporta la ingeniería de software es el *enfoque de calidad* y que la gestión de la calidad es una *actividad protectora* que se aplica a todo lo largo del proceso de software [13]. A esta actividad pertenece el aseguramiento de la calidad del software que trata esta sección.

El aseguramiento de la calidad es un conjunto de actividades sistemáticas y planificadas, necesarias para garantizar que los productos y servicios se ajustan a los requerimientos especificados y satisfacen las necesidades del cliente [11]. El aseguramiento de la calidad durante el ciclo de vida permite determinar la necesidad de instaurar o modificar las metodologías utilizadas para el desarrollo, la estimación, el mantenimiento y las pruebas de software [14].

En síntesis, las actividades de aseguramiento de la calidad evalúan y miden el proceso, identificando sus debilidades, corrigiéndolas y estableciendo un proceso de mejora continua. Dentro de estas actividades se encuentran la V&V de software, la gestión de la configuración del software y el control de calidad. Sobre la V&V de software se profundizará en la próxima sección.

2.2 VERIFICACIÓN Y VALIDACIÓN DE SOFTWARE

Evaluar que el sistema producido cumple con las expectativas del usuario es uno de los objetivos de la validación de software. La prueba de software es una manera de determinarlo. Es por ello que *verificación de software* y *validación de software* son conceptos clave para la investigación propuesta.

Las actividades de V&V son parte fundamental del aseguramiento de la calidad del software. Como puede observarse en la figura 1, estas actividades pueden ejecutarse a lo largo de todo el ciclo de vida del software, por ejemplo: iniciando con inspecciones sobre los requerimientos captados, continuando con la evaluación del diseño y el código generado y terminando con la ejecución de pruebas directamente sobre el producto final.

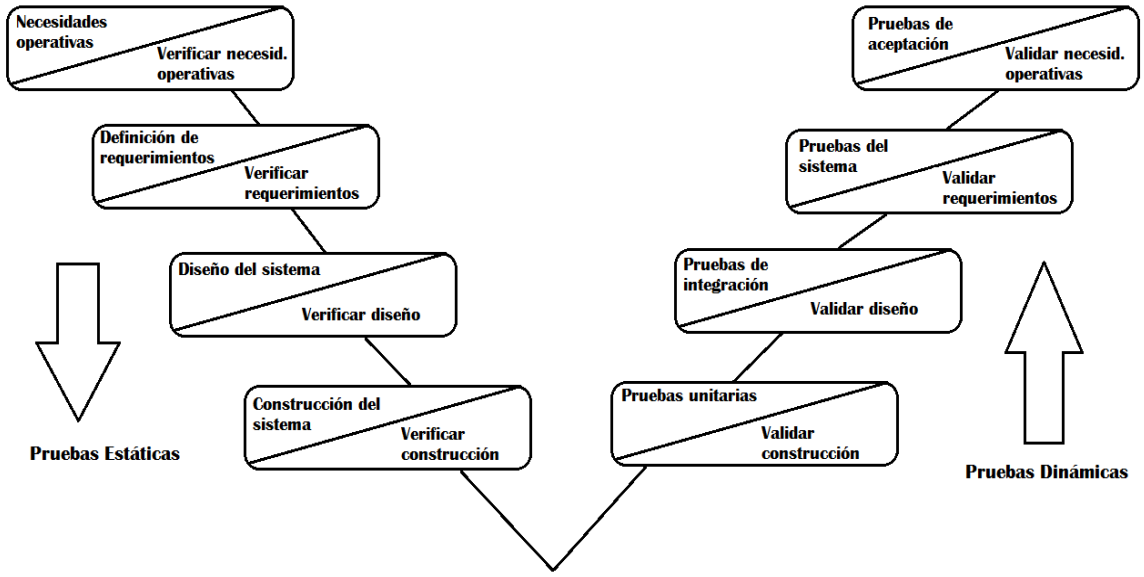


Figura 1: Actividades de V&V a lo largo del ciclo de vida del software

Los objetivos principales de la V&V son [15]:

1. Verificar que los productos obtenidos en cada fase del ciclo de vida:
    a. Cumplan con los requerimientos de la fase anterior.
    b. Satisfagan los estándares, prácticas y convenciones de la fase actual.
    c. Establezcan las bases apropiadas para iniciar la siguiente fase del ciclo de vida.

d. Validen que el producto final cumpla con los requerimientos del software establecidos.

2. Disminuir los riesgos, las desviaciones sobre los presupuestos y sobre el calendario.
3. Valorar rápidamente los cambios propuestos y sus consecuencias.

La verificación de software es el proceso a través del cual se determina si una fase del ciclo de desarrollo cubre o no las necesidades establecidas previamente. Este proceso está constituido por una serie de actividades sistemáticamente ejecutadas a todo lo largo del desarrollo del producto, asegurando que al final del camino, el software implemente correctamente los requerimientos funcionales y no funcionales que le fueron especificados [13].

En la verificación del software intervienen dos criterios fundamentales. Primero el software debe realizar correctamente todas las funciones previstas pero no ninguna función que degrade el rendimiento del sistema en general, ya sea por sí solo o en combinación con otros sistemas. Segundo, deben poderse trazar los requerimientos de software a lo largo de toda la documentación generada en el desarrollo del proyecto de software, comprobando que todos los atributos fueron diseñados, implementados y probados [12].

La validación por su parte, se encarga de la evaluación del software al final del ciclo de desarrollo. Es un proceso más general mediante el cual se asegura que el software satisface las expectativas del cliente. Más allá de comprobar si el sistema está acorde con su especificación, prueba que el software hace lo que el usuario espera que haga.

Dentro de las actividades de V&V del software encontramos las inspecciones y las pruebas de software descritas a continuación:

- Las inspecciones del software analizan y comprueban las distintas representaciones del sistema, por ejemplo: la especificación de requerimientos, los diagramas de diseño y el código fuente del programa. Ocurren en todas las etapas del proceso y pueden complementarse con algún tipo de análisis automático del texto fuente o de los documentos asociados. Estas técnicas se conocen como “pruebas estáticas”, puesto que no requieren que el sistema se ejecute [16].

- Las pruebas de software consisten en examinar el comportamiento operacional de una implementación de software, para comprobar que se desempeña conforme a lo requerido. Las respuestas obtenidas a una serie de datos de entrada se contrastan con las respuestas esperadas. Esta es una técnica “dinámica”, ya que requiere disponer de un prototipo ejecutable del sistema [16].

Sobre esta segunda actividad (las pruebas de software) tratará en mayor detalle la sección a continuación.

2.3 PRUEBAS DE SOFTWARE

La plataforma a desarrollar acompañará el proceso de pruebas de software de principio a fin: planeamiento, diseño, ejecución, evaluación. Con ella se pretende facilitar, agilizar y garantizar la calidad de este proceso. Por ello, comprender los aspectos clave de las pruebas de software es indispensable para conseguir los objetivos de investigación propuestos.

La prueba de software es una actividad en la que un sistema o componente se ejecuta bajo condiciones específicas, se observan o registran los resultados y se realiza una evaluación de un aspecto del sistema o componente [17]. Asimismo, el SWEBOK [18] define la prueba de software como una actividad para evaluar y mejorar la calidad del producto a través de la identificación de defectos y problemas.

Las pruebas de software pueden ejecutarse independientemente del ciclo de vida de construcción del software (cascada, iterativo, espiral) y de la "rapidez" de la metodología aplicada. Deben ejecutarse teniendo en cuenta los siguientes principios [13]:

1. Las pruebas de software son costosas por lo que no es posible realizarlas exhaustivamente. Las pruebas deben detectar el máximo de errores en el mínimo número de casos de prueba.
2. El principio de Pareto es aplicable a las pruebas de software: el 80% de los errores está en el 20% de los módulos. El primer desafío es identificar estos módulos.
3. Empezar por lo pequeño y progresar hacia lo grande: las pruebas comienzan a nivel de componentes y se avanza hacia la integración del sistema completo.

4. Son más eficientes las pruebas ejecutadas por un equipo independiente, manteniendo al equipo desarrollador como apoyo.

El proceso de pruebas puede considerarse como un sub-proyecto dentro del proyecto sobre el cual se están ejecutando las pruebas. En la figura 2 se ejemplifica un posible flujo de ejecución, las entradas y los productos de este sub-proyecto:

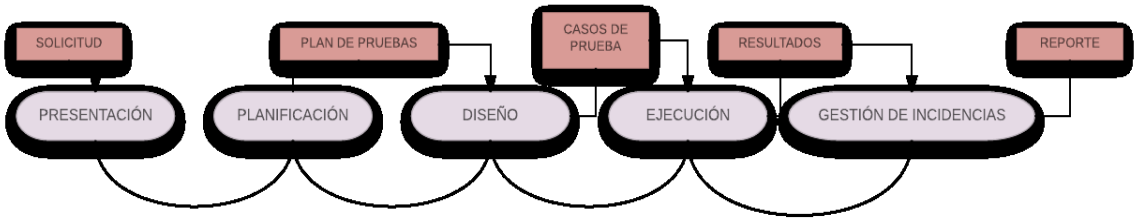


Figura 2: Ejemplo de Proceso de Pruebas de Software

En las siguientes secciones se detalla en las técnicas y niveles de pruebas según Roger Pressman en [13].

2.3.1 Técnicas de Pruebas de Software

Se dispone de tres técnicas para desarrollar las pruebas de software: técnica de caja blanca, técnica de caja negra y técnica aleatoria. Estas se describen a continuación:

1. La técnica de caja blanca se basan en el conocimiento exhaustivo del código fuente del programa e intentan probar tanto código como sea posible. Buscan asegurar que cada camino independiente en el software es ejecutado: todos los ciclos, todas las estructuras condicionales y todas las estructuras de datos.
2. La técnica de caja negra no se preocupan de la estructura del código sino que se basan en las características funcionales del software. Verifican que el comportamiento observado se apegue a las especificaciones del producto y las expectativas del usuario.
3. La técnica aleatoria consiste en la utilización de modelos (usualmente estadísticos) que representen las posibles entradas para crear a partir de ellos los casos de prueba.

De acuerdo a la técnica seleccionada, así se pueden desarrollar distintos tipos de prueba. En el siguiente apartado se caracterizan algunos de los niveles de pruebas de software existentes.

2.3.2 Niveles de Pruebas de Software

En esta sección se describen los niveles de pruebas comunes a la documentación relativa consultada. Se describen según el orden de ejecución recomendado, iniciando con las pruebas unitarias, luego las pruebas de integración y regresión, y por último las pruebas del sistema y de aceptación.

Pruebas Unitarias. Se concentran en probar las unidades más pequeñas del software, como pueden ser el componente o el módulo. Su objetivo es declarar que un módulo está listo y terminado. Para ello se prueban las estructuras de control en sus límites y dentro de ellos, las estructuras de datos para verificar que los datos dentro de ellas mantienen la integridad durante la ejecución del algoritmo y el manejo correcto de excepciones. El principio general de este tipo de prueba es que si los datos no entran y salen correctamente de los componentes, no tiene sentido ejecutar ningún otro tipo de pruebas. Suelen ser ejecutadas por el propio personal de desarrollo, pero evitando que sea el propio programador del módulo.

Pruebas de Integración. Son un tipo de prueba a través del cual se construye la arquitectura del sistema (ensamblando sistemáticamente módulos individuales, luego subsistemas y finalmente el sistema completo), mientras se desarrollan pruebas y se descubren errores en las interfaces de los componentes. Se pueden ejecutar desde dos enfoques distintos: incrementalmente o no incremental. La integración incremental combina el siguiente módulo a probar, con el conjunto de módulos que ya han sido probados. Puede ejecutarse ascendentemente (iniciando desde los niveles más bajos de la estructura del programa) o descendentemente (utilizando el módulo principal como controlador y sus módulos subordinados reemplazados por módulos simulados). Las pruebas de integración no incremental después de haber probado cada módulo por separado, prueban el programa completo.

Pruebas de Regresión. Cada vez que se incluye un módulo en la arquitectura durante las pruebas de integración, se agregan nuevos caminos de ejecución y nuevas entradas y salidas. Esto puede provocar errores en funciones que anteriormente se desempeñaron perfectamente en las pruebas unitarias. Las pruebas de software exitosas son aquellas que descubren errores. Estos errores deben corregirse. Sin embargo, cuando el software se corrige, algún aspecto de la configuración del software cambia (el programa, la documentación, los datos); las pruebas de regresión permiten garantizar que estos cambios no introdujeron nuevos errores. El objetivo de las pruebas de regresión es asegurar que los defectos identificados en la ejecución anterior de la prueba se han corregido y que los cambios realizados no han introducido nuevos defectos. Para lograrlo se repiten pruebas que se han ejecutado anteriormente, pero sobre una nueva versión modificada del producto de software.

Pruebas del Sistema. Las pruebas del sistema no se enfocan en las características internas del producto desarrollado sino en su comportamiento desde un punto de vista general. A este tipo de pruebas pertenecen las pruebas de recuperación, pruebas de seguridad, pruebas de resistencia, pruebas de desempeño y pruebas de despliegue, las cuales se describen seguidamente:

- **Pruebas de recuperación.** Este tipo de prueba provoca que el software falle de diferentes formas y verifica que éste se recupere adecuadamente (reinicio correcto, datos correctos, tiempo transcurrido entre el fallo y la recuperación dentro de lo aceptable).
- **Pruebas de seguridad.** Este tipo de prueba verifica que los mecanismos de protección integrados en el sistema realmente impiden irrupciones inapropiadas.
- **Pruebas de resistencia.** Ejecutan un sistema de manera que se demanden recursos en cantidades, frecuencias o volúmenes anormales. También se les conoce como pruebas de carga o estrés
- **Pruebas de desempeño.** Estas pruebas evalúan el desempeño del software en tiempo de ejecución dentro del contexto de un sistema integrado, verificando la utilización de los recursos (memoria, procesamiento, entre otros).

- **Pruebas de despliegue.** En ocasiones el software puede ejecutarse en variedad de plataformas, por ejemplo, en sistemas operativos distintos, en dispositivos móviles o en la nube. Las pruebas de despliegue o de configuración evalúan la ejecución del software en cada ambiente donde pudiera operar, examinando incluso los procedimientos de instalación.

Pruebas de Aceptación. Son desarrolladas por el cliente y evalúan el conjunto de características y capacidades de los componentes del sistema. Aseguran el trabajo apropiado de los requisitos funcionales, incluyendo la navegación, entrada de datos, procesamiento y obtención de resultados. Por ejemplo, se verifica que los resultados esperados ocurran cuando se usan datos válidos y que se desplieguen los mensajes de error cuando se utilicen datos inválidos. Dado que es virtualmente imposible para el desarrollador prever cómo el cliente utilizará el programa en el día a día, existen dos tipos más de pruebas de aceptación con las que se intenta identificar instrucciones mal interpretadas, combinaciones extrañas de datos, salidas que para el desarrollador tienen sentido pero para el usuario son ininteligibles, entre otros. Las pruebas alfa son pruebas realizadas por el usuario con el desarrollador como observador y en un entorno controlado mientras las pruebas beta son pruebas realizadas por el usuario en su entorno de trabajo y sin observadores.

En el próximo apartado de este marco teórico se definirán los conceptos de requerimiento funcional, requerimiento no funcional y características deseables para ambos.

2.4 REQUERIMIENTOS FUNCIONALES Y NO FUNCIONALES

En secciones anteriores se mencionó que la calidad de un producto puede determinarse por la satisfacción de sus requerimientos iniciales. A su vez, las pruebas de software evalúan si el sistema cumple los requerimientos del usuario y de qué manera los cumple. En la presente sección se desarrolla el concepto de requerimiento y su clasificación en funcionales y no funcionales.

Los requerimientos (o requisitos) de un sistema describen los servicios que éste ha de ofrecer y las restricciones asociadas a su funcionamiento. Según la IEEE, un requerimiento es una condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo, el cual debe estar presente en la solución final, para satisfacer un contrato, estándar, especificación u otro documento formal [17].

Es deseable que los requerimientos sean de la siguiente forma [16]:

- Redactados de manera que sean comprensibles tanto para los técnicos como para los usuarios sin conocimientos avanzados.
- Claros y concretos, evitando imprecisiones o ambigüedades, poseen una sola interpretación.
- Concisos, fáciles de leer y de entender.
- Completos, recolectando todo lo que el usuario necesita.
- Consistentes, sin contradicciones con otros requerimientos.
- Verificables, medibles, de esta manera se puede concluir si se cumplió con ellos o no.

Los requerimientos de software pueden dividirse en dos categorías: funcionales y no funcionales. Los requerimientos funcionales son aquellas características requeridas del sistema que expresan una capacidad de acción, una funcionalidad, y que generalmente es contenida en una declaración en forma verbal. Definen un comportamiento interno del software y suelen provenir de la descripción que los usuarios hacen del software deseado. Por su parte, los requerimientos no funcionales no describen lo que el software hará sino cómo lo hará. Estos establecen restricciones o criterios para juzgar el sistema en su totalidad. Los requerimientos no funcionales definen cualidades generales o atributos que el usuario espera sobre el software resultante, no definen acciones [16]. En la tabla 1 se listan ejemplos de ambos tipos de requerimiento.

Tabla 1: Ejemplos de requerimientos funcionales y no funcionales

Requerimientos Funcionales	Requerimientos No Funcionales
Autenticación de usuario	Seguridad
Creación de cuentas de usuario	Usabilidad
Actualizar información del usuario	Interoperabilidad
Búsqueda de artículo	Tiempo de respuesta
Realizar compra de artículo	Precisión
Dejar retroalimentación sobre transacción	Apego a estándares

Luego de haber profundizado en aquellos conceptos que determinan el marco de trabajo en el que se desarrollará la investigación, en las siguientes secciones se profundizará en temas de particular importancia para la construcción de la plataforma propuesta, siendo el primero de ellos las herramientas para la administración de proyectos.

2.5 HERRAMIENTAS PARA LA ADMINISTRACIÓN DE PROYECTOS

La ejecución de pruebas de software puede verse como un subproceso dentro del ciclo de vida del software. En este subproceso se definen actividades, se asignan responsables, se maneja un cronograma y se generan documentos. Las herramientas para la administración de proyectos se hacen cargo de estos y otros detalles que la plataforma propuesta debe tener en cuenta.

Hoy día la administración de proyectos enfrenta una serie de desafíos que atentan contra la consecución de los objetivos del negocio de una forma eficaz y eficiente. El número de individuos involucrados en el proyecto, su distribución geográfica y las diferencias socio-culturales, hacen que seleccionar de entre ellos a las personas correctas para la realización de una tarea no sea una tarea sencilla. Si a ello se le añade la incertidumbre en cuanto a la duración de las tareas y la coordinación de eventos críticos que dependen de otros, es casi evidente la necesidad de recurrir a herramientas para manejar la complejidad del proyecto [19].

Algunas de las tareas de la administración de proyectos que se pueden facilitar o agilizar a través de herramientas específicas son [20, 21]:

- Planificación: planificar el proyecto a corto y largo plazo, organizar el proyecto en hitos, tareas y sub-tareas, asignar tiempos y recursos materiales y humanos, monitorear el avance del proyecto y reajustar el plan en función de la evolución del mismo.
- Gestión de incidencias: capacidad para incluir nuevas tareas producto de cambios o solicitudes de mejora por parte del cliente, así como poder darle seguimiento a los errores detectados y su corrección.
- Gestión documental: administrar un repositorio donde se almacenen y organicen los documentos obtenidos o generados en el desarrollo del proyecto.
- Control de versiones: capacidad de desarrollo concurrente y mantener un historial del código fuente y de la documentación producida.
- Otros: gestión del cronograma y calendario, control de tiempos, gestión del portafolio de proyectos, generación de informes, construcción de un entorno colaborativo.

A nivel de implementación técnica, los requerimientos de las herramientas son por ejemplo [22, 21]:

- Manejo de múltiples proyectos, manteniendo una visión centralizada pero con posibilidad de configuración personalizada para cada proyecto.
- Administración de usuarios: mantener para cada usuario un perfil con su información personal, contraseña y privilegios asociados. Además la posibilidad de que cada usuario esté vinculado a uno o más proyectos y personalizar su vista sobre la información de cada proyecto.
- Administración de tareas: proveer una vista detallada de las tareas, poder consultar el estado de las mismas, disponer de un histórico de fechas y modificaciones y establecer relaciones de dependencia entre tareas: inicio-inicio, fin-fin, fin-inicio, inicio-fin.
- Notificaciones: generar alertas activas y/o pasivas, enviar mensajes automáticos al correo electrónico informando sobre cambios en el proyecto i.e. nuevos eventos, nuevos *tickets*, asignación de tareas, culminación de tareas, tiempo excedido en el desarrollo de una tarea, entre otras.

- Otras: seguridad de la información, control de acceso, disposición de *chats* o foros para la activa discusión de aspectos sobre el proyecto, necesidad de poco mantenimiento, facilidad de uso, capacidad de personalización e integración con otras herramientas pre-existentes.

En la actualidad existe una amplia gama de herramientas disponibles para la administración de proyectos, algunas meramente empresariales, otras especializadas en proyectos de software e incluso las hay "todo propósito", flexibles y personalizables para que cualquier usuario las pueda utilizar para administrar sus propios proyectos [22]. Se pueden encontrar herramientas privativas (por ejemplo Triskell) o de código abierto (por ejemplo Redmine), de escritorio (como Openproj) o de naturaleza web (como Achievo), siendo estas últimas las más populares porque permiten el acceso a la información desde cualquier computadora sin necesidad de instalación. No posee problemas de compatibilidad con los distintos sistemas operativos ya que su arquitectura cliente-servidor requiere de únicamente un explorador web para establecer conexión con la herramienta.

La administración de un proyecto no sólo contempla la gestión de tareas y recursos humanos, deben administrarse también los productos generados durante su ejecución. Un aspecto clave es el control de las versiones de estos productos. Sobre ello se profundiza en la siguiente sección.

2.6 HERRAMIENTAS PARA EL CONTROL DE VERSIONES

Los sistemas de control de versiones o administradores del código fuente (SCM) contribuyen al proceso de pruebas de software, facilitando la gestión de las versiones del código fuente generado por los probadores (scripts de pruebas), así como de los documentos producidos (plan de pruebas, casos de pruebas, reportes de incidencias). Por ello el control de versiones es un aspecto a tomar en cuenta en el desarrollo de la plataforma propuesta.

Durante el desarrollo de un proyecto de software es común crear, editar, renombrar, modificar y cambiar de posición tanto archivos de código fuente como de configuración. Incluso los manuales de usuario o de instalación atraviesan una serie de transformaciones y cambios realizados por personas distintas a lo largo del proyecto. De pronto, el implementar una nueva funcionalidad introduce errores en el código que se descubren tiempo después o las interacciones

descritas con el sistema cambian con la adquisición de un nuevo hardware por parte del cliente. Estas son solo dos de la gran variedad de situaciones que se pueden manejar mediante las herramientas para el control de versiones.

Por control de versiones se entiende la gestión de los diversos cambios realizados sobre algún producto o una configuración del mismo. Por su parte, una versión es el estado en el que se encuentra este producto en un momento dado de su desarrollo [23].

Algunos de los conceptos básicos que se deben manejar al utilizar una herramienta para el control de versiones son [24]:

1. Repositorio: espacio en el que se almacenan los datos y el histórico de versiones. Usualmente se trata de un servidor al que se accede en una intrared o interred.
2. Copia de trabajo: espacio local de cada computadora donde se descargan los contenidos del repositorio, en ella se realizan las modificaciones y posteriormente se sincronizan con el repositorio.
3. Revisión: es una determinada versión del conjunto total de información almacenada en el repositorio. La revisión más reciente suele llamarse *HEAD*. Cada revisión puede tener asociados metadatos, por ejemplo: responsable de los cambios, fecha y hora de los cambios, revisión de la que se derivó y palabras clave.
4. Línea base: revisión aprobada de la que surgen los cambios subsiguientes.
5. *Tag*: congelar una versión, impidiendo la realización de modificaciones sobre esos archivos.
6. *Trunk*: es el *tronco* de desarrollo principal o versión más estable. Se entiende como la línea base del proyecto en desarrollo de la que pueden derivar *ramas* para desarrollos independientes.
7. *Branch:* rama derivada del tronco principal para la evolución independiente de versiones, por ejemplo al incluirse una nueva funcionalidad o al corregir un error.

Con la utilización de una herramienta para el control de versiones al dirigir o participar de un proyecto se logra [19]:

- Mantener distintas versiones de un archivo, con la posibilidad de recuperar estados anteriores.
- Realizar comparaciones entre los diferentes estados del archivo.
- Poseer un sistema de respaldo en caso de pérdida de la información.
- Resolver conflictos en un entorno de desarrollo colaborativo, cuando un mismo archivo se modifica concurrentemente.
- Mezclar ramas de desarrollo independiente en una nueva versión.
- Bloquear archivos para impedir la modificación de archivos "delicados".

Por la forma en que la información contenida en los proyectos es compartida y manipulada, los sistemas de control de versiones se clasifican en centralizados y distribuidos. En los sistemas centralizados existe un único repositorio del cual se derivan las copias de trabajo (donde los desarrolladores descargan los archivos y los manipulan) y al que se sincronizan los cambios realizados [25]. Ejemplo de esta arquitectura es la implementada por la herramienta Subversion. Por su parte, en los sistemas distribuidos, cada usuario tiene su propio repositorio con la posibilidad de intercambiar y mezclar revisiones entre ellos. Igualmente se mantiene un repositorio común, para el punto de sincronización de los distintos repositorios locales. Esta arquitectura es utilizada por Git.

La próxima sección de este marco teórico profundiza en el tema de la metodología orientada a procesos conocida como *Business Process Management* (BPM).

2.7 BUSINESS PROCESS MANAGEMENT

Con la implementación de la metodología BPM en la presente investigación se pretende que la plataforma le conceda al proceso de pruebas de software simplicidad, agilidad, flexibilidad, transparencia y gobernabilidad, entre las muchas ventajas que se le atribuyen.

Se define *Business Process Management* como un conjunto de métodos, herramientas y tecnologías utilizados para diseñar, representar, analizar y controlar procesos de negocio

operacionales. BPM es un enfoque centrado en los procesos para mejorar el rendimiento que combina las tecnologías de la información con metodologías de proceso y gobierno [26].

Business (negocio, creación de valor), *Process* ("proceso de negocio") y *Management* (administración, lograr que personas y sistemas trabajen por los objetivos del negocio) son los conceptos clave que dirigen el rumbo de esta metodología, siendo "*Process*" el concepto de mayor relevancia y alrededor del cual giran los otros dos.

Según Thomas Davenport [27]: "proceso es un conjunto de actividades estructurado y medible, diseñado para producir una salida especificada para un cliente o mercado particular. Implica un énfasis fuerte en cómo se realiza el trabajo dentro de la empresa, en contraste a un énfasis enfocado en el producto a realizar. Un proceso es así un orden especificado de actividades de trabajo a lo largo del espacio y el tiempo, con un principio, un fin y entradas y salidas claramente especificadas: una estructura para la acción".

Para Smith y Fingaren [28]: "un proceso de negocio es el conjunto completo y coordinado de actividades colaborativas y transaccionales que proporcionan valor a los clientes". En la figura 3 se observa como a lo largo del "proceso de negocio" intervienen diversas personas, varios departamentos de una misma empresa o quizás de distintas organizaciones, todos comprometidos con un objetivo: satisfacer la necesidad del cliente.

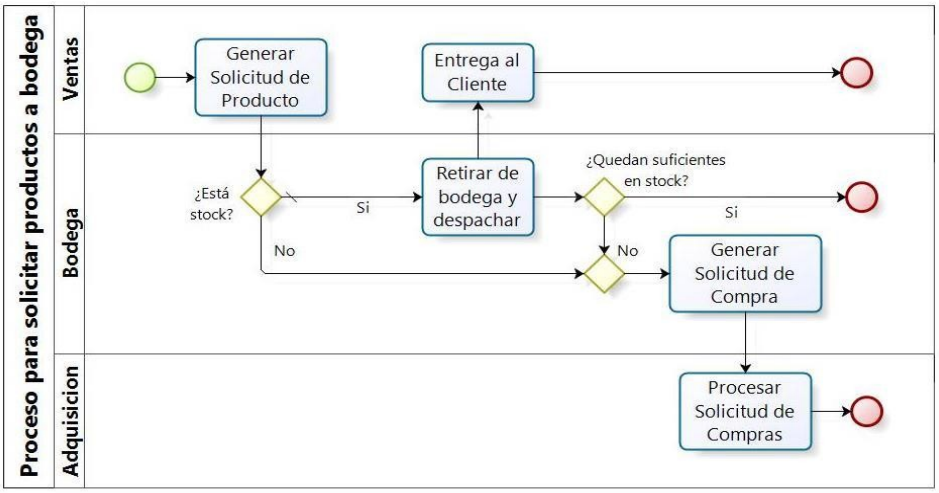


Figura 3: Diagrama del Proceso para Solicitar Productos a Bodega, usando notación BPMN

Con la aplicación de BPM dentro de una empresa, se busca mejorar la eficiencia a través de la gestión sistemática de los procesos de negocio, los cuales deben ser integrales, automatizados, optimizados, monitoreados y documentados de una forma continua, siendo esta una plataforma de soporte en la toma de decisiones gerenciales [26].

La implementación de BPM conlleva los siguientes beneficios [26, 29]:

1. Se generan y controlan los cambios de forma ágil, oportuna confiable y de calidad, con miras al logro de los objetivos estratégicos.
2. Se separa la gestión de los procesos de las aplicaciones, para que cualquier cambio en la lógica de los procesos no suponga ninguna modificación en el código de las aplicaciones.
3. Permite publicar, almacenar, crear, modificar y gestionar procesos, así como acceder a ellos en cualquier momento y lugar.
4. Es posible medir, controlar y responder a todos los aspectos de los procesos operacionales, de manera mucho más sencilla y directa.
5. Se pueden aplicar las habilidades y recursos de los directores de tecnologías de la información de forma más directa en el negocio.
6. Se alinean los esfuerzos de la administración y los empleados con los objetivos empresariales, mejorando así su productividad y su rendimiento personal.
7. Se garantiza el cumplimiento de los requisitos y mecanismos de control internos.
8. Se resuelve los problemas de sobrecarga de información.
9. Se facilita la comunicación y la colaboración entre empleados y departamentos.
10. Se implementan métodos y herramientas de gestión de la mejora continua de procesos.
11. Se genera mayor productividad, coherencia y reducción de errores.
12. Se logra mayor visibilidad/transparencia (seguimiento de transacciones empresariales individuales) a lo largo de todo el proceso y subprocesos
13. Se incrementa la satisfacción del cliente.

A nivel técnico, la implementación de BPM inicia con la selección e instalación de una BPMS (*Business Process Management Suite)*. Estas son aplicaciones con todas las capacidades necesarias para analizar, automatizar, implantar y controlar los procesos. Las BPMS poseen en conjunto

todos los módulos de software funcionando de manera integrada y sin fisuras [26], de manera que sin abandonar el contexto de ejecución se pueden modelar los procesos, ejecutarlos, monitorearlos e incluso modificarlos en tiempo de ejecución [7].

A continuación se muestra un conjunto de funcionalidades ofrecidas por las BPMS en el mercado, recogidas en [7, 26]:

1. Creación, modificación y gestión de procesos empresariales en tiempo real.
2. Definición de reglas de negocio robustas y flexibles.
3. Diseño del modelo de proceso: definición de las tareas que constituyen el proceso, los responsables, las reglas de derivación (secuencial, condicional, evaluación, paralelo), los indicadores clave de desempeño (KPI, *Key Performance Indicators*) e incluso las interfaces de usuario (formularios).
4. Modelado de procesos a través de lenguajes estandarizados: soporte para BPMN (*Business ProcessModel and Notation*) o XPDL (*XML ProcessDefinitionLanguage*).
5. Disposición de herramientas BAM (*Business Activity Monitoring*) que permiten visualizar el rendimiento individual de cada empleado y su efectividad en determinadas tareas dentro del proceso.
6. Asignación automática de actividades por roles o según carga de trabajo.
7. Transferencia de responsabilidades a niveles superiores en caso de emergencia o cumplimiento de criterios de alerta.
8. Simulación de procesos para verificar su comportamiento, dadas diversas condiciones.
9. Monitorización, auditoría y trazabilidad de procesos.
10. Generación de reportes estadísticos y de monitorización: a través de este seguimiento continuo, se obtienen información útil para la futura optimización del proceso.
11. Aprendizaje estadístico: construye una base de estadísticas de rendimiento a partir de criterios centrados en el tiempo, generando automáticamente límites superiores e inferiores, basados en patrones históricos.
12. Coordinación, comunicación y cooperación independiente de la hora y situación geográfica.

13. Disposición de bandejas de entrada de tareas: son la interfaz principal entre el entorno de ejecución de los procesos y el trabajador.
14. Envío de notificaciones a través de correo electrónico.
15. Elaboración del calendario del negocio.
16. Autenticación de usuarios y utilización de firmas digitales.
17. Gestión de *timers* (temporizadores) dinámicos.
18. Gestión de documentos.
19. Repositorio de metadatos: es un contenedor de descripciones, relaciones y políticas de los activos de los procesos.
20. Construcción de cuadros de control para toma de decisiones empresariales.
21. Identificación de "cuellos de botella" o retrasos en la ejecución.
22. Integración con servidores de aplicaciones.
23. Disposición de adaptadores de servicios web: proporciona conexiones con funciones y herramientas ya existentes.
24. Acceso a una única interfaz conocida como Espacio de Trabajo Unificado [29]: por encima de todos los módulos y utilidades de diseño, el usuario final se mueve en un único contexto dirigiendo, consultando y ejecutando cualquiera de los procesos de negocio implementados por la plataforma.

En el listado anterior se indicó que la integración con herramientas ya existentes es una de las funcionalidades ofrecidas por las BPMS. La plataforma propuesta estará integrada por herramientas que de manera individual realizan una función pero que en conjunto abarcaran todas las actividades del proceso de pruebas. La arquitectura del sistema determina de qué manera interactúan estas herramientas. En la siguiente sección se trata este tema en particular.

2.8 ARQUITECTURA DE SOFTWARE

Dado que la plataforma propuesta no es una única herramienta en sí misma sino que estará compuesta de herramientas heterogéneas, debe definirse la manera en que los componentes interactuarán y cómo la información circulará a través de la plataforma. Por ello es necesario dedicarle un espacio a la arquitectura de software en este marco teórico.

La arquitectura de software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente, los principios que orientan su diseño y su evolución [30]. Representa una visión estructural de alto nivel que define y combina estilos para una solución. Usualmente se concentra en los requerimientos no funcionales del sistema (los funcionales deben satisfacerse mediante el modelado y diseño de la aplicación) y es esencial para el éxito o fracaso de un proyecto [31].

En el proceso de definición de la arquitectura intervienen varios conceptos clave, descritos a continuación [32]:

- Estructuras estáticas: definen los elementos en tiempo de diseño y su organización. Son los módulos, servicios, hardware, procedimientos almacenados, paquetes, clases que componen el sistema y cómo estos "encajan" en términos de relaciones, asociaciones y dependencias.
- Estructuras dinámicas: definen los elementos en tiempo de ejecución y sus interacciones. Especifican cómo responde el sistema a los estímulos internos/externos, los flujos de información entre elementos, si se lleva a cabo ejecución paralela o secuencial, entre otros.
- Comportamiento externamente visible: son aquellas interacciones funcionales entre el sistema y su entorno, aquello que realiza el sistema desde el punto de vista del usuario.
- Propiedades cualitativas: aquellas propiedades externamente visibles pero no funcionales. Responden a las interrogantes ¿cómo lo hace? y ¿cómo se comporta?

Es importante acompañar la descripción arquitectural con diagramas que ilustren las estructuras que intervienen en la solución y sus relaciones. Estos diagramas sirven como marco de trabajo a partir del cual se conducen las actividades de diseño más detalladas. Sin embargo no es

posible capturar las características funcionales y cualitativas en un modelo comprensible y de valor para los involucrados en el desarrollo del proyecto [32]. Por ello se recomienda la utilización de vistas, representaciones de uno o más aspectos estructurales específicos.

