



UNIVERSIDAD CATÓLICA ANDRÉS BELLO
VICERRECTORADO ACADÉMICO
ESTUDIOS DE POSTGRADO
ÁREA DE INGENIERÍA
Postgrado en Sistemas de Información

Trabajo de Grado de Maestría

**MODELO DE REFERENCIA PARA LA SELECCIÓN DE HERRAMIENTAS DE PRUEBAS
COMO SOPORTE AL PROCESO DE DESARROLLO DE SOFTWARE EN PYMES
VENEZOLANAS**

presentado por
Mendoza Méndez, María Gabriela;
para optar al título de
Magíster en Sistemas de Información

Tutor
Pérez de Ovalles, María Angélica

Caracas, Julio 2010

UNIVERSIDAD CATÓLICA ANDRÉS BELLO
VICERRECTORADO ACADÉMICO
ESTUDIOS DE POSTGRADO
ÁREA DE INGENIERÍA
Postgrado en Sistemas de Información

**MODELO DE REFERENCIA PARA LA SELECCIÓN DE HERRAMIENTAS DE PRUEBAS
COMO SOPORTE AL PROCESO DE DESARROLLO DE SOFTWARE EN PYMES
VENEZOLANAS**

Autor: María Gabriela Mendoza Méndez

Tutor: María Angélica Pérez de Ovalles

Fecha: Julio 2010

RESUMEN

El desarrollo de software debe estar enfocado en la imperiosa realidad de garantizar calidad tanto en todo su proceso como en sus resultados (producto) para así traducirse en una estrategia empresarial y lograr a través de éste, las ventajas competitivas en un entorno globalizado. Estar consciente de esta realidad en las empresas desarrolladoras de software, constituidas en Venezuela principalmente por PYMES, se traducirá en dar formalidad a la disciplina de pruebas como apoyo a este proceso de desarrollo de software. Sin embargo, esta formalidad esperada en la disciplina de pruebas y el uso de herramientas automatizadas es escasa. El objetivo de la presente investigación, es proponer un modelo de referencia dirigido a PYMES venezolanas que permita apoyar la selección de herramientas de prueba adaptada a las características de sus proyectos y del contexto organizacional, para asegurar la calidad de sus productos. Los resultados de esta investigación se lograron siguiendo cada uno de los pasos y/o actividades enmarcados en el Framework metodológico del LISI y se cuantifican en: a) Modelo conceptual que abarca las principales definiciones de las pruebas de software, resultado de un extenso análisis de la bibliografía y las experiencias con herramientas de pruebas existentes en el mercado; b) Árbol de clasificación de herramientas de pruebas automatizadas con su correspondiente instanciación (herramientas para el nivel de pruebas del sistema del tipo no funcionales: Desempeño y carga, bajo un enfoque de Caja Negra; c) Modelo de referencia propuesto para seleccionar herramientas de pruebas que apoyen el proceso de desarrollo de software, producto de aportes al modelo de calidad sistémica MOSCA para el producto; d) Guía para aplicar el modelo propuesto a herramientas de prueba existentes en el mercado tecnológico, previa selección de acuerdo al árbol de clasificación. El método de evaluación arrojado por DESMET fue “Análisis de características por Estudio de Caso”, por lo que el modelo se aplicó en cuatro estudios de casos resultando sólo un caso con nivel de calidad Avanzado. Luego de la experiencia obtenida, el modelo fue evaluado iniciando desde sus características generales hasta sus métricas, proponiendo algunas modificaciones.

PALABRAS CLAVE

Desarrollo de software, calidad del software, disciplina de pruebas, herramientas de prueba, Pequeñas y Medianas Empresas (PYMES).

DEDICATORIA

A mi amado Dios,
gracias por permitirme culminar con éxito otro logro más de mi vida!

Todo lo que soy te lo debo a ti, quien
*“...por su amor, gracia y misericordia, le da recibimiento
a su hija pródiga, con anillo, ropas nuevas, fiesta y alegría.”*
(San Lucas 15:11-32)

AGRADECIMIENTOS

Agradezco a mi familia: padres, hermanas, sobrinos por ser personas fundamentales en mi vida, por reconocer mis logros, apoyar mis decisiones y siempre darme palabras de aliento y ánimo para seguir adelante.

Gracias a mi esposo bello por su amor, paciencia y apoyo en la culminación de esta meta... tu paz me apacigua.

Un agradecimiento muy especial a mi tutora, Prof. María Angélica Ovalles sin olvidar a mis profesores co-tutores del LISI (Edumilis Méndez y Luis E. Mendoza), por su dedicación, profesionalismo, paciencia y por enseñarme a no conformarme.

A mis amigas, amigos y demás familiares por sus palabras de apoyo y expresiones de ayuda incondicional.

A mis jefes laborales, por su paciencia y apoyo en el transitar por este camino.

Y finalmente agradezco a mi segunda casa estudiantil, la UCAB, a mis profesores y compañeros de estudios que promovieron un espíritu de excelencia académica.

INDICE GENERAL

RESUMEN	1
DEDICATORIA	2
AGRADECIMIENTOS	3
INDICE GENERAL	4
INDICE DE TABLAS	6
INDICE DE FIGURAS	7
INTRODUCCIÓN	8
CAPÍTULO I: EL PROBLEMA DE INVESTIGACIÓN	11
1.1 PLANTEAMIENTO DEL PROBLEMA	11
1.2 INTERROGANTES DE LA INVESTIGACIÓN	12
1.3 OBJETIVOS DE LA INVESTIGACIÓN	13
1.4 JUSTIFICACIÓN E IMPORTANCIA DE LA INVESTIGACIÓN	13
1.5 ALCANCES Y LIMITACIONES	15
CAPÍTULO II: MARCO TEORICO	17
2.1 ANTECEDENTES DE INVESTIGACIÓN	17
2.2 MARCO ORGANIZACIONAL	20
2.3 BASES TEÓRICAS	27
I. Calidad del Software	28
II. Aseguramiento de la Calidad del Software	33
III. Pruebas del Software	40
IV. Disciplina de Pruebas	43
V. Modelos de Calidad	52
VI. Herramientas de Pruebas	55
2.4 BASES ÉTICAS Y LEGALES	57
CAPÍTULO III: MARCO METODOLÓGICO	59
3.1 TIPO DE INVESTIGACIÓN	59
3.2 DISEÑO DE LA INVESTIGACIÓN	60
<i>Enfoque basado en objetivos, preguntas y métricas – GQM</i>	66
3.3 TÉCNICAS E INSTRUMENTOS DE RECOLECCIÓN DE DATOS	67
CAPÍTULO IV: DESARROLLO DE LA PROPUESTA	68
4.1 MODELO CONCEPTUAL	68
4.2 CLASIFICACIÓN PROPUESTA PARA LAS HERRAMIENTAS DE PRUEBA	73
4.3 ASPECTOS TOMADOS DE MOSCA PARA LA INSTANCIACIÓN	76
4.4 DESARROLLO DE LA PROPUESTA (ASPECTOS AGREGADOS A MOSCA PARA LA INSTANCIACIÓN)	79
CAPÍTULO V: EVALUACION DE LA PROPUESTA	89
5.1 OBJETIVOS DE LA EVALUACIÓN	89
5.2 METODOLOGÍA SEGUIDA	89
5.3 PROCESO DE EVALUACIÓN	89

<i>a) Análisis de Contexto</i>	<i>89</i>
<i>b) Aplicación del método DESMET.....</i>	<i>91</i>
<i>Método de Análisis de Características por Estudio de Caso</i>	<i>93</i>
<i>c) Evaluación del Modelo de referencia basado en características para la selección de herramientas de prueba</i>	<i>94</i>
CAPÍTULO VI: ANÁLISIS E INTERPRETACIÓN DE RESULTADOS	101
6.1 EVALUACIÓN DEL MODELO EN LOS ESTUDIOS DE CASO	101
6.2 EVALUACIÓN DE LAS CARACTERÍSTICAS GENERALES.....	110
6.2.1 Evaluación de las categorías del modelo	110
6.2.2 Evaluación de las características del modelo	111
6.3 EVALUACIÓN DE LAS MÉTRICAS DEL MODELO.....	113
CAPÍTULO VII: CONCLUSIONES Y RECOMENDACIONES	101
7.1 CONCLUSIONES	116
7.2 RECOMENDACIONES.....	118
ANEXO I: Modelo de Calidad MOSCA	120
ANEXO II: Aspectos de la categoría FUNCIONALIDAD tomadas de MOSCA	121
ANEXO III: Aspectos de la categoría USABILIDAD tomadas de MOSCA.....	122
ANEXO IV: Aspectos de la categoría FIABILIDAD y EFICIENCIA tomadas de MOSCA	123
ANEXO V: Descripción de algunas métricas aportadas por el autor a MOSCA en el contexto del modelo propuesto	124
ANEXO VI: Evaluación detallada de las Herramientas de prueba seleccionadas de acuerdo al Modelo propuesto.....	126
ANEXO VII: Resultados Evaluación de Características generales del modelo propuesto	133
ANEXO VIII: Resultados Evaluación de Métricas del modelo propuesto	135
REFERENCIAS BIBLIOGRÁFICAS	139

INDICE DE TABLAS

Tabla 1- <i>Tabla Comparativa entre defensores del Aseguramiento de la Calidad</i> (Fuente: Elaboración propia).....	39
Tabla 2- <i>Modelo de Calidad según ISO 9126</i> . Fuente: Elaboración propia	54
Tabla 3 - <i>Nivel de calidad del producto con respecto a las categorías satisfechas para el producto</i> . FUENTE: Mendoza et al. (2005).	55
Tabla 4 - <i>Fases del Método Investigación Acción</i> . Fuente: Elaboración Propia.	63
Tabla 5 - <i>Sub-características, sub-sub-características y métricas agregadas en la característica “Ajuste a los propósitos” de la categoría Funcionalidad de MOSCA</i> . Fuente: Elaboración Propia.....	85
Tabla 6- <i>Sub-características, sub-sub-características y métricas agregadas en la característica “Precisión” de la categoría Funcionalidad de MOSCA</i> . Fuente: Elaboración Propia.....	86
Tabla 7 - <i>Sub-característica, sub-sub-características y métricas agregadas en la característica “Interoperabilidad” de la categoría Funcionalidad de MOSCA</i> . Fuente: Elaboración Propia.....	86
Tabla 8 - <i>Sub-característica, sub-sub-características y métricas agregadas en la característica “Seguridad” de la categoría Funcionalidad de MOSCA</i> . Fuente: Elaboración Propia.	87
Tabla 9 - <i>Métricas agregadas en la categoría Usabilidad de MOSCA</i> .Fuente: Elaboración Propia.....	87
Tabla 10 - <i>Condiciones favorables presentes y no presentes en cada uno de los métodos propuestos por DESMET</i> . Fuente: Elaboración Propia.....	93
Tabla 11- <i>Características Generales a evaluar para el modelo propuesto</i> . Fuente: Sosa (2005).....	97
Tabla 12 – <i>Características de las Métricas a evaluar el modelo propuesto</i> . Fuente: Sosa (2005).....	97
Tabla 13 - <i>Herramientas de prueba del mercado seleccionadas para cada estudio de caso</i> . Fuente: Elaboración Propia.	99
Tabla 14 - <i>Evaluación del modelo propuesto en el Proyecto Piloto</i> . Fuente: Elaboración Propia.....	102
Tabla 15 – <i>Nivel de calidad del producto propuesto respecto a categorías satisfechas para el producto adaptado al modelo propuesto</i> . Fuente: Adaptado de Mendoza et al. (2005).....	108
Tabla 16 - <i>Nivel de calidad de los estudios de caso respecto a categorías satisfechas para el producto adaptado al modelo propuesto</i> . Fuente: Elaboración propia.	109

INDICE DE FIGURAS

Figura 1 – <i>Definición de pequeñas y medianas empresas en Venezuela según decreto</i> .	21
Figura 2- <i>Tamaño promedio de las empresas de la Industria Nacional del Software</i>	23
Figura 3- <i>Distribución geográfica de empresas desarrolladoras de software en Venezuela</i>	23
Figura 4 - <i>Tipos de aplicaciones que produce la Industria Nacional de Software</i> .	24
Figura 5 - <i>Necesidades de actualización en procesos de desarrollo de software</i>	25
Figura 6 - <i>Necesidades de capacitación en procesos de gestión de proyectos de software</i>	25
Figura 7 - <i>Utilización de algunas herramientas de apoyo durante el desarrollo de software</i>	26
Figura 8- <i>Procesos y Componentes para el Aseguramiento de la Calidad</i>	36
Figura 9 - <i>Modelo de Proceso de MSF para el desarrollo de aplicaciones</i> , Fuente: (Microsoft, 2003).....	47
Figura 10- <i>Fases del ciclo de vida del desarrollo de software</i>	49
Figura 11 - <i>Vista general del Proceso Unificado de Rational</i>	50
Figura 12 - <i>Framework Metodológico para la de investigación</i>	62
Figura 13- <i>Modelo Conceptual asociado a los Casos de Prueba</i>	69
Figura 14 - <i>Modelo Conceptual asociado a los Tipos, Niveles y Técnicas de Prueba</i>	70
Figura 15- <i>Modelo Conceptual asociado a las Herramientas de Pruebas y el Desarrollo de Software</i>	71
Figura 16 - <i>Modelo Conceptual asociado al Aseguramiento de la Calidad</i>	71
Figura 17- <i>Modelo Conceptual propuesto integrado</i>	72
Figura 18 - <i>Árbol de clasificación de herramientas de prueba automatizadas</i>	74
Figura 19 - <i>Árbol instanciado de ajuste a los propósitos para herramientas de prueba elegidas por el investigador</i>	76
Figura 20 - <i>Instanciación de MOSCA para el modelo de referencia para la selección de herramientas de Prueba</i>	81
Figura 21 - <i>Resultados evaluación Características de la Categoría “Funcionalidad” en Estudio de casos</i>	102
Figura 22 - <i>Resultados evaluación Características de la Categoría “Usabilidad” en Estudio de casos</i>	104
Figura 23 - <i>Resultados evaluación Características de la Categoría “Fiabilidad” en Estudio de casos</i>	105
Figura 24 - <i>Resultados evaluación Características de la Categoría “Eficiencia” en Estudio de Casos</i>	105
Figura 25 – <i>Resultados de la evaluación de los estudios de caso en cada categoría del modelo propuesto</i>	106
Figura 26- <i>Resultados de la evaluación de “Categorías” del modelo propuesto</i>	110
Figura 27 - <i>Resultados de la evaluación de “Características” de la categoría Funcionalidad del modelo propuesto</i>	111
Figura 28 - <i>Resultados de la evaluación de “Características” de la categoría Usabilidad del modelo propuesto</i>	112
Figura 29 - <i>Resultados de la evaluación de “Características” de la categoría Fiabilidad y Eficiencia del modelo propuesto</i>	112
Figura 30 - <i>Resultados de la evaluación de “Métricas” por Categoría del modelo propuesto</i> . .	113

INTRODUCCIÓN

El desarrollo de software en empresas especializadas para tal fin debe estar enfocado en la imperiosa realidad de garantizar calidad tanto en todo su proceso como en sus resultados (producto) para así traducirse en una estrategia empresarial y lograr a través de éste, las ventajas competitivas en un entorno altamente globalizado.

Las empresas desarrolladoras de software, para garantizar calidad en sus productos deben realizar pruebas del software construido. Esto se logra a través de hacer énfasis en la disciplina de Pruebas dentro del proceso de desarrollo de software, práctica que generalmente se traduce en invertir entre el 30% y 50% del costo total del software en pruebas. Sin embargo el tema de las pruebas no está bien formalizado ni justificado dentro del proyecto de software, por lo que invertir esfuerzo y costos en esta disciplina, es casi nula o cuando está presente, se estiman en dimensiones de tiempo/costo no apropiadas ni ajustadas a la realidad.

Bajo esta premisa, las pruebas tienen una gran importancia y representan un gran volumen de actividades ya que se sugiere que en un proyecto de desarrollo de software al menos el 75% de las pruebas sean automatizadas y el resto manual; esto evidencia que la estrategia de automatizar pruebas es una decisión primordial. La automatización de las pruebas se logra a través del uso de herramientas de pruebas automatizadas.

Existen numerosas herramientas de pruebas en el mercado tecnológico con propósitos tan diversos y en algunos casos, tan distantes, que tener criterios válidos para su selección se vuelve una actividad compleja. En todo caso, el primer paso es definir el propósito para lo cual se requiere y el alcance que las mismas tendrán en el proceso de desarrollo de software.

Uno de los principales problemas a los que se enfrenta la calidad del software es el siguiente: ¿Es realmente posible encontrar un conjunto de propiedades en un software que sea una indicación de su calidad?. Para dar respuesta a esta pregunta aparecen los Modelos de Calidad o Modelo de referencia, que no son más que las directrices o puntos de referencia para imitarlo o reproducirlo.

Por lo antes expuesto, en este trabajo de grado de maestría se propone un Modelo de referencia para la selección de herramientas de prueba que permitan apoyar el proceso de desarrollo de software específicamente en la disciplina de pruebas para PYMES desarrolladoras de software en Venezuela, convirtiéndose en referencia obligada ante un proceso de selección de herramientas de prueba automatizada, el cual fue ampliamente evaluado.

Este modelo surgió como producto de cumplir con cada una de las actividades y/o fases enmarcadas dentro del Framework metodológico propuesto por el Laboratorio de Investigación en Sistemas de Información (LISI-USB) dentro del un tiempo planificado y finito en donde se realizó un análisis exhaustivo del tema, producto de revisiones constantes preliminares y mesas de trabajo del alumno, con en el trabajo mancomunado de sus asesores técnicos y metodológicos, para una vez definido teóricamente, llevar este conocimiento a la práctica para crear el modelo de referencia de selección de herramientas de pruebas.

Cabe destacar que por la amplitud de los tipos, niveles y enfoques que tienen las herramientas de prueba existentes en el mercado, se hizo necesario trabajar con solo una muestra de herramientas de prueba obtenida de la instanciación del árbol de clasificación de herramientas propuesto por considerarse uno de los más importantes tipos de pruebas que debe realizar una empresa desarrolladora de software.

Sin embargo, los objetivos logrados se traducen en importantes beneficios para PYMES desarrolladoras de software pues por una parte, el modelo propuesto hace un extenso recorrido por los atributos mandatorios que debe tener una herramienta de prueba garantizando que la selección de herramientas sea suficientemente robusta y precisa en sus resultados siempre y cuando se cumplan con las condiciones de aplicación del modelo, y por la otra, proporciona una guía completa para la aplicación del modelo con efectividad comprobada que también puede usarse como guía para construir una herramienta de prueba adaptada a las necesidades de la empresa que lo desarrolle.

Finalmente, este proyecto se estructura en base a los siguientes seis capítulos:

- **El problema de investigación**, donde se introduce y plantea el tema en estudio, su importancia y justificación de la investigación. Igualmente, se definen los

objetivos, listando los objetivos específicos para la realización del tema en estudio y que sustentan el objetivo general de la investigación que también es descrito.

- El siguiente capítulo, **Marco Teórico** se traen a colación investigaciones previas enmarcadas dentro del área y del tema en estudio que podrán ser utilizadas como antecedentes para dar base a la presente investigación. Se presentan exhaustivamente las bases teóricas por una parte, y por la otra, las bases éticas y legales que sustentan la propuesta. De igual forma se plantea el marco organizacional que explica hacia cual sector productivo se dirige la presente investigación, PYMEs desarrolladoras de Software en Venezuela.
- **Marco Metodológico**, es el capítulo donde se plasma el marco de trabajo que rige la investigación y se explica cada una de sus fases, así como el tipo de investigación y las técnicas e instrumentos utilizados a lo largo de la investigación.
- En el capítulo IV **Desarrollo de la propuesta** se exponen los aportes al conocimiento de los sistemas de información que ha logrado el autor iniciando desde el modelo conceptual lleno de definiciones que giran en torno a las Pruebas, el árbol de clasificación de pruebas hasta llegar al modelo propiamente dicho (con los aspectos tomados de MOSCA y los que fueron propuestos).
- La **Evaluación de la propuesta** describe paso a paso cómo se definió la estrategia de evaluación del modelo dentro del contexto de aplicación y a través del método de evaluación DESMET para constituirse en el insumo del siguiente capítulo VI **Análisis e Interpretación de resultados**, donde se muestra primero la evaluación del modelo en los cuatro estudios de caso seleccionados y luego la evaluación del modelo en sí, desde sus características generales hasta las métricas que la conforman. Los resultados de la evaluación en forma detallada se presentan en la sección de Anexos.
- El último capítulo **Conclusiones y Recomendaciones** muestran los logros obtenidos en esta investigación en forma resumida y las sugerencias del autor para futuros trabajos de investigación, respectivamente.

CAPÍTULO I: EL PROBLEMA DE INVESTIGACIÓN

1.1 Planteamiento del Problema

Dentro del marco de los Sistemas de Información (SI) son múltiples las aplicaciones que pueden desarrollarse para las empresas en sus diversos ámbitos de acción. Actualmente las Tecnologías de Información (TI) impactan cualquier ámbito del desarrollo tecnológico y científico del ser humano; su aplicación a numerosas áreas cotidianas se ha vuelto indispensable, aportando beneficios al poder administrar, entre otras cosas el conocimiento y su gestión, que actualmente es considerado parte del capital más importante de las organizaciones. Dentro de estas organizaciones se incluye la industria del software en Venezuela, donde las pequeñas y medianas empresas (PYMES) juegan un rol importante para el desarrollo del país (Reyes, 2005) y cuyos principales factores de producción son el talento humano, las capacidades de producción, la calidad y la innovación.

Estudiar la garantía de la calidad en el desarrollo de software, tanto en su proceso de desarrollo como en el producto resultante, se ha constituido en un tópico de importancia para la Ingeniería de Software. La calidad esperada en el producto puede determinarse a través de una *disciplina* definida por RUP (2005) como una colección de actividades correspondidas que a su vez se relacionan con un “motivo de preocupación importante” dentro del proyecto total donde estén enmarcadas.

Algunas investigaciones como Rivero et al. (2007), Methodius (2005), Sánchez et al. (2001) reflejan de manera general una realidad latente de que las empresas desarrolladoras de software en Venezuela, actualmente adolecen de estrategias, métodos y modelos que afiancen sus procesos y las hagan competitivas dentro de su entorno nacional con miras al ámbito internacional. De forma específica, las escasas investigaciones nacionales enfocadas en la disciplina de pruebas dentro del proceso de desarrollo de software nacional hacen aún más difícil que dichas empresas tengan una estrategia que les permita realizar una evaluación de las diversas herramientas que presenta el mercado en cuanto a pruebas se refiere; razón por la cual, la decisión de optar por una herramienta que satisfaga las necesidades del proyecto de software, se tornan con frecuencia imprecisas. En la mayoría de los casos, esta elección se ve restringida a escoger herramientas que no son las más acordes a la naturaleza del

proyecto, y en el peor de los casos, se niega el uso de estas herramientas imposibilitando su cooperación en el proceso de desarrollo de software. Esta carencia en el uso de herramientas de pruebas igualmente refleja la poca importancia y la informalidad que a lo largo del proyecto de software se le da a quien proporciona las directrices para obtener calidad de software, la Disciplina de Pruebas (pruebas de software o sistema); que debe ser vista como la unión de roles, actividades, artefactos y su orden interno más que cómo un simple requisito para la aprobación final del producto.

Involucrarse en el tema de las *pruebas del software* con todas sus etapas es una realidad que debe estar presente en cualquier empresa que desee adentrarse en el mundo del desarrollo del software y, por supuesto, la calidad. En la actualidad existen variadas herramientas de automatización y planificación del esfuerzo de pruebas que facilitan el desarrollo de ésta disciplina. Dichas herramientas son diversas y adaptadas a ciertas necesidades, por lo que se torna difícil decidir a ‘ciencia cierta’ cuál de las mismas pueden aplicarse en una u otra situación. Y es que la dificultad de la elección de la mejor herramienta está determinada por diversos puntos de atención, como lo son: la naturaleza del proyecto, la tecnología utilizada en la construcción del software, las necesidades de pruebas, los potenciales clientes a los que está dirigido y hasta la empresa que desarrolla el proyecto (estructura física y financiera) entre otros aspectos relevantes.

En la próxima sección se indican las interrogantes de investigación que rigen esta investigación.

1.2 Interrogantes de la investigación

Ante el problema planteado, se enuncian las siguientes preguntas que rigen la investigación y clarifican el propósito de la misma.

- ¿Cuáles son las características relevantes de las herramientas de pruebas que apoyan y soportan la disciplina de pruebas en todo el proceso de desarrollo de software para las PYMEs?
- ¿Disponen las PYMEs venezolanas desarrolladoras de software de métodos para la evaluación de herramientas de pruebas de software que de apoyo a la disciplina de pruebas y aseguren la calidad de sus productos?

- ¿Son aplicables y adaptables los resultados de la investigación a las necesidades de las PYMEs desarrolladoras de software en Venezuela?

Para dar respuesta a estas interrogantes y en base a las mismas, se formulan los objetivos que rigen esta investigación en la próxima sección.

1.3 Objetivos de la investigación

Objetivo general:

Desarrollar un Modelo de referencia dirigido a PYMES venezolanas desarrolladoras de software que permita apoyar la selección de herramientas de pruebas adaptado a las características de sus proyectos y del contexto organizacional.

Objetivos específicos:

- Formular un modelo conceptual que soporte el trabajo de investigación, partiendo de puntos clave como Calidad de software, Aseguramiento de la Calidad del Software y Disciplina de Pruebas.
- Proponer un modelo de referencia que especifique las características que debe poseer la herramienta que soporte la disciplina de pruebas enmarcadas dentro de las características de las PYMES Venezolanas.
- Aplicar el modelo adaptado a un estudio de caso de PYMEs.
- Evaluar el modelo de referencia.

1.4 Justificación e importancia de la investigación

El proyecto planteado realiza un estudio de todo lo concerniente a la disciplina de Pruebas del software en el ciclo de desarrollo, dando importancia al aseguramiento de la Calidad del software como promotor y garante de calidad en el proceso de desarrollo del software y a la selección de herramientas de pruebas que soporten esta disciplina.

El modelo de referencia propuesto para la selección de las herramientas de pruebas de software que ofrece el mercado tecnológico, está dirigido a las organizaciones desarrolladoras de software y prioritariamente hacia las PYMEs venezolanas, las cuales están obteniendo popularidad y aceptación dentro de este tiempo de cambio y transformación que transita nuestro país (Sánchez et al., 2001). Dicho

modelo de referencia será validado y documentado de acuerdo a las normas establecidas en la Ingeniería del Software, con dos propósitos especiales. Por una parte, crear conciencia en las PYMEs de la realización de tan importante actividad en el desarrollo de sus sistemas/software y, que a la vez, apalanquen el desarrollo de sus actividades. Por otra parte, proporcionar a las PYMEs orientaciones necesarias para seleccionar las herramientas que más le favorezcan, en pro de fortalecer el desarrollo de software nacional con los máximos niveles de calidad, creando las directrices para buscar lograr una competitividad entre empresas locales, siempre en vías de incursionar en el ámbito internacional. He aquí la importancia de la investigación.

La Ingeniería del Software abarca actividades macro de suma importancia en el desarrollo del mismo, dentro de la que se encuentran las pruebas del software (Pressman, 2005) la cual vela, entre otros aspectos, porque exista la calidad esperada en el producto resultante, esto es, el fiel cumplimiento de los requerimientos acordados por el cliente. El Standish Group (2003), revela cifras importantes sobre la deficiencia en la entrega de los proyectos de tecnologías de información para el año 2002 en Estados Unidos, por lo que no es de extrañarse que estos mismos problemas se afronten en nuestro país. Prueba de ello lo refleja Methodius (2005), al afirmar que la mayoría de las empresas e instituciones públicas venezolanas desarrollan software usando procesos ad-hoc o métodos y técnicas inapropiadas u obsoletas, muchas de las cuales llevan utilizándose más de 25 años.

Las escasas estrategias e investigaciones que se han realizado en Venezuela, dirigidas principalmente a PYMES desarrolladoras de software en Venezuela, hacen que las mismas se caractericen por no tener formalidad en su proceso de desarrollo de software y por ende, en la práctica de la disciplina de pruebas, lo cual las hace poco competitivas en el entorno donde se desenvuelven. Un indicador importante de esta crítica situación es que, hasta la presente fecha, sólo dos empresas venezolanas han sido certificadas con el modelo CMM; el modelo de evaluación de la madurez de los procesos de desarrollo de software de mayor aplicación en el mundo (Methodius, 2005). Otro aspecto de relevancia, lo manifiesta el estudio realizado por Rivero et al. (2007) donde se demuestra la gran necesidad de actualización en el tema de pruebas de software que evidencia el capital humano en la industria del software, aspecto y campo de acción que se debe reforzar en el recurso humano, más aún cuando según Pressman

(2005), las empresas que desarrollan software deberían invertir en la disciplina de pruebas entre el 30% y 50% del costo total del software. Generalmente, de acuerdo a la experiencia laboral de la autora en empresas cuya actividad principal es el desarrollo de software o en empresas que desarrollan sistemas para apalancar sus procesos, el tema de las pruebas no está bien formalizado ni justificado dentro del proyecto de software, por lo que invertir esfuerzo y costos en esta disciplina, es casi nula o cuando está presente, se estiman en dimensiones de tiempo/costo no apropiadas ni ajustadas a la realidad; esta mínima inversión, muchas veces no se extiende al uso de herramientas de pruebas que hagan más plausible el impulso de esta disciplina.

Dedicar esfuerzos en este tema, son bien justificados para el desarrollo de la industria del software en nuestro país. Elaborar nuevos y efectivos métodos para apoyar el desarrollo de software, que sean adaptados a las características de las empresas desarrolladoras de software del país, es una necesidad evidente.

1.5 Alcances y limitaciones

El presente proyecto de investigación está enmarcado dentro de una investigación en progreso auspiciada por el FONACIT bajo el proyecto “METHODIUS” Nro. 205000165, cuyo objetivo principal es “la elaboración de métodos y modelos de desarrollo de software adaptados al contexto nacional y fundamentados en la Ingeniería de Software para que contribuyan a mejorar los procesos productivos de empresas venezolanas que desarrollan software para uso interno o comercial; y elevar la calidad de sus productos.” (Methodius, 2005). Bajo esta premisa, la investigación se orienta a buscar y aportar soluciones al problema de desarrollo de software que tienen las empresas en Venezuela y que por consiguiente, puedan ser aplicadas de forma eficaz en este contexto, específicamente en las PYMEs; esto es, el enfoque es dar soluciones específicas, y no crear métodos universales o de propósito general.

Siguiendo la línea anterior, los resultados de la investigación estarán limitados por este alcance, y se enfocará en desarrollar un modelo de referencia, un modelo bien sustentado y evaluado con directrices específicas para ser implementado o traducido en una aplicación de software en futuras investigaciones.

Se entiende que los usuarios del modelo propuesto por el investigador poseen amplios conocimientos en el tema de las Pruebas de software, dado que deben poder clasificar satisfactoriamente una herramienta de prueba de acuerdo al tipo de prueba que se requiera ejecutar, es decir, comprender el tipo de prueba que puede ejecutar las herramientas que serán objeto de selección/evaluación del modelo.

La búsqueda de información en cuanto a PYMEs se refiere, por razones de tiempo y costo, se limitó a pocas empresas venezolanas desarrolladoras de software distribuidas geográficamente en el área capital venezolana, apoyado con el Laboratorio de Investigación en Sistemas de Información LISI de la Universidad Simón Bolívar (LISI – USB).

CAPÍTULO II: MARCO TEORICO

En el siguiente capítulo se presentan las bases teóricas que sustentan la presente investigación, haciendo un recorrido por conceptos de interés que van desde los antecedentes de investigaciones relacionadas en los últimos años, pasando por definiciones relativos hasta llegar a las bases legales y éticas que soportan este trabajo.

2.1 Antecedentes de investigación

En el ámbito del desarrollo de software, son numerosos los aportes que los investigadores han realizado, siendo éste un tema bastante avanzado. Contextualizado en el ámbito de las PYMEs, Di Paula et al. (2007) hacen un interesante aporte sobre dar formalidad al proceso de desarrollo de software, construyendo una red de decisión para precisar el balance entre agilidad y disciplina dentro de dicho proceso en las PYMEs y Cooperativas de Latinoamérica, tomando como caso de estudio Venezuela. En dicho trabajo, se plasman interesantes contribuciones sobre la realidad de las empresas desarrolladoras de software de Venezuela (constituidas principalmente por PYMEs), al identificar los principales factores que inciden en esta dupla, agilidad y disciplina los cuales permiten una mejor adaptación y mantenimiento de la esencia como empresas pequeñas y/o medianas, para mejorar el desempeño de sus actividades. Dar soluciones apropiadas a las PYMEs Venezolanas, con el propósito de proveer beneficios trascendentales en la “manera de hacer las cosas”, a través de un modelo de referencia para las mismas, es uno de los propósitos de la presente investigación en progreso.

Reyes (2005) desglosa las posibilidades de un mayor desarrollo de la industria del Software en Venezuela, destacando la importancia de ésta industria en el desarrollo del país, y como a lo largo de la historia ha fluctuado esta tendencia. En su artículo, proporciona las directrices de lo que son hoy y pueden llegar a ser en el futuro, las PYMEs desarrolladoras de software. Analizar este abanico de posibilidades que tiene Venezuela para apoyar el crecimiento del desarrollo de este tipo de empresas, inspira el dedicar tiempo y esfuerzo en investigaciones que permitan garantizar calidad de los servicios que prestan, con miras a intensificar las exportaciones de software desde el país, que no se detiene desde hace unas décadas.

Esta calidad esperada en el desarrollo de software es, en gran parte lograda a través de la Disciplina de Pruebas en el proceso de desarrollo de software. Sin embargo, en el ámbito de la Disciplina de Pruebas se percibe una ausencia de investigaciones que se enfoquen en el uso de herramientas de pruebas para que las empresas puedan soportarse en las mismas y apoyar así, sus procesos informáticos.

Algunas acertadas aproximaciones respecto a garantizar aspectos de calidad dentro del proceso de desarrollo de software y su correspondiente medición, han podido lograrse a través del Laboratorio de Investigación en Sistemas de Información LISI de la Universidad Simón Bolívar (LISI - USB), cuyos aportes especiales a la presente investigación son reseñados a continuación.

Méndez et al. (2005) da respuesta a la proposición ¿Cómo se relacionan la calidad sistémica y la productividad en el proceso de desarrollo de software?, al describir las características de la Calidad y la Productividad aplicado al proceso de desarrollo de Sistemas de Software utilizando dos herramientas: El Modelo Sistémico de Calidad (MOSCA) y el Manual de FIM-Productividad (adaptado a empresas de software) con muy buenos resultados. Esto hace entender la necesidad de garantizar el aseguramiento de la calidad en todo el proceso de desarrollo.

En Mendoza et al. (2005), se presentan las bases teóricas, evolución, aplicaciones y demás información relevante sobre el Modelo Sistémico de Calidad MOSCA. Observar cómo un desarrollo de software puede ser medido bajo estrictos criterios de calidad, fundamenta la investigación para recalcar la importancia de tomar en cuenta el concepto de calidad en las actividades de las empresas de software. MOSCA ya forma parte de una de las herramientas de calidad que pueden usarse para apoyar la Administración de la Calidad del Software en sus tres actividades: Aseguramiento de la Calidad, Planeación de la Calidad y Control de la Calidad del proceso de desarrollo de software propiamente dicho.

Aún para productos ya desarrollados, la calidad del software debe estar presente. Esta idea es presentada por Ortega et al. (2007), al crear una herramienta de soporte específicamente orientado a la calidad de estrategias de reutilización de componentes, denominada R-TOOL. Los altos niveles de certificación que puedan encontrarse en los

componentes desarrollados generan confianza y estimulan la Reutilización de los mismos. Este tema constituye otra aplicabilidad e importancia de determinar calidad en los productos de software.

Por su parte, Iovine (2009) en su trabajo analiza 11 herramientas de pruebas del mercado mediante el modelo MOSCA y partiendo de los resultados, propone e implementa mejoras en una de estas herramientas. Las mejoras dieron paso a un módulo que verifica la trazabilidad desde los requerimientos a los casos de prueba planteados en el ámbito de software libre. Este se constituye como un gran aporte a la presente investigación, sentando las bases para reutilizar la metodología en la evaluación de herramientas de pruebas. Este módulo ha sido mejorado por Ricardo (2009) quien plantea e implementa mejoras sustanciales al producto desarrollado.

En cuanto a la selección de herramientas, partiendo de una metodología de software, Matos et al. (2004) muestran un modelo de apoyo para seleccionar metodologías de desarrollo de software, a través del cual podrían determinarse desde etapas tempranas, la mejor opción de estrategia de pruebas para dichos software.

Resulta interesante analizar el modelo propuesto para tratar de traer dichos conocimientos al área de disciplina de pruebas, haciendo una correlación de conceptos de cualquier proceso de desarrollo de software. Por otra parte, enfocados en la selección de herramientas específicamente las CASE, se han adelantado importantes trabajos, basados en áreas específicas como el Modelo integrado para soportar la selección de herramientas CASE (Mendoza et al., 2004) y la Aplicación de un modelo de calidad para la evaluación de herramientas CASE adaptado al estándar internacional ISO/IEC 14102 (Díaz et al., 2001). Para la selección de herramientas de soporte de gestión de la calidad, De Luca et al. (2001) presenta una guía para seleccionar herramientas en cuanto a la gestión de calidad de software se refiere, basada en indicadores identificados y bien detallados en su trabajo. Otro trabajo interesante es el propuesto por Omaña (2009) quien plantea un modelo de calidad para la selección de sistemas de gestión de aprendizaje mediante el análisis y descripción de los componentes de calidad del modelo MOSCA y que finalmente fue aplicado en PDVSA con resultados satisfactorios.

Un completo y práctico modelo basado en pruebas es presentado en forma bien explícita por Utting y Legeard (2006), el cual es está dirigido a la automatización del proceso de pruebas. En él identifican criterios para la selección de pruebas que se traducen en importantes iniciativas para el establecimiento del modelo de calidad producto de esta investigación, pues son adaptables según el propósito de la prueba que se quiera establecer, sin olvidar el tema de las herramientas de pruebas que apoyen el modelo. La forma en que adapta el modelo que ellos proponen con algunos métodos ágiles y el proceso unificado (UP, por sus siglas en ingles), sienta bases para la adaptabilidad que persigue el modelo fruto de esta investigación.

Perry (2006) plantea siete pasos que soportan el proceso de pruebas tradicional y lo adapta a los métodos ágiles, dando vital importancia a las herramientas de pruebas que puedan soportar dicho proceso. De igual forma, el autor clasifica las herramientas de pruebas en un vasto rango, que servirá de base para la selección y categorización de las herramientas de pruebas que existen en el mercado para dar forma al modelo propuesto. Otro aporte importante es reconocer que la selección de las herramientas apropiadas en pruebas, constituye un aspecto importante en el proceso de pruebas, identificando cuatro pasos que deben estar presentes al seleccionar las herramientas de pruebas cónsonas con el proyecto.

2.2 Marco Organizacional

PYMEs es el acrónimo de Pequeñas Y Medianas Empresas y representa un concepto muy difundido en el mundo ya que dentro de la economía mundial, este tipo de empresas tiene un rol muy significativo y con un auge cada vez mayor. Sin embargo dar una definición única es una tarea complicada, pues los criterios o factores utilizados en la definición de PYME, son muy variados, de acuerdo a cada país. En el caso de Venezuela, su definición viene dada en base a dos criterios: Número de trabajadores y ventas Anuales (Carrasquel, 2007), la cual es esquematizada, para facilitar su comprensión, en la Figura 1.

En el marco de la Legislación Venezolana¹, PYME es definida como:
“ARTICULO 3. A los efectos del presente Decreto Ley se entiende por:
1. Pequeña Industria, la unidad de explotación económica realizada por una persona

¹ Ley para la promoción y desarrollo de la pequeña y mediana industria, Decreto N° 1547, 09 de noviembre de 2001

jurídica que efectúe actividades de producción de bienes industriales y de servicios conexos, comprendidas en los siguientes parámetros:

1.1 Planta de trabajadores con un promedio anual no menor de once (11) trabajadores, ni mayor a cincuenta (50) trabajadores.

1.2 Ventas anuales entre nueve mil una (9.001) Unidades Tributarias y cien mil (100.000) Unidades Tributarias.

2. Mediana Industria, la unidad de explotación económica, realizada por una persona jurídica que efectúe actividades de producción de bienes industriales y de servicios conexos, comprendida en los siguientes parámetros:

2.1 Planta de trabajadores con un promedio anual no menor a cincuenta y un (51) trabajadores, ni mayor a cien (100) trabajadores.

2.2 Ventas anuales entre cien mil una (100.001) Unidades Tributarias y doscientos cincuenta mil (250.000) Unidades Tributarias.”

Las PYMEs son importantes generadoras de empleo, de crecimiento y, por ende como conglomerado, demandan una importante cantidad de servicios y productos. No obstante, su potencial de crecimiento en la economía venezolana es su principal activo (Jiménez, 2006). Importantes organizaciones en el país (gubernamentales, privadas, de financiamiento, entre otras), apoyan este tipo de empresas por su impacto en la economía nacional. Resulta cada día más evidente que existe una conciencia mayor en torno al papel que desempeñan las PYMEs, tanto en el ámbito económico como en la generación de empleo.

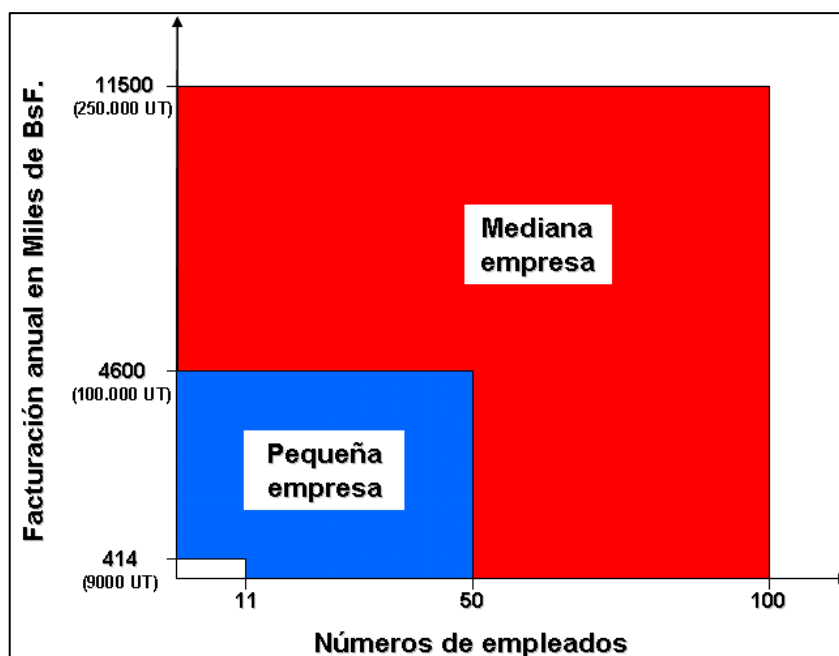


Figura 1 – Definición de pequeñas y medianas empresas en Venezuela según decreto Nro. 1547. Valor de la unidad tributaria vigente para Enero 2008 de BsF. 46,00. Fuente: Carrasquel (2008).

En el entorno de las tecnologías de información, las PYMEs en Venezuela tienen gran auge. Las empresas desarrolladoras de software dedicadas expresamente al desarrollo, mantenimiento y comercialización de productos de software para terceros, constituye el sector industrial en el que se enfoca la presente investigación que está muy cercana a ésta realidad. Castejón (2005) afirma que las PYMEs se constituyen como las mayores proveedoras de software nacionales, y en algunos casos han podido participar y ganar licitaciones de proyectos grandes y de criticidad importante; debido a sus bajos precios con respecto a la competencia foránea y las grandes empresas locales. En fin, la participación e importancia en Venezuela de la industria del software, es de gran relevancia económica. Aún, con buenas estrategias de negocio, se espera que en un corto plazo, se lograra posicionar la industria del software nacional en los mercados internacionales, colocando al país como emergente en materia de desarrollo de software.

En este sentido, el desarrollo de la industria del software como pilar importante en el desarrollo del país, está caracterizada por los siguientes rasgos (Reyes, 2005):

- *Es una industria donde las pequeñas y medianas empresas juegan un rol importante, y es, asimismo, una industria intensiva en recursos humanos de alta calificación.*
- *Los activos y las inversiones requeridas para comenzar y desarrollar una empresa de software son relativamente bajos en comparación con otros sectores industriales.*
- *Los principales factores de producción de esta industria son el talento humano, las capacidades de producción, la calidad, la innovación y la capacidad de insertarse en poderosas cadenas de suministros globales.*
- *Los principales activos de una empresa de software son los conocimientos, derechos de autor, relaciones con los clientes, etc.*

Un estudio realizado por Rivero et al. (2007), revela como es la tendencia (por tamaño de la empresa) de la industria del software en Venezuela, en base a 150 empresas ubicadas en las principales ciudades del país (Figura 2). Aquí puede observarse que el 83,68% de las empresas, son empresas que van desde ser microempresas hasta empresas pequeñas, representado el estrato principal las empresas pequeñas con un 42,86% de la muestra. Parte del estudio también revela que las empresas de software en Venezuela, son industrias muy jóvenes, donde el 74% tienen en el mercado entre 1 y 12 años, siendo las PYMEs (42%) creadas en los últimos 7 años (al combinar los factores tamaño de la empresa con la antigüedad). Dedicar esfuerzos para aportar beneficios a las empresas desarrolladoras de software en Venezuela, centradas en PYMEs, resulta ser una actividad necesaria en la búsqueda de hacerlas más competitivas.

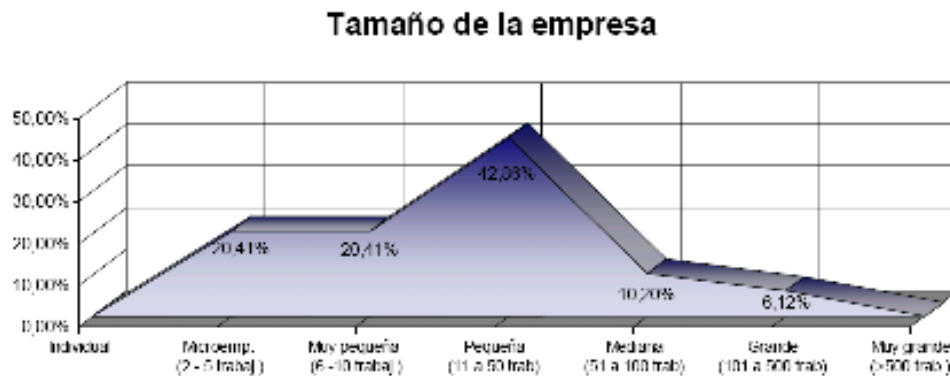


Figura 2- *Tamaño promedio de las empresas de la Industria Nacional del Software.*
Fuente: Montilva (2007)

Por su parte, Granados et al. (2008) igualmente realizó un estudio que revela aspectos interesantes sobre la industria del software en Venezuela, donde puede observarse la distribución geográfica de las empresas desarrolladoras de software en el país (Figura 3). El estudio está basado en el siguiente perfil general de la Industria del Software:

- *Universo de estudio: Empresas que comercializan productos y/o servicios de software, sin fines de autoconsumo.*
- *No. de Empresas: 71*
- *Criterio de estratificación muestral: Región Los Andes-Zulia (Mérida, Maracaibo, Punto Fijo, San Cristóbal), Región Capital (Caracas, San Antonio de los Altos, Guatire) y Región Centro-Oriente (Maturín, Barquisimeto, Maracay, Valencia y Puerto Ordaz).*

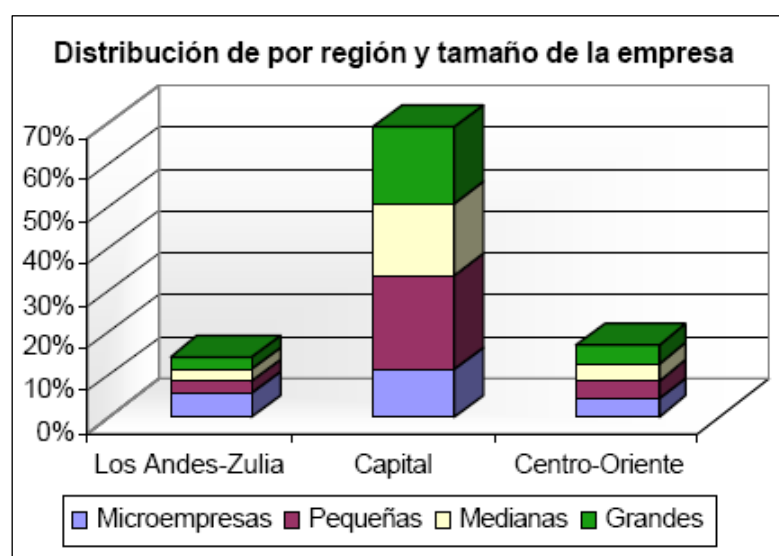


Figura 3- *Distribución geográfica de empresas desarrolladoras de software en Venezuela.*
Fuente: Granados (2003).

De forma más específica, en el estudio de Rivero et al. (2007), presentan mediante la figura 4 los tipos de aplicaciones producidas por la industria nacional de software. Aquí se revela que el 41% de las empresas encuestadas elabora aplicaciones de negocio o administrativas, el otro 59% elabora aplicaciones embebidas, industriales, software del sistema, aplicaciones científicas, software educativo, de entretenimiento y otros. En todo caso, lo aquí revelado es que la tendencia al desarrollo de software nacional está en continuo crecimiento.

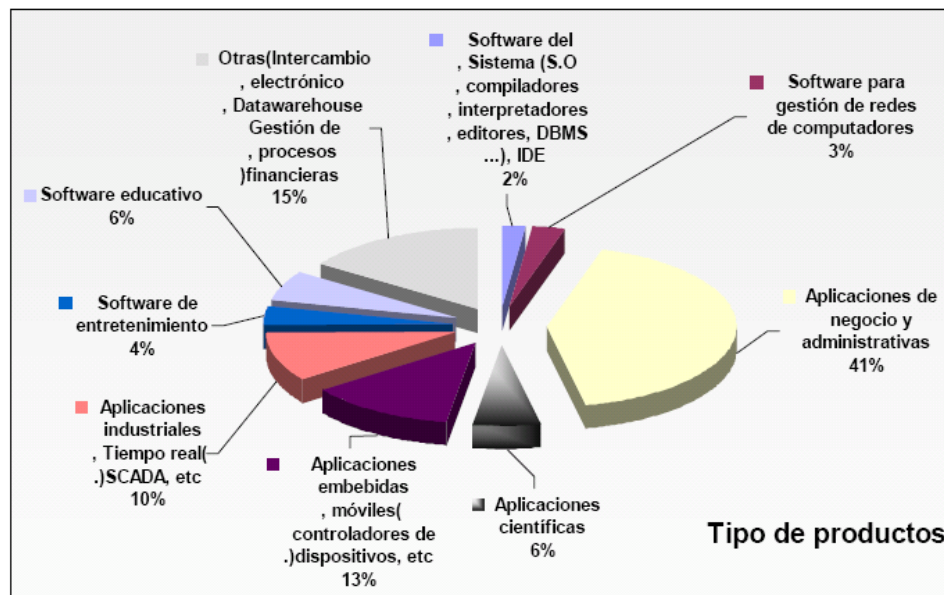


Figura 4 - Tipos de aplicaciones que produce la Industria Nacional de Software.
Fuente: Montilva (2007).

Un aspecto de suma importancia demostrado en el estudio de Rivero et al. (2007), lo constituye el descubrir las principales necesidades de conocer, manejar y actualizarse en todo lo concerniente al proceso de desarrollo de software. Vale destacar en la Figura 6 cómo las pruebas de software, aunque no es la más crítica de los procesos inherentes al desarrollo de software, se ubica en el tercer puesto, por lo que su necesidad de afianzar este rubro y especializarse en todo lo concerniente a pruebas del software, madurando el concepto, es de vital importancia para producir software con la calidad exigida por sus clientes y aun a nivel internacional. Esto trae consigo que dentro de los procesos de gestión de proyectos de software, el aseguramiento de la calidad (gestión de la calidad del software - SQA), también se vea altamente necesitada de capacitación en su recurso humano (Figura 5 y 6). Gestión de la calidad y Pruebas del software, son temas que deben estar de la mano y contar con un nivel de madurez óptimo para que pueda ser aplicado en el proceso de desarrollo de software.

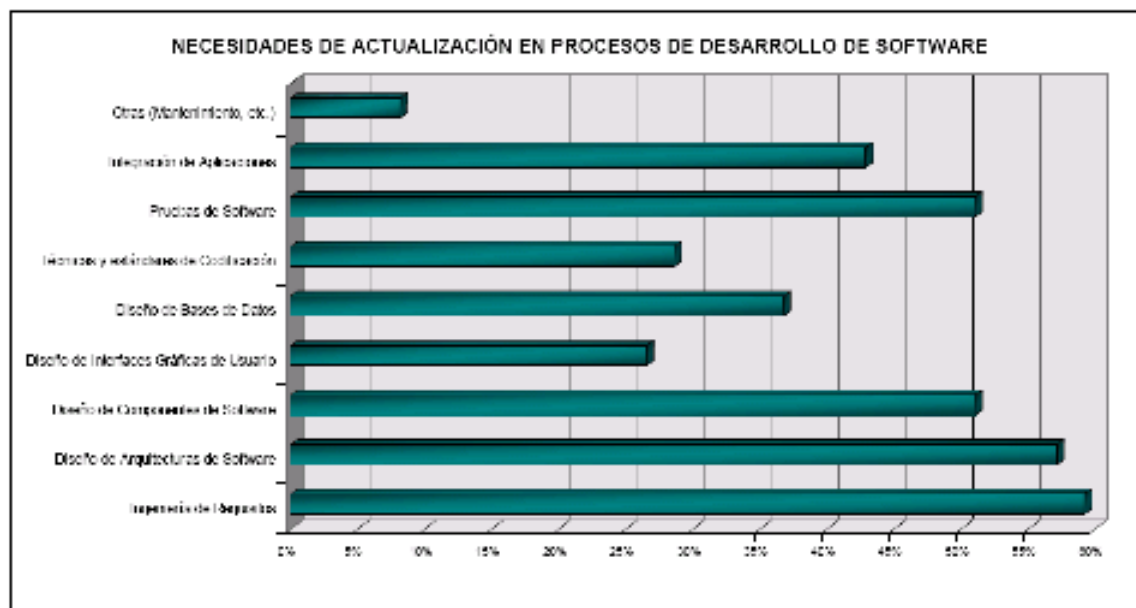


Figura 5 - *Necesidades de actualización en procesos de desarrollo de software.*
Fuente: Montilva (2007).

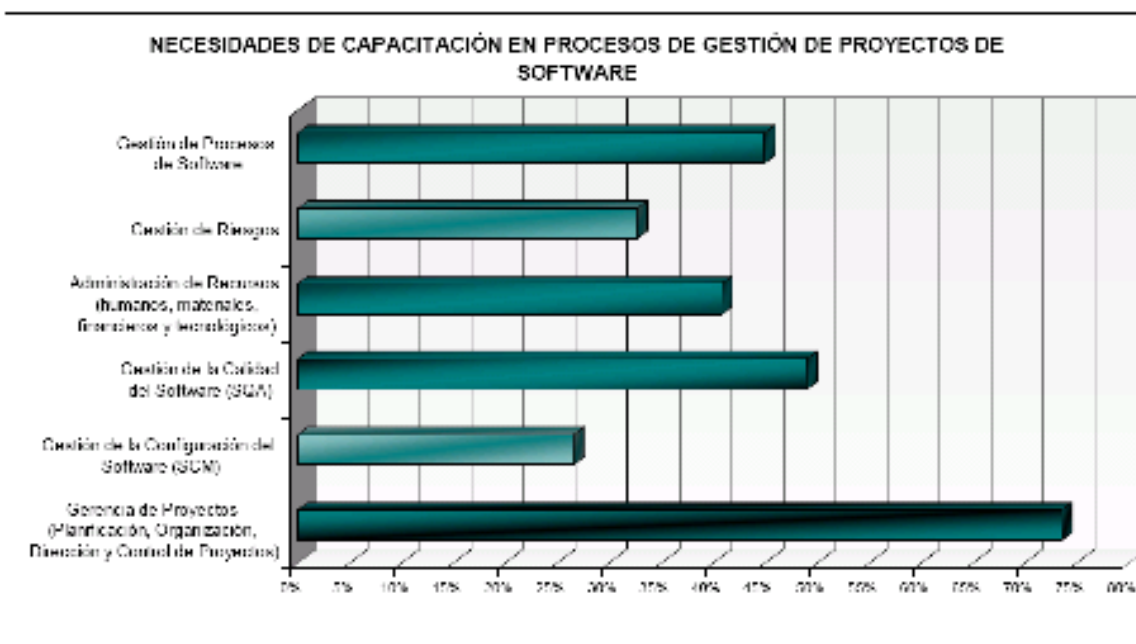


Figura 6 - *Necesidades de capacitación en procesos de gestión de proyectos de software.*
Fuente: Montilva (2007).

Por lo antes expuesto, se concluye que el campo de acción para reforzar y/o mejorar los procesos de dichas empresas, en cuanto a pruebas se refiere, es amplio y concurrido. Esta conclusión toma mayor fuerza cuando se observan algunas tendencias expresadas por Méndez et al. (2007) respecto a las PYMEs en el ámbito de las pruebas. Ellos citan a Pressman (2005) planteando que las empresas desarrolladoras de software invierten en la disciplina de pruebas entre el 30% y 50% del costo total del desarrollo del software. Hablando del tema de la automatización de pruebas, Granados et al.

(2008) presenta la figura 7 donde resume la utilización de algunas herramientas de apoyo durante el desarrollo de software, siendo que el área relativa al Testing Automático, revela que alrededor del 60% de las empresas no usan este tipo de herramientas. Esto representa un esfuerzo considerable indicando que es una disciplina cuyos resultados (confiabilidad de la prueba) pueden impactar sobre la percepción del cliente o usuario en cuanto a la calidad del software que se le está entregando.

Herramientas de apoyo en la ejecución del proyecto	Frecuencia de uso	Micro-empresas	Pequeñas	Medianas	Grandes	Total
Análisis y Diseño de Software asistido por herramientas de diagramación	Uso Frecuente	9,86%	15,49%	14,08%	16,90%	56,34%
	Uso ocasional	5,63%	7,04%	4,23%	1,41%	18,31%
	Conoce y no lo usa	5,63%	7,04%	5,63%	5,63%	23,94%
	No conoce	0,00%	0,00%	0,00%	1,41%	1,41%
Debugger	Uso Frecuente	11,27%	16,90%	9,86%	12,68%	50,70%
	Uso ocasional	5,63%	8,45%	4,23%	9,86%	28,17%
	Conoce y no lo usa	4,23%	4,23%	7,04%	2,82%	18,31%
	No conoce	0,00%	0,00%	2,82%	0,00%	2,82%
Control de Versiones	Uso Frecuente	14,08%	12,68%	12,68%	19,72%	59,15%
	Uso ocasional	0,00%	9,86%	5,63%	5,63%	21,13%
	Conoce y no lo usa	5,63%	5,63%	2,82%	0,00%	14,08%
	No conoce	1,41%	1,41%	2,82%	0,00%	5,63%
Testing Automático	Uso Frecuente	4,23%	2,82%	1,41%	2,82%	11,27%
	Uso ocasional	8,45%	7,04%	5,63%	8,45%	29,58%
	Conoce y no lo usa	5,63%	14,08%	14,08%	11,27%	45,07%
	No conoce	2,82%	5,63%	2,82%	2,82%	14,08%
Control de Configuraciones	Uso Frecuente	4,23%	8,45%	4,23%	7,04%	23,94%
	Uso ocasional	4,23%	7,04%	7,04%	8,45%	26,76%
	Conoce y no lo usa	8,45%	9,86%	12,68%	7,04%	38,03%
	No conoce	4,23%	4,23%	0,00%	2,82%	11,27%
Gestión de Requisitos	Uso Frecuente	11,27%	9,86%	16,90%	18,31%	56,34%
	Uso ocasional	2,82%	7,04%	1,41%	2,82%	14,08%
	Conoce y no lo usa	4,23%	7,04%	4,23%	2,82%	18,31%
	No conoce	2,82%	5,63%	1,41%	1,41%	11,27%

Figura 7 - Utilización de algunas herramientas de apoyo durante el desarrollo de Software. Fuente: Granados (2008).

Es importante traer a colación, en líneas generales, algunos obstáculos que enfrentan las PYMEs según Erard (1999) y que son importantes conocer para contextualizarlo dentro de las empresas desarrolladoras de software venezolanas: Se centran en poca gente las diferentes actividades de negocios y tienen flexibilidad en los roles que ejerce el personal. Hay una falta de formalidad en las relaciones y roles entre los individuos que interactúan. El flujo de trabajo flexible, es regido por las necesidades del momento. Erard afirma que la mayor dificultad de las PYMEs es conseguir las personas adecuadas y las herramientas (tecnología y maquinaria) necesarias para realizar su trabajo. Según Rivero et al. (2007), las mayores deficiencias de los expertos están en el manejo de modelos de proceso y gestión (conocimiento y herramientas). Di Paula et al. (2007) agrega de acuerdo a sus conclusiones que a pesar de ser muy capaces en su negocio y las tecnologías que manejan, sus procesos son erráticos y dependen

exclusivamente de los estándares que determine el líder del proyecto. Todas estas realidades, hacen más fuerte la necesidad ya evidenciada de establecer formalidad en procesos internos de las PYMEs y especialmente dentro del marco de la Disciplina de pruebas. Este escenario debe aprovecharse a su vez para impulsar el uso de herramientas que apalanquen la Disciplina de pruebas dentro del proceso de desarrollo de software, justificado en las carencias del uso de herramientas que agilicen sus procesos.

2.3 Bases Teóricas

El desarrollo de software debe estar enfocado en la imperiosa realidad de garantizar calidad tanto en todo su proceso como en sus resultados (producto) para así traducirse en una estrategia empresarial y lograr a través de este, las ventajas competitivas en un entorno globalizado.

Tomando como base esta realidad, la presente investigación se ha dividido en cinco áreas de importancia para tratar de profundizar en el tema de la calidad de forma incremental y tomando como base varias fuentes bibliográficas. Todas estas fuentes han sido analizadas con el propósito de encontrar el enriquecimiento de conocimiento y entender los conceptos analizados.

La primera parte aborda todo lo concerniente a la calidad del software iniciando por definir el término calidad hasta llegar a aplicar dicha definición en el ámbito del software. Seguidamente se estudia el área del aseguramiento de la calidad del software, analizada bajo diferentes ópticas (autores, estándares, tendencias) y correlacionado dichos conceptos para determinar los componentes de lo que se conoce como Aseguramiento de la calidad del software ideal. Una tercera se constituye como un glosario dedicado al tema central de esta investigación: las pruebas del software, mediante la cita y análisis de conceptos variados acerca de todo lo concerniente a Pruebas del software y su importancia en el proceso de desarrollo del mismo. Una cuarta, trata acerca de la disciplina de pruebas, constituida en la mayoría de sus defensores, como una actividad dentro del proceso de desarrollo de software, diferenciada en roles, actividades, fases y demás componentes de acuerdo a la metodología de desarrollo elegida. Finalmente una quinta, aborda lo concerniente a los modelos de calidad y su clasificación con una explicación detallada de cada uno, identificando su aporte al área de calidad.

I. Calidad del Software

Lograr un alto nivel de calidad de un producto o servicio es el objetivo de muchas organizaciones a nivel mundial. Asegurar la calidad en cada una de las fases por las que transita el desarrollo de un producto/servicio se traducirá en múltiples beneficios para que el mismo produzca los resultados esperados. Esto es lo que James (2000) trata de enseñar al desarrollar el punto Desarrollo de la calidad a través del aseguramiento de la calidad.

La industria del software no escapa de esta realidad, más cuando el software, en la actualidad, es el combustible en los negocios modernos, reglas del gobierno y sociedades que desean ser conectadas, en fin, permite el desarrollo de la economía mundial (Krucchten, 2000). Esto determina que el software tiene un valor muy importante en nuestra sociedad.

El término calidad de Software, puede ser relacionado con la aplicación de calidad en el ámbito del software; sin embargo, bien vale la pena tener claro el concepto de calidad antes de aplicarlo al software. Con este propósito, se traen a colación variados conceptos de calidad para así contextualizar su incidencia en lo que concierne al software.

La real academia española (2001), define *calidad* como la propiedad o conjunto de propiedades inherentes a algo, que permiten juzgar su valor. Llama la atención que dicho conjunto de características puedan determinar el valor de algo; es allí donde toma fuerza el decir: ‘tal producto es de alta o baja calidad’.

La ASQ (American Society for Quality, por sus siglas en ingles) plantea que en el sentido técnico de la calidad (y no de forma subjetiva, dado que cada persona o sector puede definir calidad empíricamente), tiene dos significados: 1. Las características de un producto o servicio que refieren su capacidad de satisfacer lo indicado o las necesidades implícitas; 2. Un producto o un servicio libre de deficiencias.

Juran (1998) considerado el Padre de la Calidad, destaca dos conceptos de los muchos significados atribuibles al termino calidad, describiéndola desde la perspectiva

del cliente. Calidad significa esas características de los productos que resuelven necesidades del cliente y de tal modo proporcionan la satisfacción de cliente, es decir, alta calidad significa una gran cantidad de características que satisfacen las necesidades de los clientes. El segundo aspecto se relaciona con “libertad de problemas”: Alta calidad consiste en pocos defectos, libre de deficiencias.

Crosby (1980) en comparación con algunos gurúes de la calidad (Juran, Demming e Ishikawa), flexibiliza el concepto de calidad proponiendo que ésta debe ser resultado del trabajo de todos y no solo del departamento de control de calidad; la calidad es libre pero no es un regalo. Crosby, basó sus ideas en definir la calidad como la conformidad con los requisitos, es decir, gerenciar la calidad es una manera sistemática de garantizar que las actividades organizadas suceden en la manera que se planean.

Cabe destacar que todos estos conceptos, están inmersos en el ámbito de la legislación Venezolana, donde la Ley del sistema Venezolano para la Calidad (2002) en su Capítulo I, Art. 4, define calidad como el Grado en que un conjunto de características inherentes a bienes y servicios cumple con unas necesidades o expectativas establecidas, generalmente implícitas u obligatorias (requisitos). Todo esto permite establecer que los elementos presentes en la calidad son: características, cumplimiento/satisfacción, requisitos. Esto quiere decir que siempre estarán presentes expectativas que deben ser cubiertas, de acuerdo a las necesidades expresadas (directa o indirectamente) por los clientes y/o usuarios del producto o servicio resultante. A su vez, el grado en que son cubiertas, determinará qué tan bueno o malo es el producto resultante (mientras más necesidades sean cubiertas será visto como de mejor calidad).

Por ser tan importante la calidad, se hace necesario incluir temas que permitan profundizar en los conceptos de calidad y calidad del Software así como la importancia de asegurar calidad en el proceso de desarrollo de software.

Unhelkar (2003), de una manera más formal, establece los ingredientes esenciales en la creación de “buena” calidad, describiéndolos de la siguiente manera: 1.- Un producto, 2.- Un proceso de producción (el proceso o método usado en la creación del producto es tan importante como el producto mismo), 3.- Un mecanismo de

comunicación (para comunicarse con el dominio del desarrollo y a través de la industria –usuarios, productos, procesos), 4.- Una mejora continua de los ingredientes anteriores (es un gran esfuerzo producir un buen producto de calidad, mejorar lo que se logra es importante).

Uniendo todas estas ideas en el ámbito del software, el concepto de calidad bien puede extrapolarse a lo que es calidad aplicada al software, pues mantiene su esencia pero con un tecnicismo que debe entenderse, acordarse y refinarse. El software debe ser visto como un producto. Sin embargo, Unhelkar (2003) reflexiona acerca de que la naturaleza del software es tal que dificulta juzgar su calidad objetivamente, pues necesita a los usuarios finales y los productores de tal software. Por esta razón, hay una necesidad de que lo subjetivo de este concepto prevalezca ante tan diverso grupo de personas que demandan un amplio estudio contextual para entender la calidad del software, dejando que la calidad signifique diversas cosas para diferentes personas. He aquí la importancia de darle la apropiada dirección en su definición.

Diversos conceptos de calidad de software, se traen a colación, para su discusión:

Según el IEEE (Institute of Electrical and Electronics Engineers, por su siglas en ingles) en su estándar 610.12-1990, la *calidad de software* es definida como el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario. Aquí los elementos presentes en el concepto de calidad del software son: requisitos, necesidades y expectativas.

El proceso unificado de Rational RUP (2005), define la calidad de software como las características que tienen que demostrar el logro de producir un producto que satisface o excede los requerimientos convenidos y medidos según criterios y medidas acordados y que es producido de acuerdo a un proceso estipulado. Esta definición manifiesta la no simplicidad de todo lo concerniente con la calidad de software dentro del proceso de desarrollo del mismo. Lograr la calidad no es simplemente satisfacer los requerimientos o producir un producto que satisfaga las expectativas y necesidades de los usuarios; más bien, la calidad del software también debe incluir lo planteado por RUP, la identificación de las medidas y criterios para demostrar que la calidad se ha

alcanzado y la implementación de un proceso que asegure que el producto resultante contiene el grado de calidad esperado.

En una aproximación al concepto de calidad de software, la “Guide to the Software Engineering Body of Knowledge”, conocida como SWEBOK (2004), afirma que la importancia de este concepto es que los requerimientos del software definen las características de calidad requeridas del software e influyen los métodos y criterios de aceptación para determinar esas características. Esta definición, al igual que la de RUP, hace especial énfasis en los métodos y criterios de aceptación que se determinan en el proceso de calidad del software, además de tener la importancia de tener claros los requerimientos de software, que guiarán la calidad en el software.

Por su parte, Somerville (2006) afirma que calidad del software significa, sencillamente que un producto debe resolver/cubrir su especificación. Sin embargo al hablar sobre la administración de la calidad del software, presenta algunos problemas y/o realidades de los sistemas:

- *Hay una tensión entre los requisitos de calidad del cliente (eficacia, confiabilidad, etc.) y los requisitos de calidad del desarrollador (capacidad de mantenimiento, reutilidad, etc.);*
- *Algunos requisitos de calidad son difíciles de especificar de forma inequívoca;*
- *Las especificaciones del software son generalmente incompletas y a menudo contrarias.*

Ante esta realidad, el autor está convencido de que administrar la calidad del software permitirá obtener gran calidad a pesar de las imperfecciones en las especificaciones de requerimientos, siempre y cuando las actividades concernientes, se ejecuten dentro del tiempo indicado.

Ahora bien, ante las variadas opciones presentadas, surge la de Pressman (2005) que cobra una mayor fuerza por ser más explícita e incluir conceptos netamente de software, pero que según el propio autor, puede ser ampliada o modificada con todos los conceptos presentados, hasta lograr una definición más clara: Concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados, y con las características implícitas que se espera de todo software desarrollado profesionalmente. De esta definición, se pueden

observar tres puntos importantes que deben tomarse en consideración, como lo expresa Pressman (2005):

1. *Los requisitos de software (explícitos) son la base de las medidas de la calidad. De hecho, una falta de correspondencia entre estos requerimientos, ya da por sentado una falta de calidad importante.*
2. *Existen un conjunto de estándares especificados constituidos por un conjunto de criterios de desarrollo que guiarán la mejor manera de hacer las cosas, y cuya ausencia se traducirá en falta de calidad.*
3. *Existe otro tipo de requisito de software que normalmente no se mencionan formalmente (implícitos), los cuales deben tomarse en cuenta en unión con los requisitos explícitos para que no se pueda dudar de la calidad del software que se construye.*

En todos estos conceptos lo que se quiere tratar de minimizar es la variación que pueda existir entre las expectativas de los clientes y los resultados que emita el producto final (software). Controlar esta variación es la clave de un producto de alta calidad, y se hace controlando desde el proceso en sí, los recursos utilizados y los atributos de calidad que deberían caracterizar el software final. Esto es la esencia de la Calidad de Software y que será utilizada para efectos de esta investigación, basándose en la definición de Pressman, pues es la que engloba los elementos que dirigen la disciplina de pruebas.

Sin embargo, entender la calidad del software sigue siendo una actividad a la cual no se le da la importancia requerida, aunque se tenga la certeza que garantizarla es una buena y necesaria práctica dentro del desarrollo de sistemas. Esta situación es presentada por Kruchten (2000) cuando afirma que diferentes proyectos de desarrollo del software fallan de diversas formas, por lo que es posible identificar algunos problemas que son latentes en organizaciones que buscan implementarlo, enmarcado dentro de los síntomas que caracterizan dicho tipo de proyectos:

- Comprensión inexacta de las necesidades de los usuarios finales.
- Las necesidades del usuario o la empresa no se cumplen.
- Procesos de software normalmente improvisados.
- Los requerimientos cambian rápidamente.
- Los módulos no se integran.
- Difícil de darle mantenimiento.
- Baja calidad o experiencia del usuario final.
- Descubrimiento tardío de fallas.
- Pobre calidad del software.
- Cada miembro de los equipos hacen todo a su manera, haciendo imposible reconstruir quién cambió qué, cuándo, dónde y por qué.
- Desempeño deficiente bajo carga.

- Problemas de construcción y liberación

Todas estas realidades expresadas, convergen en una realidad latente que debe menguar para dirigir en un mejor camino el desarrollo de software y por ende, todo lo concerniente al software de calidad.

Ante todos estos problemas y realidades presentes en el desarrollo del software, resalta la necesidad de intentar solucionarlos o al menos mitigarlos desde etapas tempranas. Es aquí donde surge el concepto de Aseguramiento de la Calidad del software, el cual será tratado en la siguiente sección, analizado bajo diferentes focos pero llegando a un consenso de lo que debería ser el proceso de aseguramiento de la Calidad ideal.

II. Aseguramiento de la Calidad del Software

El Aseguramiento de la calidad, emerge como una gran necesidad en el desarrollo de software. Según RUP (2005) una de las mejores prácticas que comúnmente son usadas por organizaciones exitosas instauradas en la actualidad para el desarrollo de software, es verificar calidad continuamente. Mientras más tarde se consigan fallas en el software será más costoso repararlos. RUP (2005), plantea que valorar continuamente la calidad de un sistema con respecto a su funcionalidad, fiabilidad, desempeño de la aplicación y del sistema completo, es de suma importancia al desarrollar un sistema, todo con el fin de dar respuesta a la problemática de sistemas con síntoma de calidad pobre, por la causa de pruebas pobres. Este punto, da entrada a lo que es el Aseguramiento de la calidad del software.

Aunque para este concepto las siglas en ingles son únicas SQA (Software Quality Assurance, por sus siglas en inglés), algunos autores lo traducen como: Administración de la calidad, Garantía de la Calidad del software o bien, Aseguramiento de la Calidad del Software. En todo caso, algunas de las definiciones que se plantean para este concepto están:

La IEEE - Computer Society (IEEE Std.730-2002), plantea que el *aseguramiento de la calidad* es un patrón planeado y sistemático de todas las acciones necesarias, para proveer suficiente confianza de que el software se construye conforme a los requerimientos técnicos preestablecidos.

Para Sommerville (2006), aseguramiento de la calidad es el establecimiento de un marco de trabajo de procedimientos y estándares organizacionales que conduce a software de alta calidad; define un marco de trabajo para lograr la calidad del software.

Por su parte, SWEBOK (2004) plantea que el proceso de SQA proporciona seguridad de que el producto y proceso de software en el ciclo de vida del proyecto se corresponden con sus especificaciones de requerimientos planificados, decretando y realizando un conjunto de actividades que proveerán la confianza adecuada de que la calidad se está construyendo dentro del software.

El modelo de capacidad y madurez integral, CMMI (Capability Maturity Model Integration, por sus siglas en ingles), define éste concepto como los medios planeados y sistemáticos para asegurar a la gerencia que los estándares, las prácticas, los procedimientos y los métodos definidos del proceso son aplicados. El área del proceso de aseguramiento de la calidad de proceso y producto apoya la entrega de productos y servicios de alta calidad proveyendo al equipo del proyecto (staff y encargados de todos los niveles) de la visibilidad apropiada, el feedback, procesos y trabajo asociado a los productos a través de la vida del proyecto. CMMI-DEV versión 1.2 (2006).

Pressman (2005) afirma que la garantía de la calidad del software es la preocupación de cada Ingeniero del Software para reducir costos y mejorar el tiempo para mercadear el producto. Bajo este concepto, el mismo autor realiza varios planteamientos importantes: la conformidad existente entre los requerimientos del software es el fundamento sobre el cual se mide la calidad del software. Los estándares específicos se utilizan para definir el criterio de desarrollo utilizado para tutelar la manera en la cual se dirige el software.

Todos estos conceptos tienen aspectos en común que deben resaltarse; estos planteamientos apuntan hacia demostrar la necesidad de que exista un proceso, planificación, equipo de personas, entre otros elementos para obtener un producto de calidad, la suma de los cuales excede las limitaciones de un simple control de calidad tradicional, esto es, la existencia de un proceso que asegure calidad, dando por sentado

la imposibilidad de pretender inyectar calidad al producto de software una vez finalizado el proceso de desarrollo del mismo.

Diferentes autores (Pressman, Sommerville, SWEBOOK, Schulmeyer y McManus, Unklear) hacen aportes acerca de las actividades o fases que deben estar presentes en el aseguramiento de la calidad. Dentro de cada fase, se intercalan actividades y tareas específicas a lo largo del ciclo de desarrollo o mantenimiento de una aplicación, con la intención de reducir errores y demás propósitos del aseguramiento de la calidad de software. A continuación se explican brevemente cada uno de ellos, en busca de hacer una integración valiosa en cuanto a ésta actividad.

Profundizando en el tema del Plan de la Calidad, la IEEE establece un estandar para los planes de aseguramiento de la calidad del software (IEEE Std 730-2002), en el que proporciona una ayuda al momento de determinar el contenido y preparación de los planes de calidad tanto para el desarrollo como el mantenimiento del software, dirigido a tres grupos de personas a las que aplica o afecta: usuario, proveedor y público (cliente). En todo caso, la completitud y profundidad en los elementos presentes en un plan de calidad propuesto por la IEEE, hace considerarla como una guía obligada para esta área.

De forma resumida, en la Figura 8 (diagramas de actividades UML) pueden visualizarse los roles, entregables, actividades por rol y procesos generales que propone el autor y deben cumplirse en el Aseguramiento de la calidad durante todo el proceso. De cada una de estas actividades se derivan tareas de suma importancia para todo el proceso que velan por asegurar la calidad deseada, tales como selección de técnicas de calidad, estimación y medición proceso calidad, identificar y reportar errores, validar correcciones de errores, entre otros.

Uno de los roles primarios llevados a cabo por el aseguramiento de la calidad y control de calidad es observar lo que está siendo realizado por otros, evaluarlo, criticarlo y mejorarlo. Bajo este concepto de ideas, se propone que el personal de calidad debe ser independiente de los diseñadores y desarrolladores. Esta independencia asegura que la actividad de calidad continuará observando todo el proceso desde otra perspectiva.

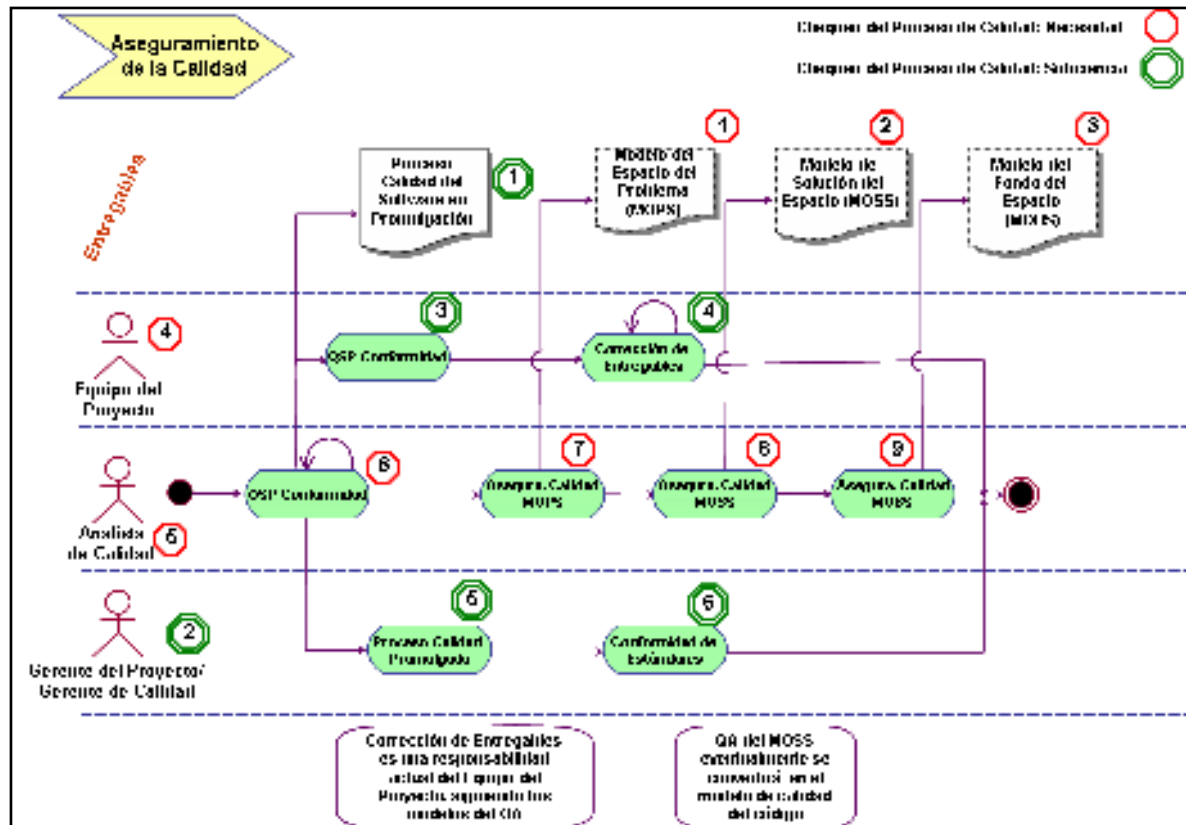


Figura 8- Procesos y Componentes para el Aseguramiento de la Calidad, Fuente: (Unhelkar, 2003).

CMMI (2006), apoya esta afirmación al expresar que tradicionalmente, el grupo de aseguramiento de la calidad que es independiente de este proyecto, puede ofrecer la objetividad esperada en el proceso. Esto puede ser apropiado en algunas organizaciones (generalmente las pequeñas), pero en organizaciones con un proceso abierto, orientado a la cultura de calidad, el aseguramiento de la calidad del proceso y producto podrán ser llevados a cabo, parcial o totalmente por pares, y la función de garantía de calidad puede estar inmersa en el proceso.

De todo el material recopilado, la Tabla 1 compara los conceptos presentados por los diferentes autores y metodologías sobre aseguramiento de la calidad, buscando establecer una correlación resultante entre los mismos, además de identificar algunas diferencias sustanciales entre sus criterios. La idea, es poder determinar cuáles serían las actividades contempladas o no, comparando todos estos criterios analizados y aproximarse al aseguramiento de la calidad ideal que contenga las mejores prácticas de todos estos autores. Los criterios de comparación fueron elegidos con propósitos descriptivos a manera de vista rápida de cada tendencia analizada, los cuales son: *nombre* (a manera de homologar criterios, cómo lo llama cada autor), *elección de*

estándares (si se basa en estándares de calidad o al menos son tomados en cuenta), *revisión de calidad* (si en el proceso contempla actividades para dichas revisiones), *registrar fallas* (tiene la posibilidad de asentar fallos en el proceso), *corrección de fallas* (están establecidos actividades para corregir defectos), *control y gestión de cambios* (si contempla la administración de cambios en sus procesos), *riesgos y su administración* (contempla los riesgos y su gestión), *recopilación y análisis de métricas* (mide sus procesos a través de métricas), *QA estadística* (medición de los procesos de aseguramiento de la calidad usando la estadística), *presente en todo el proceso de desarrollo* (se evidencia su presencia en todo el proceso de ciclo de vida del software), *especificación de Roles* (identifica roles de quienes participan en el proceso con sus responsabilidades), *plan de SQA* (propone o se guía por un plan de aseguramiento de la calidad detallado), *especificación documentación* (apoya el proceso en documentación claramente identificada), *independencia team calidad vs. desarrolladores* (considera que el equipo de calidad debe ser distinto al equipo desarrollador de software) y *mejora de procesos* (contempla el mejoramiento de los procesos).

Como puede observarse en la Tabla 1, sin importar los diferentes nombres cómo pueda ser llamado el aseguramiento de la calidad, este tema tiene más puntos comunes que discordantes entre sus defensores. Todos especifican tener un Plan de aseguramiento de la calidad, por lo que puede derivarse su gran importancia para dirigir el proceso. Igualmente, la especificación de los roles que ejecutarán las actividades, está presente en todos los autores, no así la independencia entre el equipo que pertenezca al aseguramiento de calidad y los que pertenezcan al grupo desarrollador. Esta independencia parece no ser un tema de especial cuidado entre sus defensores; con esto se podría deducir el hecho de que es innecesario dos equipos distintos, siempre y cuando los mismos, puedan trabajar con más de un rol por actividad (intercambiar roles). Los criterios revisión de calidad, registrar fallas y su corrección, la presencia en todo el proceso de aseguramiento de la calidad, implementar la mejora de procesos y tener documentación especificada, son criterios presentes en casi todos los autores, a excepción de una sola por autor. Es importante hacer notar que la presencia del aseguramiento de la calidad en todo el ciclo de vida del software, es una realidad de suma importancia pues con la presencia de ésta, la mejora de los procesos sería una actividad natural, consecuencia del control continuo. Ahora bien, es importante destacar como la administración de riesgos y el control de cambios no forman parte de criterios

comunes para sus defensores, actividad que no debería descuidarse en el desarrollo de proyectos de software, pues siempre están latentes.

Ante toda esta información recopilada y analizada, bien vale la pena poder establecer las diferencias y relaciones del tema de aseguramiento de la Calidad con la disciplina de pruebas. Esta diferencia es expresada por Unhelkar (2003), quien argumenta acerca de *‘Aseguramiento de la calidad y pruebas: no los confundamos’*. Normalmente se suele escuchar la frase: “El producto está en aseguramiento de la calidad”, pero no es una correcta interpretación de éste concepto. Viendo el aseguramiento de calidad como un conjunto de herramientas que son usadas para probar aplicaciones y componentes, solo implica que dichas herramientas ayudan a automatizar productos de pruebas, pero esto es sólo una parte de la función de QA denominada Control de calidad o pruebas, cuyo propósito es la medición del producto contra un estándar o atributo, identificando defectos y corrigiéndolos hasta tener un producto libre de defectos.

Lo antes planteado hace pensar que cuando se habla de aseguramiento de la calidad, se habla en un sentido general, presente en todo el proceso, pero no aterriza en algo específico para poder validar si esa calidad se está viendo reflejada o no en el proceso y por ende, en el producto resultante. Por esta razón, la próxima sección analiza todo lo concerniente con la Disciplina de Pruebas, que como disciplina, converge en actividades, entregables y procesos bien diferenciados y puntuales (más especializado). Allí se hace referencia a los autores que la defienden, conceptos y cuáles son los roles, fases, actividades, artefactos, entre otros, planteada por cada uno de ellos. Todo esto, con el fin dar la suficiente atención a dichos conceptos, destacar la importancia que se deriva de ellos y por ende, poder diferenciar los conceptos de aseguramiento de la calidad y disciplina de pruebas.

En la siguiente sección se agrupa todo lo concerniente a las pruebas, fin esencial de la disciplina de pruebas, modelos de calidad del producto para garantizar el aseguramiento de la calidad.

Autores	<i>Pressman (2005)</i>	<i>Sommerville (2006)</i>	<i>Unhelkar (2003)</i>	<i>Mendis (1993)</i>	<i>CMMI (2006)</i>	<i>ITIL V3 (2007)</i>
Criterios						
Nombre utilizado para definir el Aseguramiento de la Calidad (QA)	Garantía de la calidad del software	Administración de la calidad del software	Proceso de calidad del software	Evaluación de la Calidad del Software	Garantía de la calidad del proceso y del producto	Mejora Continua del Servicio de TI
Elección de estándares	Si contempla	Si contempla	Si contempla	Si contempla	Si contempla	No especifica
Revisión de Calidad	Si contempla (Auditoría de entregables)	Si contempla (Control de calidad)	Si contempla	No especifica	Si contempla (Evaluación continua calidad)	Si contempla
Registrar fallas	Si contempla	No especifica	Si contempla	Si contempla	Si contempla	Si contempla
Corrección de fallas	Si contempla	Si contempla	Si contempla	Si contempla	No especifica	Si contempla
Control y gestión de cambios	Si contempla	No especifica	No especifica en SQA sino en Admón. calidad	No especifica	Si contempla	Si contempla
Riesgos y su administración	Si contempla	Si contempla	No especifica	No especifica	Si contempla	Si contempla
Recopilación y análisis de métricas	Si contempla	Si contempla	Si contempla	No especifica	No especifica	No especifica
QA estadística	Si contempla	Si contempla	No especifica	Si contempla	No especifica	No especifica
Presente en todo el proceso de desarrollo	No especifica	Si contempla	Si contempla	No especifica	Si contempla	Si contempla
Especificación de Roles	Si contempla (grupo SQA)	Si contempla	Si contempla	Si contempla (todo el equipo)	Si contempla (asignación de responsabilidades)	Si contempla (en otra etapa, en SS)
Plan de SQA	Si contempla (Plan de aseguramiento de la Calidad)	Si contempla (Plan de Calidad)	Si contempla (Plan de calidad y Plan de pruebas)	Si contempla (IQUE)	Si contempla (Plan del proceso)	Si contempla (Plan y gestión de los Procesos)
Especificación Documentación	Si contempla	Si contempla	Si contempla	No especifica	Si contempla (típicos productos de trabajo)	Si contempla
Independencia team calidad vs desarrolladores	No especifica	No especifica	Si contempla	No especifica	Si contempla	Si contempla
Mejora de procesos	Si contempla	No especifica	Si contempla	Si contempla (continuamente)	Si contempla	Si contempla (continuamente)

Tabla 1- *Tabla Comparativa entre defensores del Aseguramiento de la Calidad* (Fuente: Elaboración propia).

