ESTUDIO COMPARATIVO DE HERRAMIENTAS PARA LA GESTION DE CASOS DE PRUEBAS FUNCIONALES

VIVIANA MARCELA HERNANDEZ

ALEXANDRA KUJAR MENA

UNIVERSIDAD DE MEDELLÍN

FACULTAD DE INGENIERIA DE SISTEMAS

MEDELLÍN

2010

ESTUDIO COMPARATIVO DE HERRAMIENTAS PARA LA GESTION DE CASOS DE PRUEBAS FUNCIONALES

VIVIANA MARCELA HERNANDEZ
ALEXANDRA KUJAR MENA

Monografía para optar al título de Especialistas en Ingeniería de Software

Asesor Temático
LILIANA GONZALEZ PALACIO
Docente Académico

Asesor Metodológico
JAIME ALBERTO ECHEVERRI
Docente Académico

UNIVERSIDAD DE MEDELLÍN
FACULTAD DE INGENIERIA DE SISEMAS
MEDELLÍN
2010

TABLA DE CONTENIDO

LISTA DE IMÁGENES **PAG.**

Lista de tablas **PAG.**

Título del trabajo:	ESTUDIO COMPARATIVO DE HERRAMIENTAS PARA LA GESTION DE CASOS DE PRUEBAS FUNCIONALES
Autor (s):	Alexandra Kujar Mena
	Viviana Marcela Hernández Rincón
Título otorgado:	Especialista en Ingeniería de Software
Asesores:	
	Temático: Liliana González Palacio
	Metodológico: Jaime Alberto Echeverri
Programa:	Especialización en Ingeniería de Software
Ciudad:	Medellín
Año:	2010

RESUMEN

Los sistemas de software cada vez más incorporan más componentes y se hace necesario garantizar que estos trabajan de manera adecuada en forma individual y en conjunto, lo que aumentaría el número de pruebas requeridas y con esto la necesidad de aplicar una metodología adecuada. Sin embargo, el reto se presenta en realizar las pruebas en el tiempo adecuado y con los recursos asignados. Es por eso que el tema de automatización y documentación en el proceso de pruebas se hace tan importante.

Este trabajo pretende dar un aporte, con una clasificación de las herramientas que existen actualmente en el mercado según el tipo de prueba que se desea realizar y el nivel de automatización que brinda la herramienta, para finalmente de toda esta gama de herramientas, enfocarse en las herramientas para documentación de pruebas de software, analizando y probando algunas de estas y poder dar un juicio verídico sobre ítems como: facilidad de instalación, tipo de licenciamiento, creación de casos de pruebas, manejo de reportes, entre otras. Lo anterior nos da la posibilidad de sugerirle a la empresa Choucair el tipo de herramienta que mejor se adapta a sus necesidades y como suplir las falencias que se tienen cuando se documentan los casos de prueba, resultados y reportes de las pruebas realizadas. Así pues la empresa Choucair, tendrá la posibilidad de estudiar las recomendaciones dadas y tomar la decisión más acertada, de acuerdo a su criterio y basándose en los resultados que arroje este trabajo.

INTRODUCCIÓN

Prueba del software es el proceso usado para determinar calidad de software. La prueba del software es una investigación técnica empírica que conduce y provee a los involucrados con información sobre la calidad del producto o del servicio bajo prueba (Kaner, 2006).

Las pruebas del software se remontan a 1979, cuando Myers, Glenford. Introdujo la separación de errores de la prueba. En ese momento él ilustró el deseo de la comunidad del software de separar actividades fundamentales del desarrollo, en otras actividades tales como la eliminación de errores de la verificación.

El proceso de pruebas de software tiene vinculadas limitaciones tales como: los factores económicos, el recurso humano, el tiempo y la complejidad asociada a los productos de software, las cuales han sido objeto de amplios estudios con el fin de mitigar o eliminar sus consecuencias.

Este trabajo tiene como finalidad realizar un estudio comparativo de las herramientas para la gestión de pruebas funcionales. Para ello fue necesario investigar sobre las herramientas de pruebas y de acuerdo a los resultados obtenidos, se propusieron dos grandes categorías: las herramientas para automatización y para la administración de las pruebas, lo cual contempla la gran mayoría de herramientas disponibles que hay en el mercado. En este caso en particular, se enfatizará en la documentación de pruebas, más específicamente de pruebas funcionales, debido a que con esta investigación se pretende dar apoyo al proceso que sigue una empresa muy importante en nuestro medio, en cuanto a la operación y gestión de pruebas de software. Después de elegidas las posibles herramientas que apoyen la documentación de los casos de prueba funcionales, se realizó un análisis desde la instalación, implementación y reportes. Todo lo anterior llevó a sugerir la herramienta más

adecuada para la documentación de pruebas funcionales en la Empresa Choucair Testing, de acuerdo a sus necesidades y expectativas.

Esta investigación está elaborada por capítulos: primero la descripción del problema, los objetivos, justificación, alcance y limitaciones. Lo cual determina el propósito y porque la escogencia del tema de pruebas. En el segundo, el marco teórico que proporcionará un conocimiento profundo de la teoría de las pruebas y sus antecedentes. Además de una revisión de la literatura. En el tercero, Solución de la propuesta. El cuarto conclusiones y quinto referencias bibliográficas.

1. INFORMACIÓN GENERAL

1.1 DESCRIPCIÓN DEL PROBLEMA

El Software testing o como se conoce en español, las pruebas de software son una etapa más del proceso de desarrollo de software y su objetivo es asegurar que la aplicación construida cumpla con las especificaciones requeridas y eliminar los posibles defectos que tenga previa entrega al cliente o usuario final (Javier J. Gutiérrez, 2007)

.

Existen diversos niveles de prueba a aplicar durante todo el proceso de desarrollo, entre ellos: pruebas de unidad, de integración, de sistema, de implantación, de aceptación.