Eoin Woods & Nick Rozanskien [33] describen con mayor detalle las siguientes vistas arquitecturales:

- Funcional: describe al sistema en tiempo de ejecución a través de sus elementos funcionales, responsabilidades, interfaces e interacciones primarias. En la figura 4 puede observarse un ejemplo de diagrama de vista funcional.
- Información: describe el modo en que la arquitectura almacena, manipula, administra y distribuye la información.
- Concurrencia: describe la estructura de concurrencia del sistema, permite identificar las partes del sistema que pueden ejecutarse concurrentemente y cómo se coordinarlas.
- Despliegue: describe el entorno en el que el sistema será desplegado.
- Desarrollo: describe la arquitectura que soporta el proceso de desarrollo de software.
- Operacional: describe cómo será operado, administrado y asistido el sistema, una vez se encuentre en producción.

Para la definición de estos modelos se recomienda la utilización de lenguajes de descripción arquitectónica (ADL por sus siglas en inglés). Estos cuentan con una semántica y una sintaxis específica para la identificación de elementos y relaciones arquitecturales [34]. Algunos de ellos son: Rapide, Abacus, Aesop, Wright, Acme. La figura 4 corresponde a un diagrama de vista funcional.

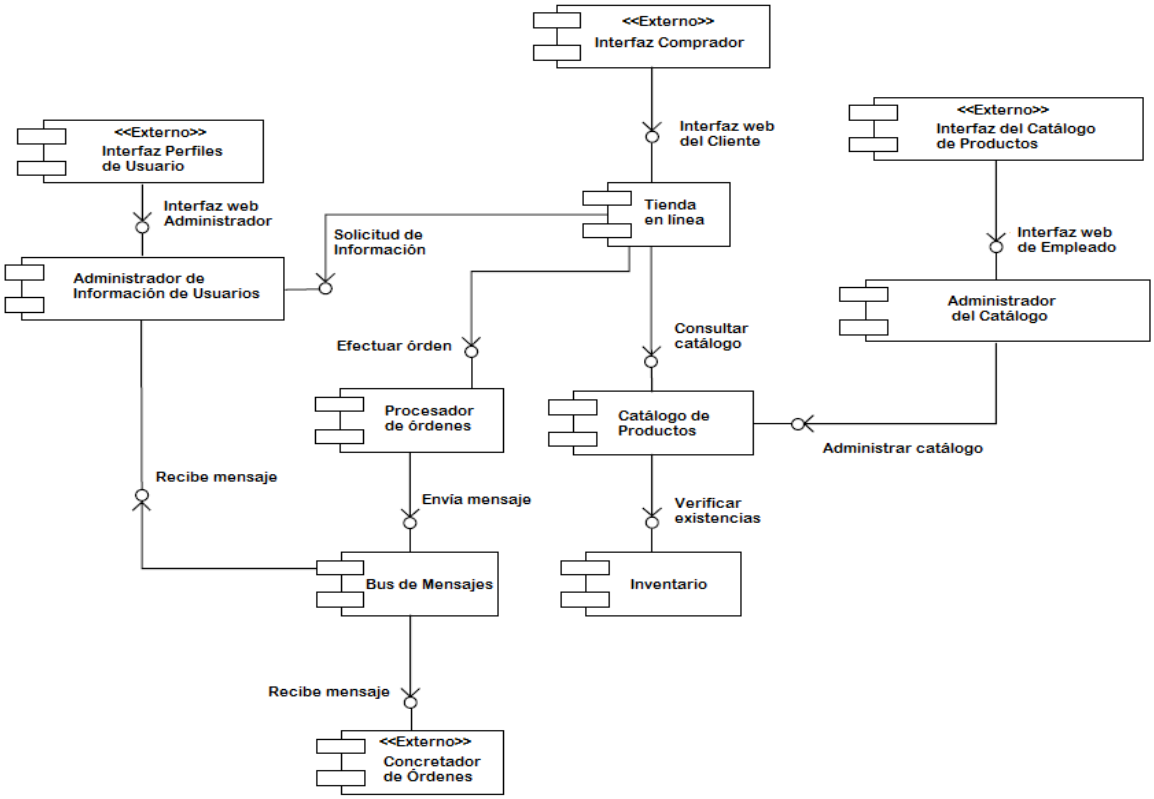


Figura 4: Vista funcional del sistema "Tienda en línea". Basado en [33]

La suite BPM permitirá la ejecución normalizada de una serie de actividades y la generación de documentación, sin embargo tanto el proceso como los documentos producidos no surgirán de forma antojadiza sino que responderán a estándares internacionales como los que tratará la siguiente sección.

2.9 ESTÁNDARES INTERNACIONALES DE CALIDAD

El prototipo de plataforma apoyará a la institución interesada en el seguimiento de estándares de calidad tanto para la ejecución de los procesos de pruebas funcionales y verificación de requerimientos como para su documentación. Conocer cuáles son las entidades que generan estos estándares y específicamente aquellos relativos a la calidad del software es parte fundamental de la investigación, de aquí que se le dedique un apartado en el Marco Teórico.

Una norma es una fórmula que posee valor de regla y tiene por finalidad, definir las características que debe poseer un bien o un servicio [35]. De aquí que la calidad esté relacionada con la normatividad, ya que es el grado en el que un conjunto de características inherentes cumple con los requisitos establecidos en un reglamento. Estas normas pueden ser tanto cualitativas como cuantitativas y aplicarse a nivel empresarial, sectorial, nacional, regional e internacional.

Los estándares internacionales son criterios o reglas establecidos por organizaciones, que ayudan a determinar la conformidad por encima de las fronteras nacionales. Estas reglas facilitan la negociación y colaboración a nivel global. Los estándares internacionales se enfocan en cuestiones como las reglas de unidades de medición, el uso de símbolos y en cómo definir un proceso para cumplir con un control de calidad [36]. En el caso específico de la ingeniería de software los estándares definen un conjunto de criterios que guían la forma en que se determinan los requerimientos, se diseña, se implementa y se evalúa, entre otras cosas. Contar con un estándar permite normalizar los procesos, aumentar la productividad y finalmente mejorar la calidad.

Diversas organizaciones han tomado el liderazgo para establecer, monitorear y administrar estos estándares. Las tres organizaciones que poseen el mayor reconocimiento internacional son la Organización Internacional para la Estandarización (ISO), la Comisión Electrotécnica Internacional (IEC) y la Unión Internacional de Telecomunicaciones (ITU). Además de estas organizaciones, existen muchas más que generan estándares en algún contexto más limitado, como por ejemplo: el Grupo de Trabajo de Ingeniería de Internet (IETF), el *World Wide Web Consortium* (W3C) y el Instituto de Ingeniería Eléctrica y Electrónica (IEEE) [37].

Las siguientes secciones tratan sobre algunos de los estándares emitidos por estas agrupaciones.

2.9.1 Estándares ISO

La ISO es una organización sin fines de lucro y de carácter no gubernamental. Su misión es la de promover el desarrollo de la normalización y sus actividades relacionadas, con el fin de facilitar el intercambio de productos y servicios entre países, así como desarrollar la cooperación internacional en actividades intelectuales, científicas, tecnológicas y económicas [38].

La necesidad de normalizar en el plano internacional se advirtió primero en el campo electrotécnico; como resultado, en 1906 fue creada la Comisión Electrotécnica Internacional (IEC). Después en 1946, delegados de 25 países decidieron crear unan nueva organización internacional con el objeto de facilitar la coordinación internacional y la unificación de normas industriales. Esta nueva organización se llamó ISO y empezó a funcionar el 23 de febrero de 1947. En la actualidad, la ISO agrupa a los organismos nacionales de normalización de 163 países. Las actividades técnicas se encuentran descentralizadas en unos 2700 comités técnicos de normalización, subcomités y grupos de trabajo. Cada uno de estos órganos agrupa representaciones, asociaciones de consumidores y organizaciones internacionales de todo el mundo. El objetivo es resolver aspectos relacionados con la normalización de una forma consensuada [38].

La ISO ha definido un conjunto de estándares de calidad conocidos como los ISO 9000. Estos fueron diseñados como una forma de aplicar técnicas específicas de calidad total (TQM, por sus siglas en inglés) al proceso de software, viéndolo desde un punto de vista de manufactura. Este conjunto de estándares se compone de tres modelos de aseguramiento de la calidad (9001, 9002, 9003) y tres guías para interpretar los modelos (9000, 9004, 9000-3).

Si bien estos estándares no son específicos para una industria (pueden seguirse por desarrolladores de automóviles, secadoras de pelo, televisores, etc.), la industria de software ha sacado también provecho de ellos. Por ejemplo, el ISO 9001 describe el sistema de calidad utilizado para mantener el desarrollo de un producto que implica diseño y el ISO 9000-3 interpreta el ISO 9000-1 específicamente para el desarrollo, la aplicación y el mantenimiento de software [39].

La ISO también ha desarrollado estándares específicos para el software, por ejemplo el ISO 9126 para la “evaluación de la calidad del software”. Este estándar identifica seis atributos de calidad: funcionalidad, confiabilidad, usabilidad, eficiencia, facilidad de mantenimiento y portabilidad [39].

También cuenta con la norma ISO/IEC 14598 que define procesos para el planeamiento y la gestión de la evaluación del producto de software, para desarrolladores, para adquirientes, para avaladores y para la documentación de módulos [40].

Hoy se encuentran ambas reemplazadas por el estándar SQuaRE (*System an Software Quality Requirements and Evaluation*) y el ISO 25000:2005. El objetivo principal de este estándar es guiar el desarrollo de los productos de software con la especificación y evaluación de requisitos de calidad, establece criterios para la especificación de requerimientos de calidad, sus métricas y su evaluación [41].

SQuaRE está integrado por las siguientes divisiones [41]: ISO/IEC 2500n División de gestión de calidad, ISO/IEC 2501n División del modelo de calidad, ISO/IEC 2502n División de mediciones de calidad, ISO/IEC 2503n División de requisitos de calidad, ISO/IEC 2504n División de evaluación de la calidad, ISO/IEC 25050–25099 División de extensión.

Al igual que la norma ISO/IEC 9126, este estándar define tres vistas diferenciadas en el estudio de la calidad de un producto: la vista interna, la externa y la vista en uso [41]. La "vista interna" se ocupa de las propiedades del software como: el tamaño, la complejidad o la conformidad con las normas de orientación a objetos. La "vista externa" analiza el comportamiento del software en producción y estudia sus atributos, por ejemplo: el rendimiento de un software en una máquina determinada, el uso de memoria de un programa o el tiempo de funcionamiento entre fallos. La "vista en uso" mide la productividad y efectividad del usuario final al utilizar el software.

2.9.2 Estándares IEEE

La IEEE es la sociedad técnico-profesional más grande del mundo, dedicada a divulgar los avances de la teoría y aplicación en las áreas de ingeniería eléctrica, electrónica y computación. Surgió en 1963 como resultado de la fusión del American Institute of Electrical Engineers (AIEE) y el Institute of Radio Engineers (IRE), para promover el libre intercambio de información técnica entre ingenieros, para que sus miembros puedan avanzar en sus profesiones y la sociedad pueda comprender la importancia de la ingeniería eléctrica, electrónica y computacional en la vida cotidiana [42].

Al igual que la ISO, la IEEE es una organización sin ánimo de lucro y cuenta con cerca de 425 mil miembros y voluntarios en 160 países, produce más del 30% de la literatura publicada en el mundo sobre ingeniería eléctrica y sus ramas derivadas; posee cerca de 900 estándares activos y otros 700 más bajo desarrollo [43].

Algunos de los estándares relativos a la calidad del software que pueden ser útiles durante la investigación son: IEEE 830 Especificación de Requerimientos Software, IEEE 730 Planes de Calidad de Software, IEEE 1012 Verificación y Validación de Software, IEEE 1028 Revisiones y Auditorías de Software e IEEE 829 Documentación de Pruebas de Software.

De esta forma se concluye el marco teórico de la investigación propuesta. En la siguiente sección se detalla la metodología propuesta para el desarrollo del proyecto de graduación y la consecución de los objetivos de investigación.

CAPÍTULO III: METODOLOGÍA

A continuación se describe en objetivos, actividades y productos la metodología seguida para el alcance de los objetivos de investigación propuestos.

3.1. ANÁLISIS DE LA SITUACIÓN ACTUAL EN EL CENTRO DE INFORMÁTICA

El objetivo de esta actividad fue conocer la manera en que se llevan a cabo los procesos de V&V de software en el Centro de Informática. La idea era profundizar en aquellos detalles que, durante el análisis preliminar llevado a cabo para el Anteproyecto, dejaban entrever carencias y/o dificultades. Se mantuvieron una serie de reuniones (8 sesiones) con el encargado de la Unidad de Calidad y Mejora Continua (UCMC) en el CI y se elaboraron dos cuestionarios para el señor Sub-Director del Centro de Informática (Anexos A y B). Al finalizar esta etapa se debía generar un reporte a manera de diagnóstico sobre la forma en que se trabaja actualmente y seleccionar los cinco problemas más importantes para el CI en materia de V&V de software.

Durante las reuniones con el ingeniero Alonso Alvarado, encargado de la Unidad de Calidad y Mejora Continua, se me facilitaron algunos de los documentos utilizados, para a partir de ellos construir una imagen completa y fidedigna de los procesos, actores, responsabilidades, además de la información recopilada y el uso que se hacía de ella. Los documentos que se me compartieron fueron los siguientes:

- Plantilla para Caso de Uso
- Plantilla Plan de Pruebas
- Plantilla para documentación de Procesos: Elaborar Casos de Uso
- Plantilla para documentación de Procesos: Planes de Pruebas
- Lista Básica para chequeo de programas
- Metodología para la Administración y Gestión de Proyectos de Sistemas de Información de la Universidad de Costa Rica

Se me solicitó generar un reporte con mi percepción sobre "cómo se obtienen los requerimientos del usuario y cómo se validan éstos, en el producto de software desarrollado" y cómo podrían mejorarse dichos procesos.

El documento se corrigió con la retroalimentación recibida de parte de las autoridades del CI, en él se identificaron varias problemáticas en materia de V&V de software y en coordinación con el ingeniero Alonso Alvarado, se eligieron los 5 principales problemas a enfrentar en el presente trabajo de investigación.

3.2 IDENTIFICACIÓN DE LOS ESTÁNDARES INTERNACIONALES APLICABLES A LOS CINCO PROBLEMAS MÁS IMPORTANTES DEL CI EN MATERIA DE V&V DE SOFTWARE

El objetivo de esta actividad fue identificar y seleccionar los estándares relativos a las actividades de V&V de software llevadas a cabo en el CI y que fuesen útiles para hacer frente a los problemas identificados en la fase anterior.

La labor principal de esta fase consistiría en una revisión de los estándares internacionales relacionados con especificación de requerimientos y pruebas de software, con el fin de confirmar algunos, descartar otros y al concluir poseer definido el subconjunto de normas sujeto de adaptación según las posibilidades del Centro de Informática.

Los estándares seleccionados fueron validados en primera instancia, por el criterio experto del comité asesor del presente trabajo de investigación y posteriormente se le presentaron al encargado de la UCMC para su conocimiento y más importante, la retroalimentación sobre las posibilidades de implementación y el posible grado de adaptación necesario. Al finalizar esta actividad se habrá alcanzado el primer objetivo de investigación propuesto.

3.3 ADAPTACIÓN DE LOS ESTÁNDARES SELECCIONADOS CONSIDERANDO LAS CARACTERÍSTICAS PARTICULARES DEL CI

El objetivo de esta actividad fue, con base en los estándares seleccionados, proponer soluciones a los cinco problemas identificados como los más importantes. Una a una las soluciones fueron presentadas al ingeniero Alonso Alvarado, encargado de la Unidad de Calidad y Mejora Continua y fueron depuradas y ajustadas hasta contar con su aprobación.

Al concluir esta actividad, debían de tenerse definidas y aprobadas, todas las tareas, los actores, las responsabilidades y las plantillas de documentación, que integraran los nuevos procedimientos para la verificación de requerimientos funcionales y para la validación de los mismos, en el producto de software desarrollado. Todo sería elaborado en colaboración con el CI, de manear que éstos productos, no solo implementaran las recomendaciones de los estándares internacionales, sino que se ajustaran a las posibilidades de la organización, para facilitar su utilización.

3.4 IDENTIFICACIÓN DE LAS HERRAMIENTAS DE SOFTWARE

El objetivo de esta actividad fue definir los componentes estructurales de la plataforma computacional y la manera en que se relacionarían. El prototipo de plataforma computacional debía implementar las soluciones propuestas en la fase anterior mediante la integración de dos o más herramientas de software, específicamente de código abierto. Por ello fue necesario identificar qué herramientas serían necesarias (gestor de procesos, administrador de incidencias, repositorio versionado, generador de scripts de prueba) y la forma en que la información transitaría entre ellas. Producto de esta actividad fue el modelado arquitectural del prototipo, en su vista funcional y su vista de información.

3.5 DISEÑO Y CONSTRUCCIÓN DEL PROTOTIPO DE PLATAFORMA

El objetivo de esta actividad fue la integración de los componentes de software seleccionados en la etapa anterior, de manera que por la combinación o integración de los mismos, se construya una plataforma computacional que apoye las tareas de V&V de proyectos en el Centro de Informática de la Universidad de Costa Rica.

Como primer paso se configuró cada herramienta por separado: se definieron los grupos y roles de usuario, se incluyeron nuevos campos personalizados y se programaron los flujos de proceso diseñados. Posteriormente se procedió a integrar las herramientas, para dar un sentido de unidad a la plataforma. Ya fuera a través de la integración de sus interfaces o la consulta de sus bases de datos, se buscó que los cambios de contexto de ejecución para los usuarios fueran nulos o mínimos, de manera que la interacción con la plataforma fuera fluida, intuitiva y el intercambio de información entre sus componentes, transparente.

El producto obtenido al concluir la actividad fue el prototipo de plataforma, completamente funcional, implementando las soluciones a los cinco problemas más importantes del CI en materia de V&V de software.

3.6 EVALUACIÓN DEL PROTOTIPO DE PLATAFORMA

El objetivo de esta actividad fue obtener la opinión de los funcionarios del CI, acerca de la aplicabilidad del prototipo desarrollado, como apoyo para las actividades de verificación y validación de proyectos de software.

Durante el análisis efectuado para la elaboración de la Propuesta de Investigación, se identificaron los actores involucrados en las actividades de verificación y validación de software llevadas a cabo en el CI. Estos son: el analista, el encargado de calidad y el usuario final. Por tanto, se consideró que para una evaluación satisfactoria de la plataforma, al menos un representante de cada rol debía ejercitar los procesos y tareas automatizados, para luego obtener su retroalimentación sobre el funcionamiento del prototipo.

Así mismo, se identificó que los proyectos de software desarrollados en el CI pueden ser: internos o externos; y encontrarse en estado: nuevo, en desarrollo o en mantenimiento. Dado que la metodología de desarrollo externo no llegó a conocerse, el prototipo de plataforma debía evaluarse con al menos un proyecto de desarrollo interno, de preferencia en un estado de desarrollo tardío o ya culminado. Además, la evaluación del prototipo se realizaría, no para la evaluación completa del proyecto sino sólo para algunos de sus módulos o funcionalidades.

Finalmente, siempre en el análisis preliminar, se identificaron en el CI, situaciones como las siguientes: las pruebas no se automatizan, no hay evidencia de seguimiento de ningún estándar internacional para los procesos de prueba o su documentación, no hay generación de reportes estadísticos sobre las pruebas, la evaluación de la calidad del producto de software se realiza con base en casos de uso, se utilizan varias herramientas de software privativas. De acuerdo a esto y dependiendo de los problemas seleccionados, algunos aspectos a evaluar en el prototipo se resumen en la tabla 2.

Tabla 2: Posible evaluación del proyecto de graduación

Posible criterio de evaluación	Posible forma de evaluación
Grado de automatización de las pruebas	Determinando la existencia o no de scripts de pruebas y la cobertura de estos respecto al diseño de pruebas.
Presencia y seguimiento de un proceso estandarizado	Utilizando la herramienta BPM para identificar las tareas ejecutadas y las personas que actuaron a lo largo del proceso.
Disposición de métricas y estadísticas del proceso	Determinando la existencia o no de las métricas elegidas por los personeros del CI.
Control de calidad del producto con base en documentos propios del proceso de pruebas	Utilizando la herramienta BPM para corroborar la trazabilidad desde la funcionalidad requerida por el usuario, la definición de sus criterios de aceptación y el resultado de los casos de prueba.
Utilización de herramientas de código abierto	Validación de la aplicabilidad de las herramientas de código abierto propuestas, frente a las utilizadas actualmente

Al concluir la utilización de la plataforma por parte de los personeros del CI, se les proveerá de una encuesta para obtener su criterio y observaciones sobre la aplicabilidad del prototipo de plataforma, como apoyo en las tareas de V&V de los proyectos de software. Con esto, se daría por conseguido el quinto y último objetivo de investigación.

CAPÍTULO IV: RESULTADOS

A continuación se presentan los resultados obtenidos en cada una de las actividades realizadas para la satisfacción de cada uno de los objetivos específicos de la investigación, con el fin de constatar el cumplimiento de cada uno de ellos.

4.1 OBJETIVO ESPECÍFICO I: IDENTIFICAR Y SELECCIONAR LOS ESTÁNDARES APLICABLES A LOS CINCO PROBLEMAS MÁS IMPORTANTES DEL CI, EN MATERIA DE V&V DE SOFTWARE

Como se indicó en la sección 3.1 de la Metodología, a partir de una serie de reuniones y del análisis de documentos, se generó un reporte sobre la situación actual del CI. Se me solicitó específicamente examinar la manera en que se obtienen los requerimientos del usuario y cómo se validan éstos, en el producto de software desarrollado. En la sección a continuación se detallan las actividades que se realizan en el Centro de Informática para la elicitación de los requerimientos de software y su posterior validación:

4.1.1 Situación Actual del Centro de Informática

- El analista asiste a una serie de reuniones con el usuario y con la información recabada en las minutas procede a llenar la Plantilla para Caso de Uso. Dentro de dicha plantilla se completan los siguientes puntos: nombre del caso de uso, actores, descripción, etapa en la WBS, control de cambios, precondiciones, flujo principal, flujos alternos, escenarios, diseño de pantallas y diagrama de clases. Cuando el caso de uso está completo, se envía por correo electrónico al encargado de control de calidad para que lo revise.
- El encargado de control de calidad evalúa la Plantilla para Caso de Uso y completa la Plantilla Revisión General de Documentos (ésta no se me facilitó). Si el Caso de Uso requiere de modificaciones se regresa al analista, cuando éstas sean implementadas, deberá ser evaluado de nuevo por el encargado de calidad. Si el Caso de Uso, cuenta con la aprobación del encargado de calidad, se le envía al usuario por correo electrónico.

- Cuando el usuario asignado al proyecto está familiarizado con la lectura de la Plantilla para Caso de Uso, procede a revisarla y a emitir sus consultas y anotaciones si fuera necesario. Si en cambio, el usuario la desconoce, se convoca a una reunión, se reserva una sala y se le explica cada sección de la plantilla hasta asegurarse que comprende el documento.
- El usuario puede o no solicitar modificaciones al documento. Si no está conforme con el caso de uso, retorna la plantilla al analista para que la mejore y luego pasa al encargado de calidad para una nueva revisión.
- Cuando el caso de uso se encuentra aprobado por el usuario, se llena la plantilla Aprobación de Documentos Generados, y se envía a todos los involucrados para su revisión y para obtener su firma.
- Finalmente el documento físico, con todas las firmas el caso se convierte en un "Entregable de Documentación". La Plantilla de Caso de Uso en versión digital se almacena en Microsoft Sharepoint, en la carpeta "Ejecución y Control/03 - Casos de Uso" mientras que el documento de aprobación en la carpeta "Ejecución y Control/03 - Casos de Uso/Aprobaciones".

La figura 5 ilustra el flujo básico para la obtención de los requerimientos de software en el CI.

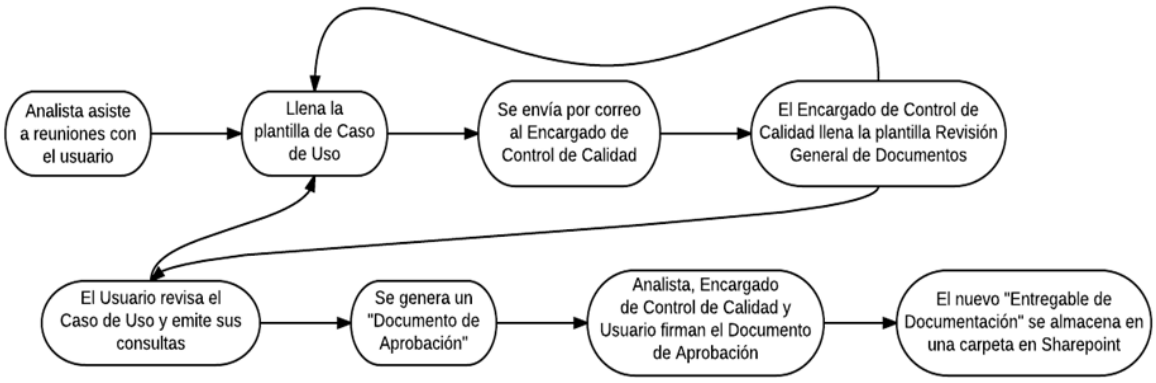


Figura 5: Flujo básico de actividades para la obtención de requerimientos en el CI

Para la validación de los requerimientos de software, se realizan las siguientes actividades:

- Cuando el analista concluye la implementación del comportamiento del sistema especificado en el caso de uso, procede a llenar la Plantilla Plan de Pruebas. En ella se especifican: el nombre del sistema, la pantalla o funcionalidad que se evalúa, los componentes de la aplicación (si requiere componentes adicionales para la prueba), el esfuerzo total estimado en horas, las características a probar y al lado si lo cumple o no, los criterios de completitud y las observaciones del control de calidad.
- Las características a probar son evaluadas por el programador quien, además de la Plantilla Plan de Pruebas, debe utilizar el documento Lista Básica para chequeo de programas. En caso de que se identifique algún error, el programador procede a implementar los cambios necesarios en el código del programa.
- Cuando el programador termina de aplicar satisfactoriamente el plan de pruebas y la lista de chequeo, envía ambos documentos firmados al encargado de calidad. Éste en primera instancia verifica que las características especificadas en la Plantilla Plan de Pruebas, coincidan con las funcionalidades documentadas en el caso de uso respectivo. Seguidamente ejecuta el plan de pruebas y la lista de chequeo. Si identifica errores, le regresa los documentos al desarrollador para que efectúe los ajustes necesarios, si por el contrario, todo está correcto, firma los dos documentos y se los envía al usuario por correo electrónico.
- El usuario debe ejecutar las pruebas según lo señalado en los documentos. Si éste realiza alguna observación o encuentra algún problema durante la validación de las funcionalidades, se retornan los documentos al desarrollador para que implemente las correcciones (en caso de que corresponda). Seguidamente se efectúan de nuevo los dos filtros de calidad (el programador y el encargado de calidad vuelven a ejecutar lo señalado por ambos documentos).

- Cuando el usuario está de acuerdo con lo demostrado por el programa, procede a firmar el documento. Con las 3 firmas el programa se convierte en un "Entregable de Programación". Finalmente los documentos digitales se almacenan en Microsoft Sharepoint, en la carpeta Ejecución y Control/06 - Plan de Pruebas.

La figura 6 ilustra las actividades desarrolladas en el CI para la validación de requerimientos.

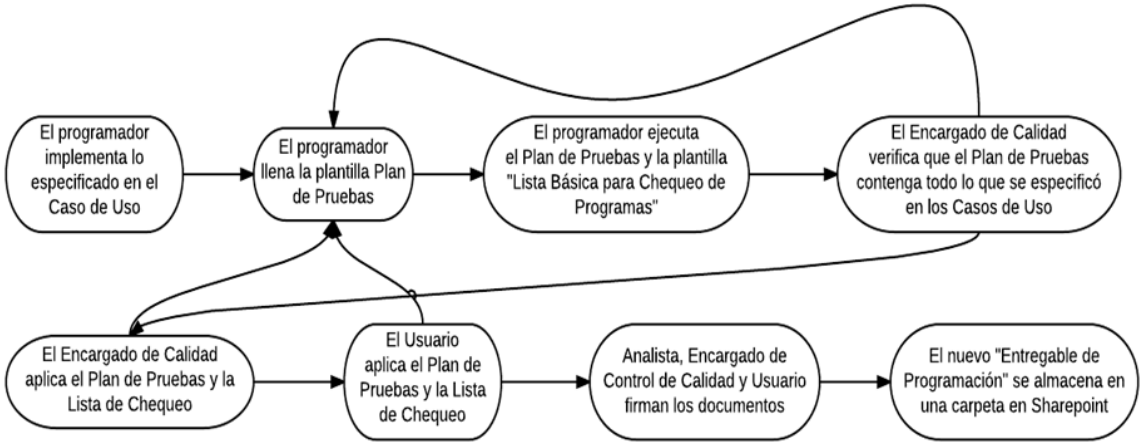


Figura 6: Diagrama de actividades en el procedimiento de validación de requerimientos en el CI

Paralelamente a la especificación de los procesos anteriormente descritos, se llevó a cabo una reestructuración en el Centro de Informática, sobre lo concerniente a "calidad de procesos", "calidad de producto", "equipo de calidad" y "equipo de desarrollo". Esta delimitación de responsabilidades impactó el trabajo de investigación, pues estableció claramente quiénes y qué tareas estarían dentro de su alcance. A continuación, parte de esa distribución de responsabilidades:

a) La Unidad de Calidad y Mejora Continua (UCMC) se encarga tanto de la gestión de calidad de los procesos como de los productos. Esta unidad estaría compuesta inicialmente por 4 personas.

b) El equipo de desarrollo será el responsable de: la generación de los casos de uso, la ejecución de "pruebas internas" al software y enviarle al equipo de calidad evidencia de éstas.
c) El equipo de calidad será el responsable de: validar la concordancia de los casos de uso con la especificación del proyecto o la solicitud del usuario y de dar el visto bueno a todo entregable de software antes de que llegue al usuario.
d) El equipo de calidad, realizará "otras pruebas" tomando en consideración las "pruebas internas" que le haya enviado el equipo de desarrollo. En caso de que el equipo de calidad registre no conformidades, el entregable de software le será devuelto al equipo de desarrollo.
e) Además, el equipo de calidad deberá registrar la "satisfacción del usuario" ya sea a través de encuestas, entrevistas, talleres o grupos focales.
f) Finalmente el equipo de calidad dirigirá esfuerzos en la automatización de las actividades relativas a evaluar la calidad del producto.

Los puntos c, d y e, establecieron como actores a tener en cuenta para el desarrollo de la investigación al equipo de calidad y al usuario. Así mismo, definieron como actividades de V&V de software a apoyar con la plataforma: la verificación de los requerimientos, la ejecución de pruebas funcionales (para dar el visto bueno del entregable de software) y la automatización progresiva de las tareas del equipo de calidad.

4.1.2 Selección de los Cinco Principales Problemas

Se procedió a señalar diferentes problemáticas, específicamente en la manera en que obtienen los requerimientos del usuario y como luego los validan. Al señor coordinador de la UCMC se le presentó una lista con 9 situaciones y junto con él se seleccionaron los 5 más prioritarios. Algunos de los problemas descartados, por ejemplo, enfrentar al usuario a plantillas técnicas o utilizar una herramienta privativa para la gestión documental, se ven también influenciados por las soluciones propuestas. Los restantes problemas, representaban cambios en ciertos paradigmas institucionales, que no se resolverían simplemente con la utilización de nuevas herramientas.

A continuación se justifica la elección de cada uno de los 5 problemas:

4.1.2.1 Problema #1: No hay automatización en las pruebas de software

El proceso de pruebas de software resulta un trabajo intensivo y su ejecución manual lo vuelve una actividad propensa a cometer errores. Basta con mirar de cerca algunas de las actividades propias del proceso, para tener una visión clara de sus dificultades inherentes [44]:

- Modelar el entorno del software: el probador debe simular la interacción entre el software y su entorno, identificar las posibles interfaces que utiliza el sistema (línea de comandos, menús, botones) y las entradas que pueden circular por cada una de ellas. Debe además tomar en cuenta diversos formatos de archivo, protocolos de comunicación e interacción con otros programas.
- Seleccionar escenarios de prueba: ejercitar cada línea de código fuente por lo menos una vez resulta complicado, por ello el probador debe diseñar una cantidad de casos de prueba suficiente para identificar la mayor cantidad de errores.
- Ejecutar los escenarios de prueba: los casos de prueba luego de diseñarse, deben ser ejecutados uno a uno, simulando la acción del usuario.
- Evaluar la ejecución de los escenarios: posterior a la ejecución de las pruebas, se debe proceder a evaluar que los resultados corresponden con las salidas esperadas, tal y como están documentadas en la especificación (que se supone correcta).
- Medir el progreso de las pruebas: número de entradas probadas, porcentaje de código cubierto, número de veces que se ha invocado la aplicación, número de veces que se ha terminado la aplicación con éxito, número de errores encontrados, interpretar estas cifras es difícil, ¿encontrar muchos errores es buena o mala noticia?

La automatización de una (o varias) de estas tareas contribuye no solo a prevenir los errores propios del factor humano sino que también influye en la productividad de los involucrados. En el Centro de Informática el desarrollador llega a dedicar hasta un 40% del tiempo total del proyecto en actividades relacionadas con las pruebas de software y la figura del encargado de calidad dedica hasta 20 horas semanales a la evaluación de documentos y valoración de funcionalidades.

4.1.2.2 Problema #2: El proceso de pruebas no se ejecuta sistemáticamente igual

Al entrar en contacto con ejemplos de las plantillas utilizadas en el proceso de pruebas, destacó el hecho de que no todas se completaban de la misma manera o lo que debía llenarse en una misma sección difería entre las plantillas examinadas. Llamó la atención también, encontrar en los planes de prueba, validaciones del tipo: ¿Usa la tecla ESC para salir?, ¿Los datos y los rótulos están alineados?, ¿Usa términos en castellano?, ¿La pantalla puede minimizarse y maximizarse?; o casos de prueba con los siguientes resultados: N/A, "No se pudo verificar", Sí, No.

En conclusión, el uso de los documentos dejó entrever lo particular del proceso de pruebas implementado en el Centro de Informática y la necesidad de una estandarización que recopilara algunas de las recomendaciones que las normas internacionales contienen al respecto.

La falta de estandarización suele tener efectos negativos, por citar algunos:

- Aumenta la incertidumbre y por tanto la probabilidad de cometer errores.
- Obstaculiza el aprendizaje de las tareas y por ende dificulta alcanzar la eficiencia operativa.
- Entorpece la evaluación de los procesos, la comparación de resultados y en consecuencia la mejoría de éstos.

4.1.2.3 Problema #3: No se generan reportes estadísticos sobre el proceso de pruebas

Durante el establecimiento de la situación actual en el Centro de Informática, en relación con las actividades de verificación y validación de software llevadas a cabo, se consultó específicamente por la recolección de métricas y la generación de reportes estadísticos. La respuesta obtenida en la UCMC fue que la metodología de desarrollo sí tiene un objetivo específico de "mantener un historial de sus proyectos, con el fin de tener referencias, métricas y lecciones aprendidas", sin embargo lo relativo a calidad del producto, no maneja nada al respecto.

El estándar IEEE 1012 Verificación y Validación, recomienda al concluir las actividades de V&V, generar un reporte de finalización en el que se tomen en consideración: los resultados obtenidos, los planes diseñados, los riesgos identificados y las métricas recolectadas. Sobre estas métricas, recomienda evaluar la densidad de las anomalías, la eficiencia y la efectividad de las actividades de V&V.