III. Pruebas del Software

Las *pruebas* constituyen el proceso de encontrar diferencias entre el comportamiento esperado, especificado por los modelos del sistema y el comportamiento observado del sistema (Bruegge y Dutoit, 2002). Las pruebas son un intento de simulación del sistema con respecto a los modelos del mismo. Su objetivo es diseñar situaciones que manifiesten los defectos que hay en el sistema y que revelen los problemas. Esta actividad es contraria a todas las actividades del desarrollo de software, pues están orientadas al quebrantamiento del sistema, es decir, lograr que el sistema falle (demostrar la presencia de defectos) de forma planeada. Méndez et al. (2007) lo resume de la siguiente forma: Una diferencia interesante entre las Pruebas y otras disciplinas del proceso de desarrollo de software es que esencialmente es una tarea que encuentra y pone de manifiesto las debilidades del producto de software. Esto puede lograrse descartando ideas preconcebidas sobre la corrección del software que se acaba de desarrollar y se supere cualquier conflicto de intereses que aparezcan cuando se descubran los errores presentes (Pressman, 2005).

Probar es una actividad realizada para la evaluación de la calidad en productos y para mejorarla, mediante la identificación de los defectos y problemas. Este concepto toma fuerza sobre el hecho de que la disciplina de pruebas debe estar formalizada y organizada mediante una estrategia dentro de un proyecto de desarrollo de software, dentro de un alcance definido, para que debidamente organizado se logren los objetivos planteados. Esta planificación de pruebas y actividades de diseño en sí mismos, constituyen un aporte útil para los diseñadores al poder exponer las debilidades potenciales (como descuidos o contradicciones de diseño, omisiones o ambigüedades en la documentación).

En líneas generales, las pruebas de un software giran alrededor de los casos de prueba, que según Leffingwell y Widrig (2006) son esenciales para todas las actividades de prueba porque:

- *Son la base para diseñar y ejecutar los procedimientos de pruebas.*
- *La profundidad de las pruebas es proporcional al número de casos de pruebas.*
- *El diseño y desarrollo, y los recursos necesarios son gobernados por los casos de pruebas requeridos.*

Bajo esta premisa, los *casos de prueba* (especificaciones de un conjunto de entradas, condiciones de ejecución e identificación de resultados los resultados esperados producto de las combinaciones de los dos primeros elementos), determinarán el ambiente necesario para ser ejecutado, podrá demostrar defectos (o su contraparte, casos de éxito) al final del proceso de pruebas apoyándose o no en herramientas de pruebas que lo gestionen formalmente, dirigido a diversos tipos de prueba y técnicas aplicadas según corresponda, en los diversos niveles de prueba que sean determinados por el equipo del proyecto/software objeto de prueba mediante procedimientos (instrucciones) de prueba claramente establecidos y documentados siguiendo una estrategia de pruebas plasmada en un plan de prueba.

Profundizando algunos conceptos, se tienen las siguientes definiciones tomadas en su mayoría de RUP (2005).

Caso de Prueba: Define un conjunto de entradas de prueba, condiciones de ejecución, y resultados esperados, identificados con el objetivo de evaluar algunos aspectos particulares de un elemento destino de la prueba.

Caso de Uso: Una especificación de un tipo de interacción con el sistema. Define un conjunto de instancias de guión de uso, donde cada instancia es una secuencia de acciones que lleva a cabo un sistema que producen un resultado observable de valor para un actor concreto.

Plan de prueba: Define los objetivos de las pruebas en el ámbito de la iteración (o el proyecto) los elementos de destino, el enfoque que se adopta, los recursos necesarios y los entregables que se deben generar.

Scripts de Pruebas: Es un guión o conjunto de instrucciones paso a paso que realiza una prueba, habilitando su ejecución. Permiten automatizar tareas creando pequeñas utilidades.

Suite de Pruebas: Es un conjunto de Scripts que se agrupan con el objetivo de gestionar y secuenciar la ejecución de las pruebas ordenadamente.

Datos de prueba: Define una recopilación de valores de entrada de prueba que se consumen durante la ejecución de una prueba, y los resultados esperados que sirven de referencia para objetivos comparativos durante la ejecución de una prueba.

Condiciones: Descripción de una condición que ejercerá durante esta prueba.

Resultados de las pruebas: Resume el análisis de uno o más registros de prueba y solicitudes de cambio, proporcionando una valoración relativamente detallada de la calidad de los elementos destino de la prueba y el estado del esfuerzo de prueba.

Resultados esperados: Corresponde a las expectativas que se tienen de los resultados.

Resultados obtenidos: Corresponde a lo realmente obtenido como resultado. Si lo que se espera es igual al obtenido, entonces el caso de prueba se considera exitoso.

Ambiente de pruebas: Respecto a la aplicación a probar, se puede incluir el hardware sobre el que se ejecuta, el software necesario para interconectarse con otros sistemas o requerimientos de la aplicación propiamente dicho, la base de datos necesaria para operar la aplicación, entre otros entes específicos.

Herramientas de pruebas: Aquellas que se pueden utilizar para aumentar la eficacia de las pruebas.

Tipos de Prueba: Tienen un objetivo específico y una técnica de soporte. Cada tipo de prueba se centra en la prueba de uno o varios atributos o características del destino de la prueba. SWEBOK (2004) plantea que los casos de prueba pueden ser diseñados para comprobar que las especificaciones funcionales se apliquen correctamente, que son comúnmente conocidas como pruebas de conformidad, pruebas de precisión o pruebas funcionales. Sin embargo, muchas otras propiedades no funcionales pueden ser probadas tales como el rendimiento, fiabilidad y facilidad de uso, entre otros. En todo caso, el objetivo de la prueba varía de acuerdo a la aplicación objeto de prueba. De esta forma algunos tipos de prueba se clasifican según la dimensión de calidad en Funcionales, No funcionales, de la siguiente forma:

- *Dimensión de funcionalidad: Prueba de función, prueba de seguridad y prueba de volumen.*
- *Dimensión de utilidad: Pruebas de utilización, prueba de integridad, prueba de usabilidad.*
- *Dimensión de fiabilidad: Prueba de estructura y de tensión.*
- *Dimensión de rendimiento: Prueba de puntos de referencia, Prueba de contienda, Prueba de carga y perfil de rendimiento, prueba back to back.*
- *Dimensión de Capacidad de soporte: Prueba de configuración, prueba de instalación, pruebas de recuperación.*
- *Otras: Pruebas alpha y beta, pruebas de regresión,*

Niveles de Pruebas: Las pruebas se aplican a diferentes tipos de destinos, en fases o niveles diferentes de esfuerzo de trabajo. Estos niveles suelen distinguirlos los roles que están más capacitados para diseñar y dirigir las pruebas. Los niveles de prueba

son conocidos como: Prueba de desarrollador (unitarias), Pruebas integrales, Pruebas certificadas y Pruebas del sistema (aceptación del usuario).

Técnicas de Pruebas: Normalmente las técnicas de pruebas se clasifican en Caja blanca (que proporcionan información de cómo el software fue diseñada y codificada) y Caja negra (que se centran solo en el comportamiento de entrada y salida del software). Sin embargo, SWEBOK (2004) plantea que es difícil encontrar una base homogénea para clasificar las técnicas de pruebas, pero plantea una clasificación que se basa en el cómo se generan los casos de prueba y se presenta a continuación:

- a) *Intuición y experiencia de los ingenieros de software.*
 - a. *Pruebas Ad hoc*
 - b. *Pruebas exploratorias*
- b) *Las especificaciones del software*
 - a. *Partición de equivalencia*
 - b. *Análisis del valor frontera*
 - c. *Tabla de decisión*
 - d. *Máquinas basadas en estados finitos*
 - e. *Pruebas desde especificaciones formales*
 - f. *Pruebas aleatorias*
- c) *La estructura del código*
 - a. *Criterio basado en el flujo de control*
 - b. *Criterio basado en el flujo de datos*
 - c. *Modelos de referencia para pruebas basadas en código*
- d) *Técnicas basadas en fallos (reales o artificiales-simuladas)*
 - a. *Adivinar el error*
 - b. *Pruebas de mutación*
- e) *La naturaleza de la aplicación*
 - a. *Pruebas orientadas a objetos*
 - b. *Pruebas basados en componentes*
 - c. *Pruebas de GUI*
 - d. *Pruebas de programas concurrentes*
 - e. *Pruebas de protocolo de conformidad*
 - f. *Pruebas de sistemas en tiempo real*
 - g. *Pruebas de sistemas de seguridad crítica*

IV. Disciplina de Pruebas

Los modelos de ciclo de vida del software hacen que las actividades para su desarrollo y sus dependencias sean visibles y manejables. Una *disciplina* es una colección de actividades correspondidas que a su vez se relacionan con un “motivo de preocupación importante” dentro del proyecto total donde estén enmarcadas (RUP, 2005). Ésta colección o agrupación es frecuentemente usada por RUP, quien concentra actividades en disciplinas como una forma de ayudar a entender el proyecto

desde una perspectiva ‘tradicional’, es decir, una secuencia de actividades ordenadas que deben cumplirse.

La disciplina de pruebas en el software, está fuertemente ligada a la calidad y los esfuerzos que se puedan ejecutar para alcanzarla, garantizando que los procesos y productos resultarán con los niveles esperados de calidad. Su importancia radica en que un acercamiento continuo a la calidad, iniciada temprano en el ciclo de vida del software, puede aminorar los costos de terminar y mantener el software significativamente. Esto reduce altamente el riesgo asociado de implantar software de mala calidad (RUP, 2005).

Kruchten (2000) afirma que la disciplina de pruebas en un proceso iterativo que ocurre en todas las fases del ciclo de vida, permitiendo tener una retroalimentación temprana sobre la calidad del producto que será usada para mejorar el producto tanto en su diseño como en su construcción. Esto evidencia el hecho de que probar no es solo una actividad final del ciclo de vida que está destinada a aprobar o rechazar el sistema desarrollado, sino que se debe abordar desde la fase de inicio del desarrollo de software. Por su parte, involucrando el capital humano que participa en la disciplina de pruebas, Dustin (2003) apoya el hecho de iniciar pruebas desde etapas tempranas incluyendo a los probadores quienes deben participar desde el comienzo de un ciclo de vida del proyecto, para que puedan comprender exactamente lo que están probando y puedan trabajar en unión con los stakeholders² para crear requerimientos comprobables.

La disciplina de pruebas actúa como abastecedor de servicios a las otras disciplinas en muchos aspectos. Las pruebas se centran sobre todo en la evaluación o evaluar la calidad del producto, que es lograda con estas prácticas base (RUP, 2005):

- *Encontrar y documentar los defectos en calidad del software.*
- *Aconsejar sobre la calidad percibida del software.*
- *Validar y probar las asunciones hechas en especificaciones del diseño y del requerimiento a través de demostración concreta.*
- *Validar que el producto de software trabaja según lo diseñado.*
- *Validar que los requerimientos están implementados apropiadamente.*

² **Stakeholders:** Cualquier persona o representante de una organización quien tiene un interés creado en los resultados de un proyecto o cuya opinión debe ser complacida. (Kruchten, 2000).

Diferentes metodologías y tendencias en el desarrollo de software (RUP, XP (eXtream Programing), MFS, Eclipse EFP, ITIL, entre otros) hacen uso de la disciplina de pruebas identificando actividades y roles entre otros, por lo que se evidencia la gran importancia del tema en el proceso de desarrollo de software.

Alguna de las tendencias más actualizadas en pruebas se conoce como Desarrollo dirigido por pruebas *TDD* (Test Driven Developmet o Test Driven Design, por sus siglas en ingles), técnica de desarrollo de software que consiste en dos prácticas: a) Escribir los casos de pruebas (para pruebas unitarias) primero antes que el código que implementará dicho caso de prueba con el fin de que pase con éxito y b) Refactorizar (mejorar el diseño del código, elevando la flexibilidad y tolerancia al cambio, entregando mas rápido y con menos depuración) incrementalmente. Esta es una de las prácticas clave en la Programación extrema, pero no necesariamente se implementa junto a ella (Astels, 2003) y su objetivo es “código limpio que funciona”. Los demás tipos de pruebas, en sus diferentes niveles, no se reemplazan, solo se complementan.

Back (2003), considerado uno de los precursores de TDD, dice que éste gira en torno a dos claves: 1) Nunca escribir una sola línea de código a menos que haya un defecto en las pruebas automatizadas. 2) Eliminar la duplicación. Bajo este concepto de ideas propone el siguiente ciclo de desarrollo basado en TDD.

1. Escribir la prueba
2. Escribir el código
3. Ejecutar las pruebas automatizadas
4. Refactorización
5. Repetición

Ambler (2007), establece que sin el uso de herramientas automatizadas, implementar TDD es prácticamente imposible. Esta se constituye como una de sus principales desventajas, pues no es aplicable a todos los sistemas o para aquellos que usen pruebas unitarias manuales. Sin embargo, aunque es una especificación técnica, un valioso efecto secundario es que provee de resultados significativamente mejores que los que lo hacen pruebas de código de las técnicas tradicionales.

Un enfoque interesante en la disciplina de pruebas, es presentado en el Perfil de Pruebas de UML (UML Testing Profile) que define un lenguaje para el diseñar,

visualizar, especificar, analizar, construir y documentar los artefactos en las pruebas de sistemas en varios dominios (OMG, 2005). Está basado en la especificación UML 2.0, uniendo el meta-modelo UML y reusando la sintaxis UML, extendiendo a UML con conceptos específicos en pruebas agrupados en 4 categorías: Arquitectura de prueba (que contiene los elementos que participan en una prueba y sus relaciones, conceptos de estructura y configuración de las pruebas), Datos de prueba (estructuras y significado de los valores que serán procesados en una prueba), Comportamiento de pruebas (que se ocupa de las observaciones y actividades durante una prueba) y tiempo (conjunto de conceptos relativos al tiempo con el fin de cuantificar las ejecuciones de pruebas).

Este perfil está enfocado a una particular tecnología de pruebas llamada “Pruebas funcionales de Caja negra”, permitiendo la captura de toda la información necesaria para el enfoque de caja negra y así evaluar la correcta implementación del sistema. El perfil de pruebas UML se destina a apoyar de una manera efectiva, eficiente, en la medida posible, las pruebas automatizadas del sistema en función de las implementaciones de los modelos computacionales de UML (Cavarra, 2002).

Este perfil constituye una iniciativa importante en la formalidad de la disciplina de pruebas dentro del proceso de desarrollo de software, bajo el enfoque de la validación de sistemas de tipo caja negra (solo entradas/salidas), pues detalla las notaciones, documentación y pasos que deben cumplirse al momento de probar sistemas. Algunos tecnicismos son aplicables a proyectos basados en la notación UML y apoyan la automatización de pruebas, además de que puede acoplarse con un marco de trabajo como JUnit³ usado dentro de la popular programación extrema (XP). Sin embargo, es importante notar el amplio alcance en los diferentes niveles de pruebas, puesto que se pueden especificar casos de prueba a nivel de unidad, integración, sistema y a nivel de la organización.

Otra tendencia interesante es Microsoft Solutions Framework (MSF por sus siglas en inglés) que es una flexible e interrelacionada serie de conceptos, modelos y mejores prácticas de uso que controlan la planificación, el desarrollo y la gestión de

³ JUnit: Marco de trabajo para que los desarrolladores realicen pruebas unitarias en Java inicialmente, pues ya se ha traducido a varios lenguajes.

proyectos tecnológicos, centrándose en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas (Microsoft, 2003). MSF es una serie de modelos y disciplina que puede adaptarse a cualquier proyecto de tecnología de información. Uno de los modelos es el “Modelo de procesos” el cual es separado en cinco principales fases, como se muestra en el Figura 9, y cuya función principal es establecer el orden por medio del cual, las diversas actividades del proyecto serán ejecutadas.

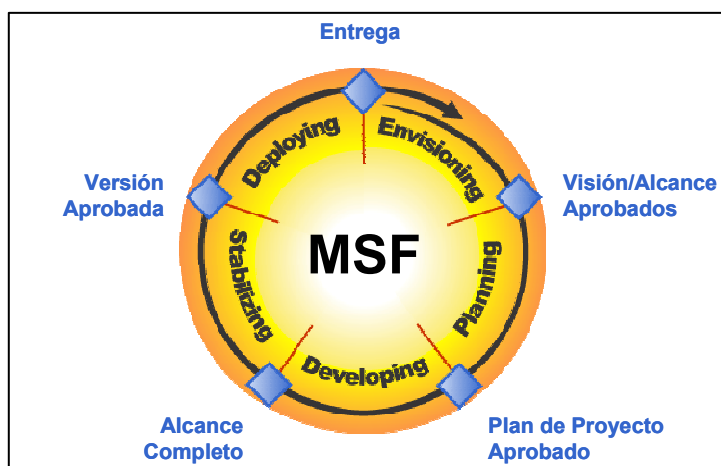


Figura 9 - Modelo de Proceso de MSF para el desarrollo de aplicaciones, Fuente: (Microsoft, 2003)

En todas las fases de este modelo de proceso se introduce la disciplina de pruebas, en menor o mayor proporción, gracias al “Rol de prueba” que se encarga de enfocar la disciplina de pruebas de acuerdo a las necesidades del proyecto en cada fase. En la *fase de visión*, el “rol de prueba” se enfoca en identificar las estrategias de pruebas, los criterios de aceptación y manejar las implicaciones que se derivan de la disciplina. En la *fase de planeación*, se realiza una evaluación del diseño y se determinan los requerimientos de pruebas y el plan de pruebas. En la *fase de desarrollo*, se dan los inicios de algunos entregables de pruebas como las especificaciones de pruebas y los casos de prueba; aquí, la aplicación ya está lista para pruebas externas y para su estabilización. El “Rol de pruebas” se enfoca en pruebas funcionales, identificación de defectos, documentación de pruebas y actualización del plan de pruebas. En la *fase de estabilización* se conducen las pruebas sobre la solución, las pruebas de esta etapa enfatizan el uso y operación bajo condiciones realistas. El equipo se enfoca en priorizar y resolver errores y preparar la solución para el lanzamiento. Es común que se reporten los errores del sistema rápidamente para que los desarrolladores los corrijan; se usan indicadores estadísticos conocidos como “bug convergente” y “zero-bug bounce” que ayudan al equipo del proyecto a

determinar cuando la aplicación está suficientemente estable. Igualmente, se usan técnicas como prueba completa pre-producción (se enfoca en preparar un release piloto) y prueba completa de aceptación del usuario (donde se determina que el sistema está habilitado y cumple satisfactoriamente con las necesidades del usuario y del negocio). En esta fase se especifican los siguientes entregables: Resultado de pruebas y herramientas de pruebas, enfocándose el “Rol de pruebas” en probar, reportar errores y estatus y en la configuración de pruebas. En la *fase de despliegue* el “Rol de pruebas” se enfoca en hacer pruebas de desempeño sobre la aplicación.

Dentro de las tendencias del código abierto, puede detallarse la comunidad Eclipse⁴ cuyos proyectos se centran en la construcción de una plataforma de desarrollo compuesta por frameworks extensibles, herramientas y módulos de ejecución para la construcción, desarrollo y gestión de software en todo el ciclo de vida (Eclipse, 2008). Dentro de sus múltiples proyectos, esta el Proyecto de Tecnología, que incluye al Eclipse Process Framework (EPF, por sus siglas en inglés), el cual tiene por objeto elaborar un marco de trabajo para el proceso de ingeniería de software personalizado, proveyendo herramientas, un metamodelo unificado y contenidos que puedan ser usados como base para apoyar una amplia variedad de tipos de proyectos y estilos de desarrollo, dirigido principalmente al equipo del proyecto y al equipo de gerencia del proyecto. El primer entregable de este proyecto, está constituido por OpenUP que es un proceso inspirado en la tendencia ágil, cuyas raíces son el Proceso Unificado, cuyo propósito es ser un release simple de una aplicación de software construido por un grupo relativamente pequeño con fácil acceso para los stakeholders. El objetivo principal en el desarrollo de OpenUP era crear un ligero proceso que pueda ampliarse fácilmente para satisfacer las necesidades de los proyectos y organizaciones que necesitan más procesos (Eclipse, 2008).

OpenUP incorpora un número de prácticas ágiles dentro del contexto de un interactivo e incremental ciclo de vida del desarrollo de software. Por ser un ciclo de vida, se distinguen cuatro fases, como se muestra en la Figura 10, con hitos asociados y una plantilla de iteración para cada una de las fases que describen las actividades clave.

⁴ La comunidad Eclipse consta de individuos y organizaciones provenientes de una sección de la industria del software.

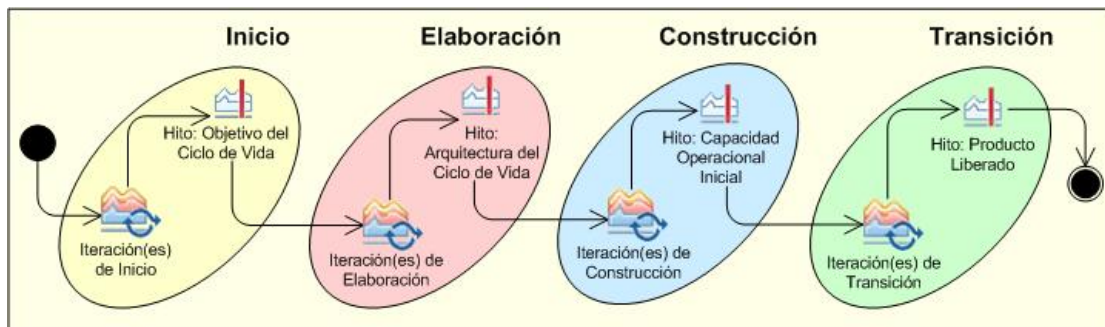


Figura 10- Fases del ciclo de vida del desarrollo de software, Fuente: (Eclipse, 2008).

Dentro de este ciclo de vida, OpenUP ubica dichas tareas que se realizan simultáneamente a través de varias disciplinas, como una forma de organizar simple y efectivamente los contenidos, entre las que se encuentra la Disciplina de Pruebas, la cual explica como proveer feedback acerca de la maduración del sistema mediante el diseño, implementación, ejecución y evaluación de pruebas, aplicando la estrategia de “probar temprano y probar frecuente” con el propósito de menguar riesgos lo más temprano posible en el ciclo de vida. Esta disciplina está constituida por tres tareas (a) crear casos de prueba, (b) implementar scripts de pruebas y (c) ejecución de pruebas, y siete elementos de orientación, un rol principal llamado “probador”, que se encarga de las actividades básicas del esfuerzo en pruebas, dentro de las que están: identificación, definición, ejecución y realización de las pruebas necesarias, así como el registro de los resultados de las pruebas y el análisis de los resultados, y algunos productos asociados al trabajo inherentes a la disciplina, como los casos de prueba, log de pruebas y scripts de pruebas.

Resulta interesante indagar el propósito de EPF y relacionarlo con la investigación en curso, en cuanto a la intención de poder personalizar mediante marcos de trabajo, herramientas y estrategias que proporcionen soluciones adaptadas a diferentes tipos de proyectos de software, tomando en cuenta sus propias limitaciones y alcances, escalable a sus propios requerimientos. El modelo de referencia que se propondrá, deberá tener este propósito de poder ser implementado en numerosas organizaciones, buscando adaptarlo a sus propias necesidades.

A la par del proceso unificado, se encuentra el proceso de ingeniería de software que proporciona un acercamiento disciplinado para asignar tareas y responsabilidades dentro de una organización de desarrollo, llamado Rational Unified

Process (RUP, por sus siglas en ingles), constituido por dos dimensiones: Eje Horizontal (dinámico y representa el tiempo y los aspectos del ciclo de vida del proceso) y el Eje Vertical (la descripción del dinamismo en términos de componentes de proceso, disciplinas, actividades, workflows, artefactos, y de roles), como se muestra en la Figura 11 (RUP, 2005).

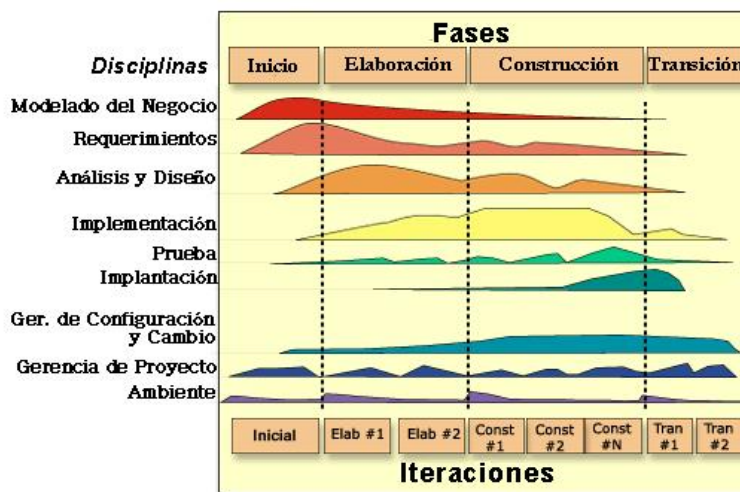


Figura 11 - Vista general del Proceso Unificado de Rational, Fuente: (RUP, 2005).

Como complemento de lo ya expresado anteriormente, RUP define claramente los roles, actividades e hitos importantes para la Disciplina de pruebas, que como puede observarse en la figura anterior, está presente en gran parte de la vida del proyecto y se interrelaciona con las demás disciplinas. La Disciplina de pruebas según RUP, define 4 roles especiales que son: Gerente de pruebas, analista de pruebas, diseñador de pruebas y el probador, quienes realizan una serie de actividades específicas de acuerdo al rol, entre las que destacan: acordar la misión, identificar los motivadores de pruebas, obtener compromiso para probar, estimar y mejorar el esfuerzo de pruebas, identificar requerimientos e ideas de pruebas, determinar los resultados de las pruebas, definir el enfoque de pruebas, definir las configuraciones de los ambientes de pruebas, identificar los mecanismos para probar, definir los elementos a probar, implementar las pruebas con la respectiva suite de pruebas, ejecutar las pruebas y analizar las fallas de pruebas. Todas estas actividades se apoyan en los artefactos (entregables) donde intervienen los roles, siendo alguno de ellos: el plan de pruebas, resumen de la evaluación de pruebas, suite de pruebas, log de pruebas, lista de ideas de pruebas, casos de prueba, datos de prueba, resultados de pruebas, configuración del ambiente de pruebas, arquitectura de automatización de pruebas.

RUP es ampliamente utilizado por más y más proyectos de desarrollo de software, para dirigir el esfuerzo en pruebas, apoyándose en herramientas que permitan la automatización de pruebas y su gestión.

Por su parte, ITIL en su fase de Transición del Servicio (ST-Service Transition, por sus siglas en inglés) provee orientación sobre la administración del cambio del portafolio de servicios y a través del ciclo de vida de los servicios. Aquí se incluyen algunos tópicos importantes que ubican algunas áreas de importancia, en la cual se incluye el Servicio de pruebas y validación, que junto con el soporte temprano (Early Life Support⁵, por sus siglas en inglés) garantizan que los servicios son aptos para el propósito y uso al que se destinará el producto, de conformidad con los requisitos de servicio y de las limitaciones especificadas (Lacy y Macfarlane, 2007).

Este proceso de servicio de pruebas y validación, tiene el propósito de asegurar que las versiones desarrolladas y los servicios resultantes satisfagan las expectativas de los clientes, y a su vez, verificar que las operaciones de TI son capaces de apoyar el nuevo servicio (Kempter, 2008). Identifica los siguientes roles: gerente de pruebas (quien es el propietario del proceso), gerente de cumplimiento, cliente, gerente de seguridad TI, gerente de versiones, gerente de nivel de servicio y el usuario. Algunos sub-procesos (actividades) dentro de esta rama, son: definición del modelo de pruebas, servicio de validación del diseño, versión de componente de adquisición, versión de pruebas y el servicio de aceptación de pruebas.

Con todo lo anterior, se destaca la importancia que tiene la Disciplina de pruebas para crear software de calidad, calidad que no se logra instantáneamente, sino que dependerá en gran medida del proceso de aseguramiento de la calidad que transitó a la par con la creación de este producto.

En el mismo marco de ideas, una realidad latente es que hay varios factores que han hecho de las Pruebas de software y de sistemas una necesidad: por un lado la creciente complejidad, heterogeneidad y variabilidad de las tecnologías, que

⁵ Soporte provisto para un Servicio TI nuevo o cambiado para un periodo de tiempo, posterior a su Entrega. (ITIL® Glossary v01, 2006).

incrementa el riesgo. Por otro lado la presión del *time to market* y también la madurez del mercado (García y Encuera, 2007). He aquí otro gran paso para dar la importancia que merece la disciplina de pruebas en las empresas desarrolladoras de software, especialmente las PYMEs, que en su interés de ganar presencia en el mercado nacional, deben apalancarse en calidad de los servicios prestados.

Esta importancia, también es indicada por Grimán et al. (2003), al afirmar que una disciplina importante en el proceso de desarrollo de los Sistemas de Software es la relativa a Pruebas. Esta disciplina, generalmente no se implementa de forma organizada y sistemática. Mecanismos para garantizar esta formalidad dentro de las empresas desarrolladoras de software, es lo que se plantea en la investigación en progreso y se constituye como la evidencia de concientizar esta cultura de pruebas en las PYMEs Venezolanas. Parte de las razones de esta evidencia, se mostraron en la sección “Marco Organizacional” del presente Capítulo, al contextualizar las PYMEs desarrolladoras de software venezolanas.

La siguiente sección hace un breve recorrido sobre los principales modelos de calidad del software que existen, profundizando en el modelo elegido para esta investigación.

V. Modelos de Calidad

Uno de los principales problemas a los que se enfrenta la calidad del software es el siguiente: ¿Es realmente posible encontrar un conjunto de propiedades en un software que sea una indicación de su calidad?. Para dar respuesta a esta pregunta aparecen los Modelos de Calidad, que no son más que las directrices o punto de referencia para imitarlo o reproducirlo; es por ello que para los efectos de esta investigación, se homologa el término modelo de referencia a modelo de calidad, pues el segundo finalmente es un modelo de referencia a seguir.

Como se ha dicho en secciones anteriores, la calidad puede describirse desde diferentes perspectivas y en el desarrollo del software hay dos especiales y tradicionalmente importantes: la calidad del producto en sí y la calidad del proceso para obtenerlo con un enfoque sistémico. Contar con estándares y/o modelos de

calidad para garantizar calidad en los productos de software, es de suma importancia para las organizaciones que quieran diferenciarse dentro de su entorno competitivo.

En lo que se refiere a calidad del producto la norma ISO/IEC 25000 proporciona una guía para el uso de las nuevas series de estándares internacionales, llamados Requisitos y Evaluación de Calidad de Productos de Software (SQuaRE), las cuales constituyen una serie de normas basadas en la ISO 9126 y en la ISO 14598 (Evaluación del Software), siendo su objetivo principal guiar el desarrollo de los productos de software con la especificación y evaluación de requisitos de calidad (Piatinni et al., 2007).

La norma ISO 9126 propone un modelo de calidad que se divide en tres vistas: interior, exterior y en uso que están compuestas por características, que se dividen en subcaracterísticas y que estas a su vez se componen de atributos. Los atributos obtienen sus valores tras realizar mediciones sobre el software que pueden usarse como una lista de comprobación de aspectos relacionados con la calidad. A continuación se presenta un cuadro resumen con los conceptos propuestos por éste estándar de calidad en sus perspectivas (Ver Tabla 2).

Para encontrar la calidad con la perspectiva del proceso, se pueden nombrar algunos modelos como lo son: CMM, PSP, Bootstrap, Spice, ISO 9000, IDEAL. De igual manera para la perspectiva del personal (la cual no se puede obviar según Voas, 1998) están los modelos de PSP, TSP, P-CMM. En virtud de esta investigación, se profundizará en el modelo de calidad del producto de software puesto que se evaluarán herramientas de pruebas presentes en el mercado constituidas como producto de software (concluido).

Cabe destacar que el MOdelo Sistémico de CALidad MOSCA, toma en cuenta las tres perspectivas de calidad del software que son Producto, Proceso y Personal, como se puede ver en la sección de ANEXOS, denotado como Anexo I. MOSCA para cada una de las perspectivas está constituido por cuatro 4 niveles que se detallan a continuación: (Mendoza *et al.*, 2005).

- **Nivel 0 Dimensiones:** Abarcadas en cada perspectiva. Estas son el aspecto contextual e interno del producto, del proceso y de lo humano.

Vista	Características	Capacidad del Producto de Software para...	Sub-característica	Capacidad del Producto de Software para...
Interna y externa	FUNCIONALIDAD	<i>Proporcionar funciones que satisfacen necesidades declaradas e implícitas cuando se usa bajo condiciones específicas</i>	Adecuación	Proveer funciones apropiadas según tareas y objetivos
			Exactitud	Proveer resultados o efectos correctos o acordados
			Interoperabilidad	Interactuar con uno o más sistemas especificados
			Seguridad de acceso	Proteger datos e información de personas/sistemas no autorizados
			Cumplimiento Funcional	Adherirse a normas, convenciones o regulaciones realcionadas con FUN.
	FIABILIDAD	<i>Mantener un nivel especificado de prestaciones bajo condiciones específicas</i>	Madurez	Evitar fallar como resultado de fallas en el software
			Tolerancia a fallos	Mantener un nivel especificado ante fallas del SW o interfaces
			Capacidad de recuperación	Restablecer un nivel de prestaciones y recuperar datos afectados
			Cumplimiento Fiabilidad	Adherirse a normas, convenciones o regulaciones realcionadas con FIA.
	USABILIDAD	<i>Ser entendido, aprendido, usado y atractivo cuando se usa bajo condiciones específicas</i>	Ser entendido	Entender si es adecuado y cómo puede ser usado particularmente
			Ser aprendido	Permitir el aprendizaje fácil por el usuario
			Ser operado	Permitir al usuario operarlo y controlarlo
			Interfaz gráfica	Ser atractivo al usuario
			Cumplimiento Usabilidad	Adherirse a normas, convenciones o regulaciones realcionadas con USA.
	EFICIENCIA	<i>Proporcionar prestaciones apropiadas, relativas a cantidad de recursos usados, bajo condiciones específicas</i>	Comportamiento temporal	Proporcionar tiempos de respuesta, proceso y potencia apropiados
			Utilización de recursos	Usar cantidades y tipos de recursos adecuados bajo ciertas condiciones
			Cumplimiento Eficiencia	Adherirse a normas, convenciones o regulaciones realcionadas con EFI.
	MANTENIBILIDAD	<i>Ser modificado (correcciones, mejoras, adaptaciones a cambios del entorno, requisitos y especificaciones funcionales</i>	Capacidad de análisis	Diagnosticarle deficiencias o causas de fallos o identificar modificaciones
			Capacidad de cambio	Permitir que se implemente una determinada modificación
			Estabilidad	Evitar efectos inesperados por modificaciones del SW
			Capacidad de ser probado	Permitir que el SW modificado sea validado.
	PORTABILIDAD	<i>Ser transferido de un entorno a otro</i>	Cumplimiento Manetenibi	Adherirse a normas, convenciones o regulaciones realcionadas con MAN.
			Adaptabilidad	Adaptarse a diferentes entornos especificados sin aplicar muchos cambios
			Instalabilidad	Instalarse en un entorno espeificado
			Coexistencia	Coexistir con otro SW independiente en entorno con recursos comunes
			Capacidad de reemplazo	Usarlo en lugar de otro producto SW con el mismo propósito y entorno
En uso	Efectividad	<i>Permitir a los usuarios alcanzar objetivos con exactitud y completitud</i>		
	Productividad	<i>Permitir a los usuarios gastar la cantidad apropiada de recursos en relación a la efectividad obtenida.</i>		
	Seguridad de uso	<i>Alcanzar niveles aceptables de riesgo en personas, negocio, SW, propiedades o medio ambiente</i>		
	Satisfacción	<i>Cumplir con las expectativas de los usuarios en un contexto determinado</i>		

Tabla 2- Modelo de Calidad según ISO 9126. Fuente: Elaboración propia

- **Nivel 1 Categorías:** Definidas para cada perspectiva. **Producto:** funcionalidad (FUN), fiabilidad (FIA), usabilidad (USA), eficiencia (EFI), mantenibilidad (MAB) y portabilidad (POR); estas están basadas en las normas internacionales ISO 9126 que se establecen como obligatorias para garantizar la calidad del producto de software. **Proceso:** cliente –proveedor (CUS), ingeniería (ENG), soporte (SUP), gestión (MAN) y organizacional (ORG). **Humana:** individual (IND), equipo (EQU) y entorno empresarial (ENT).
- **Nivel 2 Características:** Asociadas a cada categoría, las cuales definen las áreas claves a satisfacer para lograr, asegurar y controlar la calidad en las tres perspectivas.

Son 56 características asociadas al producto, 27 al proceso de desarrollo y 15 a la parte humana.

- **Nivel 3 Métricas:** Utilizadas para medir la calidad sistémica. Son propuestas en total 715 métricas.

Una vez definidos los niveles, se procede a evaluar a través del modelo, la calidad del software usando un algoritmo ampliamente probado propuesto por Mendoza *et al.* (2005). Finalmente el algoritmo detalla que el nivel de calidad del producto de software se relacionan con las categorías satisfechas luego de la evaluación de acuerdo a la Tabla 3, siendo que si la categoría Funcionalidad no es satisfecha, el algoritmo finaliza y la calidad del producto será nula.

Funcionalidad	Segunda categoría evaluada	Tercera categoría evaluada	Nivel de calidad del producto de software
<i>Satisfecha</i>	<i>No satisfecha</i>	<i>No satisfecha</i>	Básico
<i>Satisfecha</i>	<i>Satisfecha</i>	<i>No satisfecha</i>	Intermedio
<i>Satisfecha</i>	<i>No satisfecha</i>	<i>Satisfecha</i>	Intermedio
<i>Satisfecha</i>	<i>Satisfecha</i>	<i>Satisfecha</i>	Avanzado

Tabla 3 - Nivel de calidad del producto con respecto a las categorías satisfechas para el producto.

FUENTE: Mendoza et al. (2005).

Esta investigación se basa en los conceptos antes desarrollados y su enfoque está dado al modelo de calidad MOSCA por ser el más idóneo al momento de evaluar herramientas de pruebas, demostrado por múltiples aplicaciones del modelo.

En la siguiente sección, se abordarán las bases éticas y legales sobre la que se sustenta esta investigación.

VI. Herramientas de Pruebas

La calidad en el software que se espera como un producto final, bien puede afianzarse con la disciplina de pruebas dentro del proceso de desarrollo de software, que a su vez se apoya en herramientas tecnológicas diseñadas con propósitos especiales para cubrir estas necesidades. Las *herramientas de pruebas* permiten en gran medida coordinar, dirigir y abastecer los diversos entregables, actividades y roles que surgen de la aplicabilidad de la disciplina de pruebas, y que por su envergadura imposibilitan los procesos manuales para la obtención de los resultados esperados. Braude (2003) expresa que el gran volumen de las pruebas suelen requerir herramientas de prueba automatizadas y se sugiere que al menos el 75% de las

pruebas sean automatizadas y el resto manual. Las herramientas de automatización ejecutan únicamente scripts diseñados para testear un requisito o funcionalidad específicos y no poseen la habilidad de “toma de decisiones” ni grabación de discrepancias no incluidas en el script. Es por ello que es necesario realizar pruebas manuales a lo largo de toda la aplicación, antes de realizar la automatización de actividades.

En el mercado, existen numerosas herramientas de pruebas con propósitos tan diversos y en algunos casos, tan distantes, que tener criterios válidos para su selección se vuelve una actividad compleja. En todo caso, siempre es necesario definir el propósito para lo cual se requiere y el alcance que las mismas tendrán en el proceso de desarrollo de software (dirigido a su implementación) para intentar hacer la mejor elección. Estas herramientas pueden ser usadas para la planificación de pruebas, diseño de pruebas funcionales y no funcionales, automatización de pruebas, entre otras aplicaciones, lo cual hacen que el tema de las pruebas se vuelva tan especializado y a la vez postergado; para ambos casos se necesitan medidas de acción oportunas. RUP (2005) plantea esta realidad: *“Existen una serie de herramientas de automatización, pero es poco probable que una sola herramienta pueda automatizar todas las tareas de prueba. La mayoría de las herramientas se centran en una tarea o grupo de tareas específicas, aunque algunas sólo tratan un aspecto de la tarea”*. Las pruebas automatizadas son efectivas en entornos donde los cambios son frecuentes o en aplicaciones en las que se esperan builds y releases críticos.

Por su parte, SWEBOK (2004) clasifica las herramientas de pruebas en estos grandes grupos:

- a) *Generadores de prueba, para ayudar en el desarrollo de casos de prueba.*
- b) *Marcos de trabajo de ejecución de pruebas que permiten la ejecución de los casos de prueba en un ambiente controlado donde se observa el comportamiento de la aplicación objeto de prueba.*
- c) *Herramientas de evaluación de pruebas, herramientas de apoyo a la evaluación de los resultados de ejecución de pruebas, determinando si el comportamiento observado se ajusta al comportamiento esperado.*
- d) *Herramientas de gestión de pruebas las cuales proporcionan apoyo para todos los aspectos del proceso de pruebas del software.*
- e) *Herramientas de análisis del rendimiento, utilizadas para medir y analizar el performance del software objeto de pruebas, la cual se corresponde con una forma especializada de probar donde la meta es evaluar el comportamiento del desempeño más que el funcional.*

Braude (2003) especifica las siguientes capacidades que deben encontrarse en las herramientas automatizadas de pruebas de sistemas, las cuales se tomarán en cuenta para el modelo de referencia propuesto:

1. *Registrar acciones de pantalla (ratón y teclado) para permitir reproducción repetida*
2. *Correr varias veces los textos de prueba*
3. *Activar registro de resultados de prueba*
4. *Registrar tiempo de ejecución*
5. *Registrar errores de la corrida (ejecución)*
6. *Crear y administrar pruebas de regresión*
7. *Generar informes de pruebas*
8. *Generar datos de prueba*
9. *Registrar uso de memoria*
10. *Administrar casos de pruebas*
11. *Analizar la cobertura*

Puede que la selección de dichas herramientas sea compleja, pero ya existen las herramientas; esto se constituye como un gran paso para crear la cultura de la Disciplina de pruebas, apoyada en herramientas. Sin embargo, seleccionarlas y evitar que entren en desuso, es parte de la cultura en progreso que debe afianzarse en las PYMEs, a través de mecanismos innovadores y de comunicación exitosos entre quienes asumen los roles dentro de la Disciplina de Pruebas.

2.4 Bases éticas y legales

En una sociedad de la información tan avanzada como la nuestra, recalcar los aspectos éticos que den base a la investigación resulta obligado, entendiéndose la ética en informática como un conjunto de normas que guían al personal involucrado en la informática y a las personas que se relacionan con el mundo de las nuevas tecnologías (Carretero, 2006).

Esta investigación dentro del marco ético respeta la propiedad intelectual y los derechos de los autores aquí citados, atribuyéndoles las ideas propias creadas por un individuo y reconociendo las contribuciones a la ciencia que han realizado los mismos. Cuando se haga uso de alguna conclusión, método, estrategia o herramienta de prueba, será claramente identificado el autor de dicho aporte, ya sea individuo, organismo o empresa, y se usará con propósitos netamente académicos.

Dado que el trabajo de investigación está enmarcado dentro de un proyecto del FONACIT⁶, el uso y derecho sobre el conocimiento generado está regido por el marco legal que el mismo FONACIT establece, entre las que destacan la Ley Orgánica de Ciencia, Tecnología e Innovación, el Decreto del uso del software libre en Administración Pública y demás regulaciones definidas por este.

La presente investigación asume que el marco regulatorio de las PYMEs en Venezuela, viene dado por el Código de Comercio (en cuanto a la actividad de creación y funcionamiento de una PYME) y el Decreto con fuerza de Ley para la promoción y desarrollo de la pequeña y mediana industria del 2001, según Bastidas (2003).

⁶ Ver sección de Alcances y Limitaciones.

CAPÍTULO III: MARCO METODOLÓGICO

Este capítulo tiene como objetivo principal especificar el marco de trabajo metodológico utilizado para la realización de esta investigación. Primeramente se ubica el proyecto dentro de un tipo específico de investigación. Seguidamente, se identifica el diseño de la investigación, basada principalmente en el aporte de Pérez et al. (2004) adaptado en el Laboratorio de Investigación en Sistemas de Información (LISI) y que rige los aspectos metodológicos de la investigación. Posteriormente, dentro del mismo diseño se presenta el enfoque basado en el programa Objetivos, Preguntas y Métricas (GQM, por sus siglas en inglés) que es utilizado para la elaboración de los indicadores que se usarán para medir el modelo de referencia propuesto para la selección de herramientas que apoyen la disciplina de pruebas.

3.1 Tipo de Investigación

La presente investigación, según su naturaleza y propósito está enmarcada dentro del tipo de investigación aplicada, la cual desarrolla productos para satisfacer necesidades, implanta acciones para mejorar o evalúa la adecuación de medios y fines (Yáber y Valerino, 2006). Con el presente proyecto se investiga, elabora y desarrolla una propuesta de un modelo de referencia/modelo de calidad (con su respectivo fundamento teórico, ejecución y evaluación) para PYMEs desarrolladoras de software, con el fin de solventar las necesidades de las mismas en el ámbito de la Disciplina de pruebas, es decir, ofrecerles un método o procedimiento para la selección de herramientas de pruebas con las que puedan apoyar sus procesos dentro de la industria del software en Venezuela.

Bajo este concepto de ideas, la sub-clasificación de la investigación aplicada en *investigación-acción*, es la que mejor se acopla con el propósito del presente proyecto de investigación, ya que la misma tiene como propósito investigar la condición actual y condición deseada de un grupo, equipo, proyectos, programas, unidades o la organización en su conjunto, para luego realizar intervenciones que conduzcan al mejoramiento de su gestión para lograr la condición deseada (Valerino et.al, 2006).

Mediante una metodología de desarrollo, se presentará un profundo análisis sobre el tema en estudio y por ende, las conclusiones sobre la viabilidad y realización del proyecto, ejecutando la propuesta hasta conformar el cuerpo de conclusiones y resultados del trabajo realizado. La metodología de desarrollo sigue la línea de la investigación-acción.

3.2 Diseño de la Investigación

Para lograr diseñar el modelo de referencia para PYMES en la selección de Herramientas de pruebas, se utiliza el Framework Metodológico del LISI que está inspirado en el método Investigación-Acción y utiliza la metodología DESMET para la fase de evaluación. Este Framework Metodológico, ha sido en gran parte resultado de aportes relevantes de cada investigador haciendo uso del mismo. Cada uno de las fases/pasos, están bien sustentadas con actividades de envergadura que permiten satisfacerlos.

Existe un tipo de sistema, denominado “Sistema de Actividad Humana”, donde según Checkland (1993), están formados por un número de actividades conectadas como resultado de algún principio de coherencia y que generalmente son complejos, multivariantes y poseen fenómenos del mundo real que no pueden ser pasados por alto. El método investigación acción está orientado a atacar este tipo de problemas en los que el componente humano juega un papel crucial. Bajo esta premisa, enmarcar la presente investigación dentro de la metodología científica tradicional, se torna insostenible. Es por ello que este marco de trabajo (framework) es suficiente y reemplaza, en esta investigación, al método científico tradicional.

Susman y Evered (1978) proponen una descripción de Investigación Acción, en la cual se detalla un proceso cíclico compuesto por cinco fases (Diagnosticar, Planificar la acción, Tomar la acción, Evaluar y Especificar el aprendizaje). Por su parte, Baskerville (1999) identifica cinco fases que interactúan entre sí:

1. **Diagnosticar:** Corresponde a la identificación de los principales problemas que atañen a la organización y que motivan su deseo de cambiar. Envuelve todo lo referente a la interpretación del problema complejo de la organización de forma holística.

2. **Planificar la Acción:** Esta actividad especifica las acciones organizacionales que deberían tomarse para relevar o mejorar los problemas detectados al diagnosticar. El descubrimiento de los planes de acción es guiado por el marco teórico, el cual indica el estado futuro deseado por la organización y los cambios requeridos para alcanzar ese estado. El plan establece el objetivo del cambio y el enfoque para cambiar.
3. **Tomar la acción:** Implementa el plan de acción. Los investigadores y participantes colaboran en la intervención activa dentro de la organización cliente, provocando ciertos cambios. Diversas estrategias de intervención pueden ser adoptadas: directiva (los investigadores dirigen el cambio), no directiva y tácita.
4. **Evaluar:** Una vez completadas las acciones, los investigadores y demás participantes evalúan las salidas. Ésta incluye determinar si los efectos teóricos de la acción fueron alcanzados y si estos efectos relevaban a los problemas. Si los cambios fueron exitosos, la evaluación se pregunta si los cambios propuestos fueron los únicos causantes de este éxito. Si los cambios no fueron exitosos, es necesario establecer un marco para la próxima iteración del ciclo de investigación acción.
5. **Especificar el aprendizaje:** A partir del resultado de la evaluación, los investigadores especifican el conocimiento adquirido.

Ahora bien, según la naturaleza empírica de la fase de Evaluación del Método Investigación Acción, es necesario reforzarla con otras metodologías como DESMET, la cual según Kitchenham, (1996) ayuda al evaluador de una organización a planificar y a ejecutar la evaluación de manera imparcial y fiable.

El *Framework* Metodológico del LISI consta de 11 pasos o actividades agrupadas en 5 grandes fases, que se corresponden con las fases del Método Investigación Acción descrito anteriormente, en conjunción con el método de evaluación DESMET, se visualiza en la Figura 12, que ilustra la adaptación de este *Framework* Metodológico propuesto a la investigación en curso.

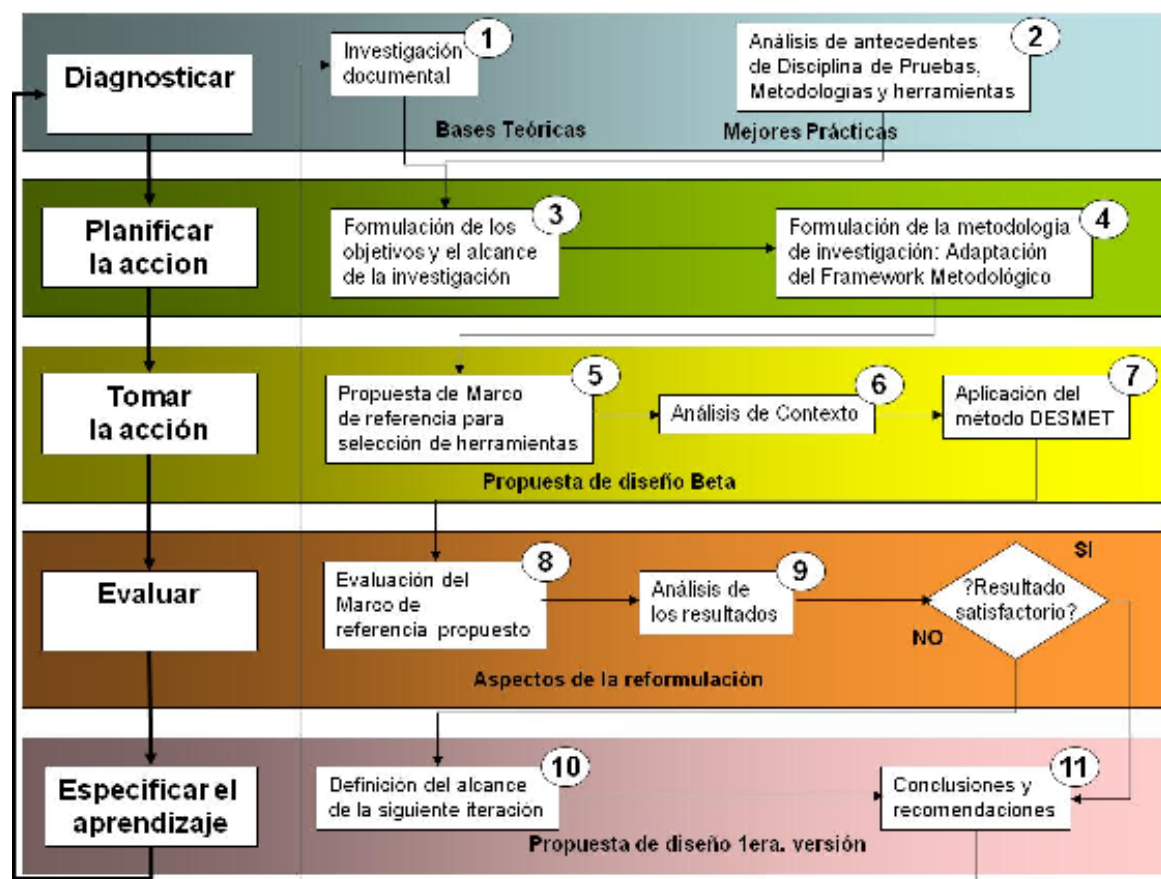


Figura 12 - *Framework Metodológico para la de investigación*. Fuente: Adaptado de Pérez et al. (2004)

Cada una de las actividades presentadas en la figura referenciada anteriormente, con su objetivo y breve descripción, se muestran a continuación en la Tabla 4, estableciendo la secuencia y principales salidas de dichas actividades, que conforman el *Framework* metodológico de la presente investigación.

Actividad	Descripción aplicada a la investigación
1. Investigación documental.	Etapa de la fase de Diagnosticar, consiste en la revisión del material bibliográfico relacionado con las herramientas de Pruebas, aseguramiento de la calidad y calidad del software. El objetivo es concretar un marco conceptual que soporte el trabajo de investigación. El resultado de esta actividad se presenta en el Capítulo II (Marco Teórico).
2. Análisis de antecedentes de Herramientas de Pruebas.	Etapa de la fase de Diagnosticar, que consiste en la descomposición de las Herramientas de Pruebas y sus relaciones con la disciplina de pruebas dentro del desarrollo de software, para indicar su importancia en el proceso. Igualmente, en esta etapa se identifican los trabajos relacionados antecedentes del mismo. El resultado de esta actividad se presenta en el Capítulo I y II.
3. Formulación de los objetivos y el alcance de la investigación.	Etapa de la fase Planificar la Acción donde se definen claramente los objetivos de la investigación, así como las directrices y el alcance. El objetivo es delimitar el área de investigación. El resultado de esta actividad está reflejado en la Justificación del Problema y la especificación de los Objetivos del Trabajo de Grado, presentados en el capítulo I de la presente investigación.

4. Formulación de la metodología de investigación: Adaptación de Investigación Acción.	Etapa de la fase Planificar la Acción donde se elabora la adaptación para esta investigación de la metodología de investigación acción. El objetivo es elaborar el marco metodológico que soporta el trabajo de investigación. El presente Capítulo III es el resultado de esta actividad.
5. Propuesta de Modelo de referencia para selección de herramientas	Etapa de la fase Tomar la Acción donde se realizan los análisis para el diseño del modelo de referencia en la selección de herramientas. El objetivo es proponer dicho modelo de referencia para la disciplina de Pruebas. Esta información es registrada en el Capítulo IV (Desarrollo de la Propuesta).
6. Análisis de Contexto.	Etapa de la fase Tomar la Acción, donde se determinan las especificaciones y acuerdos necesarios para implementar el modelo de referencia. El objetivo es preparar las herramientas, ambiente y el contexto donde será evaluada la propuesta. Esta información fue registrada en el Capítulo V (Evaluación de la Propuesta).
7. Aplicación del método DESMET.	Etapa de la fase Tomar la Acción, donde se ejecuta la metodología DESMET. El objetivo es obtener un método fiable e imparcial para la evaluación. Dicha evaluación se registrará en el Capítulo V (Evaluación de la propuesta).
8. Evaluación a través del método arrojado por DESMET.	Etapa de la fase Evaluar, que consiste en aplicar el método de evaluación arrojado por la actividad 7 al modelo propuesto. El objetivo es la evaluación del nuevo modelo mediante un método de evaluación apropiado y se registra en el Capítulo V (Evaluación de la propuesta).
9. Análisis de los resultados.	Etapa de la fase Evaluar, que consiste en estudiar los resultados a partir de los objetivos planteados en el trabajo de investigación, en términos de: el modelo propuesto y los productos tangibles alcanzados. Al terminar esta fase se evalúa si el resultado es satisfactorio, en caso afirmativo se pasa a la actividad 11, en caso negativo se pasa a la actividad 10. Esto se registró en el Capítulo VI (Análisis e interpretación de resultados).
10. Definición del alcance de la siguiente iteración.	Etapa de la fase Especificar el Aprendizaje, consiste en delimitar el alcance de las modificaciones que se deben realizar al modelo propuesto para aumentar la confiabilidad del mismo y sus posibilidades de éxito. Esto se registró en el Capítulo VI (Análisis e interpretación de resultados) solo para el nivel de métricas.
11. Conclusiones y recomendaciones.	Etapa de la fase Especificar el Aprendizaje, donde se establecen algunas conclusiones relativas al modelo propuesto aplicado y sus resultados. Finalmente, se sugieren algunas recomendaciones para futuros refinamientos del modelo propuesto y para investigaciones relacionadas. Esta información está presente en el Capítulo VII de la presente investigación (Conclusiones y recomendaciones).

Tabla 4 -*Fases del Método Investigación Acción*. Fuente: Elaboración Propia.

Apoyándose en el método de evaluación DESMET para validar el modelo de referencia propuesto, a continuación se presenta la descripción, objetivos y criterios de dicho método.

Según Kitchenham (1996), la metodología DESMET está compuesta por un conjunto de métodos con sus respectivas herramientas que pueden ser aplicadas o utilizadas por una organización en particular. Esta metodología, en el caso del presente trabajo de grado, será utilizada para la planificación y posterior evaluación

confiable del modelo propuesto que será aplicado en PYMES dedicadas al desarrollo de software.

De forma general, DESMET destaca la existencia de dos tipos de evaluación: la evaluación cualitativa y la evaluación cuantitativa. El propósito de las *evaluaciones cualitativas* es establecer efectos mensurables relativos al uso de un método o herramienta, basadas en la identificación de los beneficios esperados de los mismos y que aporten en términos mensurables y en la recolección de datos. Por su parte, las *evaluaciones cuantitativas* tienen como objetivo establecer qué tanto se ajusta un método o herramienta a las necesidades y a la cultura de una organización. Por lo general esto se determina en términos de las características del método o herramienta. Estas características se basan en los requerimientos de la población de usuarios y en los estándares de la organización. DESMET se refiere a este tipo de evaluación como Análisis de Características, el cual es identificado como una evaluación cualitativa o subjetiva (Kitchenham, 1996).

Este mismo autor establece que las técnicas de evaluación propuestas por DESMET son dependientes del contexto; puesto que no se espera que el objeto de evaluación sea el más apropiado en todas las circunstancias. En este sentido las diferencias en los resultados obtenidos están determinadas por las características de las empresas, no por los objetos a evaluar. Es por ello que al evaluar el modelo para seleccionar la mejor herramienta de pruebas en una organización específica que desarrolle sistemas de software, es muy probable que la importancia de cada característica varíe de acuerdo a las necesidades y a la realidad de la empresa.

Según Kitchenham, (1996), DESMET identifica 7 criterios que influyen en la selección del método de evaluación:

- *El contexto de la evaluación*
- *La naturaleza del impacto esperado de la utilización del método o la herramienta.*
- *La naturaleza del objeto de evaluación, es decir, si es un método, una herramienta o un método genérico.*
- *El alcance del impacto del método o la herramienta.*
- *La madurez del método o herramienta.*
- *La curva de aprendizaje asociada al método o la herramienta.*
- *Las habilidades de medición que posee la organización encargada de evaluar.*

Los criterios de selección permitirán escoger uno de los nueve métodos de evaluación propuestos por Kitchenham (1996):

1. **Experimento cuantitativo:** conforma el método científico básico para la determinación de diferencias entre métodos o herramientas.
2. **Estudio de caso cuantitativo:** consiste en la evaluación de un método o herramienta después que éste ha sido utilizado en un proyecto real.
3. **Encuesta cuantitativa:** es utilizada cuando varias herramientas o métodos han sido usados en la organización, y consiste en interrogar a los usuarios para que suministren información acerca de alguna característica de interés.
4. **Análisis de características por proyección:** por lo general se basa en la documentación de la herramienta o del método a evaluar.
5. **Análisis de características por estudio de caso:** consiste en el análisis de características del método o herramienta luego que el mismo ha sido aplicado en un proyecto real.
6. **Análisis de características por experimento:** en este caso se sugiere que un conjunto de usuarios potenciales pruebe el objeto de evaluación antes de realizar sus evaluaciones.
7. **Análisis de características por encuesta:** es realizada por personas con experiencia en el uso de la herramienta o con conocimientos teóricos de la misma.
8. **Análisis cualitativo de efectos:** se refiere a la utilización del juicio de expertos para determinar los efectos cuantitativos de diferentes métodos y herramientas.
9. **Benchmarking:** consiste en evaluar el comportamiento de la herramienta o método en relación con otros ya establecidos.

El ciclo metodológico usado soporta la investigación de problemas complejos, partiendo de prácticas exitosas y evaluando el modelo propuesto en versión Beta. Absorbe el aprendizaje y lo incorpora a una nueva propuesta mejorada. Este ciclo puede repetirse n veces; pero para efectos de esta investigación se considera una sola iteración, dado que es un Trabajo de Grado de Maestría con limitaciones de tiempo.

Por otra parte, según Pressman (2002), la única forma racional de mejorar cualquier proceso es medir atributos del proceso, desarrollar un juego de métricas significativas según estos atributos y entonces utilizar las métricas para proporcionar

indicadores que conducirán a una estrategia de mejora. A continuación se presenta un enfoque basado justamente en objetivos, preguntas y métricas que permite cuantificar o parametrizar el modelo de referencia a proponer.

Enfoque basado en objetivos, preguntas y métricas – GQM

Según Basili et al. (1994), el desarrollo de software como cualquier disciplina de ingeniería, requiere un mecanismo de medición para su corrección y evaluación. Para que este mecanismo sea efectivo debe estar centrado en objetivos específicos; debe ser aplicado a todo el ciclo de vida de los productos, procesos y recursos; y su interpretación debe estar basada en las características del contexto organizacional, ambiente y objetivos.

En este sentido, estos autores han desarrollado un enfoque basado en objetivos, preguntas y métricas (GQM – del inglés *Goal Question Metric*). Este enfoque, según Basili et al. (1994), posee tres niveles que se citan a continuación:

Primer Nivel - Conceptual (OBJETIVO): Un objetivo es definido por un objeto, un conjunto de razones con respecto a varios modelos de calidad y puede tener varios puntos de vista e incluso es relativo a un ambiente particular. Los objetos mensurables son productos, procesos y recursos.

Segundo Nivel – Operacional (PREGUNTA): Un conjunto de preguntas es utilizado para caracterizar la forma en que se alcanzará un objetivo específico. Las preguntas tratan de caracterizar el objeto de medición con respecto a un atributo de calidad seleccionado, para determinar su calidad desde un cierto punto de vista.

Tercer Nivel – Cuantitativo (METRICA): Un conjunto de datos es asociado con cada pregunta para responderla de forma cuantitativa. Los datos pueden ser objetivos (si dependen sólo del objeto medido) o subjetivos (si dependen del objeto medido y del punto de vista desde el cual son tomados).

Para Pressman (2006), la importancia de GQM proviene no solamente del hecho de que es uno de los primeros intentos de desarrollar un conjunto de medidas adecuado que pueda ser aplicado al software, sino también al hecho de que está relacionado con el paradigma de mejora de procesos. El enfoque GQM se presenta como un mecanismo detallado para la definición y elaboración de objetivos medibles.

3.3 Técnicas e instrumentos de recolección de datos

Es importante destacar que tomando en cuenta el framework metodológico con todos sus aportes, explicado en la sección anterior, algunas de las restantes fases que especificadas en el método científico tradicional, como lo son: a) Población y muestra, b) Operacionalización de variables y c) Validación de instrumentos, que formarían parte del presente capítulo III, son suplidas dentro del marco de trabajo que rige esta investigación, considerado como el más idóneo para el estudio de los Sistemas de información, por varias razones de acuerdo a lo analizado en base al autor Baskerville (1999):

- *Dentro del quehacer científico consiste no solo en la comprensión de los aspectos de la realidad existente, sino también en la identificación de las fuerzas sociales y las relaciones que están detrás de la experiencia humana.*
- *En la investigación - acción no hay mucho énfasis en el empleo del instrumental técnico de estadísticas y de muestreo. El método investigación-acción, no trabaja con población y muestra, puesto que se enfoca en un caso bien detallado, que para la investigación en desarrollo, son las PYMEs desarrolladoras de software en Venezuela.*
- *Los resultados se prueban en la realidad. Las experiencias que resultan en el campo social proporcionan las informaciones acerca de los procesos históricos, luego de su análisis.*

En cuanto a las técnicas e instrumentos para recolección de información, se usarán las encuestas en su modalidad de entrevistas con expertos e investigadores del Laboratorio de Investigación en Sistemas de Información (LISI). Igualmente, se usaran algunas técnicas que sustenten la investigación documental. Esto es, para el análisis documental, se utilizará el análisis de contenido y algunos cuadros de registro y clasificación de las categorías para comparar las herramientas de pruebas del mercado.

CAPÍTULO IV: DESARROLLO DE LA PROPUESTA

En este capítulo se describe el modelo de calidad propuesto, el cual establece las características que una herramienta de software debe cumplir según el tipo de prueba que se requiera ejecutar en pro de permitir la selección de herramientas de prueba, así como también, para orientar la construcción de un software adaptado a necesidades particulares de una organización como herramienta de prueba. El resultado de este modelo permite seleccionar herramientas de software de prueba, como propósito principal (enfoque en esta investigación). Bajo esta perspectiva, se realizan aportes al conocimiento y se presenta primeramente el modelo de dominio o conceptual en UML, con su respectiva explicación, basado en la identificación de las características que deben estar presentes en el modelo y luego de haber realizado una revisión bibliográfica de los tópicos asociados. Seguidamente, se profundiza sobre las herramientas de prueba existentes en el mercado, presentando un árbol de clasificación propuesto de herramientas de prueba automatizadas. Finalmente, se presenta el desarrollo de la propuesta con la instanciación del modelo MOSCA para pruebas a nivel de producto.

4.1 Modelo Conceptual

El modelo conceptual presentado en la Figura 17, resume todos los conceptos y relaciones que se obtuvieron a lo largo de esta investigación, así como los aportes que ha dado el investigador en este documento (tales como la adaptación de la figura 13) y el análisis de un conjunto de herramientas de prueba.

Teniendo como parte central las pruebas, se desglosan varios diagramas conceptuales específicos y ciertamente relacionados entre los que están: Casos de Prueba, Tipos de prueba, Niveles de prueba, Herramientas de prueba, apoyo al desarrollo del software y el aseguramiento de la calidad, los cuales se agrupan en diagramas distintos a fin de poder explotar sus relaciones de forma más detallada, seguido de su correspondiente explicación.

Como se observa en la Figura 13, los casos de prueba están íntimamente relacionados con los requerimientos, pues los Casos de Uso dan el aporte básico para plantear los respectivos Casos de prueba. De igual forma, los casos de prueba son

descritos a través de los procedimientos de prueba que con sus pasos, condiciones y valores o datos de la prueba, se conjugan para ser candidatos a automatizarse a través de los Scripts de prueba. Estos últimos, en forma ordenada, se pueden ejecutar planteando una secuencia lógica del conjunto de casos de prueba (Suite de prueba), sobre un ambiente de prueba predefinido constituido principalmente por el software, hardware y base de datos donde se ejecute la aplicación objeto de prueba. A su vez, la ejecución de las prueba trae consigo resultados satisfactorios o dignos de atención como lo son las fallas, que determinarán finalmente el resultado de la prueba obtenidos en contraste con los esperados. Los casos de prueba se ejecutan en un ambiente de prueba que puede estar conformado por uno o más de los siguientes elementos: software, hardware y base de datos.

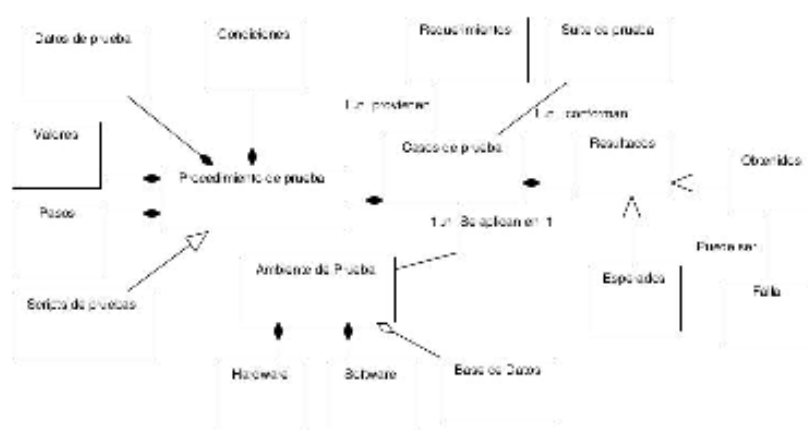


Figura 13- Modelo Conceptual asociado a los Casos de Prueba. Fuente: Adaptado de Méndez et al. (2007).

En la Figura 14 se presentan 23 de los muchos tipos de prueba que son propuestos por diversos autores, ya que el investigador consideró éste subconjunto más relevante y a la vez general entre la clasificación de prueba y/o las diversas tipologías de prueba. En cuanto a los niveles de prueba, se detalla la clasificación de acuerdo al objeto de prueba; esto es, basadas en las pruebas del desarrollador (unitarias), con otros módulos (integración), certificadas (aceptación del usuario) y del sistema con sus interfaces y demás componentes simulando que ya está en ambientes de producción. De la misma forma existen las técnicas o enfoques de pruebas (caja blanca y caja negra) que permiten determinar el enfoque en el marco de las pruebas, pues buscan verificar (internamente – estructura de implementación) y validar (externamente – cumplimiento de la funcionalidad) respectivamente.

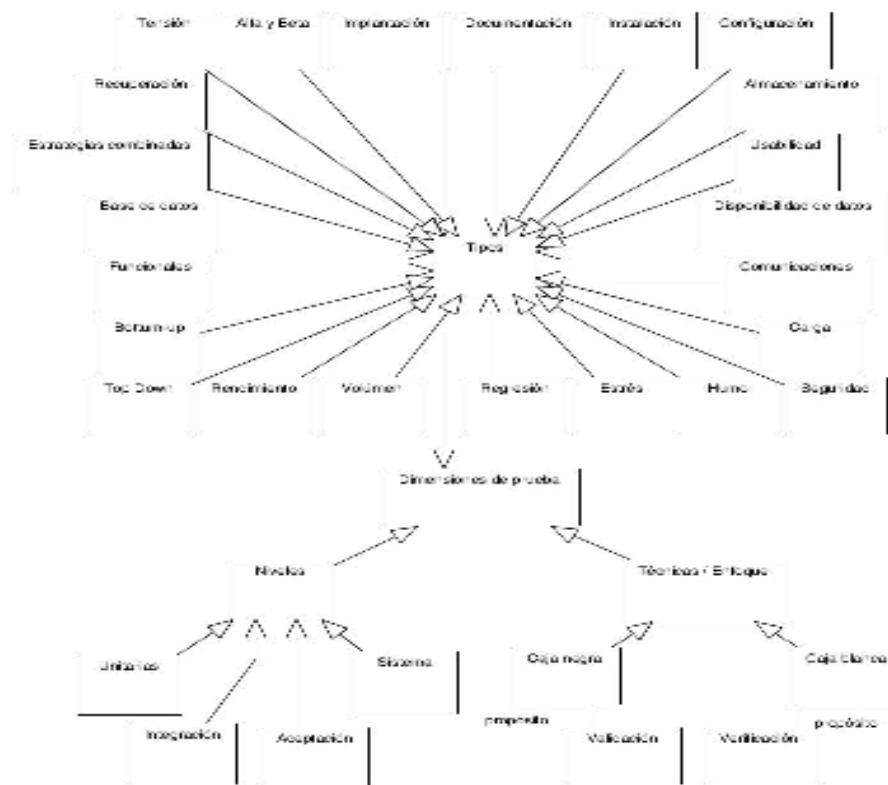


Figura 14 - Modelo Conceptual asociado a los Tipos, Niveles y Técnicas de Prueba.
Fuente: Elaboración Propia

Los casos de prueba pueden ser gestionados y/o automatizados a través de herramientas de prueba destinadas para tal fin de acuerdo a los diversos tipos de prueba que existen. Esencia de esta investigación, el modelo de la Figura 15 presenta una clasificación de las mismas que puede seguir descomponiéndose en otras subclasificaciones. Por otra parte, en la misma figura, se presenta un aporte al estudio de las herramientas de prueba en cuanto al tipo de aplicaciones que pueden ser probadas a través de las herramientas; esto es, qué tipo de tecnología, plataformas o propósitos deben tener las herramientas de prueba para adaptarse a una u otra aplicación. Cabe destacar que no todas las herramientas de prueba sirven para probar todas las aplicaciones que se deseen probar; depende de los objetos de prueba que se planteó abarcaría su construcción.

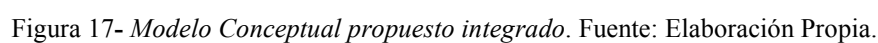
Como bien se ha mencionado en la justificación de esta investigación, la disciplina de pruebas apoya en gran manera el proceso de desarrollo de software, por lo que se presentan algunas de las metodologías que expresamente hacen mención a las pruebas dentro de sus procesos (ver Figura 15).

El diagrama de arquitectura de software se divide en varias secciones:

- Automatización y Herramientas:** En la parte superior izquierda, hay una sección para "Automatización" y "Herramientas". Debajo de ellas, se encuentran los "Herramientas de pruebas".
- Red de comunicación:** A la izquierda, se indica la "Red de comunicación".
- Aplicaciones:** En el centro, hay una sección para "Aplicaciones".
- Interfaces humanas:** A la derecha, se encuentran las "Interfaces humanas".
- Entornos de desarrollo y ejecución:** En la parte inferior, se listan varios entornos: "Aplic. Web", "Servidores", "Aplic. móviles", "Aplic. windows", "Multiplataformas", "Sistemas de BD", "Aplic. Cliente/Servidor", "Aplic. no GUI", "Aplic. GUI", "Aplic. WML", "Aplic. móviles", "Aplic. windows".
- Entorno de desarrollo y ejecución:** En la parte superior derecha, se encuentran los entornos de desarrollo y ejecución: "TDD", "RUP", "XP", "MPS", "Guardian", "Desarrollo de SW", "Análisis".

Las flechas indican las relaciones y flujos de datos entre estos componentes.

[illegible]



4.2 Clasificación propuesta para las Herramientas de prueba

Como se estableció en la sección VI de las Bases Teóricas, existen numerosas herramientas de Prueba en el mercado con características bien diferenciadoras en algunos casos y en otros, con más similitudes que diferencias y con propósitos diversos de acuerdo a lo establecido por SWEBOK (2004). Sin embargo, esta investigación está dirigida a herramientas de prueba que permitan probar el software de manera fiable más que administrar y/o gestionar pruebas, es decir, herramientas que permitan automatizar la ejecución de las pruebas.

Proponer una guía para la selección de herramientas supone el hecho de que se debe tener claro, en primera instancia, el propósito por el cual se busca automatizar las pruebas de un software en desarrollo (aplicación objeto de prueba); esto es la base de cualquier decisión de selección entre múltiples alternativas. Seguidamente, entre este mundo de posibilidades, es necesario intentar agrupar las opciones que tengan algo en común, a fin de elegir la más óptima; es decir, obtener una muestra significativa de un todo, de acuerdo a un criterio establecido. Traducido esto a esta investigación, las alternativas de herramientas de prueba que serán objetos de evaluación, deben tener alguna afinidad, deben estar ajustadas a un mismo propósito tal que puedan reflejar su ventaja competitiva frente a sus pares. En otras palabras, se puede penalizar una herramienta poderosa por el simple hecho de que no cumpla con las características esenciales que permitan plasmar el propósito definido en la primera instancia. Por ejemplo, comparar dos herramientas muy buenas funcionalmente hablando, pero descartar alguna en específico porque no permite hacer pruebas de regresión, siendo que el requerimiento inicial es ejecutar pruebas de regresión automatizadas.

Bajo ésta premisa, en la Figura 18 se propone una clasificación de herramientas de prueba automatizadas en base a la exploración por varios portales que exponen herramientas del mercado tecnológico. Esta clasificación se basa en ciertas particularidades presentes en cada herramienta de prueba analizada, en la agrupación de características de interés común entre las mismas y tomando en cuenta el modelo conceptual presentado en la sección anterior, todo esto con el fin de determinar la característica que presenta MOSCA dentro de la categoría Funcionalidad, la cual es el ajuste a los propósitos, factor determinante al momento de

elegir entre varias opciones de herramientas de prueba automatizadas para suplir una necesidad de prueba específica.

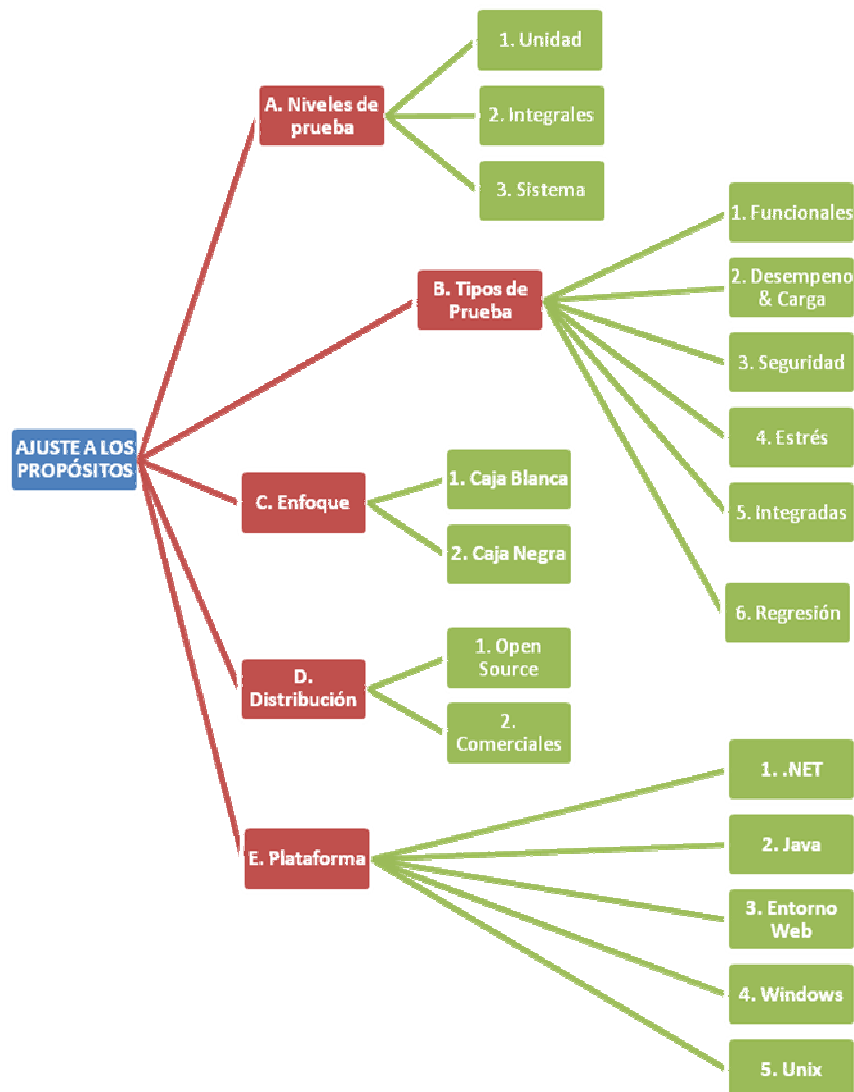


Figura 18 - *Árbol de clasificación de herramientas de prueba automatizadas.*
Fuente: Elaboración Propia.

Cabe destacar que una herramienta en específico puede estar en más de una rama del árbol por lo que algunas ramas no son exclusivas (Niveles, Tipos de prueba y Plataforma); por ejemplo, se pueden encontrar herramientas de prueba de nivel integral con distribución comercial o bien, herramientas a nivel de sistema que permita hacer pruebas funcionales y a su vez de estrés (repitiendo la prueba funcional muchas veces) con distribución Opensource. En términos cuantitativos, según el Instituto de automatización de pruebas (ATI, 2010 - www.testingfaqs.org, www.automatedtestinginstitute.com) existen en el mercado alrededor de 460

herramientas de prueba para la rama de Niveles de prueba unitarias, de las cuales un 65% son de distribución opensource y el resto comerciales, dirigidas principalmente a las pruebas de tipo funcional, con algunas excepciones que pueden ser usadas para herramientas no funcionales como estrés o carga; de esta muestra, algunas pueden probar aplicaciones en varias plataformas, mientras que otras están orientadas a una plataforma en específico por una parte, y por la otra, están enfocadas en su mayoría a caja blanca.

La realidad ejemplificada en el párrafo anterior se traduce en infinitas combinaciones posibles dentro del extenso conjunto de herramientas de prueba automatizadas, especialmente para esas ramas del árbol que no son estrictamente excluyentes. Estas combinaciones pueden producirse entre las categorías o niveles del árbol o bien, entre los hijos de una categoría en particular, a excepción de los niveles siguientes por ser excluyentes: distribución y enfoque. En consecuencia, se puede instanciar el árbol de clasificación de herramientas de muchas maneras y mezclas, combinando algunos niveles con sus hijos, entre otros.

Con el propósito de validar el modelo de referencia para seleccionar herramientas de prueba y sustentar la presente investigación, se debe de alguna manera instanciar el árbol de clasificación de herramientas basado en el ajuste a los propósitos para lograr una muestra representativa de herramientas de dicha instanciación. Por tal motivo, se eligió una de las tantas instanciaciones del árbol, representada por la Figura 19, que sigue la ruta A3 B2 C2 D*E* (ver cuadros rellenos de color), es decir, la rama de herramientas de prueba automatizadas para los tipos de prueba de sistema con los tipos de desempeño y de carga, a fin indagar sobre sus prestaciones y estudiar las características tanto comunes como distintivas que las especializan.

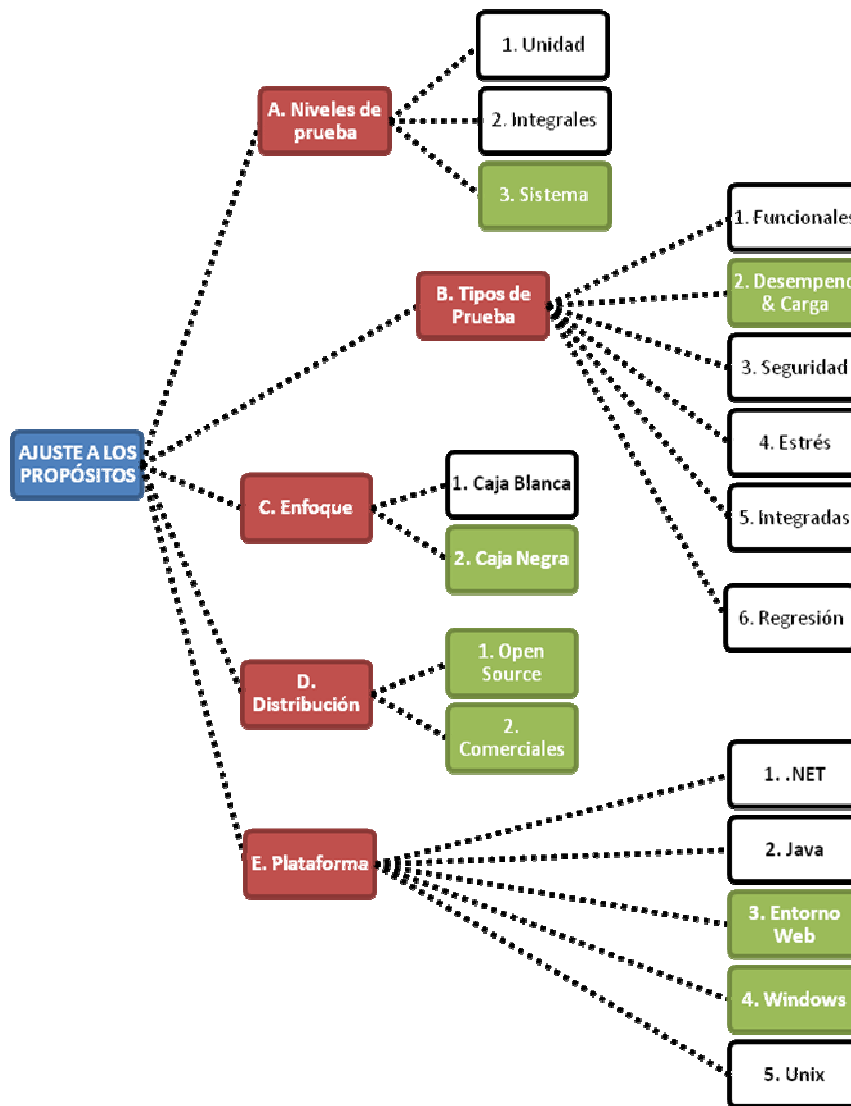


Figura 19 - Árbol instanciado de ajuste a los propósitos para herramientas de prueba elegidas por el investigador. Fuente: Elaboración Propia.

4.3 Aspectos tomados de MOSCA para la instanciación

En esta sección se presentan los diversos aspectos analizados y utilizados del Modelo Sistemático de Calidad (MOSCA) de forma general a fin de adaptarlo al objetivo de la presente investigación, selección de herramientas de pruebas, recorriendo cada uno de los niveles que lo conforman, de acuerdo al ANEXO I. Bajo ésta premisa y siguiendo la guía para la adaptación de un modelo genérico de calidad de software propuesto por Rincón *et al.* (2004) representado por MOSCA hasta llegar a un modelo de calidad específico, representado por el modelo de referencia para la selección de herramientas de prueba propuesto, se definieron los niveles del modelo de acuerdo a los siguientes planteamientos:

- a) Selección del submodelo: Se tomó el submodelo *Producto*, puesto que el objeto de esta investigación es poder seleccionar productos ya desarrollados constituido por las Herramientas de prueba.
- b) Selección de la *Dimensión* (Nivel 0): Por el punto anterior, las dimensiones asociadas al submodelo *Proceso* y *Humana* fueron descartadas al igual que la dimensión *Aspectos Internos* del *Producto*, ya que no se cuenta con el código fuente de todas las herramientas a ser evaluadas. Por lo tanto, la única dimensión tomada en la instanciación es la de los *Aspectos Contextuales*.
- c) Selección de las *Categorías* (Nivel 1): Según el algoritmo de evaluación de MOSCA (Mendoza *et al.*, 2005), la primera la categoría que debe evaluarse para el submodelo *Producto* es la *Funcionalidad*, ya que esta determina la capacidad del software para cumplir con las funciones para las que fue fabricado, dando garantía de pruebas de software realmente confiables. Seguidamente deben elegirse las categorías restantes para la evaluación entre Eficiencia, Mantenibilidad, Confiabilidad, Portabilidad y Usabilidad. Se eligió la categoría *Usabilidad* en pro de seleccionar herramientas que sean fáciles en su manejo, fáciles de aprender, de entender y hagan más viable el probar una aplicación a fin de no desviar el propósito de la prueba en entrenamiento de la herramienta de prueba. La categoría *Eficiencia* se escogió porque es fundamental conocer con cuánta celeridad se pueden obtener los resultados, proveyendo un rendimiento esperado, aún cuando aumenten los recursos utilizados (al probar un software) en condiciones específicas. La categoría *Fiabilidad* fue elegida porque es necesario garantizar que la Herramienta de prueba mantenga un nivel especificado de rendimiento bajo ciertas condiciones especificadas, es decir, que se mantenga a tono aún cuando la aplicación que está probando falle. La categoría *Mantenibilidad* que corresponde a la capacidad del producto para ser modificado, con el fin de implementar mejoras, actualizaciones en plataformas, excede el propósito de esta investigación que solo quiere determinar las prestaciones en el presente, por lo que se descartó así como la *Portabilidad*, que aunque es importante saber su capacidad de instalación, el proyecto piloto no contempla evaluar la herramientas en varios ambientes.
- d) Selección de las *Características* (Nivel 2): En cuanto al Nivel 2, correspondiente a las *características* de cada una de estas categorías, se

presentan a continuación las que se tomaron del modelo por considerarlas pertinentes a la investigación, excluyendo las que no se consideraron necesarias o aplicables dentro de una categoría ya sea porque corresponden a los Aspectos internos del Producto o por las razones que se presentan a continuación:

- ✓ **Funcionalidad (FUN)**
 - FUN 1. Ajuste a los propósitos
 - FUN 2. Precisión
 - FUN 3. Interoperabilidad
 - FUN 4. Seguridad
- ✓ **Usabilidad (USA)**
 - USA 1. Usabilidad
 - USA 2. Capacidad de Aprendizaje
 - USA 3. Interfaz Gráfica
 - USA 4. Operabilidad

La USA 5. Conformidad con los estándares, se excluyó porque se desconocen estándares mundialmente aceptados que debe poseer una herramientas de prueba; proponer las características básicas que debe poseer una herramienta de prueba (de acuerdo a su tipo) es parte fundamental de esta investigación.