En el nivel de sistema se realizan pruebas funcionales y no funcionales, y para las primeras es preciso diseñar casos de prueba que indiquen los aspectos a chequear en el software y los pasos que debe seguir el tester encargado. El proceso de diseño y ejecución de casos de prueba debe estar muy bien documentado y constituir un proceso sistemático en el que se facilite la comunicación formal entre diferentes actores como el analista de pruebas (encargado de diseñar casos de prueba y analizar resultados), el tester (llamado a ejecutar casos de prueba y reportar errores), y el desarrollador (responsable de solucionar los errores encontrados y reportarlo al tester), esto para evitar la pérdida de información, el aumento del tiempo dedicado a la documentación y la omisión de errores graves en el software, para lograr que el cliente tenga un producto estable, de calidad y que satisface sus necesidades.

En el caso de Choucair Testing, empresa dedicada a realizar pruebas de software, este proceso se hace más pesado y ocupa mayor tiempo porque no se utilizan herramientas que automaticen la comunicación entre los diferentes actores, por lo tanto, cuando el analista de pruebas tiene diseñados los casos de prueba de una aplicación, debe construir un documento adicional para registrar la asignación de

éstos a los diferentes tester y usar elementos como el correo electrónico o comunicación informal para reportar dicha asignación. Por otro lado y posterior a la ejecución, el tester debe llevar un archivo aparte para indicar los resultados, y asignar la solución de errores a los diferentes desarrolladores que participaron en la construcción de la aplicación, usando igualmente para esta tarea el correo electrónico. De esta manera el proceso se hace tedioso, con el riesgo de olvidar la actualización de acciones realizadas, generando consecuencias como la pérdida de información y la repetición de trabajo ya realizado.

En este proyecto de investigación, que forma parte del Proyecto HERRAMIENTA PARA LA DOCUMENTACIÓN DE PRUEBAS FUNCIONALES, que impacta el área de software testing, presente dentro de la ingeniería de software, se pretende adaptar una herramienta informática de apoyo a la documentación de pruebas funcionales para la empresa Choucair Testing.

Para abordar el problema planteado, inicialmente se realizará un estudio de las herramientas informáticas existentes para la gestión de casos de pruebas funcionales, y luego, sobre esta base, efectuar el análisis de la herramienta para pruebas funcionales a adaptar, diseñar, y codificar.

Para el alcance de este proyecto HERRAMIENTAS PARA LA DOCUMENTACION DE PRUEBAS FUNCIONALES interesa conocer que las pruebas funcionales están orientadas a verificar si la aplicación en desarrollo satisface los requisitos funcionales establecidos (Javier J. Gutiérrez), y para realizar esta actividad, posterior a elaborar un plan de pruebas, se hace el diseño de casos de prueba, para luego ejecutarlos y documentar los resultados. De lo anterior se generan dos artefactos importantes: un documento con los casos de prueba que deben ser ejecutados, y un documento que presenta resultados de la ejecución con los errores que se generaron.
Dichos artefactos pueden ser construidos con el apoyo de herramientas como las que se mencionan a continuación:

- T-Plan Professional
- Testlink
- RTH
- QaTraq
- TestDirector
- Case Maker
- Qualify 2.4

1.2 OBJETIVOS

1.2.1 Objetivo general

- Realizar un estudio comparativo de herramientas para la gestión de casos de pruebas funcionales.

1.2.2 Objetivos específicos

- Elaborar un estado del arte de la documentación que existe de herramientas para la gestión de pruebas funcionales.

- Realizar un análisis comparativo de las herramientas seleccionadas para la agilización en la implementación y las ventajas que ofrece la herramienta para las pruebas funcionales.
- Selección de la herramienta más apropiada de acuerdo a las necesidades de la empresa.

1.3 JUSTIFICACIÓN

El proyecto contribuye al desarrollo del área de pruebas de software, enmarcada en el cuerpo de conocimiento de ingeniería de software, aportando un estudio de las herramientas de soporte a pruebas funcionales más conocidas en el medio y

que ha sido posible la consecución de una versión de prueba, continuando con la construcción o adaptación y validación de una herramienta que facilite el trabajo de los analistas de pruebas funcionales, tomando como referente el trabajo desarrollado por Choucair Testing.

Adicionalmente, el proyecto estrecha el vínculo entre empresa y universidad, necesario para formar profesionales que aporten soluciones a problemas reales de la industria.

Este puede ser el inicio de una línea de proyectos conjuntos que potencian la investigación aplicada. Por otro lado, el proyecto contribuye a la formación de estudiantes en investigación como punto crucial en los planes de relevo generacional del programa y de la facultad

Usuarios Directos e Indirectos potenciales de los resultados de investigación:

- Empresa Choucair testing.
- Comunidad científica.
- Comunidad estudiantil y profesional de la ingenieria de software

1.4 ALCANCE Y LIMITACIONES

El alcance de esta investigación será la proposición de la herramienta más adecuada según las necesidades más relevantes de la empresa Choucair Testing.

Estas limitaciones están contempladas en el trabajo de grado (VELÁSQUEZ, 2009):

Algunas de las limitaciones existentes a nivel de pruebas de software son:

- No existe una recopilación formal que especifique y describa los tipos de prueba que se deben aplicar a una pieza de software y se detalle la implementación precisa de cada uno de ellos. Aunque existen algunos documentos, en su mayoría son de carácter muy académico y poco práctico. Por otro lado, y aunque ISO plantea en normas como la ISO/FDIS 9126 ciertos aspectos en los cuales el producto de software debe ser conforme, no es suficientemente especifico como para relacionarlo con una prueba de software con pormenores de ejecución, llevando esto a que a que dicho estándar por si solo sea poco práctico. Las técnicas de pruebas generalmente son subestimadas como útiles, en tanto que no se usan en la práctica para el diseño de casos de prueba formales. En general y en entornos productivos, las técnicas utilizadas para la detección de errores son del tipo suposición de errores y en el mejor de los casos de tipo aleatorio el cual por sus características propias no es un método que aporte mucho a la detección de errores en el producto de software; todo esto se une a que habitualmente cuando se construyen pruebas de software (casos de prueba) no se formaliza el uso de técnicas específicas en tanto que no es considerado importante o no se cuenta con el conocimiento y entrenamiento suficiente para su uso.