La generación de los reportes estadísticos, se encuentra muy ligada a conceptos como “toma de decisiones informada” y “mejora continua”. La recolección de datos sobre los procesos, provee una manera sistemática de valorar su calidad, esto pues, el análisis mensurable de las tareas ejecutadas conduce a la identificación de oportunidades de mejora, a la puesta en marcha de acciones correctivas o preventivas y en consecuencia a la estabilización del proceso y al aumento de su calidad. Por estas razones, el hecho de que el CI no recolecte datos de su proceso de pruebas, se consideró uno de los problemas a enfrentar con la investigación

4.1.2.4 Problema #4: El registro de los criterios de aceptación es bastante irregular

Durante el proceso de desarrollo de software en el Centro de Informática, los analistas especifican los requerimientos del usuario de diversas formas: durante la obtención se utiliza la Plantilla de Caso de Uso, luego durante el establecimiento del alcance del proyecto, se utiliza la Matriz de Rastreabilidad de Requerimientos y más recientemente con la implementación de la metodología SCRUM, se elaboran Historias de Usuario.

La Matriz de Rastreabilidad de Requerimientos es el único documento que cuenta con un espacio para el registro de los criterios de aceptación por requerimiento (funcional o no funcional). En los casos de uso y en las historias de usuario, pueden encontrarse dispersos en la prosa de las secciones “Validadores y Condiciones” y “Especificaciones del Mantenimiento”, o no encontrarse del todo. Sin embargo, aun cuando éstos se registraran de alguna manera, al consultar en la UCMC si recurrían a los criterios de aceptación para la generación de los casos de prueba, la respuesta fue negativa: los encargados de calidad no suelen tener acceso a los criterios de aceptación y diseñan las pruebas de acuerdo a su experiencia.

Esta situación se reconoció como problemática y más aún tras examinar el estándar IEEE 1012. La norma internacional establece que los criterios de aceptación, asociados a la definición puntual de la funcionalidad solicitada por el usuario, es uno de los insumos más importantes del proceso de pruebas funcionales de software. El estándar también los reconoce como la principal entrada de las pruebas de aceptación, ya que éstas no pueden llevarse a cabo sin los criterios que determinan si la funcionalidad implementada le satisface o no al usuario.

Finalmente, el estándar IEEE 830 define como una de las características deseables de los requerimientos, la capacidad de ser "verificables". Esto sólo es posible si se cuenta con su(s) respectivo(s) criterio(s) de aceptación.

4.1.2.5 Problema #5: El seguimiento y control de los proyectos es dificultoso

Uno de los reclamos más fuertes escuchados durante el análisis de la situación actual, fue sobre la dificultad para conocer el estado de los proyectos y evaluar su apego a lo planificado. Se señaló como causante directo de esta dificultad, la ausencia de documentación.

La cantidad de documentos solicitada a los proyectos es grande (supera la treintena), sin embargo al momento de realizar el análisis, sólo una cuarta parte de los proyectos en desarrollo la había generado completamente. Debido a esto, para ciertos proyectos, su seguimiento (conocer a los responsables de las tareas, cuánto tiempo se tardó en realizarlas, si alguna tarea fue omitida y por qué) se vuelve complicado.

De manera similar a lo enunciado en el problema anterior, desconocer detalles como el esfuerzo dedicado a una tarea o el flujo de actividades seguido hasta la liberación del producto, impide el análisis sobre la calidad del proceso ejecutado y por consiguiente mejorar la forma en que se realiza. Todo esto a mediano o largo plazo tiene un impacto negativo sobre la imagen de la organización, pues los procesos se vuelven obsoletos y la satisfacción del cliente disminuye progresivamente.

4.1.3 Selección de los Estándares Internacionales

Para la selección de los estándares internacionales aplicables a los problemas descritos anteriormente, se tomaron en cuenta las normas consultadas en la ejecución del proyecto de investigación "B1-723 Creación del Laboratorio de Aseguramiento de la Calidad del Software (LACSOFT)" y en el proyecto "Definición de una metodología de calidad de software para la Universidad de Costa Rica". De todos ellos se consultó su versión más actualizada y en los caso de haber sido sustituidos, se procedió a consultar el nuevo estándar recomendado. La tabla 3 contiene la lista total de normas analizadas.

Tabla 3: Estándares de Calidad para el Desarrollo y Mantenimiento de Software examinados

IEEE 829 Estándar para Documentación de Pruebas	IEEE 830 Práctica Recomendada para la Especificación de Requerimientos de Software
IEEE 1012 Estándar para la Verificación y Validación de Software	IEEE 1028 Estándar para Revisiones de Software y Auditorías
IEEE 1044 Estándar para la Clasificación de Anomalías de Software	IEEE 1061 Metodología de Métricas de Calidad
IEEE 1298 Sistemas de SQA: Requerimientos	ISO/IEC 14598 Evaluación del Producto de Software
ISO/IEC/IEEE 29119 Pruebas de Software	ISO/IEC 9126 Calidad del Producto de Software

De la lista anterior se eligieron 3 estándares: el IEEE 1012 para el apoyo general de las actividades de V&V de software, el ISO/IEC/IEEE 29119 para todo lo concerniente al proceso y documentación de las pruebas de software y el IEEE 830 para mejorar la especificación de los requerimientos.

El estándar IEEE 1012 “Verificación y Validación de Software” fue seleccionado, porque en él se establece un marco de trabajo para los procesos de V&V: define puntualmente las tareas, actividades, entradas y salidas necesarias en cada etapa del ciclo de vida.

El estándar ISO/IEC/IEEE 29119 “Pruebas de Software” fue seleccionado por ser éste el estándar sobre pruebas de software de más reciente aprobación (2013). Define una serie de procesos posibles de implementar en una organización, independientemente del modelo de desarrollo utilizado y apoya pruebas funcionales y no funcionales, manuales y automatizadas, que utilicen *scripts* o sin ellos. Además se encuentra subdividido en 4 documentos, cada uno especializado en una temática particular. De éstos se eligieron el 29119-2 y el 29119-3, los cuales tratan sobre "Procesos de Prueba" y "Documentación de Pruebas" respectivamente.

Por su parte el estándar IEEE 830 “Práctica Recomendada para la Especificación de Requerimientos de Software” desde 1984 (renovado en 1993 y en 1998) define las características deseables para el documento en el que se especifican los requerimientos del sistema a desarrollar. Fue seleccionado porque, de acatar sus recomendaciones, se logra una descripción de requerimientos inteligible para el cliente y útil a los desarrolladores.

4.2 OBJETIVO ESPECÍFICO II: CON BASE EN LOS ESTÁNDARES DE CALIDAD SELECCIONADOS, PROPONER SOLUCIONES A LOS PROBLEMAS ESCOGIDOS

Una vez definidos los problemas de mayor prioridad para el CI, se procedió a proponer soluciones, apoyadas en su mayoría en los estándares de calidad seleccionados. Estas soluciones fueron una a una presentadas al Centro de Informática y algunas de ellas atravesaron un proceso de negociación, hasta obtener su completa aprobación. A continuación las soluciones formuladas.

4.2.1 Solución Propuesta al Problema #1

En el CI "no hay automatización en las pruebas de software", la solución propuesta a este problema fue: la utilización de alguna herramienta para la generación automática de *scripts* de prueba y la automatización de ciertas tareas de la documentación del proceso de pruebas.

Sobre la herramienta generadora de *scripts*, se sugirió específicamente la utilización de alguna del tipo *record and play-back* (grabar y reproducir), para automatizar la ejecución de las pruebas funcionales de software. Estas herramientas permiten grabar las diferentes interacciones que el usuario mantiene con el programa (los valores ingresados, las opciones seleccionadas, los mensajes mostrados, las páginas visitadas, entre otras cosas), almacenarlas a manera de casos de prueba y posteriormente replicarlas de forma acelerada y automática.

A los funcionarios de la Unidad de Calidad y Mejora Continua les pareció una idea atractiva, sin embargo, por las limitaciones de personal, la sobrecarga de proyectos que poseen y la falta de experiencia en el uso de este tipo de herramientas, rechazaron en primera instancia su utilización.

Esta negativa a la automatización de las pruebas, representaba un golpe fuerte a la estrategia planeada para el desarrollo de la investigación. No significaba solamente utilizar una herramienta menos en la plataforma, sino que atentaba de lleno contra la solución del primer problema señalado en las actividades de V&V de software en el CI. Por tanto, se entró en un proceso de negociación en el cual se destacó la gran cantidad de tiempo que estaban consumiendo la ejecución repetida de las pruebas diseñadas (primero por el analista, luego por el encargado de calidad y por último por el usuario) y cómo éstas podían programarse por el analista la primera vez y en las siguientes ser reproducidas automática y rápidamente.

Después de ésta negociación, se tomaron los siguientes acuerdos:

- Las pruebas funcionales no se automatizarían en una primera etapa.
- Las pruebas de regresión sí podrían automatizarse, lo cual significa que en caso de planificarse este tipo de pruebas, se generarían los *scripts* durante las pruebas funcionales para su posterior utilización. Sin embargo es necesario algún tipo de capacitación en el uso de la herramienta propuesta.

Sobre la automatización de las tareas de documentación, se les propuso a los funcionarios de la UCMC el llenado automático de información en los documentos a utilizar, con el fin de agilizar el llenado de las plantillas y garantizar que todas ellas se utilicen. Se generarían automáticamente detalles como nombres, fechas, roles, y la información ingresada en un documento estaría disponible para su reutilización en otro. Esta solución fue aceptada sin objeción alguna.

4.2.2 Solución Propuesta al Problema #2

En el CI "el proceso de pruebas no se ejecuta sistemáticamente igual", la solución propuesta a este problema fue: sistematizar un nuevo proceso de pruebas funcionales, inspirado en normas internacionales, en una plataforma computacional que asegure que su ejecución se llevará a cabo siempre de la misma manera (o registrando evidencia de aquellas ocasiones en que alguna de las tareas no se ejecutó). Esta solución fue desarrollada en 2 etapas, construyéndose primero el nuevo proceso de pruebas funcionales (tareas y actores) y diseñándose después las plantillas para su documentación. En las siguientes secciones se profundiza en lo realizado en ambas etapas.

4.2.2.1 Diseño del Nuevo Proceso de Pruebas

Para el diseño del nuevo proceso de pruebas, se consultó el estándar ISO/IEC/IEE 29119 Pruebas de Software, específicamente el segundo documento que trata sobre Procesos de Prueba. El estándar se construye siguiendo un modelo de 3 capas: "Organizacional", "Administración de las Pruebas" y "Pruebas Dinámicas", definiendo para cada una de ellas tareas, entradas y salidas.

La capa Organizacional establece una serie de pasos para la definición y el mantenimiento de las políticas, estrategias y procedimientos de pruebas a nivel organizacional. Dado que este proceso involucra a las más altas esferas de la organización y se sale de las posibilidades en cuanto a toma de decisiones de la UCMC, se decidió no tomar en cuenta esta primera capa.

La capa Administración de las Pruebas, se encarga de los procesos para gestionar las pruebas y se divide en 3 procesos: “Planeamiento de Pruebas”, “Monitoreo y Control de Pruebas” y “Finalización de Pruebas”. De ellos, se decidió dejar por fuera todas las actividades relativas al proceso Monitoreo. El Centro de Informática está iniciando con la aplicación de la metodología SCRUM para el desarrollo de software y según lo conversado con el señor coordinador de la UCMC, las pruebas a los entregables funcionales debieran realizarse en un período entre 5 y 7 días. Por lo tanto se consideró que las tareas sugeridas para el monitoreo (preparación, monitoreo, control, generación de reportes) sobrecargarían el trabajo de esa semana, tomando en cuenta el total de actividades y tareas del nuevo proceso diseñado.

Sobre los procesos "Planeamiento de Pruebas" y "Finalización de Pruebas" fueron simplificados al extremo. Para la planificación de las pruebas el estándar define 9 actividades, cada una con una entrada o salida propia. Se consideró que muchas de ellas significan realmente una serie de pasos preparatorios para finalmente registrar la información en plan de pruebas, por ejemplo "organizar el desarrollo del Plan de Pruebas, "obtener consenso sobre el Plan de Pruebas", son actividades sugeridas por el estándar que no tienen razón de ser en el CI. En la UCMC el gestor de calidad tiene la potestad de planificar las pruebas sin someterlo a aprobación y sobre la organización del documento no es necesario "reflexionar" porque sólo se decidió como obligatorio, el establecimiento del alcance de las pruebas (qué se probará y qué no). El resultado total fue dejar una única tarea que combina la creación del plan de pruebas con su divulgación.

En cuanto a la simplificación del proceso "Finalización de las Pruebas", el estándar especifica 4 actividades: "Archivar los Activos de Prueba", "Limpiar el Ambiente", "Identificar las Lecciones Aprendidas" y "Generación del Reporte de Finalización". Todas ellas fueron resumidas en una única tarea, en la cual, se genera el reporte, con espacio para registrar las lecciones aprendidas. Todos los activos de prueba quedan automáticamente almacenados en las bases de datos de la plataforma y sobre la limpieza del ambiente, se incluiría solamente un recordatorio.

Finalmente la capa de “Pruebas Dinámicas” está integrada por los procesos que definen las tareas relativas al diseño de pruebas, ambientación, ejecución de pruebas y reporte de incidencias. Todos estos procesos fueron tomados en cuenta para el diseño del nuevo proceso de pruebas funcionales y también requirieron de adaptación a las particularidades de la UCMC.

En el proceso "Diseño de Pruebas", fueron eliminadas las tareas relativas a la creación de conjuntos "lógicamente relacionados" de requerimientos funcionales y de casos de prueba. En la UCMC se acostumbra a ejecutar secuencialmente los casos de prueba, por lo que se consideraron actividades innecesarias.

Debido a que en la UCMC, trabajan 4 personas y es común que una misma persona cree las pruebas y las ejecute, las actividades sugeridas sobre la "Ambientación de las Pruebas", fueron suprimidas como parte de un proceso independiente, y se fusionaron con el diseño de las pruebas, de tal forma que antes de la ejecución, se asegure que los requerimientos del ambiente están preparados.

Finalmente para el proceso de "Ejecución de las Pruebas", se eliminó la necesidad de llevar una bitácora de pruebas, esto con el objetivo de reducir la documentación a utilizar. Y el proceso de "Finalización de las Pruebas" sí se dejó tal y como lo especifica el estándar.

En la figura 7 se ilustran en color verde los procesos del estándar ISO/IEC/IEEE 29119 que se tomaron en cuenta y en color rojo los que fueron dejados de lado.

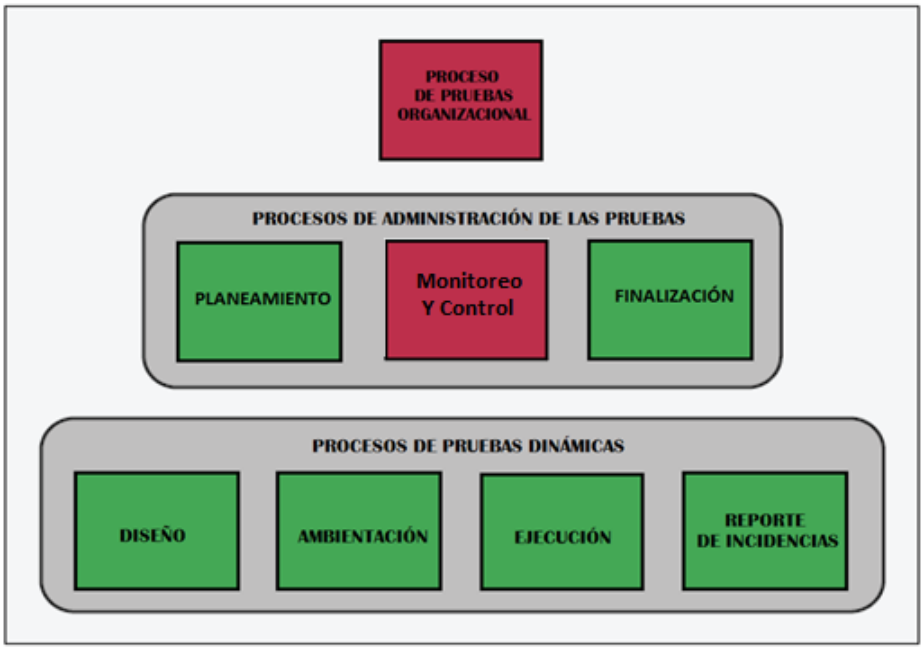


Figura 7: Modelo por Capas basado en el definido por el estándar ISO/IEC/IEEE 29119

Mientras se realizaba esta selección de subprocesos a sistematizar en la plataforma, la UCMC trazó toda una nueva "hoja de ruta" para la unidad. Los conceptos de verificación y validación de requerimientos manejados hasta el momento en el CI se vieron ampliados, gracias a talleres y capacitaciones sucedidas en paralelo al desarrollo de esta investigación y en consecuencia también se vieron ampliadas las expectativas sobre las posibilidades de la plataforma propuesta.

En la nueva "hoja de ruta" el equipo de calidad, sería responsable de las siguientes actividades:

- Verificación de Requerimientos
- Verificación de Diseño/Accesibilidad
- Verificación de Arquitectura y Bases de Datos
- Pruebas de Portabilidad
- Pruebas de Seguridad
- Pruebas Funcionales (Exploratorias y del Sistema)
- Pruebas de Aceptación

Debió entrarse en un proceso de negociación para definir, de qué manera el trabajo de investigación apoyaría, o no, estas nuevas actividades. Se acordó lo siguiente para cada actividad:

- Verificación de Requerimientos: desde un principio se consideró parte del alcance del proyecto por lo tanto no significaba ninguna novedad.
- Verificación del Diseño/Accesibilidad: esta evaluación será efectuada con base a directrices que la Oficina de Divulgación de la Universidad de Costa Rica aún debe definir. Se acordó de momento dejarla fuera de la plataforma.
- Verificación de Arquitectura y Base de datos: aún no se define de qué manera se realizará esta verificación, pero se piensa en consultar a expertos en el campo para obtener su visto bueno. Se le ofreció al CI un espacio al final de la verificación de requerimientos, para registrar la aprobación o no de los diseños propuestos.
- Pruebas de Portabilidad: se realizó una breve revisión bibliográfica sobre cómo realizar pruebas de portabilidad en distintos sistemas operativos, navegadores web y dispositivos móviles (Anexo C). Se le mostró al señor coordinador de la UCMC la variedad de pruebas que podían realizarse (adaptabilidad, compatibilidad, interoperabilidad, internacionalización, entre otras) y se llegó al acuerdo de que ni el postulante ni los funcionarios de la UCMC contábamos con la suficiente experiencia para desarrollar pruebas de portabilidad y que el sólo hecho de buscar las herramientas idóneas para integrar la plataforma retrasaría el avance de la investigación. Por todo esto, fueron descartadas.
- Pruebas de Seguridad: estas fueron descartadas por el postulante debido a la complejidad que agregaban a la solución.
- Pruebas Funcionales: dado que las pruebas exploratorias son muy similares a las pruebas funcionales, y determinan si el software se encuentra preparado para la ejecución de pruebas más formales, se le ofreció al CI un espacio en la plataforma para registrar los resultados de las pruebas exploratorias.
- Pruebas de Aceptación: el señor coordinador de la UCMC destacó la importancia de medir la satisfacción del usuario con el producto. Se acordó agregarle a la plataforma un espacio para registrar la opinión del usuario sobre el sistema desarrollado.

Según lo anterior, las actividades de V&V de software que se apoyarán con el prototipo de plataforma computacional y solución final para el problema #2, se muestran en la figura 8.

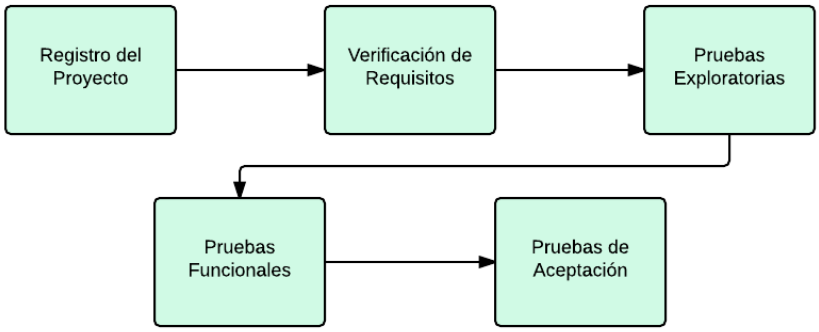


Figura 8: Modelo de proceso para las actividades de V&V de software en el Centro de Informática

A continuación se detallan las actividades de este proceso modelado.

- Registro del Proyecto: La tarea inicial del proceso corresponde al ingreso de la información básica para identificar el proyecto de pruebas.

- Verificación de Requisitos: El objetivo propuesto para esta actividad es, determinar si los requerimientos del sistema son completos, consistentes, verificables, modificables, inequívocos, correctos, priorizables y trazables.

- Pruebas Exploratorias: Las pruebas exploratorias se ejecutan para determinar si el software se encuentra lo suficientemente estable para proceder con las pruebas funcionales, a la vez que se obtiene conocimiento de la aplicación a probar, lo cual es de mucha utilidad para darle agilidad al diseño de las pruebas funcionales.

- Pruebas Funcionales: El modelado de esta actividad se basó en el estándar ISO/IEC/IEEE 29119-2, específicamente en los procesos “Planificación de Pruebas”, “Finalización de Pruebas” y la capa de “Pruebas Dinámicas” del estándar. Por tanto las pruebas

funcionales están integradas por las tareas Planificación, Diseño, Ejecución y Registro de Incidencias, Finalización. En la tarea Planificación, se construye el documento Plan de Pruebas. En él se define el alcance de las pruebas, en otras palabras, qué funcionalidades se van a probar y cuáles no. Se establece la estrategia de pruebas, esto es qué tipo de pruebas se ejecutarán, los criterios para suspenderlas, reanudarlas y considerarlas completas. Finalmente se registran los recursos con los que se dispone para realizar las pruebas, en términos de personal y tiempo. Durante la tarea Diseño de Pruebas se crean los casos de prueba, se especifican los datos de entrada necesarios para ejecutar los casos y también las condiciones del ambiente (hardware, software) requeridas para ejercitar el programa. En la tarea Ejecución de Pruebas y Registro de Incidencias, el sistema se ejecuta según lo especificado en el diseño de pruebas y al mismo tiempo se registran los errores. Cuando el resultado obtenido en un caso de prueba no corresponde con el esperado, debe reportarse una incidencia y así dejar constancia del error. Estos reportes deben ser claros y detallados, de manera que el error pueda luego localizarse, reproducirse y solucionarse. Al concluir, en la tarea de Finalización, se elabora el Reporte de Finalización. En éste se registran las pruebas satisfactorias, los errores encontrados y un conjunto de métricas. También se establece si se cumplieron o no los criterios de completitud de las pruebas.

- Pruebas de Aceptación: Antes de liberar el entregable de software, es necesario determinar si éste satisface realmente los requerimientos del usuario. Para el CI no sólo es suficiente con validar que las funcionalidades solicitadas se encuentren correctamente implementadas, sino que también consideran importante validar que fueron implementadas tal y como el usuario las necesita.

De acuerdo a los modelos, actividades y tareas definidas, se propusieron los siguientes roles y responsabilidades:

- Líder de Pruebas: encargado del Registro de Proyecto, de las Pruebas Exploratorias y la Planificación de las pruebas funcionales.

- Encargado de Pruebas: responsable del Diseño de las pruebas.
- Soporte Técnico: responsable de la ambientación de la pruebas.
- Probador: encargado de ejecutar los casos de prueba y registrar las incidencias.

En su mayoría fueron todos aprobados, solo debió eliminarse el rol de Soporte Técnico, de manera que la ambientación sea responsabilidad del Encargado de Pruebas y al Líder de Pruebas se le añadió la responsabilidad de dar seguimiento al proceso a lo largo de las diferentes tareas.

4.2.2.2 Documentación del Proceso de Pruebas

Las plantillas construidas para la documentación del nuevo proceso de pruebas diseñado, tienen dos orígenes bien definidos: aquellos pertenecientes a la fase de Pruebas Funcionales tienen como base el estándar ISO/IEC/IEEE 29119-3, los restantes documentos son de creación propia. Todos finalmente fueron depurados, tomando en cuenta las recomendaciones y solitudes de los actores implicados.

A continuación se especifica el contenido de las plantillas producidas para las fases Registro del Proyecto, Verificación de Requisitos, Pruebas Exploratorias y Pruebas de Aceptación:

- Registro del Proyecto de Pruebas: este documento recopila los datos mínimos para identificar el proyecto de pruebas y para darle seguimiento. Los datos recogidos son nombre del proyecto, número de iteración de desarrollo, módulo o producto a probar, unidad que solicitó el proyecto, gestor de calidad responsable, fecha de inicio (Anexo D).

- Guía para la Revisión de Requerimientos en los Desarrollos de Sistemas de Información: se le ofreció a la UCMC una lista de chequeo de 52 preguntas, basada en el estándar IEEE 830 “Práctica Recomendada para la Especificación de Requerimientos”, para la evaluación de la calidad de los requerimientos especificados. Sin embargo esta propuesta no fue del todo aceptada, y decidieron utilizar la lista sólo como herramienta de consulta (Anexo E).

- Evaluación de Caso de Uso: consiste en una tabla con 13 preguntas del tipo Sí o No y con la posibilidad de añadir comentarios. Estas preguntas están basadas principalmente en las secciones que integran la plantilla de CU, y sólo se incluyó una validación sobre la descripción correcta, no ambigua y verificable de la funcionalidad requerida (Anexo F).

- Evaluación de Historia de Usuario: de manera similar al anterior posee 16 cuestionamientos, con respuesta Sí o No y la posibilidad de añadir comentarios. Sin embargo, en cuanto a la naturaleza de la evaluación, la mitad de las preguntas giran en torno a detalles de formato y la otra mitad verifica detalles recomendados por la industria, sobre la correcta construcción de las Historias de Usuario (Anexo G).

- Resultado de las Pruebas Exploratorias: en este documento se registran los resultados de la evaluación preliminar del entregable de software. Se recoge el nombre del proyecto de pruebas, módulo o producto evaluado, gestor de calidad responsable de las pruebas exploratorias, funcionalidades probadas, resultados obtenidos y finalmente la decisión "de calidad" sobre continuar con las pruebas funcionales o no (Anexo H).

- Resultado de las Pruebas de Aceptación: consiste en un formulario (Anexo I), en el que el usuario registra los resultados (qué funcionalidades ejercitó, los defectos encontrados y la evidencia al respecto) de la prueba del sistema realizada en un ambiente controlado, antes de lanzarlo a producción.

Para la documentación de la etapa de Pruebas Funcionales, el estándar ISO/IEC/IEEE 29119-3 dispone de hasta 12 documentos, pertenecientes a las capas "Administración de las Pruebas" y "Pruebas Dinámicas". En un esfuerzo por la simplificación y evitar el rechazo a los nuevos documentos, al Centro de Informática le fueron propuestas 4 plantillas.

En las tablas 4, 5, 6, 7 y 8 se detallan los documentos seleccionados, los unificados y la justificación de aquellos que fueron omitidos por completo. Se resaltan con color rojo las secciones del estándar que fueron descartadas y en color azul las que fueron agregadas específicamente para el Centro de Informática.

Tabla 4: Propuesta de Plantillas para Documentar el Proceso de Pruebas Funcionales en el CI

Documentos ISO/IEC/IEEE 29119-3	Documentos Propuestos	Adaptación Realizada
Plan de Pruebas	Plan de Pruebas (Anexo J)	
Especificación del Diseño de Pruebas	Diseño de Pruebas (Anexo K)	El Diseño de Pruebas propuesto, resulta de la combinación de elementos de 6 documentos sugeridos por el Estándar.
Especificación del Caso de Prueba		
Requerimientos de los Datos de Prueba		
Requerimientos del Ambiente de Pruebas		
Reporte de Preparación de los Datos de Prueba		
Reporte de Preparación del Ambiente de Pruebas		
Reporte de Incidente	Reporte de Incidente (Anexo L)	
Reporte de Finalización de las Pruebas	Reporte de Finalización (Anexo M)	
Especificación del Procedimiento de Prueba	Omitido	Este documento organiza los casos de prueba en conjuntos y establece una secuencia para su ejecución. Fue considerado un paso extra que no contribuía a la agilización del proceso.
Bitácora de Ejecución de Pruebas	Omitido	En este documento se registran los detalles sobre la ejecución de uno o más procedimientos de prueba. Dado que los casos se ejecutarán uno a uno tal y como fueron diseñados, el documento se consideró innecesario.
Reporte sobre el Estado de las Pruebas	Omitido	Corresponde a uno de los documentos propios del Monitoreo de las Pruebas, actividad que no fue tomada en cuenta para el modelo de proceso propuesto. Como el mismo estándar recomienda, para metodologías ágiles, puede no ser un documento escrito sino únicamente un tema de discusión en las reuniones durante el *sprint*.

Tabla 5: Adaptación del Plan de Pruebas con respecto al Estándar IEEE 29119

Contenido del Plan de Pruebas según el Estándar	Contenido del Plan de Pruebas propuesto	Omitidos
a) Contexto de las pruebas: i) Proyecto ii) Ítems de prueba iii) Alcance de las pruebas iv) Supuestos y restricciones v) Interesados	a) Contexto de las pruebas: i) Proyecto ii) Módulo o producto iii) Número de Iteración iv) Características que se probarán (combinación Ítems de prueba y Alcance)	-"Supuestos y restricciones": no proveen un valor agregado al proceso actual. -"Interesados": siempre son los mismos (cliente y CI). El cliente se define en la plantilla de Registro de Proyecto. -"Comunicación de las pruebas": la metodología de desarrollo ya posee una Matriz de Comunicaciones. -"Registro de riesgos": la metodología de desarrollo ya posee tres documentos para la documentación y evaluación de los riesgos. -"Sub-procesos de prueba": el plan es relativo a una única iteración. -"Métricas que se recolectarán": siempre son las mismas. -"Desviaciones de la Estrategia Organizacional": no existe una estrategia organizacional per sé. -"Actividades de prueba y estimados": se consideró suficiente el Cronograma. -"Necesidades de contratación y de entrenamiento": de momento no son necesarias, pues el personal de pruebas de momento será la UCMC
b) Comunicación de las pruebas		
c) Registro de riesgos i)Riesgos del producto ii)Riesgos del proyecto		
d) Estrategia de pruebas: i) Sub-procesos de prueba ii) Entregables de prueba iii) Técnicas de pruebas iv) Criterios de completitud v) Métricas que se recolectarán vi) Requerimientos de los datos de prueba vii) Requerimientos del ambiente de pruebas xi) Pruebas de regresión xii) Criterios de suspensión y reanudación xiii) Desviaciones de la Estrategia Organizacional	b) Estrategia de pruebas: i) Entregables de prueba ii) Nivel/Tipo/Técnica de prueba iii) Criterios de completitud iv) Criterios de suspensión y reanudación v) Requerimientos de los datos de prueba vi) Requerimientos del ambiente de pruebas vii) Pruebas de Regresión	
e) Actividades de prueba y estimados		
f) Personal: i) Roles, actividades y responsabilidades ii) Necesidades de contratación iii) Necesidades de entrenamiento	c) Personal: i) Funcionario ii) Rol iii) Responsabilidades iv) Horas dedicadas	
g) Cronograma	d) Cronograma: i) Tarea ii) Responsable y participantes iii) Esfuerzo iv) Fecha de finalización	

Tabla 6: Adaptación del Diseño de Pruebas con respecto al Estándar IEEE 29119

Contenido del Diseño de Pruebas según el Estándar	**Contenido del Diseño de Pruebas propuesto (continuación)**	**Omitidos**
a) Conjunto de características i) Identificador único ii) Objetivo iii) Prioridad iv) Estrategia específica v) Trazabilidad	a) Características que se probarán i) Identificador único ii) Requerimiento ii) Descripción de la funcionalidad iii) Criterio de aceptación iv) Prioridad	-"Objetivo": no se consideró que otorgara valor agregado al documento. -"Estrategia específica": se decidió especificar lo suficiente la Estrategia del Plan de Pruebas y así simplificar el documento de Diseño. -"Trazabilidad": no se omitió pero se interpretó como la inclusión del identificador único del requerimiento. -"Condiciones de prueba" e "Ítems de cobertura de pruebas": fueron eliminados por considerarse que volvían muy compleja la especificación de casos de prueba. -"Resumen": considerado innecesario. -"Salida obtenida": se decidió registrar solo las incidencias y no así los casos satisfactorios. Estos se determinarían de manera automática.
b) Condiciones de prueba i) Resumen ii) Identificador único iii) Descripción iv) Prioridad v) Trazabilidad		
Contenido del Caso de Pruebas según el Estándar		
a) Ítems de cobertura de prueba i) Resumen ii) Identificador único iii) Descripción iv) Prioridad v) Trazabilidad		
b) Casos de prueba: i) Resumen ii) Identificador único iii) Objetivo iv) Prioridad v) Trazabilidad vi) Pre-condiciones vii) Entradas viii) Resultado esperado ix) Salida obtenida	b) Casos de prueba i) Identificador único ii) Objetivo iii) Prioridad iv) Inicialización v) Pre-condiciones vi) Entradas vii) Resultado esperado	

Contenido de Requerimientos de los Datos de Prueba según el Estándar	Contenido del Diseño de Pruebas propuesto	Omitidos
a) Detalle de los requerimientos para los datos de prueba i) Resumen ii) Identificador único iii) Descripción iv) Responsable v) Período de necesidad vi) Necesidad de limpieza vii) Archivado o desecho	c) Requerimientos de los datos de prueba i) Descripción ii) Responsable iii) Archivado o Desechado iv) Observaciones	-"Resumen Requerimiento de Dato": se consideró innecesario dado que se eliminó el rol de Soporte Técnico. -"Identificador de Requerimiento de Dato": se consideró suficiente la descripción. -"Período de Necesidad": se consideró innecesario. -"Necesidad de limpieza": se interpretó limpieza como que existe la posibilidad de que algunos datos deban volver a su estado inicial. -"Preparación de los datos": al eliminarse el rol de Soporte Técnico, el mismo diseñador de pruebas determina si el requerimiento está satisfecho o pendiente. -"Resumen Requerimiento de Ambiente": se consideró innecesario. -"Identificador de Requerimiento de Ambiente": se tomó por suficiente la descripción. -"Preparación del ambiente": al igual que con los datos, al eliminarse el rol de Soporte Técnico, el mismo diseñador de pruebas determina si el requerimiento está satisfecho o pendiente.
Contenido del Reporte de Preparación de los Datos según el Estándar		
a) Preparación de los datos i) Resumen ii) Identificador único iii) Descripción del estado	v) Estado	
Contenido de Requerimientos del Ambiente de Pruebas según el Estándar		
a) Detalle de los requerimientos del ambiente de pruebas i) Resumen ii) Identificador único iii) Descripción iv) Responsable v) Período de necesidad	d) Requerimientos del ambiente de prueba i) Descripción ii) Responsable iii) Observaciones	
Contenido del Reporte de Preparación del Ambiente según el Estándar		
a) Preparación del ambiente i) Resumen ii) Descripción del estado	iv) Estado	

Tabla 7: Adaptación del Reporte de Incidente con respecto al Estándar IEEE 29119

Contenido del Reporte de Incidente según el Estándar	Contenido del Reporte de Incidente propuesto	Omitidos
a) Detalle del incidente i) Información temporal ii) Registrante iii) Contexto iv) Descripción del incidente v) Evaluación de la severidad vi) Evaluación de la prioridad vii) Riesgo viii) Estado del incidente	a) Detalle del incidente: i) Información temporal ii) Responsable del registro iii) Ítem de Prueba iv) Caso de Prueba v) Descripción del incidente vi) Prioridad vii) Estado del incidente viii) Anexo	-"Evaluación de la severidad": se consideró suficiente la prioridad. -"Riesgo": la documentación sobre riesgos fue omitida en todas las plantillas.