- ✓ **Fiabilidad (FIA)**
 - FIA 2. Tolerancia a Fallas
 - FIA 3. Recuperación

La FIA 1. Madurez se excluyó de la investigación porque las empresas que representan las herramientas de pruebas, omiten información sobre los procesos que usan para certificar su producto antes de sacarlo al mercado (pruebas, componentes, mantenimiento). Se asume que tienen altos niveles de calidad del software.

- ✓ **Eficiencia (EFI)**
 - EFI 1. Comportamiento del tiempo
 - EFI 2. Utilización de recursos

Finalmente, las características correspondientes a las dos categorías no elegidas, se descartaron del modelo por las razones expuestas en el punto c).

- e) Selección de las *Métricas* (Nivel 3): Finalmente, las *métricas* de cada característica a ser evaluada correspondientes al Nivel 3 de MOSCA, de la misma forma como las características, se excluyeron aquellas que no eran pertinentes en esta investigación, mientras que otras se usaron tal como lo

expresa el modelo, de acuerdo a las herramientas de prueba que existen en el mercado. Las métricas tomadas de MOSCA se observan en los anexos II, III y IV correspondiente a cada una de las categorías seleccionadas, que se cuantifican de la siguiente manera: 15 métricas para Funcionalidad, 17 métricas para Usabilidad, 4 métricas para Fiabilidad y 4 para Eficiencia, para un total de 36 métricas usadas de MOSCA.

En la sección siguiente, se realiza la instanciación de MOSCA, mostrándose los aportes al modelo sistémico de calidad enfocado en las herramientas de prueba en pro de proponer un modelo para su selección.

4.4 Desarrollo de la propuesta (Aspectos agregados a MOSCA para la instanciación)

Con el propósito de instanciar MOSCA expresamente para poder hacer una selección de Herramientas de Prueba de software, se agregaron los siguientes aspectos, producto de esta investigación.

Se implementó el enfoque de GQM (Meta – Pregunta - Métrica) explicada en el Capítulo III para poder operacionalizar y elaborar los indicadores que se utilizaron para medir el modelo de referencia propuesto en las nuevas sub-características y métricas aportadas al modelo MOSCA con sus correspondientes preguntas para verificar la métrica que varían entre al menos una pregunta y máximo cuatro preguntas por métrica. Los aportes al modelo se basaron en ciertos aspectos como lo son: a) El modelo conceptual propuesto; b) El árbol de clasificación de herramientas de prueba propuesto; c) Inspección inicial de aspectos que caracterizan algunas herramientas de prueba que surgen del árbol de clasificación de herramientas instanciado elegidas del mercado tecnológico que son en número mayor a las que finalmente fueron elegidas para ser evaluadas.

Categoría FUNCIONALIDAD

Para la característica “Ajuste a los propósitos – FUN 1” se agregaron nueve (9) sub-características con sub-sub-características y métricas las cuales pueden observarse en la Tabla Nro. 5.

- (FUN 1.1) Niveles de Prueba: posee 1 sub-sub-característica y 1 métricas.
- (FUN 1.2) Tipos de Prueba: posee 2 sub-sub-característica y 2 métricas.

- (FUN 1.3) Enfoque de Prueba: posee 1 sub-sub-característica y 2 métricas.
- (FUN 1.4) Tipo de Licencias de distribución de la Herramienta de Prueba: posee 2 sub-sub-característica y 2 métricas.
- (FUN 1.5) Administración de la Herramienta de Prueba: posee 1 sub-sub-característica y 3 métricas.
- (FUN 1.6) Entregables de Prueba: posee 6 sub-sub-característica y 25 métricas.
- (FUN 1.7) Planificación de Prueba: posee 1 sub-sub-característica y 6 métricas.
- (FUN 1.8) Gestión de Resultados: posee 3 sub-sub-característica y 14 métricas.
- (FUN 1.9) Aplicación objeto de Prueba: posee 2 sub-sub-característica y 8 métricas.

En la característica “Precisión – FUN 2” se agregó una (1) sub-característica con subsub-características y métricas, las cuales pueden observarse en la Tabla Nro. 6.

- (FUN 2.1) Criterios de aceptación de Pruebas: posee 2 sub-sub-característica y 2 métricas.

Para la característica “Interoperabilidad – FUN 3” se agregó una (1) sub-característica con sub-sub-características y métricas, las cuales pueden observarse en la Tabla Nro. 7.

- (FUN 3.1) Interacción con otras aplicaciones: posee 3 sub-sub-característica y 6 métricas.

Para la característica “Seguridad – FUN 4” se agregaron dos (2) sub-características con sub-sub-características y métricas, las cuales pueden observarse en la Tabla Nro. 8.

- (FUN 4.1) Control de acceso: posee 1 sub-sub-característica y 3 métricas.
- (FUN 4.2) Capacidad de auditoría: posee 2 sub-sub-característica y 3 métricas.

Categoría USABILIDAD

Para la característica “Capacidad de aprendizaje – USA 2” se agregaron dos (2) métricas; en la característica “Interfaz Gráfica – USA 3” se agregaron dos (2) métricas y en la “Operabilidad - USA 4” se crearon dos (2) métricas, las cuales pueden observarse en la Tabla Nro. 8.

Finalmente, el modelo MOSCA instanciado, con todos los aspectos tomados del modelo original, las adaptaciones y nuevos aspectos propuestos por el investigador, excluyendo los aspectos no relevantes para el modelo de referencia de

selección de herramientas de prueba, se presenta en la Figura 20. De forma cuantitativa, se han agregado al modelo 13 nuevas sub-características, 27 nuevas sub-sub-características y 83 métricas. En el anexo V puede observarse la descripción de algunas de las características propuestas a fin de contextualizarlas en el modelo propuesto.

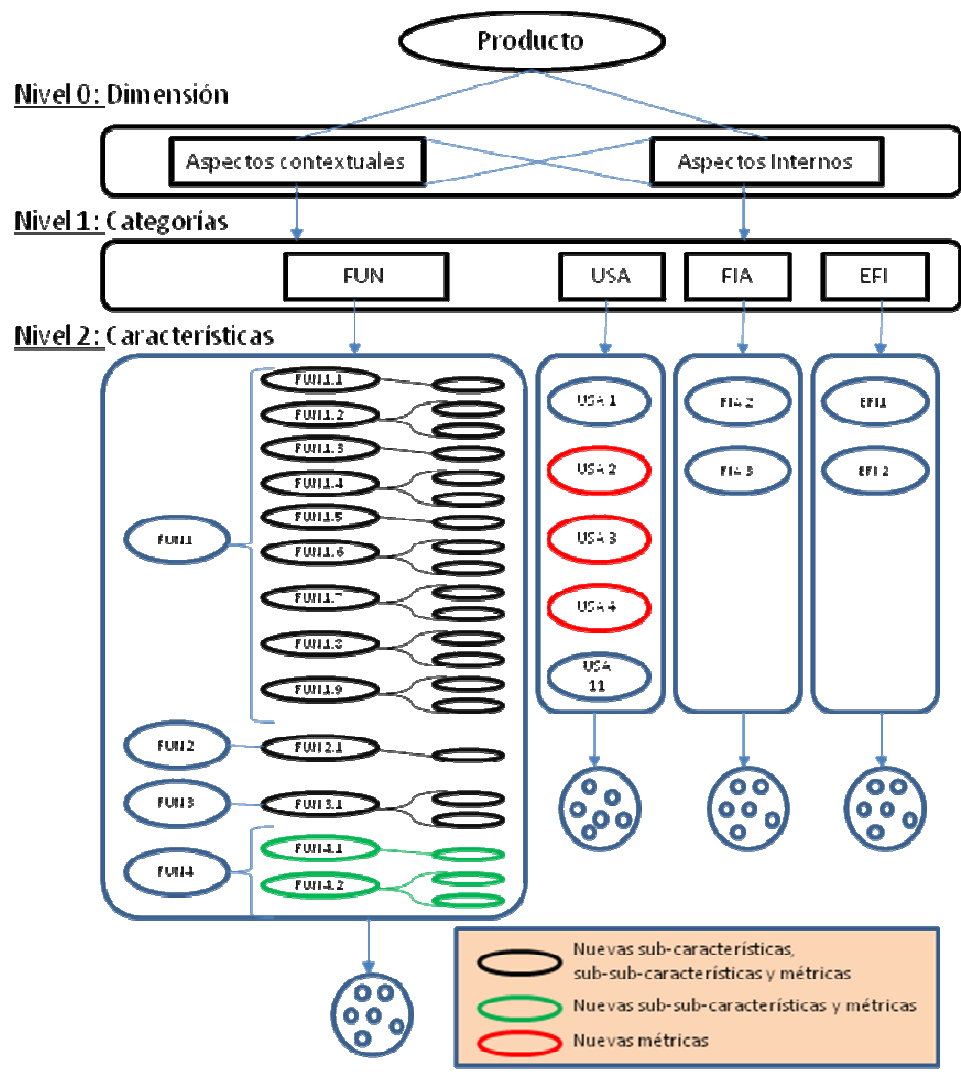


Figura 20 - Instanciación de MOSCA para el modelo de referencia para la selección de herramientas de Prueba. Fuente: Elaboración propia.

FUNCIONALIDAD (FUN) - Característica: Ajuste a los Propósitos (FUN 1)						
Sub-característica	Sub-característica	Métrica	Pregunta	Formulación	Dirigido a	
Niveles de Prueba (FUN 1.1)	Pruebas de sistema (FUN 1.1.1)	Ejecución de pruebas de a nivel de sistema	¿Permite realizar pruebas al nivel de pruebas del sistema?	5 Si, 1 No	Usuario	
Tipos de Prueba (FUN 1.2)	Pruebas de Desempeño (FUN1.2.1)	Ejecución del Tipo de Pruebas de Desempeño	¿Permite realizar pruebas del tipo Desempeño a la Aplicación objeto de prueba?	5 Si, 1 No	Usuario	
	Pruebas de	Ejecución del Tipo de	¿Permite realizar pruebas del	5 Si, 1 No	Usuario	

	Carga/Estrés (FUN1.2.2)	Pruebas de Carga/Estrés	tipo Carga/Estrés a la Aplicación objeto de prueba?		
Enfoque de Prueba (FUN 1.3)	Pruebas de Caja Negra (FUN 1.3.1)	Ejecución de Pruebas con enfoque de Caja Negra	¿Las pruebas son de enfoque de Caja Negra?	5 Si, 1 No	Usuario
		Exploración en el código de la Aplicación objeto de prueba	¿Omite el Profundizar/analizar el código de la Aplicación objeto de prueba?	5 Si, 1 No	Usuario
Tipo de Licencias de distribución de la Herramienta de Prueba (FUN 1.4)	Open source (FUN 1.4.1)	Distribución de la Herramienta de Prueba Opensource	¿La licencia de la Herramienta de Prueba es de tipo opensource (libre)?	5 Si, 1 No	Usuario
	Comercial (FUN 1.4.1)	Distribución de la Herramienta de Prueba es comercial	¿La licencia de la Herramienta de Prueba es de tipo comercial (privado)?	5 Si, 1 No	Usuario
Administración de la Herramienta de Prueba (FUN 1.5)	Lenguaje de Programación de la Herramienta de Prueba (FUN 1.5.1)	Código único/proceso unificado para multiplataformas	¿Proporciona un lenguaje de programación único para la construcción de entregables?	5=Muy difícil 4=Difícil 3=Promedio 2=Fácil 1=Muy fácil	Usuario
		Código Propietario (propone comandos, funciones propietarias)	¿Propone su propio pseudo-lenguaje de programación para construir entregables?	5=Muy difícil 4=Difícil 3=Promedio 2=Fácil 1=Muy fácil	Usuario
		Necesidad de amplios conocimientos en programación	¿Requiere conocimientos de programación a un nivel específico para poder construir los entregables?	5=Muy alta 4=Alta 3= Mediana 2= Básica 1= No tiene	Usuario
Entregables de Pruebas (FUN 1.6)	Casos de Prueba (FUN 1.6.1)	Generación automática de Casos de Prueba	¿Genera automáticamente los Casos de Prueba de acuerdo a ciertas entradas?	5 Si, 1 No	Usuario
			¿Cómo es la completitud de los Casos de Prueba generados automáticamente en cuanto a: pasos, pre-condiciones, post-condiciones, valores, resultados esperados?	5=Completa. 4=Casi todos 3= Med. 2= Pocos 1= No aplica	Usuario
		Construcción de Casos de Prueba paso a paso	¿Permite construir Casos de Prueba paso a paso?	5 Si, 1 No	Usuario
		Identificación de los requerimientos de prueba	¿Permite identificar los requerimientos de prueba que se requieren probar?	5 Si, 1 No	Usuario
		Reutilización de Casos de Prueba entre proyectos	¿Los Casos de Prueba sirven para alimentar proyectos afines?	5 Si, 1 No	Usuario
		Documentación de Pre-condiciones	¿Permite documentar las pre-condiciones en los Casos de Prueba?	5 Si, 1 No	Usuario
		Documentación de Post-condiciones	¿Permite documentar las post-condiciones en los Casos de Prueba?	5 Si, 1 No	Usuario
		Datos de Prueba (FUN 1.6.2)	¿Se establecen manualmente los Datos de Prueba en los Casos de Prueba?	5 Si, 1 No	Usuario
		Generación automática de	¿Genera automáticamente	5 Si, 1 No	Usuario

		Datos de Prueba desde datos reales	los Datos de Prueba desde los datos reales identificados?		
		Generación automática de Datos de Prueba desde sentencias SQL	¿Se establecen automáticamente los Datos de Prueba desde sentencias SQL, base de datos?	5 Si, 1 No	Usuario
		Generación automática de Datos de Prueba de forma aleatoria	¿Genera automáticamente los Datos de Prueba aleatoriamente de acuerdo a datos iniciales?	5 Si, 1 No	Usuario
		Reutilización de Datos de Prueba entre proyectos	¿Los Datos de Prueba sirven para alimentar proyectos afines?	5 Si, 1 No	Usuario
	Script de Pruebas (FUN 1.6.3)	Generación automática de Scripts de Prueba desde Casos de Prueba + Datos de Prueba	¿Genera automáticamente los Scripts de Prueba de en base a los Casos de Prueba + Datos de Prueba?	5 Si, 1 No	Usuario
		Construcción de Scripts de Prueba paso a paso	¿Permite construir Scripts de Prueba paso a paso, manualmente?	5 Si, 1 No	Usuario
		Modificar Scripts de Prueba luego de generarlos	¿Permite modificar Scripts de Prueba luego de generarlos?	5 Si, 1 No	Usuario
		Modificar Scripts de Prueba en línea/ejecución	¿Permite modificar Scripts de Prueba al momento de ejecutarlo?	5 Si, 1 No	Usuario
		Reutilización de Scripts de Prueba entre proyectos	¿Permite usar/importar Scripts de Prueba existentes entre proyectos?	5 Si, 1 No	Usuario
	Suite de Pruebas (FUN 1.6.4)	Organizar/secuenciar conjunto de pruebas	¿Permite organizar y/o secuenciar conjunto de Scripts de Prueba?	5 Si, 1 No	Usuario
		Programación de Scripts de Prueba en el tiempo	¿Permite programar pruebas en el tiempo (scheduler)?	5 Si, 1 No	Usuario
	Salidas de pruebas (FUN 1.6.5)	Generación de evidencias de Pruebas	¿Genera evidencias de resultados de pruebas automáticos?	5 Si, 1 No	Usuario
		Gestión de defectos	¿Documenta defectos en forma automática luego de presentado el error?	5 Si, 1 No	Usuario
			¿Permite agregar/crear defectos?	5 Si, 1 No	Usuario
			¿Permite modificar defectos?	5 Si, 1 No	Usuario
			¿Permite notificar defectos al responsable de la aplicación/módulo?	5 Si, 1 No	Usuario
		Log de pruebas/histórico de ejecuciones	¿Permite guardar un log de las pruebas ejecutadas?	5 Si, 1 No	Usuario
	Relación entre entregables de Pruebas (FUN 1.6.6)	Relación entre requerimientos de prueba y Casos de Prueba	¿Posee mecanismo para relacionar requerimientos de prueba a los Casos de Prueba?	5 Si, 1 No	Usuario
		Relación entre Casos de Prueba y Script de Prueba	¿Posee mecanismo para relacionar Casos de Prueba a los Scripts de Pruebas?	5 Si, 1 No	Usuario
		Relación de defectos a Casos de Prueba	¿Posee mecanismo para relacionar defectos a los	5 Si, 1 No	Usuario

Planificación de Pruebas (FUN 1.7)	Condiciones a probar en la Aplicación objeto de prueba (FUN 1.7.1)		Casos/Script de Pruebas?		
		Relaciona defectos a su causa	¿Permite determinar la posible causa del defecto?	5 Si, 1 No	Usuario
		Emulación de usuarios concurrentes	¿Emula población/acceso de usuarios concurrentes?	5 Si, 1 No	Usuario
		Emulación de transacciones concurrentes	¿Emula gran carga de transacciones concurrentes?	5 Si, 1 No	Usuario
		Variación de carga de trabajo durante la prueba	¿Permite variar la carga de trabajo en el tiempo de la ejecución de la prueba?	5 Si, 1 No	Usuario
		Definición de límites/umbrales de finalización de prueba	¿Permite definir límites/umbrales donde la prueba debe concluir?	5 Si, 1 No	Usuario
			¿Permite definir tiempos donde la prueba debe concluir?	5 Si, 1 No	Usuario
		Definición de estrategias de pruebas	¿Permite ingresar retardos en la ejecución de pruebas?	5 Si, 1 No	Usuario
			¿Permite definir hilos de ejecución en la prueba?	5 Si, 1 No	Usuario
			¿Permite definir ráfagas aleatorias en la ejecución de pruebas?	5 Si, 1 No	Usuario
			¿Permite establecer un script para la estrategia de carga de trabajo?	5 Si, 1 No	Usuario
		Cancelación de pruebas al superar umbrales	¿Permite cancelar la ejecución de las pruebas al superar los umbrales establecidos?	5 Si, 1 No	Usuario
Gestión de resultados (FUN 1.8)	Estadísticas de resultados (FUN 1.8.1)	Estadísticas en tiempo real	¿Genera estadísticas/resultados en tiempo real?	5 Si, 1 No	Usuario
		Estadísticas post-mortem	¿Genera estadísticas luego de la ejecución de las pruebas?	5 Si, 1 No	Usuario
		Notifica superación de umbrales al momento	¿Genera alertas, mensajes ante la cercanía de los umbrales permitidos?	5 Si, 1 No	Usuario
		Generación de métricas	¿Permite establecer indicadores o métricas esperadas?	5 Si, 1 No	Usuario
			¿Permite configurar/parametrizar indicadores o métricas esperadas?	5 Si, 1 No	Usuario
	Análisis de resultados (FUN 1.8.2)	Provee de reportes predeterminados para gestionar resultados	¿Tiene reportes preexistentes para emitir resultados de pruebas?	5 Si, 1 No	Usuario
		Permite modificar los reportes predeterminados	¿Permite modificar reportes de resultados preexistentes?	5 Si, 1 No	Usuario
		Permite construir reportes varios de resultados	¿Permite construir reportes de resultados?	5 Si, 1 No	Usuario
		Provee gráficas predeterminados para gestionar resultados	¿Tiene gráficas preexistentes para emitir resultados de pruebas?	5 Si, 1 No	Usuario
		Permite modificar los gráficos predeterminados	¿Permite modificar gráficos de resultados preexistentes?	5 Si, 1 No	Usuario
		Permite construir gráficos varios de resultados	¿Permite construir gráficos de resultados?	5 Si, 1 No	Usuario

		Análisis de resultados automáticos	¿Proporciona conclusiones en base a los resultados obtenidos?	5 Si, 1 No	Usuario
	Manipulación de resultados (FUN 1.8.3)	Exportar resultados en formatos intercambiables	¿Permite exportar los resultados obtenidos en formato Excel (*.csv)?	5 Si, 1 No	Usuario
			¿Permite exportar los resultados obtenidos en otros formatos como pdf, doc, html, rtf, xml?	5=Completa. 4=Casi todos 3= Med. 2= Pocos 1= No	Usuario
		Formato de generación de Reportes/entregables son similares entre si	¿Son similares los formatos cómo se generan los reportes/entregables de resultados obtenidos?	5=Completa. 4=Casi todos 3= Med. 2= Pocos 1= Ninguno	Usuario
		Disponibilidad de los resultados obtenidos (base de datos)	¿Existe acceso permanente a los resultados obtenidos?	5 Si, 1 No	Usuario
Aplicación objeto de prueba (FUN 1.9)	Interacción con la Aplicación objeto de prueba (FUN 1.9.1)	Sincronización de Scripts de Prueba por nuevas versiones en la Aplicación objeto de prueba	¿Permite sincronizar los Scripts de Prueba seleccionados ante nuevas versiones de la Aplicación objeto de prueba?	5 Si, 1 No	Usuario
		Las pruebas se generan al grabar acciones posibles de los usuarios	¿Permite grabar acciones cotidianas del usuario en la Aplicación objeto de prueba?	5 Si, 1 No	Usuario
		Las pruebas se generan especificando manualmente las acciones posible	¿Permite especificar acciones cotidianas del usuario en la Aplicación objeto de prueba?	5 Si, 1 No	Usuario
		Las pruebas se generan mediante documentos/requerimientos	¿Permite obtener por documentos/requerimientos las acciones cotidianas del usuario en la Aplicación objeto de prueba?	5 Si, 1 No	Usuario
		Ante errores/defectos permite continuar la ejecución	¿Permite continuar la ejecución de la prueba ante defectos en la Aplicación objeto de prueba?	5 Si, 1 No	Usuario
		Capacidad de guardar pruebas para continuar después	¿Permite guardar pruebas para continuar después la ejecución?	5 Si, 1 No	Usuario
	Plataformas de construcción de la Aplicación objeto de prueba (FUN 1.9.2)	Aplicación objeto de prueba basada en entorno Web	¿Se pueden probar aplicaciones construidas en entorno Web?	5 Si, 1 No	Usuario
		Aplicación objeto de prueba basada en aplicación Windows	¿Se pueden probar aplicaciones construidas como aplicación Windows?	5 Si, 1 No	Usuario

Tabla 5 - Sub-características, sub-sub-características y métricas agregadas en la característica “Ajuste a los propósitos” de la categoría Funcionalidad de MOSCA. Fuente: Elaboración Propia.

FUNCIONALIDAD (FUN) - Característica: Precisión (FUN 2)					
Sub-característica	Sub-característica	Métrica	Pregunta	Formulación	Dirigido a
Aceptación de pruebas realizadas (FUN 2.1)	Criterios de aceptación de pruebas (FUN 2.1.1)	Criterios de prueba superada	¿Permite establecer los criterios/condiciones para aceptar/superar la prueba?	5 Si, 1 No	Usuario
		Criterios de prueba no superada	¿Permite establecer los criterios/condiciones para rechazar/no superar la prueba?	5 Si, 1 No	Usuario

Tabla 6- Sub-características, sub-sub-características y métricas agregadas en la característica “Precisión” de la categoría Funcionalidad de MOSCA. Fuente: Elaboración Propia.

FUNCIONALIDAD (FUN) - Característica: Interoperabilidad (FUN 3)					
Sub-característica	Sub-característica	Métrica	Pregunta	Formulación	Dirigido a
Interacción con otras aplicaciones (FUN 3.1)	Funcionamiento con terceros (FUN 3.1.1)	Componentes/plugins que complementan la Herramienta de Prueba	¿Tiene algún(o) componente/plugin que complementa y/o aumenta sus potencialidades?	5 Si, 1 No	Usuario
		Dependencia de componentes/ plugins para funcionar	¿Necesita de algún(o) componente/plugin para poder funcionar?	5 Si, 1 No	Usuario
	Operaciones con terceros (FUN 3.1.2)	Importar scripts de otras aplicaciones/lenguajes	¿Permite importar scripts desde otras Herramienta de Prueba del mismo fabricante?	5 Si, 1 No	Usuario
			¿Permite importar scripts desde otras Herramienta de Prueba de distinto fabricante?	5 Si, 1 No	Usuario
		Importar datos de prueba desde otros sistemas	¿Permite importar datos para alimentar los Datos de Prueba desde archivos planos?	5 Si, 1 No	Usuario
			¿Permite importar datos para alimentar los Datos de Prueba desde archivos Excel?	5 Si, 1 No	Usuario
	Almacenamiento en Base de Datos (FUN 3.1.3)	Manejo de Base de Datos propia	¿Posee Base de Datos propia donde almacena información relevante?	5 Si, 1 No	Usuario
		Manejo de Base de Datos no propias	¿Almacena información relevante en Base de Datos no propietaria?	5 Si, 1 No	Usuario

Tabla 7 - Sub-característica, sub-sub-características y métricas agregadas en la característica “Interoperabilidad” de la categoría Funcionalidad de MOSCA. Fuente: Elaboración Propia.

FUNCIONALIDAD (FUN) - Característica: Seguridad (FUN 4)					
Sub-característica	Sub-sub-característica	Métrica	Pregunta	Formulación	Dirigido a
Control de acceso (FUN 4.1)	Usuarios del sistema (FUN 4.1.1)	Definición de usuarios del sistema	¿Permite configurar usuarios de la Herramienta de Prueba de acuerdo a perfiles?	5 Si, 1 No	Usuario
		Control de acceso a los usuarios	¿Mantiene registro/auditoría del acceso de los usuarios?	5 Si, 1 No	Usuario
		Definición de roles dentro del sistema	¿Permite configurar roles para distinguir entre permisos de usuarios?	5 Si, 1 No	Usuario
			¿Permite implementar roles para distinguir entre permisos de usuarios?	5 Si, 1 No	Usuario
Capacidad de auditoría (FUN 4.2)	Registro de auditoría (FUN 4.2.1)	Registro de acceso a la Herramienta de Prueba	¿Permite registrar el acceso a los usuarios registrados?	5 Si, 1 No	Usuario
		Registro de acciones en la Herramienta de Prueba	¿Permite registrar los cambios a configuraciones sobre la Herramienta de Prueba o sobre proyectos existentes?	5 Si, 1 No	Usuario
	Log de auditoría (FUN 4.2.2)	Mantenimiento de log de auditoría	¿Mantiene los logs de Auditoría sobre tiempos definidos?	5 Si, 1 No	Usuario

Tabla 8 - Sub-característica, sub-sub-características y métricas agregadas en la característica “Seguridad” de la categoría Funcionalidad de MOSCA. Fuente: Elaboración Propia.

USABILIDAD (USA)					
Característica	Sub-sub-característica	Métrica	Pregunta	Formulación	Dirigido a
Capacidad de aprendizaje (USA 2)		Wizard para ciertas funcionalidades	¿Posee ayudas guiadas en la instalación o configuración de ciertas bondades de la Herramienta de Prueba?	5 Si, 1 No	Usuario
		Documentación de la ayuda	¿Ofrece ayuda en línea para las diversas funcionalidades de la Herramienta de Prueba?	5 Si, 1 No	Usuario
			¿Provee manuales de usuario para las diversas funcionalidades de la Herramienta de Prueba?	5 Si, 1 No	Usuario
			¿Provee manual de instalación para la Herramienta de Prueba?	5 Si, 1 No	Usuario
			¿Existen foros de discusión o ayuda en internet?	5 Si, 1 No	Usuario
Interfaz Gráfica (USA 3)		Barra de herramientas customizables	¿La barra de herramientas es personalizable? (elegir opciones a mostrar)		Usuario
		Accesos rápidos a opciones	¿Posee acceso rápido a funciones a través de las teclas?		Usuario
Operabilidad (USA 4)		Lista de posibles valores a ingresar/elegir en campos funcionales	¿Despliega lista de valores tipos para alguna funcionalidad en especial?		Usuario
		Operación de la Aplicación objeto de prueba con la Herramienta de Prueba	¿Cuán fácil es operar la aplicación objeto de prueba dentro de la herramienta de prueba?	5=Muy fácil 4=Fácil 3=Promedio 2=Difícil 1=Muy difícil	Usuario

Tabla 9 - Métricas agregadas en la categoría Usabilidad de MOSCA. Fuente: Elaboración Propia.

El próximo capítulo se corresponde con la fase “Tomar la acción” del Framework metodológico en su actividad número 9, donde se analiza el contexto de la investigación para luego aplicar el método DESMET en aras de evaluar el modelo de referencia propuesto.

CAPÍTULO V: EVALUACION DE LA PROPUESTA

En el presente capítulo se exponen los objetivos, la metodología, el proceso seguido y los resultados de la evaluación de la propuesta del modelo de referencia para la selección de herramientas de prueba por una parte, y por la otra, su respectiva aplicación a un conjunto de herramientas de prueba preseleccionadas del mercado tecnológico, a fin de comprobar su efectividad.

5.1 Objetivos de la evaluación

Con el propósito de proponer un modelo plenamente garantizado, se plantearon los siguientes objetivos a fin de realizar una evaluación eficaz del modelo.

a) Objetivo General

Evaluar la propuesta del modelo de referencia para la selección de herramientas de prueba que soporten en proceso de desarrollo de software en PYMES Venezolanas.

b) Objetivos específicos

- Analizar el contexto donde será evaluado el nuevo modelo.
- Seleccionar un método fiable e imparcial para la evaluación del modelo.
- Evaluar el modelo mediante el método seleccionado.
- Elaborar un análisis de los resultados de la evaluación.

5.2 Metodología seguida

La presente evaluación está enmarcada dentro del Framework metodológico seleccionado para esta investigación, específicamente el método de evaluación DESMET que fue presentado con mayor detalle en el Capítulo III (Marco Metodológico). Igualmente sigue los pasos de Rincón *et al.* (2004).

5.3 Proceso de evaluación

a) Análisis de Contexto

La actividad número 6 del Framework metodológico corresponde al Análisis de Contexto, donde se determinan las especificaciones y acuerdos necesarios para implementar la instanciación de MOSCA, a fin de proporcionar un marco donde será evaluado el nuevo modelo.

El análisis de contexto se realiza de acuerdo a los siguientes criterios propuestos por Kitchenham (1996): el contexto de la evaluación, la naturaleza del impacto esperado, la naturaleza del objeto de evaluación, el alcance del impacto del objeto de evaluación, la madurez del objeto de evaluación, el tiempo de aprendizaje y la madurez de la organización evaluadora.

1) Contexto de la Evaluación

Los elementos que conforman el contexto son:

- Objeto a evaluar: Modelo de referencia para la selección de herramientas de prueba.
- Investigador: Corresponde al autor del modelo y encargado de realizar su evaluación, con su respectivo análisis.
- Organización para implementar y evaluar el modelo: Laboratorio de Investigación en Sistemas de Información (LISI).

2) Naturaleza del impacto esperado

La naturaleza del impacto esperado es cualitativa. El fin de esta evaluación es que el modelo objeto de evaluación se cristalice en una guía para los equipos de desarrollo de software en PYMES Venezolanas al momento de elegir herramientas de prueba para el software que éstas construyan.

3) Naturaleza del Objeto de Evaluación

El objeto de evaluación es de naturaleza sistémica, pues dentro de su contexto debe convertirse en una guía para selección de herramientas de pruebas que permitan garantizar calidad en las aplicaciones que se desarrollen.

4) Alcance del impacto del Objeto de Evaluación

El modelo objeto de evaluación impacta directamente la fase de pruebas del proceso de desarrollo del software que produce el equipo desarrollador en las PYMES desarrolladoras de software, puesto que el uso de herramientas de pruebas se justifica para garantizar la calidad del software que se está construyendo/probando, que se traduce en confianza en los procesos que dicha empresa ejecuta y/o apoya.

5) Madurez del Objeto de Evaluación

Se considera inmaduro el modelo objeto de evaluación, puesto que se encuentra en una fase de propuesta, es decir, no está ampliamente comprobada su efectividad.

6) Tiempo requerido para entender los principios delineados por el Objeto de Evaluación

En una escala de bajo, medio y alto, el objeto de evaluación se considera medio, puesto que los evaluadores deben tener al menos conocimientos previos de los tipos de prueba que pueden realizarse en una aplicación objeto de prueba a fin de exigir prestancias en una herramienta de prueba, por una parte, y por la otra, se debe conocer la perspectiva del Producto de MOSCA.

7) Tiempo requerido para llegar a ser un experto en el uso del Objeto de Evaluación

Se considera medio, puesto que partiendo del conocimiento de los conceptos de pruebas del software, sólo resta conocer MOSCA, en la perspectiva del producto.

8) Madurez de la organización

El equipo evaluador está compuesto por el LISI, quienes poseen una amplia experiencia en evaluación de modelos. De acuerdo al criterio propuesto por Kitchenham (1996), la organización tiene un nivel de madurez 3.

Con éste análisis de contexto se culmina la actividad 6 del Framework metodológico al destacar el entorno de evaluación del modelo propuesto. Igualmente, este es el insumo principal para la aplicación del Método DESMET que sigue a continuación.

b) Aplicación del método DESMET

La actividad 7 del Framework Metodológico corresponde a la Aplicación del Método DESMET. Esta es la última etapa de la Fase tomar la Acción de la metodología de investigación, y es aquí donde se obtendrá un método fiable para la evaluación del modelo.

En el capítulo III se describió el método DESMET y sus nueve métodos de evaluación con un conjunto de criterios que ayudan a seleccionar el método apropiado (Kitchenham, 1996). Es por ello que en la Tabla 9 se resume las condiciones favorables presentes y no presentes en cada uno de los métodos propuestos por DESMET, tomando en cuenta en análisis de contexto anterior.

Método de evaluación	Condiciones favorables	Presente		%
		SI	NO	
Experimento cuantitativo	Tiempo de aprendizaje relativamente corto		X	16,66%
	Deseo en lograr que el modelo para seleccionar herramientas sea independiente del contexto	X		
	Beneficios claramente cuantificables		X	
	El modelo para seleccionar herramientas relacionado con una actividad o tarea simple		X	
	Beneficios directamente mensurables desde las salidas obtenidas		X	
	Disponibilidad de personal para tomar parte en el experimento		X	
Estudio de un caso cuantitativo	Personal con experiencia en mediciones	X		20%
	Beneficios cuantificables en un solo proyecto		X	
	Proceso de desarrollo estable		X	
	Tiempos de evaluación proporcionales con el tiempo de desarrollo de los proyectos		X	
	Beneficios cuantificables antes de retirar los indicadores del modelo para seleccionar Herramienta de Prueba		X	
Encuestas cuantitativas	Beneficios cuantificables en un solo proyecto		X	0%
	Existencia de una base de datos con datos como: productividad y calidad medida por el modelo para seleccionar Herramienta de Prueba		X	
	Experiencia en proyectos haciendo uso del modelo para seleccionar Herramienta de Prueba		X	
Análisis de características por proyección	Período cortos de tiempo para realizar la evaluación	X		50%
	Número grande de características del modelo para seleccionar herramientas de pruebas		X	
Análisis de características por Estudio de Caso	Dificultad para cuantificar beneficios	X		60%
	Beneficios observables en un proyecto determinado		X	
	Población de usuarios del modelo para seleccionar herramientas de pruebas variado		X	
	Proceso de desarrollo estable	X		
	Escala de tiempo para evaluación proporcional con el tiempo de desarrollo de los proyectos	X		
Análisis de características por experimento	Dificultad para cuantificar beneficios	X		40%
	Período cortos de tiempo para realizar la evaluación	X		
	Tiempos de aprendizaje relativamente cortos		X	
	Beneficios directamente mensurables desde los resultados obtenidos de tareas específicas.		X	

	Población de usuarios del modelo para seleccionar herramientas de prueba variado		X	
Análisis de características por encuesta	Dificultad para cuantificar beneficios	X		25%
	Población de usuarios del modelo para seleccionar herramientas de prueba variado		X	
	Beneficios observables en un solo proyecto		X	
	Experiencia en proyectos haciendo uso del modelo para seleccionar herramientas de prueba		X	
Análisis de efectos cualitativos	Ausencia de procesos de desarrollo estables		X	25%
	Disponibilidad de opiniones expertas en evaluaciones de modelos similares		X	
	Requerimiento para mezclar y combinar las características del modelo para seleccionar Herramienta de Prueba		X	
	Interés en la evaluación del modelo para seleccionar herramientas de prueba	X		
Benchmarking	El modelo para seleccionar Herramienta de Prueba requiere intervención humana dedicada en su aplicación	X		50%
	La salida del modelo para seleccionar Herramienta de Prueba puede ser clasificada de acuerdo a las bondades del modelo		X	

Tabla 10 - Condiciones favorables presentes y no presentes en cada uno de los métodos propuestos por DESMET. Fuente: Elaboración Propia.

De acuerdo a lo expuesto en la tabla anterior, el método que tiene mayor porcentaje de condiciones favorables es el método de “Análisis de características por Estudio de Caso”; por lo tanto, es el que deberíamos usar ya que es el sugerido por DESMET. En la siguiente sección se describen y desarrollan los pasos que conforman dicho método de evaluación.

Método de Análisis de Características por Estudio de Caso

El método Análisis de Características por Estudio de Caso según Kitchenham et al. (1997), consiste en la evaluación de un modelo una vez que este se aplique a un proyecto de software real. Los pasos que componen el modelo están enmarcados en dos grandes procesos, con cada uno de los siguientes pasos:

1) Análisis de Características:

- Definir el alcance de la evaluación
- Definir la base de la evaluación
- Definir roles y responsabilidades
- Definir premisas y restricciones
- Definir escalas de tiempo y esfuerzo requeridos

- Aplicar el procedimiento de evaluación escogido

2) Estudio de Caso

- Seleccionar el método a evaluar
- Identificar el conjunto de características que permiten la evaluación
- Seleccionar el proyecto piloto
- Probar el modelo en el proyecto piloto
- Asignar puntuación a las características a evaluar
- Analizar los resultados obtenidos y realizar un reporte de la evaluación

Con este análisis, se culmina la actividad siete del Framework metodológico, terminando con la fase “Tomar la acción” y provee los insumos necesarios para ejecutar la subsiguiente fase “Evaluar”, por lo que en la siguiente sección se desarrolla la actividad 8 “Evaluación a través del método arrojado por DESMET”.

c) Evaluación del Modelo de referencia basado en características para la selección de herramientas de prueba

En esta sección se realiza la evaluación del modelo de referencia desde el proceso general correspondiente al “Análisis de Características” hasta el específico “Estudio de Caso”, de acuerdo a las premisas detalladas en la sección anterior.

(c.1) Proceso Análisis de Características

Los pasos para el análisis de Características dentro del primer proceso se desarrollan a continuación.

Paso 1. Definir el alcance de la evaluación

El presente trabajo de investigación contempla la evaluación de una propuesta de un modelo de referencia basado en características para la selección de herramientas de pruebas dentro de un equipo encargado de velar por las pruebas del software para apoyar el proceso de desarrollo de software en PYMES Venezolanas.

Paso 2. Definir la base de la evaluación

La presente evaluación está basada en el estudio bibliográfico y metodológico documentado en Capítulos anteriores.

Paso 3. Definir roles y responsabilidades

Los roles relacionados con la actividad de evaluación son los siguientes:

- ✓ El patrocinante: Está constituido por el Laboratorio de Investigación de Sistemas de Información de la Universidad Simón Bolívar (LISI) expertos en la propuesta y evaluación de modelos en el ámbito del desarrollo del software, que suministran su experiencia en esta investigación.
- ✓ El evaluador: es el responsable de llevar a cabo la evaluación, en este caso se trata del autor del presente trabajo. Sus responsabilidades son: preparación del plan de evaluación, identificación del objeto de la evaluación, identificación y definición de las características a evaluar en el modelo, recopilación y análisis de los resultados obtenidos y preparación del informe final y recomendaciones.
- ✓ Usuarios del modelo: son usuarios potenciales del modelo de referencia, para los efectos del presente trabajo este rol debe ser desempeñado por las PYMES Venezolanas desarrolladoras de Software, tomando en consideración las opiniones de personal experto en el área de sistemas.

Paso 4. Definir premisas y restricciones

El proceso de evaluación se encuentra sujeto a los siguientes factores:

- ✓ Disponibilidad de tiempo y dedicación por parte del evaluador, el cual realizará el proceso como parte fundamental de su trabajo de grado.
- ✓ Duración máxima para la realización de la evaluación, la cual está determinada por el tiempo estipulado para la culminación del trabajo de grado.
- ✓ Disponibilidad del ambiente requerido para realizar la evaluación, constituida por el tiempo y atención prestada por el personal de la organización patrocinante.
- ✓ Los usuarios del modelo deben ser personas con experiencia en el tema de pruebas de software.

Paso 5. Definir escalas de tiempo y esfuerzo requerido

A continuación se presentan las actividades necesarias para especificar las actividades generales que se realizarán en pro de completar el proceso de evaluación y el tiempo que se dedicará a cada actividad.

- ✓ Elaboración del plan de evaluación: 2 días
- ✓ Elaboración de los instrumentos de evaluación: 10 días
- ✓ Solicitud y aprobación del proceso de evaluación por parte de la organización 2 días.
- ✓ Evaluación y ponderación de las características: 3 días

- ✓ Análisis de resultados: 3 días
- ✓ Elaboración de conclusiones y recomendaciones: 5 días

Paso 6. Aplicar procedimiento de Evaluación escogido

Una vez cumplidos los pasos generales correspondientes al Análisis de Características se procede a llevar a cabo el proceso relativo al Estudio de Caso, actividad desarrollada a continuación.

(c.2) Proceso Estudio de Caso

En la presente sección se detallan las actividades que deben llevarse a cabo durante un estudio de caso.

Actividad 1. Seleccionar el método a evaluar

Como se presentó con anterioridad el objeto de evaluación es el modelo de referencia para la selección de herramientas de prueba, siendo el método seleccionado el análisis de características por estudio de caso.

Actividad 2. Identificar el conjunto de características que permiten la evaluación

Para poder evaluar de una forma efectiva el modelo propuesto con cada una de sus categorías, características y métricas, es necesario establecer un conjunto de aspectos a evaluar que van desde lo más general: el modelo MOSCA; a lo más específico: las métricas. Estos aspectos son los que Kitchenham y Jones (1997) llaman características. Kitchenham y Jones (1997) señalan que existen dos tipos de característica:

- **Simples:** se utilizan cuando la característica se encuentra presente o ausente en el contexto de evaluación, esto es determinado mediante una escala nominal (Si, No)
- **Compuestas:** son utilizadas cuando la existencia o conformidad de una característica puede ser medida con una escala ordinal.

De acuerdo a lo anterior, para que la evaluación sea efectiva, debe contemplarse los aspectos del modelo que van desde lo más general (constituido por el modelo en sí) hasta los más específico (las métricas), abarcando sus categorías, características, sub características y sub-sub-características que lo conforman. Para efectos de la presente evaluación se utilizaran sólo características simples para obligar al evaluador a establecer un criterio claro de aceptación.

Las características más generales evalúan el modelo en un nivel macro, es decir, el subárbol Producto como un todo y sus Categorías. Las características generales y su definición conceptual se presentan en la Tabla 11.

Característica General	Descripción	Escala
Pertinencia del modelo propuesto	Se refiere a si el modelo propuesto es pertinente o no dentro del proceso de especificación de la calidad del software.	1: significa que el modelo propuesto es pertinente 0: significa que el modelo propuesto no es pertinente.
Complejidad de las categorías involucradas	Se refiere a si las Categorías: Funcionalidad, Usabilidad, Fiabilidad y Eficiencia dan cobertura total a los aspectos de calidad de la especificación del software.	1: significa que el modelo propuesto es completo en cuanto a las categorías utilizadas. 0: significa que de acuerdo al contexto existen categorías que deberían ser consideradas en el modelo propuesto.
Adecuación al contexto	Se refiere a si la especificación del modelo propuesto es adecuada en el contexto de la evaluación.	1: significa que el modelo propuesto se adecúa al contexto de la evaluación 0: significa que el modelo propuesto no es adecuado en el contexto de la evaluación
Precisión del nivel de calidad especificado por el modelo propuesto	Se refiere a si la calidad especificada por el modelo propuesto en el proyecto piloto fue precisa	1: significa que el nivel de calidad especificado es preciso 0: significa que el nivel de calidad especificado es impreciso

Tabla 11- *Características Generales a evaluar para el modelo propuesto.*
Fuente: Sosa (2005).

Luego de especificadas las características generales, es necesario establecer un conjunto de características que permitan evaluar las métricas del modelo. Las características específicas a considerar se presentan en la Tabla 12.

Característica Específica	Descripción	Escala
Pertinencia de la métrica	Se refiere a si una métrica es adecuada para medir la existencia o no de la característica donde se encuentra	1: significa que la métrica es pertinente 0: significa que la métrica no es pertinente
Factibilidad de la métrica	Se refiere a si es factible medir la característica propuesta en la métrica dentro del contexto de evaluación	1: significa que la métrica es factible 0: significa que la métrica no es factible
Nivel de profundidad	Se refiere a si la métrica a verificar tiene el nivel de profundidad adecuado para que el resultado sea relevante	1: significa que la métrica tiene el nivel de profundidad adecuado 0: significa que se requiere una mayor profundidad en la métrica
Escala de la métrica	Se refiere a si la escala propuesta es adecuada para medir la métrica	1: significa que la escala es adecuada 0: significa que la escala no es adecuada

Tabla 12 – *Características de las Métricas a evaluar el modelo propuesto.* Fuente: Sosa (2005).

Una vez establecidas las características generales y específicas a aplicar para evaluar el modelo propuesto se requiere especificar el criterio de aceptación, constituido por:

- Para que el modelo propuesto se considere aceptable todas sus Categorías deben ser aceptables con más del 75%.
- Para que una Característica dentro de una Categoría se considere aceptable debe obtener el 75% o más del promedio de todas las respuestas para dicha característica.
- Para que una Métrica dentro de una Característica se considere aceptable debe obtener el 75% o más del promedio de todas las respuestas para dicha métrica.

Actividad 3. Seleccionar el Caso de estudio

La siguiente actividad dentro del proceso que se debe seguir para realizar el Estudio de Caso es precisamente la selección del caso de estudio.

El estudio de caso está constituido por la selección de herramientas de prueba de tipo no funcional, específicamente herramientas que permitan hacer pruebas automatizadas para el nivel de prueba de sistema con los tipos de desempeño y de carga, como se expresó en la sección 4.2 del capítulo IV (Árbol de clasificación de herramientas de pruebas automatizadas instanciado).

Cabe destacar que las pruebas de rendimiento pueden ayudar a responder preguntas como las siguientes:

- *¿Cuántos usuarios puede soportar el sistema en condiciones normales?*
- *¿Existen situaciones en las que el rendimiento del sistema disminuya repentinamente en condiciones normales?*
- *¿Cuál es el rendimiento del sistema cuando se sobrepasan las condiciones normales?*
- *En el peor de los casos, ¿el sistema se degrada de una forma paulatina o se bloquea completamente?*

Es por ello que el objetivo de este caso de estudio es usar el modelo propuesto para seleccionar herramientas de prueba que satisfagan las necesidades anteriormente descritas, con el fin de tomar decisiones de adquisición de herramientas acertadas y

que puedan apoyar el proceso de desarrollo de software ‘in house’. Bajo esta premisa, como cada herramienta hace uso del modelo, se decidió que el estudio de caso está conformado por cada herramienta a evaluar, es decir, cada caso de estudio se corresponde con cada herramienta objeto de evaluación a fin de seleccionar la mejor.

En la Tabla 13 se identifican un subconjunto de herramientas de prueba elegidas por ser las más reconocidas en el ámbito tecnológico, por la empresa que la representa y de acuerdo a los datos obtenidos por el Automated Testing Institute (ATI, 2010). Dicha tabla especifica la versión actual en el mercado, fabricante, tiempo de instalación y dónde encontrar información al respecto. Cabe destacar, que de un subconjunto de herramientas de prueba pre-seleccionadas, la evaluación del modelo finalmente se realizó en base a cuatro herramientas que disponían de versión de prueba limitada (en número de días) descargables desde su Website, lo que permitió usar como método de aprendizaje, la manipulación de la versión de prueba disponible aplicada a un sistema en entorno web de acuerdo con cada una de las funcionalidades que éstas herramientas ofrecían.

Nro	Nombre Herramienta	Versión	Fabricante	Tiempo de Instalación	Disponible en
1	HP LoadRunner software	9.5	Hewlett-Packard	50 min.	https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-11-126-17%5e8_4000_100
2	SilkPerformer	No especifica	Borland	40 min.	http://www.borland.com/us/products/silk/silkperformer/index.html
3	IBM/Rational Performance Tester	8.1	IBM	45 min.	http://www-01.ibm.com/software/awdtools/tester/performance/index.html?S_TACT=105AGX23&S_CMP=DWNL
4	SoapUI Pro	3.5	Eviware	15 min.	http://www.soapui.org/features.html

Tabla 13 - Herramientas de prueba del mercado seleccionadas para cada estudio de caso. Fuente: Elaboración Propia.

Una breve descripción de cada una de las herramientas de pruebas, se presentan a continuación:

- Estudio de Caso 1, **HP LoadRunner software**: Permite realizar pruebas de Rendimiento determinando los cuellos de botella en las aplicaciones y si está acorde a los requerimientos de performance exigidos.
- Estudio de Caso 2, **SilkPerformer**: Permite realizar pruebas automatizadas de carga y rendimiento.

- Estudio de Caso 3, **IBM/Rational Performance Tester**: Identifica la presencia y causa de los cuellos de botella en el rendimiento del sistema.
- Estudio de Caso 4, **SoapUI Pro**: Permite realizar pruebas funcionales, de carga y conformidad de Web Services. Es una herramienta Open source.

Actividad 4. Probar el modelo propuesto en cada Estudio de caso

El tesista realizó la prueba del modelo propuesto basado en los cuatro estudios de caso correspondiente a cada una de las herramientas pre-seleccionadas antes descritas, aplicando el cuestionario resultante del modelo propuesto condensado, con los aspectos tomados y propuestos de MOSCA (Anexo VI), cuyos resultados de la aplicación de la prueba con su correspondiente análisis se presentan en el siguiente capítulo.

Actividad 5. Asignar puntuación a las características a evaluar

Para la recolección de esta información se registraron cada uno de los puntajes correspondiente a la dupla métrica-pregunta para cada uno de los estudios de caso, cuyos valores pueden observarse en el Anexo VI. Luego de finalizada esta actividad, se procedió a realizar el análisis respectivo presentado en la sección siguiente.

Actividad 6. Analizar resultados obtenidos y preparar reporte de evaluación

La sexta y última actividad del proceso Estudio de Caso corresponde al análisis de resultados, los cuales se visualizan en el siguiente capítulo VI, para así dar fin a la instanciación y evaluación del modelo propuesto.

CAPÍTULO VI: ANÁLISIS E INTERPRETACIÓN DE RESULTADOS

En este capítulo se presentan los resultados del proceso de evaluación presentado en el capítulo anterior, con su respectivo análisis, el cual va desde la aplicación del modelo propuesto en los estudio de caso escogidos hasta la evaluación del modelo propiamente dicho a fin de determinar su aplicabilidad, luego de la experiencia obtenida. Es por ello que la presentación de los resultados se han divididos en 3 secciones que son las siguientes:

- Sección 6.1: Evaluación del modelo en los estudios de caso (herramientas seleccionadas)
- Sección 6.2: Evaluación de las características generales del modelo
- Sección 6.3: Evaluación de las métricas del modelo

6.1 Evaluación del modelo en los estudios de caso

En esta sección se expone el resultado de la evaluación en cada uno de los estudios de caso seleccionados, compuesto por cuatro (4) herramientas del mercado tecnológico enmarcadas dentro del árbol de clasificación propuesto, los cuales fueron evaluadas usando el modelo propuesto.

El Anexo VI se muestra el detalle de la evaluación del modelo propuesto realizada por el autor luego de obtener un corto y aceptable aprendizaje manipulando cada una de las herramientas por una parte, y por la otra, respondiendo a cada una de las preguntas de las métricas establecidas en el modelo propuesto, de acuerdo a la columna “Formulación”. En la tabla 14 se muestra un resumen de cada uno de los niveles que componen el modelo aplicados para cada una de las cuatro herramientas evaluadas, desde el nivel más específico hasta el más general.

A continuación, se hará un análisis detallado del nivel “Características”, por ser el que está presente en todas las categorías del modelo, esto es el nivel de Sub-características y sub-sub-característica, solo está presente en la Categoría Funcionalidad, no así en el resto de las categorías del modelo: Usabilidad, Eficiencia y Fiabilidad. En las figuras 21, 22, 23 y 24 se presentan los resultados de los estudios de caso por categoría para el nivel del modelo Características.

Categoría	Característica	Sub-característica	Sub Sub-característica	% Sub-característica				% Sub-característica				% Característica				% Categoría			
				HP (1)	Silk (2)	IBM (3)	Soap(4)	HP (1)	Silk (2)	IBM (3)	Soap(4)	HP (1)	Silk (2)	IBM (3)	Soap(4)	HP (1)	Silk (2)	IBM (3)	Soap(4)
FUNCIONALIDAD (FUN)	Ajuste a los propósitos (FUN 1)	Herramienta de Prueba (FUN 1.1)	Herramienta de Prueba (Estudio de Caso) -->	Pruebas de sistema (FUN 1.1.1)				Pruebas de sistema (FUN 1.1.1)				Pruebas de sistema (FUN 1.1.1)				Pruebas de sistema (FUN 1.1.1)			
				Pruebas de Desempeño (FUN 1.2.1)				Pruebas de Desempeño (FUN 1.2.1)				Pruebas de Desempeño (FUN 1.2.1)				Pruebas de Desempeño (FUN 1.2.1)			
				Pruebas de Carga/Estres (FUN 1.2.2)				Pruebas de Carga/Estres (FUN 1.2.2)				Pruebas de Carga/Estres (FUN 1.2.2)				Pruebas de Carga/Estres (FUN 1.2.2)			
				Pruebas de Caja Negra (FUN 1.3.1)				Pruebas de Caja Negra (FUN 1.3.1)				Pruebas de Caja Negra (FUN 1.3.1)				Pruebas de Caja Negra (FUN 1.3.1)			
				Open source (FUN 1.4.1)				Open source (FUN 1.4.1)				Open source (FUN 1.4.1)				Open source (FUN 1.4.1)			
				Tipo de Licencias de distribución de la HP (FUN 1.4)				Tipo de Licencias de distribución de la HP (FUN 1.4)				Tipo de Licencias de distribución de la HP (FUN 1.4)				Tipo de Licencias de distribución de la HP (FUN 1.4)			
				Administración HP (FUN 1.5)				Administración HP (FUN 1.5)				Administración HP (FUN 1.5)				Administración HP (FUN 1.5)			
				Lenguaje de Programación de la HP (FUN 1.5.1)				Lenguaje de Programación de la HP (FUN 1.5.1)				Lenguaje de Programación de la HP (FUN 1.5.1)				Lenguaje de Programación de la HP (FUN 1.5.1)			
				Casos de Prueba-CP (FUN 1.6.1)				Casos de Prueba-CP (FUN 1.6.1)				Casos de Prueba-CP (FUN 1.6.1)				Casos de Prueba-CP (FUN 1.6.1)			
				Datos de Prueba-DP (FUN 1.6.2)				Datos de Prueba-DP (FUN 1.6.2)				Datos de Prueba-DP (FUN 1.6.2)				Datos de Prueba-DP (FUN 1.6.2)			
				Script de Pruebas-SP (FUN 1.6.3)				Script de Pruebas-SP (FUN 1.6.3)				Script de Pruebas-SP (FUN 1.6.3)				Script de Pruebas-SP (FUN 1.6.3)			
				Suite de Pruebas (FUN 1.6.4)				Suite de Pruebas (FUN 1.6.4)				Suite de Pruebas (FUN 1.6.4)				Suite de Pruebas (FUN 1.6.4)			
				Salidas de pruebas (FUN 1.6.5)				Salidas de pruebas (FUN 1.6.5)				Salidas de pruebas (FUN 1.6.5)				Salidas de pruebas (FUN 1.6.5)			
				Relación entre entregables de Pruebas (FUN 1.6.6)				Relación entre entregables de Pruebas (FUN 1.6.6)				Relación entre entregables de Pruebas (FUN 1.6.6)				Relación entre entregables de Pruebas (FUN 1.6.6)			
USABILIDAD (USA)	Precisión (FUN 2)	Planifica. de Pruebas (FUN 1.7)	Condiciones a probar en la AOP (FUN 1.7.1)	Condiciones a probar en la AOP (FUN 1.7.1)				Condiciones a probar en la AOP (FUN 1.7.1)				Condiciones a probar en la AOP (FUN 1.7.1)				Condiciones a probar en la AOP (FUN 1.7.1)			
				Estrategias de resultados (FUN 1.8.1)				Estrategias de resultados (FUN 1.8.1)				Estrategias de resultados (FUN 1.8.1)				Estrategias de resultados (FUN 1.8.1)			
				Análisis de resultados (FUN 1.8.2)				Análisis de resultados (FUN 1.8.2)				Análisis de resultados (FUN 1.8.2)				Análisis de resultados (FUN 1.8.2)			
				Manipulación de resultados (FUN 1.8.3)				Manipulación de resultados (FUN 1.8.3)				Manipulación de resultados (FUN 1.8.3)				Manipulación de resultados (FUN 1.8.3)			
				Interacción con la AOP (FUN 1.9.1)				Interacción con la AOP (FUN 1.9.1)				Interacción con la AOP (FUN 1.9.1)				Interacción con la AOP (FUN 1.9.1)			
				Plataformas de construcción de la AOP (FUN 1.9.2)				Plataformas de construcción de la AOP (FUN 1.9.2)				Plataformas de construcción de la AOP (FUN 1.9.2)				Plataformas de construcción de la AOP (FUN 1.9.2)			
				Criterios de aceptación de pruebas (FUN 2.1.1)				Criterios de aceptación de pruebas (FUN 2.1.1)				Criterios de aceptación de pruebas (FUN 2.1.1)				Criterios de aceptación de pruebas (FUN 2.1.1)			
				Funcionamiento con terceros (FUN 3.1.1)				Funcionamiento con terceros (FUN 3.1.1)				Funcionamiento con terceros (FUN 3.1.1)				Funcionamiento con terceros (FUN 3.1.1)			
				Operaciones con terceros (FUN 3.1.2)				Operaciones con terceros (FUN 3.1.2)				Operaciones con terceros (FUN 3.1.2)				Operaciones con terceros (FUN 3.1.2)			
				Almacenamiento en BD (FUN 3.1.3)				Almacenamiento en BD (FUN 3.1.3)				Almacenamiento en BD (FUN 3.1.3)				Almacenamiento en BD (FUN 3.1.3)			
				Usuarios del sistema (FUN 4.1.1)				Usuarios del sistema (FUN 4.1.1)				Usuarios del sistema (FUN 4.1.1)				Usuarios del sistema (FUN 4.1.1)			
				Registro de auditoría (FUN 4.2.1)				Registro de auditoría (FUN 4.2.1)				Registro de auditoría (FUN 4.2.1)				Registro de auditoría (FUN 4.2.1)			
				Log de auditoría (FUN 4.2.2)				Log de auditoría (FUN 4.2.2)				Log de auditoría (FUN 4.2.2)				Log de auditoría (FUN 4.2.2)			
FIABILIDAD (FIA)	Comprensión (USA 1)	Capacidad de Aprendizaje(USA2)	Interfaz Gráfica (USA 3)	Operabilidad (USA 4)				Operabilidad (USA 4)				Operabilidad (USA 4)				Operabilidad (USA 4)			
				Tolerancia a Fallos (FIA 2)				Tolerancia a Fallos (FIA 2)				Tolerancia a Fallos (FIA 2)				Tolerancia a Fallos (FIA 2)			
				Recuperación (FIA3)				Recuperación (FIA3)				Recuperación (FIA3)				Recuperación (FIA3)			
				Comportamiento del tiempo (EFI 1)				Comportamiento del tiempo (EFI 1)				Comportamiento del tiempo (EFI 1)				Comportamiento del tiempo (EFI 1)			
EFICIENCIA (EFI)	Utilización recursos (EFI 2)																		

Tabla 14 - Evaluación del modelo propuesto en el Proyecto Piloto. Fuente: Elaboración Propia.

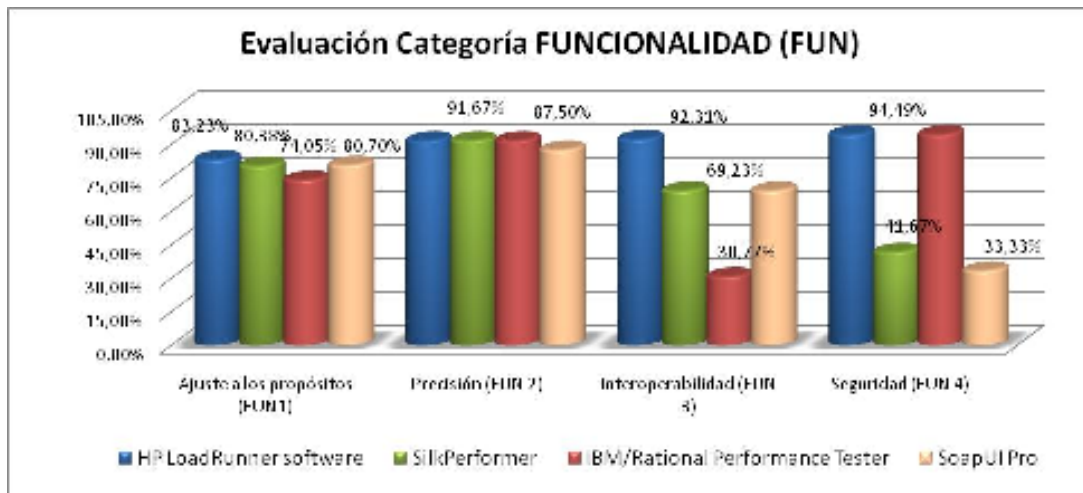


Figura 22 - Resultados evaluación Características de la Categoría “Funcionalidad” en Estudio de casos. Fuente: Elaboración propia.

En la Figura 21 se observan cada una de las características de la categoría Funcionalidad, detallándose lo siguiente:

- En FUN1- “Ajuste a los propósitos”, se muestra que tres de los cuatro estudios de casos (casos 1, 3 y 4) se consideran Aceptables, porque su porcentaje de aceptación superan el 75% mínimo establecido. El caso 2 (SilkPerformer) no se considera aceptable debido a que de acuerdo a la tabla 14, específicamente en la presentación de resultados, como se muestran en las sub-sub-características Salidas de pruebas (FUN 1.6.2) y Análisis de resultados (FUN 1.8.2) presenta ciertas debilidades, pero no está muy alejado de ser aceptado (74,05%), por lo que si se ajusta a los propósitos para lo que fue creado.
- En cuanto a “Precisión”-FUN 2 todos los estudios de caso se consideran Aceptables (mayor al 75%) con un 91,67% para los casos 1, 2 y 3 y para el caso 4 un 87,50%, por presentar debilidades en los resultados que genera.
- “Interoperabilidad”- FUN 3, fue considerada Aceptable solo en el estudio de caso 1 (HP LoadRunner) con un 92,31%. El resto de los casos, no se acercan al porcentaje mínimo requerido (75%) porque en la Interacción con otras aplicaciones (FUN 3.1), muestran debilidades aunado a que el estudio de caso 3 es muy bajo porque tiende a ser exclusiva en cuanto a sus operaciones.
- En FUN 4- “Seguridad” se observa que solo los casos 1 y 3 son Aceptables con un 94,44%, mientras que los otros dos casos (2 y 4) están muy alejados de ser aceptados, puesto que no tienen manejo de usuarios del sistema y por ende, no pueden auditar las acciones de los mismos y/o resguardar los proyectos

configurados según lo que muestra la Tabla 14 en las sub-características Control de acceso FUN 4.1 y Capacidad de auditoría FUN 4.2.

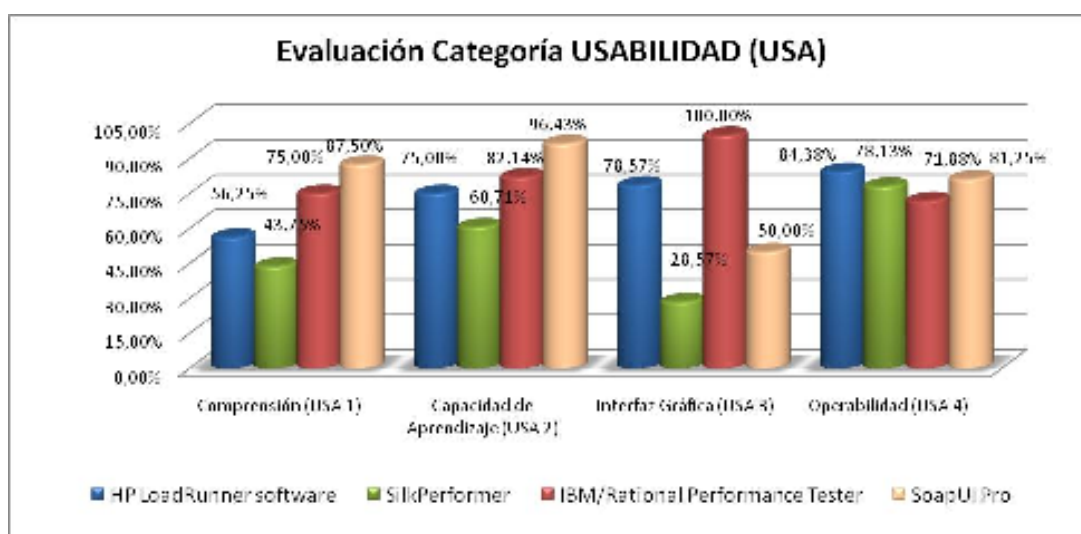


Figura 23 - Resultados evaluación Características de la Categoría “Usabilidad” en Estudio de casos.
Fuente: Elaboración propia.

La figura 22, muestra los resultados obtenidos en la categoría Usabilidad, donde cada Característica evaluada responde a lo siguiente:

- USA 1-“Facilidad Comprensión” muestra que los Estudios de casos 1 y 2 están por debajo del porcentaje de Aceptación, ya que las herramientas presentan dificultades para comprenderla por parte del usuario final.
- En “Capacidad de aprendizaje”- USA2, la única herramienta que no se consideró Aceptable fue SilkPerformer (Estudio de Caso 2), pues presenta debilidades en la documentación y en wizard para ejecutar ciertas funciones de la herramienta. Los demás Estudios de Caso (1,3 y 4) si se consideraron Aceptables.
- Para la característica USA 3- “Interfaz Gráfica” nuevamente el estudio de Caso 2 presenta debilidades en la localización y acceso rápido a ciertas opciones, además que es limitado la capacidad de tener una interfaz personalizable y el diseño de pantallas no satisface. Por su parte, el estudio de Caso 4 presenta las mismas debilidades del Caso 2 excluyendo la parte del diseño de pantallas. Por lo tanto los estudios 2 y 4 no se consideraron Aceptables en la característica evaluada, USA 3.
- En cuanto a la característica “Operabilidad” – USA 4, el Estudio de Caso 3 se encontró débil al momento de operarlo pues no presenta funcionalidades que hagan más fácil su uso. Seguidamente está el Estudio de Caso 2, pero al igual que los Casos 1 y 4, se consideraron Aceptables para la evaluación en el modelo.

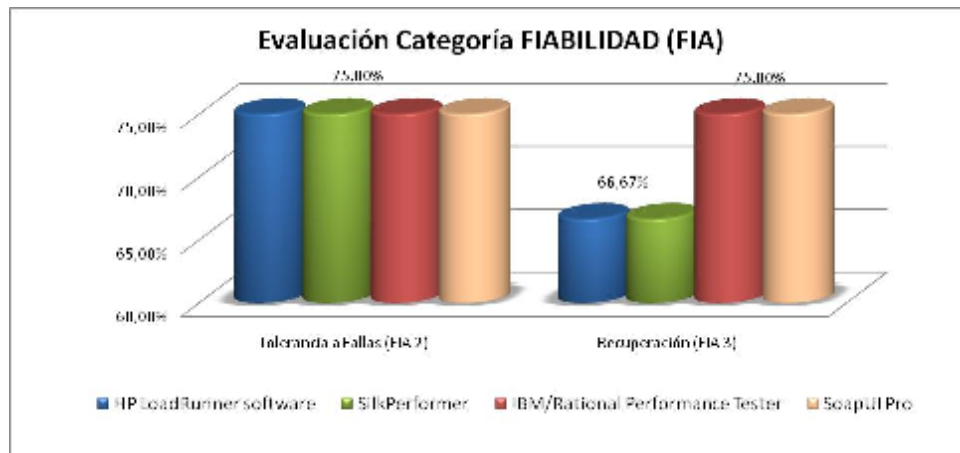


Figura 24 - Resultados evaluación Características de la Categoría “Fiabilidad” en Estudio de casos.
Fuente: Elaboración propia.

La Figura 23 muestra los resultados de la categoría Fiabilidad, en la que los resultados se muestran parejos entre sí. Los detalles son mostrados a continuación:

- Para la Característica FIA 2 – “Tolerancia a fallas”, todos los Estudios de caso se consideraron Aceptables aunque con el mínimo de porcentaje permitido; esto se debe a que presentan debilidades en evitar fallas en su operación aún cuando ésta se realice de forma incorrecta.
- En la característica “Recuperación”- FIA 3, los estudios de casos 1 y 2 no alcanzaron el porcentaje mínimo de Aceptación debido a que no tienen una buena capacidad de reiniciarse ante alguna caída de la herramienta, situación que se repite en los casos 3 y 4 pero en menor proporción, lo que les permitió considerarla “Aceptable” con el porcentaje mínimo.

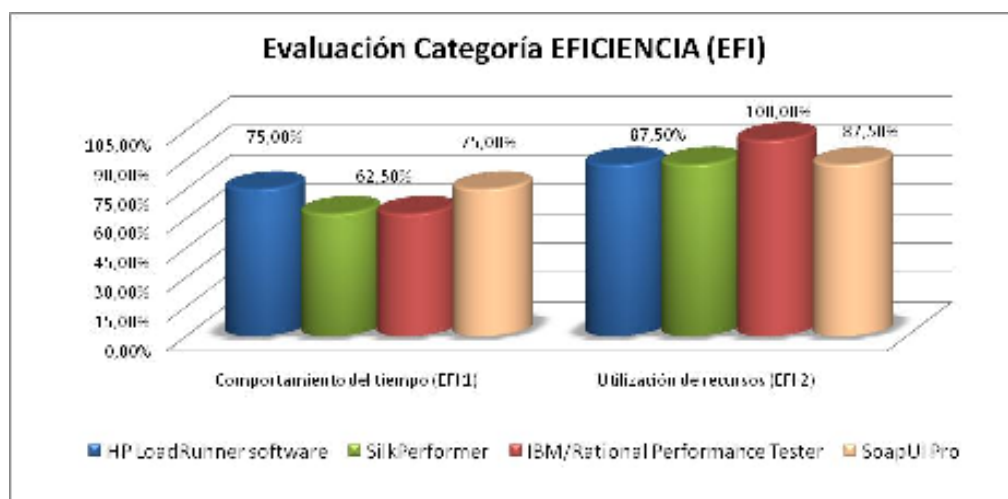


Figura 25 - Resultados evaluación Características de la Categoría “Eficiencia” en Estudio de Casos.
Fuente: Elaboración propia.

Como se observa en la Figura 24, los resultados de la evaluación de las características que conforman la categoría Fiabilidad, se detallan así:

- En “Comportamiento del tiempo” – EFI 1, los Estudios de Caso 2 y 3, no lograron el porcentaje mínimo de Aceptación por considerarse lentos al momento de dar los resultados luego de ser sometido a ciertas funciones, con una tasa de respuesta alta. Ahora bien, los Estudios de caso 1 y 4 se consideraron Aceptables por llegar el porcentaje mínimo requerido debido a que presentan las mismas fallas de los otros Estudios de Caso, pero en menor proporción, es decir, son más rápidos.
- La característica EFI 2 - “Utilización de recursos” fue superada por todos los Estudios de Caso, por lo que se consideraron Aceptables, pues muestran se equitativos al momento de administrar los recursos de la máquina donde estos operan.

Luego evaluar cada una de las características del modelo propuesto que componen las cuatro categorías elegidas en todos los Estudios de Caso correspondientes, se presenta a continuación la evaluación general de todos los Estudios de Caso en el nivel “Categorías” del modelo propuesto.

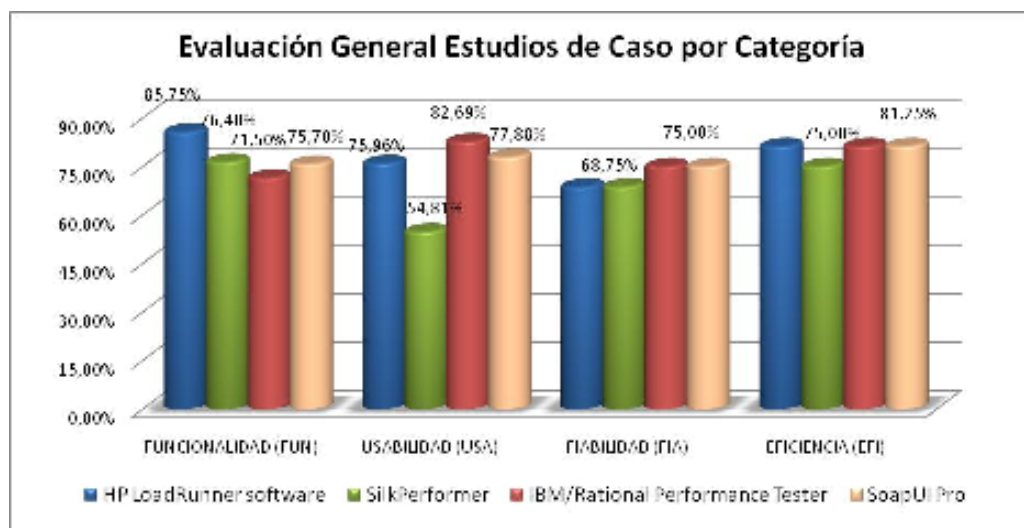


Figura 26 – Resultados de la evaluación de los estudios de caso en cada categoría del modelo propuesto. Fuente: Elaboración Propia.

En la figura 25, se observa cómo resultó ser el nivel de aceptación de cada uno de los estudios de caso para cada una de las categorías del modelo propuesto, resultados que están directamente relacionados con el nivel Característica detallado en

la sección anterior y en los demás niveles del modelo que aplica (Categoría Funcionalidad) y a continuación su respectivo análisis tomando en cuenta la Tabla 14.

- *Funcionalidad:* La única herramienta que obtuvo un porcentaje menor al 75% esperado es IBM/Rational Performance Tester que corresponde al estudio de caso 3, razón por la cual no se considera Aceptable funcionalmente hablando, pero se acerca mucho al porcentaje esperado con un 71,50%. La característica que hace bajar su porcentaje es la Interoperabilidad FUN 3 (ver Tabla 14). Los demás estudios de caso superaron el 75% de aceptación, siendo el de mayor porcentaje el estudio de caso 1 (HP Load Runner) con 85,75%, por lo que cumple en gran medida el con los requisitos prácticos de una herramienta para pruebas no funcionales. Los casos de estudio 2 y 3 tienen porcentaje menor porque la característica Seguridad FUN 4 no está suficientemente presente en las herramientas.
- *Usabilidad:* Tres de los cuatro estudios de casos resultaron aceptables en esta categoría siendo el primero el estudio de caso 3 con 82,69% seguido del caso 4 con 77,88% y muy cerca del mínimo permitido está el estudio de caso 1 con 75,96%, por no exponer una gran facilidad de comprensión (USA 1). El estudio de caso 2 Silk Performer, resultó no aceptable porque revela gran debilidad frente a sus competidores en la Interfaz gráfica USA 3 y en la facilidad de comprensión USA 1, condiciones que puede traducirse en elevación de costos y tiempo invertidos en entrenamiento de quienes harán uso de la herramienta.
- *Fiabilidad:* Esta categoría evidencia gran debilidad entre los estudios de caso, pues aunque los casos 3 y 4 resultaron Aceptables, su porcentaje estuvo en el límite, 75% respectivamente. Los casos 1 y 2 presentaron un porcentaje de 68,75% de aceptación debido a que su capacidad de recuperación FIA 3 ante cualquier evento es baja, lo que pudiera traducirse en un atraso en las pruebas que se tengan planificadas.
- *Eficiencia:* Es la única categoría de la evaluación en las que todos los estudios de caso resultaron aceptables. Sin embargo, cabe destacar que el estudio de caso 2 obtuvo el porcentaje mínimo de aceptación (75%) debido a su baja calificación en el comportamiento del tiempo (EFI 1).

De acuerdo al análisis anterior y tomando en cuenta el algoritmo de aplicación de MOSCA, los estudios de caso en los que se debe continuar la evaluación, son aquellas que hayan superado la categoría Funcionalidad, por lo que la única herramienta que se obvia en este estudio es IBM/Rational Performance Tester, pero se continúa con los casos de estudio 1, 2 y 4. Ahora bien, el Algoritmo de MOSCA plantea que para medir el nivel de calidad de un software se debe tomar en cuenta el nivel de satisfacción (porcentaje de aceptación) de las categorías incluidas. Sin embargo, este nivel de calidad es determinado por 3 categorías y no se corresponde con las cuatro categorías que comprenden el modelo propuesto. Es por ello que se requiere adaptarlo a la presente investigación.

Propuesta de adaptación del algoritmo de MOSCA

La categoría *Funcionalidad* se mantiene como la determinante en continuar o no con la evaluación de un producto. Por otra parte, dado las condiciones de aplicación del modelo propuesto basado específicamente en herramientas de pruebas de desempeño y carga, la categoría *Eficiencia* pasa a tomar gran relevancia en el modelo, puesto que las prestaciones de una herramienta de este tipo debe ser tal que no entorpezca el trabajo la ejecución de las pruebas de la aplicación objeto de pruebas, es decir, que no afecte el desempeño ni los resultados de la misma. Es por ello, que se propone que cuando hayan cuatro categorías a evaluar en la aplicación de MOSCA y dos de ellas sean determinantes para continuar o no con la evaluación de producto, el nivel de calidad del producto sea determinado de acuerdo las categorías satisfechas o no satisfechas reflejadas en la Tabla 15.

Funcionalidad + categoría “condicional” evaluada	Segunda categoría evaluada	Tercera categoría evaluada	Nivel de calidad del producto de software
<i>Satisfecha</i>	<i>No satisfecha</i>	<i>No satisfecha</i>	<i>Básico</i>
<i>Satisfecha</i>	<i>Satisfecha</i>	<i>No satisfecha</i>	<i>Intermedio</i>
<i>Satisfecha</i>	<i>No satisfecha</i>	<i>Satisfecha</i>	<i>Intermedio</i>
<i>Satisfecha</i>	<i>Satisfecha</i>	<i>Satisfecha</i>	<i>Avanzado</i>

Tabla 15 – Nivel de calidad del producto propuesto respecto a categorías satisfechas para el producto adaptado al modelo propuesto. Fuente: Adaptado de Mendoza et al. (2005)

Como se mencionó anteriormente, la categoría Eficiencia, se convierte en la categoría “condicional” para la aplicación del modelo propuesto. Bajo esta premisa,

los casos de estudio que cumplen con la condición indicada son los mismos que hasta ahora se mantienen en la evaluación, esto es, los casos 1, 2 y 4, dado que son aceptables tanto en las categorías Funcionalidad como en Eficiencia.

Finalmente se aplica lo propuesto en la tabla 15 para la presente investigación, quedando el nivel de calidad del producto que representa cada uno de los estudios de caso que se mantienen en la evaluación de acuerdo a lo previsto en la Tabla 16.

Estudio de Caso		Funcionalidad + Eficiencia	Usabilidad	Fiabilidad	Nivel de calidad
1 - HP Load Runner	Porcentaje obtenido	85,75% y 75%	75,96%	68,75%	Intermedio
	Criterio de satisfacción	<i>Satisfecha</i>	<i>Satisfecha</i>	<i>No satisfecha</i>	
2 - Silk Performer	Porcentaje obtenido	76,4% y 75%	54,81%	75%	Intermedio
	Criterio de satisfacción	<i>Satisfecha</i>	<i>No Satisfecha</i>	<i>Satisfecha</i>	
4 - SoapUI Pro	Porcentaje obtenido	75,7% y 81,25%	77,88%	75%	Avanzado
	Criterio de satisfacción	<i>Satisfecha</i>	<i>Satisfecha</i>	<i>Satisfecha</i>	

Tabla 16 - Nivel de calidad de los estudios de caso respecto a categorías satisfechas para el producto adaptado al modelo propuesto. Fuente: Elaboración propia.

De acuerdo a lo demostrado en la Tabla 16, el estudio de caso que presenta un nivel de calidad **Avanzado** es el caso 4 representado por la herramienta SoapUI Pro del fabricante Eviware con tecnología Open source, ya que todas las categorías del modelo fueron satisfechas en la evaluación realizada por el autor. Los estudios de caso 1 y 2, HP Load Runner y Silk Performer respectivamente, arrojaron un nivel de calidad del producto **Intermedio**. Por lo tanto, las PYMES desarrolladoras de software en Venezuela pueden hacer uso de esta herramienta de prueba sin acarrear mayores costos imputados en los productos por ellas desarrollados.

A continuación se realiza la evaluación del modelo propuesto.

6.2 Evaluación de las características generales

Las características generales del modelo evaluadas se corresponden con los primeros niveles de MOSCA, que son categorías, características, sub-características y sub-sub-características. Sin embargo, se presenta el análisis sólo de los niveles “Características” y “Categorías”, por ser los comunes en todas las categorías del modelo y resumen el comportamiento de las niveles siguientes. En el anexo VI se presenta el detalle de los resultados de todos los niveles evaluados.

6.2.1 Evaluación de las categorías del modelo

La primera evaluación está relacionada a las categorías del modelo (Nivel 1 de MOSCA) que son Funcionalidad, Usabilidad, Fiabilidad y Eficiencia. Los resultados se muestran en el gráfico (Figura 26).

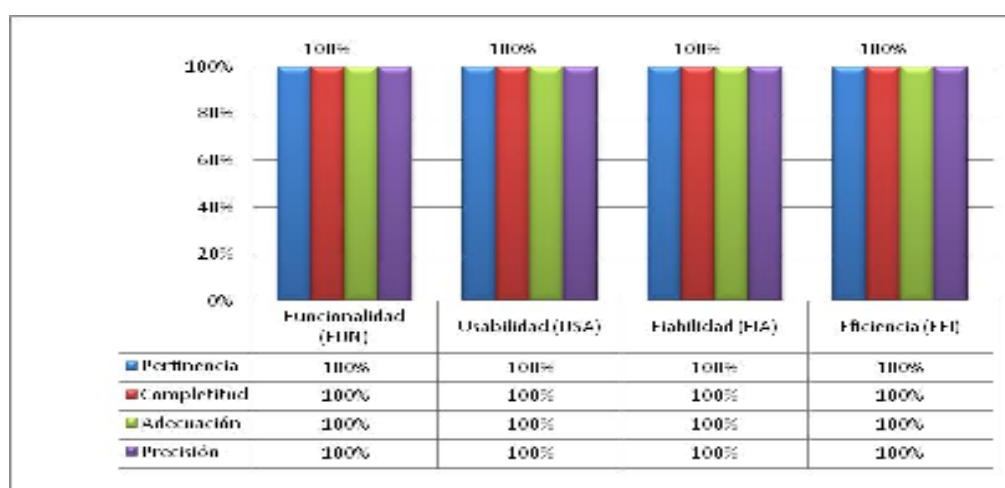


Figura 27- Resultados de la evaluación de “Categorías” del modelo propuesto.
Fuente: Elaboración propia.

Como puede observarse en la Figura 26, todas las categorías alcanzaron el máximo porcentaje en la evaluación (100%). Esto demuestra que las categorías Funcionalidad, Usabilidad, Fiabilidad y Eficiencia se consideraron pertinentes, completas en cuanto a su especificación, adecuadas en el contexto de evaluación y precisas en el resultado alcanzado. Por lo tanto, el modelo se considera **Aceptable** de acuerdo a lo establecido en el Proceso de Estudio de Caso, específicamente la Actividad 2 “Identificar el conjunto de características que permiten la evaluación”.

6.2.2 Evaluación de las características del modelo

Seguidamente se presenta la segunda evaluación relacionada con las características específicas que corresponde a cada una de las categorías mostradas en la sección anterior, como lo son:

- Funcionalidad (FUN): FUN 1. Ajuste a los propósitos, FUN 2. Precisión, FUN 3. Interoperabilidad y FUN 4. Seguridad.
- Usabilidad (USA): USA 1. Facilidad de Comprensión, USA 2. Capacidad de Aprendizaje, USA 3. Interfaz Gráfica y USA 4. Operabilidad.
- Fiabilidad (FIA): FIA 2. Tolerancia a Fallas y FIA 3. Recuperación.
- Eficiencia (EFI): EFI 1. Comportamiento del tiempo y EFI 2. Utilización de recursos.

Las figuras 27, 28 y 29 muestran los resultados obtenidos en cada una las características correspondientes. Estos resultados se presentan en base a los datos mostrados en el ANEXO VII.

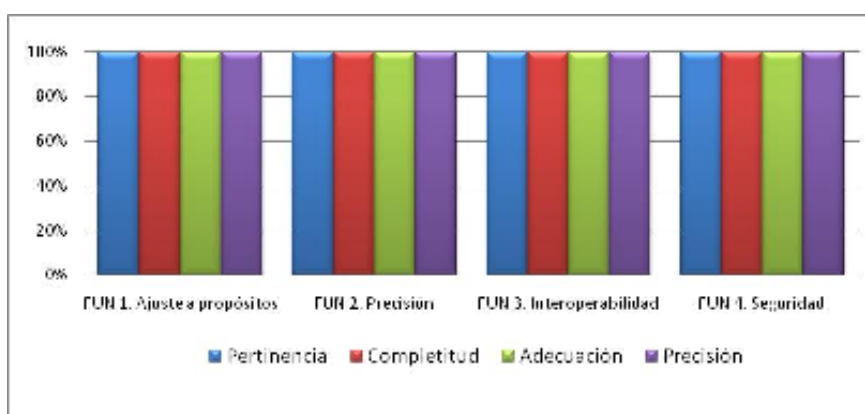


Figura 28 - Resultados de la evaluación de "Características" de la categoría Funcionalidad del modelo propuesto. Fuente: Elaboración propia

Como puede observarse en la Figura 27, todas las características de la categoría funcionalidad, alcanzaron el máximo porcentaje en la evaluación (100%). Esto demuestra que las características relacionadas a la categoría Funcionalidad, se consideraron: pertinente dentro del proceso de especificación de calidad, completa en cuanto a su especificación, adecuadas en el contexto de evaluación y precisas en el resultado alcanzado.

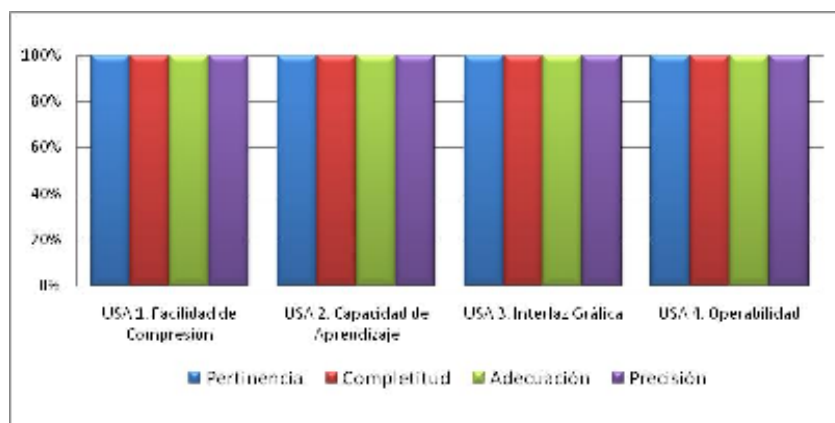


Figura 29 - Resultados de la evaluación de “Características” de la categoría Usabilidad del modelo propuesto. Fuente: Elaboración propia.

De acuerdo a la Figura 28, al igual que la categoría Funcionalidad, las características de la categoría Usabilidad alcanzaron el máximo porcentaje en la evaluación (100%) que declaran ser Aceptable para dicha categoría, por lo que efectivamente se consideraron: pertinente dentro del proceso de especificación de calidad, completa en cuanto a su especificación, adecuadas en el contexto de evaluación y precisas en el resultado alcanzado.