- Los estándares existentes que se refieren a pruebas de software y aseguramiento de la calidad del producto tanto en ISO como en la IEEE tienen un acceso muy restringido, siendo difícil lograr adquirir la documentación relacionada. Adicionalmente, y basados en que dichos estándares en la mayoría de los casos son muy específicos y aplicarlos sin un modelo de calidad y desarrollo es sumamente difícil, modelos habituales como CMMI e ISO 9000 no son tomados como referencia en la mayoría de los casos para describir procedimientos o prácticas propias.

- La implementación de prácticas como verificación, validación y aseguramiento de la calidad contempladas en CMMI y en normativas específicas de la IEEE y la implantación de modelos específicos de pruebas de software, suele ser

sumamente difícil de implantar debido a que requiere gran inversión por parte de de las organizaciones en recursos como: personal capacitado, conocimiento y orientación experta, esfuerzo adicional, tiempo, dinero, compromiso organizacional, entre otras (Jokella).

Adicionalmente por ser modelos de calidad orientados a la madurez de los procesos y los productos, los periodos de tiempo necesarios para que la madurez llegue a un nivel deseable son relativamente largos, lo cual conlleva a que aunque se logren avances en la calidad del los productos paulatinamente se logra la madurez, solo se empieza a tener un esfuerzo más proporcional al beneficio cuando todo el proceso es idealmente maduro y la organización logra converger hacia las prácticas que plantea el modelo de calidad general implantado.

- Los estándares diseñados por la ISO específicamente el ISO/FDIS 9126, ISO 14598 e ISO/IEC 15504 presentan un nivel de complejidad variable al momento de ser usados de acuerdo a la organización en donde se desee implantar, sin embargo, una característica anexa, es que aunque es posible su establecimiento en una organización que no cuenta de antemano con un modelo de calidad ya preestablecido, lo recomendable y casi requisito no especificado, es que si lo exista, ya que dichos estándares exigen elementos como proceso definidos, políticas de aseguramiento de la calidad claras, y prácticas que lleven a la madurez entre otras para cumplir sus propios objetivos. Lo más común es tener modelos de calidad y de mejoramiento continuo como ISO 9000 y CMMI con el fin de que la aplicación de estándares de aseguramiento de calidad exigentes como los mencionados tengan elementos suficientes como para que su adhesión a los procesos organizacionales sea exitosa.

- La razón por la cual la industria no adopta metodologías y estándares es la dificultad de la utilización de estos para pruebas de software. En las pequeñas

y medianas empresas la dificultad radica en la necesidad que estas tienen de competir mediante tiempos y precios y dada la poco practicidad que los estándares tienen asociados se incurre en consumos de recursos poco tolerables; por otra parte las grandes compañías además de los factores anteriores, compiten mediante calidad y para lograr un buen indicador deben implementar los diferentes metodologías y estándares que garanticen la madurez de los procesos, para estas compañías la falta de practicidad de los estándares en más un requisito de sus clientes que un riesgo asociado al uso de recursos.

- Implementar un proceso de pruebas al interior de una compañía es costoso, además el proceso debe madurar para que se adapte a las necesidades específicas de la organización; este hecho obliga a muchas empresas a subcontratar los procesos de pruebas. En ocasiones esta es una buena opción puesto que evita posibles sesgos en las pruebas pero en general puede conllevar serios problemas al interior de la organización que van (IEEE - 0730, 2002, Standard for software quality assurance plans) desde la confidencialidad de la información hasta el incremento de los tiempos de desarrollo.

- Las pruebas de software más complejas basan su diseño y su ejecución en los artefactos desarrollados en etapas de análisis y diseño; si estos artefactos no son construidos o tienen deficiencias como falta de consistencia y coherencia; aparte de afectar todo el proceso de desarrollo se verá comprometido el aseguramiento de la calidad y dentro de esté las pruebas los tipos de pruebas de software que buscan hallar defectos más profundos. La buena ingeniería de software se convierte en una necesidad básica para el correcto desarrollo de las pruebas de software.

- En general diversos son los problemas que se encuentran en un proceso de pruebas; algunos de ellos son:

- Los probadores desconocen el dominio o los tipos y técnicas a utilizar cuando se requieren pruebas de alto nivel, en general este problema se da al tener personal poco capacitado para la labor.

- Los proveedores de pruebas generalmente posponen la labor de pruebas a las etapas finales del ciclo de vida del software, lo que ocasiona problemas por retrasos. Estos retrasos se dan generalmente por que el equipo de pruebas debe adquirir el conocimiento del dominio necesario y diseñar las pruebas, tareas que se pueden realizar de forma transversal al proceso de desarrollo optimizando los recursos disponibles.

- En ocasiones los probadores no cuentan con los insumos necesarios para realizar un proceso de pruebas maduro y completo. Estos insumos corresponden a una buena documentación y a un completo y correcto levantamiento de requisitos, que orienten al probador, tanto en la ejecución y desarrollo de las pruebas como en el aprendizaje del dominio.

- Muchas organizaciones contemplan su proceso de QA basado en las pruebas de software lo cual trae perjuicios en la calidad de los productos al no incluir procesos como medición, análisis, verificación y validación, todos ellos componentes esenciales de QA.

2. MARCO DE REFERENCIA

2.1 MARCO TEORICO

Este proyecto tiene repercusión en el área software testing, que hace parte de la Ingeniería de Software. Procedemos a dar algunas definiciones y términos relevantes.