Tabla 8: Adaptación del Reporte de Finalización con respecto al Estándar IEEE 29119

Contenido del Reporte de Finalización según el Estándar	Contenido del Reporte de Finalización propuesto	Omitidos
a) Pruebas realizadas: i) Resumen ii) Desviaciones de las pruebas planificadas iii) Factores que bloquearon el progreso iv) Métricas v) Riesgos residuales vi) Entregables de prueba vii) Activos reutilizables viii) Lecciones aprendidas	a) Pruebas ejecutadas b) Incidencias c) Métricas d) Evaluación del Proceso de Pruebas: i) Apego al Plan de Pruebas ii) Factores que bloquearon el proceso iii) Evaluación de la completitud	-"Riesgos residuales": se acordó eliminar todo lo relativo a riesgos. -"Entregables de prueba": siempre son los mismos entregables. -"Activos reutilizables": se eliminó para simplificar el documento. -"Lecciones aprendidas": se acordó incluir posteriormente.

4.2.3 Solución Propuesta al Problema #3

En el CI “no se generan reportes estadísticos sobre el proceso de pruebas”, la solución propuesta a este problema fue: proveer a la plataforma computacional la capacidad de generar automáticamente métricas, específicamente de las pruebas funcionales. Estas métricas se calcularán a partir de la información registrada en los formularios y se incluirán en el documento Reporte de Finalización. Específicamente, se le planteó a la UCMC el cómputo de las siguientes métricas:

- Cobertura de Pruebas: esta métrica divide la cantidad de casos de prueba diseñados, entre la cantidad de casos de prueba mínimos necesarios para cubrir las funcionalidades requeridas. Un valor de cobertura alto (cercano al 1), indica un buen desarrollo de las pruebas.
- Madurez de las Pruebas: este indicador se enfoca en los resultados obtenidos en las pruebas, de manera que divide el número de casos de prueba cuyo resultado fue el esperado, entre el total de casos definidos. Mientras más cercano a 0 sea el valor de la madurez, más inmaduro es el estado del software.
- Densidad de Defectos: esta métrica ofrece una proporción de defectos con respecto al total de funcionalidades probadas. Es un dato que puede utilizarse al concluir las pruebas para valorar la completitud de las pruebas.
- Perfiles de Fallos: son valores que permiten categorizar y priorizar las no conformidades identificadas. El indicador divide el número de defectos de cierto tipo entre el total de errores.

Este grupo de métricas se conformó, tras consultar el registro de calidad “Reporte de Métricas”, perteneciente al conjunto de documentos elaborados por el proyecto de investigación B2723 Creación del Laboratorio de Aseguramiento de Calidad del Software (LACSOFT). De él se decidió extraer aquellas métricas que no involucraban código fuente (por ejemplo la densidad de falla en miles de líneas de código) o análisis de tiempos (como el cálculo de la productividad de las pruebas).

Se creyó recomendable iniciar con la recolección de datos basados en los requerimientos especificados y en los resultados de las pruebas. En primer lugar porque la información recabada en la plataforma permite su automatización y en segundo lugar porque una de las principales preocupaciones en el CI es la insatisfacción de los requerimientos del usuario.

4.2.4 Solución Propuesta al Problema #4

En el CI “El registro de los criterios de aceptación es bastante irregular”, la solución propuesta a este problema fue: la utilización de una plantilla para especificar los requerimientos funcionales, inspirada en el estándar IEEE 830 “Práctica Recomendada para la Especificación de Requerimientos” (Anexo N).

Sin embargo, por la familiaridad que los analistas poseen con los formatos utilizados actualmente para registrar las funcionalidades solicitadas por el usuario, a saber: Casos de Uso e Historias de Usuario; la plantilla fue rechazada y debió proponerse otra solución.

Dada la incursión del Centro de Informática en la metodología de desarrollo SCRUM, se pensó en mejorar el documento que utilizan para la definición de las Historias de Usuario. Éste es bastante extenso y para el equipo de calidad resulta complicado extraer de este los requerimientos funcionales y sus criterios de aceptación. Actualmente éstos vienen dispersos en varias secciones (y no se asocian a ningún requerimiento puntualmente.

Se le propuso a la UCMC una nueva plantilla (Anexo Ñ), más pequeña y simple, en la que se requiere del analista específicamente: el Enunciado de la Historia de la forma rol + funcionalidad + razón (por ejemplo: Como usuario administrador, necesito ver un listado de los talleres disponibles, con la finalidad de realizar una búsqueda alfabéticamente) y su(s) criterio(s) de aceptación de la forma contexto + evento + resultado esperado (En caso de que no haya ningún taller disponible, cuando se deba desplegar el listado, se mostrará el mensaje "No hay talleres"). Esta solución no fue aceptada formalmente, por lo que se utilización en el proceso de pruebas es opcional, aunque se recomienda para agilizar el diseño de las pruebas.

4.2.5 Solución Propuesta al Problema #5

En el CI “el seguimiento y control de los proyectos es dificultoso”, la solución propuesta a este problema fue: utilizar una suite BPM de código abierto para la implementación de los nuevos procesos diseñados y aprovechar las funcionalidades que provee para el seguimiento y control.

Algunas de las funcionalidades de las suites BPM con las que se facilita el seguimiento de los procesos son:

- Acceso al "mapa del proceso": se dispone de una herramienta gráfica para observar aquellas tareas que ya se ejecutaron, cuáles se omitieron, cuál se está ejecutando y cuáles se encuentran pendientes.
- Gestión del cronograma: las herramientas BPM permiten establecer el tiempo máximo que debe dedicarse a cada tarea en los procesos. Estos tiempos (en días, horas y minutos) pueden incluso definirse de manera específica para cada caso, es decir, en un proyecto de grandes dimensiones es posible definir tiempos mayores que los requeridos por un proyecto más pequeño. Además, queda registro de las tareas que excedieron el tiempo definido y pueden generarse alertas al respecto.
- Acceso a formularios y documentos sin necesidad de participación: es posible definir un rol de "Administrador de Proceso", con el cual se pueden revisar los formularios llenados, los reportes generados y los documentos adjuntos, sin necesidad de haber tenido participación directa en las tareas. Con ello se apoya el monitoreo de los procesos.
- Automatización de tareas: las suites BPM permiten además de la asignación automática de tareas (de forma cíclica, por carga de trabajo, de acuerdo a los roles), se puede mecanizar la generación y el llenado de las plantillas de documentación. Con ello se puede asegurar que se construyen todos los documentos y si estos no son llenados correctamente, el responsable de la tarea queda en evidencia.

A continuación, en la tabla 9 se resumen las soluciones propuestas a los problemas seleccionados.

Tabla 9: Resumen de soluciones propuestas

Problema Seleccionado	Solución Propuesta
No hay automatización en las pruebas de software	-Utilización de una herramienta del tipo *record and playback* (grabar y reproducir) para la automatización de pruebas funcionales y de regresión. -Automatizar ciertas tareas de la documentación del proceso de pruebas, por ejemplo: llenado automático de campos, cálculo de valores y generación de reportes.
El proceso de pruebas no se ejecuta sistemáticamente igual	-Sistematizar un nuevo proceso de pruebas, en una plataforma computacional que asegure se ejecute siempre de la misma manera. El nuevo proceso de pruebas y su documentación se basan en las recomendaciones de los estándares IEEE 1012 e ISO/IEC/IEEE 29119. -El proceso integra las tareas para: Registro del Proyecto, Verificación de Requisitos, Pruebas Exploratorias, Pruebas Funcionales y Pruebas de Aceptación. -La documentación está conformada por: Registro del Proyecto de Pruebas, Evaluación de Caso de Uso, Evaluación de Historia de Usuario, Registro de Historia de Usuario, Resultado de las Pruebas Exploratorias, Plan de Pruebas, Diseño de Pruebas, Reporte de Incidente, Reporte de Finalización
No se generan reportes estadísticos sobre el proceso de pruebas	-Recolección automática de métricas y cómputo de: Cobertura de pruebas, Madurez de las pruebas, Densidad de defectos, Perfiles de fallos. -Construcción automática del Reporte de Finalización.
El registro de los criterios de aceptación es bastante irregular	-Utilización de una plantilla inspirada en el Estándar IEEE 830. -Utilización de una nueva plantilla para las Historias de Usuario.
El seguimiento y control de los proyectos es dificultoso	-Utilización de una suite BPM de código abierto para la implementación del nuevo proceso de pruebas y aprovechar las funcionalidades que provee para el seguimiento y control de procesos.

4.3 OBJETIVO ESPECÍFICO: IDENTIFICAR LAS HERRAMIENTAS DE SOFTWARE APROPIADAS PARA CONSTRUIR UN PROTOTIPO DE PLATAFORMA QUE LE PERMITA AL CI SOLUCIONAR LOS PROBLEMAS ESCOGIDOS

Teniendo en mira las soluciones propuestas, se procedió a identificar y seleccionar las herramientas de software adecuadas para su implementación. Para ello se consultó un listado de herramientas elaborado durante el proyecto de investigación B2723 Creación del Laboratorio de Aseguramiento de Calidad del Software (LACSOFT) y con el que se pretendía definir el portafolio de herramientas del laboratorio.

Este listado contiene más de 300 herramientas, organizadas según su naturaleza (código abierto, gratuita, privativa) y su principal funcionalidad (pruebas funcionales, pruebas de regresión, pruebas de integración, pruebas de carga, monitoreo de recursos y rendimiento, pruebas de aceptación, pruebas de interfaz, seguimiento de errores, manejo de versiones, pruebas de portabilidad, monitoreo de bases de datos, modelado de procesos, entre muchas más).

Tras tomar en cuenta las posibilidades de configuración e integración ofrecidas, experiencias anteriores en su utilización y algunos reportes técnicos sobre sus fortalezas y debilidades, se seleccionaron las siguientes herramientas para conformar el prototipo de plataforma computacional:

4.3.1 Processmaker

Se seleccionó Processmaker - *Open Source Workflow Software* y *BPM Software*, para la administración general de todo el proceso de pruebas. La suite de BPM contendría los procesos diseñados y las plantillas para la documentación de los mismos. Se decidió así porque dentro de sus principales funcionalidades están: la posibilidad de modelar los flujos de trabajo desde una interfaz sencilla e intuitiva, el diseño de formularios web y su asignación a tareas específicas del proceso y el seguimiento de los procesos a través de notificaciones automatizadas e indicadores clave de desempeño. Todas estas características son indispensables para implementar las soluciones propuestas.

Processmaker fue desarrollado por la empresa Colosa y se ofrece primordialmente como herramienta ASP (*Application Service Provider*) donde la misma compañía dispone sus servidores por un alquiler. Sin embargo, también ofrece una versión de subscripción "local" y la versión *open source*, esta última recomendada para desarrolladores y pequeñas empresas.

Durante el proceso de selección de la herramienta, se compararon sus especificaciones con otras suites BPM, particularmente con Bizagi y con Bonita Open Solution. No obstante se terminó por elegir Processmaker primeramente por la experiencia positiva en su utilización y posteriormente porque la versión de código abierto de Bonita posee algunas limitaciones y porque Bizagi posee separados el modelado de los procesos, de su transformación en flujos de trabajo ejecutables (*Modeler-Studio-Engine*), y no se quería añadir más cambios de contexto al prototipo de plataforma.

4.3.2 Redmine

Se seleccionó Redmine, una herramienta para la gestión de proyectos, específicamente para la gestión de las incidencias. De manera que, independiente del flujo principal de actividades coordinado por Processmaker, el registro de las no conformidades identificadas durante la ejecución de los casos de prueba se realiza en Redmine.

Redmine es una aplicación desarrollada con el *framework Ruby on Rails*, dedicada a la administración de proyectos y al seguimiento de errores. Es gratuita, puede descargarse desde el sitio oficial www.redmine.org y se distribuye bajo la licencia GNU *General Public License*. La principal característica que convirtió a *Redmine* en una herramienta atractiva para la presente investigación es su plasticidad. La aplicación si bien es cierto fue pensada para el ámbito empresarial, la facilidad con la que puede configurarse le permite al usuario utilizarla en la administración de sus propios proyectos, independientemente del área en la que se desarrollen. *Redmine* permite configurar de manera específica cada tipo de proyecto que se desee gestionar. Es posible activar o desactivar módulos para, agregar campos personalizados (de variedad de formatos: texto, lista, numérico, fecha) que le confieren una identidad propia a cada proyecto, e incluso, un mismo usuario puede ocupar un rol distinto en cada proyecto. Esta fue una de las características más atractivas, para seleccionarla como herramienta parte de la plataforma.

4.3.3 Selenium

Selenium es en sí un conjunto de herramientas, cada una desarrollada con propósitos específicos, pero todas enfocadas en la automatización de pruebas de software, tanto funcionales como de regresión y de aceptación. La más popular de ellas, Selenium IDE, permite grabar y reproducir casos de prueba, construyendo de forma automática los *scripts* de prueba. Por su parte Selenium WebDriver provee la capacidad de crear pruebas de regresión para distintos navegadores web y distribuir la ejecución de los *script* a través de varios ambientes. Por ello fue elegida para la automatización de las pruebas funcionales y de regresión en el Centro de Informática.

Específicamente Selenium IDE (por las siglas en inglés de "ambiente de desarrollo integrado") es una herramienta utilizada para la generación automática de casos de prueba en aplicaciones Web. Se integra al navegador y graba la interacción que tiene el usuario con el sitio, en *scripts* con formato HTML. Sin embargo, como su nombre lo deja entrever, los casos de prueba no sólo se generan automáticamente sino que es posible, editarlos, parametrizarlos, añadir nuevos, depurarlos y repetir tantas veces como sea necesario.

Es una de las herramientas preferidas como el paso inicial dentro de la experiencia de pruebas funcionales (una de las razones por las que se tomó en cuenta para el CI), pues aunque el ejecutante no sea un programador experimentado, puede generar casos de prueba importantes en cuestión de segundos y posteriormente, conforme se alcanza familiaridad con los comandos, elaborar casos de prueba más complejos. Esta es una de las características que la hicieron sobresalir entre el conjunto de herramientas para la generación automática de *scripts*.

Selenium IDE posibilita crear *Test Suites*, conjuntos de pruebas, permitiendo tener para un mismo sistema, varias pruebas y ejecutarlas todas automáticamente. Al finalizar la prueba, puede accederse a una bitácora en la que se reporta la ejecución de cada comando y si ocurrió o no un error.

4.3.4 Git

La herramienta seleccionada para funcionar como repositorio versionado de la plataforma y para la gestión del código fuente de los *scripts* de prueba fue Git. Ésta herramienta para el control de versiones, posee una naturaleza distribuida (en oposición a los otros sistemas centralizados como Subversion), de manera que todos los clientes al realizar una sincronización con el servidor principal, no descargan únicamente la última versión (HEAD) de los archivos, sino que se replica completamente el repositorio. Esta característica le confiere mayor robustez, pues si el repositorio central se corrompe, cualquiera de los clientes puede copiarse para restaurarlo.

Los SCM suelen almacenar el conjunto de archivos y las modificaciones hechas sobre cada uno de ellos a lo largo del tiempo. Git por su parte, modela los datos como un conjunto de instantáneas, de manera que cada vez que ocurre un *commit* (confirmación de las modificaciones), se guarda el estado de todos los archivos en ese momento y para mayor eficiencia, los archivos no modificados no son duplicados, sino que se genera una referencia al archivo anterior.

En Git la mayoría de las operaciones sólo necesitan de recursos locales para ejecutarse, con lo cual se evita sobrecarga de la red y le otorga mayor velocidad. Esto también le confiere independencia de la red y permite trabajar y versionar archivos en cualquier lugar, sin necesidad de conexión a Internet.

En un repositorio de Git, los archivos pueden encontrarse en tres estados: confirmado (*committed*) cuando ya se encuentran almacenados correctamente en la copia local, modificado (*modified*) si el archivo ha cambiado pero aún no se ha confirmado y preparado (*staged*) cuando el archivo modificado ha sido "marcado" para incluirlo en el próximo *commit*. Es así que el flujo de trabajo básico con Git es:

- Se modifica un conjunto de archivos en la copia local.
- Se añaden al área de preparación.
- Se confirman los cambios, lo cual toma los archivos tal y como están en el área de preparación y almacena las "instantáneas" en el directorio Git.

Por lo anterior y otras características ventajosas (como la posibilidad de integración con Redmine), se seleccionó Git como SCM del prototipo de plataforma computacional a implementar.

Una vez seleccionados los componentes, se procedió a modelar las vistas arquitecturales, funcional y de información del prototipo de plataforma. Ambas se detallan en las figuras 11 y 12.

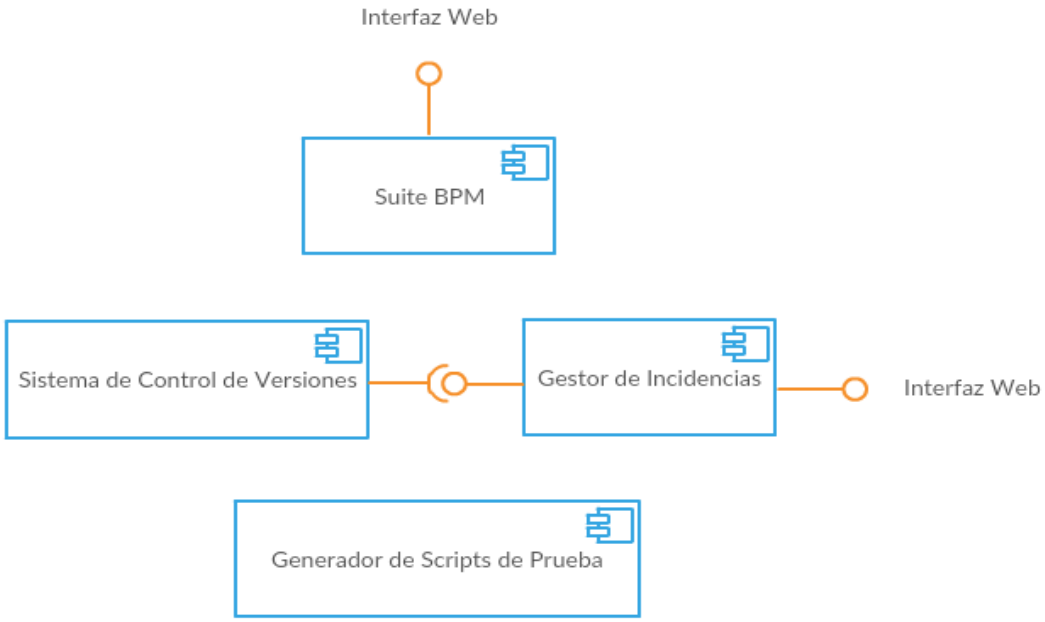


Figura 9: Vista Funcional del Prototipo de Plataforma propuesto

El prototipo de plataforma está integrado por: una suite BPM encargada de la gestión de los procesos, una herramienta para el seguimiento de errores, en la que se realizará la gestión de incidencias del proceso de pruebas funcionales, un sistema de control de código fuente (SCM) donde se almacenarán los scripts de pruebas en caso de ser generados y una herramienta para la generación automática de casos de prueba para aplicaciones web.

El acceso a la herramienta BPM se realiza a través de su interfaz web y lo mismo sucede con el gestor de incidencias. En cuanto al SCM, éste se encontrará imbuido en el gestor de incidencias y la intención es acceder a él por este medio, aunque también puede disponer de una interfaz web propia. Por último, el generador de scripts es una extensión del navegador web.

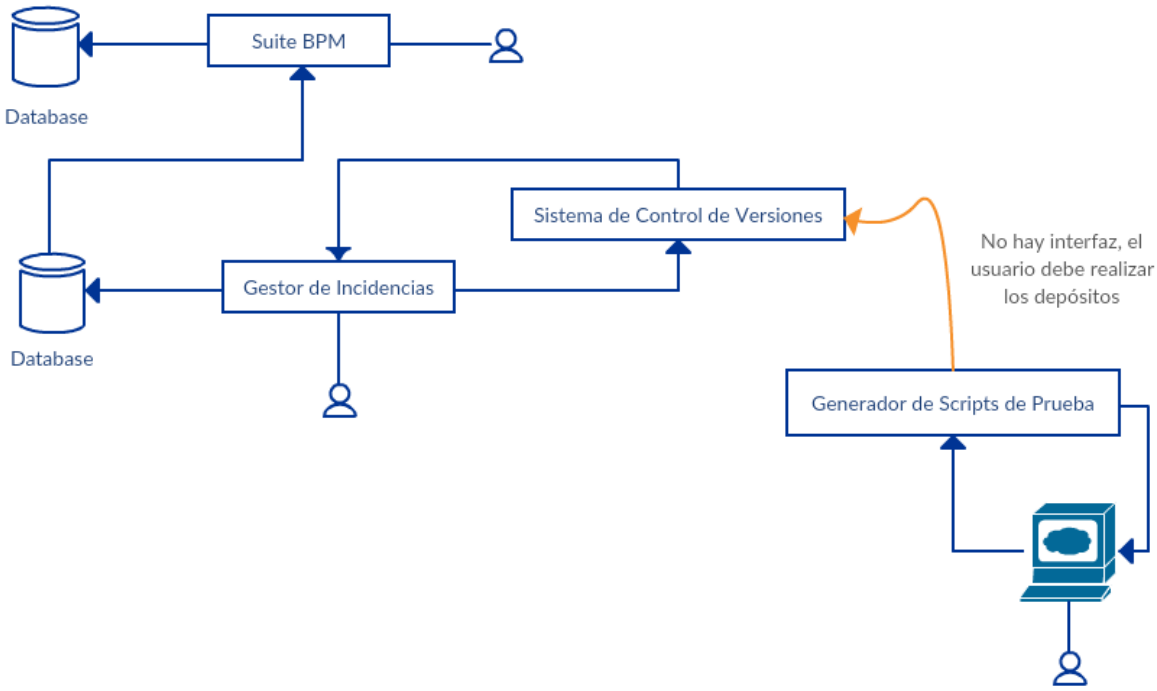


Figura 10: Vista de Información del Prototipo de Plataforma Propuesto

En lo que se refiere al tránsito de la información, la suite BPM la obtiene en primera instancia del usuario en su interacción durante el modelado y la ejecución de los procesos. Toda esta información es almacenada de manera persistente en una base de datos. En segunda instancia, recolecta información específica sobre las incidencias registradas en la herramienta para el seguimiento de errores.

El gestor de incidencias, participa en tres flujos de información: el que proviene desde el usuario, su interrelación con el sistema de control de versiones y el acopio en su base de datos.

Por su parte el SCM, se encuentra integrado al gestor de incidencias pero también puede recibir información "de fuera", por ejemplo a través de una interfaz web o por línea de comandos.

Finalmente, como se mencionó anteriormente, la herramienta para automatizar la creación de *scripts* es una extensión del navegador web y se comunica directamente con éste o con el usuario.

4.4 OBJETIVO ESPECÍFICO: DISEÑAR UN PROTOTIPO DE PLATAFORMA COMPUTACIONAL QUE IMPLEMENTE LAS SOLUCIONES ENCONTRADAS PARA LOS PROBLEMAS IDENTIFICADOS EN EL CI

Después de haber seleccionado las herramientas y definida la forma en que interaccionan y cómo la información circula entre ellas, se procedió a configurar cada herramienta de manera que funcionasen realmente como componentes de una plataforma y ya no como soluciones de software aisladas. La primera herramienta que se instaló y configuró fue Processmaker, esto por cuanto la suite BPM es la encargada de gestionar las actividades de V&V de software seleccionados. Seguidamente se instaló y personalizó Redmine, para el registro de las no conformidades. Con ambas herramientas preparadas, se programó la lógica necesaria para su integración, de tal forma que la información insertada en Redmine (las incidencias) fuera accesible desde Processmaker (en el Reporte de Finalización). Para finalizar, se implementó la integración entre Redmine y Git, para que todo proyecto de pruebas tuviera acceso a un repositorio versionado desde el cual acceder a los scripts, datos de prueba y demás documentación necesaria. A continuación el detalle de éstas y otras configuraciones llevadas a cabo. Para la correcta utilización de la plataforma, se elaboró un Manual de Usuario, adjunto como Anexo O.

4.4.1 Configuración de Processmaker

La suite de BPM fue seleccionada para sistematizar la ejecución del proceso diseñado y explicado en la sección 4.2.2.1 de este documento, a excepción de dos actividades: la ejecución de los casos de prueba y el registro de las incidencias. Así mismo, todas las plantillas propuestas para la documentación de los procesos especificadas en la sección de la sección 4.2.2.2 (a excepción del Registro de Incidente), cuentan con su equivalente digital en esta herramienta.

Los procesos y plantillas diseñados fueron traducidos a código ejecutable gracias a las grandes capacidades de personalización de Processmaker. Se configuraron roles, tareas y formularios de la siguiente manera:

4.4.1.1 Roles y Usuarios

Según los roles y responsabilidades definidos para el proceso de pruebas, Processmaker es utilizado por 3 tipos de usuario, por lo tanto se crearon los siguientes grupos (estos se observan en la figura 11):

- Administrador: encargado de la instalación, aquel con la capacidad de crear y modelar los procesos, crear formularios y definir los demás roles.
- Líder de Pruebas: encargado de la verificación de los requerimientos especificados en los casos de uso o historias de usuario. Es el encargado también, del registro de los resultados de las pruebas exploratorias, de la planificación de las pruebas, y de la elaboración del informe final.
- Encargado de Pruebas: es el responsable del diseño de las pruebas funcionales.

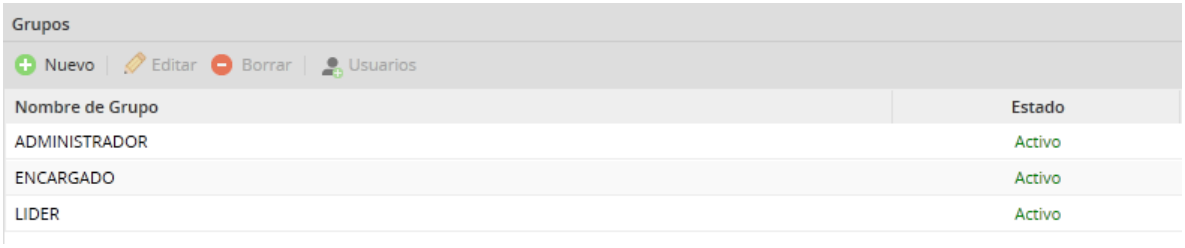


Figura 11: Grupos de usuario definidos en Processmaker

Debido a que el usuario Administrador tiene capacidades de "súper usuario", se le concedió el rol de *System Administrator*, mientras que los otros usuarios poseen el rol de *Operator* (figura 12).

Figura 12: Usuarios de Processmaker y su rol asociado

4.4.1.2 Modelado de Procesos

Para el apoyo de las tareas de verificación y validación de software seleccionadas, fue necesario implementar 6 procesos. De ellos 1 constituye el proceso general y los otros 5 sub-procesos, que se ejecutan de manera síncrona al primero. Los procesos son los siguientes (figura 13):

- Pruebas de Software U. Calidad
- Verificación de Requerimientos
- Pruebas Exploratorias
- Pruebas Funcionales
- Pruebas Dinámicas
- Pruebas de Aceptación

Nuevo | Editar | Estado | Borrar | Exportar | Importar | Delete Cases

Título del Proceso	Tipo	Categoría	Estado	Usuario Propietario
Pruebas Aceptación (BPMN Project)	bpmn	- Sin Categoría -	Activo	Administrator gemg (gemg)
Pruebas de Software U. Calidad (BPMN Project)	bpmn	- Sin Categoría -	Activo	Administrator gemg (gemg)
Pruebas Dinámicas (BPMN Project)	bpmn	- Sin Categoría -	Activo	Administrator gemg (gemg)
Pruebas Exploratorias (BPMN Project)	bpmn	- Sin Categoría -	Activo	Administrator gemg (gemg)
Pruebas Funcionales (BPMN Project)	bpmn	- Sin Categoría -	Activo	Administrator gemg (gemg)
Verificación de Requerimientos (BPMN Project)	bpmn	- Sin Categoría -	Activo	Administrator gemg (gemg)

Figura 13: Procesos diseñados en Processmaker

El "macro-proceso" Pruebas de Software U. Calidad (que se ilustra en la figura 14), se encuentra integrado por una única actividad, Registro del Proyecto, y 4 sub-procesos. Estos al concluir, desembocan en una estructura de control (OR exclusivo) y en ella se determina si se prosigue o no, con el siguiente sub-proceso. Por ejemplo, si los resultados de las Pruebas Exploratorias no son satisfactorios, entonces no se continúa con las Pruebas Funcionales y se retorna a la actividad inicial del sub-proceso Pruebas Exploratorias.

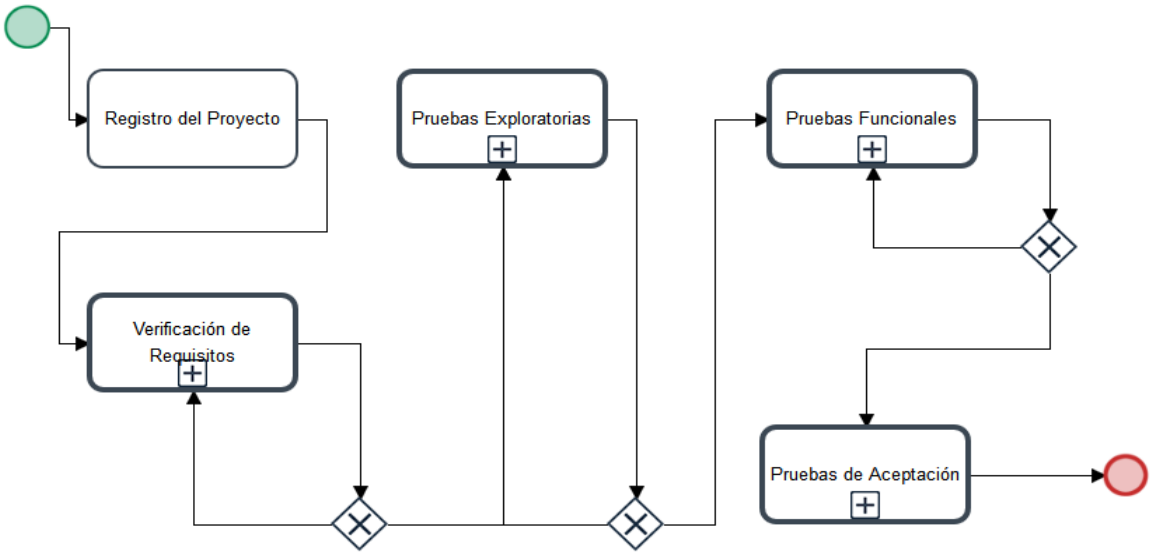


Figura 14: Macro-proceso "Pruebas de Software U. Calidad"

El sub-proceso Verificación de Requisitos (ilustrado en la figura 15), se encuentra conformado por 2 tareas. La primera es la Evaluación de los Requisitos de Software, en ella se decide si se evaluará un Caso de Uso o una Historia de Usuario y posteriormente se procede a completar el formulario digital para la respectiva evaluación (Anexo F o Anexo G). Al concluir dicho formulario, se ofrece la posibilidad de registrar los requerimientos y sus criterios de aceptación. Si la respuesta es afirmativa, se procede con la tarea Registro de Funcionalidades y Criterios de Aceptación, si la respuesta es negativa, el sub-proceso concluye y se retorna al proceso general.

Si la evaluación del Caso de Uso o de la Historia de Usuario fue insatisfactoria y el documento debe ser devuelto al analista encargado, Processmaker exporta la evaluación efectuada en formato PDF o DOCX para adjuntarla al documento rechazado.

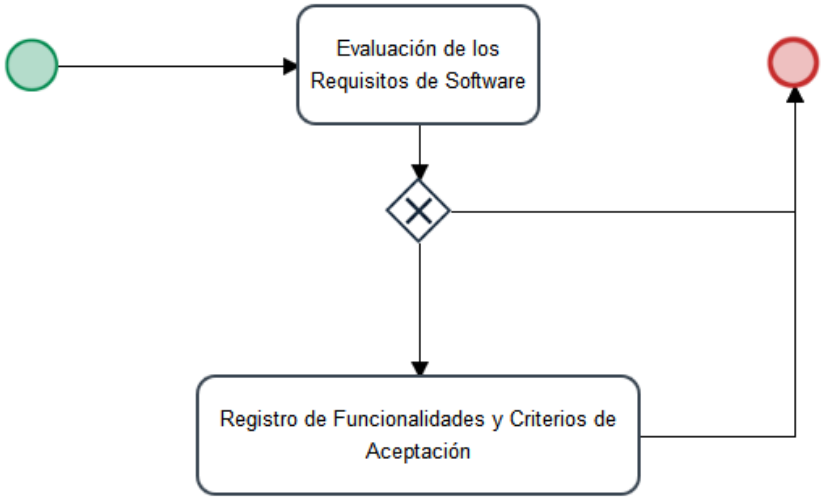


Figura 15: Sub-proceso "Verificación de Requisitos"

El siguiente sub-proceso a ejecutarse es Pruebas Exploratorias (obsérvese en la figura 16). Éste se encuentra compuesto de sólo una tarea, Registro de Resultados de las Pruebas Exploratorias. La tarea posee asignada un formulario equivalente al Anexo H, para registrar las funcionalidades ejercitadas por el usuario Líder y los resultados obtenidos. Al finalizar el formulario se le consulta al usuario si se procede o no con las Pruebas Funcionales. Si la respuesta es afirmativa el sub-proceso concluye y se vuelve al proceso general para la ejecución del siguiente sub-proceso, si se respondió de forma negativa, se exporta el reporte de evaluación en formato PDF o DOCX y se regresa a la tarea inicial del sub-proceso Pruebas Exploratorias.

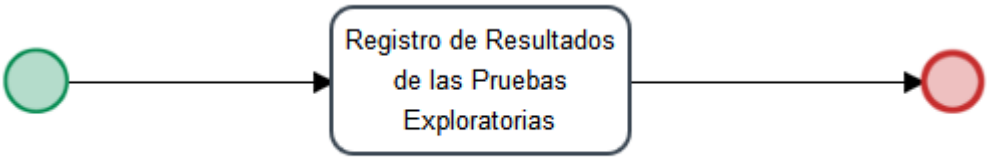


Figura 16: Sub-proceso "Pruebas Exploratorias"

A continuación se ejecuta el sub-proceso Pruebas Funcionales, el cual además de las tareas Planificación y Finalización, contiene dentro de sí al sub-proceso Pruebas Dinámicas (puede observarse en la figura 17). Esta organización se hizo inspirada en el modelo de 3 capas del estándar ISO/IEC/IEEE 29119 Pruebas de Software. En la tarea de Planificación el usuario líder completa en formato electrónico, el Plan de Pruebas (tal y cómo se incluye en el Anexo J). Seguidamente, inicia la ejecución del sub-proceso Pruebas Dinámicas, con la tarea Diseño de Pruebas, como puede verse en la figura 18. Cuando Pruebas Dinámicas concluye, se prosigue con la tarea Finalización, en la que se genera automáticamente el Reporte de Finalización. En éste, se dispone de un espacio para decidir si las pruebas cumplen con los criterios de completitud o no. Si se decide que sí, el sub-proceso concluye y se procede al siguiente. Si se decidió que las pruebas no eran completas, se regresa a la tarea Planificación. En ambos casos, Processmaker exporta el Reporte de Finalización en PDF o DOCX.

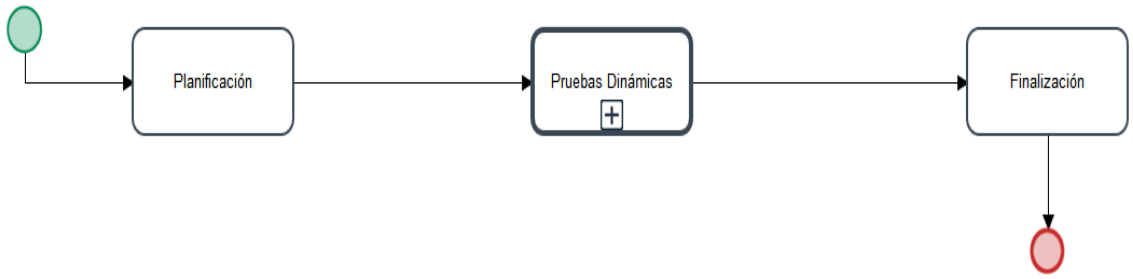


Figura 17: Sub-proceso "Pruebas Funcionales"

El sub-proceso de Pruebas Dinámicas incluye 2 tareas, Diseño de Pruebas y Ejecución de Pruebas y Registro de Incidencias. Originalmente contenía la tarea Preparación del Ambiente de Pruebas, pero se eliminó la transición a dicha tarea, al suprimirse el rol de Soporte Técnico.