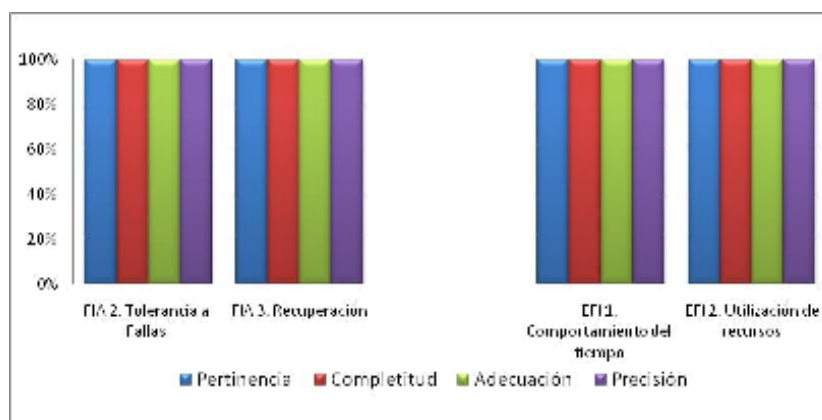


Figura 30 - Resultados de la evaluación de “Características” de la categoría Fiabilidad y Eficiencia del modelo propuesto. Fuente: Elaboración propia

En la Figura 29, se muestran las características propias correspondientes a las categorías Fiabilidad y Eficiencia respectivamente. Para la categoría Fiabilidad, las dos características evaluadas obtuvieron un 100% de aceptación. Para la categoría Eficiencia, ambas características alcanzaron el máximo porcentaje en la evaluación. Por lo anterior, se deduce que ambas categorías están en el nivel Aceptable de acuerdo al promedio de todas las respuestas para dicha categoría, por lo que dichas características se consideraron: pertinente dentro del proceso de especificación de

calidad, completa en cuanto a su especificación, adecuadas en el contexto de evaluación y precisas en el resultado alcanzado.

De acuerdo a lo analizado en base a las figuras 27 a 29, se concluye que todas las características evaluadas para el modelo son **Aceptables** en cuanto a pertinencia, completitud, adecuación y precisión, puesto que están en el máximo rango establecido para ser Aceptables de acuerdo a lo establecido en la Actividad 2 del Proceso de Estudio de Caso. Este mismo comportamiento en cuanto a aceptación se refiere, se deriva hacia las sub-características y sub-subcaracterísticas propuestas en el modelo (en la categoría Funcionalidad). El detalle de los resultados pueden observarse en el ANEXO VII. Esto conlleva a decir que el modelo propuesto en sus características generales es considerado Pertinente, Completo, Adecuado y Preciso.

Concluida la evaluación de las características generales hasta las más específicas del modelo, se procede a evaluar las métricas del modelo propuestas.

6.3 Evaluación de las Métricas del modelo

Es esta sección se presentan los resultados obtenidos luego de la evaluación de las métricas del modelo con su respectivo análisis. En la Figura 30, se presenta un resumen de los promedios obtenidos en la evaluación de las métricas del modelo, agrupadas por característica a evaluar dentro de cada Categoría del modelo. Los insumos para éste resultado pueden observarse en el Anexo VIII.

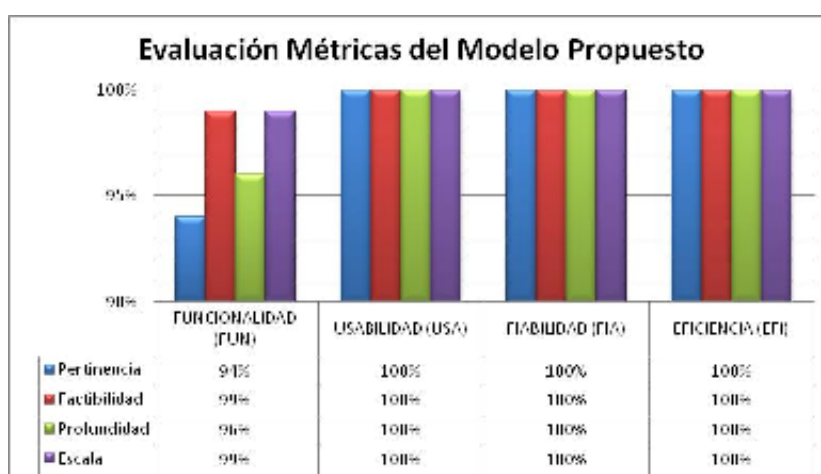


Figura 31 - Resultados de la evaluación de “Métricas” por Categoría del modelo propuesto.
Fuente: Elaboración propia.

Como puede observarse en la figura 30, el conjunto de métricas para cada Categoría en promedio superó el porcentaje esperado para dichas métricas, es decir, entre 75% y 100%, por lo que de acuerdo a los criterios de aceptación, se considera que el modelo propuesto en cuanto a métricas se refiere es **Aceptable** de acuerdo a los criterios establecidos en la Actividad 2 del Proceso de Estudio de Caso. Éste resultado se logra detallando cada característica específica de la siguiente manera:

- En cuanto a la *Pertinencia*, todas las métricas se consideran pertinentes en cada una de las categorías del modelo, por estar encima del mínimo esperado.
- Todas las métricas son consideradas *Factibles* en casi un 100% para cada una de las categorías del modelo.
- En el caso de la *Profundidad*, para las categorías Usabilidad, Fiabilidad y Eficiencia sus métricas correspondientes se consideran profundas, mientras que en la categoría Funcionalidad, las métricas propuestas están muy cercanas al mínimo establecido, pero siguen siendo Aceptables.
- Para los evaluadores, la *Escala* es apropiada en un 100% para todas las Categorías del modelo.

Con el fin de que todas las métricas lleguen al 100%, se proponen las siguientes modificaciones en las métricas del modelo de acuerdo a las clasificaciones obtenidas en el ANEXO VIII:

- “Reutilización de Casos de Prueba entre proyectos”, “Reutilización de Datos de Prueba entre proyectos” y “Reutilización de Scripts de Prueba entre proyectos” se consideraron no pertinentes pues no permiten definir la existencia o no de la característica que representan, las cuales son Casos de prueba FUN 1.6.1., Datos de Prueba (FUN 1.6.2) y Scripts de Prueba (FUN 1.6.3) respectivamente. Se propone trasladarlas a la característica Operaciones con terceros FUN 3.1.2.
- “Definición de estrategias de pruebas” de la característica Condiciones a probar en la Aplicación objeto de prueba (FUN 1.7.2) se consideró poco profunda pues no es muy específica en lo que busca lograr. Se propone redactarla como: Parametrización de estrategias de pruebas para condicionar la prueba.

- “Notifica superación de umbrales al momento” de la característica Estadísticas de resultados (FUN 1.8.1) muestra ser no profunda porque no especifica que el mecanismo de notificación sea visible; por ejemplo, puede notificar en un log, que no permite darse cuenta en tiempo real. Se propone redactarla como: Provee mecanismos visibles de notificación de umbrales al momento.
- “Formato de generación de Reportes/entregables son similares entre sí” se consideró no pertinente para evaluar la característica Manipulación de resultados (FUN 1.8.3), por lo que se propone trasladarla a la característica Interfaz Gráfica (USA 3).
- “Sincronización de Scripts de Prueba por nuevas versiones en la Aplicación objeto de prueba” de la característica Interacción con la Aplicación objeto de prueba (FUN 1.9.1) se consideró no factible porque es muy difícil medir este resultado pues puede que se actualice pero los scripts generados carezcan de calidad; se propone modificarlo como: Permite actualizar los scripts de pruebas generados ante actualización de la aplicación objeto de pruebas.
- “Control de acceso” de la característica control de acceso (FUN 4.1) se consideró no profunda, pues carece de especificidad. Se propone modificarla así: Control de acceso inicial a la herramienta.

CAPÍTULO VII: CONCLUSIONES Y RECOMENDACIONES

En esta sección, se presentan las ideas principales de cada uno de los capítulos que conforman la presente investigación, para plasmar las conclusiones producto de la experiencia obtenida a lo largo del desarrollo de este trabajo de maestría, por una parte, y por la otra, se exponen las recomendaciones a fin de continuar con ésta línea de investigación en futuros trabajos o mejoras del modelo propuesto.

7.1 Conclusiones

El tema de las *pruebas del software* con todas sus etapas es una realidad que debe estar presente en cualquier empresa que desee adentrarse en el mundo del desarrollo del software y, por supuesto, la calidad en los productos que éstas generan. Esta calidad esperada es en gran parte lograda a través de la formalización de la Disciplina de Pruebas en el proceso de desarrollo de software. Dedicar esfuerzos en este tema, son bien justificados para el desarrollo de la industria del software en nuestro país, elaborando nuevos y efectivos métodos adaptados a las características de las empresas desarrolladoras de software del país, específicamente las PYMES.

Un extenso recorrido y análisis de bibliografía relevante tomando en cuenta antecedentes de investigaciones propuestas por varios investigadores concernientes al tema de estudio y una completa definición del marco organizacional enmarcado en bases éticas y legales son presentados en el capítulo II. Allí se expresan conocimientos que van desde la definición de la calidad y calidad del software hasta la importancia que tienen las pruebas del software y el uso de modelos de calidad.

La metodología Investigación-Acción fue un factor relevante para el éxito de esta investigación, pues provee herramientas que unificadas con el método de evaluación DESMET forman el completo Framework Metodológico del Laboratorio de Investigación en Sistemas de Información LISI que permite secuenciar en forma ordenada y metódica cada uno de los pasos y/o procesos necesarios para cumplir con los objetivos propuestos a la vez de llegar rápida y acertadamente a la fase de “Tomar la acción” insumo para la evaluación de dicha acción y la especificación final del aprendizaje obtenido, con la retroalimentación necesaria.

El desarrollo de la propuesta en sí (capítulo IV) estuvo coordinado con los objetivos específicos realizando cuatro grandes aportes al conocimiento en el área de los sistemas de información:

- a) Modelo conceptual que abarca las principales definiciones en cuanto a pruebas de software se refiere, producto de un extenso análisis de la bibliografía y las experiencias con herramientas de pruebas existentes en el mercado.
- b) Árbol de clasificación de herramientas de pruebas automatizadas con su correspondiente instanciación de acuerdo a las necesidades de investigación, específicamente herramientas para el nivel de pruebas del sistema del tipo no funcionales: Desempeño y carga, bajo un enfoque de Caja Negra.
- c) Modelo de referencia propuesto para seleccionar herramientas de pruebas que apoyan empresas representadas por las PYMEs desarrolladoras de software, producto de aportes al modelo de calidad sistémica MOSCA para el producto.
- d) Guía para aplicar el modelo propuesto a herramientas de prueba existentes en el mercado tecnológico, previa selección de acuerdo al árbol de clasificación.

El modelo propuesto fue evaluado de acuerdo al proceso de evaluación establecido por DESMET que va desde el análisis de contexto hasta su aplicación de acuerdo al método arrojado “Análisis de características por estudio de caso”. Primeramente el modelo se aplicó en cuatro estudios de casos representados por herramientas de pruebas meticulosamente elegidas del mercado tecnológico de acuerdo a ciertos criterios. Una vez adquirida experiencia en el modelo y garantizada su efectividad, se procedió a evaluarlo en sí considerando cada uno de los niveles propuestos (categorías, características, sub-características, sub-sub-características y métricas) como pertinentes, completos, adecuados y precisos en el entorno de evaluación. El modelo fue producto de exhaustivas revisiones y mesas de trabajo con el patrocinador (LISI) cuyos integrantes con su gran experiencia, retroalimentaron y evaluaron el modelo hasta refinar el modelo propuesto.

Dicho modelo incluye aportes realizados por el investigador en cuanto a la incorporación de nuevas sub-características, sub-sub-características y métricas. Igualmente, se propusieron modificaciones a algunas métricas del modelo propuesto, puesto que en la evaluación de las mismas, no se consideraron pertinentes y completas en un 100%. Cabe destacar que también se propuso una adaptación del algoritmo de

aplicación de MOSCA en la determinación del nivel de calidad del producto dado que éste modelo se basa en tres categorías y no se correspondía con las cuatro categorías que contiene el modelo propuesto.

De los cuatro estudios de casos, representados por las herramientas HP LoadRunner (1), SilkPerformer (2), IBM/Rational Performance Tester (3), SoapUI Pro (4), se excluyó el estudio de caso 2 por no cumplir con la categoría Funcionalidad de acuerdo al algoritmo de MOSCA. De los tres estudios de caso restantes, se obtuvo que el que tiene un Nivel de calidad *Avanzado* es el 4 “SoapUI Pro” del fabricante Eviware con tecnología Open source, por lo que las PYMES desarrolladoras de software en Venezuela pueden hacer uso de esta herramienta de prueba sin acarrear mayores costos imputados en los productos por ellas desarrollados. Los estudios de caso 1 y 2, HP Load Runner y Silk Performer respectivamente, arrojaron un nivel de calidad del producto *Intermedio*.

7.2 Recomendaciones

De acuerdo a los resultados obtenidos, se consideran las siguientes recomendaciones para futuros trabajos de investigación:

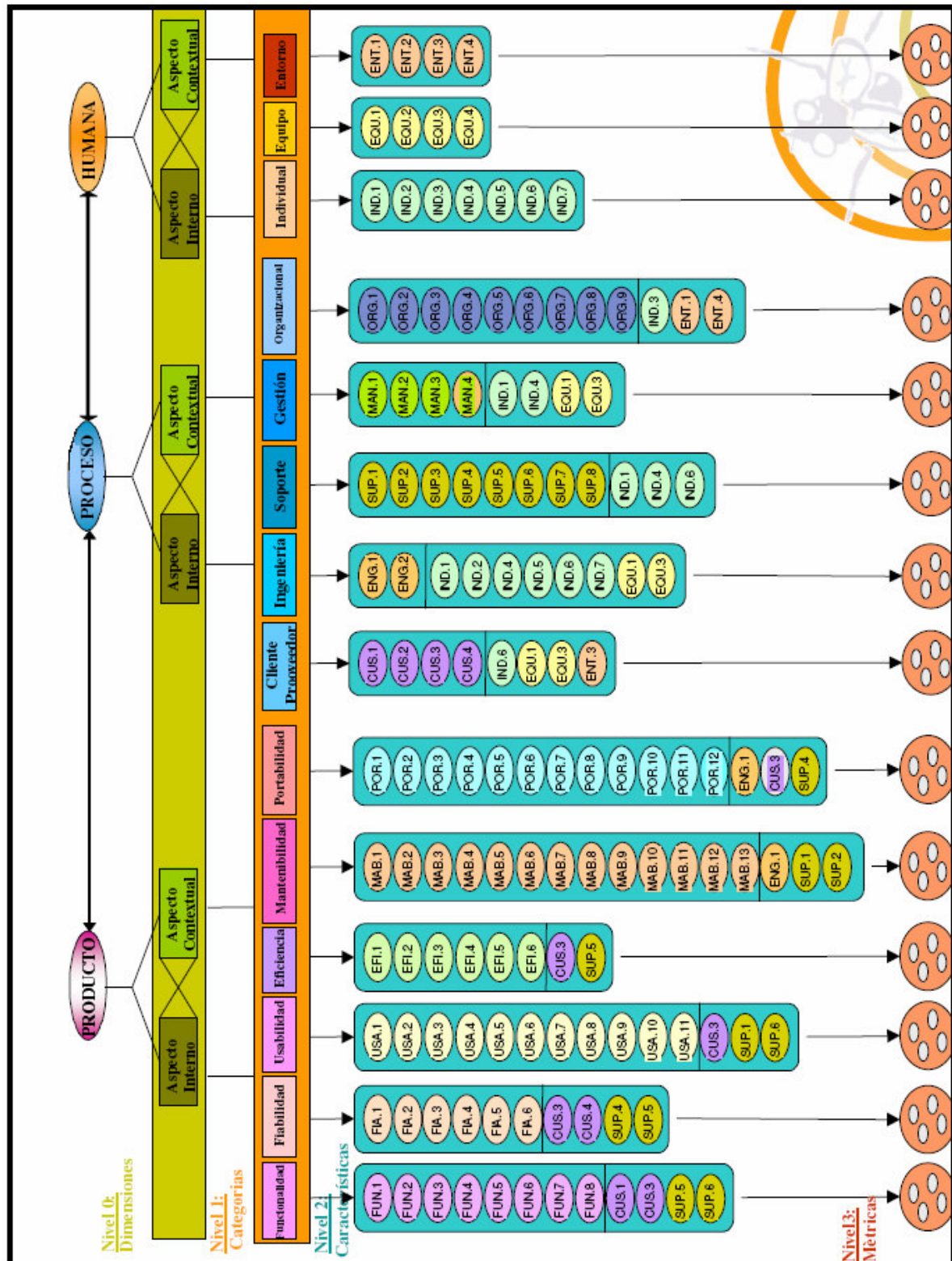
Ampliar el entorno de aplicación del modelo propuesto para abarcar la mayor cantidad de herramientas de pruebas que existen en el mercado con su adecuada clasificación, y que a su vez, suplen todos los tipos de pruebas, recalcando las características propias y distintivas de cada tipo de prueba que no necesariamente aplican a otro tipo de pruebas. Por esta razón, es recomendable que la aplicación del modelo sea realizada por personal experto en pruebas que tenga muy claro el objeto de evaluación y/o propósito de las pruebas enmarcadas en un proyecto de pruebas para hacer una acertada selección previa de herramientas que satisfagan los requerimientos explícitamente planteados por la organización para su implementación.

Profundizar el árbol de clasificación de herramientas de pruebas automatizadas en el mercado, propuesto por el investigador, a fin de poder estructurar y guiar la construcción de un modelo que cumpla con las indicaciones del punto anterior, haciendo énfasis en las ramas excluyentes (procurando mostrar todas las opciones

posibles) y en las ramas incluyentes (previendo las múltiples combinaciones que pueden surgir). Para esto, se recomienda proponer un Sistema de soporte de decisiones para obtener los mejores resultados al momento de obtener una o más instancias del árbol de clasificación de herramientas propuesto.

Promover el modelo de selección de herramientas de pruebas propuesto a la mayor cantidad de PYMES venezolanas desarrolladoras de software a fin de garantizar la mayor calidad en los productos que éstas desarrollan, enfatizando en la facilidad de la aplicación del modelo y sus innumerables beneficios, sensibilizando a este sector sobre la importancia de dar valor a la disciplina de pruebas como ventaja competitiva entre sus homólogos.

ANEXO I: Modelo de Calidad MOSCA



ANEXO II: Aspectos de la categoría FUNCIONALIDAD tomadas de MOSCA

Característica	Métrica	Pregunta	Formulación	Dirigidos a:
Funcionalidad(FUN)				
FUN.1 Ajuste a los propósitos	Cumplimiento de las necesidades funcionales	¿El producto cumple con todas las necesidades funcionales?	5= TODAS 4= CASI TODAS 3= MUCHAS 2= MUY POCAS 1= NINGUNA	USUARIO
	Definición de las necesidades de adquisición, desarrollo o mejora	¿Se tienen establecidas las necesidades de adquisición, desarrollo o mejora del sistema (software, hardware y personal)?	5= Completam. def. 4= Casi todo def. 3= Med. Def. 2= Poco def. 1= No están def.	LIDER DESARROLLADOR
	Definición de la estrategia de adquisición o desarrollo de nuevos sistemas	¿Existe una estrategia desarrollada para la adquisición?	5= Completam. def. 4= Casi todo def. 3= Med. Def. 2= Poco def. 1= No están def.	LIDER DESARROLLADOR
	Definición del criterio de aceptación	¿Existe una definición del criterio de aceptación?	5= Completam. def. 4= Casi todo def. 3= Med. Def. 2= Poco def. 1= No están def.	LIDER DESARROLLADOR
	Registro de los resultados de la prueba	¿Los resultados de la prueba fueron registrados?	5= TODAS 4= CASI TODAS 3= MUCHAS 2= MUY POCAS 1= NINGUNA	DESARROLLADOR
FUN.2 Precisión	Resultados incompletos	¿Existe falta de precisión en la función debido a resultados incompletos?	5= TODAS 4= CASI TODAS 3= MUCHAS 2= MUY POCAS 1= NINGUNA	USUARIO
	Resultados incorrectos	¿Existe falta de precisión en la función debido a resultados incorrectos?	5= TODAS 4= CASI TODAS 3= MUCHAS 2= MUY POCAS 1= NINGUNA	USUARIO
	Resultados emitidos no esperados	¿Existe falta de precisión en la función debido a la emisión de resultados no esperados?	5= TODAS 4= CASI TODAS 3= MUCHAS 2= MUY POCAS 1= NINGUNA	USUARIO
	Criterios de validación	¿Existen criterios de validación para el producto?	5= Completam. def. 4= Casi todo def. 3= Med. Def. 2= Poco def. 1= No están def.	DESARROLLADOR
FUN. 3 Interoperabilidad	Existencia de funcionalidades pertenecientes al producto que son utilizadas por otros sistemas	¿Existen funcionalidades utilizadas por otros sistemas, que pertenecen al producto?	5= no tiene 1= tiene	DESARROLLADOR / USUARIO
	Intercambio de datos	¿Existe intercambio de datos con otros sistemas?	5= no tiene 1= tiene	DESARROLLADOR / USUARIO
	Consistencia de interfaces	¿Existe consistencia con las interfaces de los otros sistemas?	5= es consistente 1= no es consistente	DESARROLLADOR
	Existencia de funcionalidades de otros sistemas que sean necesarias	¿Existen funcionalidades en otros sistemas que son necesarias?	5= no tiene 1= tiene	USUARIO / DESARROLLADOR
FUN.4 Seguridad	Control de acceso	¿Qué tan completa es la detección de acceso a usuarios del sistema, cuando se trata de obtener acceso al sistema de forma no autorizada?	5= Muy alta 4= Alta 3= Mediana 2= Básica 1= No tiene.	DESARROLLADOR / USUARIO
	Capacidad de auditoría	¿Qué tan completa es la auditoría? ¿Se evalúa la cantidad de registros de acceso que el sistema registró en la base de datos histórica?	5= Muy alta 4= Alta 3= Mediana 2= Básica 1= No tiene.	DESARROLLADOR

ANEXO III: Aspectos de la categoría USABILIDAD tomadas de MOSCA

Característica	Métrica	Pregunta	Formulación	Dirigidos a:
Usabilidad (USA)				
USA.1 Facilidad de comprensión	Compleitud de las demostraciones funcionales	¿Están completas las demostraciones funcionales implementadas respecto al número de funciones que requieren la demostración de uso?	5=Todas 4=Casi todas 3= Algunas 2= Pocas 1= Ninguna	DESARROLLADOR USUARIO
	¿Cuál es el tiempo en que un usuario adquiere las destrezas necesarias para usar el sistema?	¿Cuál es el tiempo en que un usuario adquiere las destrezas necesarias para usar el sistema?	5= 6 días o menos 4= 7-14 días 3= 15-30 días 2= 2-5 meses 1= 6 meses o más	USUARIO
	Nivel de dificultad de la aplicación	Nivel de dificultad de la aplicación	5=Muy difícil 4=Difícil 3=Promedio 2=Fácil 1=Muy fácil	USUARIO
	Facilidad para ubicar funcionalidades	¿Es fácil ubicar las funcionalidades del sistema?	5= Muy fácilmente 4= Fácilmente 3= Dificultad media 2= Difícil 1= Muy difícil	USUARIO
USA.2 Capacidad de aprendizaje	Calidad en la clasificación de los temas	¿Es fácil utilizar el material de apoyo, por una buena clasificación de los temas	5=Excelente 4=Buena 3= Promedio 2=Debajo del Prom. 1= Inaceptable.	USUARIO
	Compleitud de la especificación de los documentos	¿Están identificados todos los documentos a ser producidos por el proceso o proyecto?	5=Todos 4=Casi todos 3= Algunos 2= Pocos 1= Ninguno	LIDER DESARROLLADOR USUARIO
USA.3 Interfaz Gráfica	Interfaz Personalizable	¿La interfaz es personalizable?	5= Si 1= No	USUARIO
	Localización rápida de opciones	¿Se localizan rápidamente las opciones, es consistente la ubicación de utensilios?	5= Si 1= No	USUARIO
	Presencia de metáforas	¿Hay presencia de metáforas?	5= Si 1= No	USUARIO
	Satisfacción del Diseño visual	¿Es bueno el diseño visual de las pantallas?	5=Excelente 4=Buena 3=Promedio 2=Debajo del Prom. 1= Inaceptable	USUARIO
	Versatilidad de la navegación	¿Es versátil la navegación, entre pantallas?	5=Excelente 4=Buena 3=Promedio 2=Debajo del Prom. 1= Inaceptable	USUARIO
USA.4 Operabilidad	Satisfacción con las ayudas disponibles respecto a las requeridas	¿Cómo es la satisfacción con las ayudas disponibles respecto a las requeridas?	5=Excelente 4=Buena 3=Promedio 2=Debajo del Prom. 1= Inaceptable	USUARIO
	Tasa de disponibilidad de valores por defecto	¿Existen valores por defecto en las pantallas?	Nº de valores x defecto / Nº entradas la pantalla	DESARROLLADOR
	Evaluación continua de la Satisfacción del cliente	¿Existe una evaluación continua de la satisfacción del cliente respecto a los servicios de apoyo y al producto?	5=Siempre 4=Casi siempre 3=Algunas veces 2=Pocas veces 1=Nunca	LIDER DESARROLLADOR USUARIO
	Soporte operacional continuo	¿Se provee apoyo para resolver problemas operacionales según requerimientos del cliente?	5=Siempre 4=Casi siempre 3=Algunas veces 2=Pocas veces 1=Nunca	LIDER DESARROLLADOR USUARIO
	Operación apropiada del software en su ambiente	¿Existe una correcta operación del software en su ambiente?	5= Si 1= No	DESARROLLADOR USUARIO
	Asistencia a los clientes en cuanto a operación	¿Existe asistencia a los clientes en cuanto a su operación?	5= Si 1= No	USUARIO

ANEXO IV: Aspectos de la categoría FIABILIDAD y EFICIENCIA tomadas de MOSCA

Fiabilidad (FIA)				
FIA.2 Tolerancia a Fallas	Anulación de operación incorrecta	¿Qué tan frecuente puede el producto de software <i>evitar</i> fallas, aún si el usuario lo opera incorrectamente?	5=Siempre 4=Casi siempre 3= Algunas veces 2= Pocas veces 1= Nunca	DESARROLLAD OR
FIA.3 Recuperación	Capacidad de reiniciar	¿Puede el producto de software recuperarse fácilmente después de una caída?	5=Muy fácilmente 4=Fácilmente 3=Dificultad Media 2= Difícil 1= Muy difícil	DESARROLLAD OR USUARIO
	Velocidad de reinicialización	Velocidad de la reinicialización.	5=Muy rápida 4=Rápida 3= Promedio 2=Lenta 1= Muy lenta	DESARROLLAD OR
	Disponibilidad del producto de software	¿Pueden los usuarios trabajar con el producto de software el tiempo necesario?	5=Siempre 4=Casi siempre 3= Algunas veces 2= Pocas veces 1= Nunca	DESARROLLAD OR USUARIO

Eficiencia (EFI)				
EFI.1 Comportamiento del tiempo	Tiempo de respuesta	Tasa del tiempo de respuesta para lo que es diseñado a ser ejecutado	Tiempo de respuesta real/tiempo de respuesta requerido	DESARROLLAD OR
	Satisfacción con el número de tareas completadas en un tiempo especificado	Satisfacción con el número de tareas que pueden ser completadas durante un tiempo especificado con respecto al número de tareas que son requeridas a ser completadas	5=Muy Alta 4=Alta 3= Media 2= Baja 1= Muy baja	DESARROLLAD OR
EFI.2 Utilización de recursos	Utilización del CPU	Tasa de tiempo de CPU utilizado	Tiempo de CPU utilizado / Tiempo de CPU requerido	DESARROLLAD OR
	Utilización de memoria	Tasa del tamaño de memoria utilizada con respecto al tamaño de la memoria que se requiere que sea usada	Memoria utilizada / Requerimientos del uso de memoria	DESARROLLAD OR

ANEXO V: Descripción de algunas métricas aportadas por el autor a MOSCA en el contexto del modelo propuesto

Característica/métrica	Descripción
Generación automática de Casos de prueba	Generación automática de casos de prueba y/o escenarios que se van a probar de acuerdo a datos de entrada específicos.
Generación de evidencias de pruebas	Registra evidencias de pantallas, resultados, eventos en forma automática de acuerdo a parámetros
Generación automática de datos de pruebas	Permite generar/construir el set de datos automático, réplicas de los existentes. Poblamiento de datos automáticos
Gestión de defectos	Registra los defectos automáticamente, con cierto detalle (momento de ocurrencia, posible origen, trazabilidad, entre otros)
Reportes/entregables mantienen el mismo formato	Consistencia en el formato de los reportes que genera la herramienta de prueba
Programación de Scripts de pruebas en el tiempo (scheduler)	Una vez ordenada la prueba, permite iniciar la ejecución automáticamente de acuerdo al tiempo previamente establecido
Variación de carga de trabajo durante la prueba	Factores que cambiará durante la prueba de rendimiento. Ejm., variar el número de usuarios para comprender cómo se degrada el tiempo de respuesta a medida que la carga de trabajo aumenta
Log de Auditoría	Permite guardar un log de las pruebas ejecutadas y los pasos de la misma
Criterios de prueba superada y no superada	Permite determinar/establecer cuándo una prueba se considera superada y cuándo no a través de porcentajes, umbrales, relaciones entre variables, etc.
Estadísticas de resultados permanente	Acceso permanente a los resultados obtenidos. Se mantiene un BD de los resultados en cualquier momento.
Estadísticas en tiempo real	Genera estadísticas al momento sin necesidad de detener la prueba
Notifica superación de umbrales en la aplicación	Genera alertas, mensajes o distinción ante la cercanía de los umbrales permitidos y/o establecidos por la aplicación
Generación de métricas	Permite establecer. Configurar, parametrizar indicadores o métricas esperadas
Relaciona defectos a su causa	Determina (al menos a un alto nivel) la posible causa del defecto
Análisis de resultados automáticos	Analiza los resultados con conclusiones generales, permitiendo configurar los resultados a conveniencia del usuario
Sincronización de Scripts de Prueba por nuevas versiones en la Aplicación objeto de prueba	Ante una nueva versión de la aplicación objeto de pruebas, permite realizar cambios a lo configurado sin demasiados pasos adicionales. Se relaciona sin problemas con la aplicación, es decir, se monta sobre la aplicación sin problemas, sus funciones, capacidades, etc.
Emula población/acceso de usuarios concurrentes	Simula el acceso y uso de la aplicación objeto de prueba desde varias entradas (usuarios concurrentes) con cargas de trabajo múltiples
Modificar scripts de pruebas en línea/ejecución	Permite hacer cambios mientras se hace la ejecución de la prueba a fin de obtener mejores resultados
Modificar scripts de pruebas luego de generados	Scripts dinámicos que pueden ser modificados luego de su creación
Reutilización de Scripts entre proyectos	Permite usar scripts definidos para un proyecto, en otro proyecto con sus respectivos cambios (importar scripts)
Importar scripts de otras aplicaciones/lenguajes	Permite importar scripts desde otras aplicaciones/herramientas de pruebas aún con lenguajes de programación distintos
Ante errores/defectos permite continuar	No se detiene la ejecución de pruebas por ciertos errores, sino que lo reporta y continua con el proceso
Provee Wizard para ciertas funcionalidades	Ayudas en la instalación/utilización/configuración de ciertas bondades de la herramienta de prueba
Provee ayudas en línea	Ofrece ayuda para las diversas funcionalidades de la herramienta
Barra de herramientas (entorno) customizables	Tiene barra de herramientas y se puede personalizar por perfiles de usuarios (colores, entorno, funciones/módulos activos, etc.)
Código único/proceso unificado para multiplataformas	Proporciona un lenguaje de programación único para la construcción de casos de prueba y script de pruebas (ejemplo XML)

Código Propietario (propone comandos, funciones propietarias)	La herramienta trabaja con código de programación propietario, único para la herramienta, con sentencias propias inherentes a la herramienta. Propone su propio pseudo-lenguaje de programación
Necesita amplios conocimientos en programación	Requiere conocimientos de programación a un nivel específico para poder construir los script/casos de prueba
Facilidad en el manejo de la aplicación	Es manejable la aplicación, su recorrido es sencillo
Ofrece ejemplos, valores pre-cargados en las pantallas	Proporciona guías para llenar campos y/o generar algunas funcionalidad
Componentes/plugins que complementa la herramienta	Tiene algún módulo, componente, sub-aplicación, extensiones, plugins que la conforma y/o aumenta sus potencialidades
Capacidad de guardar pruebas para continuar después	Permite detener una prueba, guardar su ejecución y continuar más adelante
Control de acceso a los usuarios	Mantiene registro/auditoría del acceso de los usuarios que usan el sistema
Definición de roles dentro del sistema	Permite configurar e implementar roles por usuario donde cada uno tiene su propia configuración o permisología para ciertas funcionalidades

ANEXO VI: Evaluación detallada de las Herramientas de prueba seleccionadas de acuerdo al Modelo propuesto

Estudios de Caso:

HP LoadRunner (1), SilkPerformer (2), IBM/Rational Performance Tester (3), SoapUI Pro (4)

Sub-característica	Sub Sub-característica	Métrica	Pregunta	Formulación	Dirigido a	Estudio de Caso			
						1	2	3	4
FUNCIONALIDAD (FUN) - Característica: Ajuste a los propósitos (FUN 1)									
		Cumplimiento de las necesidades funcionales	¿El producto cumple con todas las necesidades funcionales?	5= Todas 4= Casi todas 3= Muchas 2= Muy pocas 1= Ninguna	Usuario	5	5	5	5
		Definición de las necesidades de adquisición, desarrollo o mejora	¿Se tienen establecidas las necesidades de adquisición, desarrollo o mejora del sistema (software, hardware y personal)?	5=Completa def. 4=Casi todo def. 3= Med. Def. 2= Poco def. 1= No están def.	lider Usuario	5	5	5	5
		Definición de la estrategia de adquisición o desarrollo de nuevos sistemas	¿Existe una estrategia desarrollada para la adquisición?	5=Completa. def 4=Casi todo def. 3= Med. Def. 2= Poco def. 1= No están def.	lider Usuario	5	5	5	5
		Definición del criterio de aceptación	¿Existe una definición del criterio de aceptación?	5=Completa def. 4=Casi todo def. 3= Med. Def. 2= Poco def. 1= No están def.	lider Usuario	5	5	5	5
		Registro de los resultados de la prueba	¿Los resultados de la prueba fueron registrados?	5=Todas 4=Casi Todas 3=Muchas 2=Muy Pocas 1= Ninguna	Usuario	5	5	5	5
Niveles de Prueba (FUN 1.1)	Pruebas de sistema (FUN 1.1.1)	Ejecución de pruebas de a nivel de sistema	¿Permite realizar pruebas al nivel de pruebas del sistema?	5 = Si 1 = No	Usuario	5	5	5	5
Tipos de Prueba (FUN 1.2)	Pruebas de Desempeño (FUN1.2.1)	Ejecución del Tipo de Pruebas de Desempeño	¿Permite realizar pruebas del tipo Desempeño a la Aplicación objeto de prueba?	5 = Si 1 = No	Usuario	5	5	5	5
	Pruebas de Carga/Estrés (FUN1.2.2)	Ejecución del Tipo de Pruebas de Carga/Estrés	¿Permite realizar pruebas del tipo Carga/Estrés a la Aplicación objeto de prueba?	5 = Si 1 = No	Usuario	5	5	5	5
Enfoque de Prueba (FUN 1.3)	Pruebas de Caja Negra (FUN 1.3.1)	Ejecución de Pruebas con enfoque de Caja Negra	¿Las pruebas son de enfoque de Caja Negra?	5 = Si 1 = No	Usuario	5	5	5	5
		Exploración en el código de la Aplicación objeto de prueba	¿Omite el Profundizar/analizar el código de la Aplicación objeto de prueba?	5 = Si 1 = No	Usuario	5	5	5	5
Licencias de distribución de la Herramienta de Prueba (FUN 1.4)	Open source (FUN 1.4.1)	Distribución de la Herramienta de Prueba Opensource	¿La licencia de la Herramienta de Prueba es de tipo opensource (libre)?	5 = Si 1 = No	Usuario	1	1	1	5
	Comercial (FUN 1.4.2)	Distribución de la Herramienta de Prueba es comercial	¿La licencia de la Herramienta de Prueba es de tipo comercial (privado)?	5 = Si 1 = No	Usuario	5	5	5	1
Administración de la Herramienta de Prueba (FUN 1.5)	Lenguaje de Programación de la Herramienta de Prueba (FUN 1.5.1)	Código único/proceso unificado para multiplataformas	¿Proporciona un lenguaje de programación único para la construcción de entregables?	5 = Si 1 = No	Usuario	1	1	5	5
		Código Propietario (propone comandos, funciones propietarias)	¿Propone su propio pseudo-lenguaje de programación para construir entregables?	5 = Si 1 = No	Usuario	5	5	1	1
		Necesidad de amplios conocimientos en programación	¿Requiere conocimientos de programación a un nivel específico para poder construir los entregables?	5=Básica 3= Mediana 1= Alta	Usuario	3	5	5	3
Entregables	Casos de Prueba	Generación automática	¿Genera automáticamente los	5 = Si	Usuario	5	5	1	5

de Pruebas (FUN 1.6)	(FUN 1.6.1)	de Casos de Prueba	Casos de Prueba de acuerdo a ciertas entradas?	1 = No					
			¿Cómo es la completitud de los Casos de Prueba generados automáticamente en cuanto a: pasos, pre-condiciones, post-condiciones, valores, resultados esperados?	5=Completam 4=Casi todos 3= Med. 2= Pocos 1= No aplica	Usuario	3	4	1	4
		Construcción de Casos de Prueba paso a paso	¿Permite construir Casos de Prueba paso a paso?	5 = Si 1 = No	Usuario	1	5	5	5
		Identificación de los requerimientos de prueba	¿Permite identificar los requerimientos de prueba que se requieren probar?	5 = Si 1 = No	Usuario	5	5	5	5
		Reutilización de Casos de Prueba entre proyectos	¿Los Casos de Prueba sirven para alimentar proyectos afines?	5 = Si 1 = No	Usuario	5	5	1	1
		Documentación de Pre-condiciones	¿Permite documentar las pre-condiciones en los Casos de Prueba?	5 = Si 1 = No	Usuario	5	5	1	1
		Documentación de Post-condiciones	¿Permite documentar las post-condiciones en los Casos de Prueba?	5 = Si 1 = No	Usuario	5	5	1	1
	Datos de Prueba (FUN 1.6.2)	Generación manual de Datos de Prueba desde Casos de Prueba	¿Se establecen manualmente los Datos de Prueba en los Casos de Prueba?	5 = Si 1 = No	Usuario	5	5	5	5
		Generación automática de Datos de Prueba desde datos reales	¿Genera automáticamente los Datos de Prueba desde los datos reales identificados?	5 = Si 1 = No	Usuario	5	5	5	5
		Generación automática de Datos de Prueba desde sentencias SQL	¿Se establecen automáticamente los Datos de Prueba desde sentencias SQL, base de datos?	5 = Si 1 = No	Usuario	5	5	1	5
		Generación automática de Datos de Prueba de forma aleatoria	¿Genera automáticamente los Datos de Prueba aleatoriamente de acuerdo a datos iniciales?	5 = Si 1 = No	Usuario	5	1	5	5
		Reutilización de Datos de Prueba entre proyectos	¿Los Datos de Prueba sirven para alimentar proyectos afines?	5 = Si 1 = No	Usuario	5	5	1	5
	Script de Pruebas (FUN 1.6.3)	Generación automática de Scripts de Prueba desde Casos de Prueba + Datos de Prueba	¿Genera automáticamente los Scripts de Prueba de en base a los Casos de Prueba + Datos de Prueba?	5 = Si 1 = No	Usuario	1	5	1	1
		Construcción de Scripts de Prueba paso a paso	¿Permite construir Scripts de Prueba paso a paso, manualmente?	5 = Si 1 = No	Usuario	5	5	5	5
		Modificar Scripts de Prueba luego de generarlos	¿Permite modificar Scripts de Prueba luego de generarlos?	5 = Si 1 = No	Usuario	5	1	5	5
		Modificar Scripts de Prueba en línea/ejecución	¿Permite modificar Scripts de Prueba al momento de ejecutarlo?	5 = Si 1 = No	Usuario	1	5	5	5
		Reutilización de Scripts de Prueba entre proyectos	¿Permite usar/importar Scripts de Prueba existentes entre proyectos?	5 = Si 1 = No	Usuario	5	5	1	5
	Suite de Pruebas (FUN 1.6.4)	Organizar/secuenciar conjunto de pruebas	¿Permite organizar y/o secuenciar conjunto de Scripts de Prueba?	5 = Si 1 = No	Usuario	5	5	5	5
		Programación de Scripts de Prueba en el tiempo	¿Permite programar pruebas en el tiempo (scheduler)?	5 = Si 1 = No	Usuario	5	5	5	5
	Salidas de pruebas (FUN 1.6.5)	Generación de evidencias de Pruebas	¿Genera evidencias de resultados de pruebas automáticos?	5 = Si 1 = No	Usuario	5	5	5	5
		Gestión de defectos	¿Documenta defectos en forma automática luego de presentado el error?	5 = Si 1 = No	Usuario	5	5	5	5
			¿Permite agregar/crear defectos?	5 = Si / 1 = No	Usuario	5	1	1	5
			¿Permite modificar defectos?	5 = Si / 1 = No	Usuario	5	5	5	5
			¿Permite notificar defectos al responsable de la aplicación/módulo?	5 = Si 1 = No	Usuario	5	1	1	5
		Log de pruebas/histórico de ejecuciones	¿Permite guardar un log de las pruebas ejecutadas?	5 = Si 1 = No	Usuario	5	5	5	5
	Relación entre entregables de Pruebas (FUN 1.6.6)	Relación entre requerimientos de prueba y Casos de Prueba	¿Posee mecanismo para relacionar requerimientos de prueba a los Cados de Prueba?	5 = Si 1 = No	Usuario	5	5	1	5
		Relación entre Casos de	¿Posee mecanismo para	5 = Si	Usuario	5	5	5	5

		Prueba y Script de Prueba	relacionar Casos de Prueba a los Scripts de Pruebas?	1 = No					
		Relación de defectos a Casos de Prueba	¿Posee mecanismo para relacionar defectos a los Casos/Script de Pruebas?	5 = Si 1 = No	Usuario	5	1	1	5
		Relaciona defectos a su causa	¿Permite determinar la posible causa del defecto?	5 = Si 1 = No	Usuario	5	5	5	5
Planificación de Pruebas (FUN 1.7)	Condiciones a probar en la Aplicación de objeto de prueba (FUN 1.7.1)	Emulación de usuarios concurrentes	¿Emula población/acceso de usuarios concurrentes?	Emulación de usuarios concurrentes	Usuario	5	5	5	5
		Emulación de transacciones concurrentes	¿Emula gran carga de transacciones concurrentes?	5 = Si 1 = No	Usuario	5	5	5	5
		Variación de carga de trabajo durante la prueba	¿Permite variar la carga de trabajo en el tiempo de la ejecución de la prueba?	5 = Si 1 = No	Usuario	5	1	5	5
		Definición de límites/umbrales de finalización de prueba	¿Permite definir límites/umbrales donde la prueba debe concluir?	5 = Si 1 = No	Usuario	5	5	5	5
			¿Permite definir tiempos donde la prueba debe concluir?	5 = Si 1 = No	Usuario	5	5	5	5
		Definición de estrategias de pruebas	¿Permite ingresar retardos en la ejecución de pruebas?	5 = Si 1 = No	Usuario	5	5	5	5
			¿Permite definir hilos de ejecución en la prueba?	5 = Si 1 = No	Usuario	5	5	5	5
			¿Permite definir ráfagas aleatorias en la ejecución de pruebas?	5 = Si 1 = No	Usuario	5	5	5	5
			¿Permite establecer un script para la estrategia de carga de trabajo?	5 = Si 1 = No	Usuario	5	1	1	5
		Cancelación de pruebas al superar umbrales	¿Permite cancelar la ejecución de las pruebas al superar los umbrales establecidos?	5 = Si 1 = No	Usuario	5	5	5	5
Gestión de resultados (FUN 1.8)	Estadísticas de resultados (FUN 1.8.1)	Estadísticas en tiempo real	¿Genera estadísticas/resultados en tiempo real?	5 = Si 1 = No	Usuario	5	5	5	5
		Estadísticas post-mortem	¿Genera estadísticas luego de la ejecución de las pruebas?	5 = Si 1 = No	Usuario	5	5	5	5
		Notifica superación de umbrales al momento	¿Genera alertas, mensajes ante la cercanía de los umbrales permitidos?	5 = Si 1 = No	Usuario	5	5	5	5
		Generación de métricas	¿Permite establecer indicadores o métricas esperadas?	5 = Si 1 = No	Usuario	5	5	5	5
			¿Permite configurar/parametrizar indicadores o métricas esperadas?	5 = Si 1 = No	Usuario	5	1	5	1
	Análisis de resultados (FUN 1.8.2)	Provee de reportes predeterminados para gestionar resultados	¿Tiene reportes preexistentes para emitir resultados de pruebas?	5 = Si 1 = No	Usuario	5	5	5	5
		Permite modificar los reportes predeterminados	¿Permite modificar reportes de resultados preexistentes?	5 = Si 1 = No	Usuario	5	1	5	1
		Permite construir reportes varios de resultados	¿Permite construir reportes de resultados?	5 = Si 1 = No	Usuario	5	1	5	5
		Provee gráficas predeterminadas para gestionar resultados	¿Tiene gráficas preexistentes para emitir resultados de pruebas?	5 = Si 1 = No	Usuario	5	5	5	5
		Permite modificar los gráficos predeterminados	¿Permite modificar gráficos de resultados preexistentes?	5 = Si 1 = No	Usuario	5	5	5	5
		Permite construir gráficos varios de resultados	¿Permite construir gráficos de resultados?	5 = Si 1 = No	Usuario	1	5	5	5
		Análisis de resultados automáticos	¿Proporciona conclusiones en base a los resultados obtenidos?	5 = Si 1 = No	Usuario	1	1	5	1
	Manipulación de resultados (FUN 1.8.3)	Exportar resultados en formatos intercambiables	¿Permite exportar los resultados obtenidos en formato Excel?	5 = Si 1 = No	Usuario	5	5	5	5
			¿Permite exportar los resultados obtenidos en otros formatos como pdf, doc, html, rtf, xml?	5=Completam 4=Casi todos 3= Med. 2= Pocos 1= No	Usuario	4	4	3	3
		Formato de generación de Reportes/entregables son similares entre si	¿Son similares los formatos cómo se generan los reportes/entregables de resultados obtenidos?	5=Completam 4=Casi todos 3= Med. 2= Pocos 1= Ninguno	Usuario	5	5	5	5

		Disponibilidad de los resultados obtenidos (base de datos)	¿Existe acceso permanente a los resultados obtenidos?	5 = Si 1 = No	Usuario	5	5	5	5
Aplicación objeto de prueba (FUN 1.9)	Interacción con la Aplicación objeto de prueba (FUN 1.9.1)	Sincronización de Scripts de Prueba por nuevas versiones en la Aplicación objeto de prueba	¿Permite sincronizar los Scripts de Prueba seleccionados ante nuevas versiones de la Aplicación objeto de prueba?	5 = Si 1 = No	Usuario	1	5	5	1
		Las pruebas se generan al grabar acciones posibles de los usuarios	¿Permite grabar acciones cotidianas del usuario en la Aplicación objeto de prueba?	5 = Si 1 = No	Usuario	1	5	5	5
		Las pruebas se generan especificando manualmente las acciones posible	¿Permite especificar acciones cotidianas del usuario en la Aplicación objeto de prueba?	5 = Si 1 = No	Usuario	5	5	5	5
		Las pruebas se generan mediante documentos/requerimientos	¿Permite obtener por documentos/requerimientos las acciones cotidianas del usuario en la Aplicación objeto de prueba?	5 = Si 1 = No	Usuario	1	1	1	1
		Ante errores/defectos permite continuar la ejecución	¿Permite continuar la ejecución de la prueba ante defectos en la Aplicación objeto de prueba?	5 = Si 1 = No	Usuario	1	5	1	1
		Capacidad de guardar pruebas para continuar después	¿Permite guardar pruebas para continuar después la ejecución?	5 = Si 1 = No	Usuario	5	5	5	5
	Plataformas de construcción de la Aplicación objeto de prueba (FUN 1.9.2)	Aplicación objeto de prueba basada en entorno Web	¿La Aplicación objeto de prueba está construida en entorno Web?	5 = Si 1 = No	Usuario	5	1	1	5
		Aplicación objeto de prueba basada en aplicación Windows	¿La Aplicación objeto de prueba está construida como aplicación Windows?	5 = Si 1 = No	Usuario	1	5	5	1
FUNCIONALIDAD (FUN) - Característica: Precisión (FUN 2)									
		Resultados incompletos	¿Existe falta de precisión en la función debido a resultados incompletos?	5=Ninguna 4=muy pocas 3=muchas 2=Casi todas 1=Todas	Usuario	4	4	4	3
		Resultados incorrectos	¿Existe falta de precisión en la función debido a resultados incorrectos?	5=Ninguna 4=muy pocas 3=muchas 2=Casi todas 1=Todas	Usuario	4	4	4	4
		Resultados emitidos no esperados	¿Existe falta de precisión en la función debido a la emisión de resultados no esperados?	5=Ninguna 4=muy pocas 3=muchas 2=Casi todas 1=Todas	Usuario	5	5	5	5
		Criterios de validación	¿Existen criterios de validación para el producto?	5=Completam def. 4=Casi todo def. 3= Med. Def. 2= Poco def. 1= No están def.	Usuario	5	5	5	5
Aceptación de pruebas realizadas (FUN 2.1)	Criterios de aceptación de pruebas (FUN 2.1.1)	Criterios de prueba superada	¿Permite establecer los criterios/condiciones para aceptar/superar la prueba?	5 = Si 1 = No	Usuario	5	5	5	5
		Criterios de prueba no superada	¿Permite establecer los criterios/condiciones para rechazar/no superar la prueba?	5 = Si 1 = No	Usuario	5	5	5	5
FUNCIONALIDAD (FUN) - Característica: Interoperabilidad (FUN 3)									
		Existencia funcionalidades pertenecientes al producto que son utilizadas por otros sistemas	¿Existen funcionalidades utilizadas por otros sistemas, que pertenecen al producto?	5 = Si tiene 1 = No tiene	Usuario	5	1	1	5
		Intercambio de datos	¿Existe intercambio de datos con otros sistemas?/Producto?	5 = Si tiene 1 = No tiene	Usuario	5	5	1	1
		Consistencia de interfaces	¿Existe consistencia con las interfaces de los otros sistemas?	5 = Si tiene 1 = No tiene	Usuario	5	5	1	1
		Existencia de funcionalidades de otros sistemas que sean necesarias	¿Existen funcionalidades en otros sistemas que son necesarias?	5 = Si tiene 1 = No tiene	Usuario	5	5	1	5

Interacción con otras aplicaciones (FUN 3.1)	Funcionamiento con terceros (FUN 3.1.1)	Componentes/plugins que complementan la Herramienta de Prueba	¿Tiene algún(o) componente/plugin que complementa y/o aumenta sus potencialidades?	5 = Si 1 = No	Usuario	5	5	1	5
		Dependencia de componentes/ plugins para funcionar	¿Necesita de algún(o) componente/plugin para poder funcionar?	5 = Si 1 = No	Usuario	5	1	1	5
	Operaciones con terceros (FUN 3.1.2)	Importar scripts de otras aplicaciones/lenguajes	¿Permite importar scripts desde otras Herramienta de Prueba del mismo fabricante?	5 = Si 1 = No	Usuario	5	5	1	5
			¿Permite importar scripts desde otras Herramienta de Prueba de distinto fabricante?	5 = Si 1 = No	Usuario	1	5	1	1
		Importar datos de prueba desde otros sistemas	¿Permite importar datos para alimentar los Datos de Prueba desde archivos planos?	5 = Si 1 = No	Usuario	5	1	5	5
			¿Permite importar datos para alimentar los Datos de Prueba desde archivos Excel?	5 = Si 1 = No	Usuario	5	5	5	5
			¿Permite importar datos para alimentar los Datos de Prueba desde sentencias sql?	5 = Si 1 = No	Usuario	5	5	5	5
	Almacenamiento en Base de Datos (FUN 3.1.3)	Manejo de Base de Datos propia	¿Posee Base de Datos propia donde almacena información relevante?	5 = Si 1 = No	Usuario	5	1	1	5
		Manejo de Base de Datos no propias	¿Almacena información relevante en Base de Datos no propietaria?	5 = Si 1 = No	Usuario	5	5	5	1
FUNCIONALIDAD (FUN) - Característica: Seguridad (FUN 4)									
Control de acceso (FUN 4.1)		Control de acceso	¿Qué tan completa es la detección de acceso a usuarios del sistema, cuando se trata de obtener acceso al sistema de forma no autorizada?	5=Muy alta 4=Alta 3= Mediana 2= Básica 1= No tiene.	Usuario	4	1	3	1
	Usuarios del sistema (FUN 4.1.1)	Definición de usuarios del sistema	¿Permite configurar usuarios de la Herramienta de Prueba de acuerdo a perfiles?	5 = Si 1 = No	Usuario	5	1	5	1
		Control de acceso a los usuarios	¿Mantiene registro/auditoría del acceso de los usuarios?	5 = Si 1 = No	Usuario	5	1	5	1
		Definición de roles dentro del sistema	¿Permite configurar roles para distinguir entre permisos de usuarios?	5 = Si 1 = No	Usuario	5	1	5	1
			¿Permite implementar roles para distinguir entre permisos de usuarios?	5 = Si 1 = No	Usuario	5	1	5	1
Capacidad de auditoría (FUN 4.2)		Capacidad de Auditoría	¿Qué tan completa es la auditoría? ¿Se evalúa la cantidad de registros de acceso que el sistema registró en la base de datos histórica?	5=Muy alta 4=Alta 3= Mediana 2= Básica 1= No tiene.	Usuario	4	4	5	5
	Registro de auditoría (FUN 4.2.1)	Registro de acceso a la Herramienta de Prueba	¿Permite registrar el acceso a los usuarios registrados?	5 = Si 1 = No	Usuario	5	5	5	1
		Registro de acciones en la Herramienta de Prueba	¿Permite registrar los cambios a configuraciones sobre la Herramienta de Prueba o sobre proyectos existentes?	5 = Si 1 = No	Usuario	5	5	5	5
		Log de auditoría (FUN 4.2.2)	Mantenimiento de log de auditoría	¿Mantiene los logs de Auditoría sobre tiempos definidos?	5 = Si 1 = No	Usuario	5	5	5
USABILIDAD (USA) - Característica: Facilidad de Comprensión (USA 1)									
		Complejidad de las demostraciones funcionales	¿Están completas las demostraciones funcionales implementadas respecto al número de funciones que requieren la demostración de uso?	5=Todas 4=Casi todas 3= Algunas 2= Pocas 1= Ninguna	Usuario	2	2	4	4
		¿Cuál es el tiempo en que un usuario adquiere las destrezas necesarias para usar el sistema?	¿Cuál es el tiempo en que un usuario adquiere las destrezas necesarias para usar el sistema?	5= 6 días o menos 4= 7-14 días 3= 15-30 días 2= 2-5 meses 1= 6 meses o más	Usuario	3	3	4	4
		Nivel de dificultad de la aplicación	Nivel de dificultad de la aplicación	5=Muy fácil 4=Fácil 3=Promedio 2= Difícil 1=Muy difícil	Usuario	4	2	4	5

		Facilidad para ubicar funcionalidades	¿Es fácil ubicar las funcionalidades del sistema?	5= Muy fácilmente 4= Fácilmente 3= Dificultad media 2= Difícil 1= Muy difícil	Usuario	4	4	4	5
USABILIDAD (USA) - Característica: Capacidad de Aprendizaje (USA 2)									
		Calidad en la clasificación de los temas	¿Es fácil utilizar el material de apoyo, por una buena clasificación de los temas	5=Excelente 4=Buena 3= Promedio 2=Debajo del Prom. 1= Inaceptable	Usuario	3	3	5	5
		Compleitud de la especificación de los documentos	¿Están identificados todos los documentos a ser producidos por el proceso o proyecto?	5=Todos 4=Casi todos 3= Algunos 2= Pocos 1= Ninguno	Líder Usuario	4	4	4	4
		Wizard para ciertas funcionalidades	¿Posee ayudas guiadas en la instalación o configuración de ciertas bondades de la Herramienta de Prueba?	5 = Si 1 = No	Usuario	5	1	5	5
		Documentación de la ayuda	¿Ofrece ayuda en línea para las diversas funcionalidades de la Herramienta de Prueba?	5 = Si 1 = No	Usuario	5	1	5	5
			¿Provee manuales de usuario para las diversas funcionalidades de la Herramienta de Prueba?	5 = Si 1 = No	Usuario	5	5	5	5
			¿Provee manual de instalación para la Herramienta de Prueba?	5 = Si 1 = No	Usuario	5	5	5	5
			¿Existen foros de discusión o ayuda en internet?	5 = Si 1 = No	Usuario	1	5	1	5
USABILIDAD (USA) - Característica: Interfaz Gráfica (USA 3)									
		Interfaz personalizable	¿La interfaz es personalizable?	5 = Si 1 = No	Usuario	5	1	5	1
		Localización rápida de opciones	¿Se localizan rápidamente las opciones, consistente la ubicación de utensilios?	5 = Si 1 = No	Usuario	1	1	5	5
		Presencia de metáforas	¿Hay presencia de metáforas?	5 = Si 1 = No	Usuario	5	5	5	5
		Satisfacción del Diseño visual	¿Es bueno el diseño visual de las pantallas?	5=Excelente 4=Buena 3=Promedio 2=Debajo del Prom. 1= Inaceptable	Usuario	4	3	5	4
		Versatilidad de la navegación	¿Es versátil la navegación, entre pantallas?	5=Excelente 4=Buena 3=Promedio 2=Debajo del Prom. 1= Inaceptable	Usuario	4	3	5	4
		Barra de herramientas personalizables	¿La barra de herramientas es personalizable?	5 = Si 1 = No	Usuario	5	1	5	1
		Accesos rápidos a opciones	¿Posee acceso rápido a funciones a través de las teclas?	5 = Si 1 = No	Usuario	5	1	5	1
USABILIDAD (USA) - Característica: Operabilidad (USA 4)									
		Satisfacción con las ayudas disponibles respecto a las requeridas	¿Cómo es la satisfacción con las ayudas disponibles respecto a las requeridas?	5=Excelente 4=Buena 3=Promedio 2=Debajo del Prom. 1= Inaceptable	Usuario	5	5	5	5
		Tasa de disponibilidad de valores por defecto	¿Existen valores por defecto en las pantallas? (N° de valores por defecto en pantalla / N° de entradas en la pantalla)	5=Siempre 4=Casi siempre 3= Algunas veces 2= Pocas veces 1= Nunca	Usuario	4	4	4	3
		Evaluación continua de la Satisfacción del cliente	¿Existe una evaluación continua de la satisfacción del cliente respecto a los servicios de apoyo	5=Siempre 4=Casi siempre 3= Algunas veces	Usuario	4	3	5	5

			y al producto?	2= Pocas veces 1= Nunca					
		Soporte operacional continuo	¿Se provee apoyo para resolver problemas operacionales según requerimientos del cliente?	5=Siempre 4=Casi siempre 3= Algunas veces 2= Pocas veces 1= Nunca	Usuario	3	3	3	3
		Operación apropiada del software en su ambiente	¿Existe una correcta operación del software en su ambiente?	5 = Si 1 = No	Usuario	5	5	5	5
		Asistencia a los clientes en cuanto a operación	¿Existe asistencia a los clientes en cuanto a su operación?	5=Siempre 4=Casi siempre 3= Algunas veces 2= Pocas veces 1= Nunca	Usuario	4	4	4	4
		Lista de posibles valores a ingresar/elegir en campos funcionales	¿Despliega lista de valores tipos para alguna funcionalidad en especial?	5 = Si 1 = No	Usuario	5	5	1	5
		Operación de la Aplicación objeto de prueba con la Herramienta de Prueba	¿Cuán fácil es operar la aplicación objeto de prueba dentro de la herramienta de prueba?	5=Muy fácil 4=Fácil 3=Promedio 2=Difícil 1=Muy difícil	Usuario	5	4	4	4
FIBILIDAD (FIA) - Característica: Tolerancia a Fallas (FIA 2)									
		Anulación de operación incorrecta	¿Qué tan frecuente puede el producto de software <i>evitar</i> fallas, aún si el usuario lo opera incorrectamente?	5=Siempre 4=Casi siempre 3= Algunas veces 2= Pocas veces 1= Nunca	Usuario	4	4	4	4
FIBILIDAD (FIA) - Característica: Recuperación (FIA 3)									
		Capacidad de reiniciar	¿Puede el producto de software recuperarse fácilmente después de una caída?	5=Muy fácilmente 4=Fácilmente 3=Dificultad Media 2= Difícil 1= Muy difícil	Usuario	3	3	4	5
		Velocidad de reinicialización	Velocidad de la reinicialización.	5=Muy rápida 4=Rápida 3= Promedio 2=Lenta 1= Muy lenta	Usuario	3	3	3	3
		Disponibilidad del producto de software	¿Pueden los usuarios trabajar con el producto de software el tiempo necesario?	5=Siempre 4=Casi siempre 3= Algunas veces 2= Pocas veces 1= Nunca	Usuario	5	5	5	5
EFICIENCIA (EFI) - Característica: Comportamiento del Tiempo (EFI 1)									
		Tiempo de respuesta	Tasa del tiempo de respuesta para lo que es diseñado a ser ejecutado	5=Muy rápida 4=Rápida 3= Promedio 2=Lenta 1= Muy lenta	Usuario	4	3	3	4
		Satisfacción con el número de tareas completadas en un tiempo especificado	Satisfacción con el número de tareas que pueden ser completadas durante un tiempo especificado con respecto al número de tareas que son requeridas a ser completadas	5=Muy Alta 4=Alta 3= Media 2= Baja 1= Muy baja	Usuario	4	4	4	4
EFICIENCIA (EFI) - Característica: Utilización de recursos (EFI 2)									
		Utilización del CPU	Tasa de tiempo de CPU utilizado (Tiempo de CPU utilizado / Tiempo de CPU requerido)	5=Muy rápida 4=Rápida 3= Promedio 2=Lenta 1= Muy lenta	Usuario	5	5	5	5
		Utilización de memoria	Tasa del tamaño de memoria utilizada con respecto al tamaño de la memoria que se requiere que sea usada (Memoria utilizada / Requerimientos del uso de memoria)	5=Muy baja 4=Baja 3= Media 2=Alta 1= Muy Alta	Usuario	4	4	5	4

ANEXO VII: Resultados Evaluación de Características generales del modelo propuesto

Evaluación de las categorías del modelo

<i>Categoría (Carac. General)</i>	<i>Pertinencia</i>	<i>Complejidad</i>	<i>Adecuación</i>	<i>Precisión</i>
Funcionalidad (FUN)	1	1	1	1
Usabilidad (USA)	1	1	1	1
Fiabilidad (FIA)	1	1	1	1
Eficiencia (EFI)	1	1	1	1

Evaluación de características del modelo

<i>Categoría</i>	<i>Característica (Carac. específica)</i>	<i>Pertinencia</i>	<i>Complejidad</i>	<i>Adecuación</i>	<i>Precisión</i>
Funcionalidad (FUN)	FUN 1. Ajuste a los propósitos	1	1	1	1
	FUN 2. Precisión	1	1	1	1
	FUN 3. Interoperabilidad	1	1	1	1
	FUN 4. Seguridad	1	1	1	1
Usabilidad (USA)	USA 1. Usabilidad	1	1	1	1
	USA 2. Capacidad de Aprendizaje	1	1	1	1
	USA 3. Interfaz Gráfica	1	1	1	1
	USA 4. Operabilidad	1	1	1	1
Fiabilidad (FIA)	FIA 2. Tolerancia a Fallas	1	1	1	1
	FIA 3. Recuperación	1	1	1	1
Eficiencia (EFI)	EFI 1. Comportamiento del tiempo	1	1	1	1
	EFI 2. Utilización de recursos	1	1	1	1

Evaluación de sub-características del modelo

<i>Categoría</i>	<i>Característica</i>	<i>Sub-Subcaracterística (Carac. específica)</i>	<i>Pertinencia</i>	<i>Complejidad</i>	<i>Adecuación</i>	<i>Precisión</i>
Funcionalidad (FUN)	FUN 1. Ajuste a los propósitos	(FUN 1.1) Niveles de Prueba	1	1	1	1
		(FUN 1.2) Tipos de Prueba	1	1	1	1
		(FUN 1.3) Enfoque de Prueba	1	1	1	1
		(FUN 1.4) Tipo de Licencias de distribución de la Herramienta de Prueba	0	1	1	1
		(FUN 1.5) Administración de la Herramienta de Prueba	1	1	0	1
		(FUN 1.6) Entregables de Prueba	1	1	1	0
		(FUN 1.7) Planificación de Prueba	1	1	1	1
		(FUN 1.8) Gestión de Resultados	1	1	1	1
		(FUN 1.9) Aplicación objeto de Prueba	1	1	1	1
	FUN 2. Precisión	(FUN 2.1) Aceptación pruebas realizadas	1	1	1	1
	FUN 3. Interoperabilidad	(FUN 3.1) Interacción con otras aplicaciones	1	1	1	1
	FUN 4. Seguridad	(FUN 4.1) Control de acceso	0	1	1	1
		(FUN 4.2) Capacidad de auditoría	1	1	1	1

Evaluación de sub-sub-características del modelo

<i>Categoría</i>	<i>Característica</i>	<i>Sub-Subcaracterística</i>	<i>Sub-sub-Subcaracterística (Característica específica)</i>	<i>Pertinencia</i>	<i>Complejidad</i>	<i>Adecuación</i>	<i>Precisión</i>
Funcionalidad (FUN)	FUN 1. Ajuste a los propósitos	(FUN 1.1) Niveles de Prueba	Pruebas de sistema (FUN 1.1.1)	1	1	1	1
		(FUN 1.2) Tipos de Prueba	Pruebas de Desempeño (FUN1.2.1)	1	1	1	1
			Pruebas de Carga/Estrés (FUN1.2.2)	1	1	1	1
		(FUN 1.3) Enfoque de Prueba	Pruebas de Caja Negra (FUN 1.3.1)	1	1	1	1
		(FUN 1.4) Tipo de Licencias de	Open source (FUN 1.4.1)	1	1	1	1

		distribución de la Herramienta de Prueba	Comercial (FUN 1.4.2)	1	1	1	1
		(FUN 1.5) Administración de la Herramienta de Prueba	Lenguaje de Programación de la Herramienta de Prueba (FUN 1.5.1)	1	0	1	1
		(FUN 1.6) Entregables de Prueba	Casos de Prueba (FUN 1.6.1)	1	1	1	1
			Datos de Prueba (FUN 1.6.2)	1	1	1	1
			Script de Pruebas (FUN 1.6.3)	1	1	1	1
			Suite de Pruebas (FUN 1.6.4)	1	0	1	1
			Salidas de pruebas (FUN 1.6.5)	1	1	1	1
			Relación entre entregables de Pruebas (FUN 1.6.6)	1	1	1	0
		(FUN 1.7) Planificación de Prueba	Condiciones a probar en la Aplicación objeto de prueba (FUN 1.7.2)	1	1	1	1
		(FUN 1.8) Gestión de Resultados	Estadísticas de resultados (FUN 1.8.1)	1	1	1	1
			Análisis de resultados (FUN 1.8.2)	1	1	1	1
			Manipulación de resultados (FUN 1.8.3)	0	1	1	1
		(FUN 1.9) Aplicación objeto de Prueba	Interacción con la Aplicación objeto de prueba (FUN 1.9.1)	1	1	1	1
			Plataformas de construcción de la Aplicación objeto de prueba (FUN 1.9.2)	1	1	1	1
	FUN 2. Precisión	(FUN 2.1) Aceptación de pruebas realizadas	Criterios de aceptación de pruebas (FUN 2.1.1)	1	1	1	1
	FUN 3. Interoperabilidad	(FUN 3.1) Interacción con otras aplicaciones	Funcionamiento con terceros (FUN 3.1.1)	1	0	1	1
			Operaciones con terceros (FUN 3.1.2)	0	1	1	1
			Almacenamiento en Base de Datos (FUN 3.1.3)	1	1	1	1
	FUN 4. Seguridad	(FUN 4.1) Control de acceso	Usuarios del sistema (FUN 4.1.1)	1	0	1	1
		(FUN 4.2) Capacidad de auditoría	Registro de auditoría (FUN 4.2.1)	1	1	1	1

ANEXO VIII: Resultados Evaluación de Métricas del modelo propuesto

Característica	Sub- Característica	Sub-sub- Característica	Métrica (Característica Específica)	Pertin.	Factibi- lidad	Profun- didad	Escala
CATEGORIA – F U N C I O N A L I D A D							
FUN 1. Ajuste a los propósitos			Cumplimiento de las necesidades funcionales	1	1	1	1
			Definición de las necesidades de adquisición, desarrollo o mejora	1	1	1	1
			Definición de la estrategia de adquisición o desarrollo de nuevos sistemas	1	1	1	1
			Definición del criterio de aceptación	1	1	1	1
			Registro de los resultados de la prueba	1	1	1	1
	(FUN 1.1) Niveles de Prueba	Pruebas de Sistema (FUN 1.1.1)	Ejecución de pruebas de a nivel de sistema	1	1	1	1
	(FUN 1.2) Tipos de Prueba	Pruebas de Desempeño (FUN 1.2.1)	Ejecución del Tipo de Pruebas de Desempeño	1	1	1	1
		Pruebas de Carga/Estrés (FUN 1.2.2)	Ejecución del Tipo de Pruebas de Carga/Estrés	1	1	1	1
	(FUN 1.3) Enfoque de Prueba	Pruebas de Caja Negra (FUN 1.3.1)	Ejecución de Pruebas con enfoque de Caja Negra	1	1	1	1
			Exploración en el código de la Aplicación objeto de prueba	1	1	1	1
	(FUN 1.4) Tipo de Licencias de distribución de la Herramienta de Prueba	Open source (FUN 1.4.1)	Distribución de la Herramienta de Prueba Opensource	1	1	1	1
		Comercial (FUN 1.4.1)	Distribución de la Herramienta de Prueba es comercial	1	1	1	1
	(FUN 1.5) Administración de la Herramienta de Prueba	Lenguaje de Programación de la Herramienta de Prueba (FUN 1.5.1)	Código único/proceso unificado para multiplataformas	1	1	1	1
			Código Propietario (propone comandos, funciones propietarias)	1	1	1	1
			Necesidad de amplios conocimientos en programación	1	1	1	1
	(FUN 1.6) Entregables de Prueba	Casos de Prueba (FUN 1.6.1)	Generación automática de Casos de Prueba	1	1	1	1
			Construcción de Casos de Prueba paso a paso	1	1	1	1
			Identificación de los requerimientos de prueba	1	1	1	1
			Reutilización de Casos de Prueba entre proyectos	0	1	1	1
			Documentación de Pre-condiciones	1	1	1	1
			Documentación de Post-condiciones	1	1	1	1
			Documentación de Post-condiciones	1	1	1	1
		Datos de Prueba (FUN 1.6.2)	Generación manual de Datos de Prueba desde Casos de Prueba	1	1	1	1
			Generación automática de Datos de Prueba desde datos reales	1	1	1	1
			Generación automática de Datos de Prueba desde sentencias SQL	1	1	1	1
			Generación automática de Datos de Prueba de forma aleatoria	1	1	1	1
			Reutilización de Datos de Prueba entre proyectos	0	1	1	1
		Script de	Generación automática de Scripts	1	1	1	1

		Pruebas (FUN 1.6.3)	de Prueba desde Casos de Prueba + Datos de Prueba				
			Construcción de Scripts de Prueba paso a paso	1	1	1	1
			Modificar Scripts de Prueba luego de generarlos	1	1	1	1
			Modificar Scripts de Prueba en línea/ejecución	1	1	1	1
			Reutilización de Scripts de Prueba entre proyectos	0	1	1	1
		Suite de Pruebas (FUN 1.6.4)	Organizar/secuenciar conjunto de pruebas	1	1	1	1
			Programación de Scripts de Prueba en el tiempo	1	1	1	1
		Salidas de pruebas (FUN 1.6.5)	Generación de evidencias de Pruebas	1	1	1	1
			Gestión de defectos	1	1	1	1
			Log de pruebas/histórico de ejecuciones	1	1	1	1
		Relación entre entregables de Pruebas (FUN 1.6.6)	Relación entre requerimientos de prueba y Casos de Prueba	1	1	1	1
			Relación entre Casos de Prueba y Script de Prueba	1	1	1	1
			Relación de defectos a Casos de Prueba	1	1	1	1
			Relaciona defectos a su causa	1	1	1	1
	(FUN 1.7) Planificación de Prueba	Condiciones a probar en la Aplicación objeto de prueba (FUN 1.7.2)	Emulación de usuarios concurrentes	1	1	1	1
			Emulación de transacciones concurrentes	1	1	1	1
			Variación de carga de trabajo durante la prueba	1	1	1	1
			Definición de límites/umbrales de finalización de prueba	1	1	1	1
			Definición de estrategias de pruebas	1	1	0	1
			Cancelación de pruebas al superar umbrales	1	1	1	1
	(FUN 1.8) Gestión de Resultados	Estadísticas de resultados (FUN 1.8.1)	Estadísticas en tiempo real	1	1	1	1
			Estadísticas post-mortem	1	1	1	1
			Notifica superación de umbrales al momento	1	1	0	1
			Generación de métricas	1	1	1	1
		Análisis de resultados (FUN 1.8.2)	Provee de reportes predeterminados para gestionar resultados	1	1	1	1
			Permite modificar los reportes predeterminados	1	1	1	1
			Permite construir reportes varios de resultados	1	1	1	1
			Provee gráficas predeterminados para gestionar resultados	1	1	1	1
			Permite modificar los gráficos predeterminados	1	1	1	1
			Permite construir gráficos varios de resultados	1	1	1	1
			Análisis de resultados automáticos	1	1	1	1
		Manipulación de resultados (FUN 1.8.3)	Exportar resultados en formatos intercambiables	1	1	1	1
			Formato de generación de Reportes/entregables son similares entre sí	0	1	1	1
			Disponibilidad de los resultados obtenidos (base de datos)	1	1	1	1
	(FUN 1.9) Aplicación objeto de Prueba	Interacción con la Aplicación objeto de prueba (FUN 1.9.1)	Sincronización de Scripts de Prueba por nuevas versiones en la Aplicación objeto de prueba	1	0	1	1
			Las pruebas se generan al grabar acciones posibles de los usuarios	1	1	1	1
			Las pruebas se generan especificando manualmente las	1	1	1	1

			acciones posible				
			Las pruebas se generan mediante documentos/requerimientos	1	1	1	1
			Ante errores/defectos permite continuar la ejecución	1	1	1	1
			Capacidad de guardar pruebas para continuar después	1	1	1	1
		Plataformas de construcción de la Aplicación objeto de prueba (FUN 1.9.2)	Aplicación objeto de prueba basada en entorno Web	1	1	1	1
			Aplicación objeto de prueba basada en aplicación Windows	1	1	1	1
FUN 2. Precisión			Resultados incompletos	1	1	1	1
			Resultados incorrectos	1	1	1	1
			Resultados emitidos no esperados	1	1	1	1
			Criterios de validación	1	1	1	1
	Aceptación de pruebas realizadas (FUN 2.1)	Criterios de aceptación de pruebas (FUN 2.1.1)	Criterios de prueba superada	1	1	1	1
			Criterios de prueba no superada	1	1	1	1
FUN 3. Interoperabilidad			Existencia de funcionalidades pertenecientes al producto que son utilizadas por otros sistemas	1	1	1	1
			Intercambio de datos	1	1	1	1
			Consistencia de interfaces	1	1	1	1
			Existencia de funcionalidades de otros sistemas que sean necesarias	1	1	1	1
	(FUN 3.1) Interacción con otras aplicaciones	Funcionamiento con terceros (FUN 3.1.1)	Componentes/plugins que complementan la Herramienta de Prueba	1	1	1	1
			Dependencia de componentes/plugins para funcionar	1	1	1	1
		Operaciones con terceros (FUN 3.1.2)	Importar scripts de otras aplicaciones/lenguajes	1	1	1	1
			Importar datos de prueba desde otros sistemas	1	1	1	1
		Almacenamiento en Base de Datos (FUN 3.1.3)	Manejo de Base de Datos propias	1	1	1	1
			Manejo de Base de Datos no propias	1	1	1	1
FUN 4. Seguridad	(FUN 4.1) Control de acceso		Control de acceso	1	1	0	1
		Usuarios del sistema (FUN 4.1.1)	Definición de usuarios del sistema	1	1	1	1
			Control de acceso a los usuarios	1	1	1	1
			Definición de roles dentro del sistema	1	1	1	1
	(FUN 4.2) Capacidad de auditoría		Capacidad de Auditoría	1	1	1	1
		Registro de auditoría (FUN 4.2.1)	Registro de acceso a la Herramienta de Prueba	1	1	1	1
			Registro de acciones en la Herramienta de Prueba	1	1	1	1
		Log de auditoría (FUN 4.2.2)	Mantenimiento de log de auditoría	1	1	1	1
Característica	Sub- Característica	Sub-sub- Característica	Métrica (Característica Específica)	Pertin.	Factibilidad	Profundidad	Escala
CATEGORIA – U S A B I L I D A D							
Facilidad de comprensión (USA 1)			Compleitud de las demostraciones funcionales	1	1	1	1
			¿Cuál es el tiempo en que un usuario adquiere las destrezas necesarias para usar el sistema?	1	1	1	1
			Nivel de dificultad de la aplicación	1	1	1	1
			Facilidad para ubicar funcionalidades	1	1	1	1
Capacidad de Aprendizaje (USA 2)			Calidad en la clasificación de los temas	1	1	1	1
			Compleitud de la especificación de los documentos	1	1	1	1

			Wizard para ciertas funcionalidades	1	1	1	1
			Documentación de la ayuda	1	1	1	1
Interfaz Gráfica (USA 3)			Interfaz personalizable	1	1	1	1
			Localización rápida de opciones	1	1	1	1
			Presencia de metáforas	1	1	1	1
			Satisfacción del Diseño visual	1	1	1	1
			Versatilidad de la navegación	1	1	1	1
			Barra de herramientas personalizables	1	1	1	1
			Accesos rápidos a opciones	1	1	1	1
Operabilidad (USA 4)			Satisfacción con las ayudas disponibles respecto a las requeridas	1	1	1	1
			Tasa de disponibilidad de valores por defecto	1	1	1	1
			Evaluación continua de la Satisfacción del cliente	1	1	1	1
			Soporte operacional continuo	1	1	1	1
			Operación apropiada del software en su ambiente	1	1	1	1
			Asistencia a los clientes en cuanto a operación	1	1	1	1
			Lista de posibles valores a ingresar/elegir en campos funcionales	1	1	1	1
Característica	Sub- Característica	Sub-sub- Característica	Métrica (Característica Específica)	Pertin.	Factibi- lidad	Profun- didad	Escala
CATEGORIA – F I A B I L I D A D							
Tolerancia a Fallas (FIA 2)			Anulación de operación incorrecta	1	1	1	1
Recuperación (FIA 3)			Capacidad de reiniciar	1	1	1	1
			Velocidad de reinicialización	1	1	1	1
			Disponibilidad del producto de software	1	1	1	1
Característica	Sub- Característica	Sub-sub- Característica	Métrica (Característica Específica)	Pertin.	Factibi- lidad	Profun- didad	Escala
CATEGORIA – E F I C I E N C I A							
Comportamiento del tiempo (EFI 1)			Tiempo de respuesta	1	1	1	1
			Satisfacción con el número de tareas completadas en un tiempo especificado	1	1	1	1
Utilización de recursos (EFI 2)			Utilización del CPU	1	1	1	1
			Utilización de memoria	1	1	1	1

REFERENCIAS BIBLIOGRÁFICAS

- American Society for Quality (2007) *Glosary*, USA. Recuperado en Agosto 03, 2007 de <http://www.asq.org/glossary/>
- Asamblea Nacional de la Republica Bolivariana de Venezuela (2002) *Ley orgánica del sistema Venezolano para la Calidad*. Venezuela.
- Automated Testing Institute - ATI (2010) *Testing, techniques & tools*, USA. Recuperado en Marzo 05, 2010 de <http://www.automatedtestinginstitute.com/home/>
- Basili, V.R., Caldiera, C., Rombach H.D. (1994) *Goal Questions Metric Paradigm. Encyclopedia of Software Engineering*. USA: John Wiley & Sons.
- Baskerville, Richard (1999) *Organizational and Social Perspectives on Information Technology* (1er ed.) USA: Springer.
- Berkhout, Michiel et. al. (2001) Office of Government Commerce. *Best Practice Best Service – Service Support*. USA.
- Braude, Eric J. (2003) *Ingeniería de Software: Una perspectiva orientada a objetos* (1er ed.). México: Alfaomega Grupo Editor.
- Bruegge, Bernd; Dutoit, Allen H. (2002) *Ingeniería de Software Orientado a Objetos* (2da ed.). México: Prentice Hall.
- CMMI-DEV,V1.2 (2006) CMMI® for Development, Version 1.2. USA: Software Engineering Institute – Carnegie Mellon.
- Carrasquel, Raúl (2008) *Los canales de Distribución de las PYMEs en Venezuela: un estudio exploratorio*. Maestría en Administración de Empresas. USB. Venezuela.
- Carretero G., María J. (2006) *Libro web sobre ética aplicada a la informática*. Universidad de Málaga. España.
- Castejón, Adolfo (2005) *Tendencias del sector de las tecnologías de información y comunicaciones (TIC) en Venezuela*. Venezuela Tecnológica: Presente y Futuro (1er ed.). Venezuela: CONAPRI.
- Checkland, Peter (2003) *Pensamiento de sistemas-Practica de sistemas* (1er ed.). Mexico: Limusa.
- Crosby, Philip B. (1980) *Quality is free: The Art of Making Quality certain* (1era. ed.) USA: Signet Shakespeare.
- De Luca, L., Mendoza, L., Pérez, M., Rojas, T. (2001) *Indicators for the selection of software quality management tools*. CLEI 2001, Mérida, Venezuela.
- Díaz, A., Ovalles, M., Mendoza, L., Rojas, T. (2001) *Aplicación de un modelo de calidad para la evaluación de herramientas CASE adaptado al estándar*

internacional ISO/IEC 14102. 1ras Jornadas Iberoamericanas de Ingenierías de Software e Ingeniería del Conocimiento, Buenos Aires, Argentina.

Di Paula, George; Parada, Dakar; Pérez, María A.; Mendoza, Luis (2007) Proyecto: *Agilidad y disciplina del proceso de desarrollo de software para las Pequeñas y Medianas Empresas (PYMEs) y las Cooperativas en Latinoamérica. Caso: Venezuela*. Jornadas Iberoamericanas de Ingeniería del Software e Ingeniería del Conocimiento.

Dustin, Elfriede (2003) *Effective software testing: 50 specific ways to improve your testing*. (1era. ed.) USA: Addison-Wesley.

Erard, Pedro (1999). *¿Cómo vencer los obstáculos al desarrollo de la pyme?* El aporte de Fundes. *Debates Iesa*, Caracas Venezuela 1999. 5, 9-19.

Granados, Gladys; Murúa, Mario (2008) *Estado de La Industria Venezolana de Software: Informe Técnico 2007*. Centro de Excelencia en Ingeniería del Software (CEISoft) de la Corporación Parque Tecnológico de Mérida

Grimán, Anna C.; Pérez, María A.; Mendoza, Luis E. (2003) *Estrategia de Pruebas para Software OO que garantiza Requerimientos No Funcionales*. III Workshop de Ingeniería de Software, Chillán, Chile.

IEEE Std. 610.12-1990 *Standard Glossary of Software Engineering Terminology*.

IEEE Std. 730-2002 *Standard for Software Quality Assurance Plans*.

International Standard (1995) *Iso/Iec 12207 Software Life Cycle Processes*. USA: Federal Aviation Administration.

ITIL® Glossary v01, (2006) best-management-practice.com. *Glosario de Términos ITIL®, Definiciones y Acrónimos*. USA. Recuperado en Noviembre 24, 2007 de http://www.best-management-practice.com/gempdf/ITILV3_Glossary_Spanish_v3.1.24.pdf

Iovine V., Doménico (2009) Implementación de mejoras de una herramienta en Software libre para las pruebas de software. Venezuela. UNEFA.

James, Paúl (2000) *Gestión de la calidad total: un texto introductorio* (1era. ed.). España. Pearson Educación.

Jiménez, Carlos (2006) *Las Pymes y las TIC: Hasta qué punto las Pymes han incorporado el desarrollo de las tecnologías de la información y comunicación*. Presentado en el evento: “Perspectivas y Nuevas Realidades de la Pequeña y Mediana Empresa”, Venamcham, Venezuela.

Juran, Joseph M., Godfrey, A. Blanton (1998) *Juran's Quality Handbook* (5ta. ed.) USA: McGraw-Hill Professional.

Kitchenham, Ba (1996) *DESMET: A methodology for evaluating software engineering methods and tools*. USA: Keele University Technical Report.

Kruchten, Philippe (2000) *The Rational Unified Process: An introduction* (2da. ed.). USA: Addison-Wesley.

Lacy, Shirley; Macfarlane, Ivor (2007) best-management-practice.com. *Service Transition*. USA. Recuperado en Noviembre 24, 2007 de <http://www.best-management-practice.com/bookstore.asp?FO=1229358&DI=591438>

Llorens F., Juan (2005) *Gerencia de Proyectos de Tecnología de información* (1ra ed.). Venezuela: Los libros de El Nacional.

Office of Government Commerce – OGC (2007), best-management-practice.com *Continual Service Improvement*. USA. Recuperado en Noviembre 24, 2007 de <http://www.best-management-practice.com/Online-Bookshop/IT-Service-Management-ITIL/ITIL-Version-3/Continual-Service-Improvement/PDF/>

Leffingwell, Dean; Widrig, Don (2006) *Managing Software Requirements, a Use Case Approach*. (2da. Ed.) USA: Addison-Wesley, Pearson Education.

Matos, Giancarlo; Anglés, Rafael y Ortiz, Lourdes (2004) *Modelo de apoyo en la selección de la metodología para un desarrollo de software, basado en la gerencia del conocimiento y soportado en tecnología de Información*. LIV Convención Anual ASOVAC. Venezuela.

Méndez, Edumilis; Pérez, María A.; Mendoza, Luis E. (2007) *Aplicación de un Método para Especificar Casos de Prueba de Software en la Administración Pública*. Actas de Talleres de Ingeniería del Software y Bases de Datos, Vol. 1, No. 4, 2007

Méndez, E., Pérez, M., Grimán, A. & Mendoza, L. (2005). *¿Cómo se relacionan la calidad sistémica y la productividad en el proceso de desarrollo de software?*. Iberoamericana de Sistemas, Cibernética e Informática, 2 (2), 1-8.

Mendoza, L., Grimán, A., Pérez, M. (2004) *Modelo integrado para soportar la selección de herramientas CASE*. IV Workshop de Ingeniería del Software - XII Jornadas Chilenas de Computación, Arica, Chile.

Mendoza, Luis E.; Pérez, Maria A.; Grimán, Anna (2005). *Prototipo de Modelo Sistémico de Calidad (MOSCA) del Software*. Mexico. Computación y Sistemas Vol. 8 Núm. 3, pp. 196-217.

Methodius (2005) *Métodos y modelos de desarrollo de software para empresas venezolanas*. Proyecto de Investigación Nro. 2005000165 del FONACIT. Venezuela. Recuperado en Mayo 15, 2010 de <http://www.methodius.org.ve/>

Montilva C., Jonas A. (2007). *La Industria de Software en Venezuela: Una caracterización de su recurso*. Presentación del I Encuentro Nacional sobre Tecnologías de Información e Ingeniería de Software EVETIS'07. Venezuela.

Mora B., Freddy (2003) *El marco jurídico regulatorio de las pequeñas y medianas empresas (PYMEs) en Venezuela*. Venezuela: VISIÓN GERENCIAL.

Omaña M., Mayuri S. (2009) *Modelo de calidad basado en características para la selección de un sistema de gestión de aprendizaje (LMS-Learning Managment System)*. Maestría en Sistemas de Información. UCAB. Venezuela.

Perry, William E. (2006) *Effective Methods for Software Testing* (3era ed.) USA: Wiley Publishing, Inc

Piattini V., Mario G.; García R., Félix O.; Caballero M., Ismael (2007) *Calidad De Sistemas Informáticos* (1era ed.) México. Alfaomega, Ra-Ma.

Pressman, Roger S. (2005) *Ingeniería del software: un enfoque práctico* (6ta ed.). México. McGraw-Hill.

Qualitatis.org (2006) *Antítesis (Ágiles)* USA. Recuperado en Agosto 03, 2007 de <http://www.qualitatis.org/>

Rational Unified Process® Version 2003.06.15 © Copyright IBM Corp. 1987-2005

Real Academia Española (2001) *Diccionario de la lengua española* (22da. ed.). España. Disponible en www.rae.es

Reyes, Esteban (2005) *La industria del software en Venezuela: Pasado, presente y futuro*. Venezuela Tecnológica: Presente y Futuro (1er ed.). Venezuela: CONAPRI.

Rivero, D.; Montilva, J; Granados, G.; Barrios, J.; Besembel, I.; Sandia, B. (2007) *La Industria de Software en Venezuela: Una Caracterización de su Recurso Humano*. Encuentro Venezolano sobre Tecnologías de la Información e Ingeniería del Software (EVETIS'07).