- Significado y origen de la “Ingeniería del Software”, existen diferentes versiones, sin embargo, en la mayoría de los casos su origen es atribuido a dos congresos organizados por la OTAN en 1967 y 1968 para tratar la llamada “crisis del software”, concepto nacido de la situación que se presentaba con los grandes programas que no llegaban nunca a poderse entregar, además de ser altamente ineficientes, tener un gran número de errores y llevar asociados unos costos impredecibles. Los problemas que se presentan en la construcción de grandes sistemas no son simples versiones a gran escala de los problemas de escribir pequeños programas de computador (F. Javier Zarazaga Soria, 2003). Surge entonces la necesidad de darle seriedad a la construcción de software, y se crea la ingeniería de software (F. Javier Zarazaga Soria, 2003), para la cual se han adoptado diversas definiciones:

- (F.L.Bauer, 1972) indica que la Ingeniería del Software es el establecimiento y uso de sólidos principios de ingeniería y buenas prácticas de gestión, así como la evolución de herramientas y métodos aplicables y su uso cuando sea apropiado para obtener, dentro de las limitaciones de recursos existentes, software que sea de alta calidad en un sentido explícitamente definido.

- (Boehm, 1976) presenta la Ingeniería del Software como la aplicación práctica del conocimiento científico en el diseño y construcción de programas de

computadora y la documentación asociada requerida para desarrollarlos, operarlos y mantenerlos.

- Otra definición es proporcionada por Zelkovitz y otros (M.V.Zelkovitz, 1979), quienes describen la Ingeniería del Software como el estudio de los principios y metodologías para desarrollo y mantenimiento de sistemas de software.

- En la colección de estándares publicados por la IEEE en 1990 (IEEE Std 610.12-1990 (R2002), IEEE Standard Glossary of Software Engineering Terminology, 1990) se define la Ingeniería como la aplicación de un método sistemático, estructurado y cuantificable a estructuras, máquinas, productos, sistemas o procesos; y la Ingeniería del Software como la aplicación de un método sistemático, estructurado y cuantificable al desarrollo, operación y mantenimiento de software.

  Son múltiples las definiciones de Ingeniería del Software, pero la bibliografía existente coincide a la hora de proponer las grandes Áreas de Conocimiento que la componen, y una prueba de esto es la estructura propuesta por la IEEE, en su documento “SWEBOK: Software Engineering Body of Knowledge” (IEEE Computer Society, 2004), que se indica gráficamente en la Figura 1.

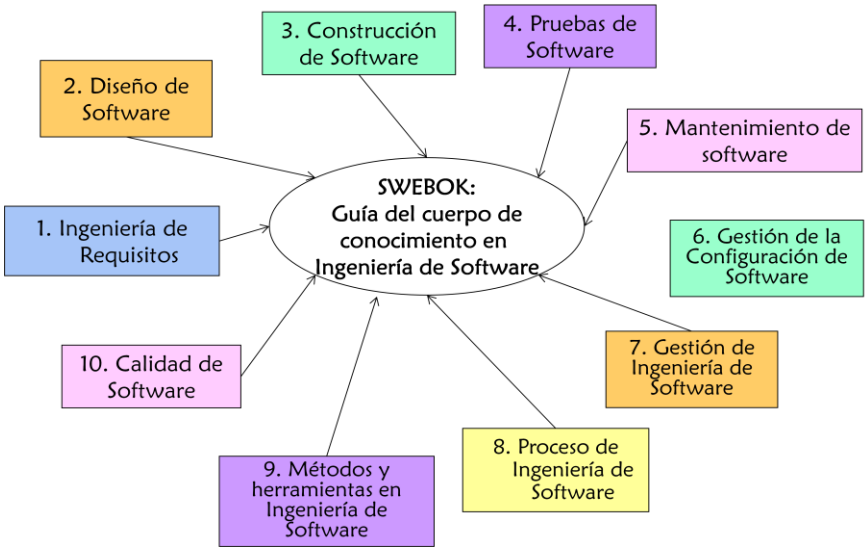


Figura 2.1 Áreas de conocimiento que componen el cuerpo de conocimiento en ingeniería de software, según lo propuesto en el swebok. Fuente: (IEEE Computer Society, 2004).

El software testing o pruebas de software es una de las áreas que conforma la ingeniería de software, y es la que compete en este proyecto.

Existen diversas definiciones de software testing. Myers lo define como el proceso de ejecutar un aplicativo o sistema con la intención de encontrar errores (G., 2004). Por su parte, Craig lo refiere como el proceso de ingeniería concurrente al ciclo de vida que busca utilizar y mantener el testware, o artefactos de pruebas, para medir y mejorar la calidad del software que está siendo probado (Craig R, 2002).

Debido al aumento en el tamaño y complejidad del software, y a los requerimientos de calidad cada vez más exigentes que hacen los clientes, el proceso de prueba debe ser sistemático y organizado. Diversas pruebas deben ser hechas a un producto software, y una clasificación aceptada es la propuesta en la metodología Métrica V3, y referenciada por diversos autores (Javier J. Gutiérrez) (Ver tabla 2.1):

Tabla 2.1 Tipos de prueba de software

Tipo de pruebas	Descripción
Pruebas Unitarias.	Prueban el diseño y el comportamiento de cada uno de los componentes del sistema una vez construidos.
Pruebas de Integración.	Comprueban la correcta unión de los componentes del sistema entre sí a través de sus interfaces, y si cumplen con la funcionalidad establecida.
Pruebas de Sistema.	Prueban a fondo el sistema, comprobando su funcionalidad e integridad globalmente, en un entorno lo más parecido posible al entorno final de producción.
Pruebas de Implantación.	Comprueba el correcto funcionamiento del sistema dentro del entorno real de producción.
Pruebas de Aceptación.	Verifican que el sistema cumple con todos los requisitos indicados y permite que los usuarios del sistema den el visto bueno definitivo.
Pruebas de Regresión.	El objetivo es comprobar que los cambios sobre un componente del sistema, no generan errores adicionales en otros componentes no modificados.

Dentro de las pruebas de sistema se encuentran las pruebas funcionales y no funcionales. Para el alcance de este proyecto interesa conocer que las pruebas funcionales están orientadas a verificar si la aplicación en desarrollo satisface los requisitos funcionales establecidos (Javier J. Gutiérrez).