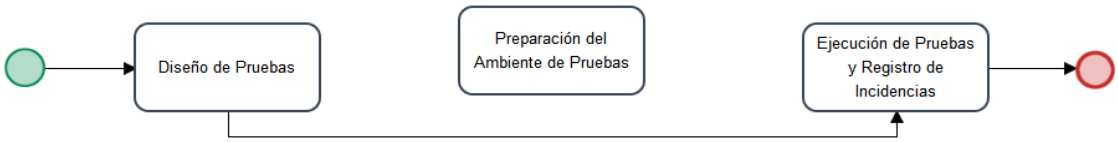


Figura 18: Sub-proceso Pruebas Dinámicas

En la tarea Diseño de Pruebas, el usuario con el rol de Encargado, completa el formulario equivalente a la plantilla que se incluye en el Anexo K. En esta plantilla se incluyen los casos de prueba y se determina si el ambiente de pruebas está preparado o no. La tarea Ejecución de Pruebas y Registro de Incidencias, exporta los casos de prueba diseñados en la tarea anterior (en formato PDF o DOCX) y también las instrucciones necesarias para la configuración de Redmine, herramienta en la que se realmente se registran las incidencias.

Si el sub-proceso de Pruebas Funcionales concluye satisfactoriamente, se procede a la ejecución del último sub-proceso, Pruebas de Aceptación. Como se ilustra en la figura 19, éste se compone de una única tarea, Registro de Errores por parte del Usuario. Esta tarea tiene la particularidad de no requerir autenticación en Processmaker para su acceso, sino que se puede acceder anónimamente a través de un URL específico. La idea es que llegada a esta tarea, el usuario Líder les envía el URL a los usuarios que ejecutarán las pruebas de aceptación y accederán al formulario digital equivalente al Anexo I. La información que los usuarios registren, llegará a la tarea "Registro de Errores por parte del Usuario" y allí podrá ser accedida por el usuario Líder.

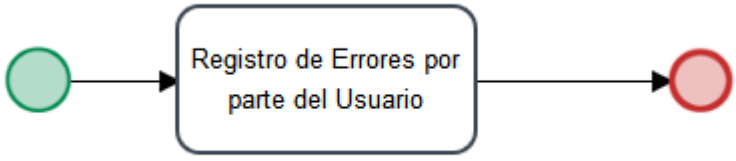


Figura 19: Sub-proceso Pruebas de Aceptación

La ejecución síncrona, en otras palabras, la sensación de participar de un único flujo de tareas, se logra mediante la definición de "variables compartidas". Esta funcionalidad, fue una grata sorpresa y facilitó en gran manera la automatización de más comportamientos de los pensados. Con ellas, la información puede transitar a lo largo de todos los sub-procesos e incluso dirigir, en las estructuras de control, si el sub-proceso debe ejecutarse más de una vez. Así por ejemplo, es posible que la información ingresada en el Registro de Proyecto (tarea perteneciente al macro-proceso) se comparta a todos los sub-procesos o los criterios de aceptación registrados durante la Verificación de Requisitos, estén disponibles durante el Diseño y la Finalización.

En la figura 20 se muestran algunas variables compartidas

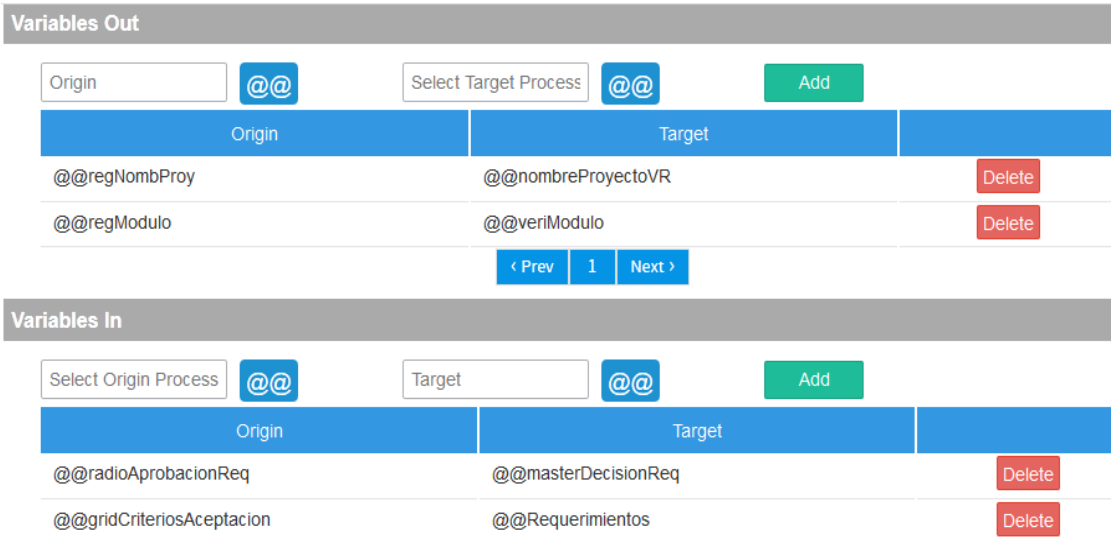


Figura 20: Variables compartidas entre el proceso general y el sub-proceso Verificación de Requisitos

Otra configuración necesaria durante el modelado, fue la definición de permisos (*Permissions)* para cada proceso. Para facilitar el seguimiento de todas las actividades, se creó un permiso para que el Líder de Pruebas tenga acceso a: el mapa del proceso (diagrama que muestra las tareas realizadas, las que se están ejecutando, las omitidas y las pendientes), a los formularios y a los documentos generados, en cualquier momento.

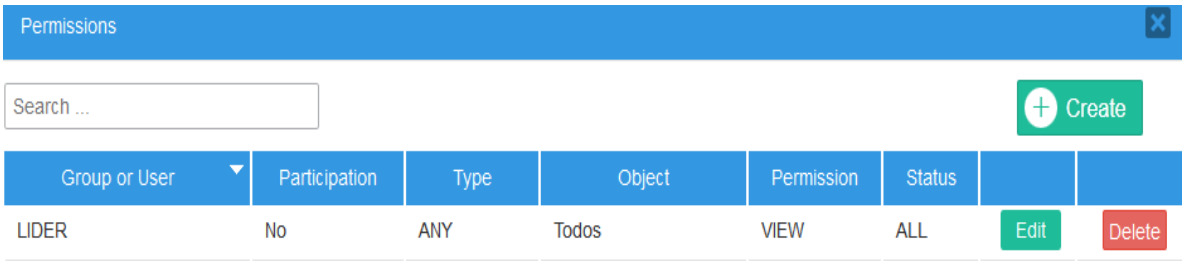


Figura 21: Permisos de proceso para el grupo Líder

4.4.1.3 Formularios dinámicos

Processmaker permite la creación de formularios dinámicos (*Dynaforms*) y su asignación a tareas de proceso, a través de lo que se conocen como "pasos" (*Steps*). Estos formularios son construidos con el simple arrastre (*drag and drop)* de controles (*Web Controls*) como: caja de texto, lista desplegable, cuadro de chequeo, botón de radio, fecha, archivo adjunto, grilla (*grid*), botón, imagen, entre otros. A cada control es posible asociarle una variable, ya sea que se haya definido anteriormente y forme parte del proceso, o sea del todo nueva.

La configuración más importante, luego de construir los formularios equivalentes a las plantillas definidas para la documentación de los procesos de V&V de software en el CI, fue dotarlas de "automatización". Mediante la programación de disparadores (*triggers)* en PHP y sub-rutinas en Javascript, se dotó a los formularios de campos de llenado automático y cómputo de variables derivadas.

En la figura 22 se ejemplifica tanto la asignación de un formulario dinámico a una tarea, como la programación de disparadores. Específicamente se ilustra cómo la tarea Finalización posee asignado el *dynaform* llamado "Reporte" y cómo antes de desplegar el formulario, se deben de ejecutar dos *triggers*, "traeNC" y "llenaCasosEjecutados".

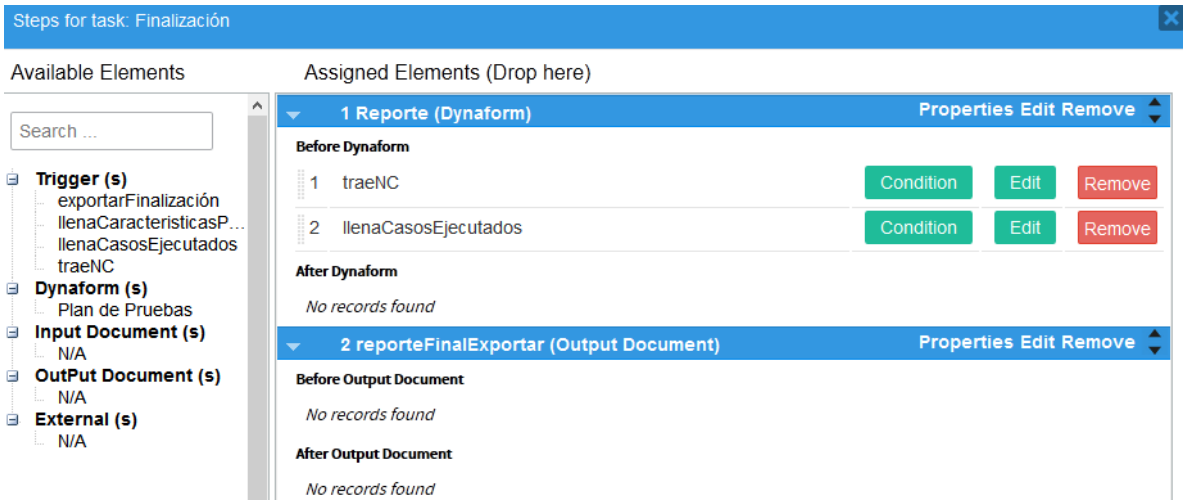


Figura 22: Formulario y disparadores asignados a la tarea Finalización

Los disparadores son fragmentos de código que se ejecutan mecánicamente cuando se cumple una condición. En Processmaker puede programarse su ejecución antes del despliegue de un formulario o después de haberlo completado. Fue así que se logró que varios de los controles de los formularios, se encuentren "listos", "llenos", al instante de abrirlos. En el extracto de código #1, se muestra cómo se obtienen el nombre y el apellido del usuario que ha iniciado sesión en la plataforma, así como la fecha y la hora.

Código 1: Obtención de datos de Usuario, la fecha y la hora del sistema:

```
1 $arrayUser = userInfo(@@USER_LOGGED);
2 @@regLiderPruebas = $arrayUser['firstname'] . ' ' . $arrayUser['lastname'] ;
3
4 @@regFechaInicio = getCurrentDate() . ' ' . getCurrentTime();
```

La programación de comportamientos automáticos, ya no antes o después de mostrar el formulario, sino en plena utilización del mismo, es posible a través de la invocación a funciones Javascript. Existen rutinas predefinidas y asociadas a cada *Web Control,* pero también pueden programarse nuevas. En los extractos de código #2 y #3 se muestra como se añaden filas a las grillas y cómo se calcula la cobertura de pruebas, respectivamente.

Código 2: Utilización de métodos Javascript predefinidos

```
29
30 var oGrid = $('#gridPlanProbar');
31
32 oGrid.addRow();
33 oGrid.addRow();
34
35 var wGrid = $('#gridPlanDatosPrueba');
36
37 wGrid.addRow();
38 wGrid.addRow();
39
40 var xGrid = $('#gridPlanHardware');
41
42 xGrid.addRow();
43 xGrid.addRow();
```

Código 3: Programación de sub-rutina Javascript para el cálculo de la cobertura de pruebas

```
$("#casosPEjectutados").setOnchange(function(newVal, oldVal) {
  if ($("#casosRequeridos").getValue() > 0)
        $("#cobertura").setValue(newVal/$("#casosRequeridos").getValue());
    else
        $("#cobertura").setValue(0);
});
```

4.4.1.4 Exportación de documentos

Processmaker ofrece la posibilidad de generar *output-documents* o "documentos de salida". Estos son archivos .doc o .pdf generados durante la ejecución del proceso, destinados a ser impresos o a compartirse digitalmente fuera de Processmaker. Se consideraron útiles para la creación de reportes en aquellas tareas donde se realiza algún tipo de evaluación o toma de decisión para continuar (por ejemplo la evaluación de casos de uso e historias de usuario y la determinación sobre la estabilidad del software a través de pruebas exploratorias). La construcción de estos documentos se realiza mediante plantillas HTML a las que se incluyen referencias a variables del sistema y del proceso.

En la figura 23, se observa la construcción del documento de salida utilizado para exportar los resultados de las pruebas exploratorias.

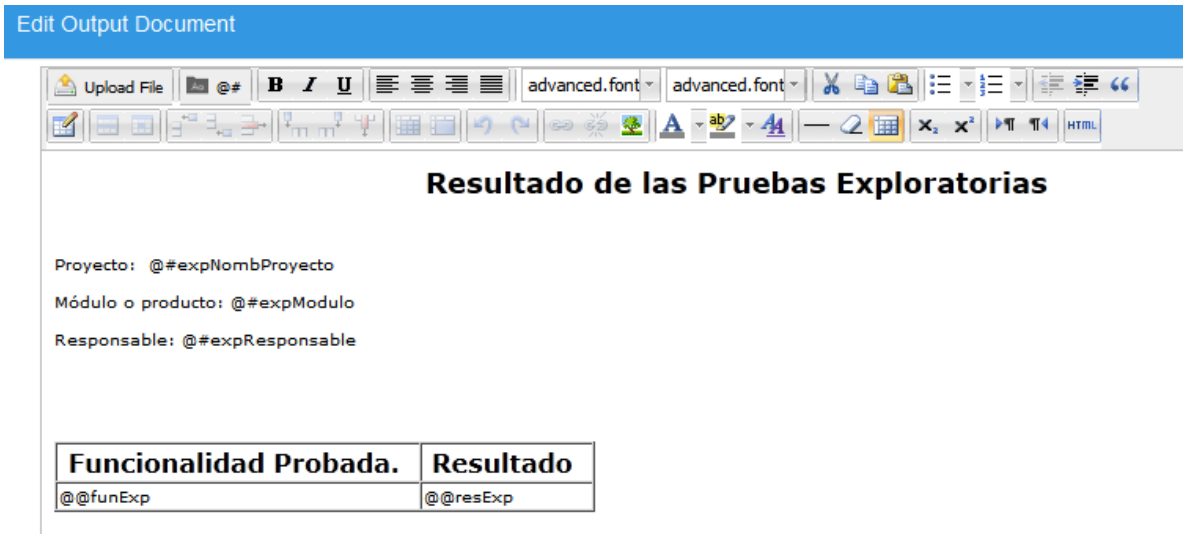


Figura 23: Documento de salida utilizado para exportar el resultado de las pruebas exploratorias

4.4.2 Configuración de Redmine

Redmine es una herramienta cuyo uso principal es el seguimiento de errores (conocido también como *bug tracking*), sin embargo gracias a las posibilidades de configuración y extensión de sus características, puede utilizarse para administrar variedad de proyectos. En la presente investigación, se seleccionó para la gestión de las incidencias, es decir, para el registro de los errores encontrados durante la ejecución de las pruebas funcionales de software. Las configuraciones realizadas para ofrecer esta funcionalidad fueron las siguientes:

4.4.2.1 Roles y Usuarios

En el diseño de la plataforma, se tomó la decisión de que Redmine sea utilizado únicamente por los probadores o *testers*, por lo tanto, los perfiles definidos son (figura 24):

- Gestor de Calidad: súper usuario, capaz de realizar todas las acciones sobre los proyectos, foros, calendario, documentos, peticiones y repositorio.
- Probador: únicamente puede crear peticiones y realizar operaciones de escritura sobre el repositorio.

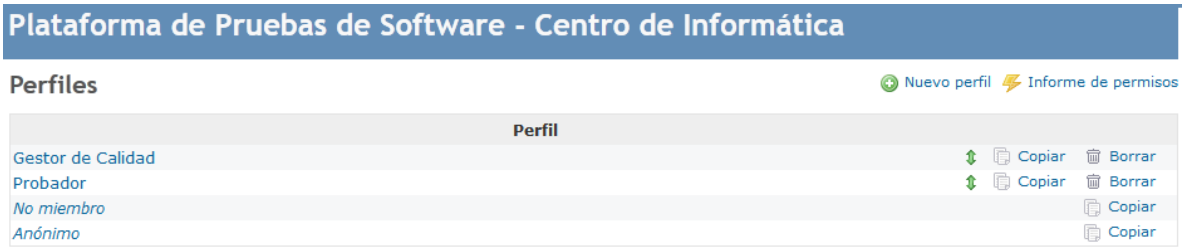


Figura 24: Perfiles definidos en Redmine

4.4.2.2 Peticiones y Campos Personalizados

Redmine ofrece la posibilidad de crear nuevos campos personalizados para la caracterización de algunos de los objetos disponibles en la herramienta (proyecto, usuario, documento, entre otros). Siendo Redmine el entorno elegido para la gestión de incidencias, se decidió utilizar las peticiones (*issues*) como equivalente del documento Reporte de Incidente y por ello debieron agregársele nuevos descriptores, para su coincidencia con la plantilla propuesta.

Lo primero que se creó fue nuevo tipo de petición llamado “No Conformidad”. Seguidamente se seleccionó del listado de descriptores por defecto, aquellos útiles para detallar al incidente: asunto, descripción, fecha de inicio y prioridad. Y finalmente se procedió a crear los siguientes campos personalizados:

- Caso de Prueba: de tipo texto. Corresponde al identificador del CU que generó la no conformidad.
- Iteración: de tipo numérico. Junto con el nombre del proyecto permite la correspondencia unívoca con el proyecto de pruebas registrado en Processmaker.
- Tipo de error: de tipo lista. Especifica el tipo de incidente reportado, por ejemplo “validación”, “funcionalidad”, “no correspondencia con lo documentado”, “usabilidad”.

En la figura 25 se observa el formulario destinado en Redmine para el registro de las incidencias. Se construyó según la plantilla del documento Reporte de Incidente y hace uso de todos los campos personalizados incluidos en la herramienta.

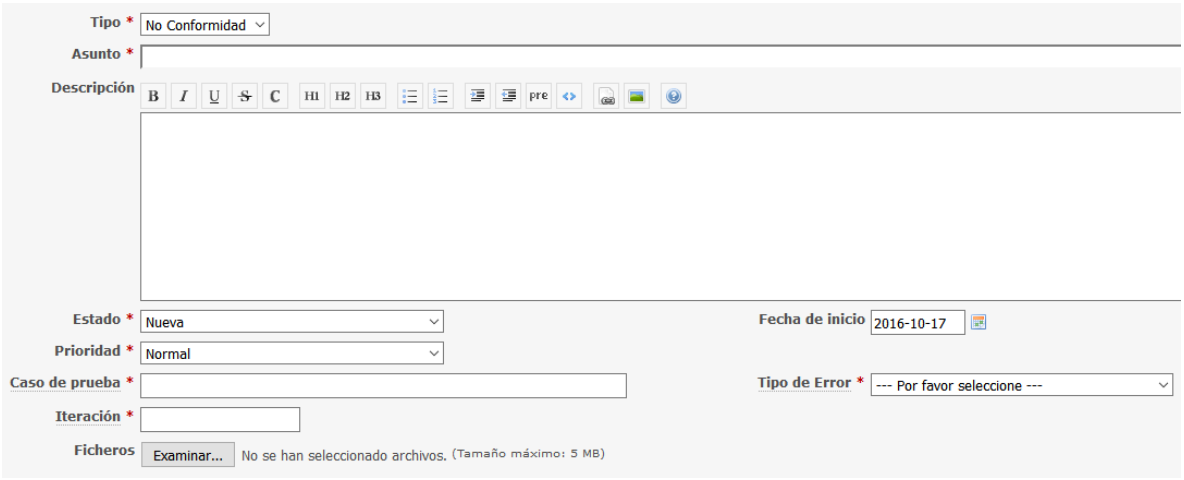


Figura 25: Reporte de Incidente en Redmine

4.4.3 Integración Processmaker-Redmine

Durante la definición de la arquitectura de la plataforma y como pudo observarse en la vista de información del prototipo (figura 10), los datos almacenados en Redmine relativos a las incidencias debían ser accesibles desde Processmaker para la construcción del Reporte de Finalización.

Se decidió aprovechar la capacidad de Processmaker para conectar procesos a bases de datos externas, además de la propia, utilizada para el almacenamiento de sus variables. Específicamente, se agregó la base de datos de Redmine al proceso "Pruebas Funcionales", a través del formulario que se observa en la figura 26.

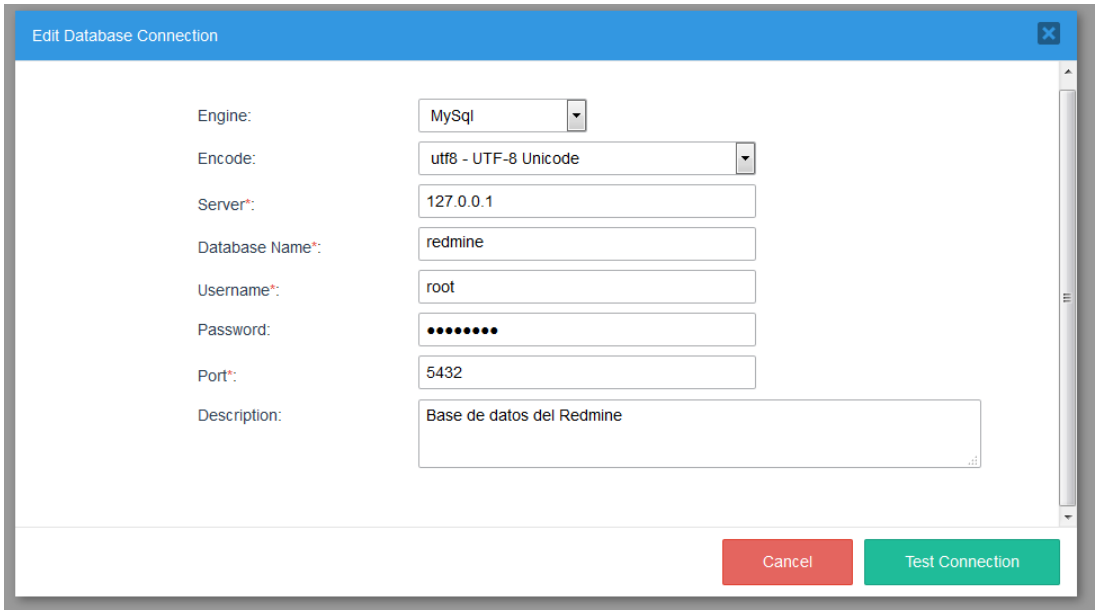


Figura 26: Inclusión de la base de datos de Redmine en Processmaker

Ahora, dado que los controles de los formularios dinámicos también pueden ligarse a una base de datos y obtener su información mediante consultas SQL, se resolvió presentar la información obtenida desde Redmine en un *grid*. Para ello fue necesario enlazar cada uno de los campos de la grilla a los datos solicitados en la consulta a la base de datos de Redmine. Esto se muestra en la figura 27.

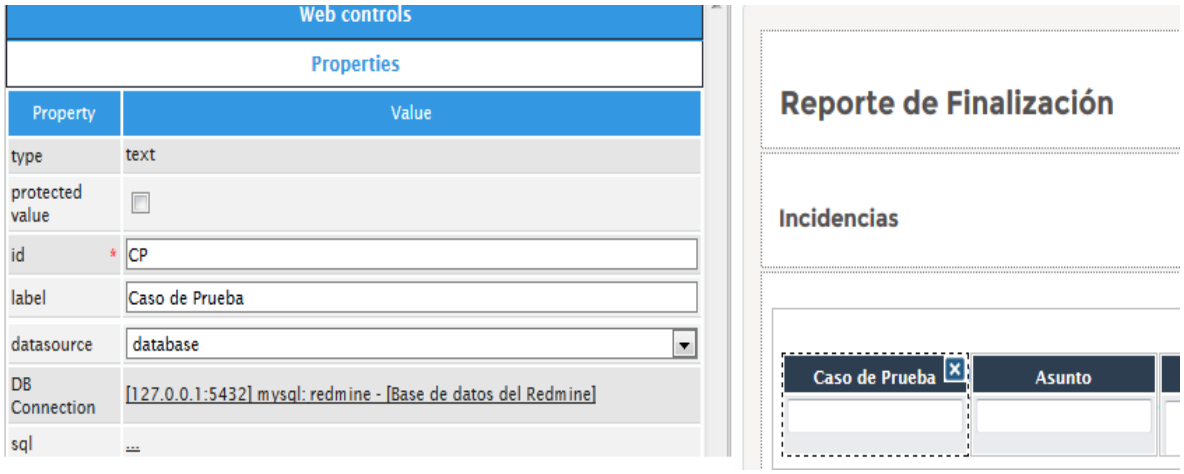


Figura 27: Enlace de grilla de incidencias con la base de datos de Redmine

Al realizar esta asignación de controles, aparecieron algunos mensajes de advertencia, indicando que la sentencia "mysql_connect" se encontraba obsoleta. Se realizó una breve consulta por los foros de Processmaker y se encontró que la solución recomendada era utilizar la función "mysqli_connect". De manera que se sustituyeron las ocurrencias de la función obsoleta en el código fuente de la aplicación y los mensajes de advertencia, dejaron de aparecer.

Para el llenado automático de la grilla de incidencias, se programó un disparador que se ejecuta antes de mostrar el formulario. En este *trigger* se hace uso de la función *executeQuery(String sql)* con la cual se consulta la base de datos de Redmine y se colocan los valores obtenidos en cada columna de la grilla. Para que esta integración suceda lo más transparente posible, es necesario que los campos solicitados en el *SELECT* se llamen igual que en el *grid*.

En la base de datos de Redmine, la información de cada petición es almacenada en varias tablas y con nombres poco significativos. Por tanto y en miras de facilitar la integración, se decidió añadir una nueva tabla al esquema y guardar en ella únicamente la información requerida para la consulta desde Processmaker, ya con los nombres y el orden requerido. Para lograrlo, se implementó un *trigger* en MySQL para que después de cada inserción (*INSERT)* en la tabla *issues*, se agregue una tupla en la nueva tabla "noconformidades". Este disparador se muestra con mayor detalle en el extracto de código 4.

Código 4: Disparador para el llenado de la tabla "noconformidades"

```
DELIMITER $$
CREATE
    TRIGGER `llenarNC` AFTER INSERT
    ON `issues`
    FOR EACH ROW BEGIN
        INSERT INTO noconformidades
        (nomProyecto, CP, asunto, detalle, iteracion, tipo, prioridad)
        SELECT P.name, CV.value, I.subject, I.description AS detalle,
        (SELECT custom_values.value FROM custom_values, issues WHERE issues.id = custom_values.customized_id
         AND custom_values.custom_field_id=2 AND I.id = issues.id) AS iteracion,
         (SELECT custom_values.value FROM issues, custom_values WHERE issues.id = custom_values.customized_id
         AND custom_values.custom_field_id=3 AND I.id = issues.id) AS tipo,
         I.priority_id AS prioridad FROM projects as P, issues as I, custom_values as CV
         WHERE P.id = I.project_id AND CV.custom_field_id = 1 and CV.customized_id = I.id
         ON DUPLICATE KEY UPDATE noconformidades.id = noconformidades.id
    END$$
DELIMITER ;
```

La construcción de este disparador fue uno de los momentos más complicados del desarrollo de la investigación. Luego de lograr la definición de la consulta que obtiene de las diferentes tablas de la base de datos de Redmine, los valores que se desplegarían en Processmaker, se debió solucionar una serie de "problemas", los cuales no se habían tenido en mente, en la conceptualización de la integración de las herramientas. Por ejemplo, las "No Conformidades" se almacenaban bien en el primer INSERT pero luego comenzaban a duplicarse. Cuando las peticiones eran modificadas o eliminadas desde Redmine, los cambios no se veían reflejados en la tabla "noconformidades" pues se necesitaban las restricciones (constraints) de actualización y eliminación. Al final se logró exitosamente implementar el comportamiento deseado.

Luego de superar éstas dificultades, se debió hacer frente a las particularidades de Processmaker para trabajar con *grids*. Ya en ocasiones anteriores se habían manipulado arreglos multidimensionales en PHP, sin embargo los que debían construirse para luego asignarlos a variables del tipo grilla, diferían de todo lo antes experimentado. Fue necesario depurar paso a paso el proceso y observar el contenido y la estructura de las variables, incluso con operaciones para imprimir a consola. Finalmente se resolvió realizar recorridos por varios vectores independientes para finalmente construir el arreglo multidimensional definitivo, si bien se alcanzó el cometido de desplegar la información correctamente, si hubiese querido construir la grilla de una manera más eficiente.

4.4.4 Integración Redmine-Git

Buscando disminuirle al usuario la cantidad de cambios de contexto, sobre todo durante la ejecución de tareas íntimamente relacionadas, se integró el SCM con la herramienta para la gestión de incidencias. Esto por cuanto el flujo de actividades es el siguiente:

- El encargado de pruebas, luego de generar los *scripts* los deposita en el repositorio versionado.
- El probador ingresa a Redmine, selecciona el proyecto y dentro de éste selecciona el repositorio.
- El probador descarga la última versión disponible de los scripts y procede a su ejecución.
- El probador registra en Redmine las no conformidades.

La versión más reciente de Redmine (3.3.0) soporta la integración con varios SCMs, por ejemplo Bazaar, CVS, Git, Mercurial y Subversion. Provee una interfaz desde la que es posible definir el SCM, un identificador y la ruta al repositorio (se ilustra en la figura 28). Incluso, es posible durante la definición de los roles, seleccionar si el usuario puede o no realizar operaciones de escritura sobre el repositorio, algo que anteriormente sólo era posible con la instalación de "extensiones".

Figura 28: Definición de las características del repositorio versionado

Ahora, para enriquecer la integración de ambas herramientas, se instaló el *plug-in* "*Redmine Git Hosting*", con el cual además de la gestión del repositorio es posible: la creación automática de repositorios (la funcionalidad que posee Redmine es la de enlazarse con un repositorio existente), el borrado de repositorios (va más allá del "desvincular" repositorio), manejo de listas de correo, pre-visualización del archivo Readme, visualización de estadísticas, entre otras funciones más.

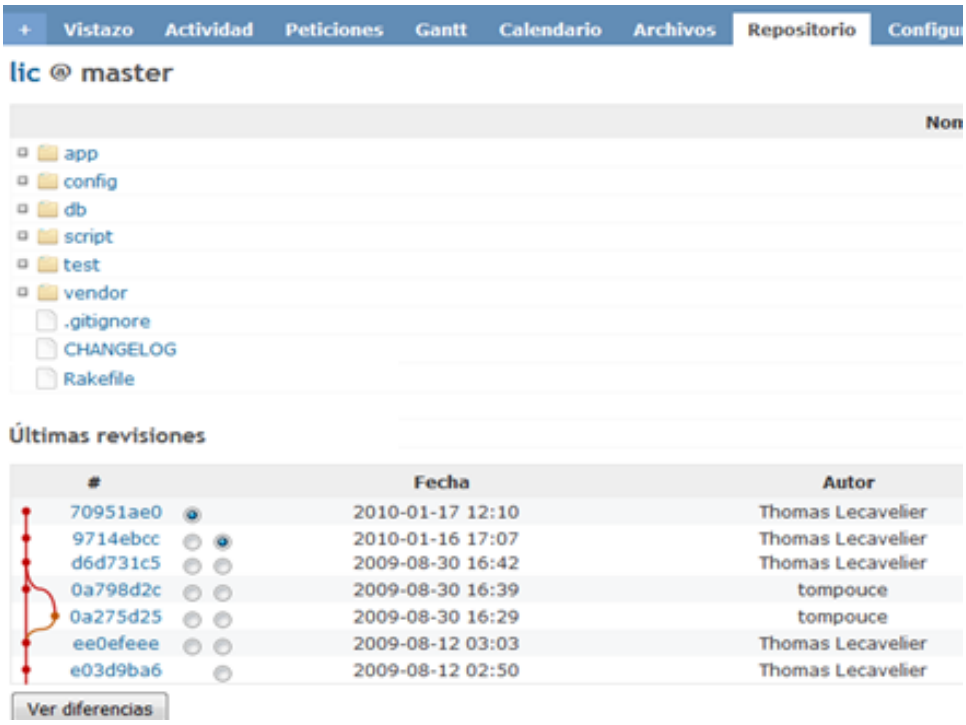


Figura 29: Repositorio Git, con identificador "lic" integrado en Redmine

4.5 OBJETIVO ESPECÍFICO: EVALUAR LA APLICABILIDAD DEL PROTOTIPO DE PLATAFORMA IMPLEMENTADO, COMO APOYO AL PERSONAL DEL CI EN LAS TAREAS DE V&V DE LOS PROYECTOS DE SOFTWARE

Concluida la implementación de todas las funcionalidades, se procedió a evaluar la aplicabilidad del prototipo, como apoyo al personal del CI en las tareas de verificación y validación de software seleccionadas, a saber: verificación de requerimientos, ejecución de pruebas exploratorias, ejecución de pruebas funcionales y ejecución de pruebas de aceptación.

Se llevaron a cabo 3 evaluaciones del prototipo: una "demostración inicial", una "prueba preliminar" y una "prueba definitiva"; y al concluir con la última evaluación, de forma satisfactoria, se le facilitó al señor coordinador de la UCMC un cuestionario (Anexo P) para la valoración de los objetivos propuestos para el presente trabajo de investigación.

La primera demostración del funcionamiento total del prototipo, se realizó en un laboratorio de cómputo ante el señor coordinador de la UCMC. El objetivo de ésta evaluación fue: evidenciar que todos los procesos diseñados se encontraban implementados, que la información transitaba correctamente entre las herramientas y que la automatización de tareas sucedía correctamente. Para ello se ejercitaron todos los caminos posibles, de éxito y de fallo, utilizando datos ficticios sin ningún significado. De la demostración surgieron las siguientes solicitudes:

- Definición de los campos obligatorios en los formularios.
- Mejorar la apariencia de los grids.
- Corregir el cálculo de las métricas "Cobertura de Pruebas" y "Perfil de Fallos".

Efectuados los cambios solicitados, se coordinó con otras unidades del CI para la evaluación preliminar del prototipo desarrollado. Esta prueba fue realizada con datos ficticios, en una sala de reuniones y en presencia del señor coordinador de la Unidad de Desarrollo del Centro de Informática, el señor coordinador de la Unidad de Calidad y Mejora Continua y una de las funcionarias del CI, encargada de calidad. La opinión general luego de la demostración, fue de aprobación y se reconoció el valor que puede tener la utilización de la plataforma, tanto en procesos de calidad como de desarrollo. Al igual que en la evaluación previa, fueron solicitadas las siguientes modificaciones:

- Inclusión de las definiciones para ciertos conceptos en los formularios (por ejemplo: trazabilidad, estimable, verificable).
- Inclusión de hints (pistas) en algunos campos de los formularios para explicar con qué información deben de completarse.
- Inclusión de la posibilidad de adjuntar los datos de prueba utilizados en las pruebas funcionales de software.
- Adición de un campo para la identificación de los usuarios que ejecutan las pruebas de aceptación.
- Corrección del cálculo de la métrica "Perfil de Fallo: Error de Excepción".
- Inclusión de un espacio para observaciones al definir los requerimientos de los datos de prueba y los requerimientos de ambientación.

Implementadas todas las modificaciones solicitadas, se llevó a cabo la evaluación definitiva del prototipo de plataforma. En conjunto con la encargada de calidad, Annia Castro, se realizó un ejercicio de "camino exitoso", de principio a fin, con el proyecto de software (ya culminado): Órdenes de Trabajo, de la Oficina de Servicios Generales de la Universidad de Costa Rica.