Sánchez, Amarilis; Romero, Engelbert; Zúñiga, Roy. (2001) *Proyecto Mejorando la Competitividad del Cluster de Software en Venezuela: Diagnóstico y Benchmarking*. Instituto Centroamericano de Administración de Empresas INCAE. Recuperado en Mayo 08, 2008 de http://www.cid.harvard.edu/archive/andes/documents/workingpapers/microfoundations/it_venezuela/diagnostico_software_venezuela.pdf

Schulmeyer, g. Gordon; McManus, James I. (1993) *Total Quality Management for software* (1ra ed.) USA: Van Nostrand Reinhold. Kenneth S. Mendis, artículo: "Evaluación de la calidad del software".

Software Engineering Institute – Carnegie Mellon (2007) *SEI Areas of work: Management*. USA. Recuperado en Julio 24, 2007 de <http://www.sei.cmu.edu/>

Software QA and Testing Resource Center (2007) *Software QA and Testing Frequently-Asked-Questions, Part 1 by Hower Rick*. USA. Recuperado en Agosto 03, 2007 de http://www.softwareqatest.com/qatfaq1.html#FAQ1_1

Sommerville, Ian (2006) *Software Engineering* (8ta ed.). USA: Addison-Wesley.

Susman, G., y Evered, R. (1978) *An assessment of the scientific merit of action research*. Administrative Science Quarterly.

SWEBOK (2004) *Guide to the Software Engineering Body of Knowledge* (IEEE – 2004 Version). USA: A project of the IEEE Computer Society Professional Practices Committee.

Taylor, Sharon (2007) Axios System. *ITIL V3 - The Future is Here*. USA. Recuperado en en Noviembre 24, 2007 de [http://www.best-management-practice.com/gempdf/ITILV3 The Future is Here Sharon Taylor.pdf](http://www.best-management-practice.com/gempdf/ITILV3%20The%20Future%20is%20Here%20Sharon%20Taylor.pdf)

The Standish Group International (2003) *CHAOS Chronicles Version III*. USA. Recuperado en Abril 19, 2008 de <http://www.standishgroup.com/chaos/toc.php>

Utting, Mark; Legeard, Bruno (2006) *Practical Model-Based Testing: A Tools Approach* (1ra ed.) USA: Morgan-Kaufmann.

Unhelkar, Bhuvan (2003) *Process Quality Assurance for UML-Based Projects* (1ra ed.). USA: Addison Wesley.

Wells, Don (2000) [extremeprogramming.org](http://www.extremeprogramming.org). *Extreme Programming: A gentle introduction*. USA. Recuperado en Agosto 13, 2007 de <http://www.extremeprogramming.org/>

Yáber, Guillermo y Valarino, Elizabeth (2006) *Clasificación, organización y gestión de la investigación en los postgrados de administración y gerencia*. Venezuela: Grupo de Creatividad y Productividad. Universidad Simón Bolívar.



UNIVERSIDAD CATÓLICA ANDRÉS BELLO
VICERRECTORADO ACADÉMICO
ESTUDIOS DE POSTGRADO
ÁREA DE INGENIERÍA
Postgrado en Sistemas de Información

Trabajo de Grado de Maestría

**MODELO DE REFERENCIA PARA LA SELECCIÓN DE HERRAMIENTAS DE PRUEBAS
COMO SOPORTE AL PROCESO DE DESARROLLO DE SOFTWARE EN PYMES
VENEZOLANAS**

presentado por
Mendoza Méndez, María Gabriela;
para optar al título de
Magíster en Sistemas de Información

Tutor
Pérez de Ovalles, María Angélica

Caracas, Julio 2010

UNIVERSIDAD CATÓLICA ANDRÉS BELLO
VICERRECTORADO ACADÉMICO
ESTUDIOS DE POSTGRADO
ÁREA DE INGENIERÍA
Postgrado en Sistemas de Información

**MODELO DE REFERENCIA PARA LA SELECCIÓN DE HERRAMIENTAS DE PRUEBAS
COMO SOPORTE AL PROCESO DE DESARROLLO DE SOFTWARE EN PYMES
VENEZOLANAS**

Autor: María Gabriela Mendoza Méndez

Tutor: María Angélica Pérez de Ovalles

Fecha: Julio 2010

RESUMEN

El desarrollo de software debe estar enfocado en la imperiosa realidad de garantizar calidad tanto en todo su proceso como en sus resultados (producto) para así traducirse en una estrategia empresarial y lograr a través de éste, las ventajas competitivas en un entorno globalizado. Estar consciente de esta realidad en las empresas desarrolladoras de software, constituidas en Venezuela principalmente por PYMES, se traducirá en dar formalidad a la disciplina de pruebas como apoyo a este proceso de desarrollo de software. Sin embargo, esta formalidad esperada en la disciplina de pruebas y el uso de herramientas automatizadas es escasa. El objetivo de la presente investigación, es proponer un modelo de referencia dirigido a PYMES venezolanas que permita apoyar la selección de herramientas de prueba adaptada a las características de sus proyectos y del contexto organizacional, para asegurar la calidad de sus productos. Los resultados de esta investigación se lograron siguiendo cada uno de los pasos y/o actividades enmarcados en el Framework metodológico del LISI y se cuantifican en: a) Modelo conceptual que abarca las principales definiciones de las pruebas de software, resultado de un extenso análisis de la bibliografía y las experiencias con herramientas de pruebas existentes en el mercado; b) Árbol de clasificación de herramientas de pruebas automatizadas con su correspondiente instanciación (herramientas para el nivel de pruebas del sistema del tipo no funcionales: Desempeño y carga, bajo un enfoque de Caja Negra; c) Modelo de referencia propuesto para seleccionar herramientas de pruebas que apoyen el proceso de desarrollo de software, producto de aportes al modelo de calidad sistémica MOSCA para el producto; d) Guía para aplicar el modelo propuesto a herramientas de prueba existentes en el mercado tecnológico, previa selección de acuerdo al árbol de clasificación. El método de evaluación arrojado por DESMET fue “Análisis de características por Estudio de Caso”, por lo que el modelo se aplicó en cuatro estudios de casos resultando sólo un caso con nivel de calidad Avanzado. Luego de la experiencia obtenida, el modelo fue evaluado iniciando desde sus características generales hasta sus métricas, proponiendo algunas modificaciones.

PALABRAS CLAVE

Desarrollo de software, calidad del software, disciplina de pruebas, herramientas de prueba, Pequeñas y Medianas Empresas (PYMES).

DEDICATORIA

A mi amado Dios,
gracias por permitirme culminar con éxito otro logro más de mi vida!

Todo lo que soy te lo debo a ti, quien
*“...por su amor, gracia y misericordia, le da recibimiento
a su hija pródiga, con anillo, ropas nuevas, fiesta y alegría.”*

(San Lucas 15:11-32)

AGRADECIMIENTOS

Agradezco a mi familia: padres, hermanas, sobrinos por ser personas fundamentales en mi vida, por reconocer mis logros, apoyar mis decisiones y siempre darme palabras de aliento y ánimo para seguir adelante.

Gracias a mi esposo bello por su amor, paciencia y apoyo en la culminación de esta meta... tu paz me apacigua.

Un agradecimiento muy especial a mi tutora, Prof. María Angélica Ovalles sin olvidar a mis profesores co-tutores del LISI (Edumilis Méndez y Luis E. Mendoza), por su dedicación, profesionalismo, paciencia y por enseñarme a no conformarme.

A mis amigas, amigos y demás familiares por sus palabras de apoyo y expresiones de ayuda incondicional.

A mis jefes laborales, por su paciencia y apoyo en el transitar por este camino.

Y finalmente agradezco a mi segunda casa estudiantil, la UCAB, a mis profesores y compañeros de estudios que promovieron un espíritu de excelencia académica.

INDICE GENERAL

RESUMEN	1
DEDICATORIA	2
AGRADECIMIENTOS	3
INDICE GENERAL	4
INDICE DE TABLAS	6
INDICE DE FIGURAS	7
INTRODUCCIÓN	8
CAPÍTULO I: EL PROBLEMA DE INVESTIGACIÓN	11
1.1 PLANTEAMIENTO DEL PROBLEMA	11
1.2 INTERROGANTES DE LA INVESTIGACIÓN	12
1.3 OBJETIVOS DE LA INVESTIGACIÓN	13
1.4 JUSTIFICACIÓN E IMPORTANCIA DE LA INVESTIGACIÓN	13
1.5 ALCANCES Y LIMITACIONES	15
CAPÍTULO II: MARCO TEORICO	17
2.1 ANTECEDENTES DE INVESTIGACIÓN	17
2.2 MARCO ORGANIZACIONAL	20
2.3 BASES TEÓRICAS	27
I. Calidad del Software	28
II. Aseguramiento de la Calidad del Software	33
III. Pruebas del Software	40
IV. Disciplina de Pruebas	43
V. Modelos de Calidad	52
VI. Herramientas de Pruebas	55
2.4 BASES ÉTICAS Y LEGALES	57
CAPÍTULO III: MARCO METODOLÓGICO	59
3.1 TIPO DE INVESTIGACIÓN	59
3.2 DISEÑO DE LA INVESTIGACIÓN	60
<i>Enfoque basado en objetivos, preguntas y métricas – GQM</i>	66
3.3 TÉCNICAS E INSTRUMENTOS DE RECOLECCIÓN DE DATOS	67
CAPÍTULO IV: DESARROLLO DE LA PROPUESTA	68
4.1 MODELO CONCEPTUAL	68
4.2 CLASIFICACIÓN PROPUESTA PARA LAS HERRAMIENTAS DE PRUEBA	73
4.3 ASPECTOS TOMADOS DE MOSCA PARA LA INSTANCIACIÓN	76
4.4 DESARROLLO DE LA PROPUESTA (ASPECTOS AGREGADOS A MOSCA PARA LA INSTANCIACIÓN)	79
CAPÍTULO V: EVALUACION DE LA PROPUESTA	89
5.1 OBJETIVOS DE LA EVALUACIÓN	89
5.2 METODOLOGÍA SEGUIDA	89
5.3 PROCESO DE EVALUACIÓN	89

<i>a) Análisis de Contexto</i>	<i>89</i>
<i>b) Aplicación del método DESMET.....</i>	<i>91</i>
<i>Método de Análisis de Características por Estudio de Caso</i>	<i>93</i>
<i>c) Evaluación del Modelo de referencia basado en características para la selección de herramientas de prueba</i>	<i>94</i>
CAPÍTULO VI: ANÁLISIS E INTERPRETACIÓN DE RESULTADOS	101
6.1 EVALUACIÓN DEL MODELO EN LOS ESTUDIOS DE CASO	101
6.2 EVALUACIÓN DE LAS CARACTERÍSTICAS GENERALES.....	110
6.2.1 Evaluación de las categorías del modelo	110
6.2.2 Evaluación de las características del modelo	111
6.3 EVALUACIÓN DE LAS MÉTRICAS DEL MODELO.....	113
CAPÍTULO VII: CONCLUSIONES Y RECOMENDACIONES	101
7.1 CONCLUSIONES	116
7.2 RECOMENDACIONES.....	118
ANEXO I: Modelo de Calidad MOSCA	120
ANEXO II: Aspectos de la categoría FUNCIONALIDAD tomadas de MOSCA	121
ANEXO III: Aspectos de la categoría USABILIDAD tomadas de MOSCA.....	122
ANEXO IV: Aspectos de la categoría FIABILIDAD y EFICIENCIA tomadas de MOSCA	123
ANEXO V: Descripción de algunas métricas aportadas por el autor a MOSCA en el contexto del modelo propuesto	124
ANEXO VI: Evaluación detallada de las Herramientas de prueba seleccionadas de acuerdo al Modelo propuesto.....	126
ANEXO VII: Resultados Evaluación de Características generales del modelo propuesto	133
ANEXO VIII: Resultados Evaluación de Métricas del modelo propuesto	135
REFERENCIAS BIBLIOGRÁFICAS	139

INDICE DE TABLAS

Tabla 1- <i>Tabla Comparativa entre defensores del Aseguramiento de la Calidad</i> (Fuente: Elaboración propia).....	39
Tabla 2- <i>Modelo de Calidad según ISO 9126</i> . Fuente: Elaboración propia	54
Tabla 3 - <i>Nivel de calidad del producto con respecto a las categorías satisfechas para el producto</i> . FUENTE: Mendoza et al. (2005).	55
Tabla 4 - <i>Fases del Método Investigación Acción</i> . Fuente: Elaboración Propia.	63
Tabla 5 - <i>Sub-características, sub-sub-características y métricas agregadas en la característica “Ajuste a los propósitos” de la categoría Funcionalidad de MOSCA</i> . Fuente: Elaboración Propia.....	85
Tabla 6- <i>Sub-características, sub-sub-características y métricas agregadas en la característica “Precisión” de la categoría Funcionalidad de MOSCA</i> . Fuente: Elaboración Propia.....	86
Tabla 7 - <i>Sub-característica, sub-sub-características y métricas agregadas en la característica “Interoperabilidad” de la categoría Funcionalidad de MOSCA</i> . Fuente: Elaboración Propia.....	86
Tabla 8 - <i>Sub-característica, sub-sub-características y métricas agregadas en la característica “Seguridad” de la categoría Funcionalidad de MOSCA</i> . Fuente: Elaboración Propia.	87
Tabla 9 - <i>Métricas agregadas en la categoría Usabilidad de MOSCA</i> .Fuente: Elaboración Propia.....	87
Tabla 10 - <i>Condiciones favorables presentes y no presentes en cada uno de los métodos propuestos por DESMET</i> . Fuente: Elaboración Propia.....	93
Tabla 11- <i>Características Generales a evaluar para el modelo propuesto</i> . Fuente: Sosa (2005).....	97
Tabla 12 – <i>Características de las Métricas a evaluar el modelo propuesto</i> . Fuente: Sosa (2005).....	97
Tabla 13 - <i>Herramientas de prueba del mercado seleccionadas para cada estudio de caso</i> . Fuente: Elaboración Propia.	99
Tabla 14 - <i>Evaluación del modelo propuesto en el Proyecto Piloto</i> . Fuente: Elaboración Propia.....	102
Tabla 15 – <i>Nivel de calidad del producto propuesto respecto a categorías satisfechas para el producto adaptado al modelo propuesto</i> . Fuente: Adaptado de Mendoza et al. (2005).....	108
Tabla 16 - <i>Nivel de calidad de los estudios de caso respecto a categorías satisfechas para el producto adaptado al modelo propuesto</i> . Fuente: Elaboración propia.	109

INDICE DE FIGURAS

Figura 1 – <i>Definición de pequeñas y medianas empresas en Venezuela según decreto</i> .	21
Figura 2- <i>Tamaño promedio de las empresas de la Industria Nacional del Software</i>	23
Figura 3- <i>Distribución geográfica de empresas desarrolladoras de software en Venezuela</i>	23
Figura 4 - <i>Tipos de aplicaciones que produce la Industria Nacional de Software</i> .	24
Figura 5 - <i>Necesidades de actualización en procesos de desarrollo de software</i>	25
Figura 6 - <i>Necesidades de capacitación en procesos de gestión de proyectos de software</i>	25
Figura 7 - <i>Utilización de algunas herramientas de apoyo durante el desarrollo de software</i>	26
Figura 8- <i>Procesos y Componentes para el Aseguramiento de la Calidad</i>	36
Figura 9 - <i>Modelo de Proceso de MSF para el desarrollo de aplicaciones</i> , Fuente: (Microsoft, 2003).....	47
Figura 10- <i>Fases del ciclo de vida del desarrollo de software</i>	49
Figura 11 - <i>Vista general del Proceso Unificado de Rational</i>	50
Figura 12 - <i>Framework Metodológico para la de investigación</i>	62
Figura 13- <i>Modelo Conceptual asociado a los Casos de Prueba</i>	69
Figura 14 - <i>Modelo Conceptual asociado a los Tipos, Niveles y Técnicas de Prueba</i>	70
Figura 15- <i>Modelo Conceptual asociado a las Herramientas de Pruebas y el Desarrollo de Software</i>	71
Figura 16 - <i>Modelo Conceptual asociado al Aseguramiento de la Calidad</i>	71
Figura 17- <i>Modelo Conceptual propuesto integrado</i>	72
Figura 18 - <i>Árbol de clasificación de herramientas de prueba automatizadas</i>	74
Figura 19 - <i>Árbol instanciado de ajuste a los propósitos para herramientas de prueba elegidas por el investigador</i>	76
Figura 20 - <i>Instanciación de MOSCA para el modelo de referencia para la selección de herramientas de Prueba</i>	81
Figura 21 - <i>Resultados evaluación Características de la Categoría “Funcionalidad” en Estudio de casos</i>	102
Figura 22 - <i>Resultados evaluación Características de la Categoría “Usabilidad” en Estudio de casos</i>	104
Figura 23 - <i>Resultados evaluación Características de la Categoría “Fiabilidad” en Estudio de casos</i>	105
Figura 24 - <i>Resultados evaluación Características de la Categoría “Eficiencia” en Estudio de Casos</i>	105
Figura 25 – <i>Resultados de la evaluación de los estudios de caso en cada categoría del modelo propuesto</i>	106
Figura 26- <i>Resultados de la evaluación de “Categorías” del modelo propuesto</i>	110
Figura 27 - <i>Resultados de la evaluación de “Características” de la categoría Funcionalidad del modelo propuesto</i>	111
Figura 28 - <i>Resultados de la evaluación de “Características” de la categoría Usabilidad del modelo propuesto</i>	112
Figura 29 - <i>Resultados de la evaluación de “Características” de la categoría Fiabilidad y Eficiencia del modelo propuesto</i>	112
Figura 30 - <i>Resultados de la evaluación de “Métricas” por Categoría del modelo propuesto</i> . .	113

INTRODUCCIÓN

El desarrollo de software en empresas especializadas para tal fin debe estar enfocado en la imperiosa realidad de garantizar calidad tanto en todo su proceso como en sus resultados (producto) para así traducirse en una estrategia empresarial y lograr a través de éste, las ventajas competitivas en un entorno altamente globalizado.

Las empresas desarrolladoras de software, para garantizar calidad en sus productos deben realizar pruebas del software construido. Esto se logra a través de hacer énfasis en la disciplina de Pruebas dentro del proceso de desarrollo de software, práctica que generalmente se traduce en invertir entre el 30% y 50% del costo total del software en pruebas. Sin embargo el tema de las pruebas no está bien formalizado ni justificado dentro del proyecto de software, por lo que invertir esfuerzo y costos en esta disciplina, es casi nula o cuando está presente, se estiman en dimensiones de tiempo/costo no apropiadas ni ajustadas a la realidad.

Bajo esta premisa, las pruebas tienen una gran importancia y representan un gran volumen de actividades ya que se sugiere que en un proyecto de desarrollo de software al menos el 75% de las pruebas sean automatizadas y el resto manual; esto evidencia que la estrategia de automatizar pruebas es una decisión primordial. La automatización de las pruebas se logra a través del uso de herramientas de pruebas automatizadas.

Existen numerosas herramientas de pruebas en el mercado tecnológico con propósitos tan diversos y en algunos casos, tan distantes, que tener criterios válidos para su selección se vuelve una actividad compleja. En todo caso, el primer paso es definir el propósito para lo cual se requiere y el alcance que las mismas tendrán en el proceso de desarrollo de software.

Uno de los principales problemas a los que se enfrenta la calidad del software es el siguiente: ¿Es realmente posible encontrar un conjunto de propiedades en un software que sea una indicación de su calidad?. Para dar respuesta a esta pregunta aparecen los Modelos de Calidad o Modelo de referencia, que no son más que las directrices o puntos de referencia para imitarlo o reproducirlo.

Por lo antes expuesto, en este trabajo de grado de maestría se propone un Modelo de referencia para la selección de herramientas de prueba que permitan apoyar el proceso de desarrollo de software específicamente en la disciplina de pruebas para PYMES desarrolladoras de software en Venezuela, convirtiéndose en referencia obligada ante un proceso de selección de herramientas de prueba automatizada, el cual fue ampliamente evaluado.

Este modelo surgió como producto de cumplir con cada una de las actividades y/o fases enmarcadas dentro del Framework metodológico propuesto por el Laboratorio de Investigación en Sistemas de Información (LISI-USB) dentro del un tiempo planificado y finito en donde se realizó un análisis exhaustivo del tema, producto de revisiones constantes preliminares y mesas de trabajo del alumno, con en el trabajo mancomunado de sus asesores técnicos y metodológicos, para una vez definido teóricamente, llevar este conocimiento a la práctica para crear el modelo de referencia de selección de herramientas de pruebas.

Cabe destacar que por la amplitud de los tipos, niveles y enfoques que tienen las herramientas de prueba existentes en el mercado, se hizo necesario trabajar con solo una muestra de herramientas de prueba obtenida de la instanciación del árbol de clasificación de herramientas propuesto por considerarse uno de los más importantes tipos de pruebas que debe realizar una empresa desarrolladora de software.

Sin embargo, los objetivos logrados se traducen en importantes beneficios para PYMES desarrolladoras de software pues por una parte, el modelo propuesto hace un extenso recorrido por los atributos mandatorios que debe tener una herramienta de prueba garantizando que la selección de herramientas sea suficientemente robusta y precisa en sus resultados siempre y cuando se cumplan con las condiciones de aplicación del modelo, y por la otra, proporciona una guía completa para la aplicación del modelo con efectividad comprobada que también puede usarse como guía para construir una herramienta de prueba adaptada a las necesidades de la empresa que lo desarrolle.

Finalmente, este proyecto se estructura en base a los siguientes seis capítulos:

- **El problema de investigación**, donde se introduce y plantea el tema en estudio, su importancia y justificación de la investigación. Igualmente, se definen los

objetivos, listando los objetivos específicos para la realización del tema en estudio y que sustentan el objetivo general de la investigación que también es descrito.

- El siguiente capítulo, **Marco Teórico** se traen a colación investigaciones previas enmarcadas dentro del área y del tema en estudio que podrán ser utilizadas como antecedentes para dar base a la presente investigación. Se presentan exhaustivamente las bases teóricas por una parte, y por la otra, las bases éticas y legales que sustentan la propuesta. De igual forma se plantea el marco organizacional que explica hacia cual sector productivo se dirige la presente investigación, PYMEs desarrolladoras de Software en Venezuela.
- **Marco Metodológico**, es el capítulo donde se plasma el marco de trabajo que rige la investigación y se explica cada una de sus fases, así como el tipo de investigación y las técnicas e instrumentos utilizados a lo largo de la investigación.
- En el capítulo IV **Desarrollo de la propuesta** se exponen los aportes al conocimiento de los sistemas de información que ha logrado el autor iniciando desde el modelo conceptual lleno de definiciones que giran en torno a las Pruebas, el árbol de clasificación de pruebas hasta llegar al modelo propiamente dicho (con los aspectos tomados de MOSCA y los que fueron propuestos).
- La **Evaluación de la propuesta** describe paso a paso cómo se definió la estrategia de evaluación del modelo dentro del contexto de aplicación y a través del método de evaluación DESMET para constituirse en el insumo del siguiente capítulo VI **Análisis e Interpretación de resultados**, donde se muestra primero la evaluación del modelo en los cuatro estudios de caso seleccionados y luego la evaluación del modelo en sí, desde sus características generales hasta las métricas que la conforman. Los resultados de la evaluación en forma detallada se presentan en la sección de Anexos.
- El último capítulo **Conclusiones y Recomendaciones** muestran los logros obtenidos en esta investigación en forma resumida y las sugerencias del autor para futuros trabajos de investigación, respectivamente.

CAPÍTULO I: EL PROBLEMA DE INVESTIGACIÓN

1.1 Planteamiento del Problema

Dentro del marco de los Sistemas de Información (SI) son múltiples las aplicaciones que pueden desarrollarse para las empresas en sus diversos ámbitos de acción. Actualmente las Tecnologías de Información (TI) impactan cualquier ámbito del desarrollo tecnológico y científico del ser humano; su aplicación a numerosas áreas cotidianas se ha vuelto indispensable, aportando beneficios al poder administrar, entre otras cosas el conocimiento y su gestión, que actualmente es considerado parte del capital más importante de las organizaciones. Dentro de estas organizaciones se incluye la industria del software en Venezuela, donde las pequeñas y medianas empresas (PYMES) juegan un rol importante para el desarrollo del país (Reyes, 2005) y cuyos principales factores de producción son el talento humano, las capacidades de producción, la calidad y la innovación.

Estudiar la garantía de la calidad en el desarrollo de software, tanto en su proceso de desarrollo como en el producto resultante, se ha constituido en un tópico de importancia para la Ingeniería de Software. La calidad esperada en el producto puede determinarse a través de una *disciplina* definida por RUP (2005) como una colección de actividades correspondidas que a su vez se relacionan con un “motivo de preocupación importante” dentro del proyecto total donde estén enmarcadas.

Algunas investigaciones como Rivero et al. (2007), Methodius (2005), Sánchez et al. (2001) reflejan de manera general una realidad latente de que las empresas desarrolladoras de software en Venezuela, actualmente adolecen de estrategias, métodos y modelos que afiancen sus procesos y las hagan competitivas dentro de su entorno nacional con miras al ámbito internacional. De forma específica, las escasas investigaciones nacionales enfocadas en la disciplina de pruebas dentro del proceso de desarrollo de software nacional hacen aún más difícil que dichas empresas tengan una estrategia que les permita realizar una evaluación de las diversas herramientas que presenta el mercado en cuanto a pruebas se refiere; razón por la cual, la decisión de optar por una herramienta que satisfaga las necesidades del proyecto de software, se tornan con frecuencia imprecisas. En la mayoría de los casos, esta elección se ve restringida a escoger herramientas que no son las más acordes a la naturaleza del

proyecto, y en el peor de los casos, se niega el uso de estas herramientas imposibilitando su cooperación en el proceso de desarrollo de software. Esta carencia en el uso de herramientas de pruebas igualmente refleja la poca importancia y la informalidad que a lo largo del proyecto de software se le da a quien proporciona las directrices para obtener calidad de software, la Disciplina de Pruebas (pruebas de software o sistema); que debe ser vista como la unión de roles, actividades, artefactos y su orden interno más que cómo un simple requisito para la aprobación final del producto.

Involucrarse en el tema de las *pruebas del software* con todas sus etapas es una realidad que debe estar presente en cualquier empresa que desee adentrarse en el mundo del desarrollo del software y, por supuesto, la calidad. En la actualidad existen variadas herramientas de automatización y planificación del esfuerzo de pruebas que facilitan el desarrollo de ésta disciplina. Dichas herramientas son diversas y adaptadas a ciertas necesidades, por lo que se torna difícil decidir a ‘ciencia cierta’ cuál de las mismas pueden aplicarse en una u otra situación. Y es que la dificultad de la elección de la mejor herramienta está determinada por diversos puntos de atención, como lo son: la naturaleza del proyecto, la tecnología utilizada en la construcción del software, las necesidades de pruebas, los potenciales clientes a los que está dirigido y hasta la empresa que desarrolla el proyecto (estructura física y financiera) entre otros aspectos relevantes.

En la próxima sección se indican las interrogantes de investigación que rigen esta investigación.

1.2 Interrogantes de la investigación

Ante el problema planteado, se enuncian las siguientes preguntas que rigen la investigación y clarifican el propósito de la misma.

- ¿Cuáles son las características relevantes de las herramientas de pruebas que apoyan y soportan la disciplina de pruebas en todo el proceso de desarrollo de software para las PYMEs?
- ¿Disponen las PYMEs venezolanas desarrolladoras de software de métodos para la evaluación de herramientas de pruebas de software que de apoyo a la disciplina de pruebas y aseguren la calidad de sus productos?

- ¿Son aplicables y adaptables los resultados de la investigación a las necesidades de las PYMEs desarrolladoras de software en Venezuela?

Para dar respuesta a estas interrogantes y en base a las mismas, se formulan los objetivos que rigen esta investigación en la próxima sección.

1.3 Objetivos de la investigación

Objetivo general:

Desarrollar un Modelo de referencia dirigido a PYMES venezolanas desarrolladoras de software que permita apoyar la selección de herramientas de pruebas adaptado a las características de sus proyectos y del contexto organizacional.

Objetivos específicos:

- Formular un modelo conceptual que soporte el trabajo de investigación, partiendo de puntos clave como Calidad de software, Aseguramiento de la Calidad del Software y Disciplina de Pruebas.
- Proponer un modelo de referencia que especifique las características que debe poseer la herramienta que soporte la disciplina de pruebas enmarcadas dentro de las características de las PYMES Venezolanas.
- Aplicar el modelo adaptado a un estudio de caso de PYMEs.
- Evaluar el modelo de referencia.

1.4 Justificación e importancia de la investigación

El proyecto planteado realiza un estudio de todo lo concerniente a la disciplina de Pruebas del software en el ciclo de desarrollo, dando importancia al aseguramiento de la Calidad del software como promotor y garante de calidad en el proceso de desarrollo del software y a la selección de herramientas de pruebas que soporten esta disciplina.

El modelo de referencia propuesto para la selección de las herramientas de pruebas de software que ofrece el mercado tecnológico, está dirigido a las organizaciones desarrolladoras de software y prioritariamente hacia las PYMEs venezolanas, las cuales están obteniendo popularidad y aceptación dentro de este tiempo de cambio y transformación que transita nuestro país (Sánchez et al., 2001). Dicho

modelo de referencia será validado y documentado de acuerdo a las normas establecidas en la Ingeniería del Software, con dos propósitos especiales. Por una parte, crear conciencia en las PYMEs de la realización de tan importante actividad en el desarrollo de sus sistemas/software y, que a la vez, apalanquen el desarrollo de sus actividades. Por otra parte, proporcionar a las PYMEs orientaciones necesarias para seleccionar las herramientas que más le favorezcan, en pro de fortalecer el desarrollo de software nacional con los máximos niveles de calidad, creando las directrices para buscar lograr una competitividad entre empresas locales, siempre en vías de incursionar en el ámbito internacional. He aquí la importancia de la investigación.

La Ingeniería del Software abarca actividades macro de suma importancia en el desarrollo del mismo, dentro de la que se encuentran las pruebas del software (Pressman, 2005) la cual vela, entre otros aspectos, porque exista la calidad esperada en el producto resultante, esto es, el fiel cumplimiento de los requerimientos acordados por el cliente. El Standish Group (2003), revela cifras importantes sobre la deficiencia en la entrega de los proyectos de tecnologías de información para el año 2002 en Estados Unidos, por lo que no es de extrañarse que estos mismos problemas se afronten en nuestro país. Prueba de ello lo refleja Methodius (2005), al afirmar que la mayoría de las empresas e instituciones públicas venezolanas desarrollan software usando procesos ad-hoc o métodos y técnicas inapropiadas u obsoletas, muchas de las cuales llevan utilizándose más de 25 años.

Las escasas estrategias e investigaciones que se han realizado en Venezuela, dirigidas principalmente a PYMES desarrolladoras de software en Venezuela, hacen que las mismas se caractericen por no tener formalidad en su proceso de desarrollo de software y por ende, en la práctica de la disciplina de pruebas, lo cual las hace poco competitivas en el entorno donde se desenvuelven. Un indicador importante de esta crítica situación es que, hasta la presente fecha, sólo dos empresas venezolanas han sido certificadas con el modelo CMM; el modelo de evaluación de la madurez de los procesos de desarrollo de software de mayor aplicación en el mundo (Methodius, 2005). Otro aspecto de relevancia, lo manifiesta el estudio realizado por Rivero et al. (2007) donde se demuestra la gran necesidad de actualización en el tema de pruebas de software que evidencia el capital humano en la industria del software, aspecto y campo de acción que se debe reforzar en el recurso humano, más aún cuando según Pressman

(2005), las empresas que desarrollan software deberían invertir en la disciplina de pruebas entre el 30% y 50% del costo total del software. Generalmente, de acuerdo a la experiencia laboral de la autora en empresas cuya actividad principal es el desarrollo de software o en empresas que desarrollan sistemas para apalancar sus procesos, el tema de las pruebas no está bien formalizado ni justificado dentro del proyecto de software, por lo que invertir esfuerzo y costos en esta disciplina, es casi nula o cuando está presente, se estiman en dimensiones de tiempo/costo no apropiadas ni ajustadas a la realidad; esta mínima inversión, muchas veces no se extiende al uso de herramientas de pruebas que hagan más plausible el impulso de esta disciplina.

Dedicar esfuerzos en este tema, son bien justificados para el desarrollo de la industria del software en nuestro país. Elaborar nuevos y efectivos métodos para apoyar el desarrollo de software, que sean adaptados a las características de las empresas desarrolladoras de software del país, es una necesidad evidente.

1.5 Alcances y limitaciones

El presente proyecto de investigación está enmarcado dentro de una investigación en progreso auspiciada por el FONACIT bajo el proyecto “METHODIUS” Nro. 205000165, cuyo objetivo principal es “la elaboración de métodos y modelos de desarrollo de software adaptados al contexto nacional y fundamentados en la Ingeniería de Software para que contribuyan a mejorar los procesos productivos de empresas venezolanas que desarrollan software para uso interno o comercial; y elevar la calidad de sus productos.” (Methodius, 2005). Bajo esta premisa, la investigación se orienta a buscar y aportar soluciones al problema de desarrollo de software que tienen las empresas en Venezuela y que por consiguiente, puedan ser aplicadas de forma eficaz en este contexto, específicamente en las PYMEs; esto es, el enfoque es dar soluciones específicas, y no crear métodos universales o de propósito general.

Siguiendo la línea anterior, los resultados de la investigación estarán limitados por este alcance, y se enfocará en desarrollar un modelo de referencia, un modelo bien sustentado y evaluado con directrices específicas para ser implementado o traducido en una aplicación de software en futuras investigaciones.

Se entiende que los usuarios del modelo propuesto por el investigador poseen amplios conocimientos en el tema de las Pruebas de software, dado que deben poder clasificar satisfactoriamente una herramienta de prueba de acuerdo al tipo de prueba que se requiera ejecutar, es decir, comprender el tipo de prueba que puede ejecutar las herramientas que serán objeto de selección/evaluación del modelo.

La búsqueda de información en cuanto a PYMEs se refiere, por razones de tiempo y costo, se limitó a pocas empresas venezolanas desarrolladoras de software distribuidas geográficamente en el área capital venezolana, apoyado con el Laboratorio de Investigación en Sistemas de Información LISI de la Universidad Simón Bolívar (LISI – USB).

CAPÍTULO II: MARCO TEORICO

En el siguiente capítulo se presentan las bases teóricas que sustentan la presente investigación, haciendo un recorrido por conceptos de interés que van desde los antecedentes de investigaciones relacionadas en los últimos años, pasando por definiciones relativos hasta llegar a las bases legales y éticas que soportan este trabajo.

2.1 Antecedentes de investigación

En el ámbito del desarrollo de software, son numerosos los aportes que los investigadores han realizado, siendo éste un tema bastante avanzado. Contextualizado en el ámbito de las PYMEs, Di Paula et al. (2007) hacen un interesante aporte sobre dar formalidad al proceso de desarrollo de software, construyendo una red de decisión para precisar el balance entre agilidad y disciplina dentro de dicho proceso en las PYMEs y Cooperativas de Latinoamérica, tomando como caso de estudio Venezuela. En dicho trabajo, se plasman interesantes contribuciones sobre la realidad de las empresas desarrolladoras de software de Venezuela (constituidas principalmente por PYMEs), al identificar los principales factores que inciden en esta dupla, agilidad y disciplina los cuales permiten una mejor adaptación y mantenimiento de la esencia como empresas pequeñas y/o medianas, para mejorar el desempeño de sus actividades. Dar soluciones apropiadas a las PYMEs Venezolanas, con el propósito de proveer beneficios trascendentales en la “manera de hacer las cosas”, a través de un modelo de referencia para las mismas, es uno de los propósitos de la presente investigación en progreso.

Reyes (2005) desglosa las posibilidades de un mayor desarrollo de la industria del Software en Venezuela, destacando la importancia de ésta industria en el desarrollo del país, y como a lo largo de la historia ha fluctuado esta tendencia. En su artículo, proporciona las directrices de lo que son hoy y pueden llegar a ser en el futuro, las PYMEs desarrolladoras de software. Analizar este abanico de posibilidades que tiene Venezuela para apoyar el crecimiento del desarrollo de este tipo de empresas, inspira el dedicar tiempo y esfuerzo en investigaciones que permitan garantizar calidad de los servicios que prestan, con miras a intensificar las exportaciones de software desde el país, que no se detiene desde hace unas décadas.

Esta calidad esperada en el desarrollo de software es, en gran parte lograda a través de la Disciplina de Pruebas en el proceso de desarrollo de software. Sin embargo, en el ámbito de la Disciplina de Pruebas se percibe una ausencia de investigaciones que se enfoquen en el uso de herramientas de pruebas para que las empresas puedan soportarse en las mismas y apoyar así, sus procesos informáticos.

Algunas acertadas aproximaciones respecto a garantizar aspectos de calidad dentro del proceso de desarrollo de software y su correspondiente medición, han podido lograrse a través del Laboratorio de Investigación en Sistemas de Información LISI de la Universidad Simón Bolívar (LISI - USB), cuyos aportes especiales a la presente investigación son reseñados a continuación.

Méndez et al. (2005) da respuesta a la proposición ¿Cómo se relacionan la calidad sistémica y la productividad en el proceso de desarrollo de software?, al describir las características de la Calidad y la Productividad aplicado al proceso de desarrollo de Sistemas de Software utilizando dos herramientas: El Modelo Sistémico de Calidad (MOSCA) y el Manual de FIM-Productividad (adaptado a empresas de software) con muy buenos resultados. Esto hace entender la necesidad de garantizar el aseguramiento de la calidad en todo el proceso de desarrollo.

En Mendoza et al. (2005), se presentan las bases teóricas, evolución, aplicaciones y demás información relevante sobre el Modelo Sistémico de Calidad MOSCA. Observar cómo un desarrollo de software puede ser medido bajo estrictos criterios de calidad, fundamenta la investigación para recalcar la importancia de tomar en cuenta el concepto de calidad en las actividades de las empresas de software. MOSCA ya forma parte de una de las herramientas de calidad que pueden usarse para apoyar la Administración de la Calidad del Software en sus tres actividades: Aseguramiento de la Calidad, Planeación de la Calidad y Control de la Calidad del proceso de desarrollo de software propiamente dicho.

Aún para productos ya desarrollados, la calidad del software debe estar presente. Esta idea es presentada por Ortega et al. (2007), al crear una herramienta de soporte específicamente orientado a la calidad de estrategias de reutilización de componentes, denominada R-TOOL. Los altos niveles de certificación que puedan encontrarse en los

componentes desarrollados generan confianza y estimulan la Reutilización de los mismos. Este tema constituye otra aplicabilidad e importancia de determinar calidad en los productos de software.

Por su parte, Iovine (2009) en su trabajo analiza 11 herramientas de pruebas del mercado mediante el modelo MOSCA y partiendo de los resultados, propone e implementa mejoras en una de estas herramientas. Las mejoras dieron paso a un módulo que verifica la trazabilidad desde los requerimientos a los casos de prueba planteados en el ámbito de software libre. Este se constituye como un gran aporte a la presente investigación, sentando las bases para reutilizar la metodología en la evaluación de herramientas de pruebas. Este módulo ha sido mejorado por Ricardo (2009) quien plantea e implementa mejoras sustanciales al producto desarrollado.

En cuanto a la selección de herramientas, partiendo de una metodología de software, Matos et al. (2004) muestran un modelo de apoyo para seleccionar metodologías de desarrollo de software, a través del cual podrían determinarse desde etapas tempranas, la mejor opción de estrategia de pruebas para dichos software.

Resulta interesante analizar el modelo propuesto para tratar de traer dichos conocimientos al área de disciplina de pruebas, haciendo una correlación de conceptos de cualquier proceso de desarrollo de software. Por otra parte, enfocados en la selección de herramientas específicamente las CASE, se han adelantado importantes trabajos, basados en áreas específicas como el Modelo integrado para soportar la selección de herramientas CASE (Mendoza et al., 2004) y la Aplicación de un modelo de calidad para la evaluación de herramientas CASE adaptado al estándar internacional ISO/IEC 14102 (Díaz et al., 2001). Para la selección de herramientas de soporte de gestión de la calidad, De Luca et al. (2001) presenta una guía para seleccionar herramientas en cuanto a la gestión de calidad de software se refiere, basada en indicadores identificados y bien detallados en su trabajo. Otro trabajo interesante es el propuesto por Omaña (2009) quien plantea un modelo de calidad para la selección de sistemas de gestión de aprendizaje mediante el análisis y descripción de los componentes de calidad del modelo MOSCA y que finalmente fue aplicado en PDVSA con resultados satisfactorios.

Un completo y práctico modelo basado en pruebas es presentado en forma bien explícita por Utting y Legeard (2006), el cual es está dirigido a la automatización del proceso de pruebas. En él identifican criterios para la selección de pruebas que se traducen en importantes iniciativas para el establecimiento del modelo de calidad producto de esta investigación, pues son adaptables según el propósito de la prueba que se quiera establecer, sin olvidar el tema de las herramientas de pruebas que apoyen el modelo. La forma en que adapta el modelo que ellos proponen con algunos métodos ágiles y el proceso unificado (UP, por sus siglas en ingles), sienta bases para la adaptabilidad que persigue el modelo fruto de esta investigación.

Perry (2006) plantea siete pasos que soportan el proceso de pruebas tradicional y lo adapta a los métodos ágiles, dando vital importancia a las herramientas de pruebas que puedan soportar dicho proceso. De igual forma, el autor clasifica las herramientas de pruebas en un vasto rango, que servirá de base para la selección y categorización de las herramientas de pruebas que existen en el mercado para dar forma al modelo propuesto. Otro aporte importante es reconocer que la selección de las herramientas apropiadas en pruebas, constituye un aspecto importante en el proceso de pruebas, identificando cuatro pasos que deben estar presentes al seleccionar las herramientas de pruebas cónsonas con el proyecto.

2.2 Marco Organizacional

PYMEs es el acrónimo de Pequeñas Y Medianas Empresas y representa un concepto muy difundido en el mundo ya que dentro de la economía mundial, este tipo de empresas tiene un rol muy significativo y con un auge cada vez mayor. Sin embargo dar una definición única es una tarea complicada, pues los criterios o factores utilizados en la definición de PYME, son muy variados, de acuerdo a cada país. En el caso de Venezuela, su definición viene dada en base a dos criterios: Número de trabajadores y ventas Anuales (Carrasquel, 2007), la cual es esquematizada, para facilitar su comprensión, en la Figura 1.

En el marco de la Legislación Venezolana¹, PYME es definida como:
“ARTICULO 3. A los efectos del presente Decreto Ley se entiende por:
1. Pequeña Industria, la unidad de explotación económica realizada por una persona

¹ Ley para la promoción y desarrollo de la pequeña y mediana industria, Decreto N° 1547, 09 de noviembre de 2001

jurídica que efectúe actividades de producción de bienes industriales y de servicios conexos, comprendidas en los siguientes parámetros:

1.1 Planta de trabajadores con un promedio anual no menor de once (11) trabajadores, ni mayor a cincuenta (50) trabajadores.

1.2 Ventas anuales entre nueve mil una (9.001) Unidades Tributarias y cien mil (100.000) Unidades Tributarias.

2. Mediana Industria, la unidad de explotación económica, realizada por una persona jurídica que efectúe actividades de producción de bienes industriales y de servicios conexos, comprendida en los siguientes parámetros:

2.1 Planta de trabajadores con un promedio anual no menor a cincuenta y un (51) trabajadores, ni mayor a cien (100) trabajadores.

2.2 Ventas anuales entre cien mil una (100.001) Unidades Tributarias y doscientos cincuenta mil (250.000) Unidades Tributarias.”

Las PYMEs son importantes generadoras de empleo, de crecimiento y, por ende como conglomerado, demandan una importante cantidad de servicios y productos. No obstante, su potencial de crecimiento en la economía venezolana es su principal activo (Jiménez, 2006). Importantes organizaciones en el país (gubernamentales, privadas, de financiamiento, entre otras), apoyan este tipo de empresas por su impacto en la economía nacional. Resulta cada día más evidente que existe una conciencia mayor en torno al papel que desempeñan las PYMEs, tanto en el ámbito económico como en la generación de empleo.

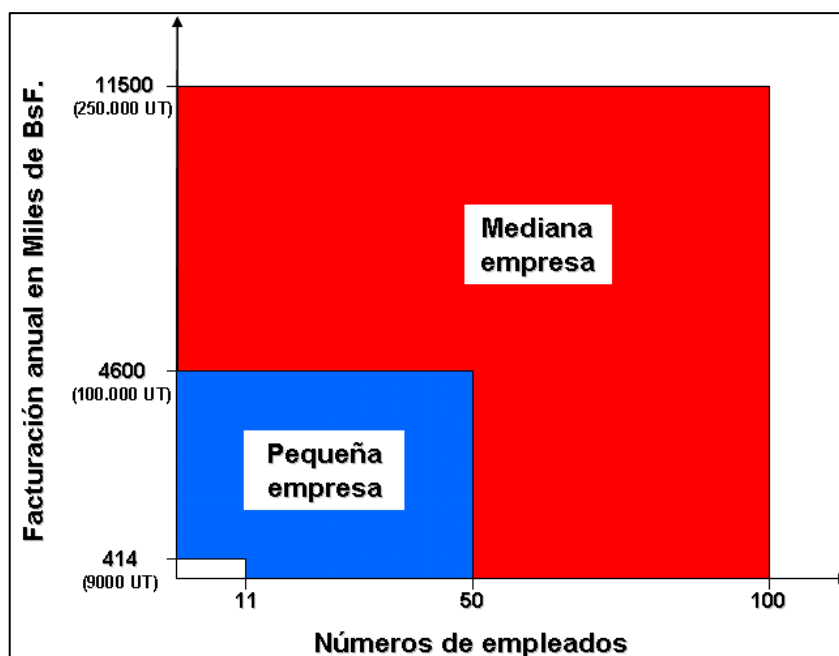


Figura 1 – Definición de pequeñas y medianas empresas en Venezuela según decreto Nro. 1547. Valor de la unidad tributaria vigente para Enero 2008 de BsF. 46,00. Fuente: Carrasquel (2008).

En el entorno de las tecnologías de información, las PYMEs en Venezuela tienen gran auge. Las empresas desarrolladoras de software dedicadas expresamente al desarrollo, mantenimiento y comercialización de productos de software para terceros, constituye el sector industrial en el que se enfoca la presente investigación que está muy cercana a ésta realidad. Castejón (2005) afirma que las PYMEs se constituyen como las mayores proveedoras de software nacionales, y en algunos casos han podido participar y ganar licitaciones de proyectos grandes y de criticidad importante; debido a sus bajos precios con respecto a la competencia foránea y las grandes empresas locales. En fin, la participación e importancia en Venezuela de la industria del software, es de gran relevancia económica. Aún, con buenas estrategias de negocio, se espera que en un corto plazo, se lograra posicionar la industria del software nacional en los mercados internacionales, colocando al país como emergente en materia de desarrollo de software.

En este sentido, el desarrollo de la industria del software como pilar importante en el desarrollo del país, está caracterizada por los siguientes rasgos (Reyes, 2005):

- *Es una industria donde las pequeñas y medianas empresas juegan un rol importante, y es, asimismo, una industria intensiva en recursos humanos de alta calificación.*
- *Los activos y las inversiones requeridas para comenzar y desarrollar una empresa de software son relativamente bajos en comparación con otros sectores industriales.*
- *Los principales factores de producción de esta industria son el talento humano, las capacidades de producción, la calidad, la innovación y la capacidad de insertarse en poderosas cadenas de suministros globales.*
- *Los principales activos de una empresa de software son los conocimientos, derechos de autor, relaciones con los clientes, etc.*

Un estudio realizado por Rivero et al. (2007), revela como es la tendencia (por tamaño de la empresa) de la industria del software en Venezuela, en base a 150 empresas ubicadas en las principales ciudades del país (Figura 2). Aquí puede observarse que el 83,68% de las empresas, son empresas que van desde ser microempresas hasta empresas pequeñas, representado el estrato principal las empresas pequeñas con un 42,86% de la muestra. Parte del estudio también revela que las empresas de software en Venezuela, son industrias muy jóvenes, donde el 74% tienen en el mercado entre 1 y 12 años, siendo las PYMEs (42%) creadas en los últimos 7 años (al combinar los factores tamaño de la empresa con la antigüedad). Dedicar esfuerzos para aportar beneficios a las empresas desarrolladoras de software en Venezuela, centradas en PYMEs, resulta ser una actividad necesaria en la búsqueda de hacerlas más competitivas.

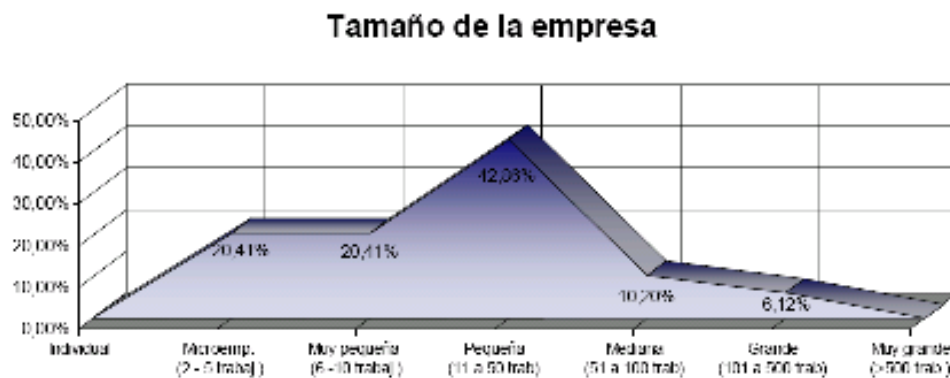


Figura 2- *Tamaño promedio de las empresas de la Industria Nacional del Software.*
Fuente: Montilva (2007)

Por su parte, Granados et al. (2008) igualmente realizó un estudio que revela aspectos interesantes sobre la industria del software en Venezuela, donde puede observarse la distribución geográfica de las empresas desarrolladoras de software en el país (Figura 3). El estudio está basado en el siguiente perfil general de la Industria del Software:

- *Universo de estudio: Empresas que comercializan productos y/o servicios de software, sin fines de autoconsumo.*
- *No. de Empresas: 71*
- *Criterio de estratificación muestral: Región Los Andes-Zulia (Mérida, Maracaibo, Punto Fijo, San Cristóbal), Región Capital (Caracas, San Antonio de los Altos, Guatire) y Región Centro-Oriente (Maturín, Barquisimeto, Maracay, Valencia y Puerto Ordaz).*

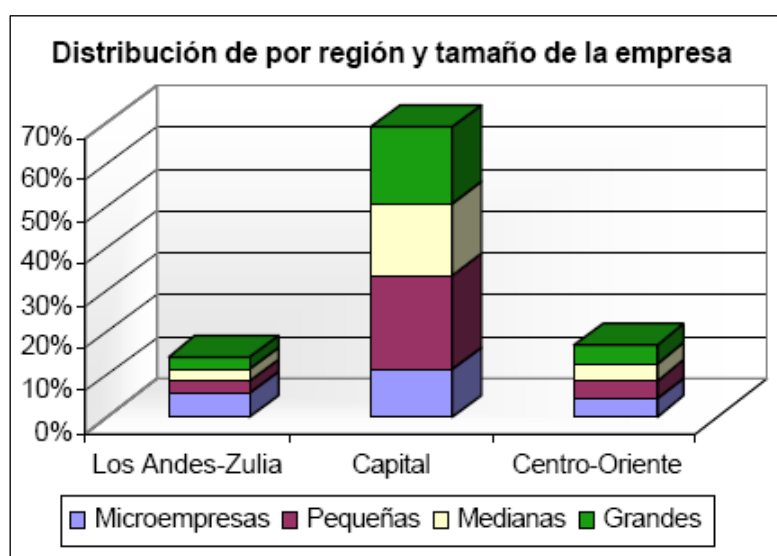


Figura 3- *Distribución geográfica de empresas desarrolladoras de software en Venezuela.*
Fuente: Granados (2003).

De forma más específica, en el estudio de Rivero et al. (2007), presentan mediante la figura 4 los tipos de aplicaciones producidas por la industria nacional de software. Aquí se revela que el 41% de las empresas encuestadas elabora aplicaciones de negocio o administrativas, el otro 59% elabora aplicaciones embebidas, industriales, software del sistema, aplicaciones científicas, software educativo, de entretenimiento y otros. En todo caso, lo aquí revelado es que la tendencia al desarrollo de software nacional está en continuo crecimiento.

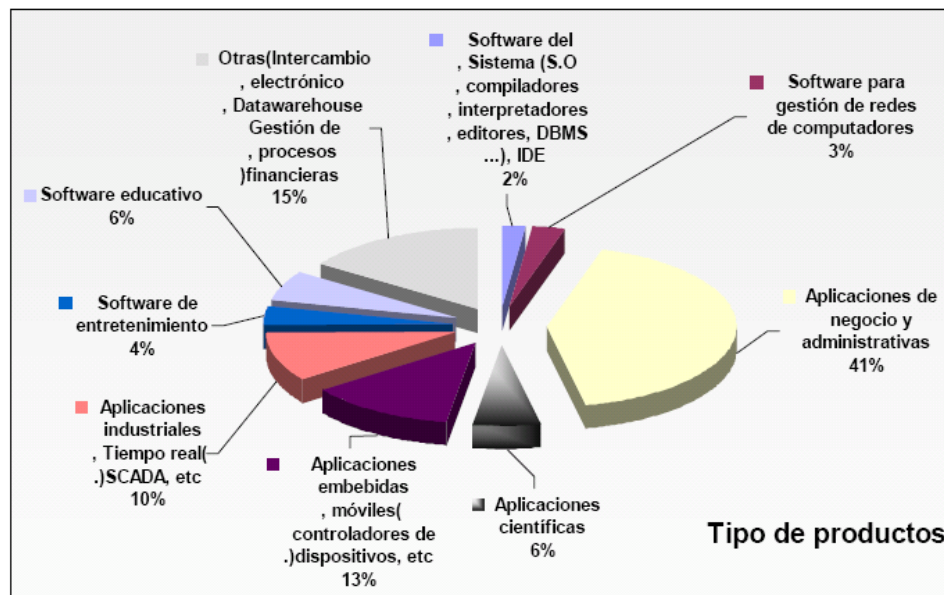


Figura 4 - Tipos de aplicaciones que produce la Industria Nacional de Software.
Fuente: Montilva (2007).

Un aspecto de suma importancia demostrado en el estudio de Rivero et al. (2007), lo constituye el descubrir las principales necesidades de conocer, manejar y actualizarse en todo lo concerniente al proceso de desarrollo de software. Vale destacar en la Figura 6 cómo las pruebas de software, aunque no es la más crítica de los procesos inherentes al desarrollo de software, se ubica en el tercer puesto, por lo que su necesidad de afianzar este rubro y especializarse en todo lo concerniente a pruebas del software, madurando el concepto, es de vital importancia para producir software con la calidad exigida por sus clientes y aun a nivel internacional. Esto trae consigo que dentro de los procesos de gestión de proyectos de software, el aseguramiento de la calidad (gestión de la calidad del software - SQA), también se vea altamente necesitada de capacitación en su recurso humano (Figura 5 y 6). Gestión de la calidad y Pruebas del software, son temas que deben estar de la mano y contar con un nivel de madurez óptimo para que pueda ser aplicado en el proceso de desarrollo de software.

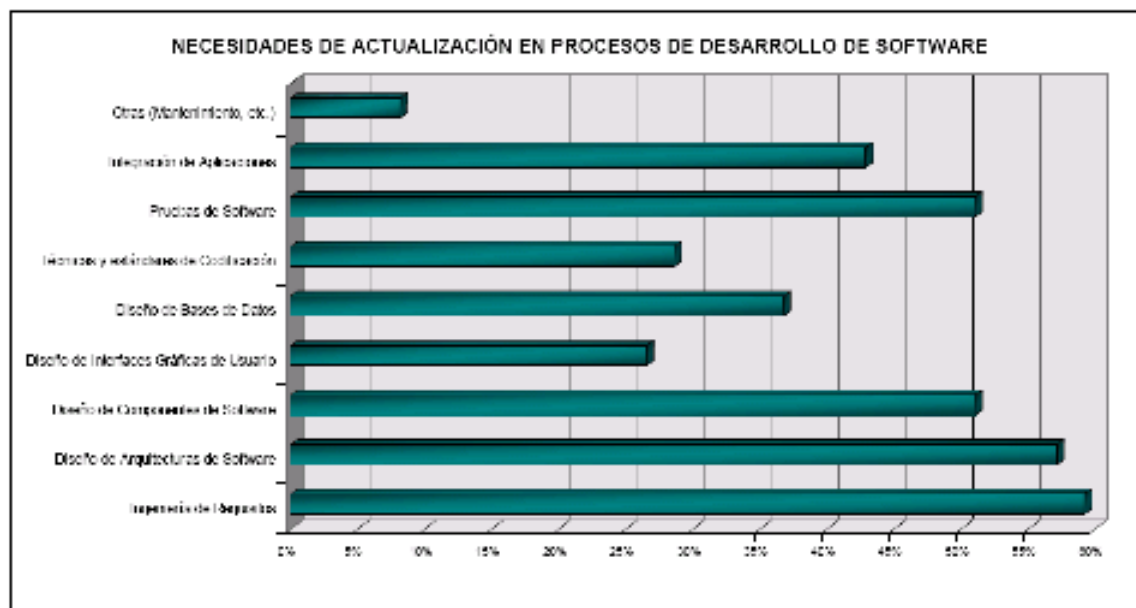


Figura 5 - *Necesidades de actualización en procesos de desarrollo de software.*
Fuente: Montilva (2007).

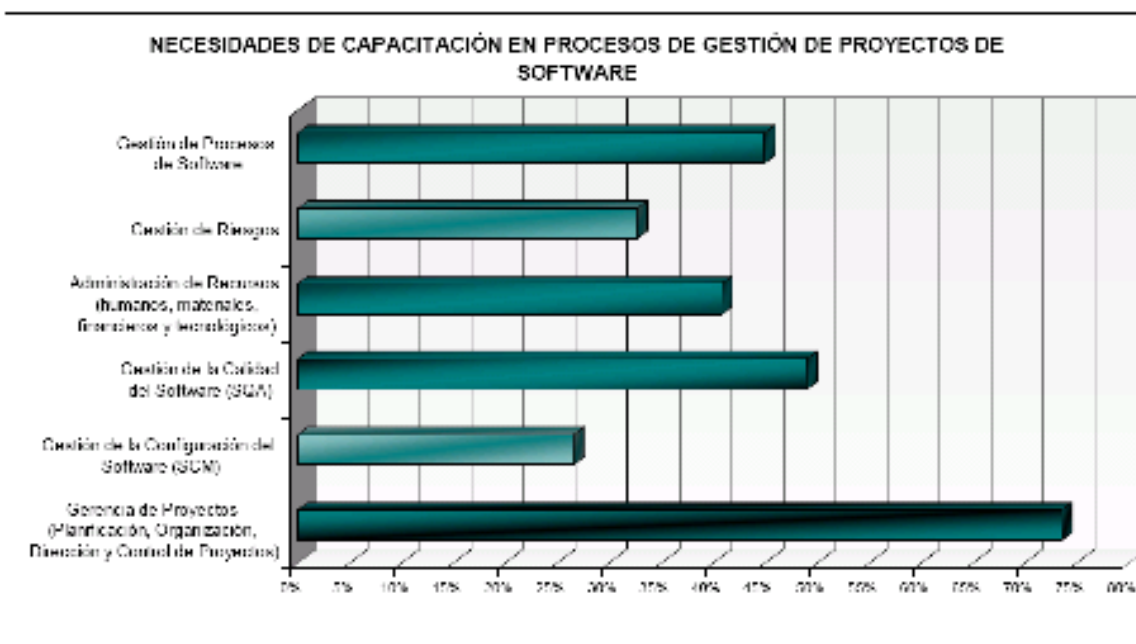


Figura 6 - *Necesidades de capacitación en procesos de gestión de proyectos de software.*
Fuente: Montilva (2007).

Por lo antes expuesto, se concluye que el campo de acción para reforzar y/o mejorar los procesos de dichas empresas, en cuanto a pruebas se refiere, es amplio y concurrido. Esta conclusión toma mayor fuerza cuando se observan algunas tendencias expresadas por Méndez et al. (2007) respecto a las PYMEs en el ámbito de las pruebas. Ellos citan a Pressman (2005) planteando que las empresas desarrolladoras de software invierten en la disciplina de pruebas entre el 30% y 50% del costo total del desarrollo del software. Hablando del tema de la automatización de pruebas, Granados et al.

(2008) presenta la figura 7 donde resume la utilización de algunas herramientas de apoyo durante el desarrollo de software, siendo que el área relativa al Testing Automático, revela que alrededor del 60% de las empresas no usan este tipo de herramientas. Esto representa un esfuerzo considerable indicando que es una disciplina cuyos resultados (confiabilidad de la prueba) pueden impactar sobre la percepción del cliente o usuario en cuanto a la calidad del software que se le está entregando.

Herramientas de apoyo en la ejecución del proyecto	Frecuencia de uso	Micro-empresas	Pequeñas	Medianas	Grandes	Total
Análisis y Diseño de Software asistido por herramientas de diagramación	Uso Frecuente	9,86%	15,49%	14,08%	16,90%	56,34%
	Uso ocasional	5,63%	7,04%	4,23%	1,41%	18,31%
	Conoce y no lo usa	5,63%	7,04%	5,63%	5,63%	23,94%
	No conoce	0,00%	0,00%	0,00%	1,41%	1,41%
Debugger	Uso Frecuente	11,27%	16,90%	9,86%	12,68%	50,70%
	Uso ocasional	5,63%	8,45%	4,23%	9,86%	28,17%
	Conoce y no lo usa	4,23%	4,23%	7,04%	2,82%	18,31%
	No conoce	0,00%	0,00%	2,82%	0,00%	2,82%
Control de Versiones	Uso Frecuente	14,08%	12,68%	12,68%	19,72%	59,15%
	Uso ocasional	0,00%	9,86%	5,63%	5,63%	21,13%
	Conoce y no lo usa	5,63%	5,63%	2,82%	0,00%	14,08%
	No conoce	1,41%	1,41%	2,82%	0,00%	5,63%
Testing Automático	Uso Frecuente	4,23%	2,82%	1,41%	2,82%	11,27%
	Uso ocasional	8,45%	7,04%	5,63%	8,45%	29,58%
	Conoce y no lo usa	5,63%	14,08%	14,08%	11,27%	45,07%
	No conoce	2,82%	5,63%	2,82%	2,82%	14,08%
Control de Configuraciones	Uso Frecuente	4,23%	8,45%	4,23%	7,04%	23,94%
	Uso ocasional	4,23%	7,04%	7,04%	8,45%	26,76%
	Conoce y no lo usa	8,45%	9,86%	12,68%	7,04%	38,03%
	No conoce	4,23%	4,23%	0,00%	2,82%	11,27%
Gestión de Requisitos	Uso Frecuente	11,27%	9,86%	16,90%	18,31%	56,34%
	Uso ocasional	2,82%	7,04%	1,41%	2,82%	14,08%
	Conoce y no lo usa	4,23%	7,04%	4,23%	2,82%	18,31%
	No conoce	2,82%	5,63%	1,41%	1,41%	11,27%

Figura 7 - Utilización de algunas herramientas de apoyo durante el desarrollo de Software. Fuente: Granados (2008).

Es importante traer a colación, en líneas generales, algunos obstáculos que enfrentan las PYMEs según Erard (1999) y que son importantes conocer para contextualizarlo dentro de las empresas desarrolladoras de software venezolanas: Se centran en poca gente las diferentes actividades de negocios y tienen flexibilidad en los roles que ejerce el personal. Hay una falta de formalidad en las relaciones y roles entre los individuos que interactúan. El flujo de trabajo flexible, es regido por las necesidades del momento. Erard afirma que la mayor dificultad de las PYMEs es conseguir las personas adecuadas y las herramientas (tecnología y maquinaria) necesarias para realizar su trabajo. Según Rivero et al. (2007), las mayores deficiencias de los expertos están en el manejo de modelos de proceso y gestión (conocimiento y herramientas). Di Paula et al. (2007) agrega de acuerdo a sus conclusiones que a pesar de ser muy capaces en su negocio y las tecnologías que manejan, sus procesos son erráticos y dependen

exclusivamente de los estándares que determine el líder del proyecto. Todas estas realidades, hacen más fuerte la necesidad ya evidenciada de establecer formalidad en procesos internos de las PYMEs y especialmente dentro del marco de la Disciplina de pruebas. Este escenario debe aprovecharse a su vez para impulsar el uso de herramientas que apalanquen la Disciplina de pruebas dentro del proceso de desarrollo de software, justificado en las carencias del uso de herramientas que agilicen sus procesos.

2.3 Bases Teóricas

El desarrollo de software debe estar enfocado en la imperiosa realidad de garantizar calidad tanto en todo su proceso como en sus resultados (producto) para así traducirse en una estrategia empresarial y lograr a través de este, las ventajas competitivas en un entorno globalizado.

Tomando como base esta realidad, la presente investigación se ha dividido en cinco áreas de importancia para tratar de profundizar en el tema de la calidad de forma incremental y tomando como base varias fuentes bibliográficas. Todas estas fuentes han sido analizadas con el propósito de encontrar el enriquecimiento de conocimiento y entender los conceptos analizados.

La primera parte aborda todo lo concerniente a la calidad del software iniciando por definir el término calidad hasta llegar a aplicar dicha definición en el ámbito del software. Seguidamente se estudia el área del aseguramiento de la calidad del software, analizada bajo diferentes ópticas (autores, estándares, tendencias) y correlacionado dichos conceptos para determinar los componentes de lo que se conoce como Aseguramiento de la calidad del software ideal. Una tercera se constituye como un glosario dedicado al tema central de esta investigación: las pruebas del software, mediante la cita y análisis de conceptos variados acerca de todo lo concerniente a Pruebas del software y su importancia en el proceso de desarrollo del mismo. Una cuarta, trata acerca de la disciplina de pruebas, constituida en la mayoría de sus defensores, como una actividad dentro del proceso de desarrollo de software, diferenciada en roles, actividades, fases y demás componentes de acuerdo a la metodología de desarrollo elegida. Finalmente una quinta, aborda lo concerniente a los modelos de calidad y su clasificación con una explicación detallada de cada uno, identificando su aporte al área de calidad.

I. Calidad del Software

Lograr un alto nivel de calidad de un producto o servicio es el objetivo de muchas organizaciones a nivel mundial. Asegurar la calidad en cada una de las fases por las que transita el desarrollo de un producto/servicio se traducirá en múltiples beneficios para que el mismo produzca los resultados esperados. Esto es lo que James (2000) trata de enseñar al desarrollar el punto Desarrollo de la calidad a través del aseguramiento de la calidad.

La industria del software no escapa de esta realidad, más cuando el software, en la actualidad, es el combustible en los negocios modernos, reglas del gobierno y sociedades que desean ser conectadas, en fin, permite el desarrollo de la economía mundial (Krucchten, 2000). Esto determina que el software tiene un valor muy importante en nuestra sociedad.

El término calidad de Software, puede ser relacionado con la aplicación de calidad en el ámbito del software; sin embargo, bien vale la pena tener claro el concepto de calidad antes de aplicarlo al software. Con este propósito, se traen a colación variados conceptos de calidad para así contextualizar su incidencia en lo que concierne al software.

La real academia española (2001), define *calidad* como la propiedad o conjunto de propiedades inherentes a algo, que permiten juzgar su valor. Llama la atención que dicho conjunto de características puedan determinar el valor de algo; es allí donde toma fuerza el decir: ‘tal producto es de alta o baja calidad’.

La ASQ (American Society for Quality, por sus siglas en ingles) plantea que en el sentido técnico de la calidad (y no de forma subjetiva, dado que cada persona o sector puede definir calidad empíricamente), tiene dos significados: 1. Las características de un producto o servicio que refieren su capacidad de satisfacer lo indicado o las necesidades implícitas; 2. Un producto o un servicio libre de deficiencias.

Juran (1998) considerado el Padre de la Calidad, destaca dos conceptos de los muchos significados atribuibles al termino calidad, describiéndola desde la perspectiva

del cliente. Calidad significa esas características de los productos que resuelven necesidades del cliente y de tal modo proporcionan la satisfacción de cliente, es decir, alta calidad significa una gran cantidad de características que satisfacen las necesidades de los clientes. El segundo aspecto se relaciona con “libertad de problemas”: Alta calidad consiste en pocos defectos, libre de deficiencias.

Crosby (1980) en comparación con algunos gurúes de la calidad (Juran, Demming e Ishikawa), flexibiliza el concepto de calidad proponiendo que ésta debe ser resultado del trabajo de todos y no solo del departamento de control de calidad; la calidad es libre pero no es un regalo. Crosby, basó sus ideas en definir la calidad como la conformidad con los requisitos, es decir, gerenciar la calidad es una manera sistemática de garantizar que las actividades organizadas suceden en la manera que se planean.

Cabe destacar que todos estos conceptos, están inmersos en el ámbito de la legislación Venezolana, donde la Ley del sistema Venezolano para la Calidad (2002) en su Capítulo I, Art. 4, define calidad como el Grado en que un conjunto de características inherentes a bienes y servicios cumple con unas necesidades o expectativas establecidas, generalmente implícitas u obligatorias (requisitos). Todo esto permite establecer que los elementos presentes en la calidad son: características, cumplimiento/satisfacción, requisitos. Esto quiere decir que siempre estarán presentes expectativas que deben ser cubiertas, de acuerdo a las necesidades expresadas (directa o indirectamente) por los clientes y/o usuarios del producto o servicio resultante. A su vez, el grado en que son cubiertas, determinará qué tan bueno o malo es el producto resultante (mientras más necesidades sean cubiertas será visto como de mejor calidad).

Por ser tan importante la calidad, se hace necesario incluir temas que permitan profundizar en los conceptos de calidad y calidad del Software así como la importancia de asegurar calidad en el proceso de desarrollo de software.

Unhelkar (2003), de una manera más formal, establece los ingredientes esenciales en la creación de “buena” calidad, describiéndolos de la siguiente manera: 1.- Un producto, 2.- Un proceso de producción (el proceso o método usado en la creación del producto es tan importante como el producto mismo), 3.- Un mecanismo de

comunicación (para comunicarse con el dominio del desarrollo y a través de la industria –usuarios, productos, procesos), 4.- Una mejora continua de los ingredientes anteriores (es un gran esfuerzo producir un buen producto de calidad, mejorar lo que se logra es importante).

Uniendo todas estas ideas en el ámbito del software, el concepto de calidad bien puede extrapolarse a lo que es calidad aplicada al software, pues mantiene su esencia pero con un tecnicismo que debe entenderse, acordarse y refinarse. El software debe ser visto como un producto. Sin embargo, Unhelkar (2003) reflexiona acerca de que la naturaleza del software es tal que dificulta juzgar su calidad objetivamente, pues necesita a los usuarios finales y los productores de tal software. Por esta razón, hay una necesidad de que lo subjetivo de este concepto prevalezca ante tan diverso grupo de personas que demandan un amplio estudio contextual para entender la calidad del software, dejando que la calidad signifique diversas cosas para diferentes personas. He aquí la importancia de darle la apropiada dirección en su definición.

Diversos conceptos de calidad de software, se traen a colación, para su discusión:

Según el IEEE (Institute of Electrical and Electronics Engineers, por su siglas en ingles) en su estándar 610.12-1990, la *calidad de software* es definida como el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario. Aquí los elementos presentes en el concepto de calidad del software son: requisitos, necesidades y expectativas.

El proceso unificado de Rational RUP (2005), define la calidad de software como las características que tienen que demostrar el logro de producir un producto que satisface o excede los requerimientos convenidos y medidos según criterios y medidas acordados y que es producido de acuerdo a un proceso estipulado. Esta definición manifiesta la no simplicidad de todo lo concerniente con la calidad de software dentro del proceso de desarrollo del mismo. Lograr la calidad no es simplemente satisfacer los requerimientos o producir un producto que satisfaga las expectativas y necesidades de los usuarios; más bien, la calidad del software también debe incluir lo planteado por RUP, la identificación de las medidas y criterios para demostrar que la calidad se ha

alcanzado y la implementación de un proceso que asegure que el producto resultante contiene el grado de calidad esperado.

En una aproximación al concepto de calidad de software, la “Guide to the Software Engineering Body of Knowledge”, conocida como SWEBOK (2004), afirma que la importancia de este concepto es que los requerimientos del software definen las características de calidad requeridas del software e influyen los métodos y criterios de aceptación para determinar esas características. Esta definición, al igual que la de RUP, hace especial énfasis en los métodos y criterios de aceptación que se determinan en el proceso de calidad del software, además de tener la importancia de tener claros los requerimientos de software, que guiarán la calidad en el software.

Por su parte, Somerville (2006) afirma que calidad del software significa, sencillamente que un producto debe resolver/cubrir su especificación. Sin embargo al hablar sobre la administración de la calidad del software, presenta algunos problemas y/o realidades de los sistemas:

- *Hay una tensión entre los requisitos de calidad del cliente (eficacia, confiabilidad, etc.) y los requisitos de calidad del desarrollador (capacidad de mantenimiento, reutilidad, etc.);*
- *Algunos requisitos de calidad son difíciles de especificar de forma inequívoca;*
- *Las especificaciones del software son generalmente incompletas y a menudo contrarias.*

Ante esta realidad, el autor está convencido de que administrar la calidad del software permitirá obtener gran calidad a pesar de las imperfecciones en las especificaciones de requerimientos, siempre y cuando las actividades concernientes, se ejecuten dentro del tiempo indicado.

Ahora bien, ante las variadas opciones presentadas, surge la de Pressman (2005) que cobra una mayor fuerza por ser más explícita e incluir conceptos netamente de software, pero que según el propio autor, puede ser ampliada o modificada con todos los conceptos presentados, hasta lograr una definición más clara: Concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados, y con las características implícitas que se espera de todo software desarrollado profesionalmente. De esta definición, se pueden

observar tres puntos importantes que deben tomarse en consideración, como lo expresa Pressman (2005):

1. *Los requisitos de software (explícitos) son la base de las medidas de la calidad. De hecho, una falta de correspondencia entre estos requerimientos, ya da por sentado una falta de calidad importante.*
2. *Existen un conjunto de estándares especificados constituidos por un conjunto de criterios de desarrollo que guiarán la mejor manera de hacer las cosas, y cuya ausencia se traducirá en falta de calidad.*
3. *Existe otro tipo de requisito de software que normalmente no se mencionan formalmente (implícitos), los cuales deben tomarse en cuenta en unión con los requisitos explícitos para que no se pueda dudar de la calidad del software que se construye.*

En todos estos conceptos lo que se quiere tratar de minimizar es la variación que pueda existir entre las expectativas de los clientes y los resultados que emita el producto final (software). Controlar esta variación es la clave de un producto de alta calidad, y se hace controlando desde el proceso en sí, los recursos utilizados y los atributos de calidad que deberían caracterizar el software final. Esto es la esencia de la Calidad de Software y que será utilizada para efectos de esta investigación, basándose en la definición de Pressman, pues es la que engloba los elementos que dirigen la disciplina de pruebas.

Sin embargo, entender la calidad del software sigue siendo una actividad a la cual no se le da la importancia requerida, aunque se tenga la certeza que garantizarla es una buena y necesaria práctica dentro del desarrollo de sistemas. Esta situación es presentada por Kruchten (2000) cuando afirma que diferentes proyectos de desarrollo del software fallan de diversas formas, por lo que es posible identificar algunos problemas que son latentes en organizaciones que buscan implementarlo, enmarcado dentro de los síntomas que caracterizan dicho tipo de proyectos:

- Comprensión inexacta de las necesidades de los usuarios finales.
- Las necesidades del usuario o la empresa no se cumplen.
- Procesos de software normalmente improvisados.
- Los requerimientos cambian rápidamente.
- Los módulos no se integran.
- Difícil de darle mantenimiento.
- Baja calidad o experiencia del usuario final.
- Descubrimiento tardío de fallas.
- Pobre calidad del software.
- Cada miembro de los equipos hacen todo a su manera, haciendo imposible reconstruir quién cambió qué, cuándo, dónde y por qué.
- Desempeño deficiente bajo carga.

- Problemas de construcción y liberación

Todas estas realidades expresadas, convergen en una realidad latente que debe menguar para dirigir en un mejor camino el desarrollo de software y por ende, todo lo concerniente al software de calidad.

Ante todos estos problemas y realidades presentes en el desarrollo del software, resalta la necesidad de intentar solucionarlos o al menos mitigarlos desde etapas tempranas. Es aquí donde surge el concepto de Aseguramiento de la Calidad del software, el cual será tratado en la siguiente sección, analizado bajo diferentes focos pero llegando a un consenso de lo que debería ser el proceso de aseguramiento de la Calidad ideal.

II. Aseguramiento de la Calidad del Software

El Aseguramiento de la calidad, emerge como una gran necesidad en el desarrollo de software. Según RUP (2005) una de las mejores prácticas que comúnmente son usadas por organizaciones exitosas instauradas en la actualidad para el desarrollo de software, es verificar calidad continuamente. Mientras más tarde se consigan fallas en el software será más costoso repararlos. RUP (2005), plantea que valorar continuamente la calidad de un sistema con respecto a su funcionalidad, fiabilidad, desempeño de la aplicación y del sistema completo, es de suma importancia al desarrollar un sistema, todo con el fin de dar respuesta a la problemática de sistemas con síntoma de calidad pobre, por la causa de pruebas pobres. Este punto, da entrada a lo que es el Aseguramiento de la calidad del software.

Aunque para este concepto las siglas en ingles son únicas SQA (Software Quality Assurance, por sus siglas en inglés), algunos autores lo traducen como: Administración de la calidad, Garantía de la Calidad del software o bien, Aseguramiento de la Calidad del Software. En todo caso, algunas de las definiciones que se plantean para este concepto están:

La IEEE - Computer Society (IEEE Std.730-2002), plantea que el *aseguramiento de la calidad* es un patrón planeado y sistemático de todas las acciones necesarias, para proveer suficiente confianza de que el software se construye conforme a los requerimientos técnicos preestablecidos.

Para Sommerville (2006), aseguramiento de la calidad es el establecimiento de un marco de trabajo de procedimientos y estándares organizacionales que conduce a software de alta calidad; define un marco de trabajo para lograr la calidad del software.

Por su parte, SWEBOK (2004) plantea que el proceso de SQA proporciona seguridad de que el producto y proceso de software en el ciclo de vida del proyecto se corresponden con sus especificaciones de requerimientos planificados, decretando y realizando un conjunto de actividades que proveerán la confianza adecuada de que la calidad se está construyendo dentro del software.

El modelo de capacidad y madurez integral, CMMI (Capability Maturity Model Integration, por sus siglas en ingles), define éste concepto como los medios planeados y sistemáticos para asegurar a la gerencia que los estándares, las prácticas, los procedimientos y los métodos definidos del proceso son aplicados. El área del proceso de aseguramiento de la calidad de proceso y producto apoya la entrega de productos y servicios de alta calidad proveyendo al equipo del proyecto (staff y encargados de todos los niveles) de la visibilidad apropiada, el feedback, procesos y trabajo asociado a los productos a través de la vida del proyecto. CMMI-DEV versión 1.2 (2006).

Pressman (2005) afirma que la garantía de la calidad del software es la preocupación de cada Ingeniero del Software para reducir costos y mejorar el tiempo para mercadear el producto. Bajo este concepto, el mismo autor realiza varios planteamientos importantes: la conformidad existente entre los requerimientos del software es el fundamento sobre el cual se mide la calidad del software. Los estándares específicos se utilizan para definir el criterio de desarrollo utilizado para tutelar la manera en la cual se dirige el software.

Todos estos conceptos tienen aspectos en común que deben resaltarse; estos planteamientos apuntan hacia demostrar la necesidad de que exista un proceso, planificación, equipo de personas, entre otros elementos para obtener un producto de calidad, la suma de los cuales excede las limitaciones de un simple control de calidad tradicional, esto es, la existencia de un proceso que asegure calidad, dando por sentado

la imposibilidad de pretender inyectar calidad al producto de software una vez finalizado el proceso de desarrollo del mismo.

Diferentes autores (Pressman, Sommerville, SWEBOOK, Schulmeyer y McManus, Unklear) hacen aportes acerca de las actividades o fases que deben estar presentes en el aseguramiento de la calidad. Dentro de cada fase, se intercalan actividades y tareas específicas a lo largo del ciclo de desarrollo o mantenimiento de una aplicación, con la intención de reducir errores y demás propósitos del aseguramiento de la calidad de software. A continuación se explican brevemente cada uno de ellos, en busca de hacer una integración valiosa en cuanto a ésta actividad.

Profundizando en el tema del Plan de la Calidad, la IEEE establece un estandar para los planes de aseguramiento de la calidad del software (IEEE Std 730-2002), en el que proporciona una ayuda al momento de determinar el contenido y preparación de los planes de calidad tanto para el desarrollo como el mantenimiento del software, dirigido a tres grupos de personas a las que aplica o afecta: usuario, proveedor y público (cliente). En todo caso, la completitud y profundidad en los elementos presentes en un plan de calidad propuesto por la IEEE, hace considerarla como una guía obligada para esta área.

De forma resumida, en la Figura 8 (diagramas de actividades UML) pueden visualizarse los roles, entregables, actividades por rol y procesos generales que propone el autor y deben cumplirse en el Aseguramiento de la calidad durante todo el proceso. De cada una de estas actividades se derivan tareas de suma importancia para todo el proceso que velan por asegurar la calidad deseada, tales como selección de técnicas de calidad, estimación y medición proceso calidad, identificar y reportar errores, validar correcciones de errores, entre otros.

Uno de los roles primarios llevados a cabo por el aseguramiento de la calidad y control de calidad es observar lo que está siendo realizado por otros, evaluarlo, criticarlo y mejorarlo. Bajo este concepto de ideas, se propone que el personal de calidad debe ser independiente de los diseñadores y desarrolladores. Esta independencia asegura que la actividad de calidad continuará observando todo el proceso desde otra perspectiva.

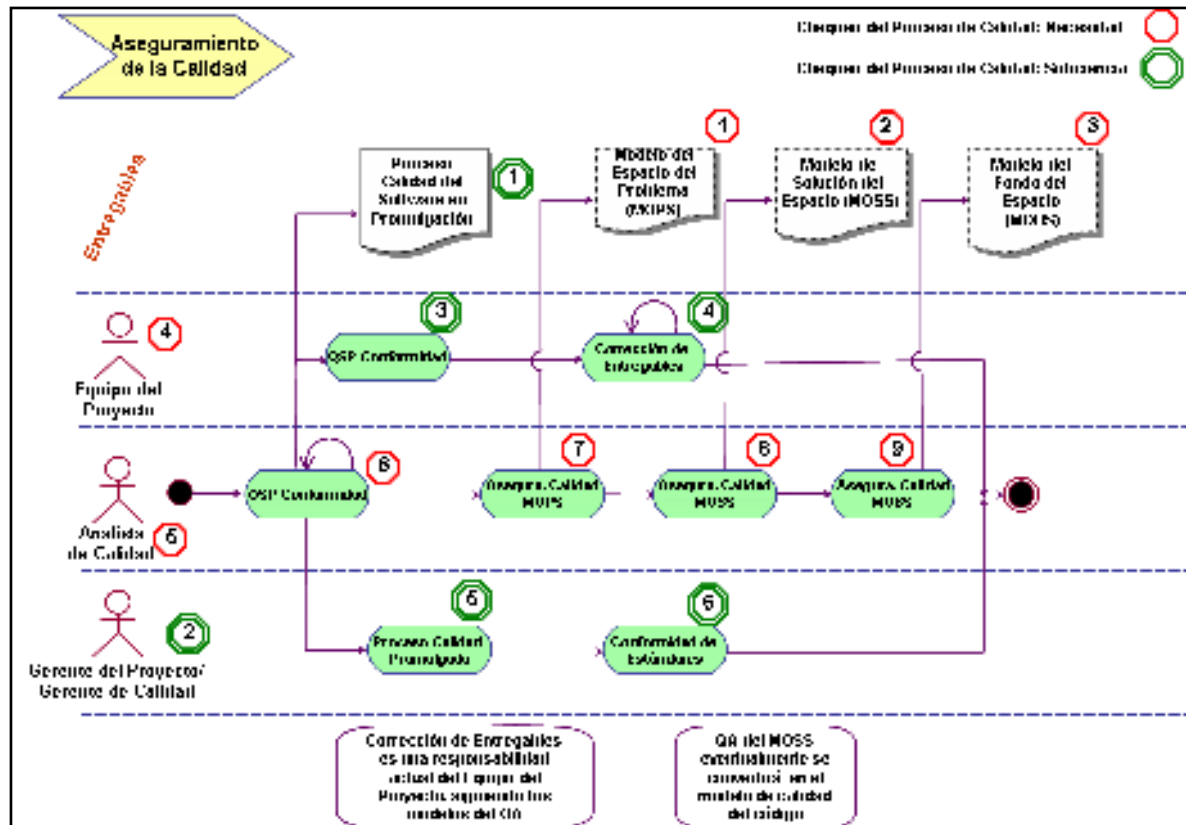


Figura 8- Procesos y Componentes para el Aseguramiento de la Calidad, Fuente: (Unhelkar, 2003).

CMMI (2006), apoya esta afirmación al expresar que tradicionalmente, el grupo de aseguramiento de la calidad que es independiente de este proyecto, puede ofrecer la objetividad esperada en el proceso. Esto puede ser apropiado en algunas organizaciones (generalmente las pequeñas), pero en organizaciones con un proceso abierto, orientado a la cultura de calidad, el aseguramiento de la calidad del proceso y producto podrán ser llevados a cabo, parcial o totalmente por pares, y la función de garantía de calidad puede estar inmersa en el proceso.

De todo el material recopilado, la Tabla 1 compara los conceptos presentados por los diferentes autores y metodologías sobre aseguramiento de la calidad, buscando establecer una correlación resultante entre los mismos, además de identificar algunas diferencias sustanciales entre sus criterios. La idea, es poder determinar cuáles serían las actividades contempladas o no, comparando todos estos criterios analizados y aproximarse al aseguramiento de la calidad ideal que contenga las mejores prácticas de todos estos autores. Los criterios de comparación fueron elegidos con propósitos descriptivos a manera de vista rápida de cada tendencia analizada, los cuales son: *nombre* (a manera de homologar criterios, cómo lo llama cada autor), *elección de*

estándares (si se basa en estándares de calidad o al menos son tomados en cuenta), *revisión de calidad* (si en el proceso contempla actividades para dichas revisiones), *registrar fallas* (tiene la posibilidad de asentar fallos en el proceso), *corrección de fallas* (están establecidos actividades para corregir defectos), *control y gestión de cambios* (si contempla la administración de cambios en sus procesos), *riesgos y su administración* (contempla los riesgos y su gestión), *recopilación y análisis de métricas* (mide sus procesos a través de métricas), *QA estadística* (medición de los procesos de aseguramiento de la calidad usando la estadística), *presente en todo el proceso de desarrollo* (se evidencia su presencia en todo el proceso de ciclo de vida del software), *especificación de Roles* (identifica roles de quienes participan en el proceso con sus responsabilidades), *plan de SQA* (propone o se guía por un plan de aseguramiento de la calidad detallado), *especificación documentación* (apoya el proceso en documentación claramente identificada), *independencia team calidad vs. desarrolladores* (considera que el equipo de calidad debe ser distinto al equipo desarrollador de software) y *mejora de procesos* (contempla el mejoramiento de los procesos).

Como puede observarse en la Tabla 1, sin importar los diferentes nombres cómo pueda ser llamado el aseguramiento de la calidad, este tema tiene más puntos comunes que discordantes entre sus defensores. Todos especifican tener un Plan de aseguramiento de la calidad, por lo que puede derivarse su gran importancia para dirigir el proceso. Igualmente, la especificación de los roles que ejecutarán las actividades, está presente en todos los autores, no así la independencia entre el equipo que pertenezca al aseguramiento de calidad y los que pertenezcan al grupo desarrollador. Esta independencia parece no ser un tema de especial cuidado entre sus defensores; con esto se podría deducir el hecho de que es innecesario dos equipos distintos, siempre y cuando los mismos, puedan trabajar con más de un rol por actividad (intercambiar roles). Los criterios revisión de calidad, registrar fallas y su corrección, la presencia en todo el proceso de aseguramiento de la calidad, implementar la mejora de procesos y tener documentación especificada, son criterios presentes en casi todos los autores, a excepción de una sola por autor. Es importante hacer notar que la presencia del aseguramiento de la calidad en todo el ciclo de vida del software, es una realidad de suma importancia pues con la presencia de ésta, la mejora de los procesos sería una actividad natural, consecuencia del control continuo. Ahora bien, es importante destacar como la administración de riesgos y el control de cambios no forman parte de criterios

comunes para sus defensores, actividad que no debería descuidarse en el desarrollo de proyectos de software, pues siempre están latentes.

Ante toda esta información recopilada y analizada, bien vale la pena poder establecer las diferencias y relaciones del tema de aseguramiento de la Calidad con la disciplina de pruebas. Esta diferencia es expresada por Unhelkar (2003), quien argumenta acerca de *‘Aseguramiento de la calidad y pruebas: no los confundamos’*. Normalmente se suele escuchar la frase: “El producto está en aseguramiento de la calidad”, pero no es una correcta interpretación de éste concepto. Viendo el aseguramiento de calidad como un conjunto de herramientas que son usadas para probar aplicaciones y componentes, solo implica que dichas herramientas ayudan a automatizar productos de pruebas, pero esto es sólo una parte de la función de QA denominada Control de calidad o pruebas, cuyo propósito es la medición del producto contra un estándar o atributo, identificando defectos y corrigiéndolos hasta tener un producto libre de defectos.