Las pruebas de software son transversales a todo el ciclo de vida del desarrollo de software, pasando por requerimientos, análisis y diseño, programación, puesta en marcha y mantenimiento.

Existen distintos procesos y metodologías que permiten hacer gestión a las pruebas de software, la mayoría de ellos distingue las fases de planificación, diseño y de ejecución de las pruebas:

- Durante la planificación de las pruebas se decide qué se probará y con qué profundidad.
- En la fase de diseño de las pruebas, la especificación se analiza para derivar los casos de prueba que son un conjunto de valores de entrada, precondiciones de ejecución, resultados esperados y pos condiciones de ejecución, desarrollados con un objetivo particular o condiciones de prueba, tal como ejercitar un camino de un programa particular o para verificar que se cumple un requerimiento especifico (IEEE, 1990).
- En la fase de ejecución es donde se ejecutan los casos de prueba diseñados previamente, se compara el resultado real con el esperado y se reportan los resultados. Para gestionar todo el ciclo de pruebas y tener bajo control estos artefactos existen herramientas de gestión de pruebas.

Estas fases del proceso de pruebas se evidencian en la figura 2.2:

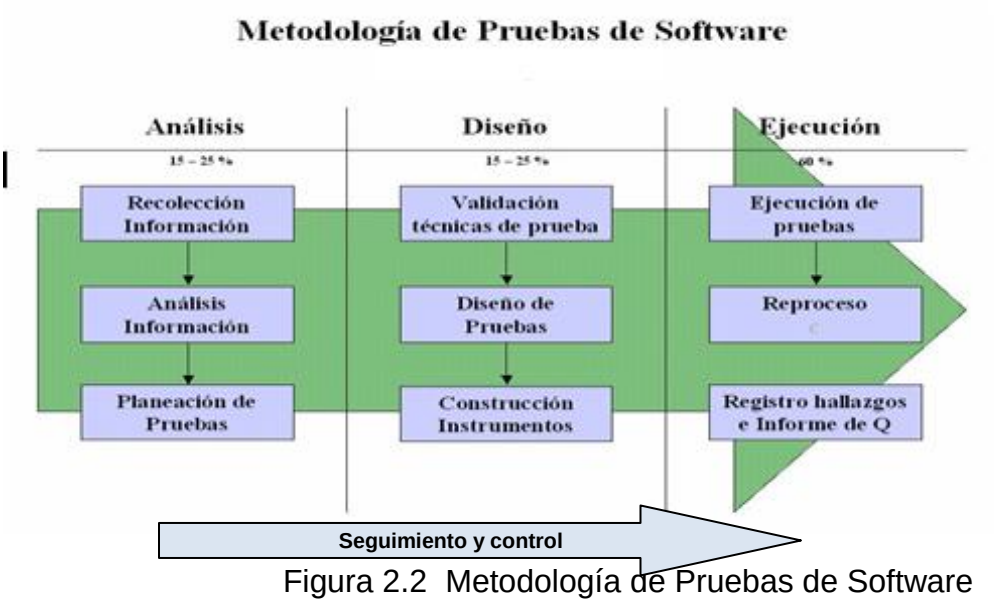


Figura 2.2 Metodología de Pruebas de Software

En cada una de las fases descritas en la figura anterior intervienen una serie de pasos o actividades propias de cada fase y otra que es transversal a todas las fases como el seguimiento y control, las cuales se describen a continuación:

a. Fase de análisis.

Esta fase inicia con *la* ***recolección de la Información*** que implica la consideración de los requisitos o especificaciones que había manifestado el cliente con base en la documentación del proyecto y documentación del software a probar.

Luego se hace un ***análisis de Información*** recolectada como:

- Requerimientos y alcance del producto software.
- Información del alcance Técnico.
- Identificación del modelo de desarrollo Software.
- Tipos de pruebas a realizar.
- Estimación de esfuerzos.
- Definición del esquema de trabajo.
- Cronograma de actividades.

Finalmente en esta fase se hace la ***planeación de las pruebas*** durante la cual se definen actividades y documentos como:

- Ficha Técnica del Producto
- Plan de Trabajo
- Cronograma de Pruebas
- Descomposición del Producto
- Inicio de Banco Preguntas Frecuente

Manteniendo durante esta fase un ***seguimiento y control*** mediante:

- Definición de indicadores para medir.
- Definición de mecanismos de seguimiento.

- Informe Avance Proceso Pruebas.

b. Fase de diseño de pruebas

Una vez finalizada la fase de análisis se procede a realizar una ***Validación de las Técnicas de Pruebas*** mediante la cual se identifica las técnicas de pruebas que son más apropiadas para cada uno de los tipos de pruebas definidos en el plan de pruebas.

Durante el ***diseño Pruebas*** se hace:

- Una identificación de los requerimientos de Pruebas.
- Definición de la estrategia de Pruebas.
- Construcción de Instrumentos de Pruebas: MRP – Scripts

Manteniendo un ***seguimiento y control*** mediante:

- Validación de MRP – Equipo Desarrollo.
- Informe Avance Proceso Pruebas.

c. Fase de ejecución de pruebas

Antes de iniciar con la ejecución de las pruebas se debe hacer una preparación de ambiente de pruebas, mediante la definición de un ambiente controlado para cada uno de los tipos pruebas definidos.

Esta etapa de ejecución de pruebas se da mediante un ciclo de iteraciones en el cual se presenta:

- Ejecución de Requerimientos de Pruebas.
- Registro de hallazgos de NC.
- Reproceso de los NC.

- Ejecución de Scripts para pruebas.
- Análisis de Resultados de Pruebas e informes de calidad.

Realizando seguimiento, control y retroalimentación mediante la utilización de técnicas como:

- Calculo de Indicadores de Producto
- Informe del avance del proceso de pruebas
- Informe de cierre de cada uno de los tipos de las pruebas.