En Processmaker se analizaron 4 Historias de Usuario: Decisión Inicial, Solicitud de Orden de Trabajo a Sede Rodrigo Facio, Catálogo de Sectores y Talleres, Ajuste Catálogo de Profesionales por Área; y se diseñaron en total 20 casos de prueba. En Redmine se crearon los proyectos "Solicitud de Órdenes" y "Catálogo de Sectores&Talleres" y para ambos proyectos la información transitó correctamente de una herramienta a la otra. De igual forma los reportes exportados se generaron correctamente y la prueba fue concluida con éxito, no sin antes sugerir algunas mejoras de forma para el llenado de los formularios:

- En el documento Plan de Pruebas, en la sección "Entregables de Prueba", incluir cajas de chequeo (*checkboxes*) para todos los documentos, pues siempre serán los mismos y agregar un espacio "Justificación" para aquellos documentos que no se generaran.
- En el documento Plan de Pruebas, en la sección "Personal", alimentar automáticamente un campo con los funcionarios registrados en la plataforma según rol (Gestor de Calidad, Diseñador de Pruebas, Probador).

- En el documento Diseño de Pruebas, Incluir la posibilidad de importar casos de prueba desde un archivo (por ejemplo .csv) pues es muy común que para los “mantenimientos” y “catálogos” las funcionalidades se repiten y por tanto los casos de prueba necesarios.
- En el documento Diseño de Pruebas, incluir la fecha en la que se diseñan los casos de prueba así como una sección para “Precondiciones iniciales”, para aquellas que todos los casos requieren (por ejemplo haberse identificado exitosamente en el sistema).

En la siguiente sección se presentan las conclusiones del trabajo de investigación realizado, así como algunas recomendaciones y posible trabajo futuro.

CAPÍTULO V: CONCLUSIONES

El presente trabajo final de graduación partió con el objetivo de diseñar un prototipo de plataforma, con herramientas de código abierto, para apoyar al personal del Centro de Informática de la Universidad de Costa Rica en sus tareas de verificación y validación de software. Tras ejecutar cada una de las etapas expuestas en la sección Metodología, tanto este objetivo, como los objetivos específicos de investigación fueron alcanzados con éxito. En las siguientes secciones se enlistan algunas conclusiones y recomendaciones derivadas de la investigación finalizada, así como posible trabajo futuro a realizar en el CI.

5.1 CONCLUSIONES

Los resultados obtenidos en la realización de esta investigación, muestran que en efecto es posible ajustar una o más normas de calidad (en esta ocasión estándares de la IEEE y la ISO) a las particularidades de una organización. Quizás no sea posible su completa implementación y su estricto seguimiento, pero sí se puede generar una estandarización propia, inspirada en las buenas prácticas recomendadas por la industria. Claro está, es necesario tomar muy en cuenta las posibilidades y limitaciones de la entidad donde se desea aplicar para sortear el rechazo inicial que suele provocar la introducción de una nueva forma de hacer las cosas.

Las empresas suelen requerir de apoyo para la adopción de los estándares existentes. Éstos ofrecen un conjunto de procesos valiosos para normalizar las actividades, pero no profundizan en el cómo llevarlos a la práctica. Con el trabajo realizado, se le brindó al Centro de Informática de la Universidad de Costa Rica un conjunto de herramientas que ya poseen implementadas y automatizadas varias de las recomendaciones de los estándares relativos a verificación y validación del software. De manera que la cultura de seguir los procedimientos diseñados, se podrá generar con el simple uso de las herramientas provistas, no será necesario memorizar ninguna secuencia de pasos, fórmulas o estructura de documentos.

Uno de los mayores aportes de la presente investigación es la simplificación de los procesos y de la documentación sugerida por el estándar ISO/IEC/IEEE 29119. La norma es algo extensa en la descripción de los procesos, sólo para los procesos seleccionados sugiere ejecutar hasta 26 tareas distintas. Con la reducción de documentación a una tercera parte y la combinación de tareas, la cantidad de tareas a realizar disminuyó a solo 10 tareas independientes.

Así mismo, otra de las contribuciones del presente trabajo fue la simplificación de los conceptos y el lenguaje utilizados por los estándares internacionales. Más allá de la traducción literal del inglés al español en los documentos, las plantillas propuestas al Centro de Informática utilizan un vocabulario más sencillo e incluyen explicaciones y/o sugerencias para su mejor comprensión. Esta sustitución de términos, manifiesta la necesidad de simplificar los vocablos utilizados en las normas de calidad, en busca de una mejor comprensión por parte de los usuarios. La utilización de términos tan específicos puede resultar un obstáculo, incluso desde la etapa de análisis del estándar y por ende un motivo de rechazo a su utilización.

Es importante considerar que con la presente investigación se proponen soluciones a 5 problemas específicos en materia de verificación y validación de software en el Centro de Informática. Con esto se da a entender que mediante la utilización de la plataforma implementada no se pretende solucionar todas las carencias del CI en torno a la calidad del software desarrollado o adquirido. Con la utilización de las herramientas propuestas se busca apoyar a los funcionarios del Centro de Informática en procesos de V&V de software muy puntuales y mejorar eventualmente la forma en que se ejecutan. Con el paso del tiempo podría lograrse un impacto positivo en el trabajo general de la organización y finalmente en el software utilizado en la Universidad de Costa Rica.

Siempre en estos términos, el hecho de que la investigación realizada se ejecutara en una unidad determinada, no significa que otras organizaciones no puedan hacer uso del producto desarrollado para el apoyo de sus tareas de verificación y validación de software. No sólo los problemas enfrentados resultan comunes a muchas organizaciones que desarrollan software sino que los estándares internacionales consultados poseen recomendaciones generales aplicables a gran variedad de situaciones.

Para la selección de las herramientas que integran la plataforma, se decidió desde un inicio, que fuesen de código abierto. Esta decisión se hizo no sólo por concordancia con la declaración de interés institucional sobre el uso, promoción y desarrollo de aplicaciones de software en la Universidad de Costa Rica, sino también por las necesidades de personalización que en ocasiones no pueden realizarse a través de la interfaz y requieren de la modificación del código fuente.

Desde una perspectiva técnica, muchas de las decisiones tomadas y soluciones implementadas se deben a los recursos disponibles durante la ejecución del proyecto de investigación, y por lo tanto pueden mejorarse:

- Al no contar con un servidor (o más de uno) para la instalación de las herramientas de software, se debió hacer uso de puertos no recomendados o que por lo general utilizan otras herramientas por defecto.
- A nivel de seguridad, no se tomó ninguna previsión más allá de la autenticación por contraseña y el acceso a la base de datos sólo desde el servidor local (*localhost*).
- Sobre la base de datos de Redmine, la estructura de la tabla "noconformidades" no fue normalizada de ninguna manera y el tamaño definido en bytes en su estructura es seguramente desproporcionado.
- Sobre la calidad del código implementado en los disparadores (*triggers)* para y funciones Javascript puede ser más eficiente (por ejemplo la forma en que se llenan los *grids*, los recorridos de los vectores, son candidatos a optimización)

Considero que la evaluación de los productos generados en la investigación, fue la actividad más importante que se llevara a cabo. Sin la valoración y la retroalimentación de los usuarios, el proyecto se habría quedado en un simple ejercicio teórico y sin la garantía de que aquello diseñado e implementado fuera de utilidad, más allá de superar algún criterio experto. Las observaciones recibidas durante las evaluaciones enriquecieron grandemente el producto final: se agilizó el llenado de la estrategia de pruebas, se mejoró la definición de los requerimientos de los datos de prueba, se incluyó información para facilitar la comprensión de los documentos y se implementaron nuevas funcionalidades (como la exportación de las evaluaciones y los reportes) que otorgaron un valor añadido al prototipo.

5.2 RECOMENDACIONES

Es necesario reflexionar acerca de la estabilidad de los requerimientos al iniciar un proceso de desarrollo de software, lo mismo que la delimitación del alcance en un proceso de investigación. El impacto que tiene la variación de los requisitos en plena ejecución de un proyecto, es bastante considerable, no sólo en términos de tiempo sino también en la motivación de los ejecutantes. En el desarrollo de la presente investigación, posterior al análisis de la situación actual y habiendo acordado entre las partes los procesos a sistematizar, se solicitó agregar 6 procesos más. Todo esto derivó en un proceso de negociación, en la ampliación del alcance, en una gran extensión de tiempo y en re-trabajo.

Considero de vital importancia, que en todos los procesos de mejora llevados a cabo en el área encargada del aseguramiento de la calidad del software, se incorpore también algún representante del área de desarrollo. Si ambas unidades desarrollan sus iniciativas de manera independiente, puede terminarse malgastando esfuerzos. Durante la investigación, se tuvo conocimiento sobre clasificación de errores y métricas recabadas por el equipo de desarrollo, sin embargo no eran iguales o compatibles con los procedimientos de la Unidad de Calidad y Mejora Continua. Del mismo modo, los documentos referentes a calidad, especificados en la Metodología para la Administración y Gestión de Proyectos de Sistemas de Información de la Universidad de Costa Rica (Plan Gestión de Calidad, Procedimiento de Aseguramiento de Calidad, Procedimiento Verificación y Control Calidad, Procedimiento Definición de Estándares de Calidad y Métricas) están llenos de conceptos generales sobre Administración de Proyectos y no específicos en ingeniería de software. Esto ha tenido como consecuencia, que la UCMC carezca de algunos insumos básicos para evaluar la calidad del software desarrollado en el CI.

No importa qué tan sencillo, simplificado y automatizado considere el desarrollador de software a su producto; el usuario final probablemente encontrará una manera de mejorarlo. Durante el diseño de los formularios dinámicos en conjunto con el coordinador de la UCMC, se pensó haber simplificado el estándar y organizado los documentos de la mejor manera. Sin embargo muchos de los detalles más importantes sobre la interacción del usuario con las herramientas y el llenado de los documentos, surgieron durante la evaluación realizada por la encargada de calidad. Las observaciones del usuario final siempre serán importantes.

Como última recomendación, me parece importante mantener abierta la posibilidad de cambiar de herramientas de software seleccionadas, sí y solo sí, en el ejercicio de investigación no existe una limitación temporal o algún apremio por la consecución de los objetivos. Conforme se avanza en el desarrollo del trabajo, es más que probable toparse con soluciones de software muy útiles que, quizás se pasaron por alto en las búsquedas iniciales o que tal vez lanzaron una nueva versión y ya implementan funcionalidades requeridas o permiten una configuración más sencilla. Comprometerse con un único conjunto de herramientas, puede también ralentizar el avance del proyecto al requerir modificaciones más complejas o el desarrollo de software intermedio para su integración. Durante la integración de Redmine con Processmaker, si bien a nivel conceptual parecía requerir solamente la consulta de una base de datos externa, para lograr que los datos se mostraran correctamente fue necesaria la programación de código de cierta complejidad y fue indispensable acudir a documentación y foros en Internet, lo cual consumió bastante tiempo.

5.3 TRABAJO FUTURO

Gracias a la facilidad con que pueden ampliarse los procesos modelados en Processmaker, una de las tareas pendientes de la plataforma es la adición de las demás actividades de verificación y validación que la Unidad de Calidad y Mejora Continua incluye en sus responsabilidades. Todas las actividades relativas a las Pruebas de Diseño-Accesibilidad, Pruebas de Arquitectura y Base de Datos, Pruebas de Portabilidad y Pruebas de Seguridad deberán ser agregadas ya como procesos independientes, ya como sub-procesos de los actualmente implementados.

De manera similar será necesario integrar al funcionamiento de la plataforma otras herramientas, para la automatización de otros tipos de prueba de software. Una vez que las pruebas funcionales se automaticen (ya sea con Selenium o con otras herramientas como por ejemplo Cucumber), se podrán incluir otras herramientas específicas para automatizar las pruebas unitarias, las pruebas de portabilidad y las pruebas de carga-desempeño, entre otras.

Las configuraciones realizadas en Redmine y en Processmaker para la investigación podrán ser enriquecidas, por ejemplo con la definición de más métricas de calidad, la inclusión de tipos de error que correspondan con los utilizados por el “equipo de desarrollo” e incluso una integración más eficiente entre las herramientas elegidas.

Durante la segunda evaluación del prototipo de plataforma desarrollado, el señor coordinador del Área de Desarrollo de Sistemas de Información reconoció el impacto positivo que puede tener en el Centro de Informática la utilización de la plataforma por parte de su equipo En el futuro podrían incluirse a la plataforma roles como el de “Analista” y “Desarrollador”, esto con la intención de lograr una mayor integración entre “desarrollo y calidad” y que ambas unidades trabajen de forma articulada. Por ejemplo, de incluirse a los usuarios analistas en Redmine, el probador podría, además de registrar el incidente, asignar su resolución a un desarrollador específico. Él podría entonces especificar en la misma herramienta si el incidente se acepta, se rechaza o se resuelve. Otro posible acoplamiento entre las unidades, podría ser la utilización del repositorio versionado para el almacenamiento y consulta de los casos de uso o las historias de usuario. El analista también podría depositar en Git la historia de usuario terminada y el encargado de calidad acceder a ella, sin requerir el envío de correos o del documento en físico.

Siempre en términos de integración, el Centro de Informática posee un Redmine con varios de sus funcionarios ya incluidos y se poseen definidos los roles propios de la organización. Uno de los trabajos futuros que podrían realizarse es, la inclusión de las configuraciones realizadas en el Redmine del prototipo de plataforma. De esta manera el probador, al registrar una No Conformidad, podrá asignar a uno de los desarrolladores para su corrección y éste podrá también posteriormente determinar si el error fue solucionado.

REFERENCIAS

1- Centro de Informática. (2010). Plan Estratégico de TI del Centro de Informática. San José, Costa Rica.

2- Centro de Informática (14 de abril de 2015). Quiénes somos. Recuperado de http://ci.ucr.ac.cr/quienessomos#page-title

3- A. Castro y A. Alvarado, comunicación personal, diciembre, 9, 2014.

4- Endres, Al. (1999). Implementing Juran's Road Map for Quality Leadership: Benchmarks and Results. John Wiley & Sons, Inc.

5- Guardati, S. y Ponce, A. (2011). Guía de pruebas de software para MoProSoft. REICIS Revista Española de Innovación, Calidad e Ingeniería del Software, volumen 7 (número 2), 28-47. Recuperado de http://www.ati.es/spip.php?article1895

6- García, J., de Amescua, A. y Velasco, D. (2006). Top 10 de factores que obstaculizan la mejora de los procesos de verificación y validación en organizaciones intensivas en software. REICIS Revista Española de Innovación, Calidad e Ingeniería del Software, volumen 2 (número 2), 18-29. Recuperado de http://www.ati.es/spip.php?article536

7- Rodríguez, A. y Cuervo, E. (2006). Nuevas tendencias en sistemas de información: procesos y servicios. PECVNIA Revista de la Facultad de Ciencias Económicas y Empresariales, número 2, 129-158. doi: 10.18002/pec.v0i2.738

8- Universidad de Costa Rica. (2014). Plan Estratégico Institucional 2013-2017. San José, Costa Rica.

9- León, N. E., Pinto, N. y Gómez, L. C. (2011). Herramienta computacional para la evaluación de calidad de productos software enmarcados en actividades de investigación. Scientia et Technica, volumen 2 (número 48), 93-98. Recuperado de http://revistas.utp.edu.co/index.php/revistaciencia/issue/view/169/showToc

10- Harvey, L. y Green, D. (1993). Defining Quality. Assessment & Evaluation in Higher Education, Volumen 18 (número 1), 9-34. doi: http://www.tandfonline.com/doi/abs/10.1080/0260293930180102#.VaArqvl_Oko

11- International Organization for Standardization (2005). Sistemas de gestión de la calidad - Fundamentos y vocabulario. Recuperado de http://www.uco.es/sae/archivo/normativa/ISO_9000_2005.pdf

12- Lewis, W. E. (3era Edición). (2009). Software Testing and Continuous Quality Improvement. Taylor & Francis Group, LLC. Boca Raton, Florida.

13- Pressman, R. (6ta Edición). (2005). Ingeniería del Software: Un Enfoque Práctico. McGrawhill. Madrid.

14- Quality Assurance Institute. (Versión 6.2). (2006). Guide to the CSTE Common Body of Knowledge. Recuperado de http://www.softwaretestinggenius.com/download/CBOK.pdf

15- Salazar, G. (2012). Metodología para enseñar a asegurar la calidad del software a través de técnicas de verificación y validación. Latin American Congress On Requirements Engineering & Software Testing. Medellín.

16- Sommerville, I. (7ma Edición). (2005). Ingeniería del Software. Pearson Educación. Madrid.

17- ISO/IEC/IEEE. (1era Edición). (2010). Systems and software engineering - Vocabulary. Recuperado de http://www.cse.msu.edu/~cse435/Handouts/Standards/IEEE24765.pdf

18- IEEE Computer Society. (2004). Guide to the Software Engineering Body of Knowledge. Los Alamitos, California.

19- Resinas, M. y Recena, M. J. (marzo, 2006). Herramientas de Gestión de Proyectos de Software. I Jornadas de Introducción a la Ingeniería. Jornadas llevadas a cabo en la Escuela Técnica Superior de Ingeniería Informática, Universidad de Sevilla. Recuperado de http://www.docstoc.com/docs/21907548/Herramientas-de-Gestion-de-Proyectos-Software

20- Sánchez, J. C. (2009). Herramientas de Software Libre para Gestión de Proyectos. Recuperado de http://www.acis.org.co/fileadmin/Conferencias/PresSoftwareLibre.pdf

21- Talens-Oliag, S. (abril, 2007). Gestión de proyectos informáticos con software libre. III Jornadas de Software Libre de Albacete. Recuperado de http://www.uv.es/sto/charlas/GPICSL/GPICSL.pdf

22- Centro de excelencia de software libre de Castilla La Mancha. (2010). Análisis de aplicación: Redmine. Recuperado de http://www.bilib.es/recursos/analisis-de-aplicaciones/analisis/doc/analisis-de-aplicacion-redmine.

23- Control de versiones con Git y Github. En Wiki Manuais IES San Clemente Santiago de Compostela. Recuperado en abril, 15, 2014 de http://manuais.iessanclemente.net/index.php/Control_de_versiones_con_Git_y_GitHub

24- Control de versiones. En Wikipedia. Recuperado en abril, 15, 2014 de https://es.wikipedia.org/wiki/Control_de_versiones

25- Tello-Leal, E., Sosa, C. M. y Tello-Leal, D. A. (2012). Revisión de los Sistemas de Control de Versiones Utilizados en el Desarrollo de Software. Revista Facultad de Ingenierías USBMed, volumen 3 (número 1). 74-81. Recuperado de http://web.usbmed.edu.co/usbmed/fing/v3n1.html

26- Díaz, F. N. (2008). Gestión de procesos de negocio BPM (Business Process Management), TIC y crecimiento empresarial. ¿Qué es BPM y cómo se articula con el crecimiento empresarial? Revista

Universidad & Empresa, volumen 10 (número 15). 151-176. Recuperado de http://revistas.urosario.edu.co/index.php/empresa/issue/view/136/showToc

27- Davenport, T. H. (1993). Process Innovation: Reengineering work through Information Technology. Harvard Business School Press.

28- Smith, H. y Fingar, P. (2003). Business Process Management: The Third Wave. Tampa, Florida: Meghan-Kiffer Press.

29- Underdahl, B. (IBM Limited Edition). (2011). Business Process Management For Dummies. Indianapolis, Indiana: Wiley Publishing Inc.

30-ISO/IEC/IEEE. (1era Edición). (2011). Systems and software engineering - Architecture description. Recuperado de http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6129467

31- Reynoso, B. (año). Architect Academy: Seminario de Arquitectura de Software. Recuperado de http://myslide.es/documents/architect-academy-seminario-de-arquitectura-de-software-billy-reynoso-universidad-de-buenos-aires-billyrmicrosoftcomar.html

32- Martínez, W. (2008). Material del curso Arquitectura de Software.

33- Woods, E. y Rozanski, N. (1era Edición). (2005). Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives. New Jersey: Pearson Education.

34- Mackenzie, F. D., Petty, M. D. y Xu, Q. (2004). Usefulness of software architecture description languages for modeling and analysis of federates and federation architectures. Simulation, volumen 80 (número 11). 559-576. Recuperado de http://sim.sagepub.com/content/80/11.toc

35- Retos en Supply Chain. Finalidad de las normas ISO: ¿para qué sirven? Recuperado en marzo, 18, 2015 de http://retos-operaciones-logistica.eae.es/2014/07/finalidad-de-las-normas-iso-para-que-sirven.htm

36- Das, R. Definición de estándares de calidad internacionales. En eHow en Español. Recuperado en marzo, 18, 2015 de http://www.ehowenespanol.com/definicion-estandares-calidad-internacionales-hechos_495257/

37- Estándar internacional. En Wikipedia. Recuperado en marzo, 18, 2015 de https://es.wikipedia.org/wiki/Est%C3%A1ndar_internacional

38- Anónimo. Introducción a las Normas Internacionales de la Calidad. Recuperado de http://bv.ujcm.edu.pe/links/cur_comercial/GesCalidad-2.pdf

39- Aguilera, S. (2015). Los Estándares de Calidad ISO para Desarrollo de Software. Recuperado de http://repositorio.ub.edu.ar:8080/xmlui/handle/123456789/5208

40- Norma ISO/IEC 14598. En EcuRed Conocimiento con todos y para todos. Recuperado en 19, marzo, 2015 de http://www.ecured.cu/index.php/Norma_ISO/IEC_14598

41- Pardo, C. M. (2013). Estándares y Modelos de Calidad del Software. Recuperado de http://evaluaciondesoftware2013.blogspot.com/

42- History of IEEE. En IEEE Advancing Technology for Humanity. Recuperado en 18, marzo, 2015 de http://www.ieee.org/about/ieee_history.html

43- IEEE. (2009). 2009 Annual Report: Celebrating 125 years of engineering the future. Recuperado de http://www.ieee.org/documents/ieee_annual_report_09_complete.pdf

44- Serna, E. y Arango, F. (2011). Prueba del software: más que una fase en el ciclo de vida. Revista de Ingeniería (Universidad de los Andes. Bogotá D.C., Colombia), número 35, 34-40. doi: 10.16924%2Friua.v0i35.144

ANEXOS

ANEXO A – CUESTIONARIO ANÁLISIS DE LA SITUACIÓN ACTUAL DEL CI

1. ¿Trabajan bajo la modalidad "por equipos"?
2. ¿Existe la figura de un Líder de Proyecto de desarrollo?
3. ¿Qué herramientas de software utilizan para el desarrollo? Por ejemplo Visual Estudio, Eclipse, Netbeans?
4. ¿Qué lenguajes de programación son los mayormente utilizados?
5. Cantidad de sistemas en: desarrollo en cola, en mantenimiento (cifra actual o promedio)
6. ¿Los equipos de desarrollo son sometidos a algún proceso de auditoría o "rendición de cuentas"?
7. ¿El proceso de desarrollo está estandarizado? ¿Todos usan las mismas plantillas, tanto de desarrollo como de outsourcing?
8. ¿Cuáles son los paradigmas de desarrollo que están utilizando actualmente (cascada/iterativo/SCRUM)?

PRUEBAS DE SOFTWARE EN EL CENTRO DE INFORMÁTICA

9. ¿Cuál es el procedimiento de pruebas que se utiliza actualmente?
10. ¿Utilizan alguna herramienta para automatizar las pruebas?
11. ¿Cuál es el tiempo promedio dedicados a pruebas respecto a tiempo total de desarrollo e implantación del proyecto?
12. ¿Quiénes ejecutan las pruebas de software (el mismo desarrollador, otro desarrollador, otro equipo)
13. ¿Existe la figura de Líder de Pruebas
14. ¿En qué momento se involucra al cliente en las pruebas de software?
15. ¿Métricas o estadísticas de pruebas? (líneas de código examinadas, porcentaje de errores identificados, tiempos de pruebas, etc.)
16. ¿Se llevan a cabo actividades de verificación o validación a lo largo del ciclo de vida?
17. ¿Siguen algún estándar ISO/IEEE/UCR para desarrollo-pruebas?
18. ¿En el pasado han recibido algún tipo de capacitación calidad, pruebas, técnicas de verificación y validación?
19. ¿Utilizan *Team Foundation Server*? ¿Para qué lo utilizan?

PREGUNTAS GENERALES

20. ¿Cuáles son las necesidades/preocupaciones a (pequeño-medio-largo plazo) identificadas por la alta gerencia?
21. ¿Cuáles son las necesidades/preocupaciones identificadas por los desarrolladores-probadores?
22. ¿Cuáles son las necesidades/solicitudes más frecuentes de los usuarios?

ANEXO B – DUDAS SOBRE EL CUESTIONARIO DE DIAGNÓSTICO

1- Sobre el uso del *Team Foundation Server*, ¿la herramienta es utilizada sólo en el proceso de desarrollo o también para la pruebas?

2-En la pregunta 8 dicen que "Francisco Solera supervisa en algunos proyectos la calidad...". ¿Esto quiere decir que no todos los proyectos son sometidos al control de calidad?

3-¿Qué tipos de proyectos requieren primordialmente la validación de requerimientos funcionales? "Nuevo" y "En Desarrollo" o "*In-Sourcing*" y "*Out-Sourcing*"

4-Cuando un caso programado llega a "Control de Calidad", éste "aplica las herramientas". ¿Qué quiere decir esto? ¿Evalúa el plan de pruebas junto a la hoja de chequeo? ¿Ejercita los casos programados?

5-¿Existe la posibilidad de conocer estos documentos (plantilla de casos de uso, plan de pruebas, hoja de chequeo) y la documentación que se encuentran integrando para ambos procesos de desarrollo?

6-¿Existe alguna manera de conocer el procedimiento de pruebas de los proyectos *out-sourcing*?

ANEXO C – RESUMEN SOBRE LAS PRUEBAS DE PORTABILIDAD

Pruebas de portabilidad en Sistemas Operativos:

Los flujos que generalmente se evalúan son: instalación-ejecución-desinstalación.

Se realizan pruebas de "adaptabilidad": esta es una validación funcional para verificar que el software puede desarrollarse y comportarse como se supone debe hacerlo en cada ambiente meta. Se evalúan dependencias de hardware y software, encapsulación, conversión de textos, entre otros.

Se realizan pruebas de "compatibilidad" o "coexistencia": se evalúa que el software pueda coexistir en el mismo ambiente con otras herramientas no-relacionadas, compartiendo recursos y sin afectar su comportamiento.

Pruebas de instalación: se prueba la habilidad de poder instalarse efectivamente en el los ambientes meta. Puede evaluarse la necesidad de espacio en disco, verificación de software pre-requisito, procedimiento de instalación, interrupción de la instalación, personalización de la instalación, inicialización y desinstalación.

Pruebas de interoperabilidad: se evalúa la capacidad de comunicarse, transferir datos a otras unidades funcionales (aplicaciones) de manera que requiera poco o ningún conocimiento de parte del usuario sobre cómo funcionan éstas.

Pruebas de internacionalización: el propósito de estas pruebas es verificar que el software sea comprendido utilizando el lenguaje local.

Pruebas de "reemplazabilidad": se evalúa la capacidad del producto de software para ser utilizado en lugar de otro producto de software específico para el mismo fin, en el mismo entorno.

Sobre la portabilidad se pueden evaluar varias combinaciones de configuraciones: hardware, RAM, espacio en disco, procesador, velocidad del procesador, resolución del monitor, sistema operativo, versión del sistema operativo (incluyendo *service packs*), navegador, versión del navegador, periféricos de entrada y salida, dispositivos de conexión a la red.

Los errores identificados en las pruebas de portabilidad generalmente se pasan por alto en las pruebas unitarias o de integración.

Pruebas de portabilidad en Navegadores Web:

El "*layout*", los márgenes, la fuente de visualización y otras características visuales de un sitio web, pueden cambiar cuando se ve en un navegador, sistema operativo o resolución de pantalla distintos.

Para realizar estas pruebas, se utilizan herramientas específicas, de forma que no hay que instalar múltiples navegadores o utilizar varias PCs o Macs, cambiando las configuraciones cientos de veces.

Cuando se trata de compatibilidad entre navegadores, hay al menos 3 factores principales que pueden afectar la forma en que el sitio se despliega en los equipos: la versión específica del navegador (más allá de si es IExplorer, Edge, Safari, Chrome, Firefox...), el sistema operativo (Firefox en Mac muestra las páginas de manera distinta a Windows) y la resolución de pantalla.

Multi-navegador, es cuando un sitio web funciona en varios navegadores web (Safari, Chrome, Firefox, Internet Explorer, etc.). Mientras que *cross-browser*, es cuando un sitio web funciona en cualquier navegador y cualquier de sus versiones, que se esté utilizando.

Por ejemplo las herramientas online gratuitas evalúan: Navegador, Motor, Javascript, Java, Flash, Sistema Operativo, Dimensiones, Tiempo de solicitud-inicio-carga.

Pruebas de portabilidad en Dispositivos Móviles:

Pruebas de aplicaciones móviles:

- **Testing de dispositivos:**
  - Pruebas en distintos dispositivos y sistemas operativos.
  - Funcionamiento con las limitaciones de distintos navegadores.
  - Dispositivos con pantalla táctil y sin pantalla táctil.
  - Uso de memoria de la aplicación y cómo funciona ante restricciones de memoria.
  - Consumo de batería de la aplicación.
  - Bloqueo de teclado.
  - Comportamiento de la aplicación al llegar una llamada o mensaje.

- **Testing de usabilidad:** pruebas de facilidad de uso, evaluar que tan intuitiva es la aplicación y probar la aplicación en el teléfono celular usando una sola mano.

- **Testing de desempeño:** Mayor exigencia de tiempos de respuesta rápidos por el servidor, pues la conexión a la red de telecomunicaciones ya de por sí puede ser más lenta.

- **Testing de seguridad:** pruebas de penetración, evaluaciones de vulnerabilidad y encriptación.

- **Testing en la red de telecomunicaciones:**
    - Cómo funciona la aplicación cuando el teléfono está online.
    - Comportamiento en conexión con red Wi-Fi o de la red de telecomunicaciones.
    - Comportamiento cuando se interrumpe la conexión (propenso a pasar con frecuencia en redes móviles).

- **Localización:** El comportamiento de muchas aplicaciones móviles puede depender de la localización del usuario:
    - Si la aplicación tiene distintos idiomas debemos verificar las traducciones.
    - Verificar que el idioma se carga correctamente y que el usuario puede cambiarlo si lo desea.
    - Si los datos o información presentados dependen de la localización, deben incluirse escenarios en las distintas localidades.

Se buscan recursos bloqueados por el robots.txt, si los enlaces están demasiado juntos, si el contenido es más ancho que la pantalla, si el texto es demasiado pequeño para leerlo, utilización de *plug-ins* incompatibles, seguimiento a estándares Web, seguridad, despliegue de gráficas, utilización de colores, errores HTTP, consumo de la red (por ejemplo recursos externos empotrados), hojas de estilo correctas

ANEXO D – REGISTRO DEL PROYECTO DE PRUEBAS

El presente documento recopila los datos mínimos para darle seguimiento al proceso de pruebas.

Nombre del Proyecto:

Número de Iteración:

Módulo o Producto:

Unidad Solicitante:

Gestor de Calidad:

Fecha de Inicio:

Observaciones:

ANEXO E – GUÍA PARA LA REVISIÓN DE REQUERIMIENTOS

Criterio	S/N/NA	Hallazgos, errores, comentario o sugerencias
GENERALIDADES		
¿El documento evaluado cumple con el estándar de documentación utilizado en el Centro de Informática?		
¿Todos los requerimientos listados poseen un identificador único?		
¿Todos los requerimientos listados poseen asociado un nivel de prioridad?		
¿Los requerimientos funcionales se encuentran separados de los no-funcionales?		
¿Todos los requerimientos se especifican de forma concisa y clara? Es decir, ¿poseen una única interpretación?		
¿Si se le entregase esta especificación de requerimientos a un grupo independiente para su implementación, sería capaz de entenderlos?		
¿El nivel de detalle de la especificación de cada requerimiento es el correcto?		
¿Existe algún tipo de error gramatical o de redacción en los requerimientos?		
¿Cada requerimiento se especificó de forma independiente, es decir, no hay requerimientos compuestos?		
¿Existen requerimientos duplicados?		
¿Se encuentran dos o más requerimientos en conflicto/contradicción?		
En caso de existir requerimientos relacionados, ¿se encuentran estos bien referenciados?		
¿Los requerimientos exhiben una distinción clara entre función, entradas, salidas y sus restricciones?		
¿Para las entradas se especificó, su origen y su rango de valores válido?		
¿Para las salidas se especificó su destino, su rango de valores válido y su formato		
¿Cada requerimiento es verificable? Es decir, ¿puede probarse su implementación?		
¿Todos los requerimientos poseen su criterio de aceptación mensurable?		

Criterio	S/N/NA	Hallazgos, errores, comentario o sugerencias
¿Se especifica la acción o el mensaje de error a mostraren caso de fallo del Sistema?		
¿Se encuentran en el documento evaluado frases como “se definirá luego” o “aún por definir”?		
¿El documento evaluado contiene algún tipo de especificación sobre el diseño?		
¿El documento evaluado contiene algún tipo de detalle de programación?		
¿Se encuentran identificados y descritos los usuarios/actores del Sistema?		
¿Cómo conjunto, los requerimientos especificados representan todas las necesidades que se conocen del usuario? En otras palabras, ¿se especificó todo lo que el usuario solicita del sistema?		
¿Se conoce el origen de cada requerimiento? Es decir ¿quién lo propuso y por qué existe?		
REQUERIMIENTOS NO FUNCIONALES		
¿Hay especificación explícita sobre la confiabilidad, robustez o rendimiento del sistema?		
¿Se encuentran requerimientos específicos sobre la seguridad de la aplicación?		
¿Se especificó para las operaciones el tiempo de respuesta?		
ATRIBUTOS DE CALIDAD		
¿Existe algún requerimiento sobre la eficiencia del Sistema?		
¿Existe algún requerimiento sobre la flexibilidad del Sistema?		
¿Existe algún requerimiento sobre la interoperabilidad del Sistema?		
¿Existe algún requerimiento sobre la mantenibilidad del Sistema?		
¿Existe algún requerimiento sobre la portabilidad del Sistema?		
¿Existe algún requerimiento sobre la usabilidad del Sistema?		
¿Existe algún requerimiento sobre la disponibilidad del Sistema?		
¿Existe algún requerimiento sobre la escalabilidad del Sistema?		
¿Se especificaron procesos de recuperación en caso de una caída del Sistema?		

Criterio	S/N/NA	Hallazgos, errores, comentario o sugerencias
DEPENDENCIAS		
¿Se especifican las características del hardware sobre el que correrá la aplicación a desarrollar?		
¿Se especifica el Sistema Operativo en el que se ejecutará la aplicación a desarrollar?		
¿Se especifican todas las interfaces internas y/o externas del Sistema? Si es así, se especifica cómo se comunicará el Sistema a través de estas interfaces?		
¿Los requerimientos especificados se encuentran en concordancia con el Alcance del proyecto?		
¿Es posible implementar todos y cada uno de los requerimientos presentes en el documento evaluado, mediante las técnicas y herramientas disponibles?		
VERIFICACIÓN DE ARQUITECTURA Y BASES DE DATOS		
¿El diseño de la solución cuenta con la firma de aprobación del Arquitecto de Software?		
¿El diseño de la(s) base(s) de datos relacional(es) cuenta con la firma de aprobación del experto?		

Definiciones:

Confiabilidad: probabilidad en que un producto realizará su función prevista sin incidentes por un período de tiempo especificado y bajo condiciones indicadas.

Robustez: capacidad de un sistema de reaccionar en forma adecuada frente a situaciones imprevistas.

Rendimiento: medida o cuantificación de la velocidad/resultado con que se realiza una tarea o proceso. Puede verse también como la cantidad de trabajo que una aplicación debe realizar por unidad de tiempo (por ejemplo transacciones por segundo o mensajes procesados por segundo).