Lo antes planteado hace pensar que cuando se habla de aseguramiento de la calidad, se habla en un sentido general, presente en todo el proceso, pero no aterriza en algo específico para poder validar si esa calidad se está viendo reflejada o no en el proceso y por ende, en el producto resultante. Por esta razón, la próxima sección analiza todo lo concerniente con la Disciplina de Pruebas, que como disciplina, converge en actividades, entregables y procesos bien diferenciados y puntuales (más especializado). Allí se hace referencia a los autores que la defienden, conceptos y cuáles son los roles, fases, actividades, artefactos, entre otros, planteada por cada uno de ellos. Todo esto, con el fin dar la suficiente atención a dichos conceptos, destacar la importancia que se deriva de ellos y por ende, poder diferenciar los conceptos de aseguramiento de la calidad y disciplina de pruebas.

En la siguiente sección se agrupa todo lo concerniente a las pruebas, fin esencial de la disciplina de pruebas, modelos de calidad del producto para garantizar el aseguramiento de la calidad.

Autores	<i>Pressman (2005)</i>	<i>Sommerville (2006)</i>	<i>Unhelkar (2003)</i>	<i>Mendis (1993)</i>	<i>CMMI (2006)</i>	<i>ITIL V3 (2007)</i>
Criterios						
Nombre utilizado para definir el Aseguramiento de la Calidad (QA)	Garantía de la calidad del software	Administración de la calidad del software	Proceso de calidad del software	Evaluación de la Calidad del Software	Garantía de la calidad del proceso y del producto	Mejora Continua del Servicio de TI
Elección de estándares	Si contempla	Si contempla	Si contempla	Si contempla	Si contempla	No especifica
Revisión de Calidad	Si contempla (Auditoría de entregables)	Si contempla (Control de calidad)	Si contempla	No especifica	Si contempla (Evaluación continua calidad)	Si contempla
Registrar fallas	Si contempla	No especifica	Si contempla	Si contempla	Si contempla	Si contempla
Corrección de fallas	Si contempla	Si contempla	Si contempla	Si contempla	No especifica	Si contempla
Control y gestión de cambios	Si contempla	No especifica	No especifica en SQA sino en Admón. calidad	No especifica	Si contempla	Si contempla
Riesgos y su administración	Si contempla	Si contempla	No especifica	No especifica	Si contempla	Si contempla
Recopilación y análisis de métricas	Si contempla	Si contempla	Si contempla	No especifica	No especifica	No especifica
QA estadística	Si contempla	Si contempla	No especifica	Si contempla	No especifica	No especifica
Presente en todo el proceso de desarrollo	No especifica	Si contempla	Si contempla	No especifica	Si contempla	Si contempla
Especificación de Roles	Si contempla (grupo SQA)	Si contempla	Si contempla	Si contempla (todo el equipo)	Si contempla (asignación de responsabilidades)	Si contempla (en otra etapa, en SS)
Plan de SQA	Si contempla (Plan de aseguramiento de la Calidad)	Si contempla (Plan de Calidad)	Si contempla (Plan de calidad y Plan de pruebas)	Si contempla (IQUE)	Si contempla (Plan del proceso)	Si contempla (Plan y gestión de los Procesos)
Especificación Documentación	Si contempla	Si contempla	Si contempla	No especifica	Si contempla (típicos productos de trabajo)	Si contempla
Independencia team calidad vs desarrolladores	No especifica	No especifica	Si contempla	No especifica	Si contempla	Si contempla
Mejora de procesos	Si contempla	No especifica	Si contempla	Si contempla (continuamente)	Si contempla	Si contempla (continuamente)

Tabla 1- *Tabla Comparativa entre defensores del Aseguramiento de la Calidad* (Fuente: Elaboración propia).

III. Pruebas del Software

Las *pruebas* constituyen el proceso de encontrar diferencias entre el comportamiento esperado, especificado por los modelos del sistema y el comportamiento observado del sistema (Bruegge y Dutoit, 2002). Las pruebas son un intento de simulación del sistema con respecto a los modelos del mismo. Su objetivo es diseñar situaciones que manifiesten los defectos que hay en el sistema y que revelen los problemas. Esta actividad es contraria a todas las actividades del desarrollo de software, pues están orientadas al quebrantamiento del sistema, es decir, lograr que el sistema falle (demostrar la presencia de defectos) de forma planeada. Méndez et al. (2007) lo resume de la siguiente forma: Una diferencia interesante entre las Pruebas y otras disciplinas del proceso de desarrollo de software es que esencialmente es una tarea que encuentra y pone de manifiesto las debilidades del producto de software. Esto puede lograrse descartando ideas preconcebidas sobre la corrección del software que se acaba de desarrollar y se supere cualquier conflicto de intereses que aparezcan cuando se descubran los errores presentes (Pressman, 2005).

Probar es una actividad realizada para la evaluación de la calidad en productos y para mejorarla, mediante la identificación de los defectos y problemas. Este concepto toma fuerza sobre el hecho de que la disciplina de pruebas debe estar formalizada y organizada mediante una estrategia dentro de un proyecto de desarrollo de software, dentro de un alcance definido, para que debidamente organizado se logren los objetivos planteados. Esta planificación de pruebas y actividades de diseño en sí mismos, constituyen un aporte útil para los diseñadores al poder exponer las debilidades potenciales (como descuidos o contradicciones de diseño, omisiones o ambigüedades en la documentación).

En líneas generales, las pruebas de un software giran alrededor de los casos de prueba, que según Leffingwell y Widrig (2006) son esenciales para todas las actividades de prueba porque:

- *Son la base para diseñar y ejecutar los procedimientos de pruebas.*
- *La profundidad de las pruebas es proporcional al número de casos de pruebas.*
- *El diseño y desarrollo, y los recursos necesarios son gobernados por los casos de pruebas requeridos.*

Bajo esta premisa, los *casos de prueba* (especificaciones de un conjunto de entradas, condiciones de ejecución e identificación de resultados los resultados esperados producto de las combinaciones de los dos primeros elementos), determinarán el ambiente necesario para ser ejecutado, podrá demostrar defectos (o su contraparte, casos de éxito) al final del proceso de pruebas apoyándose o no en herramientas de pruebas que lo gestionen formalmente, dirigido a diversos tipos de prueba y técnicas aplicadas según corresponda, en los diversos niveles de prueba que sean determinados por el equipo del proyecto/software objeto de prueba mediante procedimientos (instrucciones) de prueba claramente establecidos y documentados siguiendo una estrategia de pruebas plasmada en un plan de prueba.

Profundizando algunos conceptos, se tienen las siguientes definiciones tomadas en su mayoría de RUP (2005).

Caso de Prueba: Define un conjunto de entradas de prueba, condiciones de ejecución, y resultados esperados, identificados con el objetivo de evaluar algunos aspectos particulares de un elemento destino de la prueba.

Caso de Uso: Una especificación de un tipo de interacción con el sistema. Define un conjunto de instancias de guión de uso, donde cada instancia es una secuencia de acciones que lleva a cabo un sistema que producen un resultado observable de valor para un actor concreto.

Plan de prueba: Define los objetivos de las pruebas en el ámbito de la iteración (o el proyecto) los elementos de destino, el enfoque que se adopta, los recursos necesarios y los entregables que se deben generar.

Scripts de Pruebas: Es un guión o conjunto de instrucciones paso a paso que realiza una prueba, habilitando su ejecución. Permiten automatizar tareas creando pequeñas utilidades.

Suite de Pruebas: Es un conjunto de Scripts que se agrupan con el objetivo de gestionar y secuenciar la ejecución de las pruebas ordenadamente.

Datos de prueba: Define una recopilación de valores de entrada de prueba que se consumen durante la ejecución de una prueba, y los resultados esperados que sirven de referencia para objetivos comparativos durante la ejecución de una prueba.

Condiciones: Descripción de una condición que ejercerá durante esta prueba.

Resultados de las pruebas: Resume el análisis de uno o más registros de prueba y solicitudes de cambio, proporcionando una valoración relativamente detallada de la calidad de los elementos destino de la prueba y el estado del esfuerzo de prueba.

Resultados esperados: Corresponde a las expectativas que se tienen de los resultados.

Resultados obtenidos: Corresponde a lo realmente obtenido como resultado. Si lo que se espera es igual al obtenido, entonces el caso de prueba se considera exitoso.

Ambiente de pruebas: Respecto a la aplicación a probar, se puede incluir el hardware sobre el que se ejecuta, el software necesario para interconectarse con otros sistemas o requerimientos de la aplicación propiamente dicho, la base de datos necesaria para operar la aplicación, entre otros entes específicos.

Herramientas de pruebas: Aquellas que se pueden utilizar para aumentar la eficacia de las pruebas.

Tipos de Prueba: Tienen un objetivo específico y una técnica de soporte. Cada tipo de prueba se centra en la prueba de uno o varios atributos o características del destino de la prueba. SWEBOK (2004) plantea que los casos de prueba pueden ser diseñados para comprobar que las especificaciones funcionales se apliquen correctamente, que son comúnmente conocidas como pruebas de conformidad, pruebas de precisión o pruebas funcionales. Sin embargo, muchas otras propiedades no funcionales pueden ser probadas tales como el rendimiento, fiabilidad y facilidad de uso, entre otros. En todo caso, el objetivo de la prueba varía de acuerdo a la aplicación objeto de prueba. De esta forma algunos tipos de prueba se clasifican según la dimensión de calidad en Funcionales, No funcionales, de la siguiente forma:

- *Dimensión de funcionalidad: Prueba de función, prueba de seguridad y prueba de volumen.*
- *Dimensión de utilidad: Pruebas de utilización, prueba de integridad, prueba de usabilidad.*
- *Dimensión de fiabilidad: Prueba de estructura y de tensión.*
- *Dimensión de rendimiento: Prueba de puntos de referencia, Prueba de contienda, Prueba de carga y perfil de rendimiento, prueba back to back.*
- *Dimensión de Capacidad de soporte: Prueba de configuración, prueba de instalación, pruebas de recuperación.*
- *Otras: Pruebas alpha y beta, pruebas de regresión,*

Niveles de Pruebas: Las pruebas se aplican a diferentes tipos de destinos, en fases o niveles diferentes de esfuerzo de trabajo. Estos niveles suelen distinguirlos los roles que están más capacitados para diseñar y dirigir las pruebas. Los niveles de prueba

son conocidos como: Prueba de desarrollador (unitarias), Pruebas integrales, Pruebas certificadas y Pruebas del sistema (aceptación del usuario).

Técnicas de Pruebas: Normalmente las técnicas de pruebas se clasifican en Caja blanca (que proporcionan información de cómo el software fue diseñada y codificada) y Caja negra (que se centran solo en el comportamiento de entrada y salida del software). Sin embargo, SWEBOK (2004) plantea que es difícil encontrar una base homogénea para clasificar las técnicas de pruebas, pero plantea una clasificación que se basa en el cómo se generan los casos de prueba y se presenta a continuación:

- a) *Intuición y experiencia de los ingenieros de software.*
 - a. *Pruebas Ad hoc*
 - b. *Pruebas exploratorias*
- b) *Las especificaciones del software*
 - a. *Partición de equivalencia*
 - b. *Análisis del valor frontera*
 - c. *Tabla de decisión*
 - d. *Máquinas basadas en estados finitos*
 - e. *Pruebas desde especificaciones formales*
 - f. *Pruebas aleatorias*
- c) *La estructura del código*
 - a. *Criterio basado en el flujo de control*
 - b. *Criterio basado en el flujo de datos*
 - c. *Modelos de referencia para pruebas basadas en código*
- d) *Técnicas basadas en fallos (reales o artificiales-simuladas)*
 - a. *Adivinar el error*
 - b. *Pruebas de mutación*
- e) *La naturaleza de la aplicación*
 - a. *Pruebas orientadas a objetos*
 - b. *Pruebas basados en componentes*
 - c. *Pruebas de GUI*
 - d. *Pruebas de programas concurrentes*
 - e. *Pruebas de protocolo de conformidad*
 - f. *Pruebas de sistemas en tiempo real*
 - g. *Pruebas de sistemas de seguridad crítica*

IV. Disciplina de Pruebas

Los modelos de ciclo de vida del software hacen que las actividades para su desarrollo y sus dependencias sean visibles y manejables. Una *disciplina* es una colección de actividades correspondidas que a su vez se relacionan con un “motivo de preocupación importante” dentro del proyecto total donde estén enmarcadas (RUP, 2005). Ésta colección o agrupación es frecuentemente usada por RUP, quien concentra actividades en disciplinas como una forma de ayudar a entender el proyecto

desde una perspectiva ‘tradicional’, es decir, una secuencia de actividades ordenadas que deben cumplirse.

La disciplina de pruebas en el software, está fuertemente ligada a la calidad y los esfuerzos que se puedan ejecutar para alcanzarla, garantizando que los procesos y productos resultarán con los niveles esperados de calidad. Su importancia radica en que un acercamiento continuo a la calidad, iniciada temprano en el ciclo de vida del software, puede aminorar los costos de terminar y mantener el software significativamente. Esto reduce altamente el riesgo asociado de implantar software de mala calidad (RUP, 2005).

Kruchten (2000) afirma que la disciplina de pruebas en un proceso iterativo que ocurre en todas las fases del ciclo de vida, permitiendo tener una retroalimentación temprana sobre la calidad del producto que será usada para mejorar el producto tanto en su diseño como en su construcción. Esto evidencia el hecho de que probar no es solo una actividad final del ciclo de vida que está destinada a aprobar o rechazar el sistema desarrollado, sino que se debe abordar desde la fase de inicio del desarrollo de software. Por su parte, involucrando el capital humano que participa en la disciplina de pruebas, Dustin (2003) apoya el hecho de iniciar pruebas desde etapas tempranas incluyendo a los probadores quienes deben participar desde el comienzo de un ciclo de vida del proyecto, para que puedan comprender exactamente lo que están probando y puedan trabajar en unión con los stakeholders² para crear requerimientos comprobables.

La disciplina de pruebas actúa como abastecedor de servicios a las otras disciplinas en muchos aspectos. Las pruebas se centran sobre todo en la evaluación o evaluar la calidad del producto, que es lograda con estas prácticas base (RUP, 2005):

- *Encontrar y documentar los defectos en calidad del software.*
- *Aconsejar sobre la calidad percibida del software.*
- *Validar y probar las asunciones hechas en especificaciones del diseño y del requerimiento a través de demostración concreta.*
- *Validar que el producto de software trabaja según lo diseñado.*
- *Validar que los requerimientos están implementados apropiadamente.*

² **Stakeholders:** Cualquier persona o representante de una organización quien tiene un interés creado en los resultados de un proyecto o cuya opinión debe ser complacida. (Kruchten, 2000).

Diferentes metodologías y tendencias en el desarrollo de software (RUP, XP (eXtream Programing), MFS, Eclipse EFP, ITIL, entre otros) hacen uso de la disciplina de pruebas identificando actividades y roles entre otros, por lo que se evidencia la gran importancia del tema en el proceso de desarrollo de software.

Alguna de las tendencias más actualizadas en pruebas se conoce como Desarrollo dirigido por pruebas *TDD* (Test Driven Developmet o Test Driven Design, por sus siglas en ingles), técnica de desarrollo de software que consiste en dos prácticas: a) Escribir los casos de pruebas (para pruebas unitarias) primero antes que el código que implementará dicho caso de prueba con el fin de que pase con éxito y b) Refactorizar (mejorar el diseño del código, elevando la flexibilidad y tolerancia al cambio, entregando mas rápido y con menos depuración) incrementalmente. Esta es una de las prácticas clave en la Programación extrema, pero no necesariamente se implementa junto a ella (Astels, 2003) y su objetivo es “código limpio que funciona”. Los demás tipos de pruebas, en sus diferentes niveles, no se reemplazan, solo se complementan.

Back (2003), considerado uno de los precursores de TDD, dice que éste gira en torno a dos claves: 1) Nunca escribir una sola línea de código a menos que haya un defecto en las pruebas automatizadas. 2) Eliminar la duplicación. Bajo este concepto de ideas propone el siguiente ciclo de desarrollo basado en TDD.

1. Escribir la prueba
2. Escribir el código
3. Ejecutar las pruebas automatizadas
4. Refactorización
5. Repetición

Ambler (2007), establece que sin el uso de herramientas automatizadas, implementar TDD es prácticamente imposible. Esta se constituye como una de sus principales desventajas, pues no es aplicable a todos los sistemas o para aquellos que usen pruebas unitarias manuales. Sin embargo, aunque es una especificación técnica, un valioso efecto secundario es que provee de resultados significativamente mejores que los que lo hacen pruebas de código de las técnicas tradicionales.

Un enfoque interesante en la disciplina de pruebas, es presentado en el Perfil de Pruebas de UML (UML Testing Profile) que define un lenguaje para el diseñar,

visualizar, especificar, analizar, construir y documentar los artefactos en las pruebas de sistemas en varios dominios (OMG, 2005). Está basado en la especificación UML 2.0, uniendo el meta-modelo UML y reusando la sintaxis UML, extendiendo a UML con conceptos específicos en pruebas agrupados en 4 categorías: Arquitectura de prueba (que contiene los elementos que participan en una prueba y sus relaciones, conceptos de estructura y configuración de las pruebas), Datos de prueba (estructuras y significado de los valores que serán procesados en una prueba), Comportamiento de pruebas (que se ocupa de las observaciones y actividades durante una prueba) y tiempo (conjunto de conceptos relativos al tiempo con el fin de cuantificar las ejecuciones de pruebas).

Este perfil está enfocado a una particular tecnología de pruebas llamada “Pruebas funcionales de Caja negra”, permitiendo la captura de toda la información necesaria para el enfoque de caja negra y así evaluar la correcta implementación del sistema. El perfil de pruebas UML se destina a apoyar de una manera efectiva, eficiente, en la medida posible, las pruebas automatizadas del sistema en función de las implementaciones de los modelos computacionales de UML (Cavarra, 2002).

Este perfil constituye una iniciativa importante en la formalidad de la disciplina de pruebas dentro del proceso de desarrollo de software, bajo el enfoque de la validación de sistemas de tipo caja negra (solo entradas/salidas), pues detalla las notaciones, documentación y pasos que deben cumplirse al momento de probar sistemas. Algunos tecnicismos son aplicables a proyectos basados en la notación UML y apoyan la automatización de pruebas, además de que puede acoplarse con un marco de trabajo como JUnit³ usado dentro de la popular programación extrema (XP). Sin embargo, es importante notar el amplio alcance en los diferentes niveles de pruebas, puesto que se pueden especificar casos de prueba a nivel de unidad, integración, sistema y a nivel de la organización.

Otra tendencia interesante es Microsoft Solutions Framework (MSF por sus siglas en inglés) que es una flexible e interrelacionada serie de conceptos, modelos y mejores prácticas de uso que controlan la planificación, el desarrollo y la gestión de

³ JUnit: Marco de trabajo para que los desarrolladores realicen pruebas unitarias en Java inicialmente, pues ya se ha traducido a varios lenguajes.

proyectos tecnológicos, centrándose en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas (Microsoft, 2003). MSF es una serie de modelos y disciplina que puede adaptarse a cualquier proyecto de tecnología de información. Uno de los modelos es el “Modelo de procesos” el cual es separado en cinco principales fases, como se muestra en el Figura 9, y cuya función principal es establecer el orden por medio del cual, las diversas actividades del proyecto serán ejecutadas.

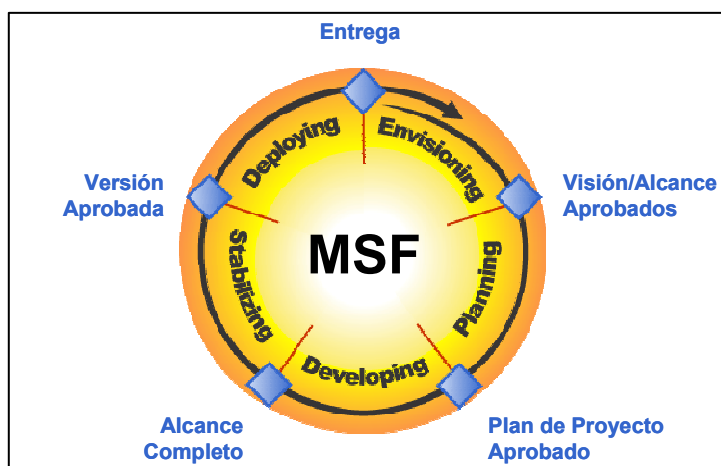


Figura 9 - Modelo de Proceso de MSF para el desarrollo de aplicaciones, Fuente: (Microsoft, 2003)

En todas las fases de este modelo de proceso se introduce la disciplina de pruebas, en menor o mayor proporción, gracias al “Rol de prueba” que se encarga de enfocar la disciplina de pruebas de acuerdo a las necesidades del proyecto en cada fase. En la *fase de visión*, el “rol de prueba” se enfoca en identificar las estrategias de pruebas, los criterios de aceptación y manejar las implicaciones que se derivan de la disciplina. En la *fase de planeación*, se realiza una evaluación del diseño y se determinan los requerimientos de pruebas y el plan de pruebas. En la *fase de desarrollo*, se dan los inicios de algunos entregables de pruebas como las especificaciones de pruebas y los casos de prueba; aquí, la aplicación ya está lista para pruebas externas y para su estabilización. El “Rol de pruebas” se enfoca en pruebas funcionales, identificación de defectos, documentación de pruebas y actualización del plan de pruebas. En la *fase de estabilización* se conducen las pruebas sobre la solución, las pruebas de esta etapa enfatizan el uso y operación bajo condiciones realistas. El equipo se enfoca en priorizar y resolver errores y preparar la solución para el lanzamiento. Es común que se reporten los errores del sistema rápidamente para que los desarrolladores los corrijan; se usan indicadores estadísticos conocidos como “bug convergente” y “zero-bug bounce” que ayudan al equipo del proyecto a

determinar cuando la aplicación está suficientemente estable. Igualmente, se usan técnicas como prueba completa pre-producción (se enfoca en preparar un release piloto) y prueba completa de aceptación del usuario (donde se determina que el sistema está habilitado y cumple satisfactoriamente con las necesidades del usuario y del negocio). En esta fase se especifican los siguientes entregables: Resultado de pruebas y herramientas de pruebas, enfocándose el “Rol de pruebas” en probar, reportar errores y estatus y en la configuración de pruebas. En la *fase de despliegue* el “Rol de pruebas” se enfoca en hacer pruebas de desempeño sobre la aplicación.

Dentro de las tendencias del código abierto, puede detallarse la comunidad Eclipse⁴ cuyos proyectos se centran en la construcción de una plataforma de desarrollo compuesta por frameworks extensibles, herramientas y módulos de ejecución para la construcción, desarrollo y gestión de software en todo el ciclo de vida (Eclipse, 2008). Dentro de sus múltiples proyectos, esta el Proyecto de Tecnología, que incluye al Eclipse Process Framework (EPF, por sus siglas en inglés), el cual tiene por objeto elaborar un marco de trabajo para el proceso de ingeniería de software personalizado, proveyendo herramientas, un metamodelo unificado y contenidos que puedan ser usados como base para apoyar una amplia variedad de tipos de proyectos y estilos de desarrollo, dirigido principalmente al equipo del proyecto y al equipo de gerencia del proyecto. El primer entregable de este proyecto, está constituido por OpenUP que es un proceso inspirado en la tendencia ágil, cuyas raíces son el Proceso Unificado, cuyo propósito es ser un release simple de una aplicación de software construido por un grupo relativamente pequeño con fácil acceso para los stakeholders. El objetivo principal en el desarrollo de OpenUP era crear un ligero proceso que pueda ampliarse fácilmente para satisfacer las necesidades de los proyectos y organizaciones que necesitan más procesos (Eclipse, 2008).

OpenUP incorpora un número de prácticas ágiles dentro del contexto de un interactivo e incremental ciclo de vida del desarrollo de software. Por ser un ciclo de vida, se distinguen cuatro fases, como se muestra en la Figura 10, con hitos asociados y una plantilla de iteración para cada una de las fases que describen las actividades clave.

⁴ La comunidad Eclipse consta de individuos y organizaciones provenientes de una sección de la industria del software.

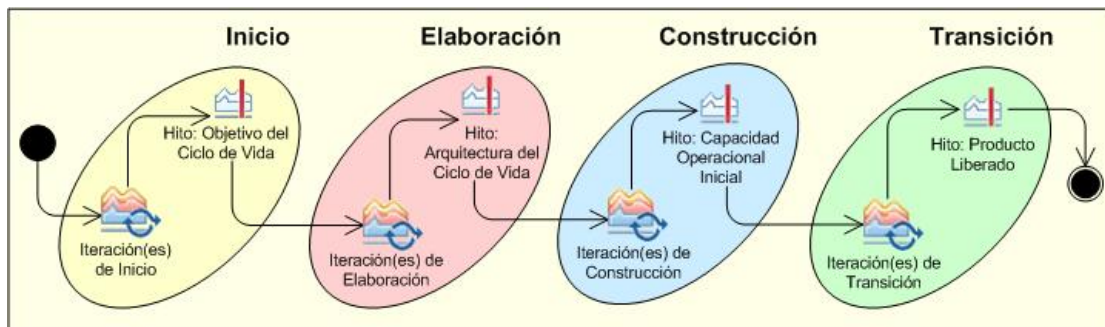


Figura 10- Fases del ciclo de vida del desarrollo de software, Fuente: (Eclipse, 2008).

Dentro de este ciclo de vida, OpenUP ubica dichas tareas que se realizan simultáneamente a través de varias disciplinas, como una forma de organizar simple y efectivamente los contenidos, entre las que se encuentra la Disciplina de Pruebas, la cual explica como proveer feedback acerca de la maduración del sistema mediante el diseño, implementación, ejecución y evaluación de pruebas, aplicando la estrategia de “probar temprano y probar frecuente” con el propósito de menguar riesgos lo más temprano posible en el ciclo de vida. Esta disciplina está constituida por tres tareas (a) crear casos de prueba, (b) implementar scripts de pruebas y (c) ejecución de pruebas, y siete elementos de orientación, un rol principal llamado “probador”, que se encarga de las actividades básicas del esfuerzo en pruebas, dentro de las que están: identificación, definición, ejecución y realización de las pruebas necesarias, así como el registro de los resultados de las pruebas y el análisis de los resultados, y algunos productos asociados al trabajo inherentes a la disciplina, como los casos de prueba, log de pruebas y scripts de pruebas.

Resulta interesante indagar el propósito de EPF y relacionarlo con la investigación en curso, en cuanto a la intención de poder personalizar mediante marcos de trabajo, herramientas y estrategias que proporcionen soluciones adaptadas a diferentes tipos de proyectos de software, tomando en cuenta sus propias limitaciones y alcances, escalable a sus propios requerimientos. El modelo de referencia que se propondrá, deberá tener este propósito de poder ser implementado en numerosas organizaciones, buscando adaptarlo a sus propias necesidades.

A la par del proceso unificado, se encuentra el proceso de ingeniería de software que proporciona un acercamiento disciplinado para asignar tareas y responsabilidades dentro de una organización de desarrollo, llamado Rational Unified

Process (RUP, por sus siglas en ingles), constituido por dos dimensiones: Eje Horizontal (dinámico y representa el tiempo y los aspectos del ciclo de vida del proceso) y el Eje Vertical (la descripción del dinamismo en términos de componentes de proceso, disciplinas, actividades, workflows, artefactos, y de roles), como se muestra en la Figura 11 (RUP, 2005).

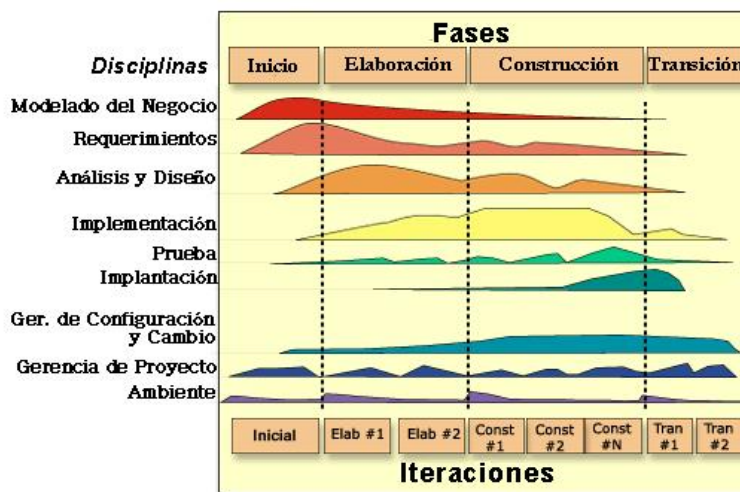


Figura 11 - Vista general del Proceso Unificado de Rational, Fuente: (RUP, 2005).

Como complemento de lo ya expresado anteriormente, RUP define claramente los roles, actividades e hitos importantes para la Disciplina de pruebas, que como puede observarse en la figura anterior, está presente en gran parte de la vida del proyecto y se interrelaciona con las demás disciplinas. La Disciplina de pruebas según RUP, define 4 roles especiales que son: Gerente de pruebas, analista de pruebas, diseñador de pruebas y el probador, quienes realizan una serie de actividades específicas de acuerdo al rol, entre las que destacan: acordar la misión, identificar los motivadores de pruebas, obtener compromiso para probar, estimar y mejorar el esfuerzo de pruebas, identificar requerimientos e ideas de pruebas, determinar los resultados de las pruebas, definir el enfoque de pruebas, definir las configuraciones de los ambientes de pruebas, identificar los mecanismos para probar, definir los elementos a probar, implementar las pruebas con la respectiva suite de pruebas, ejecutar las pruebas y analizar las fallas de pruebas. Todas estas actividades se apoyan en los artefactos (entregables) donde intervienen los roles, siendo alguno de ellos: el plan de pruebas, resumen de la evaluación de pruebas, suite de pruebas, log de pruebas, lista de ideas de pruebas, casos de prueba, datos de prueba, resultados de pruebas, configuración del ambiente de pruebas, arquitectura de automatización de pruebas.

RUP es ampliamente utilizado por más y más proyectos de desarrollo de software, para dirigir el esfuerzo en pruebas, apoyándose en herramientas que permitan la automatización de pruebas y su gestión.

Por su parte, ITIL en su fase de Transición del Servicio (ST-Service Transition, por sus siglas en inglés) provee orientación sobre la administración del cambio del portafolio de servicios y a través del ciclo de vida de los servicios. Aquí se incluyen algunos tópicos importantes que ubican algunas áreas de importancia, en la cual se incluye el Servicio de pruebas y validación, que junto con el soporte temprano (Early Life Support⁵, por sus siglas en inglés) garantizan que los servicios son aptos para el propósito y uso al que se destinará el producto, de conformidad con los requisitos de servicio y de las limitaciones especificadas (Lacy y Macfarlane, 2007).

Este proceso de servicio de pruebas y validación, tiene el propósito de asegurar que las versiones desarrolladas y los servicios resultantes satisfagan las expectativas de los clientes, y a su vez, verificar que las operaciones de TI son capaces de apoyar el nuevo servicio (Kempter, 2008). Identifica los siguientes roles: gerente de pruebas (quien es el propietario del proceso), gerente de cumplimiento, cliente, gerente de seguridad TI, gerente de versiones, gerente de nivel de servicio y el usuario. Algunos sub-procesos (actividades) dentro de esta rama, son: definición del modelo de pruebas, servicio de validación del diseño, versión de componente de adquisición, versión de pruebas y el servicio de aceptación de pruebas.

Con todo lo anterior, se destaca la importancia que tiene la Disciplina de pruebas para crear software de calidad, calidad que no se logra instantáneamente, sino que dependerá en gran medida del proceso de aseguramiento de la calidad que transitó a la par con la creación de este producto.

En el mismo marco de ideas, una realidad latente es que hay varios factores que han hecho de las Pruebas de software y de sistemas una necesidad: por un lado la creciente complejidad, heterogeneidad y variabilidad de las tecnologías, que

⁵ Soporte provisto para un Servicio TI nuevo o cambiado para un periodo de tiempo, posterior a su Entrega. (ITIL® Glossary v01, 2006).

incrementa el riesgo. Por otro lado la presión del *time to market* y también la madurez del mercado (García y Encuera, 2007). He aquí otro gran paso para dar la importancia que merece la disciplina de pruebas en las empresas desarrolladoras de software, especialmente las PYMEs, que en su interés de ganar presencia en el mercado nacional, deben apalancarse en calidad de los servicios prestados.

Esta importancia, también es indicada por Grimán et al. (2003), al afirmar que una disciplina importante en el proceso de desarrollo de los Sistemas de Software es la relativa a Pruebas. Esta disciplina, generalmente no se implementa de forma organizada y sistemática. Mecanismos para garantizar esta formalidad dentro de las empresas desarrolladoras de software, es lo que se plantea en la investigación en progreso y se constituye como la evidencia de concientizar esta cultura de pruebas en las PYMEs Venezolanas. Parte de las razones de esta evidencia, se mostraron en la sección “Marco Organizacional” del presente Capítulo, al contextualizar las PYMEs desarrolladoras de software venezolanas.

La siguiente sección hace un breve recorrido sobre los principales modelos de calidad del software que existen, profundizando en el modelo elegido para esta investigación.

V. Modelos de Calidad

Uno de los principales problemas a los que se enfrenta la calidad del software es el siguiente: ¿Es realmente posible encontrar un conjunto de propiedades en un software que sea una indicación de su calidad?. Para dar respuesta a esta pregunta aparecen los Modelos de Calidad, que no son más que las directrices o punto de referencia para imitarlo o reproducirlo; es por ello que para los efectos de esta investigación, se homologa el término modelo de referencia a modelo de calidad, pues el segundo finalmente es un modelo de referencia a seguir.

Como se ha dicho en secciones anteriores, la calidad puede describirse desde diferentes perspectivas y en el desarrollo del software hay dos especiales y tradicionalmente importantes: la calidad del producto en sí y la calidad del proceso para obtenerlo con un enfoque sistémico. Contar con estándares y/o modelos de

calidad para garantizar calidad en los productos de software, es de suma importancia para las organizaciones que quieran diferenciarse dentro de su entorno competitivo.

En lo que se refiere a calidad del producto la norma ISO/IEC 25000 proporciona una guía para el uso de las nuevas series de estándares internacionales, llamados Requisitos y Evaluación de Calidad de Productos de Software (SQuaRE), las cuales constituyen una serie de normas basadas en la ISO 9126 y en la ISO 14598 (Evaluación del Software), siendo su objetivo principal guiar el desarrollo de los productos de software con la especificación y evaluación de requisitos de calidad (Piatinni et al., 2007).

La norma ISO 9126 propone un modelo de calidad que se divide en tres vistas: interior, exterior y en uso que están compuestas por características, que se dividen en subcaracterísticas y que estas a su vez se componen de atributos. Los atributos obtienen sus valores tras realizar mediciones sobre el software que pueden usarse como una lista de comprobación de aspectos relacionados con la calidad. A continuación se presenta un cuadro resumen con los conceptos propuestos por éste estándar de calidad en sus perspectivas (Ver Tabla 2).

Para encontrar la calidad con la perspectiva del proceso, se pueden nombrar algunos modelos como lo son: CMM, PSP, Bootstrap, Spice, ISO 9000, IDEAL. De igual manera para la perspectiva del personal (la cual no se puede obviar según Voas, 1998) están los modelos de PSP, TSP, P-CMM. En virtud de esta investigación, se profundizará en el modelo de calidad del producto de software puesto que se evaluarán herramientas de pruebas presentes en el mercado constituidas como producto de software (concluido).

Cabe destacar que el MOdelo Sistémico de CALidad MOSCA, toma en cuenta las tres perspectivas de calidad del software que son Producto, Proceso y Personal, como se puede ver en la sección de ANEXOS, denotado como Anexo I. MOSCA para cada una de las perspectivas está constituido por cuatro 4 niveles que se detallan a continuación: (Mendoza *et al.*, 2005).

- **Nivel 0 Dimensiones:** Abarcadas en cada perspectiva. Estas son el aspecto contextual e interno del producto, del proceso y de lo humano.

Vista	Características	Capacidad del Producto de Software para...	Sub-característica	Capacidad del Producto de Software para...
Interna y externa	FUNCIONALIDAD	<i>Proporcionar funciones que satisfacen necesidades declaradas e implícitas cuando se usa bajo condiciones específicas</i>	Adecuación	Proveer funciones apropiadas según tareas y objetivos
			Exactitud	Proveer resultados o efectos correctos o acordados
			Interoperabilidad	Interactuar con uno o más sistemas especificados
			Seguridad de acceso	Proteger datos e información de personas/sistemas no autorizados
			Cumplimiento Funcional	Adherirse a normas, convenciones o regulaciones realcionadas con FUN.
	FIABILIDAD	<i>Mantener un nivel especificado de prestaciones bajo condiciones específicas</i>	Madurez	Evitar fallar como resultado de fallas en el software
			Tolerancia a fallos	Mantener un nivel especificado ante fallas del SW o interfaces
			Capacidad de recuperación	Restablecer un nivel de prestaciones y recuperar datos afectados
			Cumplimiento Fiabilidad	Adherirse a normas, convenciones o regulaciones realcionadas con FIA.
	USABILIDAD	<i>Ser entendido, aprendido, usado y atractivo cuando se usa bajo condiciones específicas</i>	Ser entendido	Entender si es adecuado y cómo puede ser usado particularmente
			Ser aprendido	Permitir el aprendizaje fácil por el usuario
			Ser operado	Permitir al usuario operarlo y controlarlo
			Interfaz gráfica	Ser atractivo al usuario
			Cumplimiento Usabilidad	Adherirse a normas, convenciones o regulaciones realcionadas con USA.
	EFICIENCIA	<i>Proporcionar prestaciones apropiadas, relativas a cantidad de recursos usados, bajo condiciones específicas</i>	Comportamiento temporal	Proporcionar tiempos de respuesta, proceso y potencia apropiados
			Utilización de recursos	Usar cantidades y tipos de recursos adecuados bajo ciertas condiciones
			Cumplimiento Eficiencia	Adherirse a normas, convenciones o regulaciones realcionadas con EFI.
	MANTENIBILIDAD	<i>Ser modificado (correcciones, mejoras, adaptaciones a cambios del entorno, requisitos y especificaciones funcionales</i>	Capacidad de análisis	Diagnosticarle deficiencias o causas de fallos o identificar modificaciones
			Capacidad de cambio	Permitir que se implemente una determinada modificación
			Estabilidad	Evitar efectos inesperados por modificaciones del SW
			Capacidad de ser probado	Permitir que el SW modificado sea validado.
	PORTABILIDAD	<i>Ser transferido de un entorno a otro</i>	Cumplimiento Manetenibi	Adherirse a normas, convenciones o regulaciones realcionadas con MAN.
			Adaptabilidad	Adaptarse a diferentes entornos especificados sin aplicar muchos cambios
			Instalabilidad	Instalarse en un entorno espeificado
			Coexistencia	Coexistir con otro SW independiente en entorno con recursos comunes
			Capacidad de reemplazo	Usarlo en lugar de otro producto SW con el mismo propósito y entorno
En uso	Efectividad	<i>Permitir a los usuarios alcanzar objetivos con exactitud y completitud</i>		
	Productividad	<i>Permitir a los usuarios gastar la cantidad apropiada de recursos en relación a la efectividad obtenida.</i>		
	Seguridad de uso	<i>Alcanzar niveles aceptables de riesgo en personas, negocio, SW, propiedades o medio ambiente</i>		
	Satisfacción	<i>Cumplir con las expectativas de los usuarios en un contexto determinado</i>		

Tabla 2- Modelo de Calidad según ISO 9126. Fuente: Elaboración propia

- **Nivel 1 Categorías:** Definidas para cada perspectiva. **Producto:** funcionalidad (FUN), fiabilidad (FIA), usabilidad (USA), eficiencia (EFI), mantenibilidad (MAB) y portabilidad (POR); estas están basadas en las normas internacionales ISO 9126 que se establecen como obligatorias para garantizar la calidad del producto de software. **Proceso:** cliente –proveedor (CUS), ingeniería (ENG), soporte (SUP), gestión (MAN) y organizacional (ORG). **Humana:** individual (IND), equipo (EQU) y entorno empresarial (ENT).
- **Nivel 2 Características:** Asociadas a cada categoría, las cuales definen las áreas claves a satisfacer para lograr, asegurar y controlar la calidad en las tres perspectivas.

Son 56 características asociadas al producto, 27 al proceso de desarrollo y 15 a la parte humana.

- **Nivel 3 Métricas:** Utilizadas para medir la calidad sistémica. Son propuestas en total 715 métricas.

Una vez definidos los niveles, se procede a evaluar a través del modelo, la calidad del software usando un algoritmo ampliamente probado propuesto por Mendoza *et al.* (2005). Finalmente el algoritmo detalla que el nivel de calidad del producto de software se relacionan con las categorías satisfechas luego de la evaluación de acuerdo a la Tabla 3, siendo que si la categoría Funcionalidad no es satisfecha, el algoritmo finaliza y la calidad del producto será nula.

Funcionalidad	Segunda categoría evaluada	Tercera categoría evaluada	Nivel de calidad del producto de software
<i>Satisfecha</i>	<i>No satisfecha</i>	<i>No satisfecha</i>	Básico
<i>Satisfecha</i>	<i>Satisfecha</i>	<i>No satisfecha</i>	Intermedio
<i>Satisfecha</i>	<i>No satisfecha</i>	<i>Satisfecha</i>	Intermedio
<i>Satisfecha</i>	<i>Satisfecha</i>	<i>Satisfecha</i>	Avanzado

Tabla 3 - Nivel de calidad del producto con respecto a las categorías satisfechas para el producto.

FUENTE: Mendoza et al. (2005).

Esta investigación se basa en los conceptos antes desarrollados y su enfoque está dado al modelo de calidad MOSCA por ser el más idóneo al momento de evaluar herramientas de pruebas, demostrado por múltiples aplicaciones del modelo.

En la siguiente sección, se abordarán las bases éticas y legales sobre la que se sustenta esta investigación.

VI. Herramientas de Pruebas

La calidad en el software que se espera como un producto final, bien puede afianzarse con la disciplina de pruebas dentro del proceso de desarrollo de software, que a su vez se apoya en herramientas tecnológicas diseñadas con propósitos especiales para cubrir estas necesidades. Las *herramientas de pruebas* permiten en gran medida coordinar, dirigir y abastecer los diversos entregables, actividades y roles que surgen de la aplicabilidad de la disciplina de pruebas, y que por su envergadura imposibilitan los procesos manuales para la obtención de los resultados esperados. Braude (2003) expresa que el gran volumen de las pruebas suelen requerir herramientas de prueba automatizadas y se sugiere que al menos el 75% de las

pruebas sean automatizadas y el resto manual. Las herramientas de automatización ejecutan únicamente scripts diseñados para testear un requisito o funcionalidad específicos y no poseen la habilidad de “toma de decisiones” ni grabación de discrepancias no incluidas en el script. Es por ello que es necesario realizar pruebas manuales a lo largo de toda la aplicación, antes de realizar la automatización de actividades.

En el mercado, existen numerosas herramientas de pruebas con propósitos tan diversos y en algunos casos, tan distantes, que tener criterios válidos para su selección se vuelve una actividad compleja. En todo caso, siempre es necesario definir el propósito para lo cual se requiere y el alcance que las mismas tendrán en el proceso de desarrollo de software (dirigido a su implementación) para intentar hacer la mejor elección. Estas herramientas pueden ser usadas para la planificación de pruebas, diseño de pruebas funcionales y no funcionales, automatización de pruebas, entre otras aplicaciones, lo cual hacen que el tema de las pruebas se vuelva tan especializado y a la vez postergado; para ambos casos se necesitan medidas de acción oportunas. RUP (2005) plantea esta realidad: *“Existen una serie de herramientas de automatización, pero es poco probable que una sola herramienta pueda automatizar todas las tareas de prueba. La mayoría de las herramientas se centran en una tarea o grupo de tareas específicas, aunque algunas sólo tratan un aspecto de la tarea”*. Las pruebas automatizadas son efectivas en entornos donde los cambios son frecuentes o en aplicaciones en las que se esperan builds y releases críticos.

Por su parte, SWEBOK (2004) clasifica las herramientas de pruebas en estos grandes grupos:

- a) *Generadores de prueba, para ayudar en el desarrollo de casos de prueba.*
- b) *Marcos de trabajo de ejecución de pruebas que permiten la ejecución de los casos de prueba en un ambiente controlado donde se observa el comportamiento de la aplicación objeto de prueba.*
- c) *Herramientas de evaluación de pruebas, herramientas de apoyo a la evaluación de los resultados de ejecución de pruebas, determinando si el comportamiento observado se ajusta al comportamiento esperado.*
- d) *Herramientas de gestión de pruebas las cuales proporcionan apoyo para todos los aspectos del proceso de pruebas del software.*
- e) *Herramientas de análisis del rendimiento, utilizadas para medir y analizar el performance del software objeto de pruebas, la cual se corresponde con una forma especializada de probar donde la meta es evaluar el comportamiento del desempeño más que el funcional.*

Braude (2003) especifica las siguientes capacidades que deben encontrarse en las herramientas automatizadas de pruebas de sistemas, las cuales se tomarán en cuenta para el modelo de referencia propuesto:

1. *Registrar acciones de pantalla (ratón y teclado) para permitir reproducción repetida*
2. *Correr varias veces los textos de prueba*
3. *Activar registro de resultados de prueba*
4. *Registrar tiempo de ejecución*
5. *Registrar errores de la corrida (ejecución)*
6. *Crear y administrar pruebas de regresión*
7. *Generar informes de pruebas*
8. *Generar datos de prueba*
9. *Registrar uso de memoria*
10. *Administrar casos de pruebas*
11. *Analizar la cobertura*

Puede que la selección de dichas herramientas sea compleja, pero ya existen las herramientas; esto se constituye como un gran paso para crear la cultura de la Disciplina de pruebas, apoyada en herramientas. Sin embargo, seleccionarlas y evitar que entren en desuso, es parte de la cultura en progreso que debe afianzarse en las PYMEs, a través de mecanismos innovadores y de comunicación exitosos entre quienes asumen los roles dentro de la Disciplina de Pruebas.

2.4 Bases éticas y legales

En una sociedad de la información tan avanzada como la nuestra, recalcar los aspectos éticos que den base a la investigación resulta obligado, entendiéndose la ética en informática como un conjunto de normas que guían al personal involucrado en la informática y a las personas que se relacionan con el mundo de las nuevas tecnologías (Carretero, 2006).

Esta investigación dentro del marco ético respeta la propiedad intelectual y los derechos de los autores aquí citados, atribuyéndoles las ideas propias creadas por un individuo y reconociendo las contribuciones a la ciencia que han realizado los mismos. Cuando se haga uso de alguna conclusión, método, estrategia o herramienta de prueba, será claramente identificado el autor de dicho aporte, ya sea individuo, organismo o empresa, y se usará con propósitos netamente académicos.

Dado que el trabajo de investigación está enmarcado dentro de un proyecto del FONACIT⁶, el uso y derecho sobre el conocimiento generado está regido por el marco legal que el mismo FONACIT establece, entre las que destacan la Ley Orgánica de Ciencia, Tecnología e Innovación, el Decreto del uso del software libre en Administración Pública y demás regulaciones definidas por este.

La presente investigación asume que el marco regulatorio de las PYMEs en Venezuela, viene dado por el Código de Comercio (en cuanto a la actividad de creación y funcionamiento de una PYME) y el Decreto con fuerza de Ley para la promoción y desarrollo de la pequeña y mediana industria del 2001, según Bastidas (2003).

⁶ Ver sección de Alcances y Limitaciones.

CAPÍTULO III: MARCO METODOLÓGICO

Este capítulo tiene como objetivo principal especificar el marco de trabajo metodológico utilizado para la realización de esta investigación. Primeramente se ubica el proyecto dentro de un tipo específico de investigación. Seguidamente, se identifica el diseño de la investigación, basada principalmente en el aporte de Pérez et al. (2004) adaptado en el Laboratorio de Investigación en Sistemas de Información (LISI) y que rige los aspectos metodológicos de la investigación. Posteriormente, dentro del mismo diseño se presenta el enfoque basado en el programa Objetivos, Preguntas y Métricas (GQM, por sus siglas en inglés) que es utilizado para la elaboración de los indicadores que se usarán para medir el modelo de referencia propuesto para la selección de herramientas que apoyen la disciplina de pruebas.

3.1 Tipo de Investigación

La presente investigación, según su naturaleza y propósito está enmarcada dentro del tipo de investigación aplicada, la cual desarrolla productos para satisfacer necesidades, implanta acciones para mejorar o evalúa la adecuación de medios y fines (Yáber y Valerino, 2006). Con el presente proyecto se investiga, elabora y desarrolla una propuesta de un modelo de referencia/modelo de calidad (con su respectivo fundamento teórico, ejecución y evaluación) para PYMEs desarrolladoras de software, con el fin de solventar las necesidades de las mismas en el ámbito de la Disciplina de pruebas, es decir, ofrecerles un método o procedimiento para la selección de herramientas de pruebas con las que puedan apoyar sus procesos dentro de la industria del software en Venezuela.

Bajo este concepto de ideas, la sub-clasificación de la investigación aplicada en *investigación-acción*, es la que mejor se acopla con el propósito del presente proyecto de investigación, ya que la misma tiene como propósito investigar la condición actual y condición deseada de un grupo, equipo, proyectos, programas, unidades o la organización en su conjunto, para luego realizar intervenciones que conduzcan al mejoramiento de su gestión para lograr la condición deseada (Valerino et.al, 2006).

Mediante una metodología de desarrollo, se presentará un profundo análisis sobre el tema en estudio y por ende, las conclusiones sobre la viabilidad y realización del proyecto, ejecutando la propuesta hasta conformar el cuerpo de conclusiones y resultados del trabajo realizado. La metodología de desarrollo sigue la línea de la investigación-acción.

3.2 Diseño de la Investigación

Para lograr diseñar el modelo de referencia para PYMES en la selección de Herramientas de pruebas, se utiliza el Framework Metodológico del LISI que está inspirado en el método Investigación-Acción y utiliza la metodología DESMET para la fase de evaluación. Este Framework Metodológico, ha sido en gran parte resultado de aportes relevantes de cada investigador haciendo uso del mismo. Cada uno de las fases/pasos, están bien sustentadas con actividades de envergadura que permiten satisfacerlos.

Existe un tipo de sistema, denominado “Sistema de Actividad Humana”, donde según Checkland (1993), están formados por un número de actividades conectadas como resultado de algún principio de coherencia y que generalmente son complejos, multivariantes y poseen fenómenos del mundo real que no pueden ser pasados por alto. El método investigación acción está orientado a atacar este tipo de problemas en los que el componente humano juega un papel crucial. Bajo esta premisa, enmarcar la presente investigación dentro de la metodología científica tradicional, se torna insostenible. Es por ello que este marco de trabajo (framework) es suficiente y reemplaza, en esta investigación, al método científico tradicional.

Susman y Evered (1978) proponen una descripción de Investigación Acción, en la cual se detalla un proceso cíclico compuesto por cinco fases (Diagnosticar, Planificar la acción, Tomar la acción, Evaluar y Especificar el aprendizaje). Por su parte, Baskerville (1999) identifica cinco fases que interactúan entre sí:

1. **Diagnosticar:** Corresponde a la identificación de los principales problemas que atañen a la organización y que motivan su deseo de cambiar. Envuelve todo lo referente a la interpretación del problema complejo de la organización de forma holística.

2. **Planificar la Acción:** Esta actividad especifica las acciones organizacionales que deberían tomarse para relevar o mejorar los problemas detectados al diagnosticar. El descubrimiento de los planes de acción es guiado por el marco teórico, el cual indica el estado futuro deseado por la organización y los cambios requeridos para alcanzar ese estado. El plan establece el objetivo del cambio y el enfoque para cambiar.
3. **Tomar la acción:** Implementa el plan de acción. Los investigadores y participantes colaboran en la intervención activa dentro de la organización cliente, provocando ciertos cambios. Diversas estrategias de intervención pueden ser adoptadas: directiva (los investigadores dirigen el cambio), no directiva y tácita.
4. **Evaluar:** Una vez completadas las acciones, los investigadores y demás participantes evalúan las salidas. Ésta incluye determinar si los efectos teóricos de la acción fueron alcanzados y si estos efectos relevaban a los problemas. Si los cambios fueron exitosos, la evaluación se pregunta si los cambios propuestos fueron los únicos causantes de este éxito. Si los cambios no fueron exitosos, es necesario establecer un marco para la próxima iteración del ciclo de investigación acción.
5. **Especificar el aprendizaje:** A partir del resultado de la evaluación, los investigadores especifican el conocimiento adquirido.

Ahora bien, según la naturaleza empírica de la fase de Evaluación del Método Investigación Acción, es necesario reforzarla con otras metodologías como DESMET, la cual según Kitchenham, (1996) ayuda al evaluador de una organización a planificar y a ejecutar la evaluación de manera imparcial y fiable.

El *Framework* Metodológico del LISI consta de 11 pasos o actividades agrupadas en 5 grandes fases, que se corresponden con las fases del Método Investigación Acción descrito anteriormente, en conjunción con el método de evaluación DESMET, se visualiza en la Figura 12, que ilustra la adaptación de este *Framework* Metodológico propuesto a la investigación en curso.

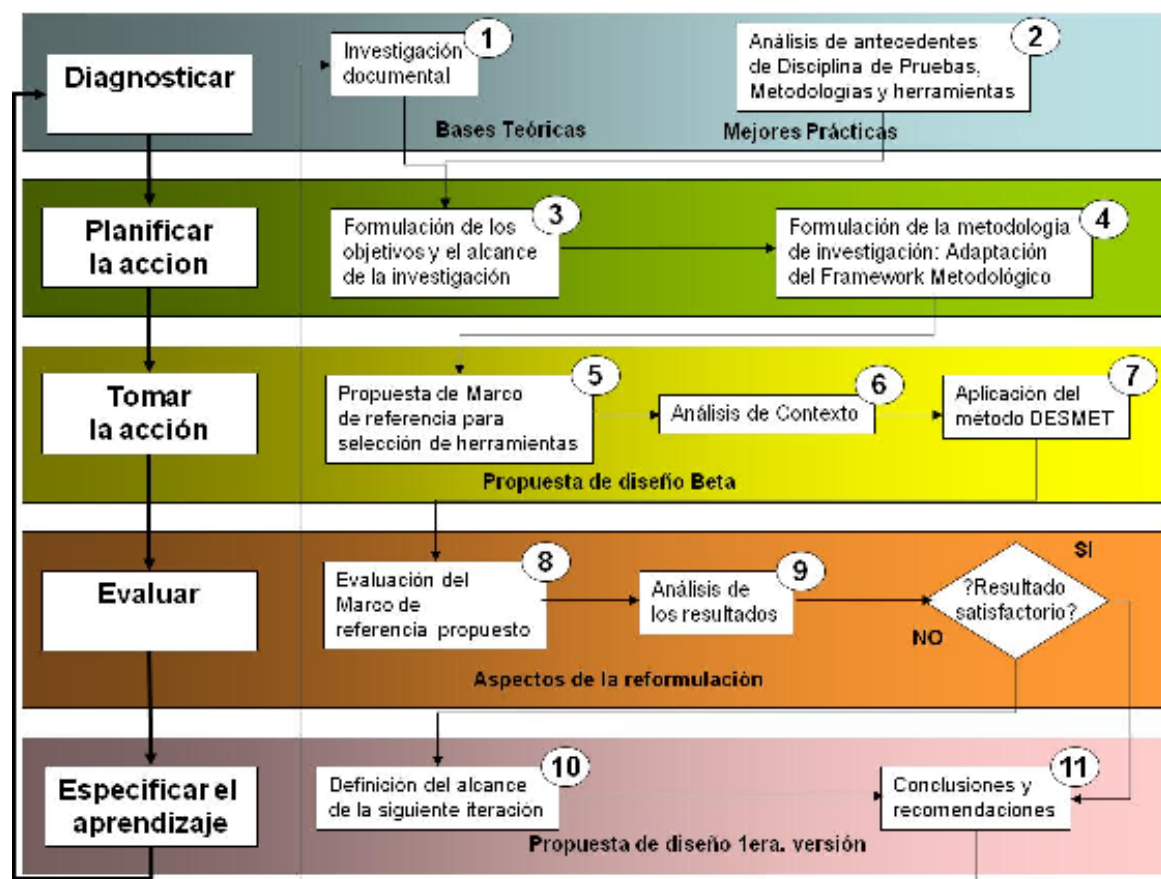


Figura 12 - *Framework Metodológico para la de investigación*. Fuente: Adaptado de Pérez et al. (2004)

Cada una de las actividades presentadas en la figura referenciada anteriormente, con su objetivo y breve descripción, se muestran a continuación en la Tabla 4, estableciendo la secuencia y principales salidas de dichas actividades, que conforman el *Framework* metodológico de la presente investigación.

Actividad	Descripción aplicada a la investigación
1. Investigación documental.	Etapa de la fase de Diagnosticar, consiste en la revisión del material bibliográfico relacionado con las herramientas de Pruebas, aseguramiento de la calidad y calidad del software. El objetivo es concretar un marco conceptual que soporte el trabajo de investigación. El resultado de esta actividad se presenta en el Capítulo II (Marco Teórico).
2. Análisis de antecedentes de Herramientas de Pruebas.	Etapa de la fase de Diagnosticar, que consiste en la descomposición de las Herramientas de Pruebas y sus relaciones con la disciplina de pruebas dentro del desarrollo de software, para indicar su importancia en el proceso. Igualmente, en esta etapa se identifican los trabajos relacionados antecedentes del mismo. El resultado de esta actividad se presenta en el Capítulo I y II.
3. Formulación de los objetivos y el alcance de la investigación.	Etapa de la fase Planificar la Acción donde se definen claramente los objetivos de la investigación, así como las directrices y el alcance. El objetivo es delimitar el área de investigación. El resultado de esta actividad está reflejado en la Justificación del Problema y la especificación de los Objetivos del Trabajo de Grado, presentados en el capítulo I de la presente investigación.

4. Formulación de la metodología de investigación: Adaptación de Investigación Acción.	Etapas de la fase Planificar la Acción donde se elabora la adaptación para esta investigación de la metodología de investigación acción. El objetivo es elaborar el marco metodológico que soporta el trabajo de investigación. El presente Capítulo III es el resultado de esta actividad.
5. Propuesta de Modelo de referencia para selección de herramientas	Etapas de la fase Tomar la Acción donde se realizan los análisis para el diseño del modelo de referencia en la selección de herramientas. El objetivo es proponer dicho modelo de referencia para la disciplina de Pruebas. Esta información es registrada en el Capítulo IV (Desarrollo de la Propuesta).
6. Análisis de Contexto.	Etapas de la fase Tomar la Acción, donde se determinan las especificaciones y acuerdos necesarios para implementar el modelo de referencia. El objetivo es preparar las herramientas, ambiente y el contexto donde será evaluada la propuesta. Esta información fue registrada en el Capítulo V (Evaluación de la Propuesta).
7. Aplicación del método DESMET.	Etapas de la fase Tomar la Acción, donde se ejecuta la metodología DESMET. El objetivo es obtener un método fiable e imparcial para la evaluación. Dicha evaluación se registrará en el Capítulo V (Evaluación de la propuesta).
8. Evaluación a través del método arrojado por DESMET.	Etapas de la fase Evaluar, que consiste en aplicar el método de evaluación arrojado por la actividad 7 al modelo propuesto. El objetivo es la evaluación del nuevo modelo mediante un método de evaluación apropiado y se registra en el Capítulo V (Evaluación de la propuesta).
9. Análisis de los resultados.	Etapas de la fase Evaluar, que consiste en estudiar los resultados a partir de los objetivos planteados en el trabajo de investigación, en términos de: el modelo propuesto y los productos tangibles alcanzados. Al terminar esta fase se evalúa si el resultado es satisfactorio, en caso afirmativo se pasa a la actividad 11, en caso negativo se pasa a la actividad 10. Esto se registró en el Capítulo VI (Análisis e interpretación de resultados).
10. Definición del alcance de la siguiente iteración.	Etapas de la fase Especificar el Aprendizaje, consiste en delimitar el alcance de las modificaciones que se deben realizar al modelo propuesto para aumentar la confiabilidad del mismo y sus posibilidades de éxito. Esto se registró en el Capítulo VI (Análisis e interpretación de resultados) solo para el nivel de métricas.
11. Conclusiones y recomendaciones.	Etapas de la fase Especificar el Aprendizaje, donde se establecen algunas conclusiones relativas al modelo propuesto aplicado y sus resultados. Finalmente, se sugieren algunas recomendaciones para futuros refinamientos del modelo propuesto y para investigaciones relacionadas. Esta información está presente en el Capítulo VII de la presente investigación (Conclusiones y recomendaciones).

Tabla 4 -Fases del Método Investigación Acción. Fuente: Elaboración Propia.

Apoyándose en el método de evaluación DESMET para validar el modelo de referencia propuesto, a continuación se presenta la descripción, objetivos y criterios de dicho método.

Según Kitchenham (1996), la metodología DESMET está compuesta por un conjunto de métodos con sus respectivas herramientas que pueden ser aplicadas o utilizadas por una organización en particular. Esta metodología, en el caso del presente trabajo de grado, será utilizada para la planificación y posterior evaluación

confiable del modelo propuesto que será aplicado en PYMES dedicadas al desarrollo de software.

De forma general, DESMET destaca la existencia de dos tipos de evaluación: la evaluación cualitativa y la evaluación cuantitativa. El propósito de las *evaluaciones cualitativas* es establecer efectos mensurables relativos al uso de un método o herramienta, basadas en la identificación de los beneficios esperados de los mismos y que aporten en términos mensurables y en la recolección de datos. Por su parte, las *evaluaciones cuantitativas* tienen como objetivo establecer qué tanto se ajusta un método o herramienta a las necesidades y a la cultura de una organización. Por lo general esto se determina en términos de las características del método o herramienta. Estas características se basan en los requerimientos de la población de usuarios y en los estándares de la organización. DESMET se refiere a este tipo de evaluación como Análisis de Características, el cual es identificado como una evaluación cualitativa o subjetiva (Kitchenham, 1996).

Este mismo autor establece que las técnicas de evaluación propuestas por DESMET son dependientes del contexto; puesto que no se espera que el objeto de evaluación sea el más apropiado en todas las circunstancias. En este sentido las diferencias en los resultados obtenidos están determinadas por las características de las empresas, no por los objetos a evaluar. Es por ello que al evaluar el modelo para seleccionar la mejor herramienta de pruebas en una organización específica que desarrolle sistemas de software, es muy probable que la importancia de cada característica varíe de acuerdo a las necesidades y a la realidad de la empresa.

Según Kitchenham, (1996), DESMET identifica 7 criterios que influyen en la selección del método de evaluación:

- *El contexto de la evaluación*
- *La naturaleza del impacto esperado de la utilización del método o la herramienta.*
- *La naturaleza del objeto de evaluación, es decir, si es un método, una herramienta o un método genérico.*
- *El alcance del impacto del método o la herramienta.*
- *La madurez del método o herramienta.*
- *La curva de aprendizaje asociada al método o la herramienta.*
- *Las habilidades de medición que posee la organización encargada de evaluar.*

Los criterios de selección permitirán escoger uno de los nueve métodos de evaluación propuestos por Kitchenham (1996):

1. **Experimento cuantitativo:** conforma el método científico básico para la determinación de diferencias entre métodos o herramientas.
2. **Estudio de caso cuantitativo:** consiste en la evaluación de un método o herramienta después que éste ha sido utilizado en un proyecto real.
3. **Encuesta cuantitativa:** es utilizada cuando varias herramientas o métodos han sido usados en la organización, y consiste en interrogar a los usuarios para que suministren información acerca de alguna característica de interés.
4. **Análisis de características por proyección:** por lo general se basa en la documentación de la herramienta o del método a evaluar.
5. **Análisis de características por estudio de caso:** consiste en el análisis de características del método o herramienta luego que el mismo ha sido aplicado en un proyecto real.
6. **Análisis de características por experimento:** en este caso se sugiere que un conjunto de usuarios potenciales pruebe el objeto de evaluación antes de realizar sus evaluaciones.
7. **Análisis de características por encuesta:** es realizada por personas con experiencia en el uso de la herramienta o con conocimientos teóricos de la misma.
8. **Análisis cualitativo de efectos:** se refiere a la utilización del juicio de expertos para determinar los efectos cuantitativos de diferentes métodos y herramientas.
9. **Benchmarking:** consiste en evaluar el comportamiento de la herramienta o método en relación con otros ya establecidos.

El ciclo metodológico usado soporta la investigación de problemas complejos, partiendo de prácticas exitosas y evaluando el modelo propuesto en versión Beta. Absorbe el aprendizaje y lo incorpora a una nueva propuesta mejorada. Este ciclo puede repetirse n veces; pero para efectos de esta investigación se considera una sola iteración, dado que es un Trabajo de Grado de Maestría con limitaciones de tiempo.

Por otra parte, según Pressman (2002), la única forma racional de mejorar cualquier proceso es medir atributos del proceso, desarrollar un juego de métricas significativas según estos atributos y entonces utilizar las métricas para proporcionar

indicadores que conducirán a una estrategia de mejora. A continuación se presenta un enfoque basado justamente en objetivos, preguntas y métricas que permite cuantificar o parametrizar el modelo de referencia a proponer.

Enfoque basado en objetivos, preguntas y métricas – GQM

Según Basili et al. (1994), el desarrollo de software como cualquier disciplina de ingeniería, requiere un mecanismo de medición para su corrección y evaluación. Para que este mecanismo sea efectivo debe estar centrado en objetivos específicos; debe ser aplicado a todo el ciclo de vida de los productos, procesos y recursos; y su interpretación debe estar basada en las características del contexto organizacional, ambiente y objetivos.

En este sentido, estos autores han desarrollado un enfoque basado en objetivos, preguntas y métricas (GQM – del inglés *Goal Question Metric*). Este enfoque, según Basili et al. (1994), posee tres niveles que se citan a continuación:

Primer Nivel - Conceptual (OBJETIVO): Un objetivo es definido por un objeto, un conjunto de razones con respecto a varios modelos de calidad y puede tener varios puntos de vista e incluso es relativo a un ambiente particular. Los objetos mensurables son productos, procesos y recursos.

Segundo Nivel – Operacional (PREGUNTA): Un conjunto de preguntas es utilizado para caracterizar la forma en que se alcanzará un objetivo específico. Las preguntas tratan de caracterizar el objeto de medición con respecto a un atributo de calidad seleccionado, para determinar su calidad desde un cierto punto de vista.

Tercer Nivel – Cuantitativo (METRICA): Un conjunto de datos es asociado con cada pregunta para responderla de forma cuantitativa. Los datos pueden ser objetivos (si dependen sólo del objeto medido) o subjetivos (si dependen del objeto medido y del punto de vista desde el cual son tomados).

Para Pressman (2006), la importancia de GQM proviene no solamente del hecho de que es uno de los primeros intentos de desarrollar un conjunto de medidas adecuado que pueda ser aplicado al software, sino también al hecho de que está relacionado con el paradigma de mejora de procesos. El enfoque GQM se presenta como un mecanismo detallado para la definición y elaboración de objetivos medibles.

3.3 Técnicas e instrumentos de recolección de datos

Es importante destacar que tomando en cuenta el framework metodológico con todos sus aportes, explicado en la sección anterior, algunas de las restantes fases que especificadas en el método científico tradicional, como lo son: a) Población y muestra, b) Operacionalización de variables y c) Validación de instrumentos, que formarían parte del presente capítulo III, son suplidas dentro del marco de trabajo que rige esta investigación, considerado como el más idóneo para el estudio de los Sistemas de información, por varias razones de acuerdo a lo analizado en base al autor Baskerville (1999):

- *Dentro del quehacer científico consiste no solo en la comprensión de los aspectos de la realidad existente, sino también en la identificación de las fuerzas sociales y las relaciones que están detrás de la experiencia humana.*
- *En la investigación - acción no hay mucho énfasis en el empleo del instrumental técnico de estadísticas y de muestreo. El método investigación-acción, no trabaja con población y muestra, puesto que se enfoca en un caso bien detallado, que para la investigación en desarrollo, son las PYMEs desarrolladoras de software en Venezuela.*
- *Los resultados se prueban en la realidad. Las experiencias que resultan en el campo social proporcionan las informaciones acerca de los procesos históricos, luego de su análisis.*

En cuanto a las técnicas e instrumentos para recolección de información, se usarán las encuestas en su modalidad de entrevistas con expertos e investigadores del Laboratorio de Investigación en Sistemas de Información (LISI). Igualmente, se usaran algunas técnicas que sustenten la investigación documental. Esto es, para el análisis documental, se utilizará el análisis de contenido y algunos cuadros de registro y clasificación de las categorías para comparar las herramientas de pruebas del mercado.

CAPÍTULO IV: DESARROLLO DE LA PROPUESTA

En este capítulo se describe el modelo de calidad propuesto, el cual establece las características que una herramienta de software debe cumplir según el tipo de prueba que se requiera ejecutar en pro de permitir la selección de herramientas de prueba, así como también, para orientar la construcción de un software adaptado a necesidades particulares de una organización como herramienta de prueba. El resultado de este modelo permite seleccionar herramientas de software de prueba, como propósito principal (enfoque en esta investigación). Bajo esta perspectiva, se realizan aportes al conocimiento y se presenta primeramente el modelo de dominio o conceptual en UML, con su respectiva explicación, basado en la identificación de las características que deben estar presentes en el modelo y luego de haber realizado una revisión bibliográfica de los tópicos asociados. Seguidamente, se profundiza sobre las herramientas de prueba existentes en el mercado, presentando un árbol de clasificación propuesto de herramientas de prueba automatizadas. Finalmente, se presenta el desarrollo de la propuesta con la instanciación del modelo MOSCA para pruebas a nivel de producto.

4.1 Modelo Conceptual

El modelo conceptual presentado en la Figura 17, resume todos los conceptos y relaciones que se obtuvieron a lo largo de esta investigación, así como los aportes que ha dado el investigador en este documento (tales como la adaptación de la figura 13) y el análisis de un conjunto de herramientas de prueba.

Teniendo como parte central las pruebas, se desglosan varios diagramas conceptuales específicos y ciertamente relacionados entre los que están: Casos de Prueba, Tipos de prueba, Niveles de prueba, Herramientas de prueba, apoyo al desarrollo del software y el aseguramiento de la calidad, los cuales se agrupan en diagramas distintos a fin de poder explotar sus relaciones de forma más detallada, seguido de su correspondiente explicación.

Como se observa en la Figura 13, los casos de prueba están íntimamente relacionados con los requerimientos, pues los Casos de Uso dan el aporte básico para plantear los respectivos Casos de prueba. De igual forma, los casos de prueba son

descritos a través de los procedimientos de prueba que con sus pasos, condiciones y valores o datos de la prueba, se conjugan para ser candidatos a automatizarse a través de los Scripts de prueba. Estos últimos, en forma ordenada, se pueden ejecutar planteando una secuencia lógica del conjunto de casos de prueba (Suite de prueba), sobre un ambiente de prueba predefinido constituido principalmente por el software, hardware y base de datos donde se ejecute la aplicación objeto de prueba. A su vez, la ejecución de las prueba trae consigo resultados satisfactorios o dignos de atención como lo son las fallas, que determinarán finalmente el resultado de la prueba obtenidos en contraste con los esperados. Los casos de prueba se ejecutan en un ambiente de prueba que puede estar conformado por uno o más de los siguientes elementos: software, hardware y base de datos.

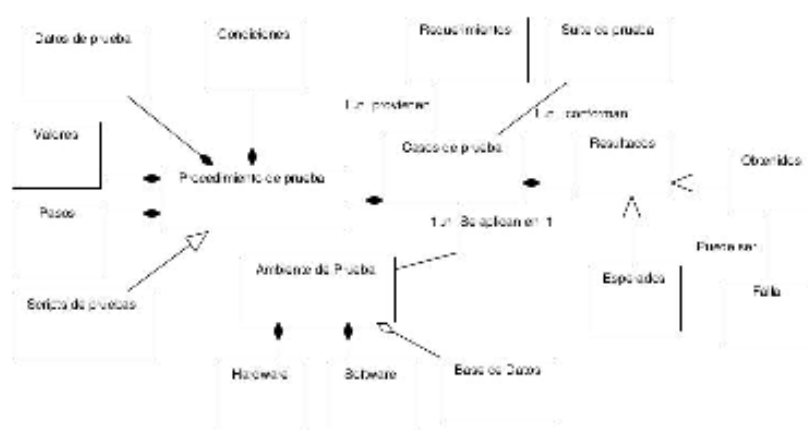


Figura 13- Modelo Conceptual asociado a los Casos de Prueba. Fuente: Adaptado de Méndez et al. (2007).

En la Figura 14 se presentan 23 de los muchos tipos de prueba que son propuestos por diversos autores, ya que el investigador consideró éste subconjunto más relevante y a la vez general entre la clasificación de prueba y/o las diversas tipologías de prueba. En cuanto a los niveles de prueba, se detalla la clasificación de acuerdo al objeto de prueba; esto es, basadas en las pruebas del desarrollador (unitarias), con otros módulos (integración), certificadas (aceptación del usuario) y del sistema con sus interfaces y demás componentes simulando que ya está en ambientes de producción. De la misma forma existen las técnicas o enfoques de pruebas (caja blanca y caja negra) que permiten determinar el enfoque en el marco de las pruebas, pues buscan verificar (internamente – estructura de implementación) y validar (externamente – cumplimiento de la funcionalidad) respectivamente.

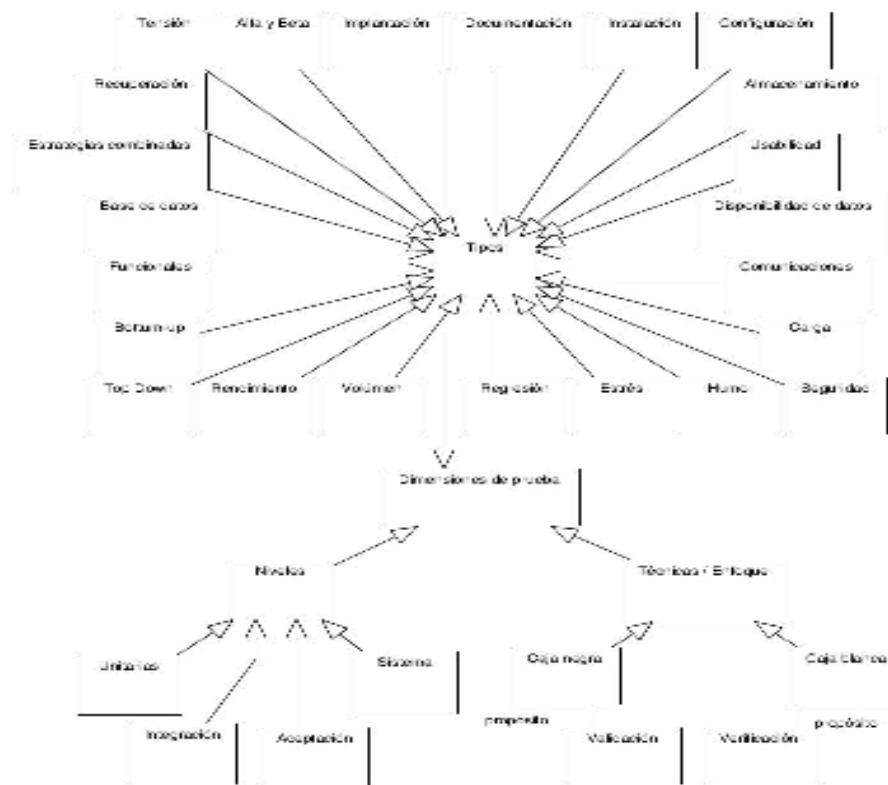
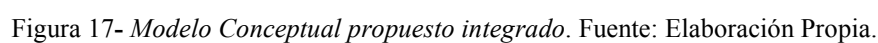


Figura 14 - Modelo Conceptual asociado a los Tipos, Niveles y Técnicas de Prueba.
Fuente: Elaboración Propia

Los casos de prueba pueden ser gestionados y/o automatizados a través de herramientas de prueba destinadas para tal fin de acuerdo a los diversos tipos de prueba que existen. Esencia de esta investigación, el modelo de la Figura 15 presenta una clasificación de las mismas que puede seguir descomponiéndose en otras subclasificaciones. Por otra parte, en la misma figura, se presenta un aporte al estudio de las herramientas de prueba en cuanto al tipo de aplicaciones que pueden ser probadas a través de las herramientas; esto es, qué tipo de tecnología, plataformas o propósitos deben tener las herramientas de prueba para adaptarse a una u otra aplicación. Cabe destacar que no todas las herramientas de prueba sirven para probar todas las aplicaciones que se deseen probar; depende de los objetos de prueba que se planteó abarcaría su construcción.

Como bien se ha mencionado en la justificación de esta investigación, la disciplina de pruebas apoya en gran manera el proceso de desarrollo de software, por lo que se presentan algunas de las metodologías que expresamente hacen mención a las pruebas dentro de sus procesos (ver Figura 15).



4.2 Clasificación propuesta para las Herramientas de prueba

Como se estableció en la sección VI de las Bases Teóricas, existen numerosas herramientas de Prueba en el mercado con características bien diferenciadoras en algunos casos y en otros, con más similitudes que diferencias y con propósitos diversos de acuerdo a lo establecido por SWEBOK (2004). Sin embargo, esta investigación está dirigida a herramientas de prueba que permitan probar el software de manera fiable más que administrar y/o gestionar pruebas, es decir, herramientas que permitan automatizar la ejecución de las pruebas.

Proponer una guía para la selección de herramientas supone el hecho de que se debe tener claro, en primera instancia, el propósito por el cual se busca automatizar las pruebas de un software en desarrollo (aplicación objeto de prueba); esto es la base de cualquier decisión de selección entre múltiples alternativas. Seguidamente, entre este mundo de posibilidades, es necesario intentar agrupar las opciones que tengan algo en común, a fin de elegir la más óptima; es decir, obtener una muestra significativa de un todo, de acuerdo a un criterio establecido. Traducido esto a esta investigación, las alternativas de herramientas de prueba que serán objetos de evaluación, deben tener alguna afinidad, deben estar ajustadas a un mismo propósito tal que puedan reflejar su ventaja competitiva frente a sus pares. En otras palabras, se puede penalizar una herramienta poderosa por el simple hecho de que no cumpla con las características esenciales que permitan plasmar el propósito definido en la primera instancia. Por ejemplo, comparar dos herramientas muy buenas funcionalmente hablando, pero descartar alguna en específico porque no permite hacer pruebas de regresión, siendo que el requerimiento inicial es ejecutar pruebas de regresión automatizadas.

Bajo ésta premisa, en la Figura 18 se propone una clasificación de herramientas de prueba automatizadas en base a la exploración por varios portales que exponen herramientas del mercado tecnológico. Esta clasificación se basa en ciertas particularidades presentes en cada herramienta de prueba analizada, en la agrupación de características de interés común entre las mismas y tomando en cuenta el modelo conceptual presentado en la sección anterior, todo esto con el fin de determinar la característica que presenta MOSCA dentro de la categoría Funcionalidad, la cual es el ajuste a los propósitos, factor determinante al momento de

elegir entre varias opciones de herramientas de prueba automatizadas para suplir una necesidad de prueba específica.

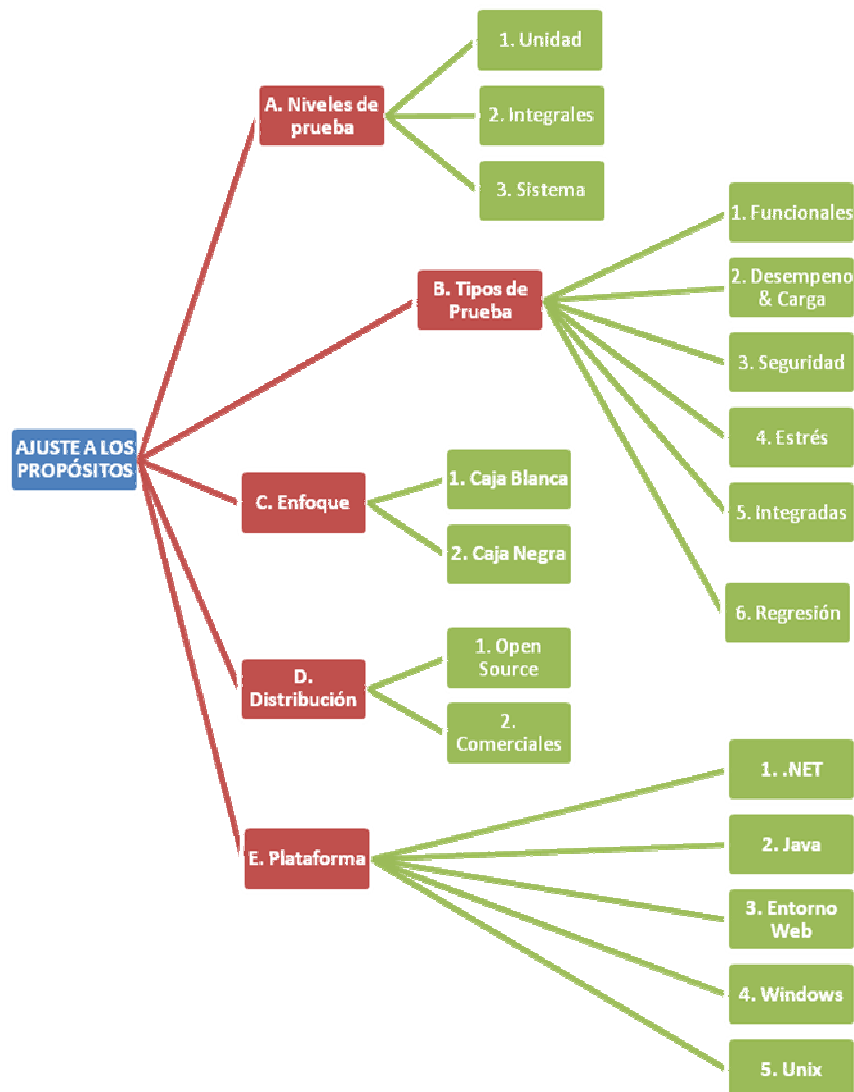


Figura 18 - *Árbol de clasificación de herramientas de prueba automatizadas.*
Fuente: Elaboración Propia.

Cabe destacar que una herramienta en específico puede estar en más de una rama del árbol por lo que algunas ramas no son exclusivas (Niveles, Tipos de prueba y Plataforma); por ejemplo, se pueden encontrar herramientas de prueba de nivel integral con distribución comercial o bien, herramientas a nivel de sistema que permita hacer pruebas funcionales y a su vez de estrés (repitiendo la prueba funcional muchas veces) con distribución Opensource. En términos cuantitativos, según el Instituto de automatización de pruebas (ATI, 2010 - www.testingfaqs.org, www.automatedtestinginstitute.com) existen en el mercado alrededor de 460

herramientas de prueba para la rama de Niveles de prueba unitarias, de las cuales un 65% son de distribución opensource y el resto comerciales, dirigidas principalmente a las pruebas de tipo funcional, con algunas excepciones que pueden ser usadas para herramientas no funcionales como estrés o carga; de esta muestra, algunas pueden probar aplicaciones en varias plataformas, mientras que otras están orientadas a una plataforma en específico por una parte, y por la otra, están enfocadas en su mayoría a caja blanca.

La realidad ejemplificada en el párrafo anterior se traduce en infinitas combinaciones posibles dentro del extenso conjunto de herramientas de prueba automatizadas, especialmente para esas ramas del árbol que no son estrictamente excluyentes. Estas combinaciones pueden producirse entre las categorías o niveles del árbol o bien, entre los hijos de una categoría en particular, a excepción de los niveles siguientes por ser excluyentes: distribución y enfoque. En consecuencia, se puede instanciar el árbol de clasificación de herramientas de muchas maneras y mezclas, combinando algunos niveles con sus hijos, entre otros.

Con el propósito de validar el modelo de referencia para seleccionar herramientas de prueba y sustentar la presente investigación, se debe de alguna manera instanciar el árbol de clasificación de herramientas basado en el ajuste a los propósitos para lograr una muestra representativa de herramientas de dicha instanciación. Por tal motivo, se eligió una de las tantas instanciaciones del árbol, representada por la Figura 19, que sigue la ruta A3 B2 C2 D*E* (ver cuadros rellenos de color), es decir, la rama de herramientas de prueba automatizadas para los tipos de prueba de sistema con los tipos de desempeño y de carga, a fin indagar sobre sus prestaciones y estudiar las características tanto comunes como distintivas que las especializan.

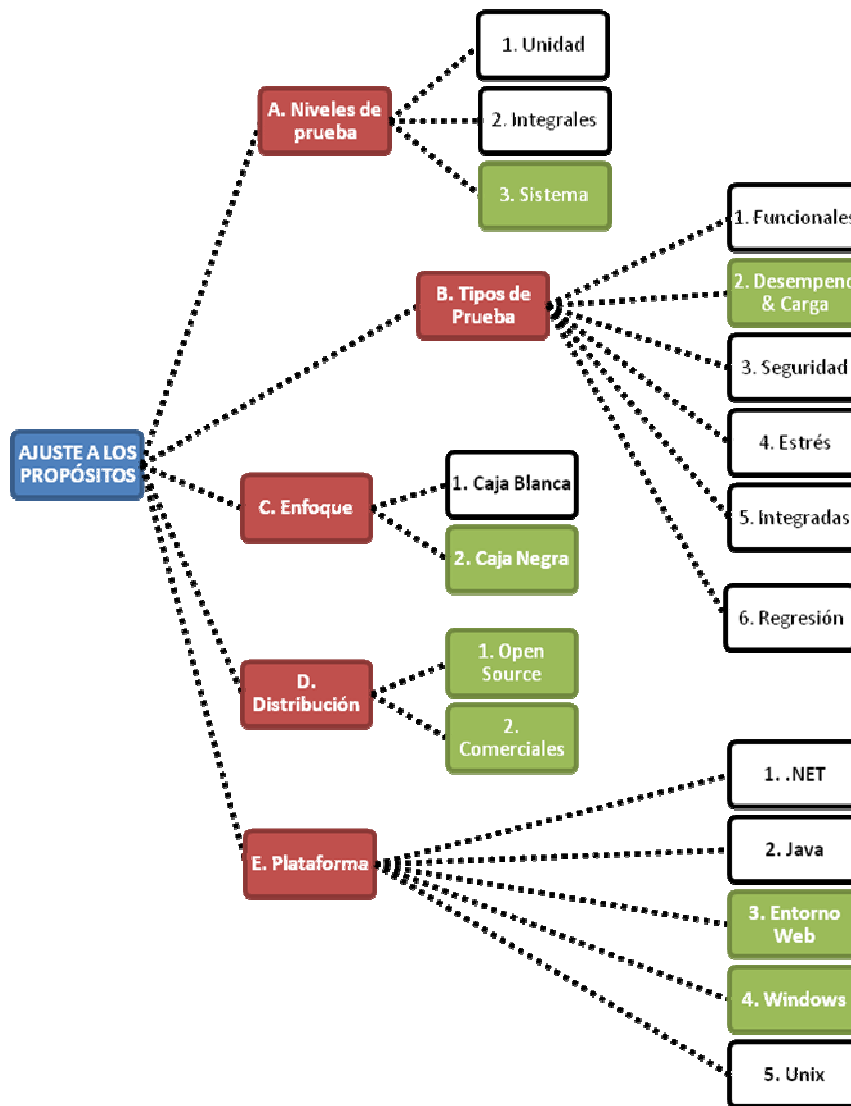


Figura 19 - Árbol instanciado de ajuste a los propósitos para herramientas de prueba elegidas por el investigador. Fuente: Elaboración Propia.