2.2 REVISIÓN DE LA LITERATURA

Inicialmente se realizará una breve descripción de las herramientas de pruebas y algunas de las clasificaciones que se le han dado a estas, y de esta manera poder hacer una propuesta de una clasificación global para las herramientas de pruebas de software, con su respectiva descripción y las herramientas más comunes para cada tipo de herramienta; de acuerdo a este estudio se podrá seleccionar un tipo de herramienta de prueba (para este caso herramientas para gestión y documentación de pruebas) y se hará un estudio comparativo de las 4 herramientas más representativas en nuestro medio, y de estas se seleccionará las más idónea para el caso de estudio en particular de la empresa Choucair Testing.

HERRAMIENTAS PARA PRUEBAS DE SOFTWARE

Las herramientas para pruebas de software ayudan a los equipos de desarrollo de software a investigar los errores de software, verificar la funcionalidad de los sistemas y asegurarse de que el software que desarrollan es seguro y confiable. Existen herramientas especiales para cada una de las etapas de un proyecto de desarrollo de software (Technology Evaluation Centers), la utilización de estas herramientas:

- Miden el desempeño del software
- Mejoran la calidad de las aplicaciones de software
- Mejoran los procesos de gestión del ciclo de vida de los productos
- Sirven para llevar a cabo análisis de riesgo y pruebas comparativas
- Ayudan a uniformizar los procedimientos de pruebas, cuando son herramientas de pruebas automatizadas (las pruebas manuales, pueden producir resultados inconstantes)
- Generan ahorros en los costos de desarrollo y mantenimiento del software.

- Mejoran los periodos de comercialización porque permiten detectar con eficacia los problemas funcionales, de desempeño y de seguridad (Technology Evaluation Centers).

2.2.1 CLASIFICACIÓN DE LAS HERRAMIENTAS

Hay herramientas que apoyan diversos aspectos de las pruebas como: herramientas para la administración de las pruebas, para seguimientos de incidentes, para la gestión de la configuración y administración de requerimientos, herramientas para el diseño de las pruebas y preparación de datos de pruebas y herramientas de ejecución de pruebas para la ejecución de casos de pruebas (Ignacio, 2010).

A las herramientas para pruebas de software les han dado diferentes clasificaciones ligadas generalmente al tipo de pruebas que soportan o a la fase de desarrollo de software en que se encuentra el producto, algunas de ellas son:

- Se han identificado 5 áreas importantes a tener en cuenta a la hora de realizar testeos (NO SOLO UNIX, 2010):

    - Herramientas de carga y rendimiento (Load and Performance Test Tools)
    - Web Functional/Regression Test Tools (Java Test Tools)
    - Validadores de HTML (HTML Validators)
    - Comprobadores de Links (Link Checkers)
    - Herramientas de comprobación de seguridad (Web Site Security Test Tools).

- En el mercado de las herramientas de prueba se ofrecen series de herramientas que se integran alrededor de un producto principal y que utilizan varias plataformas para dar soporte a todo el ciclo de vida de un proyecto, siendo las herramientas de automatización de las pruebas las

más reconocidas y probablemente las más numerosas (Lyndsay, 2005), y se presentan en la figura 2.3:

DISEÑO	CODIFICACION	EVALUACION	ADMINISTRACION DE PRUEBAS
• Herramientas de Captura y análisis de requisitos. • Herramientas de Creación de modelos visuales	• Herramientas de Pruebas estáticas • Herramientas de automatización de Pruebas unitarias	• Herramientas de Automatización de Captura y reproducción. • Herramientas de Automatización de pruebas de carga. • Herramientas de Supervisión. • Herramientas de Inyección de fallas	• Herramientas de gestión de pruebas • Herramientas de manipulación de datos • Herramientas de gestión de ambientes

Figura 2.3 Herramientas de automatización de las pruebas de acuerdo al ciclo de vida de un proyecto.

Dadas las diversas clasificaciones que se pueden encontrar para las herramientas de pruebas de software que ofrecen los autores, y que no se especifica un

parámetro de clasificación estándar, porque varían dependiendo de las fases en que se encuentra el producto, lo que se desee probar o el nivel de detalle de las pruebas entre otros, se realizará un propuesta de una nueva clasificación para las herramientas de pruebas de software, tomando como parámetro el nivel de automatización que ofrece la herramienta y la fase del ciclo de vida de desarrollo de software que soporta.

Para ello y de acuerdo a lo investigado, se propone trabajar con una clasificación para las herramientas de pruebas de software en 2 grandes grupos:

- Las herramientas que automatizan las pruebas de software en las diferentes etapas desarrollo.
- Las herramientas para la administración de pruebas que sirven de apoyo, gestión y documentación de las pruebas de software manuales durante todo el ciclo de desarrollo.

Una vez establecida esta clasificación propuesta por el equipo de investigación se hará una descripción de cada uno de estos tipos de herramientas. En la figura 2.6 se presenta la clasificación de herramientas para pruebas propuestas por el equipo.