Seguridad: habilidad de un sistema para controlar, monitorear y auditar en forma confiable quién puede realizar qué acciones sobre el sistema y sus recursos.

Eficiencia: uso racional de los recursos con que se cuenta para alcanzar un objetivo predeterminado. A mayor eficiencia menor la cantidad de recursos que se emplearán, logrando mejor optimización y rendimiento.

Flexibilidad: habilidad del sistema para ser flexible frente a cambios inevitables durante su desarrollo y luego de su despliegue.

Interoperabilidad: habilidad de dos o más sistemas de intercambiar información y utilizar la información intercambiada, sin un esfuerzo especial por parte del cliente. Debe poder ocurrir sin planeación o negociación previa entre quienes operan estas máquinas.

Mantenibilidad: Propiedad de un sistema que representa la cantidad de esfuerzo requerida para conservar su funcionamiento normal o para restituirlo una vez se ha presentado un evento de falla.

Portabilidad: capacidad de un programa o sistema de ejecutarse en diferentes plataformas o arquitecturas con mínimas modificaciones.

Usabilidad: facilidad con la cual las personas interactúan con la aplicación en forma efectiva, y como el sistema provee soporte al usuario en este sentido.

Disponibilidad: porcentaje de tiempo que un sistema es capaz de realizar las funciones para las que está diseñado. Probabilidad de que un sistema, en cierto momento, esté en funcionamiento y sea capaz de proporcionar los servicios solicitados.

Escalabilidad: propiedad deseable en un sistema, que indica su habilidad para poder hacerse más grande sin perder calidad en sus servicios.

ANEXO F – EVALUACIÓN DE CASO DE USO

Criterio	Sí/No	Comentario
¿El CU posee un nombre acorde con la funcionalidad que pretende especificar?		
¿El CU posee un código o identificador único que facilite su trazabilidad?		
¿El CU se encuentra debidamente ligado con una tarea de la WBS/EDT?		
¿El CU identifica correctamente a todos los actores involucrados en los escenarios descritos?		
¿La descripción del CU es correcta, no ambigua y verificable?		
¿Se establecen las precondiciones necesarias para poder ejecutar el CU?		
¿Se diferencian claramente el Flujo Principal de los Flujos Alternativos?		
¿Se definen las post-condiciones que deben cumplirse al concluir el CU?		
¿Es posible para el Equipo de Calidad extraer Criterios de Aceptación a partir de estas post-condiciones?		
¿En caso de que se presente un error en la ejecución del CU, se definen los mensajes de alerta o el cómo manejará el Sistema las excepciones?		
¿El CU contiene escenarios reutilizables en la generación de Casos de Prueba?		
¿El CU posee adjunto los prototipos de interfaz gráfica?		
¿El CU posee adjunto el diagrama de clases respectivo?		

Observaciones generales:

ANEXO G – EVALUACIÓN DE HISTORIA DE USUARIO

Criterio	Sí/No	Comentario
¿La Historia de Usuario posee un nombre significativo?		
¿La Historia de Usuario posee un código o identificador único que facilite su trazabilidad?		
¿Se encuentra presente el apartado “Módulo al que pertenece”?		
¿Se especifica detalladamente cómo ubicarse en el contexto en que sucede la Historia de Usuario, desde una perspectiva del Sistema total?		
¿Se especifica el rol que ejecuta la Historia de Usuario?		
¿La Historia de Usuario indica si es necesario algún tipo de filtrado de la información? En otras palabras, ¿indica si la información previa o resultante de la Historia debe poder filtrarse?		
¿La Historia de Usuario posee adjunto los prototipos de la interfaz gráfica?		
¿La Historia de Usuario referencia explícitamente la necesidad de cumplir con Requerimientos No Funcionales?		
¿La Historia de Usuario utiliza lenguaje común del usuario? Es decir, ¿evita tecnicismos y referencias a detalles de implementación?		
¿La Historia de Usuario tiene sentido por sí misma? Es decir, ¿es independiente de las demás?		
¿La Historia de Usuario es “estimable”? ¿Permite estimar el tiempo total que tomará implementarla?		

Criterio	Sí/No	Comentario
¿La Historia de Usuario es Verificable? En otras palabras, ¿puede comprobarse fácilmente si se cumple o no?		
¿Se indica la prioridad que tiene para el usuario en el conjunto de Historias de Usuario?		
¿La Historia de usuario viene acompañada de sus criterios de aceptación?		
¿Los criterios de aceptación se encuentran escritos siguiendo la fórmula “En caso de que [Contexto], cuando [Evento], el sistema [Comportamiento esperado]” o similar?		
¿Se definen los mensajes de alerta o error, o cómo manejará el Sistema las excepciones?		

Observaciones generales:

ANEXO H – RESULTADO DE LAS PRUEBAS EXPLORATORIAS

Nombre del Proyecto:

Módulo o Producto:

Gestor de Calidad Responsable:

Funcionalidad Probada	Resultado	Anexo

Decisión de Calidad:

¿El entregable de software superó las pruebas exploratorias?

SÍ _____ NO _____

ANEXO I – PRUEBAS DE ACEPTACIÓN

Nombre del Proyecto:

Módulo o Producto:

Usuario Registrante:

Actividad Realizada	Comportamiento Observado	Anexo

ANEXO J – PLAN DE PRUEBAS

1-Contexto de las Pruebas

Proyecto:	
Módulo o Producto:	
Número de Iteración:	
Características a probar	Características que no se probarán

2-Estrategia de Pruebas

Entregables de prueba: *Documentos y productos del proceso de pruebas que se generarán*
Nivel/Tipo/Técnica de prueba: *Qué pruebas se ejecutarán (funcionales, portabilidad, aceptación, caja negra, caja blanca)*
Criterios de completitud de las pruebas: *Qué debe cumplirse para considerar las pruebas como "terminadas"*
Criterios de suspensión y reanudación: *Qué debe suceder para que las pruebas se suspendan y luego se reanuden*
Pruebas de regresión: *Qué debe realizarse para las futuras pruebas de regresión cuando se encuentren más módulos integrados*

2.1-Requerimientos de Datos de Prueba

Requerimiento	Origen	Formato	Responsable	Confidencial

2.2-Requerimientos del Ambiente de Pruebas

Hardware	Software	Herramientas Específicas
Servidores, PC Clientes	*Facilidades del laboratorio*	

4-Personal

Funcionario	Rol	Responsabilidades	Horas Dedicadas
	Líder		
	Diseñador		
	Probador		

5-Cronograma

Tarea	Responsable	Participantes	Esfuerzo	Fecha de Finalización
			Días - Horas	

ANEXO K – DISEÑO DE PRUEBAS

1- Características a Probar

Identificador	Descripción	Prioridad	Criterio de Aceptación
Código del requerimiento o de la Historia de Usuario	*Funcionalidad*		*Qué debe hacer el sistema para satisfacer el requerimiento*

2-Casos de Prueba

Identificador	Objetivo	Prioridad	Precondiciones	Datos de entrada	Resultado esperado

3-Datos de Prueba

Descripción	Responsable	Período de Necesidad	Archivado o Desechado	Estado
				Se llena cuando se prepara el ambiente
				Preparado
				Ausente

4-Ambiente de Pruebas

Descripción	Responsable	Período de Necesidad	Estado
Hardware			*Se llena cuando se prepara el ambiente*
Software			*Preparado*
Herramienta			*Ausente*

ANEXO L – REPORTE DE INCIDENTE

Proyecto:

Módulo o producto:

Número de iteración:

1-Información temporal:

Fecha: Fecha en que se identifica la No Conformidad

2-Responsable del registro:

Nombre de la persona que se encuentra ejecutando el caso de prueba en cuestión.

3-Contexto:

Ítem de Prueba	Caso de Prueba
Puede ser el módulo o el componente, o lo que se haya descrito como ítem de prueba en el diseño	Nombre o Código del Caso de Prueba que identificó el error

4-Detalle del Incidente:

Descripción: Descripción del comportamiento del sistema, el cual difiere de aquel esperado.	
Prioridad	Prioridad de resolución de la no conformidad
Estado	Abierto - Cerrado - Rechazado - Corregido - Asignado para resolución - Asignado para aprobación
Anexo	Pantallazo del problema encontrado

ANEXO M – REPORTE DE FINALIZACIÓN DE LAS PRUEBAS

1-Pruebas ejecutadas

Caso de Prueba	Objetivo de la Prueba	Resultado

1.1 Incidencias

Caso de Prueba	Título	Detalle	Tipo de Error	Prioridad

2. Métricas

2.1. Cobertura de pruebas

Número de Casos de Prueba Ejecutados	Número de Casos de Prueba Diseñados	Cobertura

2.2. Madurez de las pruebas

Número de Casos de Prueba Satisfactorios	Número de Casos de Prueba Diseñados	Madurez

2.3. Densidad de defectos

Total de Defectos Encontrados	Total de Funcionalidades Probadas	Densidad

2.4. Perfiles de fallos

Cantidad de Errores de Funcionalidad	Densidad
Cantidad de Errores de Validación	Densidad
Cantidad de Errores de Excepción	Densidad
Cantidad de Errores de Usabilidad	Densidad
Cantidad de Errores de No Correspondencia con lo Documentado	Densidad

3-Evaluación del Proceso de Pruebas Concluido

3.1-Apego al Plan de Pruebas

Factores que alteraron el proceso:

3.2-Evaluación de completitud de las pruebas

Criterio de completitud:

3.3-Lecciones Aprendidas

ANEXO N – ESPECIFICACIÓN DE REQUERIMIENTOS DE USUARIO

1-Introducción

1.1- Proyecto

Nombre del sistema que se va a desarrollar

1.2- Analista Responsable

Analista que interactúa con el usuario para la obtención de sus requerimientos

1.3- Usuario Experto

Representante de los usuarios que solicitaron el nuevo sistema

1.4- Definiciones, Acrónimos y Abreviaturas

1.5- Documentos Relacionados

Referencias a las plantillas de caso de uso o a las plantillas de historia de usuario, elaboradas durante la elicitación de los requisitos, así como cualquier otra documentación necesaria.

2- Requerimientos del Sistema

2.1- Interfaces del Sistema

Se describen las interfaces, por ejemplo hardware con el que debe comunicarse, software con el que tiene algún tipo de comunicación, redes a las que deba acceder, entre otras.

2.2- Requerimientos Funcionales

Identificador	Funcionalidad Solicitada	Criterios de Aceptación	Prioridad

2.3- Requerimientos No Funcionales

Identificador	Descripción del Requerimiento
	Limitaciones de Diseño
	Estándares que deba seguir
	Atributos de Calidad

3- Aprobación del Documento

Usuario Experto	Analista Responsable	Encargado de Calidad	Fecha

ANEXO Ñ – REGISTRO DE HISTORIA DE USUARIO

Enunciado de la Historia de Usuario

Identificador	Rol	Funcionalidad	Razón/Objetivo

Criterios de Aceptación por Funcionalidad

Número	Título	Contexto	Evento	Resultado

ANEXO O – MANUAL DE USUARIO

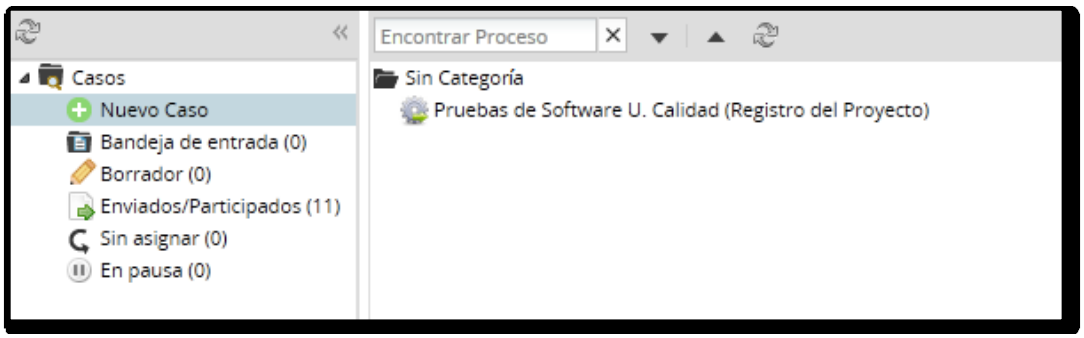
1. Flujo de tareas a ejecutar y su responsable:

i. Creación del Caso: la ejecuta el líder de pruebas.
ii. Registro del Proyecto: la ejecuta el líder de pruebas.
iii. Verificación de Requisitos: la ejecuta el líder de pruebas.
iv. Registro de Resultado de las Pruebas Exploratorias: la ejecuta el líder de pruebas.
v. Planificación de las Pruebas: la ejecuta el líder de pruebas.
vi. Diseño de Pruebas: la ejecuta el encargado de pruebas.
vii. Ejecución de las Pruebas: la ejecuta el encargado de pruebas.
viii. Registro de Incidencias: la ejecuta el probador.
ix. Finalización de las Pruebas: la ejecuta el líder de pruebas.
x. Pruebas de Aceptación: la ejecuta el usuario solicitante.

A continuación se explican una a una, el desarrollo de dichas actividades en la Plataforma Computacional.

1.1 Creación del Caso:

En Processmaker, cada instancia de proceso, se conoce como caso. Esto significa que cada vez que se va a valorar un módulo o entregable de software, debe de crearse un caso. Luego de identificarse como usuario con el rol Líder de Pruebas, en el panel de la izquierda, debe seleccionarse la opción Nuevo Caso. Esto desplegará la siguiente información:



Seleccione el proceso “Pruebas de Software U. Calidad” y seguidamente en el panel de la derecha, elija la opción **Iniciar Caso.**

Esto dará inicio al proceso e invocará automáticamente la ejecución de la tarea inicial **Registro de Proyecto**

1.2 Registro de Proyecto:

Se desplegará el siguiente formulario.

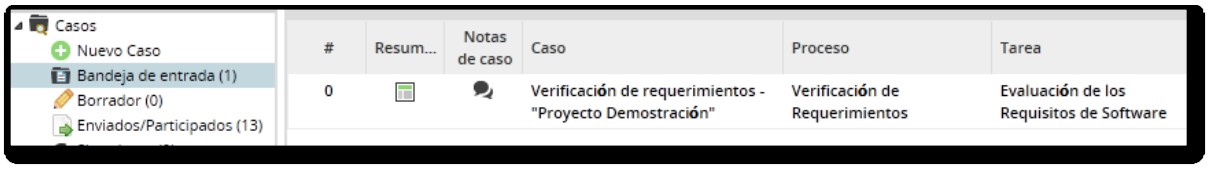
Todos los campos son obligatorios, a excepción de la sección para Observaciones. La información solicitada es la siguiente:

- Nombre del Proyecto: nombre del proyecto de desarrollo al que pertenece el producto de software a evaluar.
- Número de Iteración: número de *sprint* según la metodología SCRUM.
- Módulo o Producto: nombre del módulo o conjunto de módulos, o del entregable de software que se va a evaluar.
- Unidad Solicitante: Unidad académica o de investigación de la Universidad de Costa Rica, que solicitó el desarrollo de software.

- Gestor de Calidad: campo de llenado automático, con el nombre del usuario líder identificado al momento de llenarse el formulario.
- Fecha de Inicio: fecha del día obtenida automáticamente.
- Observaciones: campo libre para indicar cualquier particularidad del proceso a iniciar.

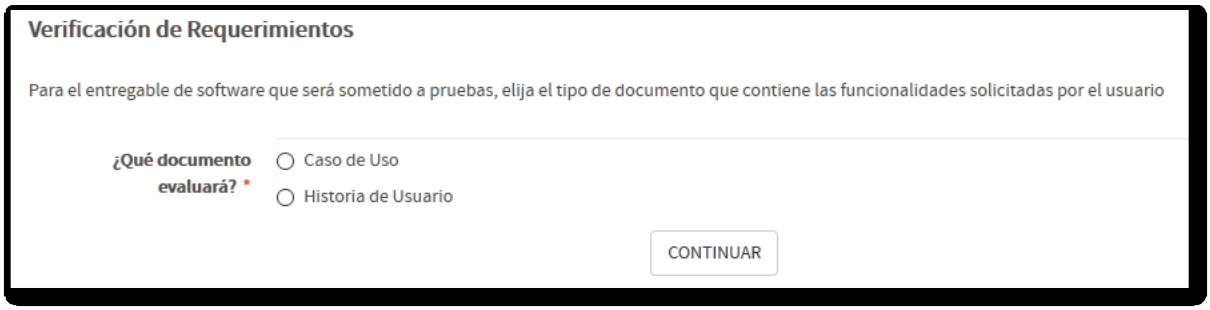
Debe presionarse el botón CONTINUAR, para avanzar al siguiente paso y para que el contenido de los campos llenados, se almacene persistentemente.

La siguiente tarea a ejecutar es la **Evaluación de los Requisitos de Software**, ésta aparecerá inmediatamente en la Bandeja de Entrada, accesible desde el panel de la izquierda.



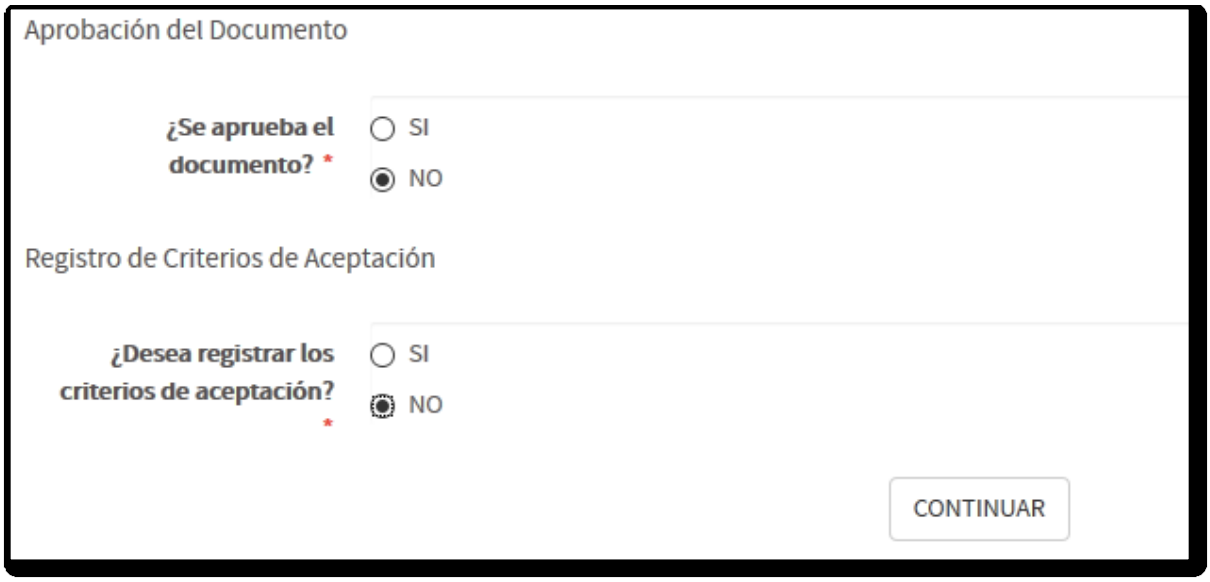
1.3 Verificación de Requisitos:

En la Verificación de Requisitos, lo primero que debe realizarse es seleccionar el tipo de documento que contiene los requerimientos del usuario: Caso de Uso o Historia de Usuario. De acuerdo a lo seleccionado, así se desplegará un formulario para evaluar la calidad del documento.



En el formulario de evaluación, el campo del Proyecto se llena automáticamente, en el espacio Caso de Uso o Historia de Usuario, se ingresa el código identificador del documento evaluado (es el único campo obligatorio), y en la sección Documento evaluado es posible adjuntar el documento revisado. Luego, una sucesión de preguntas del tipo Sí o No, y la posibilidad de ingresar un comentario si se considera necesario.

Al final del documento, se presentan dos preguntas:

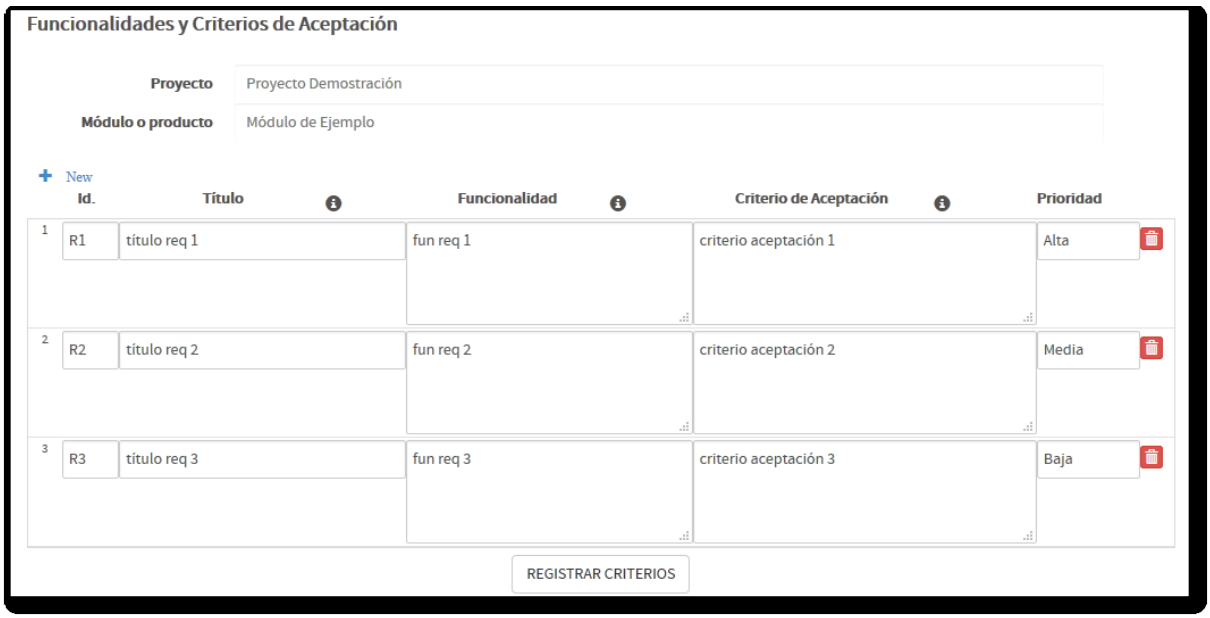


Si se rechaza el documento y se selecciona CONTINUAR, se procede a la exportación de la evaluación realizada, en formato PDF o DOCX.

Si se aprueba el documento se sigue a la siguiente tarea, la cual puede ser:

- **Registrar Funcionalidades y Criterios de Aceptación**, si se responde que SÍ a la segunda pregunta.
- **Registrar los Resultados de las Pruebas Exploratorias**, si se responde NO a la segunda pregunta.

La tarea Registrar Funcionalidades y Criterios de Aceptación, permite especificar los requerimientos del usuario, extraídos del Caso de Uso o de la Historia de Usuario y sus criterios de aceptación. A continuación el formulario a utilizar para dicha tarea:



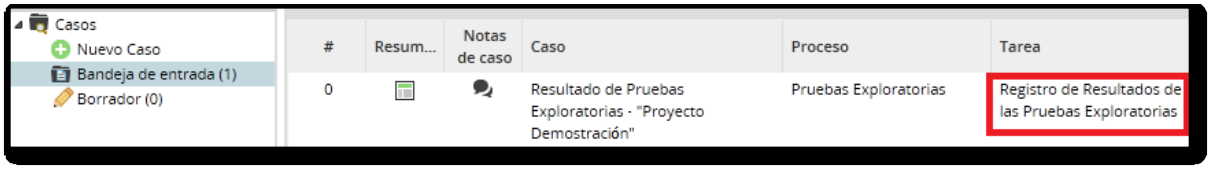
Con la opción +New pueden agregarse más filas a la tabla, mientras que seleccionando el ícono del basurero rojo se eliminan las filas. Los campos a llenar son los siguientes:

- Proyecto: se llena automáticamente con la información del Registro del Proyecto.
- Módulo o producto: igual que el campo anterior, es llenado automáticamente.
- Id: identificador único del requerimiento.
- Título: nombre de la funcionalidad solicitada.
- Funcionalidad: descripción de la funcionalidad solicitada.
- Criterio de Aceptación: comportamiento que debe implementar el sistema para demostrar que el requerimiento fue satisfactoriamente implementado, según el usuario. Pueden ser varios criterios de aceptación por cada requerimiento.
- Prioridad: importancia relativa, con la que el usuario clasifica los requerimientos.

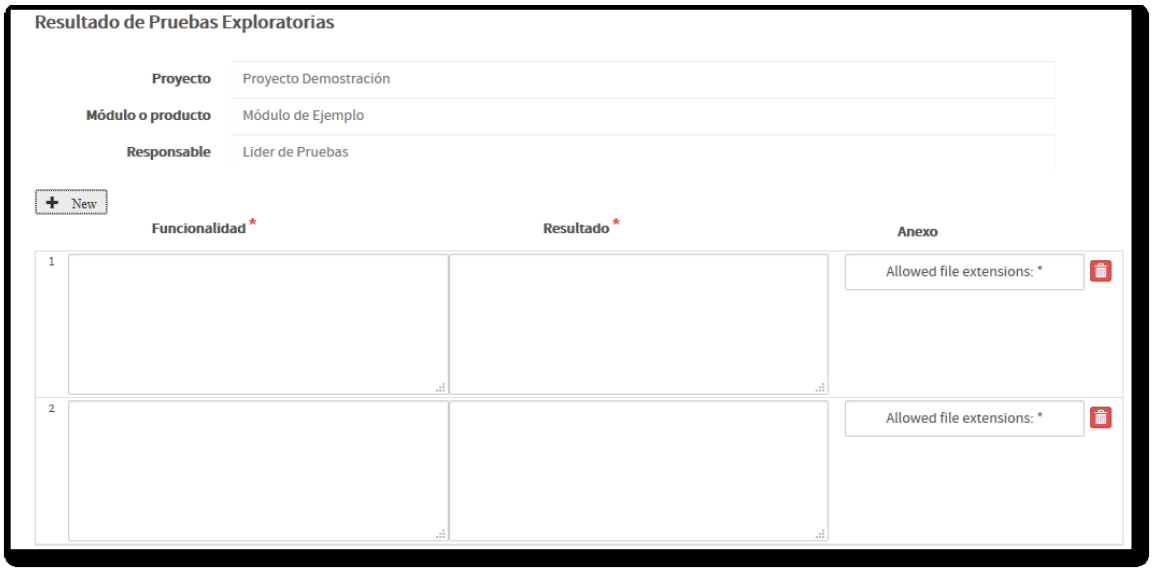
Al concluir el llenado del formulario, se debe seleccionar la opción REGISTRAR CRITERIOS, para que la información ingresada se almacene de manera persistente y así continuar con la siguiente tarea **Registro de Resultados de las Pruebas Exploratorias.**

1.4 Registro de Resultado de las Pruebas Exploratorias

En la bandeja de entrada, aparecerá disponible la posibilidad de ejecutar esta tarea.



Al realizar doble clic sobre ella, se desplegará el siguiente formulario:

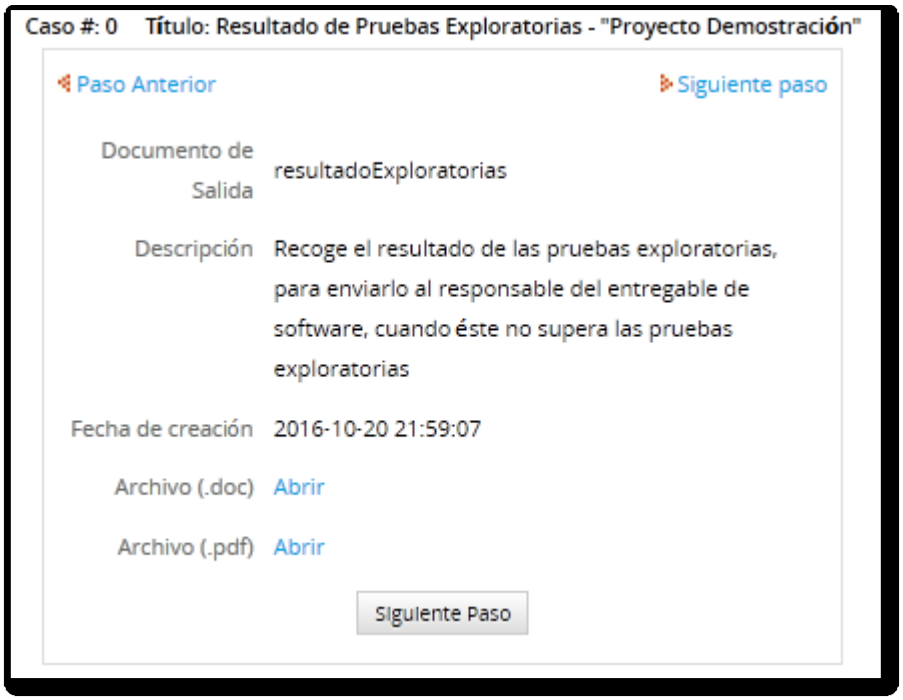


De igual forma que en la tabla de requerimientos y criterios de aceptación, es posible agregar o eliminar filas, seleccionando New o el basurero rojo, respectivamente. Los campos a llenar son los siguientes:

- Proyecto: se llena automáticamente con la información de Registro de Proyecto.
- Módulo o Producto: se llena automáticamente según Registro de Proyecto.
- Responsable: se llena según el usuario identificado con rol de Líder de Prueba.
- Funcionalidad (obligatorio): funcionalidad probada durante la examinación.
- Resultado (obligatorio): comportamiento observado del sistema durante la examinación.
- Anexo: es posible adjuntar una imagen del comportamiento observado.

Al concluir el documento se presenta una pregunta al evaluador:

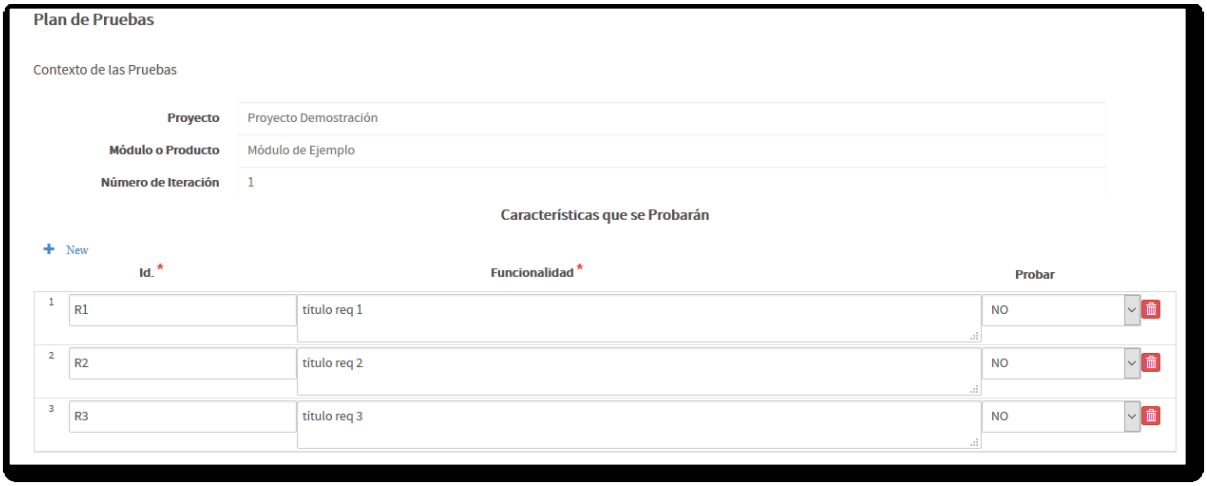
Si se responde que las pruebas exploratorias **NO** fueron satisfactorias, se procede a exportar la evaluación realizada (de manera similar que para la evaluación de los casos de uso e historias de usuario).



Si responde que el entregable de software, **SÍ** superó las pruebas exploratorias, se prosigue con la siguiente tarea: **Planificación de las Pruebas.**

1.5 Planificación de las Pruebas

Luego de seleccionar la tarea en la Bandeja de Entrada, se presentará el siguiente formulario:



La sección **Características que se Probarán** (única sección obligatoria de todo el formulario), se llena automáticamente, con la información que se ingresó en la tarea **Registro de Funcionalidades y Criterios de Aceptación**. Si esta tarea fue omitida, esta sección se encontrará vacía.

Los siguientes campos a llenar son:

- Entregables de prueba: se seleccionan los documentos y productos del proceso de pruebas que se generarán.
- Nivel/Tipo/Técnica de prueba: se define si se ejecutarán de caja negra, caja blanca, funcionales, de desempeño, de seguridad, entre otras.
- Criterios de completitud de las pruebas: condición que debe alcanzarse para dar por concluido el proceso de pruebas.
- Criterios de suspensión y reanudación de las pruebas: bajo qué condiciones se detiene la ejecución de las pruebas y bajo qué criterios se reanudarán.
- Pruebas de regresión: en caso de que las funcionalidades evaluadas se relacionen con otros módulos del sistema, qué tipos de pruebas de regresión deberían programarse.

A continuación se presenta una sección para especificar los requerimientos de los datos de prueba y los requerimientos del ambiente de pruebas.

Para los **Requerimientos de Datos de Prueba**, los campos a llenar son los siguientes:

- Nombre: cómo se conoce al dato o conjunto de datos necesario.
- Origen: procedencia de los datos.
- Formato: formato de archivo en el que deben provenir los datos.
- Responsable: responsable de conseguir los datos.
- Confidencial: los datos deben tratarse de forma confidencial sí o no.

Para los **Requerimientos de Hardware**, los campos a llenar son los siguientes:

- Hardware: hardware requerido para la ejecución de las pruebas.
- Responsable: responsable de conseguir el hardware.

Para los **Requerimientos de Software**, los campos a llenar son los siguientes:

- Software: programas o herramientas de software necesarias para realizar las pruebas.
- Responsable: responsable de conseguir el software.

Posterior a esta sección se presenta la posibilidad de registrar el **Personal** que participará en las pruebas de software y el **Cronograma** de actividades.

Para la especificación del personal, los datos disponibles son:

- Funcionario: se selecciona el nombre del funcionario, de los usuarios registrados en la plataforma.
- Rol: se define si será líder, encargado o probador.
- Responsabilidades: se especifican responsabilidades específicas.
- Horas dedicadas: esfuerzo estimado que requerirá su participación en el proceso.

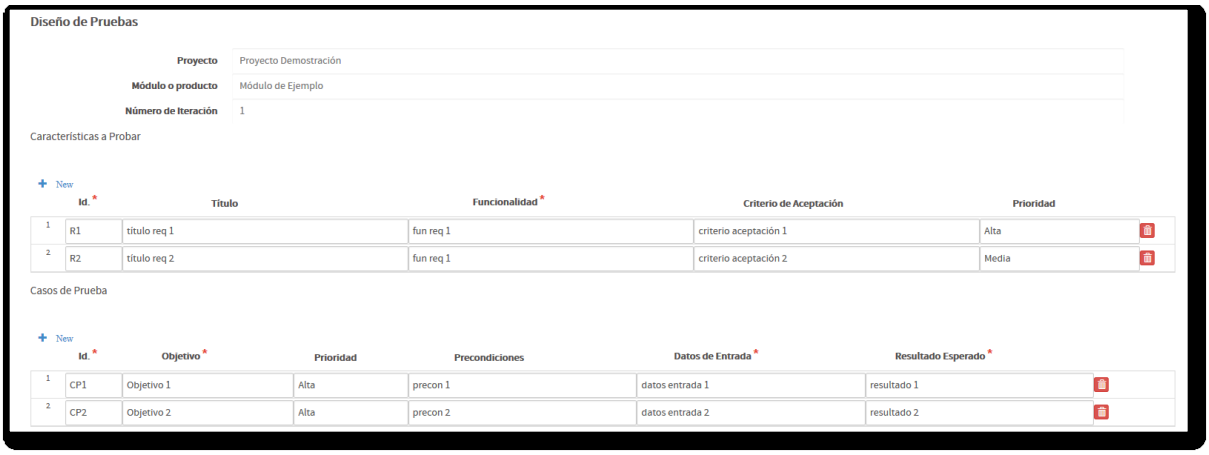
Para la construcción del cronograma, los datos disponibles son:

- Tarea: actividades que deberán realizarse en el proceso de pruebas.
- Responsable: quién es el funcionario responsable de que la tarea se lleve a cabo.
- Participantes: otros funcionarios que acompañan al responsable en la ejecución.
- Esfuerzo: horas estimadas que se pueden llegar a invertir en la tarea.
- Fecha Finalización: fecha propuesta o estimada para la finalización de la tarea

Al concluir el formulario, como en los anteriores, es necesario seleccionar la opción **CONTINUAR**, para que la información ingresada, se almacene persistentemente.