4.3 Aspectos tomados de MOSCA para la instanciación

En esta sección se presentan los diversos aspectos analizados y utilizados del Modelo Sistemático de Calidad (MOSCA) de forma general a fin de adaptarlo al objetivo de la presente investigación, selección de herramientas de pruebas, recorriendo cada uno de los niveles que lo conforman, de acuerdo al ANEXO I. Bajo ésta premisa y siguiendo la guía para la adaptación de un modelo genérico de calidad de software propuesto por Rincón *et al.* (2004) representado por MOSCA hasta llegar a un modelo de calidad específico, representado por el modelo de referencia para la selección de herramientas de prueba propuesto, se definieron los niveles del modelo de acuerdo a los siguientes planteamientos:

- a) Selección del submodelo: Se tomó el submodelo *Producto*, puesto que el objeto de esta investigación es poder seleccionar productos ya desarrollados constituido por las Herramientas de prueba.
- b) Selección de la *Dimensión* (Nivel 0): Por el punto anterior, las dimensiones asociadas al submodelo *Proceso* y *Humana* fueron descartadas al igual que la dimensión *Aspectos Internos* del *Producto*, ya que no se cuenta con el código fuente de todas las herramientas a ser evaluadas. Por lo tanto, la única dimensión tomada en la instanciación es la de los *Aspectos Contextuales*.
- c) Selección de las *Categorías* (Nivel 1): Según el algoritmo de evaluación de MOSCA (Mendoza *et al.*, 2005), la primera categoría que debe evaluarse para el submodelo *Producto* es la *Funcionalidad*, ya que esta determina la capacidad del software para cumplir con las funciones para las que fue fabricado, dando garantía de pruebas de software realmente confiables. Seguidamente deben elegirse las categorías restantes para la evaluación entre Eficiencia, Mantenibilidad, Confiabilidad, Portabilidad y Usabilidad. Se eligió la categoría *Usabilidad* en pro de seleccionar herramientas que sean fáciles en su manejo, fáciles de aprender, de entender y hagan más viable el probar una aplicación a fin de no desviar el propósito de la prueba en entrenamiento de la herramienta de prueba. La categoría *Eficiencia* se escogió porque es fundamental conocer con cuánta celeridad se pueden obtener los resultados, proveyendo un rendimiento esperado, aún cuando aumenten los recursos utilizados (al probar un software) en condiciones específicas. La categoría *Fiabilidad* fue elegida porque es necesario garantizar que la Herramienta de prueba mantenga un nivel especificado de rendimiento bajo ciertas condiciones especificadas, es decir, que se mantenga a tono aún cuando la aplicación que está probando falle. La categoría *Mantenibilidad* que corresponde a la capacidad del producto para ser modificado, con el fin de implementar mejoras, actualizaciones en plataformas, excede el propósito de esta investigación que solo quiere determinar las prestaciones en el presente, por lo que se descartó así como la *Portabilidad*, que aunque es importante saber su capacidad de instalación, el proyecto piloto no contempla evaluar la herramientas en varios ambientes.
- d) Selección de las *Características* (Nivel 2): En cuanto al Nivel 2, correspondiente a las *características* de cada una de estas categorías, se

presentan a continuación las que se tomaron del modelo por considerarlas pertinentes a la investigación, excluyendo las que no se consideraron necesarias o aplicables dentro de una categoría ya sea porque corresponden a los Aspectos internos del Producto o por las razones que se presentan a continuación:

- ✓ **Funcionalidad (FUN)**
 - FUN 1. Ajuste a los propósitos
 - FUN 2. Precisión
 - FUN 3. Interoperabilidad
 - FUN 4. Seguridad
- ✓ **Usabilidad (USA)**
 - USA 1. Usabilidad
 - USA 2. Capacidad de Aprendizaje
 - USA 3. Interfaz Gráfica
 - USA 4. Operabilidad

La USA 5. Conformidad con los estándares, se excluyó porque se desconocen estándares mundialmente aceptados que debe poseer una herramientas de prueba; proponer las características básicas que debe poseer una herramienta de prueba (de acuerdo a su tipo) es parte fundamental de esta investigación.

- ✓ **Fiabilidad (FIA)**
 - FIA 2. Tolerancia a Fallas
 - FIA 3. Recuperación

La FIA 1. Madurez se excluyó de la investigación porque las empresas que representan las herramientas de pruebas, omiten información sobre los procesos que usan para certificar su producto antes de sacarlo al mercado (pruebas, componentes, mantenimiento). Se asume que tienen altos niveles de calidad del software.

- ✓ **Eficiencia (EFI)**
 - EFI 1. Comportamiento del tiempo
 - EFI 2. Utilización de recursos

Finalmente, las características correspondientes a las dos categorías no elegidas, se descartaron del modelo por las razones expuestas en el punto c).

- e) Selección de las *Métricas* (Nivel 3): Finalmente, las *métricas* de cada característica a ser evaluada correspondientes al Nivel 3 de MOSCA, de la misma forma como las características, se excluyeron aquellas que no eran pertinentes en esta investigación, mientras que otras se usaron tal como lo

expresa el modelo, de acuerdo a las herramientas de prueba que existen en el mercado. Las métricas tomadas de MOSCA se observan en los anexos II, III y IV correspondiente a cada una de las categorías seleccionadas, que se cuantifican de la siguiente manera: 15 métricas para Funcionalidad, 17 métricas para Usabilidad, 4 métricas para Fiabilidad y 4 para Eficiencia, para un total de 36 métricas usadas de MOSCA.

En la sección siguiente, se realiza la instanciación de MOSCA, mostrándose los aportes al modelo sistémico de calidad enfocado en las herramientas de prueba en pro de proponer un modelo para su selección.

4.4 Desarrollo de la propuesta (Aspectos agregados a MOSCA para la instanciación)

Con el propósito de instanciar MOSCA expresamente para poder hacer una selección de Herramientas de Prueba de software, se agregaron los siguientes aspectos, producto de esta investigación.

Se implementó el enfoque de GQM (Meta – Pregunta - Métrica) explicada en el Capítulo III para poder operacionalizar y elaborar los indicadores que se utilizaron para medir el modelo de referencia propuesto en las nuevas sub-características y métricas aportadas al modelo MOSCA con sus correspondientes preguntas para verificar la métrica que varían entre al menos una pregunta y máximo cuatro preguntas por métrica. Los aportes al modelo se basaron en ciertos aspectos como lo son: a) El modelo conceptual propuesto; b) El árbol de clasificación de herramientas de prueba propuesto; c) Inspección inicial de aspectos que caracterizan algunas herramientas de prueba que surgen del árbol de clasificación de herramientas instanciado elegidas del mercado tecnológico que son en número mayor a las que finalmente fueron elegidas para ser evaluadas.

Categoría FUNCIONALIDAD

Para la característica “Ajuste a los propósitos – FUN 1” se agregaron nueve (9) sub-características con sub-sub-características y métricas las cuales pueden observarse en la Tabla Nro. 5.

- (FUN 1.1) Niveles de Prueba: posee 1 sub-sub-característica y 1 métricas.
- (FUN 1.2) Tipos de Prueba: posee 2 sub-sub-característica y 2 métricas.

- (FUN 1.3) Enfoque de Prueba: posee 1 sub-sub-característica y 2 métricas.
- (FUN 1.4) Tipo de Licencias de distribución de la Herramienta de Prueba: posee 2 sub-sub-característica y 2 métricas.
- (FUN 1.5) Administración de la Herramienta de Prueba: posee 1 sub-sub-característica y 3 métricas.
- (FUN 1.6) Entregables de Prueba: posee 6 sub-sub-característica y 25 métricas.
- (FUN 1.7) Planificación de Prueba: posee 1 sub-sub-característica y 6 métricas.
- (FUN 1.8) Gestión de Resultados: posee 3 sub-sub-característica y 14 métricas.
- (FUN 1.9) Aplicación objeto de Prueba: posee 2 sub-sub-característica y 8 métricas.

En la característica “Precisión – FUN 2” se agregó una (1) sub-característica con subsub-características y métricas, las cuales pueden observarse en la Tabla Nro. 6.

- (FUN 2.1) Criterios de aceptación de Pruebas: posee 2 sub-sub-característica y 2 métricas.

Para la característica “Interoperabilidad – FUN 3” se agregó una (1) sub-característica con sub-sub-características y métricas, las cuales pueden observarse en la Tabla Nro. 7.

- (FUN 3.1) Interacción con otras aplicaciones: posee 3 sub-sub-característica y 6 métricas.

Para la característica “Seguridad – FUN 4” se agregaron dos (2) sub-características con sub-sub-características y métricas, las cuales pueden observarse en la Tabla Nro. 8.

- (FUN 4.1) Control de acceso: posee 1 sub-sub-característica y 3 métricas.
- (FUN 4.2) Capacidad de auditoría: posee 2 sub-sub-característica y 3 métricas.

Categoría USABILIDAD

Para la característica “Capacidad de aprendizaje – USA 2” se agregaron dos (2) métricas; en la característica “Interfaz Gráfica – USA 3” se agregaron dos (2) métricas y en la “Operabilidad - USA 4” se crearon dos (2) métricas, las cuales pueden observarse en la Tabla Nro. 8.

Finalmente, el modelo MOSCA instanciado, con todos los aspectos tomados del modelo original, las adaptaciones y nuevos aspectos propuestos por el investigador, excluyendo los aspectos no relevantes para el modelo de referencia de

selección de herramientas de prueba, se presenta en la Figura 20. De forma cuantitativa, se han agregado al modelo 13 nuevas sub-características, 27 nuevas sub-sub-características y 83 métricas. En el anexo V puede observarse la descripción de algunas de las características propuestas a fin de contextualizarlas en el modelo propuesto.

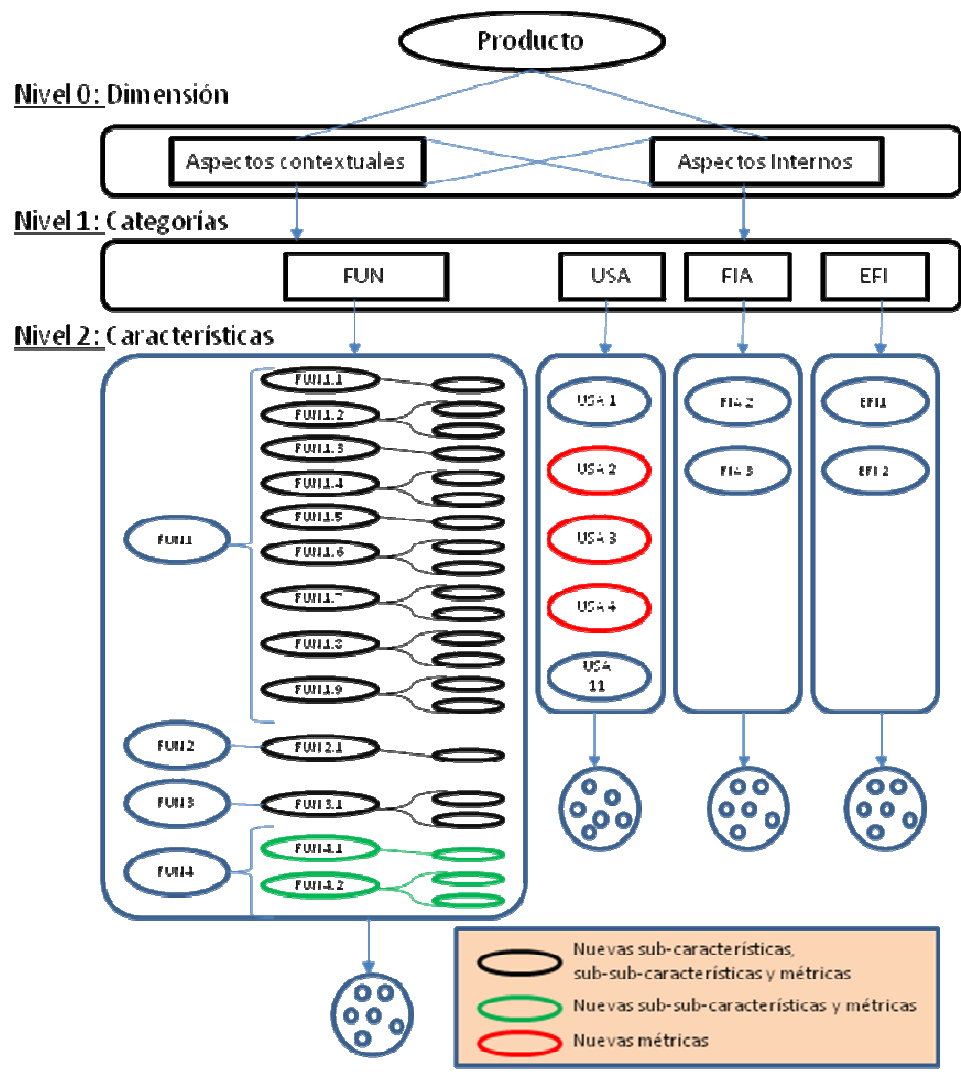


Figura 20 - Instanciación de MOSCA para el modelo de referencia para la selección de herramientas de Prueba. Fuente: Elaboración propia.

FUNCIONALIDAD (FUN) - Característica: Ajuste a los Propósitos (FUN 1)						
Sub-característica	Sub-característica	Sub-característica	Métrica	Pregunta	Formulación	Dirigido a
Niveles de Prueba (FUN 1.1)	de	Pruebas de sistema (FUN 1.1.1)	Ejecución de pruebas de a nivel de sistema	¿Permite realizar pruebas al nivel de pruebas del sistema?	5 Si, 1 No	Usuario
Tipos de Prueba (FUN 1.2)	de	Pruebas de Desempeño (FUN1.2.1)	Ejecución del Tipo de Pruebas de Desempeño	¿Permite realizar pruebas del tipo Desempeño a la Aplicación objeto de prueba?	5 Si, 1 No	Usuario
		Pruebas de	Ejecución del Tipo de	¿Permite realizar pruebas del	5 Si, 1 No	Usuario

	Carga/Estrés (FUN1.2.2)	Pruebas de Carga/Estrés	tipo Carga/Estrés a la Aplicación objeto de prueba?		
Enfoque de Prueba (FUN 1.3)	Pruebas de Caja Negra (FUN 1.3.1)	Ejecución de Pruebas con enfoque de Caja Negra	¿Las pruebas son de enfoque de Caja Negra?	5 Si, 1 No	Usuario
		Exploración en el código de la Aplicación objeto de prueba	¿Omite el Profundizar/analizar el código de la Aplicación objeto de prueba?	5 Si, 1 No	Usuario
Tipo de Licencias de distribución de la Herramienta de Prueba (FUN 1.4)	Open source (FUN 1.4.1)	Distribución de la Herramienta de Prueba Opensource	¿La licencia de la Herramienta de Prueba es de tipo opensource (libre)?	5 Si, 1 No	Usuario
	Comercial (FUN 1.4.1)	Distribución de la Herramienta de Prueba es comercial	¿La licencia de la Herramienta de Prueba es de tipo comercial (privado)?	5 Si, 1 No	Usuario
Administración de la Herramienta de Prueba (FUN 1.5)	Lenguaje de Programación de la Herramienta de Prueba (FUN 1.5.1)	Código único/proceso unificado para multiplataformas	¿Proporciona un lenguaje de programación único para la construcción de entregables?	5=Muy difícil 4=Difícil 3=Promedio 2=Fácil 1=Muy fácil	Usuario
		Código Propietario (propone comandos, funciones propietarias)	¿Propone su propio pseudo-lenguaje de programación para construir entregables?	5=Muy difícil 4=Difícil 3=Promedio 2=Fácil 1=Muy fácil	Usuario
		Necesidad de amplios conocimientos en programación	¿Requiere conocimientos de programación a un nivel específico para poder construir los entregables?	5=Muy alta 4=Alta 3= Mediana 2= Básica 1= No tiene	Usuario
Entregables de Pruebas (FUN 1.6)	Casos de Prueba (FUN 1.6.1)	Generación automática de Casos de Prueba	¿Genera automáticamente los Casos de Prueba de acuerdo a ciertas entradas?	5 Si, 1 No	Usuario
			¿Cómo es la completitud de los Casos de Prueba generados automáticamente en cuanto a: pasos, pre-condiciones, post-condiciones, valores, resultados esperados?	5=Completa. 4=Casi todos 3= Med. 2= Pocos 1= No aplica	Usuario
		Construcción de Casos de Prueba paso a paso	¿Permite construir Casos de Prueba paso a paso?	5 Si, 1 No	Usuario
		Identificación de los requerimientos de prueba	¿Permite identificar los requerimientos de prueba que se requieren probar?	5 Si, 1 No	Usuario
		Reutilización de Casos de Prueba entre proyectos	¿Los Casos de Prueba sirven para alimentar proyectos afines?	5 Si, 1 No	Usuario
		Documentación de Pre-condiciones	¿Permite documentar las pre-condiciones en los Casos de Prueba?	5 Si, 1 No	Usuario
		Documentación de Post-condiciones	¿Permite documentar las post-condiciones en los Casos de Prueba?	5 Si, 1 No	Usuario
		Datos de Prueba (FUN 1.6.2)	¿Se establecen manualmente los Datos de Prueba en los Casos de Prueba?	5 Si, 1 No	Usuario
		Generación automática de	¿Genera automáticamente	5 Si, 1 No	Usuario

		Datos de Prueba desde datos reales	los Datos de Prueba desde los datos reales identificados?		
		Generación automática de Datos de Prueba desde sentencias SQL	¿Se establecen automáticamente los Datos de Prueba desde sentencias SQL, base de datos?	5 Si, 1 No	Usuario
		Generación automática de Datos de Prueba de forma aleatoria	¿Genera automáticamente los Datos de Prueba aleatoriamente de acuerdo a datos iniciales?	5 Si, 1 No	Usuario
		Reutilización de Datos de Prueba entre proyectos	¿Los Datos de Prueba sirven para alimentar proyectos afines?	5 Si, 1 No	Usuario
	Script de Pruebas (FUN 1.6.3)	Generación automática de Scripts de Prueba desde Casos de Prueba + Datos de Prueba	¿Genera automáticamente los Scripts de Prueba de en base a los Casos de Prueba + Datos de Prueba?	5 Si, 1 No	Usuario
		Construcción de Scripts de Prueba paso a paso	¿Permite construir Scripts de Prueba paso a paso, manualmente?	5 Si, 1 No	Usuario
		Modificar Scripts de Prueba luego de generarlos	¿Permite modificar Scripts de Prueba luego de generarlos?	5 Si, 1 No	Usuario
		Modificar Scripts de Prueba en línea/ejecución	¿Permite modificar Scripts de Prueba al momento de ejecutarlo?	5 Si, 1 No	Usuario
		Reutilización de Scripts de Prueba entre proyectos	¿Permite usar/importar Scripts de Prueba existentes entre proyectos?	5 Si, 1 No	Usuario
	Suite de Pruebas (FUN 1.6.4)	Organizar/secuenciar conjunto de pruebas	¿Permite organizar y/o secuenciar conjunto de Scripts de Prueba?	5 Si, 1 No	Usuario
		Programación de Scripts de Prueba en el tiempo	¿Permite programar pruebas en el tiempo (scheduler)?	5 Si, 1 No	Usuario
	Salidas de pruebas (FUN 1.6.5)	Generación de evidencias de Pruebas	¿Genera evidencias de resultados de pruebas automáticos?	5 Si, 1 No	Usuario
		Gestión de defectos	¿Documenta defectos en forma automática luego de presentado el error?	5 Si, 1 No	Usuario
			¿Permite agregar/crear defectos?	5 Si, 1 No	Usuario
			¿Permite modificar defectos?	5 Si, 1 No	Usuario
			¿Permite notificar defectos al responsable de la aplicación/módulo?	5 Si, 1 No	Usuario
		Log de pruebas/histórico de ejecuciones	¿Permite guardar un log de las pruebas ejecutadas?	5 Si, 1 No	Usuario
	Relación entre entregables de Pruebas (FUN 1.6.6)	Relación entre requerimientos de prueba y Casos de Prueba	¿Posee mecanismo para relacionar requerimientos de prueba a los Casos de Prueba?	5 Si, 1 No	Usuario
		Relación entre Casos de Prueba y Script de Prueba	¿Posee mecanismo para relacionar Casos de Prueba a los Scripts de Pruebas?	5 Si, 1 No	Usuario
		Relación de defectos a Casos de Prueba	¿Posee mecanismo para relacionar defectos a los	5 Si, 1 No	Usuario

Planificación de Pruebas (FUN 1.7)	Condiciones a probar en la Aplicación objeto de prueba (FUN 1.7.1)		Casos/Script de Pruebas?		
		Relaciona defectos a su causa	¿Permite determinar la posible causa del defecto?	5 Si, 1 No	Usuario
		Emulación de usuarios concurrentes	¿Emula población/acceso de usuarios concurrentes?	5 Si, 1 No	Usuario
		Emulación de transacciones concurrentes	¿Emula gran carga de transacciones concurrentes?	5 Si, 1 No	Usuario
		Variación de carga de trabajo durante la prueba	¿Permite variar la carga de trabajo en el tiempo de la ejecución de la prueba?	5 Si, 1 No	Usuario
		Definición de límites/umbrales de finalización de prueba	¿Permite definir límites/umbrales donde la prueba debe concluir?	5 Si, 1 No	Usuario
			¿Permite definir tiempos donde la prueba debe concluir?	5 Si, 1 No	Usuario
		Definición de estrategias de pruebas	¿Permite ingresar retardos en la ejecución de pruebas?	5 Si, 1 No	Usuario
			¿Permite definir hilos de ejecución en la prueba?	5 Si, 1 No	Usuario
			¿Permite definir ráfagas aleatorias en la ejecución de pruebas?	5 Si, 1 No	Usuario
			¿Permite establecer un script para la estrategia de carga de trabajo?	5 Si, 1 No	Usuario
		Cancelación de pruebas al superar umbrales	¿Permite cancelar la ejecución de las pruebas al superar los umbrales establecidos?	5 Si, 1 No	Usuario
Gestión de resultados (FUN 1.8)	Estadísticas de resultados (FUN 1.8.1)	Estadísticas en tiempo real	¿Genera estadísticas/resultados en tiempo real?	5 Si, 1 No	Usuario
		Estadísticas post-mortem	¿Genera estadísticas luego de la ejecución de las pruebas?	5 Si, 1 No	Usuario
		Notifica superación de umbrales al momento	¿Genera alertas, mensajes ante la cercanía de los umbrales permitidos?	5 Si, 1 No	Usuario
		Generación de métricas	¿Permite establecer indicadores o métricas esperadas?	5 Si, 1 No	Usuario
			¿Permite configurar/parametrizar indicadores o métricas esperadas?	5 Si, 1 No	Usuario
	Análisis de resultados (FUN 1.8.2)	Provee de reportes predeterminados para gestionar resultados	¿Tiene reportes preexistentes para emitir resultados de pruebas?	5 Si, 1 No	Usuario
		Permite modificar los reportes predeterminados	¿Permite modificar reportes de resultados preexistentes?	5 Si, 1 No	Usuario
		Permite construir reportes varios de resultados	¿Permite construir reportes de resultados?	5 Si, 1 No	Usuario
		Provee gráficas predeterminados para gestionar resultados	¿Tiene gráficas preexistentes para emitir resultados de pruebas?	5 Si, 1 No	Usuario
		Permite modificar los gráficos predeterminados	¿Permite modificar gráficos de resultados preexistentes?	5 Si, 1 No	Usuario
		Permite construir gráficos varios de resultados	¿Permite construir gráficos de resultados?	5 Si, 1 No	Usuario

		Análisis de resultados automáticos	¿Proporciona conclusiones en base a los resultados obtenidos?	5 Si, 1 No	Usuario
	Manipulación de resultados (FUN 1.8.3)	Exportar resultados en formatos intercambiables	¿Permite exportar los resultados obtenidos en formato Excel (*.csv)?	5 Si, 1 No	Usuario
			¿Permite exportar los resultados obtenidos en otros formatos como pdf, doc, html, rtf, xml?	5=Completa. 4=Casi todos 3= Med. 2= Pocos 1= No	Usuario
		Formato de generación de Reportes/entregables son similares entre si	¿Son similares los formatos cómo se generan los reportes/entregables de resultados obtenidos?	5=Completa. 4=Casi todos 3= Med. 2= Pocos 1= Ninguno	Usuario
		Disponibilidad de los resultados obtenidos (base de datos)	¿Existe acceso permanente a los resultados obtenidos?	5 Si, 1 No	Usuario
Aplicación objeto de prueba (FUN 1.9)	Interacción con la Aplicación objeto de prueba (FUN 1.9.1)	Sincronización de Scripts de Prueba por nuevas versiones en la Aplicación objeto de prueba	¿Permite sincronizar los Scripts de Prueba seleccionados ante nuevas versiones de la Aplicación objeto de prueba?	5 Si, 1 No	Usuario
		Las pruebas se generan al grabar acciones posibles de los usuarios	¿Permite grabar acciones cotidianas del usuario en la Aplicación objeto de prueba?	5 Si, 1 No	Usuario
		Las pruebas se generan especificando manualmente las acciones posible	¿Permite especificar acciones cotidianas del usuario en la Aplicación objeto de prueba?	5 Si, 1 No	Usuario
		Las pruebas se generan mediante documentos/requerimientos	¿Permite obtener por documentos/requerimientos las acciones cotidianas del usuario en la Aplicación objeto de prueba?	5 Si, 1 No	Usuario
		Ante errores/defectos permite continuar la ejecución	¿Permite continuar la ejecución de la prueba ante defectos en la Aplicación objeto de prueba?	5 Si, 1 No	Usuario
		Capacidad de guardar pruebas para continuar después	¿Permite guardar pruebas para continuar después la ejecución?	5 Si, 1 No	Usuario
	Plataformas de construcción de la Aplicación objeto de prueba (FUN 1.9.2)	Aplicación objeto de prueba basada en entorno Web	¿Se pueden probar aplicaciones construidas en entorno Web?	5 Si, 1 No	Usuario
		Aplicación objeto de prueba basada en aplicación Windows	¿Se pueden probar aplicaciones construidas como aplicación Windows?	5 Si, 1 No	Usuario

Tabla 5 - Sub-características, sub-sub-características y métricas agregadas en la característica "Ajuste a los propósitos" de la categoría Funcionalidad de MOSCA. Fuente: Elaboración Propia.

FUNCIONALIDAD (FUN) - Característica: Precisión (FUN 2)					
Sub-característica	Sub-característica	Métrica	Pregunta	Formulación	Dirigido a
Aceptación de pruebas realizadas (FUN 2.1)	Criterios de aceptación de pruebas (FUN 2.1.1)	Criterios de prueba superada	¿Permite establecer los criterios/condiciones para aceptar/superar la prueba?	5 Si, 1 No	Usuario
		Criterios de prueba no superada	¿Permite establecer los criterios/condiciones para rechazar/no superar la prueba?	5 Si, 1 No	Usuario

Tabla 6- Sub-características, sub-sub-características y métricas agregadas en la característica “Precisión” de la categoría Funcionalidad de MOSCA. Fuente: Elaboración Propia.

FUNCIONALIDAD (FUN) - Característica: Interoperabilidad (FUN 3)					
Sub-característica	Sub-característica	Métrica	Pregunta	Formulación	Dirigido a
Interacción con otras aplicaciones (FUN 3.1)	Funcionamiento con terceros (FUN 3.1.1)	Componentes/plugins que complementan la Herramienta de Prueba	¿Tiene algún(o) componente/plugin que complementa y/o aumenta sus potencialidades?	5 Si, 1 No	Usuario
		Dependencia de componentes/ plugins para funcionar	¿Necesita de algún(o) componente/plugin para poder funcionar?	5 Si, 1 No	Usuario
	Operaciones con terceros (FUN 3.1.2)	Importar scripts de otras aplicaciones/lenguajes	¿Permite importar scripts desde otras Herramienta de Prueba del mismo fabricante?	5 Si, 1 No	Usuario
			¿Permite importar scripts desde otras Herramienta de Prueba de distinto fabricante?	5 Si, 1 No	Usuario
		Importar datos de prueba desde otros sistemas	¿Permite importar datos para alimentar los Datos de Prueba desde archivos planos?	5 Si, 1 No	Usuario
			¿Permite importar datos para alimentar los Datos de Prueba desde archivos Excel?	5 Si, 1 No	Usuario
	Almacenamiento en Base de Datos (FUN 3.1.3)	Manejo de Base de Datos propia	¿Posee Base de Datos propia donde almacena información relevante?	5 Si, 1 No	Usuario
		Manejo de Base de Datos no propias	¿Almacena información relevante en Base de Datos no propietaria?	5 Si, 1 No	Usuario

Tabla 7 - Sub-característica, sub-sub-características y métricas agregadas en la característica “Interoperabilidad” de la categoría Funcionalidad de MOSCA. Fuente: Elaboración Propia.

FUNCIONALIDAD (FUN) - Característica: Seguridad (FUN 4)					
Sub-característica	Sub-sub-característica	Métrica	Pregunta	Formulación	Dirigido a
Control de acceso (FUN 4.1)	Usuarios del sistema (FUN 4.1.1)	Definición de usuarios del sistema	¿Permite configurar usuarios de la Herramienta de Prueba de acuerdo a perfiles?	5 Si, 1 No	Usuario
		Control de acceso a los usuarios	¿Mantiene registro/auditoría del acceso de los usuarios?	5 Si, 1 No	Usuario
		Definición de roles dentro del sistema	¿Permite configurar roles para distinguir entre permisos de usuarios?	5 Si, 1 No	Usuario
			¿Permite implementar roles para distinguir entre permisos de usuarios?	5 Si, 1 No	Usuario
Capacidad de auditoría (FUN 4.2)	Registro de auditoría (FUN 4.2.1)	Registro de acceso a la Herramienta de Prueba	¿Permite registrar el acceso a los usuarios registrados?	5 Si, 1 No	Usuario
		Registro de acciones en la Herramienta de Prueba	¿Permite registrar los cambios a configuraciones sobre la Herramienta de Prueba o sobre proyectos existentes?	5 Si, 1 No	Usuario
	Log de auditoría (FUN 4.2.2)	Mantenimiento de log de auditoría	¿Mantiene los logs de Auditoría sobre tiempos definidos?	5 Si, 1 No	Usuario

Tabla 8 - Sub-característica, sub-sub-características y métricas agregadas en la característica "Seguridad" de la categoría Funcionalidad de MOSCA. Fuente: Elaboración Propia.

USABILIDAD (USA)					
Característica	Sub-sub-característica	Métrica	Pregunta	Formulación	Dirigido a
Capacidad de aprendizaje (USA 2)		Wizard para ciertas funcionalidades	¿Posee ayudas guiadas en la instalación o configuración de ciertas bondades de la Herramienta de Prueba?	5 Si, 1 No	Usuario
		Documentación de la ayuda	¿Ofrece ayuda en línea para las diversas funcionalidades de la Herramienta de Prueba?	5 Si, 1 No	Usuario
			¿Provee manuales de usuario para las diversas funcionalidades de la Herramienta de Prueba?	5 Si, 1 No	Usuario
			¿Provee manual de instalación para la Herramienta de Prueba?	5 Si, 1 No	Usuario
			¿Existen foros de discusión o ayuda en internet?	5 Si, 1 No	Usuario
Interfaz Gráfica (USA 3)		Barra de herramientas customizables	¿La barra de herramientas es personalizable? (elegir opciones a mostrar)		Usuario
		Accesos rápidos a opciones	¿Posee acceso rápido a funciones a través de las teclas?		Usuario
Operabilidad (USA 4)		Lista de posibles valores a ingresar/elegir en campos funcionales	¿Despliega lista de valores tipos para alguna funcionalidad en especial?		Usuario
		Operación de la Aplicación objeto de prueba con la Herramienta de Prueba	¿Cuán fácil es operar la aplicación objeto de prueba dentro de la herramienta de prueba?	5=Muy fácil 4=Fácil 3=Promedio 2=Difícil 1=Muy difícil	Usuario

Tabla 9 - Métricas agregadas en la categoría Usabilidad de MOSCA. Fuente: Elaboración Propia.

El próximo capítulo se corresponde con la fase “Tomar la acción” del Framework metodológico en su actividad número 9, donde se analiza el contexto de la investigación para luego aplicar el método DESMET en aras de evaluar el modelo de referencia propuesto.

CAPÍTULO V: EVALUACION DE LA PROPUESTA

En el presente capítulo se exponen los objetivos, la metodología, el proceso seguido y los resultados de la evaluación de la propuesta del modelo de referencia para la selección de herramientas de prueba por una parte, y por la otra, su respectiva aplicación a un conjunto de herramientas de prueba preseleccionadas del mercado tecnológico, a fin de comprobar su efectividad.

5.1 Objetivos de la evaluación

Con el propósito de proponer un modelo plenamente garantizado, se plantearon los siguientes objetivos a fin de realizar una evaluación eficaz del modelo.

a) Objetivo General

Evaluar la propuesta del modelo de referencia para la selección de herramientas de prueba que soporten en proceso de desarrollo de software en PYMES Venezolanas.

b) Objetivos específicos

- Analizar el contexto donde será evaluado el nuevo modelo.
- Seleccionar un método fiable e imparcial para la evaluación del modelo.
- Evaluar el modelo mediante el método seleccionado.
- Elaborar un análisis de los resultados de la evaluación.

5.2 Metodología seguida

La presente evaluación está enmarcada dentro del Framework metodológico seleccionado para esta investigación, específicamente el método de evaluación DESMET que fue presentado con mayor detalle en el Capítulo III (Marco Metodológico). Igualmente sigue los pasos de Rincón *et al.* (2004).

5.3 Proceso de evaluación

a) Análisis de Contexto

La actividad número 6 del Framework metodológico corresponde al Análisis de Contexto, donde se determinan las especificaciones y acuerdos necesarios para implementar la instanciación de MOSCA, a fin de proporcionar un marco donde será evaluado el nuevo modelo.

El análisis de contexto se realiza de acuerdo a los siguientes criterios propuestos por Kitchenham (1996): el contexto de la evaluación, la naturaleza del impacto esperado, la naturaleza del objeto de evaluación, el alcance del impacto del objeto de evaluación, la madurez del objeto de evaluación, el tiempo de aprendizaje y la madurez de la organización evaluadora.

1) Contexto de la Evaluación

Los elementos que conforman el contexto son:

- Objeto a evaluar: Modelo de referencia para la selección de herramientas de prueba.
- Investigador: Corresponde al autor del modelo y encargado de realizar su evaluación, con su respectivo análisis.
- Organización para implementar y evaluar el modelo: Laboratorio de Investigación en Sistemas de Información (LISI).

2) Naturaleza del impacto esperado

La naturaleza del impacto esperado es cualitativa. El fin de esta evaluación es que el modelo objeto de evaluación se cristalice en una guía para los equipos de desarrollo de software en PYMES Venezolanas al momento de elegir herramientas de prueba para el software que éstas construyan.

3) Naturaleza del Objeto de Evaluación

El objeto de evaluación es de naturaleza sistémica, pues dentro de su contexto debe convertirse en una guía para selección de herramientas de pruebas que permitan garantizar calidad en las aplicaciones que se desarrollen.

4) Alcance del impacto del Objeto de Evaluación

El modelo objeto de evaluación impacta directamente la fase de pruebas del proceso de desarrollo del software que produce el equipo desarrollador en las PYMES desarrolladoras de software, puesto que el uso de herramientas de pruebas se justifica para garantizar la calidad del software que se está construyendo/probando, que se traduce en confianza en los procesos que dicha empresa ejecuta y/o apoya.

5) Madurez del Objeto de Evaluación

Se considera inmaduro el modelo objeto de evaluación, puesto que se encuentra en una fase de propuesta, es decir, no está ampliamente comprobada su efectividad.

6) Tiempo requerido para entender los principios delineados por el Objeto de Evaluación

En una escala de bajo, medio y alto, el objeto de evaluación se considera medio, puesto que los evaluadores deben tener al menos conocimientos previos de los tipos de prueba que pueden realizarse en una aplicación objeto de prueba a fin de exigir prestancias en una herramienta de prueba, por una parte, y por la otra, se debe conocer la perspectiva del Producto de MOSCA.

7) Tiempo requerido para llegar a ser un experto en el uso del Objeto de Evaluación

Se considera medio, puesto que partiendo del conocimiento de los conceptos de pruebas del software, sólo resta conocer MOSCA, en la perspectiva del producto.

8) Madurez de la organización

El equipo evaluador está compuesto por el LISI, quienes poseen una amplia experiencia en evaluación de modelos. De acuerdo al criterio propuesto por Kitchenham (1996), la organización tiene un nivel de madurez 3.

Con éste análisis de contexto se culmina la actividad 6 del Framework metodológico al destacar el entorno de evaluación del modelo propuesto. Igualmente, este es el insumo principal para la aplicación del Método DESMET que sigue a continuación.

b) Aplicación del método DESMET

La actividad 7 del Framework Metodológico corresponde a la Aplicación del Método DESMET. Esta es la última etapa de la Fase tomar la Acción de la metodología de investigación, y es aquí donde se obtendrá un método fiable para la evaluación del modelo.

En el capítulo III se describió el método DESMET y sus nueve métodos de evaluación con un conjunto de criterios que ayudan a seleccionar el método apropiado (Kitchenham, 1996). Es por ello que en la Tabla 9 se resume las condiciones favorables presentes y no presentes en cada uno de los métodos propuestos por DESMET, tomando en cuenta en análisis de contexto anterior.

Método de evaluación	Condiciones favorables	Presente		%
		SI	NO	
Experimento cuantitativo	Tiempo de aprendizaje relativamente corto		X	16,66%
	Deseo en lograr que el modelo para seleccionar herramientas sea independiente del contexto	X		
	Beneficios claramente cuantificables		X	
	El modelo para seleccionar herramientas relacionado con una actividad o tarea simple		X	
	Beneficios directamente mensurables desde las salidas obtenidas		X	
	Disponibilidad de personal para tomar parte en el experimento		X	
Estudio de un caso cuantitativo	Personal con experiencia en mediciones	X		20%
	Beneficios cuantificables en un solo proyecto		X	
	Proceso de desarrollo estable		X	
	Tiempos de evaluación proporcionales con el tiempo de desarrollo de los proyectos		X	
	Beneficios cuantificables antes de retirar los indicadores del modelo para seleccionar Herramienta de Prueba		X	
Encuestas cuantitativas	Beneficios cuantificables en un solo proyecto		X	0%
	Existencia de una base de datos con datos como: productividad y calidad medida por el modelo para seleccionar Herramienta de Prueba		X	
	Experiencia en proyectos haciendo uso del modelo para seleccionar Herramienta de Prueba		X	
Análisis de características por proyección	Período cortos de tiempo para realizar la evaluación	X		50%
	Número grande de características del modelo para seleccionar herramientas de pruebas		X	
Análisis de características por Estudio de Caso	Dificultad para cuantificar beneficios	X		60%
	Beneficios observables en un proyecto determinado		X	
	Población de usuarios del modelo para seleccionar herramientas de pruebas variado		X	
	Proceso de desarrollo estable	X		
	Escalas de tiempo para evaluación proporcional con el tiempo de desarrollo de los proyectos	X		
Análisis de características por experimento	Dificultad para cuantificar beneficios	X		40%
	Período cortos de tiempo para realizar la evaluación	X		
	Tiempos de aprendizaje relativamente cortos		X	
	Beneficios directamente mensurables desde los resultados obtenidos de tareas específicas.		X	

	Población de usuarios del modelo para seleccionar herramientas de prueba variado		X	
Análisis de características por encuesta	Dificultad para cuantificar beneficios	X		25%
	Población de usuarios del modelo para seleccionar herramientas de prueba variado		X	
	Beneficios observables en un solo proyecto		X	
	Experiencia en proyectos haciendo uso del modelo para seleccionar herramientas de prueba		X	
Análisis de efectos cualitativos	Ausencia de procesos de desarrollo estables		X	25%
	Disponibilidad de opiniones expertas en evaluaciones de modelos similares		X	
	Requerimiento para mezclar y combinar las características del modelo para seleccionar Herramienta de Prueba		X	
	Interés en la evaluación del modelo para seleccionar herramientas de prueba	X		
Benchmarking	El modelo para seleccionar Herramienta de Prueba requiere intervención humana dedicada en su aplicación	X		50%
	La salida del modelo para seleccionar Herramienta de Prueba puede ser clasificada de acuerdo a las bondades del modelo		X	

Tabla 10 - Condiciones favorables presentes y no presentes en cada uno de los métodos propuestos por DESMET. Fuente: Elaboración Propia.

De acuerdo a lo expuesto en la tabla anterior, el método que tiene mayor porcentaje de condiciones favorables es el método de “Análisis de características por Estudio de Caso”; por lo tanto, es el que deberíamos usar ya que es el sugerido por DESMET. En la siguiente sección se describen y desarrollan los pasos que conforman dicho método de evaluación.

Método de Análisis de Características por Estudio de Caso

El método Análisis de Características por Estudio de Caso según Kitchenham et al. (1997), consiste en la evaluación de un modelo una vez que este se aplique a un proyecto de software real. Los pasos que componen el modelo están enmarcados en dos grandes procesos, con cada uno de los siguientes pasos:

1) Análisis de Características:

- Definir el alcance de la evaluación
- Definir la base de la evaluación
- Definir roles y responsabilidades
- Definir premisas y restricciones
- Definir escalas de tiempo y esfuerzo requeridos

- Aplicar el procedimiento de evaluación escogido

2) Estudio de Caso

- Seleccionar el método a evaluar
- Identificar el conjunto de características que permiten la evaluación
- Seleccionar el proyecto piloto
- Probar el modelo en el proyecto piloto
- Asignar puntuación a las características a evaluar
- Analizar los resultados obtenidos y realizar un reporte de la evaluación

Con este análisis, se culmina la actividad siete del Framework metodológico, terminando con la fase “Tomar la acción” y provee los insumos necesarios para ejecutar la subsiguiente fase “Evaluar”, por lo que en la siguiente sección se desarrolla la actividad 8 “Evaluación a través del método arrojado por DESMET”.

c) Evaluación del Modelo de referencia basado en características para la selección de herramientas de prueba

En esta sección se realiza la evaluación del modelo de referencia desde el proceso general correspondiente al “Análisis de Características” hasta el específico “Estudio de Caso”, de acuerdo a las premisas detalladas en la sección anterior.

(c.1) Proceso Análisis de Características

Los pasos para el análisis de Características dentro del primer proceso se desarrollan a continuación.

Paso 1. Definir el alcance de la evaluación

El presente trabajo de investigación contempla la evaluación de una propuesta de un modelo de referencia basado en características para la selección de herramientas de pruebas dentro de un equipo encargado de velar por las pruebas del software para apoyar el proceso de desarrollo de software en PYMES Venezolanas.

Paso 2. Definir la base de la evaluación

La presente evaluación está basada en el estudio bibliográfico y metodológico documentado en Capítulos anteriores.

Paso 3. Definir roles y responsabilidades

Los roles relacionados con la actividad de evaluación son los siguientes:

- ✓ El patrocinante: Está constituido por el Laboratorio de Investigación de Sistemas de Información de la Universidad Simón Bolívar (LISI) expertos en la propuesta y evaluación de modelos en el ámbito del desarrollo del software, que suministran su experiencia en esta investigación.
- ✓ El evaluador: es el responsable de llevar a cabo la evaluación, en este caso se trata del autor del presente trabajo. Sus responsabilidades son: preparación del plan de evaluación, identificación del objeto de la evaluación, identificación y definición de las características a evaluar en el modelo, recopilación y análisis de los resultados obtenidos y preparación del informe final y recomendaciones.
- ✓ Usuarios del modelo: son usuarios potenciales del modelo de referencia, para los efectos del presente trabajo este rol debe ser desempeñado por las PYMES Venezolanas desarrolladoras de Software, tomando en consideración las opiniones de personal experto en el área de sistemas.

Paso 4. Definir premisas y restricciones

El proceso de evaluación se encuentra sujeto a los siguientes factores:

- ✓ Disponibilidad de tiempo y dedicación por parte del evaluador, el cual realizará el proceso como parte fundamental de su trabajo de grado.
- ✓ Duración máxima para la realización de la evaluación, la cual está determinada por el tiempo estipulado para la culminación del trabajo de grado.
- ✓ Disponibilidad del ambiente requerido para realizar la evaluación, constituida por el tiempo y atención prestada por el personal de la organización patrocinante.
- ✓ Los usuarios del modelo deben ser personas con experiencia en el tema de pruebas de software.

Paso 5. Definir escalas de tiempo y esfuerzo requerido

A continuación se presentan las actividades necesarias para especificar las actividades generales que se realizarán en pro de completar el proceso de evaluación y el tiempo que se dedicará a cada actividad.

- ✓ Elaboración del plan de evaluación: 2 días
- ✓ Elaboración de los instrumentos de evaluación: 10 días
- ✓ Solicitud y aprobación del proceso de evaluación por parte de la organización 2 días.
- ✓ Evaluación y ponderación de las características: 3 días

- ✓ Análisis de resultados: 3 días
- ✓ Elaboración de conclusiones y recomendaciones: 5 días

Paso 6. Aplicar procedimiento de Evaluación escogido

Una vez cumplidos los pasos generales correspondientes al Análisis de Características se procede a llevar a cabo el proceso relativo al Estudio de Caso, actividad desarrollada a continuación.

(c.2) Proceso Estudio de Caso

En la presente sección se detallan las actividades que deben llevarse a cabo durante un estudio de caso.

Actividad 1. Seleccionar el método a evaluar

Como se presentó con anterioridad el objeto de evaluación es el modelo de referencia para la selección de herramientas de prueba, siendo el método seleccionado el análisis de características por estudio de caso.

Actividad 2. Identificar el conjunto de características que permiten la evaluación

Para poder evaluar de una forma efectiva el modelo propuesto con cada una de sus categorías, características y métricas, es necesario establecer un conjunto de aspectos a evaluar que van desde lo más general: el modelo MOSCA; a lo más específico: las métricas. Estos aspectos son los que Kitchenham y Jones (1997) llaman características. Kitchenham y Jones (1997) señalan que existen dos tipos de característica:

- **Simples:** se utilizan cuando la característica se encuentra presente o ausente en el contexto de evaluación, esto es determinado mediante una escala nominal (Si, No)
- **Compuestas:** son utilizadas cuando la existencia o conformidad de una característica puede ser medida con una escala ordinal.

De acuerdo a lo anterior, para que la evaluación sea efectiva, debe contemplarse los aspectos del modelo que van desde lo más general (constituido por el modelo en sí) hasta los más específico (las métricas), abarcando sus categorías, características, sub características y sub-sub-características que lo conforman. Para efectos de la presente evaluación se utilizaran sólo características simples para obligar al evaluador a establecer un criterio claro de aceptación.

Las características más generales evalúan el modelo en un nivel macro, es decir, el subárbol Producto como un todo y sus Categorías. Las características generales y su definición conceptual se presentan en la Tabla 11.

Característica General	Descripción	Escala
Pertinencia del modelo propuesto	Se refiere a si el modelo propuesto es pertinente o no dentro del proceso de especificación de la calidad del software.	1: significa que el modelo propuesto es pertinente 0: significa que el modelo propuesto no es pertinente.
Complejidad de las categorías involucradas	Se refiere a si las Categorías: Funcionalidad, Usabilidad, Fiabilidad y Eficiencia dan cobertura total a los aspectos de calidad de la especificación del software.	1: significa que el modelo propuesto es completo en cuanto a las categorías utilizadas. 0: significa que de acuerdo al contexto existen categorías que deberían ser consideradas en el modelo propuesto.
Adecuación al contexto	Se refiere a si la especificación del modelo propuesto es adecuada en el contexto de la evaluación.	1: significa que el modelo propuesto se adecúa al contexto de la evaluación 0: significa que el modelo propuesto no es adecuado en el contexto de la evaluación
Precisión del nivel de calidad especificado por el modelo propuesto	Se refiere a si la calidad especificada por el modelo propuesto en el proyecto piloto fue precisa	1: significa que el nivel de calidad especificado es preciso 0: significa que el nivel de calidad especificado es impreciso

Tabla 11- *Características Generales a evaluar para el modelo propuesto.*
Fuente: Sosa (2005).

Luego de especificadas las características generales, es necesario establecer un conjunto de características que permitan evaluar las métricas del modelo. Las características específicas a considerar se presentan en la Tabla 12.

Característica Específica	Descripción	Escala
Pertinencia de la métrica	Se refiere a si una métrica es adecuada para medir la existencia o no de la característica donde se encuentra	1: significa que la métrica es pertinente 0: significa que la métrica no es pertinente
Factibilidad de la métrica	Se refiere a si es factible medir la característica propuesta en la métrica dentro del contexto de evaluación	1: significa que la métrica es factible 0: significa que la métrica no es factible
Nivel de profundidad	Se refiere a si la métrica a verificar tiene el nivel de profundidad adecuado para que el resultado sea relevante	1: significa que la métrica tiene el nivel de profundidad adecuado 0: significa que se requiere una mayor profundidad en la métrica
Escala de la métrica	Se refiere a si la escala propuesta es adecuada para medir la métrica	1: significa que la escala es adecuada 0: significa que la escala no es adecuada

Tabla 12 – *Características de las Métricas a evaluar el modelo propuesto.* Fuente: Sosa (2005).

Una vez establecidas las características generales y específicas a aplicar para evaluar el modelo propuesto se requiere especificar el criterio de aceptación, constituido por:

- Para que el modelo propuesto se considere aceptable todas sus Categorías deben ser aceptables con más del 75%.
- Para que una Característica dentro de una Categoría se considere aceptable debe obtener el 75% o más del promedio de todas las respuestas para dicha característica.
- Para que una Métrica dentro de una Característica se considere aceptable debe obtener el 75% o más del promedio de todas las respuestas para dicha métrica.

Actividad 3. Seleccionar el Caso de estudio

La siguiente actividad dentro del proceso que se debe seguir para realizar el Estudio de Caso es precisamente la selección del caso de estudio.

El estudio de caso está constituido por la selección de herramientas de prueba de tipo no funcional, específicamente herramientas que permitan hacer pruebas automatizadas para el nivel de prueba de sistema con los tipos de desempeño y de carga, como se expresó en la sección 4.2 del capítulo IV (Árbol de clasificación de herramientas de pruebas automatizadas instanciado).

Cabe destacar que las pruebas de rendimiento pueden ayudar a responder preguntas como las siguientes:

- *¿Cuántos usuarios puede soportar el sistema en condiciones normales?*
- *¿Existen situaciones en las que el rendimiento del sistema disminuya repentinamente en condiciones normales?*
- *¿Cuál es el rendimiento del sistema cuando se sobrepasan las condiciones normales?*
- *En el peor de los casos, ¿el sistema se degrada de una forma paulatina o se bloquea completamente?*

Es por ello que el objetivo de este caso de estudio es usar el modelo propuesto para seleccionar herramientas de prueba que satisfagan las necesidades anteriormente descritas, con el fin de tomar decisiones de adquisición de herramientas acertadas y

que puedan apoyar el proceso de desarrollo de software ‘in house’. Bajo esta premisa, como cada herramienta hace uso del modelo, se decidió que el estudio de caso está conformado por cada herramienta a evaluar, es decir, cada caso de estudio se corresponde con cada herramienta objeto de evaluación a fin de seleccionar la mejor.

En la Tabla 13 se identifican un subconjunto de herramientas de prueba elegidas por ser las más reconocidas en el ámbito tecnológico, por la empresa que la representa y de acuerdo a los datos obtenidos por el Automated Testing Institute (ATI, 2010). Dicha tabla especifica la versión actual en el mercado, fabricante, tiempo de instalación y dónde encontrar información al respecto. Cabe destacar, que de un subconjunto de herramientas de prueba pre-seleccionadas, la evaluación del modelo finalmente se realizó en base a cuatro herramientas que disponían de versión de prueba limitada (en número de días) descargables desde su Website, lo que permitió usar como método de aprendizaje, la manipulación de la versión de prueba disponible aplicada a un sistema en entorno web de acuerdo con cada una de las funcionalidades que éstas herramientas ofrecían.

Nro	Nombre Herramienta	Versión	Fabricante	Tiempo de Instalación	Disponible en
1	HP LoadRunner software	9.5	Hewlett-Packard	50 min.	https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-11-126-17%5e8_4000_100
2	SilkPerformer	No especifica	Borland	40 min.	http://www.borland.com/us/products/silk/silkperformer/index.html
3	IBM/Rational Performance Tester	8.1	IBM	45 min.	http://www-01.ibm.com/software/awdtools/tester/performance/index.html?S_TACT=105AGX23&S_CMP=DWNL
4	SoapUI Pro	3.5	Eviware	15 min.	http://www.soapui.org/features.html

Tabla 13 - Herramientas de prueba del mercado seleccionadas para cada estudio de caso. Fuente: Elaboración Propia.

Una breve descripción de cada una de las herramientas de pruebas, se presentan a continuación:

- Estudio de Caso 1, **HP LoadRunner software**: Permite realizar pruebas de Rendimiento determinando los cuellos de botella en las aplicaciones y si está acorde a los requerimientos de performance exigidos.
- Estudio de Caso 2, **SilkPerformer**: Permite realizar pruebas automatizadas de carga y rendimiento.

- Estudio de Caso 3, **IBM/Rational Performance Tester**: Identifica la presencia y causa de los cuellos de botella en el rendimiento del sistema.
- Estudio de Caso 4, **SoapUI Pro**: Permite realizar pruebas funcionales, de carga y conformidad de Web Services. Es una herramienta Open source.

Actividad 4. Probar el modelo propuesto en cada Estudio de caso

El tesista realizó la prueba del modelo propuesto basado en los cuatro estudios de caso correspondiente a cada una de las herramientas pre-seleccionadas antes descritas, aplicando el cuestionario resultante del modelo propuesto condensado, con los aspectos tomados y propuestos de MOSCA (Anexo VI), cuyos resultados de la aplicación de la prueba con su correspondiente análisis se presentan en el siguiente capítulo.

Actividad 5. Asignar puntuación a las características a evaluar

Para la recolección de esta información se registraron cada uno de los puntajes correspondiente a la dupla métrica-pregunta para cada uno de los estudios de caso, cuyos valores pueden observarse en el Anexo VI. Luego de finalizada esta actividad, se procedió a realizar el análisis respectivo presentado en la sección siguiente.

Actividad 6. Analizar resultados obtenidos y preparar reporte de evaluación

La sexta y última actividad del proceso Estudio de Caso corresponde al análisis de resultados, los cuales se visualizan en el siguiente capítulo VI, para así dar fin a la instanciación y evaluación del modelo propuesto.

CAPÍTULO VI: ANÁLISIS E INTERPRETACIÓN DE RESULTADOS

En este capítulo se presentan los resultados del proceso de evaluación presentado en el capítulo anterior, con su respectivo análisis, el cual va desde la aplicación del modelo propuesto en los estudios de caso escogidos hasta la evaluación del modelo propiamente dicho a fin de determinar su aplicabilidad, luego de la experiencia obtenida. Es por ello que la presentación de los resultados se han dividido en 3 secciones que son las siguientes:

- Sección 6.1: Evaluación del modelo en los estudios de caso (herramientas seleccionadas)
- Sección 6.2: Evaluación de las características generales del modelo
- Sección 6.3: Evaluación de las métricas del modelo

6.1 Evaluación del modelo en los estudios de caso

En esta sección se expone el resultado de la evaluación en cada uno de los estudios de caso seleccionados, compuesto por cuatro (4) herramientas del mercado tecnológico enmarcadas dentro del árbol de clasificación propuesto, los cuales fueron evaluadas usando el modelo propuesto.

El Anexo VI se muestra el detalle de la evaluación del modelo propuesto realizada por el autor luego de obtener un corto y aceptable aprendizaje manipulando cada una de las herramientas por una parte, y por la otra, respondiendo a cada una de las preguntas de las métricas establecidas en el modelo propuesto, de acuerdo a la columna “Formulación”. En la tabla 14 se muestra un resumen de cada uno de los niveles que componen el modelo aplicados para cada una de las cuatro herramientas evaluadas, desde el nivel más específico hasta el más general.

A continuación, se hará un análisis detallado del nivel “Características”, por ser el que está presente en todas las categorías del modelo, esto es el nivel de Sub-características y sub-sub-característica, solo está presente en la Categoría Funcionalidad, no así en el resto de las categorías del modelo: Usabilidad, Eficiencia y Fiabilidad. En las figuras 21, 22, 23 y 24 se presentan los resultados de los estudios de caso por categoría para el nivel del modelo Características.

Categoría	Característica	Sub-característica	Sub Sub-característica	% Sub-característica				% Sub-característica				% Característica				% Categoría			
				HP (1)	Silk (2)	IBM (3)	Soap(4)	HP (1)	Silk (2)	IBM (3)	Soap(4)	HP (1)	Silk (2)	IBM (3)	Soap(4)	HP (1)	Silk (2)	IBM (3)	Soap(4)
FUNCIONALIDAD (FUN)	Ajuste a los propósitos (FUN 1)	Herramienta de Prueba (FUN 1.1)	Herramienta de Prueba (Estudio de Caso) -->	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	83,23%	80,38%	74,05%	80,70%	85,75%	76,40%	71,50%	75,70%
				100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%
				100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%
				100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%
				100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%
				100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%
				100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%
				100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%
				100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%
				100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%
				100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%
				100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%
				100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%
				100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%
USABILIDAD (USA)	Precisión (FUN 2)	Aceptación de pruebas realizadas (FUN 2.1)	Criterios de aceptación de pruebas (FUN 2.1.1)	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	91,67%	91,67%	91,67%	87,50%	75,96%	54,81%	82,69%	77,88%
				100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	92,31%	69,23%	30,77%	69,23%	75,00%	82,14%	96,43%	96,43%
				100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	94,44%	41,67%	94,44%	33,33%	75,00%	68,75%	75,00%	75,00%
				100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	95,00%	0,00%	90,00%	0,00%	75,00%	75,00%	75,00%	75,00%
				100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	93,33%	60,00%	36,67%	76,67%	75,00%	75,00%	75,00%	75,00%
				100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	93,75%	93,75%	100,00%	75,00%	75,00%	75,00%	75,00%	75,00%
				100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	93,75%	93,75%	100,00%	75,00%	75,00%	75,00%	75,00%	75,00%
				100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	93,75%	93,75%	100,00%	75,00%	75,00%	75,00%	75,00%	75,00%
				100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	93,75%	93,75%	100,00%	75,00%	75,00%	75,00%	75,00%	75,00%
				100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	93,75%	93,75%	100,00%	75,00%	75,00%	75,00%	75,00%	75,00%
				100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	93,75%	93,75%	100,00%	75,00%	75,00%	75,00%	75,00%	75,00%
				100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	93,75%	93,75%	100,00%	75,00%	75,00%	75,00%	75,00%	75,00%
				100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	93,75%	93,75%	100,00%	75,00%	75,00%	75,00%	75,00%	75,00%
				100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	93,75%	93,75%	100,00%	75,00%	75,00%	75,00%	75,00%	75,00%
				100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	93,75%	93,75%	100,00%	75,00%	75,00%	75,00%	75,00%	75,00%
FIABILIDAD (FIA)	Comprensión (USA 1)	Capacidad de Aprendizaje(USA2)	Interfaz Gráfica (USA 3)	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	56,25%	43,75%	75,00%	87,50%	75,96%	54,81%	82,69%	77,88%
				100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	75,00%	60,71%	82,14%	96,43%	75,00%	68,75%	75,00%	75,00%
				100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	78,57%	28,57%	100,00%	50,00%	75,00%	68,75%	75,00%	75,00%
				100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	84,38%	78,13%	71,88%	81,25%	75,00%	68,75%	75,00%	75,00%
EFICIENCIA (EFI)	Operabilidad (USA 4)	Tolerancia a Fallos (FIA 2)	Recuperación (FIA3)	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	66,67%	66,67%	75,00%	75,00%	81,25%	75,00%	81,25%	81,25%
				100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	75,00%	62,50%	62,50%	75,00%	81,25%	75,00%	81,25%	81,25%
				100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	87,50%	87,50%	100,00%	87,50%	81,25%	75,00%	81,25%	81,25%
				100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	87,50%	87,50%	100,00%	87,50%	81,25%	75,00%	81,25%	81,25%

Tabla 14 - Evaluación del modelo propuesto en el Proyecto Piloto. Fuente: Elaboración Propia.

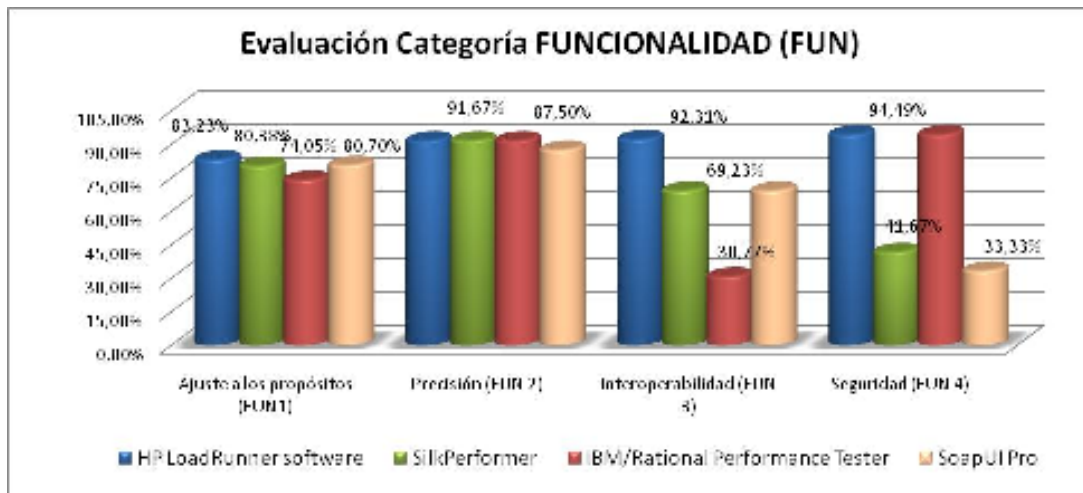


Figura 22 - Resultados evaluación Características de la Categoría “Funcionalidad” en Estudio de casos. Fuente: Elaboración propia.

En la Figura 21 se observan cada una de las características de la categoría Funcionalidad, detallándose lo siguiente:

- En FUN1- “Ajuste a los propósitos”, se muestra que tres de los cuatro estudios de casos (casos 1, 3 y 4) se consideran Aceptables, porque su porcentaje de aceptación superan el 75% mínimo establecido. El caso 2 (SilkPerformer) no se considera aceptable debido a que de acuerdo a la tabla 14, específicamente en la presentación de resultados, como se muestran en las sub-sub-características Salidas de pruebas (FUN 1.6.2) y Análisis de resultados (FUN 1.8.2) presenta ciertas debilidades, pero no está muy alejado de ser aceptado (74,05%), por lo que si se ajusta a los propósitos para lo que fue creado.
- En cuanto a “Precisión”-FUN 2 todos los estudios de caso se consideran Aceptables (mayor al 75%) con un 91,67% para los casos 1, 2 y 3 y para el caso 4 un 87,50%, por presentar debilidades en los resultados que genera.
- “Interoperabilidad”- FUN 3, fue considerada Aceptable solo en el estudio de caso 1 (HP LoadRunner) con un 92,31%. El resto de los casos, no se acercan al porcentaje mínimo requerido (75%) porque en la Interacción con otras aplicaciones (FUN 3.1), muestran debilidades aunado a que el estudio de caso 3 es muy bajo porque tiende a ser exclusiva en cuanto a sus operaciones.
- En FUN 4- “Seguridad” se observa que solo los casos 1 y 3 son Aceptables con un 94,44%, mientras que los otros dos casos (2 y 4) están muy alejados de ser aceptados, puesto que no tienen manejo de usuarios del sistema y por ende, no pueden auditar las acciones de los mismos y/o resguardar los proyectos

configurados según lo que muestra la Tabla 14 en las sub-características Control de acceso FUN 4.1 y Capacidad de auditoría FUN 4.2.

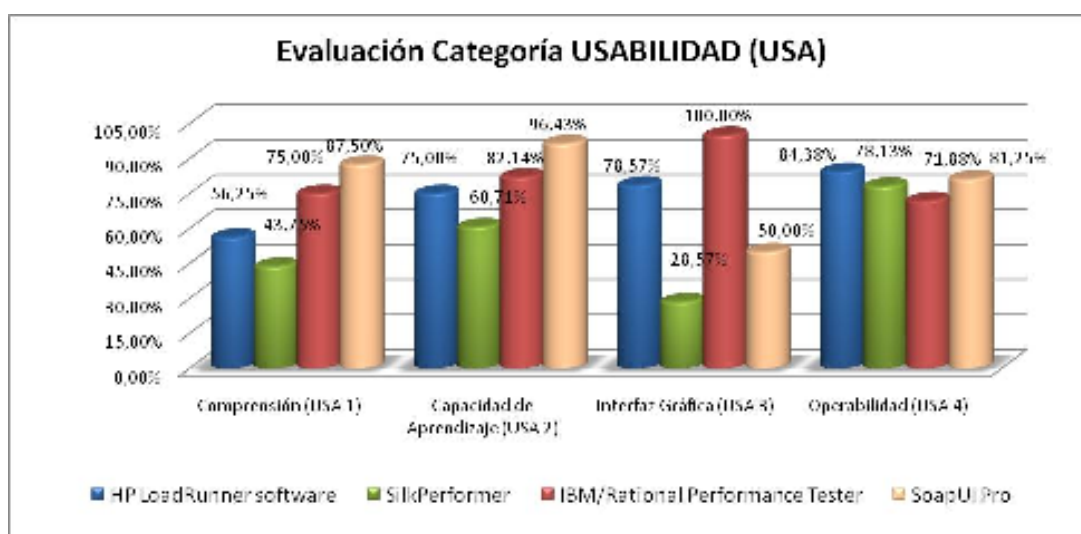


Figura 23 - Resultados evaluación Características de la Categoría “Usabilidad” en Estudio de casos.
Fuente: Elaboración propia.

La figura 22, muestra los resultados obtenidos en la categoría Usabilidad, donde cada Característica evaluada responde a lo siguiente:

- USA 1-“Facilidad Comprensión” muestra que los Estudios de casos 1 y 2 están por debajo del porcentaje de Aceptación, ya que las herramientas presentan dificultades para comprenderla por parte del usuario final.
- En “Capacidad de aprendizaje”- USA2, la única herramienta que no se consideró Aceptable fue SilkPerformer (Estudio de Caso 2), pues presenta debilidades en la documentación y en wizard para ejecutar ciertas funciones de la herramienta. Los demás Estudios de Caso (1,3 y 4) si se consideraron Aceptables.
- Para la característica USA 3- “Interfaz Gráfica” nuevamente el estudio de Caso 2 presenta debilidades en la localización y acceso rápido a ciertas opciones, además que es limitado la capacidad de tener una interfaz personalizable y el diseño de pantallas no satisface. Por su parte, el estudio de Caso 4 presenta las mismas debilidades del Caso 2 excluyendo la parte del diseño de pantallas. Por lo tanto los estudios 2 y 4 no se consideraron Aceptables en la característica evaluada, USA 3.
- En cuanto a la característica “Operabilidad” – USA 4, el Estudio de Caso 3 se encontró débil al momento de operarlo pues no presenta funcionalidades que hagan más fácil su uso. Seguidamente está el Estudio de Caso 2, pero al igual que los Casos 1 y 4, se consideraron Aceptables para la evaluación en el modelo.

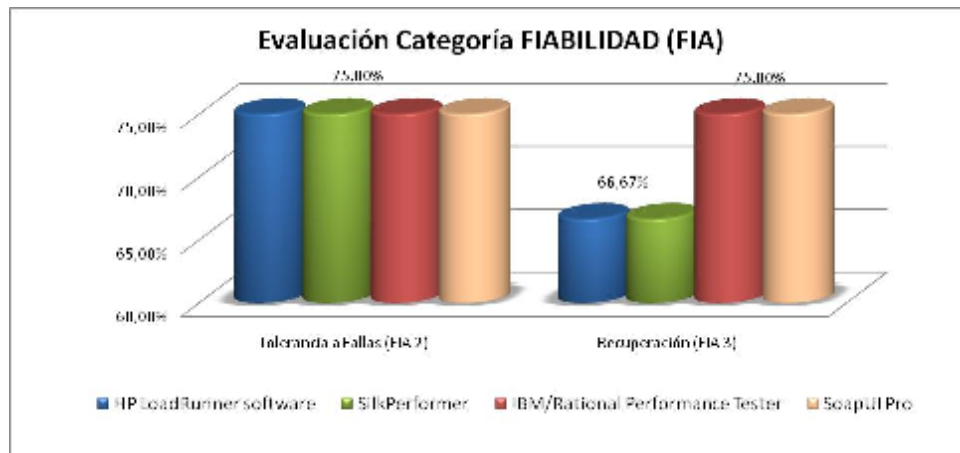


Figura 24 - Resultados evaluación Características de la Categoría “Fiabilidad” en Estudio de casos.
Fuente: Elaboración propia.

La Figura 23 muestra los resultados de la categoría Fiabilidad, en la que los resultados se muestran parejos entre sí. Los detalles son mostrados a continuación:

- Para la Característica FIA 2 – “Tolerancia a fallas”, todos los Estudios de caso se consideraron Aceptables aunque con el mínimo de porcentaje permitido; esto se debe a que presentan debilidades en evitar fallas en su operación aún cuando ésta se realice de forma incorrecta.
- En la característica “Recuperación”- FIA 3, los estudios de casos 1 y 2 no alcanzaron el porcentaje mínimo de Aceptación debido a que no tienen una buena capacidad de reiniciarse ante alguna caída de la herramienta, situación que se repite en los casos 3 y 4 pero en menor proporción, lo que les permitió considerarla “Aceptable” con el porcentaje mínimo.

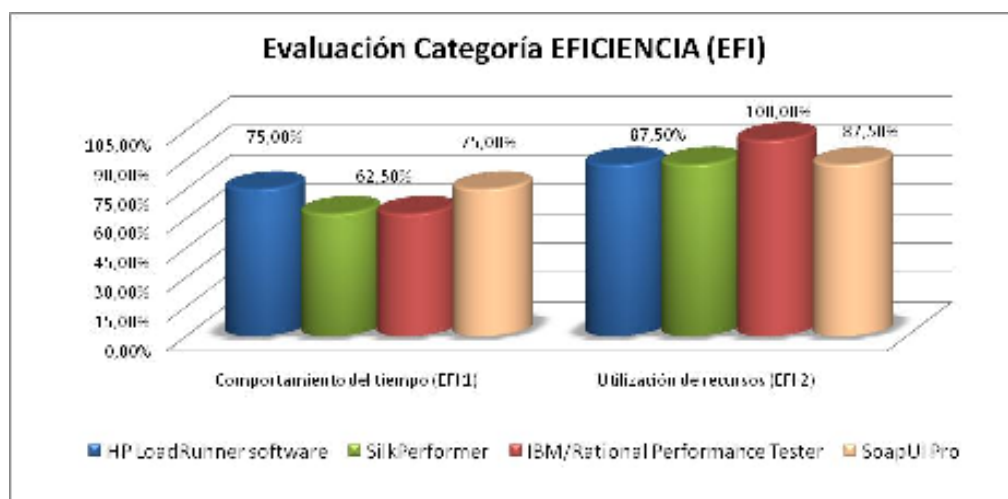


Figura 25 - Resultados evaluación Características de la Categoría “Eficiencia” en Estudio de Casos.
Fuente: Elaboración propia.

Como se observa en la Figura 24, los resultados de la evaluación de las características que conforman la categoría Fiabilidad, se detallan así:

- En “Comportamiento del tiempo” – EFI 1, los Estudios de Caso 2 y 3, no lograron el porcentaje mínimo de Aceptación por considerarse lentos al momento de dar los resultados luego de ser sometido a ciertas funciones, con una tasa de respuesta alta. Ahora bien, los Estudios de caso 1 y 4 se consideraron Aceptables por llegar el porcentaje mínimo requerido debido a que presentan las mismas fallas de los otros Estudios de Caso, pero en menor proporción, es decir, son más rápidos.
- La característica EFI 2 - “Utilización de recursos” fue superada por todos los Estudios de Caso, por lo que se consideraron Aceptables, pues muestran se equitativos al momento de administrar los recursos de la máquina donde estos operan.

Luego evaluar cada una de las características del modelo propuesto que componen las cuatro categorías elegidas en todos los Estudios de Caso correspondientes, se presenta a continuación la evaluación general de todos los Estudios de Caso en el nivel “Categorías” del modelo propuesto.

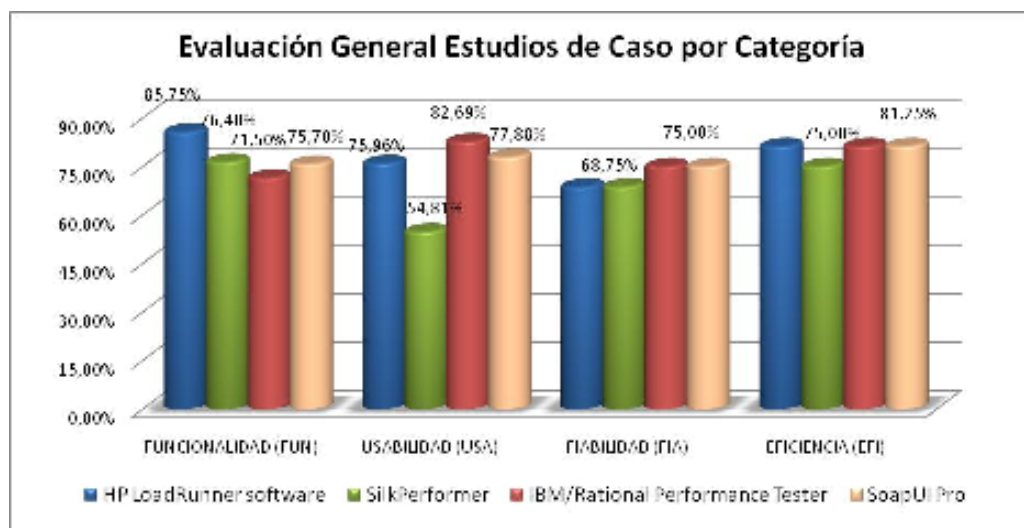


Figura 26 – Resultados de la evaluación de los estudios de caso en cada categoría del modelo propuesto. Fuente: Elaboración Propia.

En la figura 25, se observa cómo resultó ser el nivel de aceptación de cada uno de los estudios de caso para cada una de las categorías del modelo propuesto, resultados que están directamente relacionados con el nivel Característica detallado en

la sección anterior y en los demás niveles del modelo que aplica (Categoría Funcionalidad) y a continuación su respectivo análisis tomando en cuenta la Tabla 14.

- *Funcionalidad:* La única herramienta que obtuvo un porcentaje menor al 75% esperado es IBM/Rational Performance Tester que corresponde al estudio de caso 3, razón por la cual no se considera Aceptable funcionalmente hablando, pero se acerca mucho al porcentaje esperado con un 71,50%. La característica que hace bajar su porcentaje es la Interoperabilidad FUN 3 (ver Tabla 14). Los demás estudios de caso superaron el 75% de aceptación, siendo el de mayor porcentaje el estudio de caso 1 (HP Load Runner) con 85,75%, por lo que cumple en gran medida el con los requisitos prácticos de una herramienta para pruebas no funcionales. Los casos de estudio 2 y 3 tienen porcentaje menor porque la característica Seguridad FUN 4 no está suficientemente presente en las herramientas.
- *Usabilidad:* Tres de los cuatro estudios de casos resultaron aceptables en esta categoría siendo el primero el estudio de caso 3 con 82,69% seguido del caso 4 con 77,88% y muy cerca del mínimo permitido está el estudio de caso 1 con 75,96%, por no exponer una gran facilidad de comprensión (USA 1). El estudio de caso 2 Silk Performer, resultó no aceptable porque revela gran debilidad frente a sus competidores en la Interfaz gráfica USA 3 y en la facilidad de comprensión USA 1, condiciones que puede traducirse en elevación de costos y tiempo invertidos en entrenamiento de quienes harán uso de la herramienta.
- *Fiabilidad:* Esta categoría evidencia gran debilidad entre los estudios de caso, pues aunque los casos 3 y 4 resultaron Aceptables, su porcentaje estuvo en el límite, 75% respectivamente. Los casos 1 y 2 presentaron un porcentaje de 68,75% de aceptación debido a que su capacidad de recuperación FIA 3 ante cualquier evento es baja, lo que pudiera traducirse en un atraso en las pruebas que se tengan planificadas.
- *Eficiencia:* Es la única categoría de la evaluación en las que todos los estudios de caso resultaron aceptables. Sin embargo, cabe destacar que el estudio de caso 2 obtuvo el porcentaje mínimo de aceptación (75%) debido a su baja calificación en el comportamiento del tiempo (EFI 1).

De acuerdo al análisis anterior y tomando en cuenta el algoritmo de aplicación de MOSCA, los estudios de caso en los que se debe continuar la evaluación, son aquellas que hayan superado la categoría Funcionalidad, por lo que la única herramienta que se obvia en este estudio es IBM/Rational Performance Tester, pero se continúa con los casos de estudio 1, 2 y 4. Ahora bien, el Algoritmo de MOSCA plantea que para medir el nivel de calidad de un software se debe tomar en cuenta el nivel de satisfacción (porcentaje de aceptación) de las categorías incluidas. Sin embargo, este nivel de calidad es determinado por 3 categorías y no se corresponde con las cuatro categorías que comprenden el modelo propuesto. Es por ello que se requiere adaptarlo a la presente investigación.

Propuesta de adaptación del algoritmo de MOSCA

La categoría *Funcionalidad* se mantiene como la determinante en continuar o no con la evaluación de un producto. Por otra parte, dado las condiciones de aplicación del modelo propuesto basado específicamente en herramientas de pruebas de desempeño y carga, la categoría *Eficiencia* pasa a tomar gran relevancia en el modelo, puesto que las prestaciones de una herramienta de este tipo debe ser tal que no entorpezca el trabajo la ejecución de las pruebas de la aplicación objeto de pruebas, es decir, que no afecte el desempeño ni los resultados de la misma. Es por ello, que se propone que cuando hayan cuatro categorías a evaluar en la aplicación de MOSCA y dos de ellas sean determinantes para continuar o no con la evaluación de producto, el nivel de calidad del producto sea determinado de acuerdo las categorías satisfechas o no satisfechas reflejadas en la Tabla 15.

Funcionalidad + categoría “condicional” evaluada	Segunda categoría evaluada	Tercera categoría evaluada	Nivel de calidad del producto de software
<i>Satisfecha</i>	<i>No satisfecha</i>	<i>No satisfecha</i>	<i>Básico</i>
<i>Satisfecha</i>	<i>Satisfecha</i>	<i>No satisfecha</i>	<i>Intermedio</i>
<i>Satisfecha</i>	<i>No satisfecha</i>	<i>Satisfecha</i>	<i>Intermedio</i>
<i>Satisfecha</i>	<i>Satisfecha</i>	<i>Satisfecha</i>	<i>Avanzado</i>

Tabla 15 – Nivel de calidad del producto propuesto respecto a categorías satisfechas para el producto adaptado al modelo propuesto. Fuente: Adaptado de Mendoza et al. (2005)

Como se mencionó anteriormente, la categoría Eficiencia, se convierte en la categoría “condicional” para la aplicación del modelo propuesto. Bajo esta premisa,

los casos de estudio que cumplen con la condición indicada son los mismos que hasta ahora se mantienen en la evaluación, esto es, los casos 1, 2 y 4, dado que son aceptables tanto en las categorías Funcionalidad como en Eficiencia.

Finalmente se aplica lo propuesto en la tabla 15 para la presente investigación, quedando el nivel de calidad del producto que representa cada uno de los estudios de caso que se mantienen en la evaluación de acuerdo a lo previsto en la Tabla 16.

Estudio de Caso		Funcionalidad + Eficiencia	Usabilidad	Fiabilidad	Nivel de calidad
1 - HP Load Runner	Porcentaje obtenido	85,75% y 75%	75,96%	68,75%	Intermedio
	Criterio de satisfacción	<i>Satisfecha</i>	<i>Satisfecha</i>	<i>No satisfecha</i>	
2 - Silk Performer	Porcentaje obtenido	76,4% y 75%	54,81%	75%	Intermedio
	Criterio de satisfacción	<i>Satisfecha</i>	<i>No Satisfecha</i>	<i>Satisfecha</i>	
4 - SoapUI Pro	Porcentaje obtenido	75,7% y 81,25%	77,88%	75%	Avanzado
	Criterio de satisfacción	<i>Satisfecha</i>	<i>Satisfecha</i>	<i>Satisfecha</i>	

Tabla 16 - Nivel de calidad de los estudios de caso respecto a categorías satisfechas para el producto adaptado al modelo propuesto. Fuente: Elaboración propia.

De acuerdo a lo demostrado en la Tabla 16, el estudio de caso que presenta un nivel de calidad **Avanzado** es el caso 4 representado por la herramienta SoapUI Pro del fabricante Eviware con tecnología Open source, ya que todas las categorías del modelo fueron satisfechas en la evaluación realizada por el autor. Los estudios de caso 1 y 2, HP Load Runner y Silk Performer respectivamente, arrojaron un nivel de calidad del producto **Intermedio**. Por lo tanto, las PYMES desarrolladoras de software en Venezuela pueden hacer uso de esta herramienta de prueba sin acarrear mayores costos imputados en los productos por ellas desarrollados.

A continuación se realiza la evaluación del modelo propuesto.

6.2 Evaluación de las características generales

Las características generales del modelo evaluadas se corresponden con los primeros niveles de MOSCA, que son categorías, características, sub-características y sub-sub-características. Sin embargo, se presenta el análisis sólo de los niveles “Características” y “Categorías”, por ser los comunes en todas las categorías del modelo y resumen el comportamiento de las niveles siguientes. En el anexo VI se presenta el detalle de los resultados de todos los niveles evaluados.

6.2.1 Evaluación de las categorías del modelo

La primera evaluación está relacionada a las categorías del modelo (Nivel 1 de MOSCA) que son Funcionalidad, Usabilidad, Fiabilidad y Eficiencia. Los resultados se muestran en el gráfico (Figura 26).

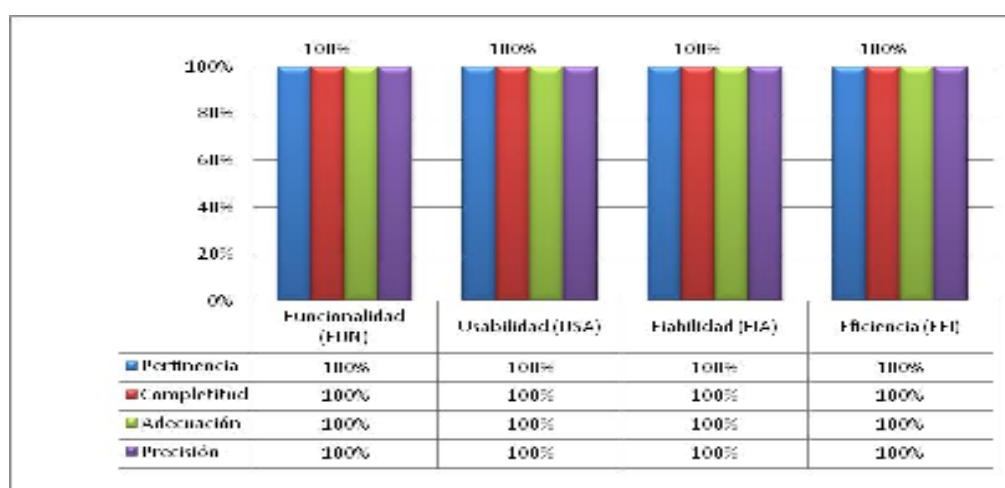


Figura 27- Resultados de la evaluación de “Categorías” del modelo propuesto.
Fuente: Elaboración propia.

Como puede observarse en la Figura 26, todas las categorías alcanzaron el máximo porcentaje en la evaluación (100%). Esto demuestra que las categorías Funcionalidad, Usabilidad, Fiabilidad y Eficiencia se consideraron pertinentes, completas en cuanto a su especificación, adecuadas en el contexto de evaluación y precisas en el resultado alcanzado. Por lo tanto, el modelo se considera **Aceptable** de acuerdo a lo establecido en el Proceso de Estudio de Caso, específicamente la Actividad 2 “Identificar el conjunto de características que permiten la evaluación”.

6.2.2 Evaluación de las características del modelo

Seguidamente se presenta la segunda evaluación relacionada con las características específicas que corresponde a cada una de las categorías mostradas en la sección anterior, como lo son:

- Funcionalidad (FUN): FUN 1. Ajuste a los propósitos, FUN 2. Precisión, FUN 3. Interoperabilidad y FUN 4. Seguridad.
- Usabilidad (USA): USA 1. Facilidad de Comprensión, USA 2. Capacidad de Aprendizaje, USA 3. Interfaz Gráfica y USA 4. Operabilidad.
- Fiabilidad (FIA): FIA 2. Tolerancia a Fallas y FIA 3. Recuperación.
- Eficiencia (EFI): EFI 1. Comportamiento del tiempo y EFI 2. Utilización de recursos.

Las figuras 27, 28 y 29 muestran los resultados obtenidos en cada una las características correspondientes. Estos resultados se presentan en base a los datos mostrados en el ANEXO VII.

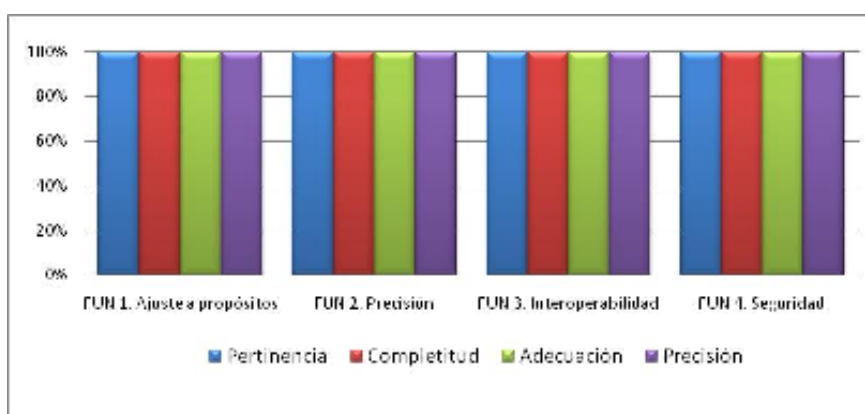


Figura 28 - Resultados de la evaluación de "Características" de la categoría Funcionalidad del modelo propuesto. Fuente: Elaboración propia

Como puede observarse en la Figura 27, todas las características de la categoría funcionalidad, alcanzaron el máximo porcentaje en la evaluación (100%). Esto demuestra que las características relacionadas a la categoría Funcionalidad, se consideraron: pertinente dentro del proceso de especificación de calidad, completa en cuanto a su especificación, adecuadas en el contexto de evaluación y precisas en el resultado alcanzado.

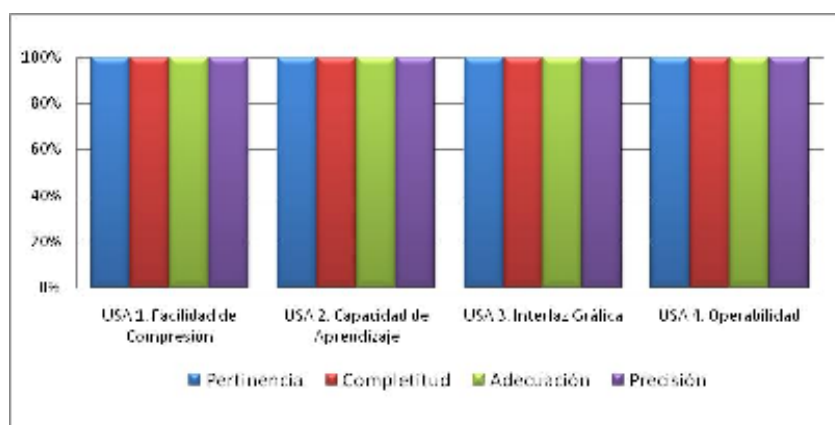


Figura 29 - Resultados de la evaluación de “Características” de la categoría Usabilidad del modelo propuesto. Fuente: Elaboración propia.

De acuerdo a la Figura 28, al igual que la categoría Funcionalidad, las características de la categoría Usabilidad alcanzaron el máximo porcentaje en la evaluación (100%) que declaran ser Aceptable para dicha categoría, por lo que efectivamente se consideraron: pertinente dentro del proceso de especificación de calidad, completa en cuanto a su especificación, adecuadas en el contexto de evaluación y precisas en el resultado alcanzado.

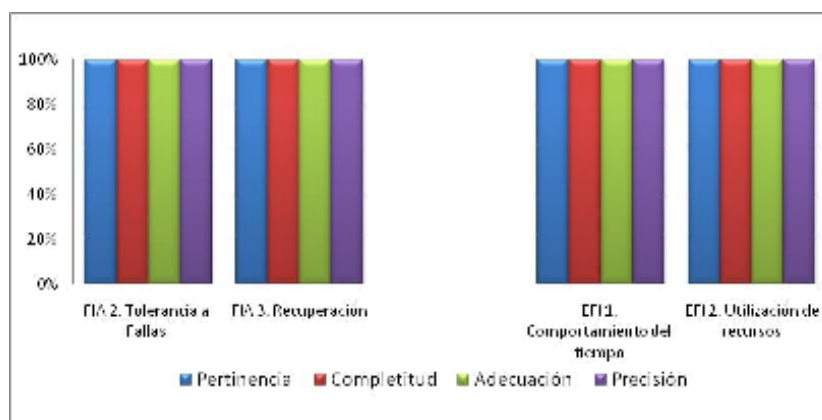


Figura 30 - Resultados de la evaluación de “Características” de la categoría Fiabilidad y Eficiencia del modelo propuesto. Fuente: Elaboración propia

En la Figura 29, se muestran las características propias correspondientes a las categorías Fiabilidad y Eficiencia respectivamente. Para la categoría Fiabilidad, las dos características evaluadas obtuvieron un 100% de aceptación. Para la categoría Eficiencia, ambas características alcanzaron el máximo porcentaje en la evaluación. Por lo anterior, se deduce que ambas categorías están en el nivel Aceptable de acuerdo al promedio de todas las respuestas para dicha categoría, por lo que dichas características se consideraron: pertinente dentro del proceso de especificación de

calidad, completa en cuanto a su especificación, adecuadas en el contexto de evaluación y precisas en el resultado alcanzado.

De acuerdo a lo analizado en base a las figuras 27 a 29, se concluye que todas las características evaluadas para el modelo son **Aceptables** en cuanto a pertinencia, completitud, adecuación y precisión, puesto que están en el máximo rango establecido para ser Aceptables de acuerdo a lo establecido en la Actividad 2 del Proceso de Estudio de Caso. Este mismo comportamiento en cuanto a aceptación se refiere, se deriva hacia las sub-características y sub-subcaracterísticas propuestas en el modelo (en la categoría Funcionalidad). El detalle de los resultados pueden observarse en el ANEXO VII. Esto conlleva a decir que el modelo propuesto en sus características generales es considerado Pertinente, Completo, Adecuado y Preciso.

Concluida la evaluación de las características generales hasta las más específicas del modelo, se procede a evaluar las métricas del modelo propuestas.

6.3 Evaluación de las Métricas del modelo

Es esta sección se presentan los resultados obtenidos luego de la evaluación de las métricas del modelo con su respectivo análisis. En la Figura 30, se presenta un resumen de los promedios obtenidos en la evaluación de las métricas del modelo, agrupadas por característica a evaluar dentro de cada Categoría del modelo. Los insumos para éste resultado pueden observarse en el Anexo VIII.