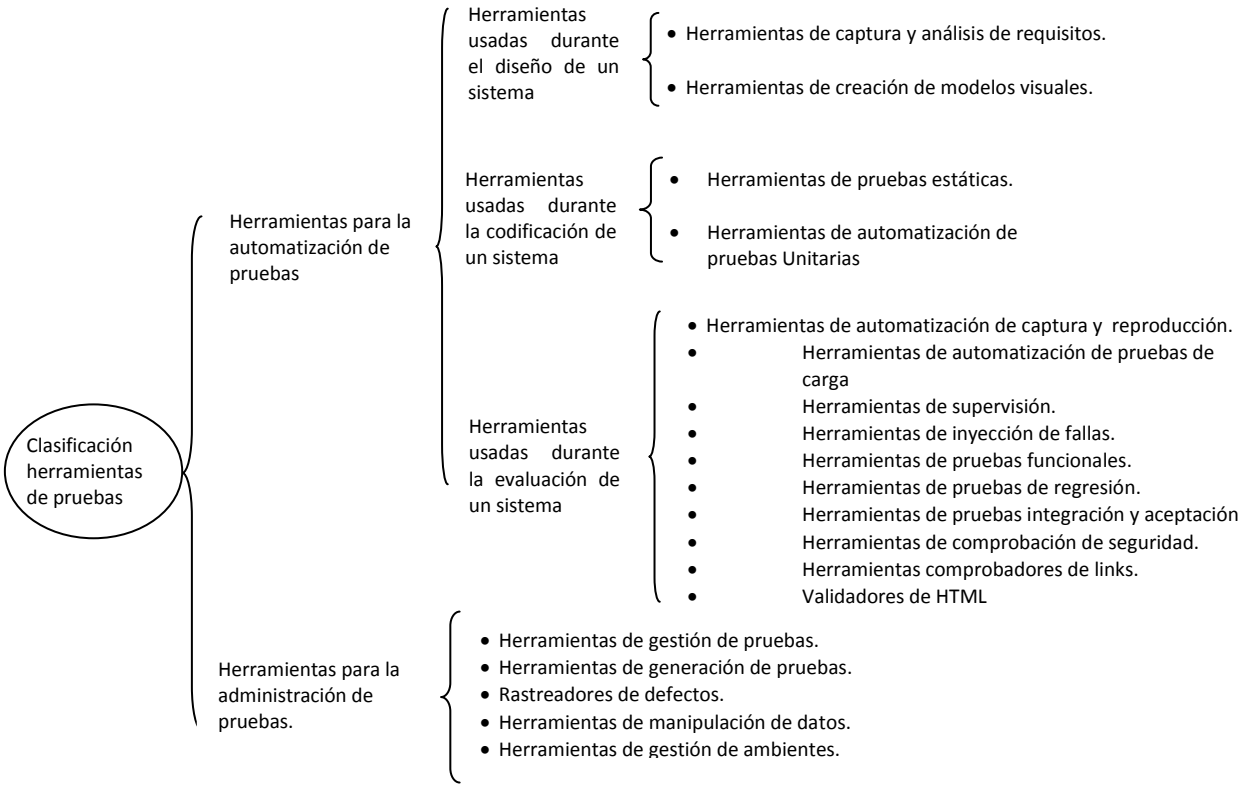


Figura 2.4 Clasificación de las herramientas pruebas propuesta por el equipo de investigación.

A continuación se hace una breve descripción de cada uno de los tipos de herramientas y algunas de las herramientas más utilizadas en cada caso de acuerdo a la clasificación propuesta por los investigadores en la figura anterior.

2.2.1.1 Herramientas para la automatización de las pruebas

Estas herramientas permiten que una máquina se encargue de realizar una prueba predefinida haciendo que el proceso de ejecución de las pruebas sea más rápido, preciso y pueda repetirse cuando se desee (Lyndsay, 2005).

Todas las herramientas de automatización de las pruebas tienen la misma base: todas utilizan guiones detallados para describir las pruebas y un conjunto

de resultados esperados. Aunque la persona que realiza las pruebas no participa en la construcción de estos guiones, implican algunas complejidades en la ingeniería del software, sobre todo al tratar de simular pruebas manuales (particularmente al trabajar con interfaces point-and-click).

Para resolver este problema, la mayoría de las herramientas incorporan funcionalidades, desde proporcionar un ambiente completo de desarrollo hasta reconocer los objetos en una GUI. Algunas herramientas permiten que los evaluadores más experimentados impulsen las capacidades de la herramienta mediante técnicas de scripting sofisticadas. Sin embargo, cuando se usan simplemente para imitar las pruebas manuales, las herramientas de automatización de las pruebas pueden ayudar a confirmar el valor existente sin resaltar los riesgos nuevos.

Las pruebas automatizadas pueden producir cantidades enormes de información no refinada, y existen muchas herramientas que proporcionan funcionalidades para procesarla y volverla más aceptable (Lyndsay, 2005), estas herramientas se utilizan durante el diseño, codificación y evaluación de un producto software.

Existen herramientas de automatización de pruebas específicas para las diferentes fases del ciclo de vida de desarrollo del software, como para la fase de diseño, fase de codificación y fase de evaluación, las cuales se especifican a continuación:

2.2.1.1.1 Las herramientas que se usan durante el diseño de un sistema

Las herramientas de diseño permiten capturar y compartir información fundamental acerca de los planes de un sistema, además generalmente se utilizan durante las primeras etapas de un proyecto. Pensar en realizar pruebas al principio de un proyecto puede convertirse en dividendos más adelante, y existen muchas herramientas de diseño que están hechas también

para realizar pruebas. A continuación se describe cada una de los tipos de herramientas usadas durante esta fase:

a. **Las herramientas de captura y análisis de los requisitos**: Permiten compartirlos con todo el equipo y controlarlos a lo largo de un proyecto mediante la creación de un depósito central y un archivo. Con frecuencia permiten ligar los requisitos a la documentación, el código y las pruebas. Estas herramientas de gestión de los requisitos pueden ser autónomas o parte de un sistema más grande de gestión de documentos. Las herramientas complejas pueden incluir análisis sintáctico y validación sofisticados de los lenguajes naturales, mientras que las más simples se parecen a los gerentes de listas.
b. **Las herramientas de creación de modelos visuales**: permiten que los arquitectos y los diseñadores de sistemas construyan modelos del sistema y los visualicen en varias formas. Permiten también usar diagramas formales para crear modelos que se pueden convertir en estructuras de código, pruebas y esquemas de datos.