1.6 Diseño de Pruebas

La tarea **Diseño de Pruebas** es ejecutada por un usuario con rol de Encargado de Pruebas. Al seleccionar la tarea desde la Bandeja de Entrada, se muestra el siguiente formulario:



En este formulario se llena automáticamente la información sobre la características a probar. Ésta información proviene desde lo que en el Plan de Pruebas se definió que sí se probaría.

La siguiente sección **Casos de Prueba**, tiene los siguientes campos:

- Id: identificador único del caso de prueba (obligatorio).
- Objetivo: finalidad del caso de prueba (obligatorio).
- Prioridad: importancia relativa entre el conjunto de casos de prueba.
- Precondiciones: pasos a ejecutar antes del caso de prueba.
- Datos de Entrada: datos con los que se evalúa el requerimiento (obligatorio).
- Resultado Esperado: comportamiento que debería de demostrar el sistema, con los datos de entrada utilizados (obligatorio).

Las siguientes secciones sobre **Datos de Prueba** y **Ambiente de Prueba**, se llenan automáticamente desde la información ingresada en el Plan de Pruebas.

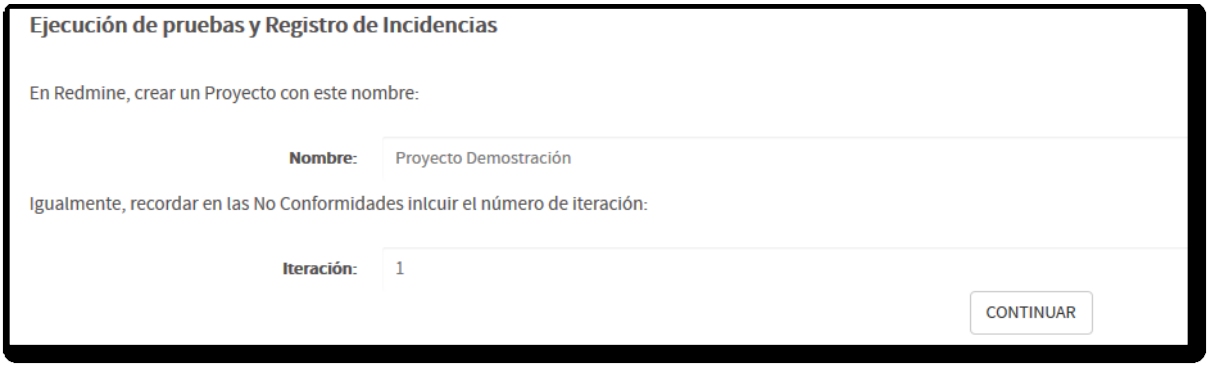
En esta sección se puede: adjuntar los datos de prueba, si éstos están ya disponibles, registrar si los requerimientos de datos y de ambiente están listos o aún pendientes, agregar observaciones si se considera necesario.

Al concluir el formulario debe seleccionarse CONTINUAR para que la información sea almacenada persistentemente.

La siguiente tarea a ejecutar es **Ejecución de Pruebas.**

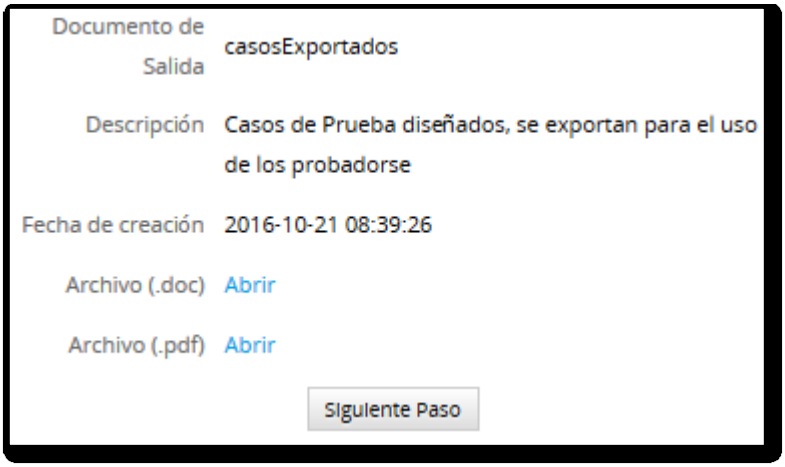
1.7 Ejecución de las Pruebas

Antes de comenzar a ejecutar los casos de prueba diseñados, Processmaker dispone del siguiente recordatorio:



De tal manera que debe de ingresarse a Redmine y crear un proyecto con el nombre que coincide con el ingresado en la tarea Registro de Proyecto en Processmaker y tener presente que cada incidente registrado debe tener la iteración que coincide con el sprint de desarrollo.

Al elegir la opción CONTINUAR, se exportarán los casos de prueba diseñados de la siguiente manera:



Casos de Prueba

Nombre del Proyecto: Proyecto Demostración

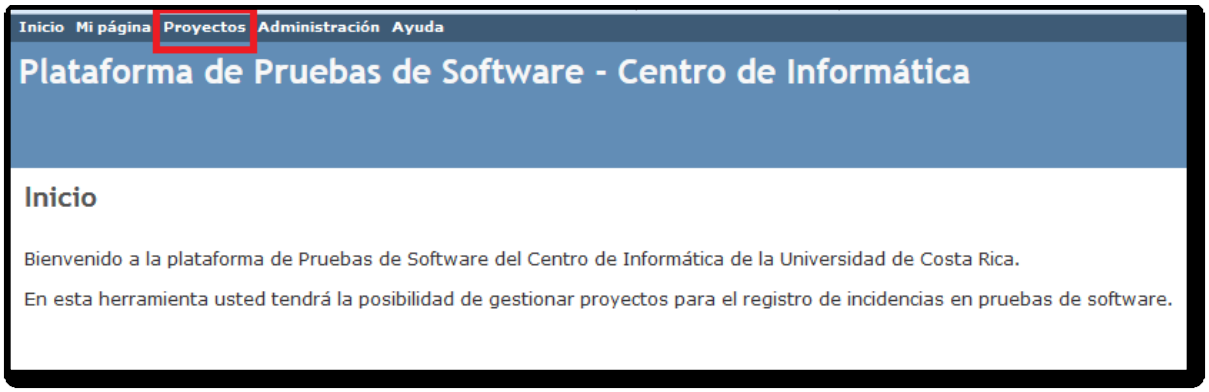
Número de Iteración: 1

Producto: Módulo de Ejemplo

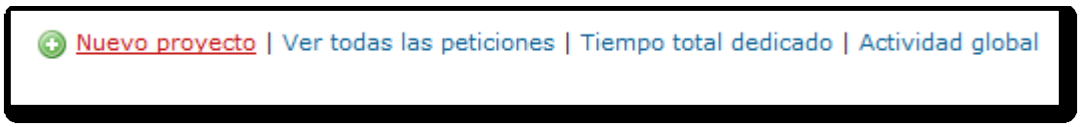
Id.	Objetivo	Precondiciones	Datos de Entrada	Resultado Esperado
CP1	Objetivo 1	precon 1	datos entrada 1	resultado 1
CP2	Objetivo 2	precon 2	datos entrada 2	resultado 2

Para crear un Proyecto en Redmine debe realizarse lo siguiente:

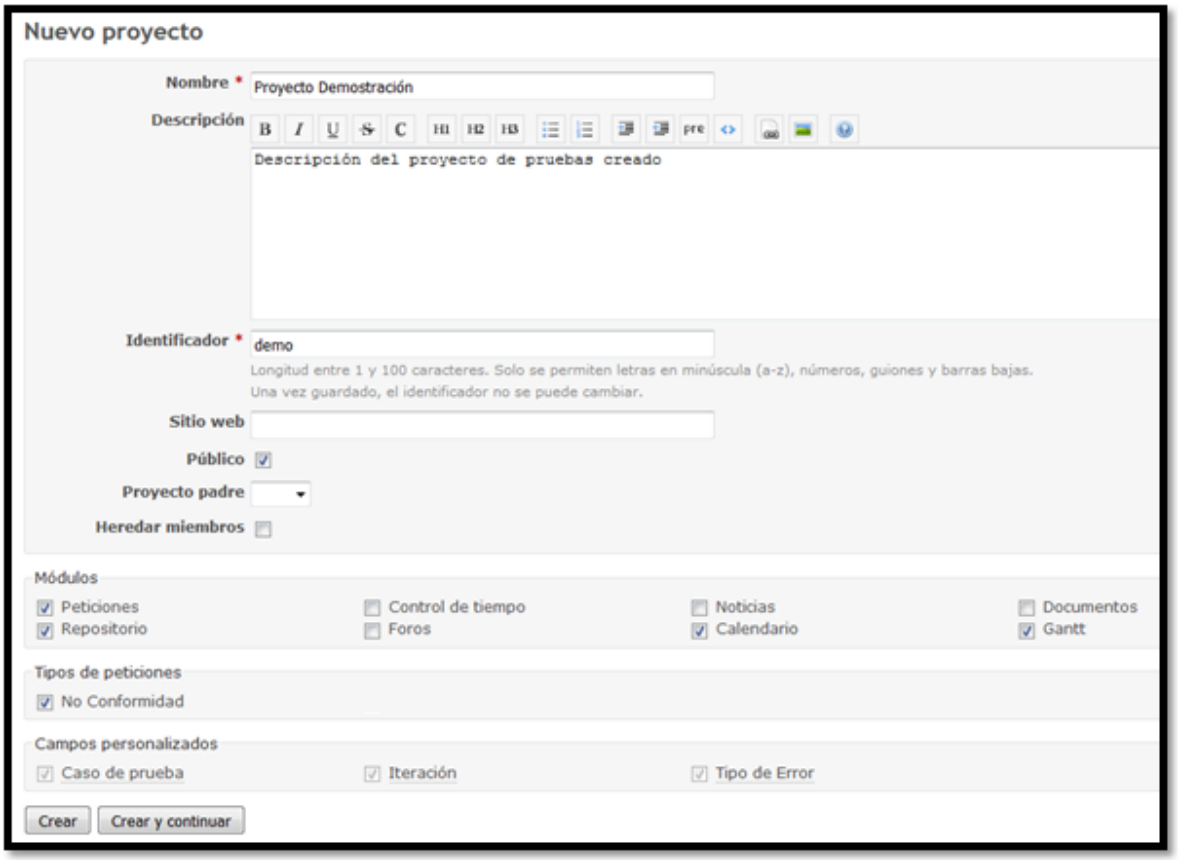
En el menú superior, seleccionar la opción "Proyectos"



Seguidamente elegir la opción "Nuevo proyecto"



Y para finalizar, se debe completar el siguiente formulario:



El único campo que debe de llenarse es:

- Nombre: nombre del proyecto de pruebas, el cual debe coincidir con el nombre del proyecto en Processmaker.

Todos los demás campos ya aparecen seleccionados y llenos, de forma automática.

Para concluir con la creación del proyecto, debe seleccionarse la opción **Crear.** Si se selecciona la opción Crear y continuar, el proyecto será creado, pero se permanecerá en el mismo formulario, para continuar creando proyectos.

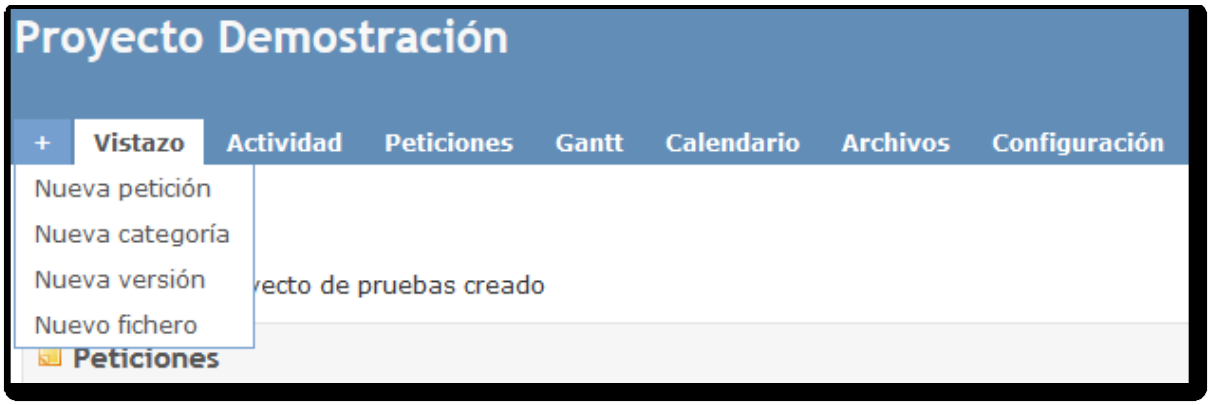
Si el proyecto fue creado correctamente, se observará lo siguiente:

A continuación la tarea que debe ejecutarse es **Registro de Incidencias**.

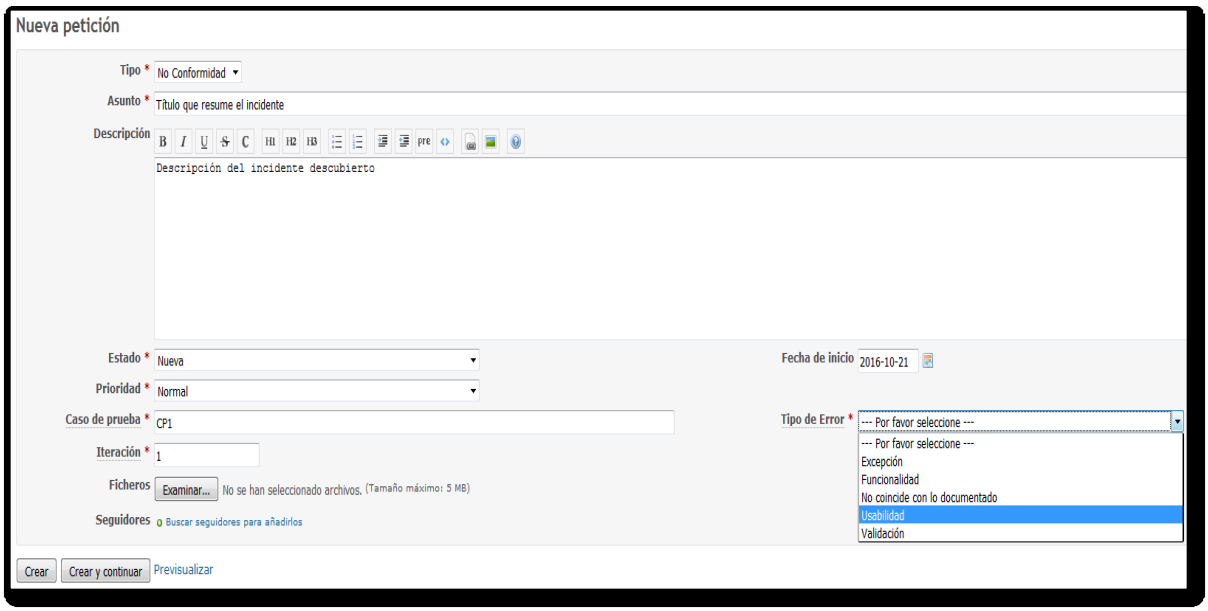
1.8 Registro de Incidencias

Durante la ejecución de los casos de prueba, en el momento en que el sistema ofrezca un resultado, distinto del especificado como "esperado", debe **registrarse un incidente**. Para ello, en Redmine se debe generar una **Petición**, de la siguiente manera:

Dentro del proyecto creado, en el menú superior, seleccione el ícono en forma de "más" y se desplegará un sub-menú.



Al elegir la opción **Nueva petición**, la herramienta mostrará la siguiente pantalla:



En este formulario de **Registro de Incidente**, todos los campos son obligatorios, a excepción de Ficheros y Seguidores. Los campos deben contener la siguiente información:

- Tipo: el único tipo de petición disponible es No Conformidad.
- Asunto: nombre que resume el incidente identificado.
- Descripción: detalle del incidente identificado.
- Estado: el único estado disponible al crear una petición es Nueva.
- Prioridad: puede elegirse entre Baja, Media, Alta, Urgente.
- Caso de Prueba: este campo es de suma importancia, pues debe ingresarse **única y exclusivamente el Id. del Caso de Prueba** que se ejecutó para que sucediera el incidente.
- Iteración: este campo también es de suma importancia y el valor numérico ingresado debe **coincidir con el número de iteración definido en Processmaker**.
- Fecha de Inicio: campo de llenado automático.
- Tipo de Error: puede elegirse entre varios valores, Excepción, Funcionalidad, No coincide con lo documentado, Usabilidad, Validación.

Esta es la definición de cada tipo de error:

Excepción: el sistema carece de robustez y su funcionamiento normal se vio interrumpido.

Funcionalidad: el resultado de la funcionalidad no es el correcto, por ejemplo, el valor no fue guardado o la fórmula no calculó correctamente.

No coincide con lo documentado: el sistema demuestra un comportamiento que no se reconoce como el esperado y tampoco corresponde con lo especificado en la documentación.

Usabilidad: el ejecutante reconoce algún detalle de usabilidad que considera perjudica la utilización de la herramienta.

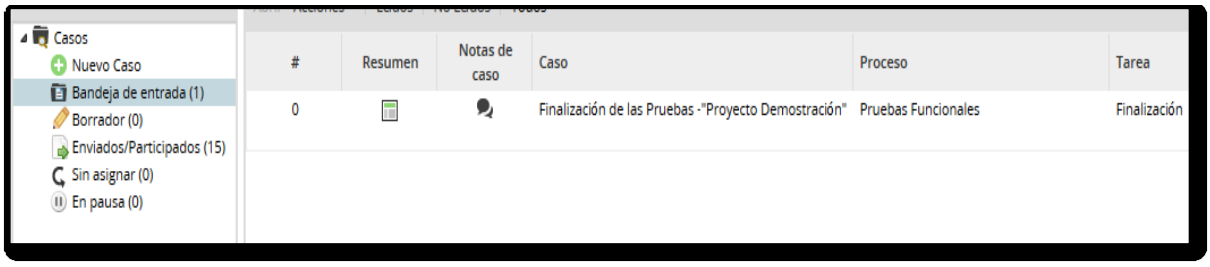
Validación: el sistema debió rechazar un dato de entrada inválido y no fue así, o lo contrario, no aceptó un dato válido.

- Ficheros: posibilidad de adjuntar evidencia (por ejemplo una imagen) del incidente.

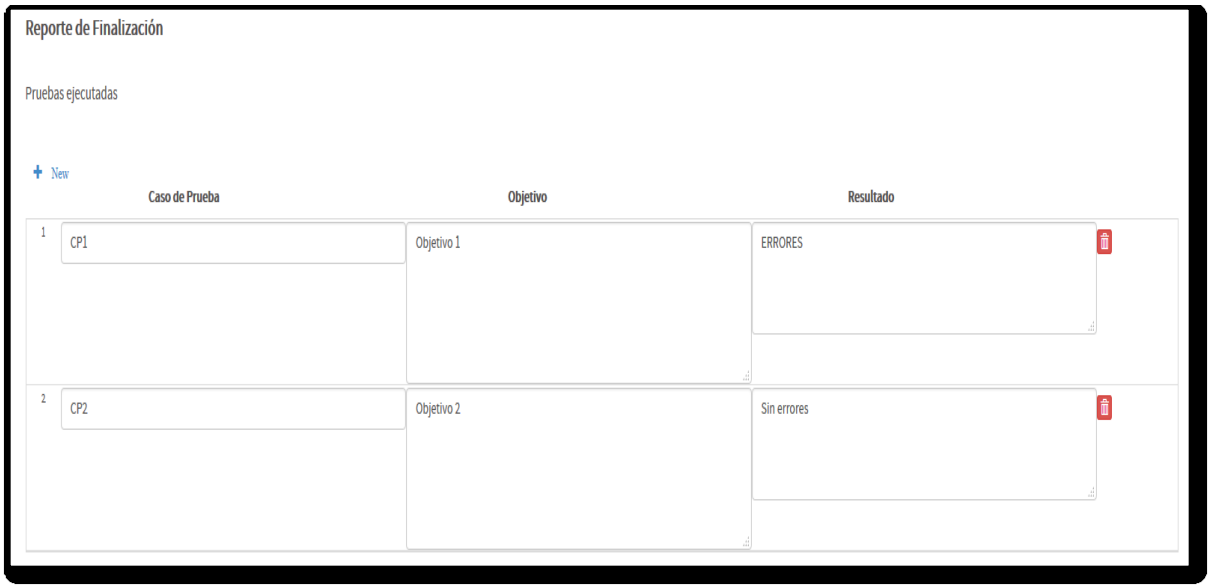
1.9 Finalización de las Pruebas

Cuando se concluye con la ejecución de las pruebas funcionales, el probador notifica al gestor y éste procede a ingresar a Processmaker, con su rol de Líder de Pruebas, para generar el **Reporte de Finalización de las Pruebas.**

El líder, debe de elegir en su Bandeja de Entrada, la tarea Finalización de las Pruebas.



Seguidamente la herramienta mostrará el siguiente formulario:



En la primera sección, se mostrarán de forma automática todos los casos de prueba diseñados y el resultado obtenido, posterior a su ejecución. Si alguno de los casos de prueba posee asociada una No Conformidad, se reportará como con ERRORES.

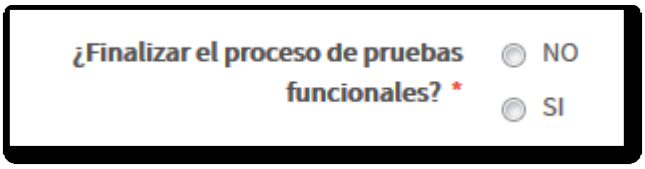
En la siguiente sección del documento, se listan automáticamente las incidencias registradas en Redmine, de la siguiente manera:

Toda la información que se incluyó en el reporte de incidente, es recuperada para el reporte: identificador del Caso de Prueba, nombre con el que se resume al incidente, detalle del incidente, la iteración, el tipo de error y la prioridad.

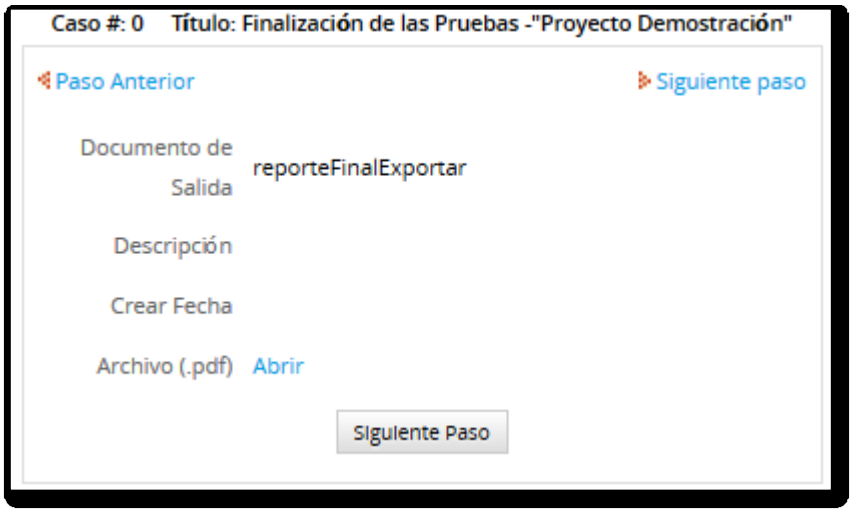
La siguiente sección, también de cálculo automático, muestra el conjunto de métricas recolectadas.

A continuación se dispone de un espacio para determinar si los criterios de completitud fueron logrados y el apego según lo planificado en el Plan de Pruebas.

Finalmente, se consulta al líder de pruebas, si se dan por concluidas las pruebas o no. Si la respuesta es afirmativa se procede a la tarea **Registro de Errores por parte del Usuario.** Si la respuesta es negativa, se retorna a la tarea **Planificación de las Pruebas.**



Cualquiera que sea la opción seleccionada, la herramienta dispone inmediatamente, la exportación del Reporte de Finalización.



1.10 Pruebas de Aceptación

Las Pruebas de Aceptación son llevadas a cabo por uno o varios usuarios seleccionados para evaluar el comportamiento del sistema, en un ambiente controlado. Para ello el líder de pruebas, les facilitará el URL, para el acceso remoto al presente formulario:

Los usuarios podrán ingresar la siguiente información:

- Proyecto: nombre del proyecto de software solicitado.
- Módulo o producto: módulo o producto que está siendo evaluado.
- Usuario: nombre del usuario que está evaluando el producto de software.
- Actividad realizada: funcionalidades ejercitadas.
- Comportamiento observado: respuesta del sistema a las funcionalidades probadas.
- Anexo: posibilidad de anexar evidencia sobre alguna de las respuesta del sistema que pudiera no satisfacer al usuario.

Al concluir el formulario, el usuario debe seleccionar la opción CONTINUAR. De esta manera la información será enviada al flujo de trabajo de Processmaker, específicamente a la Bandeja de Entrada del líder de pruebas.

Con esto concluyen las tareas relativas al proceso de pruebas.

2. Seguimiento del Proyecto

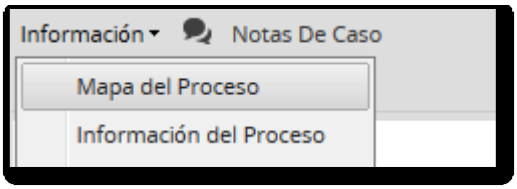
Para el seguimiento del proyecto, el líder puede realizar las siguientes actividades:

i. Revisión del Mapa del Proceso
ii. Revisión de los Formularios Llenados
iii. Revisión de los Archivos Generados o Subidos

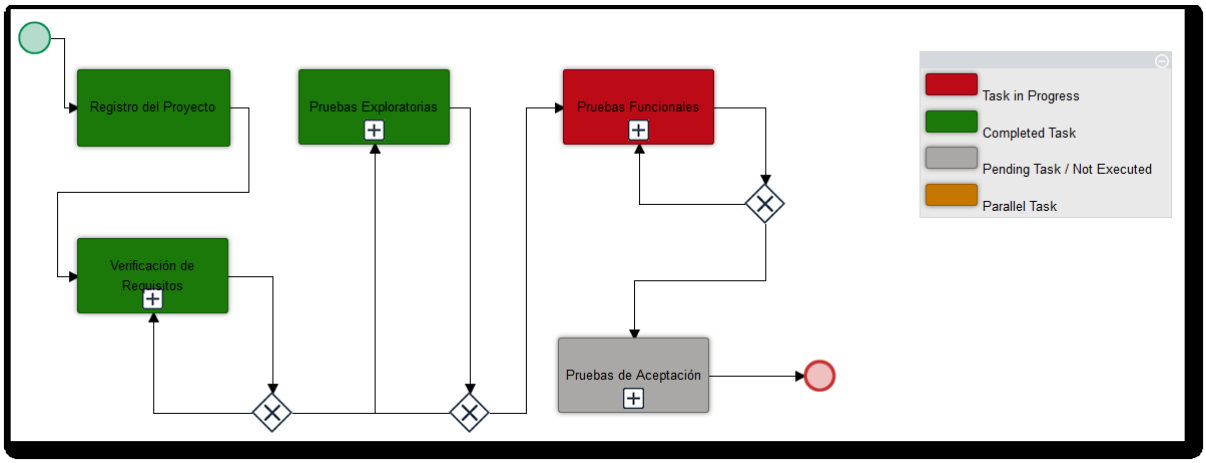
2.1 Revisión del Mapa del Proceso

El líder de pruebas, puede seleccionar los procesos que se están ejecutando y observar qué tareas se han ejecutado, cuáles falta, cuál se está ejecutando y cuáles no se realizaron. Para ello el líder deberá:

Seleccionar en la barra horizontal la opción Información y del menú desplegable, la opción Mapa del Proceso.



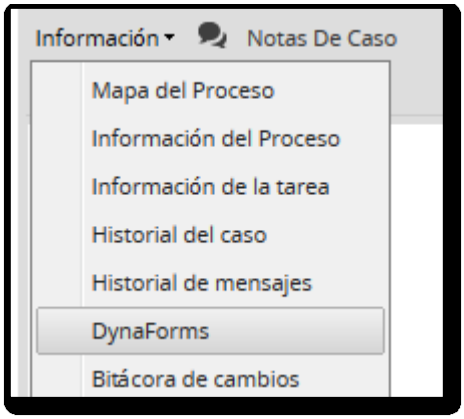
Seguidamente el sistema desplegará un mapa como este:



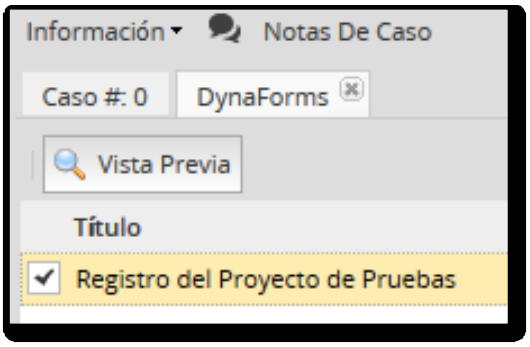
En rojo se resalta la tarea en progreso, el verde las ejecutadas y en gris las pendientes o no ejecutadas.

2.2 Revisión de los Formularios Llenados

De manera similar, para acceder a los formularios que ya fueron completados durante alguno de los procesos implementados por la plataforma, se debe seleccionar el menú Información, y seleccionar la opción *Dynaforms*.

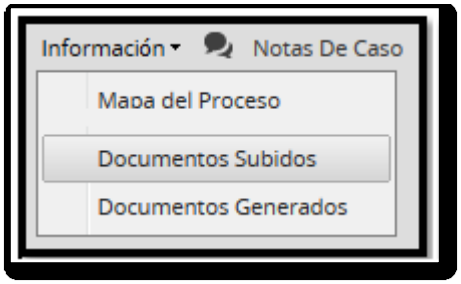


Se desplegará entonces una lista de los formularios dinámicos de Processmaker, y desde allí es posible seleccionar uno y luego elegir la opción **Vista Previa**.

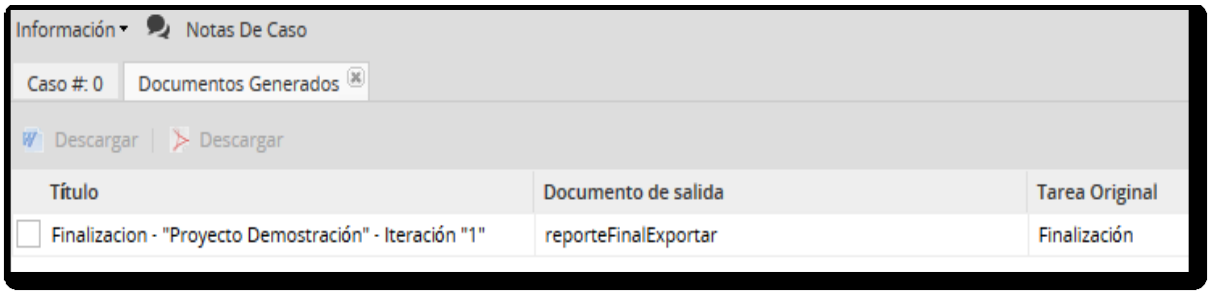


2.3 Revisión de los Archivos Generados o Subidos

De manera similar a las dos actividades anteriores, debe elegirse la opción Documentos Subidos o Documentos Generados, desde el menú que se despliega, al seleccionar Información.



Se mostrará entonces un listado de los documentos generados (evaluaciones, reportes), con la posibilidad de observarlos de nuevo e incluso descargarlos, ya sea en formato DOCX o en PDF.



ANEXO P – CONSECUCIÓN DE LOS OBJETIVOS DE INVESTIGACIÓN

Con el prototipo de plataforma desarrollado, se pretendió enfrentar los 5 problemas más importantes del CI en materia de verificación y validación de software, a saber:

I. No hay ningún tipo de automatización, ni de pruebas ni de procesos.
II. No se sigue ningún estándar internacional para el proceso de pruebas o para su documentación.
III. No se generan reportes o estadísticas sobre el proceso de pruebas.
IV. No existen un documento formal de Especificación de Requerimientos.
V. El seguimiento y control de los proyectos se ve dificultado por la ausencia de la documentación mínima.

1 - ¿Considera usted que la plataforma representa un avance en términos de la automatización de las tareas de verificación y validación de software?

2 - Los procesos y documentos diseñados durante la investigación, están inspirados en los estándares IEEE 29119 e IEEE 1012. ¿Considera usted que fueron éstos ajustados satisfactoriamente para su ejecución y utilización en el Centro de Informática?

3 - ¿Los reportes incluidos y métricas calculadas en el prototipo de plataforma resultan suficientes y significativas para la Unidad de Calidad y Mejora Continua del Centro de Informática?

4 - ¿Las distintas herramientas generadas para la evaluación de los requerimientos constituyen un apoyo para mejorar la especificación de los requerimientos en el Centro de Informática?

5 - ¿Considera usted que con la utilización de la suite BPM (Processmaker), se facilita el seguimiento y control de las tareas y actividades llevadas a cabo en la Unidad de Calidad y Mejora Continua?

Una de las metas propuestas en la investigación fue la utilización de herramientas de código abierto para la implementación de las funcionalidades requeridas por el prototipo de plataforma.

6 - ¿Considera que las herramientas seleccionadas para integrar la plataforma, luego de familiarizarse con ellas, son de utilidad tanto para los procesos seleccionados como para futuros proyectos dentro del Centro de Informática?

Cuando se planteó el problema de investigación, las actividades de verificación y validación de software elegidas para apoyar con el prototipo fueron: la verificación de requerimientos a los proyectos y las pruebas funcionales de software.

7 - Con el trabajo de investigación llevado a cabo en el Centro de Informática, ¿es posible evaluar la calidad de la especificación de requerimientos de software? ¿Es posible determinar si los requerimientos especificados son completos, consistentes, verificables, modificables, inequívocos, correctos, priorizables y trazables?

8 - ¿El prototipo de plataforma acompaña en su totalidad el proceso de pruebas funcionales de software diseñado siguiendo las recomendaciones del estándar IEEE 29119?

9 - ¿El prototipo de plataforma permite el registro del proyecto de pruebas de software y su identificación inequívoca de otros proyectos?

10 - ¿El prototipo de plataforma posibilita la especificación de un Plan de Pruebas en el que se define el alcance de las pruebas, lo que se probará y lo que no, los criterios de completitud y los recursos disponibles para ejecutar las pruebas?

11 - ¿El prototipo de plataforma cuenta con un espacio para diseñar los casos de prueba, para especificar los datos de entrada necesarios y los requerimientos de ambientación?

12 - ¿El prototipo de plataforma dispone de una herramienta mediante la cual acceder a los *scripts* de prueba (en caso de desarrollarse éstos)?

13 - ¿El prototipo de plataforma permite registrar los errores identificados? Además, ¿construye un reporte especificando las pruebas satisfactorias, las que generaron inconformidades y un conjunto de métricas del proceso de pruebas?

14 - ¿El prototipo de plataforma permite la creación y la administración de todos los documentos que forman parte del proceso de pruebas funcionales de software?

Adicionalmente al acompañamiento de las tareas de verificación y validación de software,

15 - ¿El prototipo de plataforma posibilita el seguimiento de los procesos de prueba, en cualquiera de sus etapas y desde cualquier lugar?

16 - ¿Las herramientas son accedidas únicamente con previa autenticación y de ésta dependen los privilegios y capacidades del usuario?

Finalmente,

17 ¿Considera que el prototipo de plataforma desarrollado, puede apoyar al personal del Centro de Informática, en sus tareas de verificación y validación de los proyectos de software?