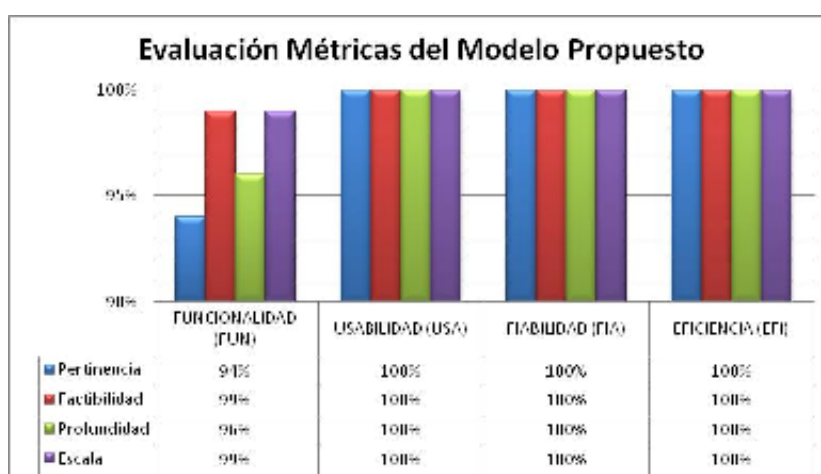


Figura 31 - Resultados de la evaluación de “Métricas” por Categoría del modelo propuesto.
Fuente: Elaboración propia.

Como puede observarse en la figura 30, el conjunto de métricas para cada Categoría en promedio superó el porcentaje esperado para dichas métricas, es decir, entre 75% y 100%, por lo que de acuerdo a los criterios de aceptación, se considera que el modelo propuesto en cuanto a métricas se refiere es **Aceptable** de acuerdo a los criterios establecidos en la Actividad 2 del Proceso de Estudio de Caso. Éste resultado se logra detallando cada característica específica de la siguiente manera:

- En cuanto a la *Pertinencia*, todas las métricas se consideran pertinentes en cada una de las categorías del modelo, por estar encima del mínimo esperado.
- Todas las métricas son consideradas *Factibles* en casi un 100% para cada una de las categorías del modelo.
- En el caso de la *Profundidad*, para las categorías Usabilidad, Fiabilidad y Eficiencia sus métricas correspondientes se consideran profundas, mientras que en la categoría Funcionalidad, las métricas propuestas están muy cercanas al mínimo establecido, pero siguen siendo Aceptables.
- Para los evaluadores, la *Escala* es apropiada en un 100% para todas las Categorías del modelo.

Con el fin de que todas las métricas lleguen al 100%, se proponen las siguientes modificaciones en las métricas del modelo de acuerdo a las clasificaciones obtenidas en el ANEXO VIII:

- “Reutilización de Casos de Prueba entre proyectos”, “Reutilización de Datos de Prueba entre proyectos” y “Reutilización de Scripts de Prueba entre proyectos” se consideraron no pertinentes pues no permiten definir la existencia o no de la característica que representan, las cuales son Casos de prueba FUN 1.6.1., Datos de Prueba (FUN 1.6.2) y Scripts de Prueba (FUN 1.6.3) respectivamente. Se propone trasladarlas a la característica Operaciones con terceros FUN 3.1.2.
- “Definición de estrategias de pruebas” de la característica Condiciones a probar en la Aplicación objeto de prueba (FUN 1.7.2) se consideró poco profunda pues no es muy específica en lo que busca lograr. Se propone redactarla como: Parametrización de estrategias de pruebas para condicionar la prueba.

- “Notifica superación de umbrales al momento” de la característica Estadísticas de resultados (FUN 1.8.1) muestra ser no profunda porque no especifica que el mecanismo de notificación sea visible; por ejemplo, puede notificar en un log, que no permite darse cuenta en tiempo real. Se propone redactarla como: Provee mecanismos visibles de notificación de umbrales al momento.
- “Formato de generación de Reportes/entregables son similares entre sí” se consideró no pertinente para evaluar la característica Manipulación de resultados (FUN 1.8.3), por lo que se propone trasladarla a la característica Interfaz Gráfica (USA 3).
- “Sincronización de Scripts de Prueba por nuevas versiones en la Aplicación objeto de prueba” de la característica Interacción con la Aplicación objeto de prueba (FUN 1.9.1) se consideró no factible porque es muy difícil medir este resultado pues puede que se actualice pero los scripts generados carezcan de calidad; se propone modificarlo como: Permite actualizar los scripts de pruebas generados ante actualización de la aplicación objeto de pruebas.
- “Control de acceso” de la característica control de acceso (FUN 4.1) se consideró no profunda, pues carece de especificidad. Se propone modificarla así: Control de acceso inicial a la herramienta.

CAPÍTULO VII: CONCLUSIONES Y RECOMENDACIONES

En esta sección, se presentan las ideas principales de cada uno de los capítulos que conforman la presente investigación, para plasmar las conclusiones producto de la experiencia obtenida a lo largo del desarrollo de este trabajo de maestría, por una parte, y por la otra, se exponen las recomendaciones a fin de continuar con ésta línea de investigación en futuros trabajos o mejoras del modelo propuesto.

7.1 Conclusiones

El tema de las *pruebas del software* con todas sus etapas es una realidad que debe estar presente en cualquier empresa que desee adentrarse en el mundo del desarrollo del software y, por supuesto, la calidad en los productos que éstas generan. Esta calidad esperada es en gran parte lograda a través de la formalización de la Disciplina de Pruebas en el proceso de desarrollo de software. Dedicar esfuerzos en este tema, son bien justificados para el desarrollo de la industria del software en nuestro país, elaborando nuevos y efectivos métodos adaptados a las características de las empresas desarrolladoras de software del país, específicamente las PYMES.

Un extenso recorrido y análisis de bibliografía relevante tomando en cuenta antecedentes de investigaciones propuestas por varios investigadores concernientes al tema de estudio y una completa definición del marco organizacional enmarcado en bases éticas y legales son presentados en el capítulo II. Allí se expresan conocimientos que van desde la definición de la calidad y calidad del software hasta la importancia que tienen las pruebas del software y el uso de modelos de calidad.

La metodología Investigación-Acción fue un factor relevante para el éxito de esta investigación, pues provee herramientas que unificadas con el método de evaluación DESMET forman el completo Framework Metodológico del Laboratorio de Investigación en Sistemas de Información LISI que permite secuenciar en forma ordenada y metódica cada uno de los pasos y/o procesos necesarios para cumplir con los objetivos propuestos a la vez de llegar rápida y acertadamente a la fase de “Tomar la acción” insumo para la evaluación de dicha acción y la especificación final del aprendizaje obtenido, con la retroalimentación necesaria.

El desarrollo de la propuesta en sí (capítulo IV) estuvo coordinado con los objetivos específicos realizando cuatro grandes aportes al conocimiento en el área de los sistemas de información:

- a) Modelo conceptual que abarca las principales definiciones en cuanto a pruebas de software se refiere, producto de un extenso análisis de la bibliografía y las experiencias con herramientas de pruebas existentes en el mercado.
- b) Árbol de clasificación de herramientas de pruebas automatizadas con su correspondiente instanciación de acuerdo a las necesidades de investigación, específicamente herramientas para el nivel de pruebas del sistema del tipo no funcionales: Desempeño y carga, bajo un enfoque de Caja Negra.
- c) Modelo de referencia propuesto para seleccionar herramientas de pruebas que apoyan empresas representadas por las PYMEs desarrolladoras de software, producto de aportes al modelo de calidad sistémica MOSCA para el producto.
- d) Guía para aplicar el modelo propuesto a herramientas de prueba existentes en el mercado tecnológico, previa selección de acuerdo al árbol de clasificación.

El modelo propuesto fue evaluado de acuerdo al proceso de evaluación establecido por DESMET que va desde el análisis de contexto hasta su aplicación de acuerdo al método arrojado “Análisis de características por estudio de caso”. Primeramente el modelo se aplicó en cuatro estudios de casos representados por herramientas de pruebas meticulosamente elegidas del mercado tecnológico de acuerdo a ciertos criterios. Una vez adquirida experiencia en el modelo y garantizada su efectividad, se procedió a evaluarlo en sí considerando cada uno de los niveles propuestos (categorías, características, sub-características, sub-sub-características y métricas) como pertinentes, completos, adecuados y precisos en el entorno de evaluación. El modelo fue producto de exhaustivas revisiones y mesas de trabajo con el patrocinador (LISI) cuyos integrantes con su gran experiencia, retroalimentaron y evaluaron el modelo hasta refinar el modelo propuesto.

Dicho modelo incluye aportes realizados por el investigador en cuanto a la incorporación de nuevas sub-características, sub-sub-características y métricas. Igualmente, se propusieron modificaciones a algunas métricas del modelo propuesto, puesto que en la evaluación de las mismas, no se consideraron pertinentes y completas en un 100%. Cabe destacar que también se propuso una adaptación del algoritmo de

aplicación de MOSCA en la determinación del nivel de calidad del producto dado que éste modelo se basa en tres categorías y no se correspondía con las cuatro categorías que contiene el modelo propuesto.

De los cuatro estudios de casos, representados por las herramientas HP LoadRunner (1), SilkPerformer (2), IBM/Rational Performance Tester (3), SoapUI Pro (4), se excluyó el estudio de caso 2 por no cumplir con la categoría Funcionalidad de acuerdo al algoritmo de MOSCA. De los tres estudios de caso restantes, se obtuvo que el que tiene un Nivel de calidad *Avanzado* es el 4 “SoapUI Pro” del fabricante Eviware con tecnología Open source, por lo que las PYMES desarrolladoras de software en Venezuela pueden hacer uso de esta herramienta de prueba sin acarrear mayores costos imputados en los productos por ellas desarrollados. Los estudios de caso 1 y 2, HP Load Runner y Silk Performer respectivamente, arrojaron un nivel de calidad del producto *Intermedio*.

7.2 Recomendaciones

De acuerdo a los resultados obtenidos, se consideran las siguientes recomendaciones para futuros trabajos de investigación:

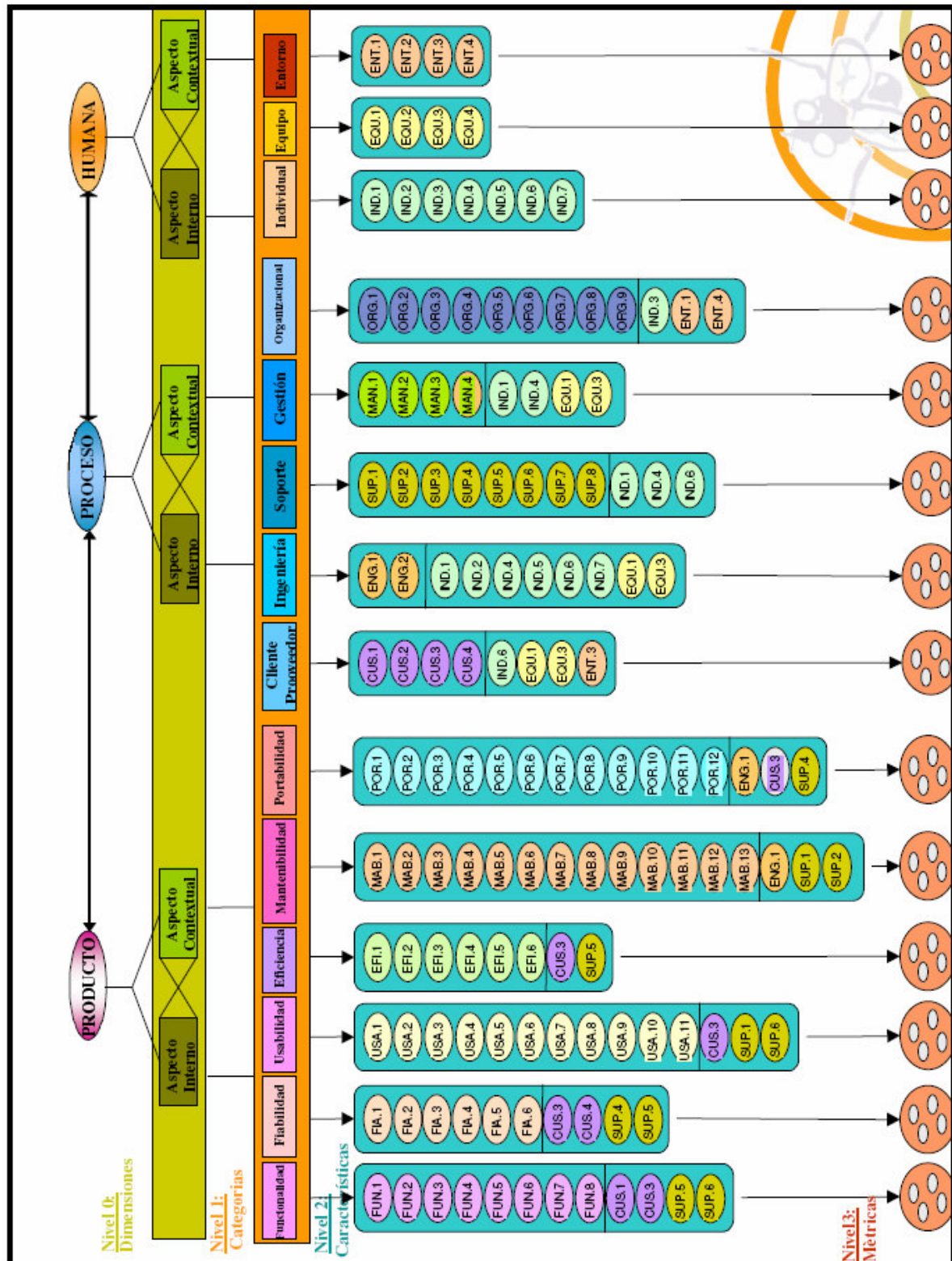
Ampliar el entorno de aplicación del modelo propuesto para abarcar la mayor cantidad de herramientas de pruebas que existen en el mercado con su adecuada clasificación, y que a su vez, suplen todos los tipos de pruebas, recalcando las características propias y distintivas de cada tipo de prueba que no necesariamente aplican a otro tipo de pruebas. Por esta razón, es recomendable que la aplicación del modelo sea realizada por personal experto en pruebas que tenga muy claro el objeto de evaluación y/o propósito de las pruebas enmarcadas en un proyecto de pruebas para hacer una acertada selección previa de herramientas que satisfagan los requerimientos explícitamente planteados por la organización para su implementación.

Profundizar el árbol de clasificación de herramientas de pruebas automatizadas en el mercado, propuesto por el investigador, a fin de poder estructurar y guiar la construcción de un modelo que cumpla con las indicaciones del punto anterior, haciendo énfasis en las ramas excluyentes (procurando mostrar todas las opciones

posibles) y en las ramas incluyentes (previendo las múltiples combinaciones que pueden surgir). Para esto, se recomienda proponer un Sistema de soporte de decisiones para obtener los mejores resultados al momento de obtener una o más instancias del árbol de clasificación de herramientas propuesto.

Promover el modelo de selección de herramientas de pruebas propuesto a la mayor cantidad de PYMES venezolanas desarrolladoras de software a fin de garantizar la mayor calidad en los productos que éstas desarrollan, enfatizando en la facilidad de la aplicación del modelo y sus innumerables beneficios, sensibilizando a este sector sobre la importancia de dar valor a la disciplina de pruebas como ventaja competitiva entre sus homólogos.

ANEXO I: Modelo de Calidad MOSCA



ANEXO II: Aspectos de la categoría FUNCIONALIDAD tomadas de MOSCA

Característica	Métrica	Pregunta	Formulación	Dirigidos a:
Funcionalidad(FUN)				
FUN.1 Ajuste a los propósitos	Cumplimiento de las necesidades funcionales	¿El producto cumple con todas las necesidades funcionales?	5=TODAS 4=CASI TODAS 3=MUCHAS 2=MUY POCAS 1= NINGUNA	USUARIO
	Definición de las necesidades de adquisición, desarrollo o mejora	¿Se tienen establecidas las necesidades de adquisición, desarrollo o mejora del sistema (software, hardware y personal)?	5=Completam. def. 4=Casi todo def. 3= Med. Def. 2= Poco def. 1= No están def.	LIDER DESARROLLADO R
	Definición de la estrategia de adquisición o desarrollo de nuevos sistemas	¿Existe una estrategia desarrollada para la adquisición?	5=Completam. def. 4=Casi todo def. 3= Med. Def. 2= Poco def. 1= No están def.	LIDER DESARROLLADO R
	Definición del criterio de aceptación	¿Existe una definición del criterio de aceptación?	5=Completam. def. 4=Casi todo def. 3= Med. Def. 2= Poco def. 1= No están def.	LIDER DESARROLLADO R
	Registro de los resultados de la prueba	¿Los resultados de la prueba fueron registrados?	5=TODAS 4=CASI TODAS 3=MUCHAS 2=MUY POCAS 1= NINGUNA	DESARROLLADO R
FUN.2 Precisión	Resultados incompletos	¿Existe falta de precisión en la función debido a resultados incompletos?	5=TODAS 4=CASI TODAS 3=MUCHAS 2=MUY POCAS 1= NINGUNA	USUARIO
	Resultados incorrectos	¿Existe falta de precisión en la función debido a resultados incorrectos?	5=TODAS 4=CASI TODAS 3=MUCHAS 2=MUY POCAS 1= NINGUNA	USUARIO
	Resultados emitidos no esperados	¿Existe falta de precisión en la función debido a la emisión de resultados no esperados?	5=TODAS 4=CASI TODAS 3=MUCHAS 2=MUY POCAS 1= NINGUNA	USUARIO
	Criterios de validación	¿Existen criterios de validación para el producto?	5=Completam. def. 4=Casi todo def. 3= Med. Def. 2= Poco def. 1= No están def.	DESARROLLADO R
FUN. 3 Interoperabilidad	Existencia de funcionalidades pertenecientes al producto que son utilizadas por otros sistemas	¿Existen funcionalidades utilizadas por otros sistemas, que pertenecen al producto?	5= no tiene 1= tiene	DESARROLLADO R USUARIO
	Intercambio de datos	¿Existe intercambio de datos con otros sistemas?	5= no tiene 1= tiene	DESARROLLADO R / USUARIO
	Consistencia de interfaces	¿Existe consistencia con las interfaces de los otros sistemas?	5= es consistente 1= no es consistente	DESARROLLADO R
	Existencia de funcionalidades de otros sistemas que sean necesarias	¿Existen funcionalidades en otros sistemas que son necesarias?	5= no tiene 1= tiene	USUARIO / DESARROLLADO R
FUN.4 Seguridad	Control de acceso	¿Qué tan completa es la detección de acceso a usuarios del sistema, cuando se trata de obtener acceso al sistema de forma no autorizada?	5=Muy alta 4=Alta 3= Mediana 2= Básica 1= No tiene.	DESARROLLADO R / USUARIO
	Capacidad de auditoría	¿Qué tan completa es la auditoría? ¿Se evalúa la cantidad de registros de acceso que el sistema registró en la base de datos histórica?	5=Muy alta 4=Alta 3= Mediana 2= Básica 1= No tiene.	DESARROLLADO R

ANEXO III: Aspectos de la categoría USABILIDAD tomadas de MOSCA

Característica	Métrica	Pregunta	Formulación	Dirigidos a:
Usabilidad (USA)				
USA.1 Facilidad de comprensión	Compleitud de las demostraciones funcionales	¿Están completas las demostraciones funcionales implementadas respecto al número de funciones que requieren la demostración de uso?	5=Todas 4=Casi todas 3= Algunas 2= Pocas 1= Ninguna	DESARROLLADOR USUARIO
	¿Cuál es el tiempo en que un usuario adquiere las destrezas necesarias para usar el sistema?	¿Cuál es el tiempo en que un usuario adquiere las destrezas necesarias para usar el sistema?	5= 6 días o menos 4= 7-14 días 3= 15-30 días 2= 2-5 meses 1= 6 meses o más	USUARIO
	Nivel de dificultad de la aplicación	Nivel de dificultad de la aplicación	5=Muy difícil 4=Difícil 3=Promedio 2=Fácil 1=Muy fácil	USUARIO
	Facilidad para ubicar funcionalidades	¿Es fácil ubicar las funcionalidades del sistema?	5= Muy fácilmente 4= Fácilmente 3= Dificultad media 2= Difícil 1= Muy difícil	USUARIO
USA.2 Capacidad de aprendizaje	Calidad en la clasificación de los temas	¿Es fácil utilizar el material de apoyo, por una buena clasificación de los temas	5=Excelente 4=Buena 3= Promedio 2=Debajo del Prom. 1= Inaceptable.	USUARIO
	Compleitud de la especificación de los documentos	¿Están identificados todos los documentos a ser producidos por el proceso o proyecto?	5=Todos 4=Casi todos 3= Algunos 2= Pocos 1= Ninguno	LIDER DESARROLLADOR USUARIO
USA.3 Interfaz Gráfica	Interfaz Personalizable	¿La interfaz es personalizable?	5= Si 1= No	USUARIO
	Localización rápida de opciones	¿Se localizan rápidamente las opciones, es consistente la ubicación de utensilios?	5= Si 1= No	USUARIO
	Presencia de metáforas	¿Hay presencia de metáforas?	5= Si 1= No	USUARIO
	Satisfacción del Diseño visual	¿Es bueno el diseño visual de las pantallas?	5=Excelente 4=Buena 3=Promedio 2=Debajo del Prom. 1= Inaceptable	USUARIO
	Versatilidad de la navegación	¿Es versátil la navegación, entre pantallas?	5=Excelente 4=Buena 3=Promedio 2=Debajo del Prom. 1= Inaceptable	USUARIO
USA.4 Operabilidad	Satisfacción con las ayudas disponibles respecto a las requeridas	¿Cómo es la satisfacción con las ayudas disponibles respecto a las requeridas?	5=Excelente 4=Buena 3=Promedio 2=Debajo del Prom. 1= Inaceptable	USUARIO
	Tasa de disponibilidad de valores por defecto	¿Existen valores por defecto en las pantallas?	Nº de valores x defecto / Nº entradas la pantalla	DESARROLLADOR
	Evaluación continua de la Satisfacción del cliente	¿Existe una evaluación continua de la satisfacción del cliente respecto a los servicios de apoyo y al producto?	5=Siempre 4=Casi siempre 3=Algunas veces 2=Pocas veces 1=Nunca	LIDER DESARROLLADOR USUARIO
	Soporte operacional continuo	¿Se provee apoyo para resolver problemas operacionales según requerimientos del cliente?	5=Siempre 4=Casi siempre 3=Algunas veces 2=Pocas veces 1=Nunca	LIDER DESARROLLADOR USUARIO
	Operación apropiada del software en su ambiente	¿Existe una correcta operación del software en su ambiente?	5= Si 1= No	DESARROLLADOR USUARIO
	Asistencia a los clientes en cuanto a operación	¿Existe asistencia a los clientes en cuanto a su operación?	5= Si 1= No	USUARIO

ANEXO IV: Aspectos de la categoría FIABILIDAD y EFICIENCIA tomadas de MOSCA

Fiabilidad (FIA)				
FIA.2 Tolerancia a Fallas	Anulación de operación incorrecta	¿Qué tan frecuente puede el producto de software <i>evitar</i> fallas, aún si el usuario lo opera incorrectamente?	5=Siempre 4=Casi siempre 3= Algunas veces 2= Pocas veces 1= Nunca	DESARROLLAD OR
FIA.3 Recuperación	Capacidad de reiniciar	¿Puede el producto de software recuperarse fácilmente después de una caída?	5=Muy fácilmente 4=Fácilmente 3=Dificultad Media 2= Difícil 1= Muy difícil	DESARROLLAD OR USUARIO
	Velocidad de reinicialización	Velocidad de la reinicialización.	5=Muy rápida 4=Rápida 3= Promedio 2=Lenta 1= Muy lenta	DESARROLLAD OR
	Disponibilidad del producto de software	¿Pueden los usuarios trabajar con el producto de software el tiempo necesario?	5=Siempre 4=Casi siempre 3= Algunas veces 2= Pocas veces 1= Nunca	DESARROLLAD OR USUARIO

Eficiencia (EFI)				
EFI.1 Comportamiento del tiempo	Tiempo de respuesta	Tasa del tiempo de respuesta para lo que es diseñado a ser ejecutado	Tiempo de respuesta real/tiempo de respuesta requerido	DESARROLLAD OR
	Satisfacción con el número de tareas completadas en un tiempo especificado	Satisfacción con el número de tareas que pueden ser completadas durante un tiempo especificado con respecto al número de tareas que son requeridas a ser completadas	5=Muy Alta 4=Alta 3= Media 2= Baja 1= Muy baja	DESARROLLAD OR
EFI.2 Utilización de recursos	Utilización del CPU	Tasa de tiempo de CPU utilizado	Tiempo de CPU utilizado / Tiempo de CPU requerido	DESARROLLAD OR
	Utilización de memoria	Tasa del tamaño de memoria utilizada con respecto al tamaño de la memoria que se requiere que sea usada	Memoria utilizada / Requerimientos del uso de memoria	DESARROLLAD OR

ANEXO V: Descripción de algunas métricas aportadas por el autor a MOSCA en el contexto del modelo propuesto

Característica/métrica	Descripción
Generación automática de Casos de prueba	Generación automática de casos de prueba y/o escenarios que se van a probar de acuerdo a datos de entrada específicos.
Generación de evidencias de pruebas	Registra evidencias de pantallas, resultados, eventos en forma automática de acuerdo a parámetros
Generación automática de datos de pruebas	Permite generar/construir el set de datos automático, réplicas de los existentes. Poblamiento de datos automáticos
Gestión de defectos	Registra los defectos automáticamente, con cierto detalle (momento de ocurrencia, posible origen, trazabilidad, entre otros)
Reportes/entregables mantienen el mismo formato	Consistencia en el formato de los reportes que genera la herramienta de prueba
Programación de Scripts de pruebas en el tiempo (scheduler)	Una vez ordenada la prueba, permite iniciar la ejecución automáticamente de acuerdo al tiempo previamente establecido
Variación de carga de trabajo durante la prueba	Factores que cambiará durante la prueba de rendimiento. Ejm., variar el número de usuarios para comprender cómo se degrada el tiempo de respuesta a medida que la carga de trabajo aumenta
Log de Auditoría	Permite guardar un log de las pruebas ejecutadas y los pasos de la misma
Criterios de prueba superada y no superada	Permite determinar/establecer cuándo una prueba se considera superada y cuándo no a través de porcentajes, umbrales, relaciones entre variables, etc.
Estadísticas de resultados permanente	Acceso permanente a los resultados obtenidos. Se mantiene un BD de los resultados en cualquier momento.
Estadísticas en tiempo real	Genera estadísticas al momento sin necesidad de detener la prueba
Notifica superación de umbrales en la aplicación	Genera alertas, mensajes o distinción ante la cercanía de los umbrales permitidos y/o establecidos por la aplicación
Generación de métricas	Permite establecer. Configurar, parametrizar indicadores o métricas esperadas
Relaciona defectos a su causa	Determina (al menos a un alto nivel) la posible causa del defecto
Análisis de resultados automáticos	Analiza los resultados con conclusiones generales, permitiendo configurar los resultados a conveniencia del usuario
Sincronización de Scripts de Prueba por nuevas versiones en la Aplicación objeto de prueba	Ante una nueva versión de la aplicación objeto de pruebas, permite realizar cambios a lo configurado sin demasiados pasos adicionales. Se relaciona sin problemas con la aplicación, es decir, se monta sobre la aplicación sin problemas, sus funciones, capacidades, etc.
Emula población/acceso de usuarios concurrentes	Simula el acceso y uso de la aplicación objeto de prueba desde varias entradas (usuarios concurrentes) con cargas de trabajo múltiples
Modificar scripts de pruebas en línea/ejecución	Permite hacer cambios mientras se hace la ejecución de la prueba a fin de obtener mejores resultados
Modificar scripts de pruebas luego de generados	Scripts dinámicos que pueden ser modificados luego de su creación
Reutilización de Scripts entre proyectos	Permite usar scripts definidos para un proyecto, en otro proyecto con sus respectivos cambios (importar scripts)
Importar scripts de otras aplicaciones/lenguajes	Permite importar scripts desde otras aplicaciones/herramientas de pruebas aún con lenguajes de programación distintos
Ante errores/defectos permite continuar	No se detiene la ejecución de pruebas por ciertos errores, sino que lo reporta y continua con el proceso
Provee Wizard para ciertas funcionalidades	Ayudas en la instalación/utilización/configuración de ciertas bondades de la herramienta de prueba
Provee ayudas en línea	Ofrece ayuda para las diversas funcionalidades de la herramienta
Barra de herramientas (entorno) customizables	Tiene barra de herramientas y se puede personalizar por perfiles de usuarios (colores, entorno, funciones/módulos activos, etc.)
Código único/proceso unificado para multiplataformas	Proporciona un lenguaje de programación único para la construcción de casos de prueba y script de pruebas (ejemplo XML)

Código Propietario (propone comandos, funciones propietarias)	La herramienta trabaja con código de programación propietario, único para la herramienta, con sentencias propias inherentes a la herramienta. Propone su propio pseudo-lenguaje de programación
Necesita amplios conocimientos en programación	Requiere conocimientos de programación a un nivel específico para poder construir los script/casos de prueba
Facilidad en el manejo de la aplicación	Es manejable la aplicación, su recorrido es sencillo
Ofrece ejemplos, valores pre-cargados en las pantallas	Proporciona guías para llenar campos y/o generar algunas funcionalidad
Componentes/plugins que complementa la herramienta	Tiene algún módulo, componente, sub-aplicación, extensiones, plugins que la conforma y/o aumenta sus potencialidades
Capacidad de guardar pruebas para continuar después	Permite detener una prueba, guardar su ejecución y continuar más adelante
Control de acceso a los usuarios	Mantiene registro/auditoría del acceso de los usuarios que usan el sistema
Definición de roles dentro del sistema	Permite configurar e implementar roles por usuario donde cada uno tiene su propia configuración o permisología para ciertas funcionalidades

ANEXO VI: Evaluación detallada de las Herramientas de prueba seleccionadas de acuerdo al Modelo propuesto

Estudios de Caso:

HP LoadRunner (1), SilkPerformer (2), IBM/Rational Performance Tester (3), SoapUI Pro (4)

Sub-característica	Sub Sub-característica	Métrica	Pregunta	Formulación	Dirigido a	Estudio de Caso			
						1	2	3	4
FUNCIONALIDAD (FUN) - Característica: Ajuste a los propósitos (FUN 1)									
		Cumplimiento de las necesidades funcionales	¿El producto cumple con todas las necesidades funcionales?	5= Todas 4= Casi todas 3= Muchas 2= Muy pocas 1= Ninguna	Usuario	5	5	5	5
		Definición de las necesidades de adquisición, desarrollo o mejora	¿Se tienen establecidas las necesidades de adquisición, desarrollo o mejora del sistema (software, hardware y personal)?	5=Completa def. 4=Casi todo def. 3= Med. Def. 2= Poco def. 1= No están def.	lider Usuario	5	5	5	5
		Definición de la estrategia de adquisición o desarrollo de nuevos sistemas	¿Existe una estrategia desarrollada para la adquisición?	5=Completa. def 4=Casi todo def. 3= Med. Def. 2= Poco def. 1= No están def.	lider Usuario	5	5	5	5
		Definición del criterio de aceptación	¿Existe una definición del criterio de aceptación?	5=Completa def. 4=Casi todo def. 3= Med. Def. 2= Poco def. 1= No están def.	lider Usuario	5	5	5	5
		Registro de los resultados de la prueba	¿Los resultados de la prueba fueron registrados?	5=Todas 4=Casi Todas 3=Muchas 2=Muy Pocas 1= Ninguna	Usuario	5	5	5	5
Niveles de Prueba (FUN 1.1)	Pruebas de sistema (FUN 1.1.1)	Ejecución de pruebas de a nivel de sistema	¿Permite realizar pruebas al nivel de pruebas del sistema?	5 = Si 1 = No	Usuario	5	5	5	5
Tipos de Prueba (FUN 1.2)	Pruebas de Desempeño (FUN1.2.1)	Ejecución del Tipo de Pruebas de Desempeño	¿Permite realizar pruebas del tipo Desempeño a la Aplicación objeto de prueba?	5 = Si 1 = No	Usuario	5	5	5	5
	Pruebas de Carga/Estrés (FUN1.2.2)	Ejecución del Tipo de Pruebas de Carga/Estrés	¿Permite realizar pruebas del tipo Carga/Estrés a la Aplicación objeto de prueba?	5 = Si 1 = No	Usuario	5	5	5	5
Enfoque de Prueba (FUN 1.3)	Pruebas de Caja Negra (FUN 1.3.1)	Ejecución de Pruebas con enfoque de Caja Negra	¿Las pruebas son de enfoque de Caja Negra?	5 = Si 1 = No	Usuario	5	5	5	5
		Exploración en el código de la Aplicación objeto de prueba	¿Omite el Profundizar/analizar el código de la Aplicación objeto de prueba?	5 = Si 1 = No	Usuario	5	5	5	5
Licencias de distribución de la Herramienta de Prueba (FUN 1.4)	Open source (FUN 1.4.1)	Distribución de la Herramienta de Prueba Opensource	¿La licencia de la Herramienta de Prueba es de tipo opensource (libre)?	5 = Si 1 = No	Usuario	1	1	1	5
	Comercial (FUN 1.4.2)	Distribución de la Herramienta de Prueba es comercial	¿La licencia de la Herramienta de Prueba es de tipo comercial (privado)?	5 = Si 1 = No	Usuario	5	5	5	1
Administración de la Herramienta de Prueba (FUN 1.5)	Lenguaje de Programación de la Herramienta de Prueba (FUN 1.5.1)	Código único/proceso unificado para multiplataformas	¿Proporciona un lenguaje de programación único para la construcción de entregables?	5 = Si 1 = No	Usuario	1	1	5	5
		Código Propietario (propone comandos, funciones propietarias)	¿Propone su propio pseudo-lenguaje de programación para construir entregables?	5 = Si 1 = No	Usuario	5	5	1	1
		Necesidad de amplios conocimientos en programación	¿Requiere conocimientos de programación a un nivel específico para poder construir los entregables?	5=Básica 3= Mediana 1= Alta	Usuario	3	5	5	3
Entregables	Casos de Prueba	Generación automática	¿Genera automáticamente los	5 = Si	Usuario	5	5	1	5

de Pruebas (FUN 1.6)	(FUN 1.6.1)	de Casos de Prueba	Casos de Prueba de acuerdo a ciertas entradas?	1 = No					
			¿Cómo es la completitud de los Casos de Prueba generados automáticamente en cuanto a: pasos, pre-condiciones, post-condiciones, valores, resultados esperados?	5=Completam 4=Casi todos 3= Med. 2= Pocos 1= No aplica	Usuario	3	4	1	4
		Construcción de Casos de Prueba paso a paso	¿Permite construir Casos de Prueba paso a paso?	5 = Si 1 = No	Usuario	1	5	5	5
		Identificación de los requerimientos de prueba	¿Permite identificar los requerimientos de prueba que se requieren probar?	5 = Si 1 = No	Usuario	5	5	5	5
		Reutilización de Casos de Prueba entre proyectos	¿Los Casos de Prueba sirven para alimentar proyectos afines?	5 = Si 1 = No	Usuario	5	5	1	1
		Documentación de Pre-condiciones	¿Permite documentar las pre-condiciones en los Casos de Prueba?	5 = Si 1 = No	Usuario	5	5	1	1
		Documentación de Post-condiciones	¿Permite documentar las post-condiciones en los Casos de Prueba?	5 = Si 1 = No	Usuario	5	5	1	1
	Datos de Prueba (FUN 1.6.2)	Generación manual de Datos de Prueba desde Casos de Prueba	¿Se establecen manualmente los Datos de Prueba en los Casos de Prueba?	5 = Si 1 = No	Usuario	5	5	5	5
		Generación automática de Datos de Prueba desde datos reales	¿Genera automáticamente los Datos de Prueba desde los datos reales identificados?	5 = Si 1 = No	Usuario	5	5	5	5
		Generación automática de Datos de Prueba desde sentencias SQL	¿Se establecen automáticamente los Datos de Prueba desde sentencias SQL, base de datos?	5 = Si 1 = No	Usuario	5	5	1	5
		Generación automática de Datos de Prueba de forma aleatoria	¿Genera automáticamente los Datos de Prueba aleatoriamente de acuerdo a datos iniciales?	5 = Si 1 = No	Usuario	5	1	5	5
		Reutilización de Datos de Prueba entre proyectos	¿Los Datos de Prueba sirven para alimentar proyectos afines?	5 = Si 1 = No	Usuario	5	5	1	5
	Script de Pruebas (FUN 1.6.3)	Generación automática de Scripts de Prueba desde Casos de Prueba + Datos de Prueba	¿Genera automáticamente los Scripts de Prueba de en base a los Casos de Prueba + Datos de Prueba?	5 = Si 1 = No	Usuario	1	5	1	1
		Construcción de Scripts de Prueba paso a paso	¿Permite construir Scripts de Prueba paso a paso, manualmente?	5 = Si 1 = No	Usuario	5	5	5	5
		Modificar Scripts de Prueba luego de generarlos	¿Permite modificar Scripts de Prueba luego de generarlos?	5 = Si 1 = No	Usuario	5	1	5	5
		Modificar Scripts de Prueba en línea/ejecución	¿Permite modificar Scripts de Prueba al momento de ejecutarlo?	5 = Si 1 = No	Usuario	1	5	5	5
		Reutilización de Scripts de Prueba entre proyectos	¿Permite usar/importar Scripts de Prueba existentes entre proyectos?	5 = Si 1 = No	Usuario	5	5	1	5
	Suite de Pruebas (FUN 1.6.4)	Organizar/secuenciar conjunto de pruebas	¿Permite organizar y/o secuenciar conjunto de Scripts de Prueba?	5 = Si 1 = No	Usuario	5	5	5	5
		Programación de Scripts de Prueba en el tiempo	¿Permite programar pruebas en el tiempo (scheduler)?	5 = Si 1 = No	Usuario	5	5	5	5
	Salidas de pruebas (FUN 1.6.5)	Generación de evidencias de Pruebas	¿Genera evidencias de resultados de pruebas automáticos?	5 = Si 1 = No	Usuario	5	5	5	5
		Gestión de defectos	¿Documenta defectos en forma automática luego de presentado el error?	5 = Si 1 = No	Usuario	5	5	5	5
			¿Permite agregar/crear defectos?	5 = Si / 1 = No	Usuario	5	1	1	5
			¿Permite modificar defectos?	5 = Si / 1 = No	Usuario	5	5	5	5
			¿Permite notificar defectos al responsable de la aplicación/módulo?	5 = Si 1 = No	Usuario	5	1	1	5
		Log de pruebas/histórico de ejecuciones	¿Permite guardar un log de las pruebas ejecutadas?	5 = Si 1 = No	Usuario	5	5	5	5
	Relación entre entregables de Pruebas (FUN 1.6.6)	Relación entre requerimientos de prueba y Casos de Prueba	¿Posee mecanismo para relacionar requerimientos de prueba a los Cados de Prueba?	5 = Si 1 = No	Usuario	5	5	1	5
		Relación entre Casos de	¿Posee mecanismo para	5 = Si	Usuario	5	5	5	5

		Prueba y Script de Prueba	relacionar Casos de Prueba a los Scripts de Pruebas?	1 = No					
		Relación de defectos a Casos de Prueba	¿Posee mecanismo para relacionar defectos a los Casos/Script de Pruebas?	5 = Si 1 = No	Usuario	5	1	1	5
		Relaciona defectos a su causa	¿Permite determinar la posible causa del defecto?	5 = Si 1 = No	Usuario	5	5	5	5
Planificación de Pruebas (FUN 1.7)	Condiciones a probar en la Aplicación de objeto de prueba (FUN 1.7.1)	Emulación de usuarios concurrentes	¿Emula población/acceso de usuarios concurrentes?	Emulación de usuarios concurrentes	Usuario	5	5	5	5
		Emulación de transacciones concurrentes	¿Emula gran carga de transacciones concurrentes?	5 = Si 1 = No	Usuario	5	5	5	5
		Variación de carga de trabajo durante la prueba	¿Permite variar la carga de trabajo en el tiempo de la ejecución de la prueba?	5 = Si 1 = No	Usuario	5	1	5	5
		Definición de límites/umbrales de finalización de prueba	¿Permite definir límites/umbrales donde la prueba debe concluir?	5 = Si 1 = No	Usuario	5	5	5	5
			¿Permite definir tiempos donde la prueba debe concluir?	5 = Si 1 = No	Usuario	5	5	5	5
		Definición de estrategias de pruebas	¿Permite ingresar retardos en la ejecución de pruebas?	5 = Si 1 = No	Usuario	5	5	5	5
			¿Permite definir hilos de ejecución en la prueba?	5 = Si 1 = No	Usuario	5	5	5	5
			¿Permite definir ráfagas aleatorias en la ejecución de pruebas?	5 = Si 1 = No	Usuario	5	5	5	5
			¿Permite establecer un script para la estrategia de carga de trabajo?	5 = Si 1 = No	Usuario	5	1	1	5
		Cancelación de pruebas al superar umbrales	¿Permite cancelar la ejecución de las pruebas al superar los umbrales establecidos?	5 = Si 1 = No	Usuario	5	5	5	5
Gestión de resultados (FUN 1.8)	Estadísticas de resultados (FUN 1.8.1)	Estadísticas en tiempo real	¿Genera estadísticas/resultados en tiempo real?	5 = Si 1 = No	Usuario	5	5	5	5
		Estadísticas post-mortem	¿Genera estadísticas luego de la ejecución de las pruebas?	5 = Si 1 = No	Usuario	5	5	5	5
		Notifica superación de umbrales al momento	¿Genera alertas, mensajes ante la cercanía de los umbrales permitidos?	5 = Si 1 = No	Usuario	5	5	5	5
		Generación de métricas	¿Permite establecer indicadores o métricas esperadas?	5 = Si 1 = No	Usuario	5	5	5	5
			¿Permite configurar/parametrizar indicadores o métricas esperadas?	5 = Si 1 = No	Usuario	5	1	5	1
	Análisis de resultados (FUN 1.8.2)	Provee de reportes predeterminados para gestionar resultados	¿Tiene reportes preexistentes para emitir resultados de pruebas?	5 = Si 1 = No	Usuario	5	5	5	5
		Permite modificar los reportes predeterminados	¿Permite modificar reportes de resultados preexistentes?	5 = Si 1 = No	Usuario	5	1	5	1
		Permite construir reportes varios de resultados	¿Permite construir reportes de resultados?	5 = Si 1 = No	Usuario	5	1	5	5
		Provee gráficas predeterminadas para gestionar resultados	¿Tiene gráficas preexistentes para emitir resultados de pruebas?	5 = Si 1 = No	Usuario	5	5	5	5
		Permite modificar los gráficos predeterminados	¿Permite modificar gráficos de resultados preexistentes?	5 = Si 1 = No	Usuario	5	5	5	5
		Permite construir gráficos varios de resultados	¿Permite construir gráficos de resultados?	5 = Si 1 = No	Usuario	1	5	5	5
		Análisis de resultados automáticos	¿Proporciona conclusiones en base a los resultados obtenidos?	5 = Si 1 = No	Usuario	1	1	5	1
	Manipulación de resultados (FUN 1.8.3)	Exportar resultados en formatos intercambiables	¿Permite exportar los resultados obtenidos en formato Excel?	5 = Si 1 = No	Usuario	5	5	5	5
			¿Permite exportar los resultados obtenidos en otros formatos como pdf, doc, html, rtf, xml?	5=Completam 4=Casi todos 3= Med. 2= Pocos 1= No	Usuario	4	4	3	3
		Formato de generación de Reportes/entregables son similares entre si	¿Son similares los formatos cómo se generan los reportes/entregables de resultados obtenidos?	5=Completam 4=Casi todos 3= Med. 2= Pocos 1= Ninguno	Usuario	5	5	5	5

		Disponibilidad de los resultados obtenidos (base de datos)	¿Existe acceso permanente a los resultados obtenidos?	5 = Si 1 = No	Usuario	5	5	5	5
Aplicación objeto de prueba (FUN 1.9)	Interacción con la Aplicación objeto de prueba (FUN 1.9.1)	Sincronización de Scripts de Prueba por nuevas versiones en la Aplicación objeto de prueba	¿Permite sincronizar los Scripts de Prueba seleccionados ante nuevas versiones de la Aplicación objeto de prueba?	5 = Si 1 = No	Usuario	1	5	5	1
		Las pruebas se generan al grabar acciones posibles de los usuarios	¿Permite grabar acciones cotidianas del usuario en la Aplicación objeto de prueba?	5 = Si 1 = No	Usuario	1	5	5	5
		Las pruebas se generan especificando manualmente las acciones posible	¿Permite especificar acciones cotidianas del usuario en la Aplicación objeto de prueba?	5 = Si 1 = No	Usuario	5	5	5	5
		Las pruebas se generan mediante documentos/requerimientos	¿Permite obtener por documentos/requerimientos las acciones cotidianas del usuario en la Aplicación objeto de prueba?	5 = Si 1 = No	Usuario	1	1	1	1
		Ante errores/defectos permite continuar la ejecución	¿Permite continuar la ejecución de la prueba ante defectos en la Aplicación objeto de prueba?	5 = Si 1 = No	Usuario	1	5	1	1
		Capacidad de guardar pruebas para continuar después	¿Permite guardar pruebas para continuar después la ejecución?	5 = Si 1 = No	Usuario	5	5	5	5
	Plataformas de construcción de la Aplicación objeto de prueba (FUN 1.9.2)	Aplicación objeto de prueba basada en entorno Web	¿La Aplicación objeto de prueba está construida en entorno Web?	5 = Si 1 = No	Usuario	5	1	1	5
		Aplicación objeto de prueba basada en aplicación Windows	¿La Aplicación objeto de prueba está construida como aplicación Windows?	5 = Si 1 = No	Usuario	1	5	5	1
FUNCIONALIDAD (FUN) - Característica: Precisión (FUN 2)									
		Resultados incompletos	¿Existe falta de precisión en la función debido a resultados incompletos?	5=Ninguna 4=muy pocas 3=muchas 2=Casi todas 1=Todas	Usuario	4	4	4	3
		Resultados incorrectos	¿Existe falta de precisión en la función debido a resultados incorrectos?	5=Ninguna 4=muy pocas 3=muchas 2=Casi todas 1=Todas	Usuario	4	4	4	4
		Resultados emitidos no esperados	¿Existe falta de precisión en la función debido a la emisión de resultados no esperados?	5=Ninguna 4=muy pocas 3=muchas 2=Casi todas 1=Todas	Usuario	5	5	5	5
		Criterios de validación	¿Existen criterios de validación para el producto?	5=Completam def. 4=Casi todo def. 3= Med. Def. 2= Poco def. 1= No están def.	Usuario	5	5	5	5
Aceptación de pruebas realizadas (FUN 2.1)	Criterios de aceptación de pruebas (FUN 2.1.1)	Criterios de prueba superada	¿Permite establecer los criterios/condiciones para aceptar/superar la prueba?	5 = Si 1 = No	Usuario	5	5	5	5
		Criterios de prueba no superada	¿Permite establecer los criterios/condiciones para rechazar/no superar la prueba?	5 = Si 1 = No	Usuario	5	5	5	5
FUNCIONALIDAD (FUN) - Característica: Interoperabilidad (FUN 3)									
		Existencia funcionalidades pertenecientes al producto que son utilizadas por otros sistemas	¿Existen funcionalidades utilizadas por otros sistemas, que pertenecen al producto?	5 = Si tiene 1 = No tiene	Usuario	5	1	1	5
		Intercambio de datos	¿Existe intercambio de datos con otros sistemas?/Producto?	5 = Si tiene 1 = No tiene	Usuario	5	5	1	1
		Consistencia de interfaces	¿Existe consistencia con las interfaces de los otros sistemas?	5 = Si tiene 1 = No tiene	Usuario	5	5	1	1
		Existencia de funcionalidades de otros sistemas que sean necesarias	¿Existen funcionalidades en otros sistemas que son necesarias?	5 = Si tiene 1 = No tiene	Usuario	5	5	1	5

Interacción con otras aplicaciones (FUN 3.1)	Funcionamiento con terceros (FUN 3.1.1)	Componentes/plugins que complementan la Herramienta de Prueba	¿Tiene algún(o) componente/plugin que complementa y/o aumenta sus potencialidades?	5 = Si 1 = No	Usuario	5	5	1	5
		Dependencia de componentes/ plugins para funcionar	¿Necesita de algún(o) componente/plugin para poder funcionar?	5 = Si 1 = No	Usuario	5	1	1	5
	Operaciones con terceros (FUN 3.1.2)	Importar scripts de otras aplicaciones/lenguajes	¿Permite importar scripts desde otras Herramienta de Prueba del mismo fabricante?	5 = Si 1 = No	Usuario	5	5	1	5
			¿Permite importar scripts desde otras Herramienta de Prueba de distinto fabricante?	5 = Si 1 = No	Usuario	1	5	1	1
		Importar datos de prueba desde otros sistemas	¿Permite importar datos para alimentar los Datos de Prueba desde archivos planos?	5 = Si 1 = No	Usuario	5	1	5	5
			¿Permite importar datos para alimentar los Datos de Prueba desde archivos Excel?	5 = Si 1 = No	Usuario	5	5	5	5
			¿Permite importar datos para alimentar los Datos de Prueba desde sentencias sql?	5 = Si 1 = No	Usuario	5	5	5	5
	Almacenamiento en Base de Datos (FUN 3.1.3)	Manejo de Base de Datos propia	¿Posee Base de Datos propia donde almacena información relevante?	5 = Si 1 = No	Usuario	5	1	1	5
		Manejo de Base de Datos no propias	¿Almacena información relevante en Base de Datos no propietaria?	5 = Si 1 = No	Usuario	5	5	5	1
FUNCIONALIDAD (FUN) - Característica: Seguridad (FUN 4)									
Control de acceso (FUN 4.1)		Control de acceso	¿Qué tan completa es la detección de acceso a usuarios del sistema, cuando se trata de obtener acceso al sistema de forma no autorizada?	5=Muy alta 4=Alta 3= Mediana 2= Básica 1= No tiene.	Usuario	4	1	3	1
	Usuarios del sistema (FUN 4.1.1)	Definición de usuarios del sistema	¿Permite configurar usuarios de la Herramienta de Prueba de acuerdo a perfiles?	5 = Si 1 = No	Usuario	5	1	5	1
		Control de acceso a los usuarios	¿Mantiene registro/auditoría del acceso de los usuarios?	5 = Si 1 = No	Usuario	5	1	5	1
		Definición de roles dentro del sistema	¿Permite configurar roles para distinguir entre permisos de usuarios?	5 = Si 1 = No	Usuario	5	1	5	1
			¿Permite implementar roles para distinguir entre permisos de usuarios?	5 = Si 1 = No	Usuario	5	1	5	1
Capacidad de auditoría (FUN 4.2)		Capacidad de Auditoría	¿Qué tan completa es la auditoría? ¿Se evalúa la cantidad de registros de acceso que el sistema registró en la base de datos histórica?	5=Muy alta 4=Alta 3= Mediana 2= Básica 1= No tiene.	Usuario	4	4	5	5
	Registro de auditoría (FUN 4.2.1)	Registro de acceso a la Herramienta de Prueba	¿Permite registrar el acceso a los usuarios registrados?	5 = Si 1 = No	Usuario	5	5	5	1
		Registro de acciones en la Herramienta de Prueba	¿Permite registrar los cambios a configuraciones sobre la Herramienta de Prueba o sobre proyectos existentes?	5 = Si 1 = No	Usuario	5	5	5	5
		Log de auditoría (FUN 4.2.2)	Mantenimiento de log de auditoría	¿Mantiene los logs de Auditoría sobre tiempos definidos?	5 = Si 1 = No	Usuario	5	5	5
USABILIDAD (USA) - Característica: Facilidad de Comprensión (USA 1)									
		Complejidad de las demostraciones funcionales	¿Están completas las demostraciones funcionales implementadas respecto al número de funciones que requieren la demostración de uso?	5=Todas 4=Casi todas 3= Algunas 2= Pocas 1= Ninguna	Usuario	2	2	4	4
		¿Cuál es el tiempo en que un usuario adquiere las destrezas necesarias para usar el sistema?	¿Cuál es el tiempo en que un usuario adquiere las destrezas necesarias para usar el sistema?	5= 6 días o menos 4= 7-14 días 3= 15-30 días 2= 2-5 meses 1= 6 meses o más	Usuario	3	3	4	4
		Nivel de dificultad de la aplicación	Nivel de dificultad de la aplicación	5=Muy fácil 4=Fácil 3=Promedio 2= Difícil 1=Muy difícil	Usuario	4	2	4	5

		Facilidad para ubicar funcionalidades	¿Es fácil ubicar las funcionalidades del sistema?	5= Muy fácilmente 4= Fácilmente 3= Dificultad media 2= Difícil 1= Muy difícil	Usuario	4	4	4	5
USABILIDAD (USA) - Característica: Capacidad de Aprendizaje (USA 2)									
		Calidad en la clasificación de los temas	¿Es fácil utilizar el material de apoyo, por una buena clasificación de los temas	5=Excelente 4=Buena 3= Promedio 2=Debajo del Prom. 1= Inaceptable	Usuario	3	3	5	5
		Compleitud de la especificación de los documentos	¿Están identificados todos los documentos a ser producidos por el proceso o proyecto?	5=Todos 4=Casi todos 3= Algunos 2= Pocos 1= Ninguno	Líder Usuario	4	4	4	4
		Wizard para ciertas funcionalidades	¿Posee ayudas guiadas en la instalación o configuración de ciertas bondades de la Herramienta de Prueba?	5 = Si 1 = No	Usuario	5	1	5	5
		Documentación de la ayuda	¿Ofrece ayuda en línea para las diversas funcionalidades de la Herramienta de Prueba?	5 = Si 1 = No	Usuario	5	1	5	5
			¿Provee manuales de usuario para las diversas funcionalidades de la Herramienta de Prueba?	5 = Si 1 = No	Usuario	5	5	5	5
			¿Provee manual de instalación para la Herramienta de Prueba?	5 = Si 1 = No	Usuario	5	5	5	5
			¿Existen foros de discusión o ayuda en internet?	5 = Si 1 = No	Usuario	1	5	1	5
USABILIDAD (USA) - Característica: Interfaz Gráfica (USA 3)									
		Interfaz personalizable	¿La interfaz es personalizable?	5 = Si 1 = No	Usuario	5	1	5	1
		Localización rápida de opciones	¿Se localizan rápidamente las opciones, consistente la ubicación de utensilios?	5 = Si 1 = No	Usuario	1	1	5	5
		Presencia de metáforas	¿Hay presencia de metáforas?	5 = Si 1 = No	Usuario	5	5	5	5
		Satisfacción del Diseño visual	¿Es bueno el diseño visual de las pantallas?	5=Excelente 4=Buena 3=Promedio 2=Debajo del Prom. 1= Inaceptable	Usuario	4	3	5	4
		Versatilidad de la navegación	¿Es versátil la navegación, entre pantallas?	5=Excelente 4=Buena 3=Promedio 2=Debajo del Prom. 1= Inaceptable	Usuario	4	3	5	4
		Barra de herramientas personalizables	¿La barra de herramientas es personalizable?	5 = Si 1 = No	Usuario	5	1	5	1
		Accesos rápidos a opciones	¿Posee acceso rápido a funciones a través de las teclas?	5 = Si 1 = No	Usuario	5	1	5	1
USABILIDAD (USA) - Característica: Operabilidad (USA 4)									
		Satisfacción con las ayudas disponibles respecto a las requeridas	¿Cómo es la satisfacción con las ayudas disponibles respecto a las requeridas?	5=Excelente 4=Buena 3=Promedio 2=Debajo del Prom. 1= Inaceptable	Usuario	5	5	5	5
		Tasa de disponibilidad de valores por defecto	¿Existen valores por defecto en las pantallas? (N° de valores por defecto en pantalla / N° de entradas en la pantalla)	5=Siempre 4=Casi siempre 3= Algunas veces 2= Pocas veces 1= Nunca	Usuario	4	4	4	3
		Evaluación continua de la Satisfacción del cliente	¿Existe una evaluación continua de la satisfacción del cliente respecto a los servicios de apoyo	5=Siempre 4=Casi siempre 3= Algunas veces	Usuario	4	3	5	5

			y al producto?	2= Pocas veces 1= Nunca					
		Soporte operacional continuo	¿Se provee apoyo para resolver problemas operacionales según requerimientos del cliente?	5=Siempre 4=Casi siempre 3= Algunas veces 2= Pocas veces 1= Nunca	Usuario	3	3	3	3
		Operación apropiada del software en su ambiente	¿Existe una correcta operación del software en su ambiente?	5 = Si 1 = No	Usuario	5	5	5	5
		Asistencia a los clientes en cuanto a operación	¿Existe asistencia a los clientes en cuanto a su operación?	5=Siempre 4=Casi siempre 3= Algunas veces 2= Pocas veces 1= Nunca	Usuario	4	4	4	4
		Lista de posibles valores a ingresar/elegir en campos funcionales	¿Despliega lista de valores tipos para alguna funcionalidad en especial?	5 = Si 1 = No	Usuario	5	5	1	5
		Operación de la Aplicación objeto de prueba con la Herramienta de Prueba	¿Cuán fácil es operar la aplicación objeto de prueba dentro de la herramienta de prueba?	5=Muy fácil 4=Fácil 3=Promedio 2=Difícil 1=Muy difícil	Usuario	5	4	4	4
FIBILIDAD (FIA) - Característica: Tolerancia a Fallas (FIA 2)									
		Anulación de operación incorrecta	¿Qué tan frecuente puede el producto de software <i>evitar</i> fallas, aún si el usuario lo opera incorrectamente?	5=Siempre 4=Casi siempre 3= Algunas veces 2= Pocas veces 1= Nunca	Usuario	4	4	4	4
FIBILIDAD (FIA) - Característica: Recuperación (FIA 3)									
		Capacidad de reiniciar	¿Puede el producto de software recuperarse fácilmente después de una caída?	5=Muy fácilmente 4=Fácilmente 3=Dificultad Media 2= Difícil 1= Muy difícil	Usuario	3	3	4	5
		Velocidad de reinicialización	Velocidad de la reinicialización.	5=Muy rápida 4=Rápida 3= Promedio 2=Lenta 1= Muy lenta	Usuario	3	3	3	3
		Disponibilidad del producto de software	¿Pueden los usuarios trabajar con el producto de software el tiempo necesario?	5=Siempre 4=Casi siempre 3= Algunas veces 2= Pocas veces 1= Nunca	Usuario	5	5	5	5
EFICIENCIA (EFI) - Característica: Comportamiento del Tiempo (EFI 1)									
		Tiempo de respuesta	Tasa del tiempo de respuesta para lo que es diseñado a ser ejecutado	5=Muy rápida 4=Rápida 3= Promedio 2=Lenta 1= Muy lenta	Usuario	4	3	3	4
		Satisfacción con el número de tareas completadas en un tiempo especificado	Satisfacción con el número de tareas que pueden ser completadas durante un tiempo especificado con respecto al número de tareas que son requeridas a ser completadas	5=Muy Alta 4=Alta 3= Media 2= Baja 1= Muy baja	Usuario	4	4	4	4
EFICIENCIA (EFI) - Característica: Utilización de recursos (EFI 2)									
		Utilización del CPU	Tasa de tiempo de CPU utilizado (Tiempo de CPU utilizado / Tiempo de CPU requerido)	5=Muy rápida 4=Rápida 3= Promedio 2=Lenta 1= Muy lenta	Usuario	5	5	5	5
		Utilización de memoria	Tasa del tamaño de memoria utilizada con respecto al tamaño de la memoria que se requiere que sea usada (Memoria utilizada / Requerimientos del uso de memoria)	5=Muy baja 4=Baja 3= Media 2=Alta 1= Muy Alta	Usuario	4	4	5	4

ANEXO VII: Resultados Evaluación de Características generales del modelo propuesto

Evaluación de las categorías del modelo

<i>Categoría (Carac. General)</i>	Pertinencia	Complejidad	Adecuación	Precisión
Funcionalidad (FUN)	1	1	1	1
Usabilidad (USA)	1	1	1	1
Fiabilidad (FIA)	1	1	1	1
Eficiencia (EFI)	1	1	1	1

Evaluación de características del modelo

<i>Categoría</i>	<i>Característica (Carac. específica)</i>	Pertinencia	Complejidad	Adecuación	Precisión
Funcionalidad (FUN)	FUN 1. Ajuste a los propósitos	1	1	1	1
	FUN 2. Precisión	1	1	1	1
	FUN 3. Interoperabilidad	1	1	1	1
	FUN 4. Seguridad	1	1	1	1
Usabilidad (USA)	USA 1. Usabilidad	1	1	1	1
	USA 2. Capacidad de Aprendizaje	1	1	1	1
	USA 3. Interfaz Gráfica	1	1	1	1
	USA 4. Operabilidad	1	1	1	1
Fiabilidad (FIA)	FIA 2. Tolerancia a Fallas	1	1	1	1
	FIA 3. Recuperación	1	1	1	1
Eficiencia (EFI)	EFI 1. Comportamiento del tiempo	1	1	1	1
	EFI 2. Utilización de recursos	1	1	1	1

Evaluación de sub-características del modelo

<i>Categoría</i>	<i>Característica</i>	<i>Sub-Subcaracterística (Carac. específica)</i>	Pertinencia	Complejidad	Adecuación	Precisión
Funcionalidad (FUN)	FUN 1. Ajuste a los propósitos	(FUN 1.1) Niveles de Prueba	1	1	1	1
		(FUN 1.2) Tipos de Prueba	1	1	1	1
		(FUN 1.3) Enfoque de Prueba	1	1	1	1
		(FUN 1.4) Tipo de Licencias de distribución de la Herramienta de Prueba	0	1	1	1
		(FUN 1.5) Administración de la Herramienta de Prueba	1	1	0	1
		(FUN 1.6) Entregables de Prueba	1	1	1	0
		(FUN 1.7) Planificación de Prueba	1	1	1	1
		(FUN 1.8) Gestión de Resultados	1	1	1	1
		(FUN 1.9) Aplicación objeto de Prueba	1	1	1	1
	FUN 2. Precisión	(FUN 2.1) Aceptación pruebas realizadas	1	1	1	1
	FUN 3. Interoperabilidad	(FUN 3.1) Interacción con otras aplicaciones	1	1	1	1
	FUN 4. Seguridad	(FUN 4.1) Control de acceso	0	1	1	1
		(FUN 4.2) Capacidad de auditoría	1	1	1	1

Evaluación de sub-sub-características del modelo

<i>Categoría</i>	<i>Característica</i>	<i>Sub-Subcaracterística</i>	<i>Sub-sub-Subcaracterística (Característica específica)</i>	Pertinencia	Complejidad	Adecuación	Precisión
Funcionalidad (FUN)	FUN 1. Ajuste a los propósitos	(FUN 1.1) Niveles de Prueba	Pruebas de sistema (FUN 1.1.1)	1	1	1	1
		(FUN 1.2) Tipos de Prueba	Pruebas de Desempeño (FUN1.2.1)	1	1	1	1
			Pruebas de Carga/Estrés (FUN1.2.2)	1	1	1	1
		(FUN 1.3) Enfoque de Prueba	Pruebas de Caja Negra (FUN 1.3.1)	1	1	1	1
		(FUN 1.4) Tipo de Licencias de	Open source (FUN 1.4.1)	1	1	1	1

		distribución de la Herramienta de Prueba	Comercial (FUN 1.4.2)	1	1	1	1
		(FUN 1.5) Administración de la Herramienta de Prueba	Lenguaje de Programación de la Herramienta de Prueba (FUN 1.5.1)	1	0	1	1
		(FUN 1.6) Entregables de Prueba	Casos de Prueba (FUN 1.6.1)	1	1	1	1
			Datos de Prueba (FUN 1.6.2)	1	1	1	1
			Script de Pruebas (FUN 1.6.3)	1	1	1	1
			Suite de Pruebas (FUN 1.6.4)	1	0	1	1
			Salidas de pruebas (FUN 1.6.5)	1	1	1	1
			Relación entre entregables de Pruebas (FUN 1.6.6)	1	1	1	0
		(FUN 1.7) Planificación de Prueba	Condiciones a probar en la Aplicación objeto de prueba (FUN 1.7.2)	1	1	1	1
		(FUN 1.8) Gestión de Resultados	Estadísticas de resultados (FUN 1.8.1)	1	1	1	1
			Análisis de resultados (FUN 1.8.2)	1	1	1	1
			Manipulación de resultados (FUN 1.8.3)	0	1	1	1
		(FUN 1.9) Aplicación objeto de Prueba	Interacción con la Aplicación objeto de prueba (FUN 1.9.1)	1	1	1	1
			Plataformas de construcción de la Aplicación objeto de prueba (FUN 1.9.2)	1	1	1	1
	FUN 2. Precisión	(FUN 2.1) Aceptación de pruebas realizadas	Criterios de aceptación de pruebas (FUN 2.1.1)	1	1	1	1
	FUN 3. Interoperabilidad	(FUN 3.1) Interacción con otras aplicaciones	Funcionamiento con terceros (FUN 3.1.1)	1	0	1	1
			Operaciones con terceros (FUN 3.1.2)	0	1	1	1
			Almacenamiento en Base de Datos (FUN 3.1.3)	1	1	1	1
	FUN 4. Seguridad	(FUN 4.1) Control de acceso	Usuarios del sistema (FUN 4.1.1)	1	0	1	1
		(FUN 4.2) Capacidad de auditoría	Registro de auditoría (FUN 4.2.1)	1	1	1	1

ANEXO VIII: Resultados Evaluación de Métricas del modelo propuesto

Característica	Sub- Característica	Sub-sub- Característica	Métrica (Característica Específica)	Pertin.	Factibi- lidad	Profun- didad	Escala
CATEGORIA – F U N C I O N A L I D A D							
FUN 1. Ajuste a los propósitos			Cumplimiento de las necesidades funcionales	1	1	1	1
			Definición de las necesidades de adquisición, desarrollo o mejora	1	1	1	1
			Definición de la estrategia de adquisición o desarrollo de nuevos sistemas	1	1	1	1
			Definición del criterio de aceptación	1	1	1	1
			Registro de los resultados de la prueba	1	1	1	1
	(FUN 1.1) Niveles de Prueba	Pruebas de Sistema (FUN 1.1.1)	Ejecución de pruebas de a nivel de sistema	1	1	1	1
	(FUN 1.2) Tipos de Prueba	Pruebas de Desempeño (FUN 1.2.1)	Ejecución del Tipo de Pruebas de Desempeño	1	1	1	1
		Pruebas de Carga/Estrés (FUN 1.2.2)	Ejecución del Tipo de Pruebas de Carga/Estrés	1	1	1	1
	(FUN 1.3) Enfoque de Prueba	Pruebas de Caja Negra (FUN 1.3.1)	Ejecución de Pruebas con enfoque de Caja Negra	1	1	1	1
			Exploración en el código de la Aplicación objeto de prueba	1	1	1	1
	(FUN 1.4) Tipo de Licencias de distribución de la Herramienta de Prueba	Open source (FUN 1.4.1)	Distribución de la Herramienta de Prueba Opensource	1	1	1	1
		Comercial (FUN 1.4.1)	Distribución de la Herramienta de Prueba es comercial	1	1	1	1
	(FUN 1.5) Administración de la Herramienta de Prueba	Lenguaje de Programación de la Herramienta de Prueba (FUN 1.5.1)	Código único/proceso unificado para multiplataformas	1	1	1	1
			Código Propietario (propone comandos, funciones propietarias)	1	1	1	1
			Necesidad de amplios conocimientos en programación	1	1	1	1
	(FUN 1.6) Entregables de Prueba	Casos de Prueba (FUN 1.6.1)	Generación automática de Casos de Prueba	1	1	1	1
			Construcción de Casos de Prueba paso a paso	1	1	1	1
			Identificación de los requerimientos de prueba	1	1	1	1
			Reutilización de Casos de Prueba entre proyectos	0	1	1	1
			Documentación de Pre-condiciones	1	1	1	1
			Documentación de Post-condiciones	1	1	1	1
			Documentación de Post-condiciones	1	1	1	1
		Datos de Prueba (FUN 1.6.2)	Generación manual de Datos de Prueba desde Casos de Prueba	1	1	1	1
			Generación automática de Datos de Prueba desde datos reales	1	1	1	1
			Generación automática de Datos de Prueba desde sentencias SQL	1	1	1	1
			Generación automática de Datos de Prueba de forma aleatoria	1	1	1	1
			Reutilización de Datos de Prueba entre proyectos	0	1	1	1
		Script de	Generación automática de Scripts	1	1	1	1

		Pruebas (FUN 1.6.3)	de Prueba desde Casos de Prueba + Datos de Prueba				
			Construcción de Scripts de Prueba paso a paso	1	1	1	1
			Modificar Scripts de Prueba luego de generarlos	1	1	1	1
			Modificar Scripts de Prueba en línea/ejecución	1	1	1	1
			Reutilización de Scripts de Prueba entre proyectos	0	1	1	1
		Suite de Pruebas (FUN 1.6.4)	Organizar/secuenciar conjunto de pruebas	1	1	1	1
			Programación de Scripts de Prueba en el tiempo	1	1	1	1
		Salidas de pruebas (FUN 1.6.5)	Generación de evidencias de Pruebas	1	1	1	1
			Gestión de defectos	1	1	1	1
			Log de pruebas/histórico de ejecuciones	1	1	1	1
		Relación entre entregables de Pruebas (FUN 1.6.6)	Relación entre requerimientos de prueba y Casos de Prueba	1	1	1	1
			Relación entre Casos de Prueba y Script de Prueba	1	1	1	1
			Relación de defectos a Casos de Prueba	1	1	1	1
			Relaciona defectos a su causa	1	1	1	1
	(FUN 1.7) Planificación de Prueba	Condiciones a probar en la Aplicación objeto de prueba (FUN 1.7.2)	Emulación de usuarios concurrentes	1	1	1	1
			Emulación de transacciones concurrentes	1	1	1	1
			Variación de carga de trabajo durante la prueba	1	1	1	1
			Definición de límites/umbrales de finalización de prueba	1	1	1	1
			Definición de estrategias de pruebas	1	1	0	1
			Cancelación de pruebas al superar umbrales	1	1	1	1
	(FUN 1.8) Gestión de Resultados	Estadísticas de resultados (FUN 1.8.1)	Estadísticas en tiempo real	1	1	1	1
			Estadísticas post-mortem	1	1	1	1
			Notifica superación de umbrales al momento	1	1	0	1
			Generación de métricas	1	1	1	1
		Análisis de resultados (FUN 1.8.2)	Provee de reportes predeterminados para gestionar resultados	1	1	1	1
			Permite modificar los reportes predeterminados	1	1	1	1
			Permite construir reportes varios de resultados	1	1	1	1
			Provee gráficas predeterminados para gestionar resultados	1	1	1	1
			Permite modificar los gráficos predeterminados	1	1	1	1
			Permite construir gráficos varios de resultados	1	1	1	1
			Análisis de resultados automáticos	1	1	1	1
		Manipulación de resultados (FUN 1.8.3)	Exportar resultados en formatos intercambiables	1	1	1	1
			Formato de generación de Reportes/entregables son similares entre sí	0	1	1	1
			Disponibilidad de los resultados obtenidos (base de datos)	1	1	1	1
	(FUN 1.9) Aplicación objeto de Prueba	Interacción con la Aplicación objeto de prueba (FUN 1.9.1)	Sincronización de Scripts de Prueba por nuevas versiones en la Aplicación objeto de prueba	1	0	1	1
			Las pruebas se generan al grabar acciones posibles de los usuarios	1	1	1	1
			Las pruebas se generan especificando manualmente las	1	1	1	1

			acciones posible				
			Las pruebas se generan mediante documentos/requerimientos	1	1	1	1
			Ante errores/defectos permite continuar la ejecución	1	1	1	1
			Capacidad de guardar pruebas para continuar después	1	1	1	1
		Plataformas de construcción de la Aplicación objeto de prueba (FUN 1.9.2)	Aplicación objeto de prueba basada en entorno Web	1	1	1	1
			Aplicación objeto de prueba basada en aplicación Windows	1	1	1	1
FUN 2. Precisión			Resultados incompletos	1	1	1	1
			Resultados incorrectos	1	1	1	1
			Resultados emitidos no esperados	1	1	1	1
			Criterios de validación	1	1	1	1
	Aceptación de pruebas realizadas (FUN 2.1)	Criterios de aceptación de pruebas (FUN 2.1.1)	Criterios de prueba superada	1	1	1	1
			Criterios de prueba no superada	1	1	1	1
FUN 3. Interoperabilidad			Existencia de funcionalidades pertenecientes al producto que son utilizadas por otros sistemas	1	1	1	1
			Intercambio de datos	1	1	1	1
			Consistencia de interfaces	1	1	1	1
			Existencia de funcionalidades de otros sistemas que sean necesarias	1	1	1	1
	(FUN 3.1) Interacción con otras aplicaciones	Funcionamiento con terceros (FUN 3.1.1)	Componentes/plugins que complementan la Herramienta de Prueba	1	1	1	1
			Dependencia de componentes/plugins para funcionar	1	1	1	1
		Operaciones con terceros (FUN 3.1.2)	Importar scripts de otras aplicaciones/lenguajes	1	1	1	1
			Importar datos de prueba desde otros sistemas	1	1	1	1
		Almacenamiento en Base de Datos (FUN 3.1.3)	Manejo de Base de Datos propias	1	1	1	1
			Manejo de Base de Datos no propias	1	1	1	1
FUN 4. Seguridad	(FUN 4.1) Control de acceso		Control de acceso	1	1	0	1
		Usuarios del sistema (FUN 4.1.1)	Definición de usuarios del sistema	1	1	1	1
			Control de acceso a los usuarios	1	1	1	1
			Definición de roles dentro del sistema	1	1	1	1
	(FUN 4.2) Capacidad de auditoría		Capacidad de Auditoría	1	1	1	1
		Registro de auditoría (FUN 4.2.1)	Registro de acceso a la Herramienta de Prueba	1	1	1	1
			Registro de acciones en la Herramienta de Prueba	1	1	1	1
		Log de auditoría (FUN 4.2.2)	Mantenimiento de log de auditoría	1	1	1	1
Característica	Sub- Característica	Sub-sub- Característica	Métrica (Característica Específica)	Pertin.	Factibilidad	Profundidad	Escala
CATEGORIA – U S A B I L I D A D							
Facilidad de comprensión (USA 1)			Complejidad de las demostraciones funcionales	1	1	1	1
			¿Cuál es el tiempo en que un usuario adquiere las destrezas necesarias para usar el sistema?	1	1	1	1
			Nivel de dificultad de la aplicación	1	1	1	1
			Facilidad para ubicar funcionalidades	1	1	1	1
Capacidad de Aprendizaje (USA 2)			Calidad en la clasificación de los temas	1	1	1	1
			Complejidad de la especificación de los documentos	1	1	1	1

			Wizard para ciertas funcionalidades	1	1	1	1
			Documentación de la ayuda	1	1	1	1
Interfaz Gráfica (USA 3)			Interfaz personalizable	1	1	1	1
			Localización rápida de opciones	1	1	1	1
			Presencia de metáforas	1	1	1	1
			Satisfacción del Diseño visual	1	1	1	1
			Versatilidad de la navegación	1	1	1	1
			Barra de herramientas personalizables	1	1	1	1
			Accesos rápidos a opciones	1	1	1	1
Operabilidad (USA 4)			Satisfacción con las ayudas disponibles respecto a las requeridas	1	1	1	1
			Tasa de disponibilidad de valores por defecto	1	1	1	1
			Evaluación continua de la Satisfacción del cliente	1	1	1	1
			Soporte operacional continuo	1	1	1	1
			Operación apropiada del software en su ambiente	1	1	1	1
			Asistencia a los clientes en cuanto a operación	1	1	1	1
			Lista de posibles valores a ingresar/elegir en campos funcionales	1	1	1	1
Característica	Sub- Característica	Sub-sub- Característica	Métrica (Característica Específica)	Pertin.	Factibi- lidad	Profun- didad	Escala
CATEGORIA – F I A B I L I D A D							
Tolerancia a Fallas (FIA 2)			Anulación de operación incorrecta	1	1	1	1
Recuperación (FIA 3)			Capacidad de reiniciar	1	1	1	1
			Velocidad de reinicialización	1	1	1	1
			Disponibilidad del producto de software	1	1	1	1
Característica	Sub- Característica	Sub-sub- Característica	Métrica (Característica Específica)	Pertin.	Factibi- lidad	Profun- didad	Escala
CATEGORIA – E F I C I E N C I A							
Comportamiento del tiempo (EFI 1)			Tiempo de respuesta	1	1	1	1
			Satisfacción con el número de tareas completadas en un tiempo especificado	1	1	1	1
Utilización de recursos (EFI 2)			Utilización del CPU	1	1	1	1
			Utilización de memoria	1	1	1	1

REFERENCIAS BIBLIOGRÁFICAS

American Society for Quality (2007) *Glosary*, USA. Recuperado en Agosto 03, 2007 de <http://www.asq.org/glossary/>

Asamblea Nacional de la Republica Bolivariana de Venezuela (2002) *Ley orgánica del sistema Venezolano para la Calidad*. Venezuela.

Automated Testing Institute - ATI (2010) *Testing, techniques & tools*, USA. Recuperado en Marzo 05, 2010 de <http://www.automatedtestinginstitute.com/home/>

Basili, V.R., Caldiera, C., Rombach H.D. (1994) *Goal Questions Metric Paradigm. Encyclopedia of Software Engineering*. USA: John Wiley & Sons.

Baskerville, Richard (1999) *Organizational and Social Perspectives on Information Technology* (1er ed.) USA: Springer.

Berkhout, Michiel et. al. (2001) Office of Government Commerce. *Best Practice Best Service – Service Support*. USA.

Braude, Eric J. (2003) *Ingeniería de Software: Una perspectiva orientada a objetos* (1er ed.). México: Alfaomega Grupo Editor.

Bruegge, Bernd; Dutoit, Allen H. (2002) *Ingeniería de Software Orientado a Objetos* (2da ed.). México: Prentice Hall.

CMMI-DEV,V1.2 (2006) CMMI® for Development, Version 1.2. USA: Software Engineering Institute – Carnegie Mellon.

Carrasquel, Raúl (2008) *Los canales de Distribución de las PYMEs en Venezuela: un estudio exploratorio*. Maestría en Administración de Empresas. USB. Venezuela.

Carretero G., María J. (2006) *Libro web sobre ética aplicada a la informática*. Universidad de Málaga. España.

Castejón, Adolfo (2005) *Tendencias del sector de las tecnologías de información y comunicaciones (TIC) en Venezuela*. Venezuela Tecnológica: Presente y Futuro (1er ed.). Venezuela: CONAPRI.

Checkland, Peter (2003) *Pensamiento de sistemas-Practica de sistemas* (1er ed.). Mexico: Limusa.

Crosby, Philip B. (1980) *Quality is free: The Art of Making Quality certain* (1era. ed.) USA: Signet Shakespeare.

De Luca, L., Mendoza, L., Pérez, M., Rojas, T. (2001) *Indicators for the selection of software quality management tools*. CLEI 2001, Mérida, Venezuela.

Díaz, A., Ovalles, M., Mendoza, L., Rojas, T. (2001) *Aplicación de un modelo de calidad para la evaluación de herramientas CASE adaptado al estándar*

internacional ISO/IEC 14102. 1ras Jornadas Iberoamericanas de Ingenierías de Software e Ingeniería del Conocimiento, Buenos Aires, Argentina.

Di Paula, George; Parada, Dakar; Pérez, María A.; Mendoza, Luis (2007) Proyecto: *Agilidad y disciplina del proceso de desarrollo de software para las Pequeñas y Medianas Empresas (PYMEs) y las Cooperativas en Latinoamérica. Caso: Venezuela*. Jornadas Iberoamericanas de Ingeniería del Software e Ingeniería del Conocimiento.

Dustin, Elfriede (2003) *Effective software testing: 50 specific ways to improve your testing*. (1era. ed.) USA: Addison-Wesley.

Erard, Pedro (1999). *¿Cómo vencer los obstáculos al desarrollo de la pyme?* El aporte de Fundes. *Debates Iesa*, Caracas Venezuela 1999. 5, 9-19.

Granados, Gladys; Murúa, Mario (2008) *Estado de La Industria Venezolana de Software: Informe Técnico 2007*. Centro de Excelencia en Ingeniería del Software (CEISoft) de la Corporación Parque Tecnológico de Mérida

Grimán, Anna C.; Pérez, María A.; Mendoza, Luis E. (2003) *Estrategia de Pruebas para Software OO que garantiza Requerimientos No Funcionales*. III Workshop de Ingeniería de Software, Chillán, Chile.

IEEE Std. 610.12-1990 *Standard Glossary of Software Engineering Terminology*.

IEEE Std. 730-2002 *Standard for Software Quality Assurance Plans*.

International Standard (1995) *Iso/Iec 12207 Software Life Cycle Processes*. USA: Federal Aviation Administration.

ITIL® Glossary v01, (2006) best-management-practice.com. *Glosario de Términos ITIL®, Definiciones y Acrónimos*. USA. Recuperado en Noviembre 24, 2007 de http://www.best-management-practice.com/gempdf/ITILV3_Glossary_Spanish_v3.1.24.pdf

Iovine V., Doménico (2009) Implementación de mejoras de una herramienta en Software libre para las pruebas de software. Venezuela. UNEFA.

James, Paúl (2000) *Gestión de la calidad total: un texto introductorio* (1era. ed.). España. Pearson Educación.

Jiménez, Carlos (2006) *Las Pymes y las TIC: Hasta qué punto las Pymes han incorporado el desarrollo de las tecnologías de la información y comunicación*. Presentado en el evento: “Perspectivas y Nuevas Realidades de la Pequeña y Mediana Empresa”, Venamcham, Venezuela.

Juran, Joseph M., Godfrey, A. Blanton (1998) *Juran's Quality Handbook* (5ta. ed.) USA: McGraw-Hill Professional.

Kitchenham, Ba (1996) *DESMET: A methodology for evaluating software engineering methods and tools*. USA: Keele University Technical Report.

Kruchten, Philippe (2000) *The Rational Unified Process: An introduction* (2da. ed.). USA: Addison-Wesley.

Lacy, Shirley; Macfarlane, Ivor (2007) best-management-practice.com. *Service Transition*. USA. Recuperado en Noviembre 24, 2007 de <http://www.best-management-practice.com/bookstore.asp?FO=1229358&DI=591438>

Llorens F., Juan (2005) *Gerencia de Proyectos de Tecnología de información* (1ra ed.). Venezuela: Los libros de El Nacional.

Office of Government Commerce – OGC (2007), best-management-practice.com *Continual Service Improvement*. USA. Recuperado en Noviembre 24, 2007 de <http://www.best-management-practice.com/Online-Bookshop/IT-Service-Management-ITIL/ITIL-Version-3/Continual-Service-Improvement/PDF/>

Leffingwell, Dean; Widrig, Don (2006) *Managing Software Requirements, a Use Case Approach*. (2da. Ed.) USA: Addison-Wesley, Pearson Education.

Matos, Giancarlo; Anglés, Rafael y Ortiz, Lourdes (2004) *Modelo de apoyo en la selección de la metodología para un desarrollo de software, basado en la gerencia del conocimiento y soportado en tecnología de Información*. LIV Convención Anual ASOVAC. Venezuela.

Méndez, Edumilis; Pérez, María A.; Mendoza, Luis E. (2007) *Aplicación de un Método para Especificar Casos de Prueba de Software en la Administración Pública*. Actas de Talleres de Ingeniería del Software y Bases de Datos, Vol. 1, No. 4, 2007

Méndez, E., Pérez, M., Grimán, A. & Mendoza, L. (2005). *¿Cómo se relacionan la calidad sistémica y la productividad en el proceso de desarrollo de software?*. Iberoamericana de Sistemas, Cibernética e Informática, 2 (2), 1-8.

Mendoza, L., Grimán, A., Pérez, M. (2004) *Modelo integrado para soportar la selección de herramientas CASE*. IV Workshop de Ingeniería del Software - XII Jornadas Chilenas de Computación, Arica, Chile.

Mendoza, Luis E.; Pérez, Maria A.; Grimán, Anna (2005). *Prototipo de Modelo Sistémico de Calidad (MOSCA) del Software*. Mexico. Computación y Sistemas Vol. 8 Núm. 3, pp. 196-217.

Methodius (2005) *Métodos y modelos de desarrollo de software para empresas venezolanas*. Proyecto de Investigación Nro. 2005000165 del FONACIT. Venezuela. Recuperado en Mayo 15, 2010 de <http://www.methodius.org.ve/>

Montilva C., Jonas A. (2007). *La Industria de Software en Venezuela: Una caracterización de su recurso*. Presentación del I Encuentro Nacional sobre Tecnologías de Información e Ingeniería de Software EVETIS'07. Venezuela.

Mora B., Freddy (2003) *El marco jurídico regulatorio de las pequeñas y medianas empresas (PYMEs) en Venezuela*. Venezuela: VISIÓN GERENCIAL.

Omaña M., Mayuri S. (2009) *Modelo de calidad basado en características para la selección de un sistema de gestión de aprendizaje (LMS-Learning Managment System)*. Maestría en Sistemas de Información. UCAB. Venezuela.

Perry, William E. (2006) *Effective Methods for Software Testing* (3era ed.) USA: Wiley Publishing, Inc

Piattini V., Mario G.; García R., Félix O.; Caballero M., Ismael (2007) *Calidad De Sistemas Informáticos* (1era ed.) México. Alfaomega, Ra-Ma.

Pressman, Roger S. (2005) *Ingeniería del software: un enfoque práctico* (6ta ed.). México. McGraw-Hill.

Qualitatis.org (2006) *Antítesis (Ágiles)* USA. Recuperado en Agosto 03, 2007 de <http://www.qualitatis.org/>

Rational Unified Process® Version 2003.06.15 © Copyright IBM Corp. 1987-2005

Real Academia Española (2001) *Diccionario de la lengua española* (22da. ed.). España. Disponible en www.rae.es

Reyes, Esteban (2005) *La industria del software en Venezuela: Pasado, presente y futuro*. Venezuela Tecnológica: Presente y Futuro (1er ed.). Venezuela: CONAPRI.

Rivero, D.; Montilva, J; Granados, G.; Barrios, J.; Besembel, I.; Sandia, B. (2007) *La Industria de Software en Venezuela: Una Caracterización de su Recurso Humano*. Encuentro Venezolano sobre Tecnologías de la Información e Ingeniería del Software (EVETIS'07).

Sánchez, Amarilis; Romero, Engelbert; Zúñiga, Roy. (2001) *Proyecto Mejorando la Competitividad del Cluster de Software en Venezuela: Diagnóstico y Benchmarking*. Instituto Centroamericano de Administración de Empresas INCAE. Recuperado en Mayo 08, 2008 de http://www.cid.harvard.edu/archive/andes/documents/workingpapers/microfoundations/it_venezuela/diagnostico_software_venezuela.pdf

Schulmeyer, g. Gordon; McManus, James I. (1993) *Total Quality Management for software* (1ra ed.) USA: Van Nostrand Reinhold. Kenneth S. Mendis, artículo: "Evaluación de la calidad del software".

Software Engineering Institute – Carnegie Mellon (2007) *SEI Areas of work: Management*. USA. Recuperado en Julio 24, 2007 de <http://www.sei.cmu.edu/>

Software QA and Testing Resource Center (2007) *Software QA and Testing Frequently-Asked-Questions, Part 1 by Hower Rick*. USA. Recuperado en Agosto 03, 2007 de http://www.softwareqatest.com/qatfaq1.html#FAQ1_1

Sommerville, Ian (2006) *Software Engineering* (8ta ed.). USA: Addison-Wesley.

Susman, G., y Evered, R. (1978) *An assessment of the scientific merit of action research*. Administrative Science Quarterly.

SWEBOK (2004) *Guide to the Software Engineering Body of Knowledge* (IEEE – 2004 Version). USA: A project of the IEEE Computer Society Professional Practices Committee.

Taylor, Sharon (2007) Axios System. *ITIL V3 - The Future is Here*. USA. Recuperado en en Noviembre 24, 2007 de [http://www.best-management-practice.com/gempdf/ITILV3 The Future is Here Sharon Taylor.pdf](http://www.best-management-practice.com/gempdf/ITILV3%20The%20Future%20is%20Here%20Sharon%20Taylor.pdf)

The Standish Group International (2003) *CHAOS Chronicles Version III*. USA. Recuperado en Abril 19, 2008 de <http://www.standishgroup.com/chaos/toc.php>

Utting, Mark; Legeard, Bruno (2006) *Practical Model-Based Testing: A Tools Approach* (1ra ed.) USA: Morgan-Kaufmann.

Unhelkar, Bhuvan (2003) *Process Quality Assurance for UML-Based Projects* (1ra ed.). USA: Addison Wesley.

Wells, Don (2000) [extremeprogramming.org](http://www.extremeprogramming.org). *Extreme Programming: A gentle introduction*. USA. Recuperado en Agosto 13, 2007 de <http://www.extremeprogramming.org/>

Yáber, Guillermo y Valarino, Elizabeth (2006) *Clasificación, organización y gestión de la investigación en los postgrados de administración y gerencia*. Venezuela: Grupo de Creatividad y Productividad. Universidad Simón Bolívar.