2.2.1.1.2 Las herramientas que se usan durante la codificación de un sistema

Las herramientas de pruebas que se usan durante la codificación de un sistema en generalmente están ligadas estrechamente a la estructura del código. Se les conoce como herramientas “white-box” o “clear-box” (como referencia a su necesidad de tener visibilidad del funcionamiento del sistema). Existen muchas herramientas que pueden integrarse con el ambiente de construcción y funcionar automáticamente detrás del sistema principal. Estas herramientas trabajan con partes del código en lugar de trabajar con los sistemas integrados y configurados, y tienden a ser específicas para un lenguaje o una tecnología en particular (Lyndsay, 2005). Los tipos de herramientas usadas durante esta fase son:

a. **Las herramientas de pruebas estáticas:** examinan el código escrito, pero no lo utilizan. Son una forma eficaz para detectar una amplia gama de fallas que serían difíciles de diagnosticar o reproducir de forma confiable en etapas posteriores. Las pruebas tienen base en las normas del código y en las fallas esperadas y no tienen un enlace directo a la funcionalidad o los requisitos del sistema, además de que deberán ser ajustadas para adaptarse a las prácticas de trabajo. Algunas herramientas de pruebas estáticas también tienen funcionalidades que ayudan a que los desarrolladores visualicen la estructura del código, midan su complejidad y muestren la forma en que los programas y los datos están entrelazados (Lyndsay, 2005). Las herramientas más comunes son:

- PHPLint: permite mejorar las tareas de programación, ya sea comenzando la codificación con esta herramienta o mejorando código ya existente. Permiten dar seguridad en el código, errores de sintaxis, variables no utilizadas, código muerto, etc.

- PMD: puede ser integrado a varias herramientas: JDeveloper, Eclipse, JEdit, etc. Permite encontrar en el código errores en el manejo de excepciones, código muerto, código sin optimizar, código duplicado, en desarrollos bajo el lenguaje Java.

- YASCA: permite encontrar vulnerabilidades de seguridad, calidad en el código, rendimiento. Aprovecha la funcionalidad de los plugins FindBugs, PMD y Jlint. Utilizada para lenguajes Java, .Net, PHP, HTML.

b. **Las herramientas de automatización de pruebas unitarias:** permiten evaluar de forma aislada un componente del sistema antes de que forme parte del sistema general. Normalmente, estas herramientas permiten

realizar una serie de pruebas unitarias por demanda y son parte de técnicas ágiles, como las técnicas de desarrollo controladas por las pruebas. Además de administrar la ejecución automatizada de las pruebas y resumir los resultados de las mismas, pueden supervisar el uso que hace el componente de los recursos o hasta las líneas específicas de código que ejercita cierta prueba. Algunas son capaces de ayudar en las pruebas aisladas, ya que simulan otros recursos que podrían ser necesarios para el código (Lyndsay, 2005).

Las pruebas unitarias aseguran que un único componente de la aplicación produce una salida correcta para una determinada entrada. Este tipo de pruebas validan la forma en la que las funciones y métodos trabajan en cada caso particular. Las pruebas unitarias se encargan de un único caso cada vez, lo que significa que un único método puede necesitar varias pruebas unitarias si su funcionamiento varía en función del contexto. Las cuales deben ser:

- Automatizable: no debería requerirse una intervención manual. Esto es especialmente útil para integración continúa.
- Completas: deben cubrir la mayor cantidad de código.
- Repetibles o Reutilizables: no se deben crear pruebas que sólo puedan ser ejecutadas una sola vez. También es útil para integración continua.
- Independientes: la ejecución de una prueba no debe afectar a la ejecución de otra.
- Profesionales: las pruebas deben ser consideradas igual que el código, con la misma profesionalidad, documentación, etc.

Las herramientas de automatización de pruebas unitarias más comunes son:

- PHPUnit: es una librería que puede ser usada en la fase de pruebas de aplicaciones PHP. Está hecho para correr pruebas y analizar resultados de manera sencilla. PHPUnit es una migración del popular JUnit utilizado en desarrollos Java, integra casos de prueba basadas en la anotación @grouping, soporta objetos Mock, pruebas de bases de datos, cálculo de métricas de software, documentación ágil, entre muchas otras características (SG VirtualConference 2010).
- JUnit: es un conjunto de librerías (framework) utilizado para la programación de pruebas unitarias de aplicaciones Java. JUnit permite realizar la prueba de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera.

  Características

  - Facilita creación de pruebas unitarias.
  - Posee visualización de los resultados en modo gráfico.
  - Existen integraciones para los entornos de desarrollo NetBeans y Eclipse, los cuales incluyen generadores de plantillas (Osvaldo Barrios, 2010).

- NUnit: es una herramienta utilizada para escribir y ejecutar pruebas en .NET, es un framework desarrollado en C# que ofrece las funcionalidades necesarias para implementar pruebas en un proyecto. Además provee una interfaz grafica para ejecutar y administrar las mismas.

  NUnit basa en la utilización de atributos personalizados. Estos atributos le indican al framework de NUnit que debe como interpretar y ejecutar las pruebas implementadas en el método o clase.

Se usan aserciones, métodos del framework de NUnit utilizados para comprobar y comparar valores (N unit 2010).

2.2.1.1.3 Las herramientas que se usan durante la evaluación de un sistema

Una vez que se detectaron los errores de código, las pruebas adoptan una visión del sistema que es más amplia y enfocada al negocio. La información técnica acerca del interior del sistema puede ser menos importante que la comprensión de los riesgos y los requisitos del negocio. Para estas pruebas, normalmente se usan herramientas que no necesitan tener mucha información acerca de la estructura interna del sistema. Durante esta fase se utilizan herramientas como:

a. **Las herramientas de automatización de captura y reproducción**: crean guiones iniciales al grabar las acciones de una sola persona normalmente mientras utiliza una interfaz gráfica. Estos guiones pueden ser modificados y reproducidos cuando se quiera, analizando y resaltando las desviaciones de las acciones esperadas o sus consecuencias (que han sido grabadas) cuando sea necesario.

b. **Las herramientas de automatización de pruebas de carga**: (conocidas también como herramientas de pruebas de volumen o de estrés) es el proceso de poner la demanda en un sistema o dispositivo y medir su respuesta, reproducen simultáneamente las acciones de varios usuarios aunque pueden necesitar un hardware dedicado para funcionar correctamente y pueden evitar por completo la interfaz con el usuario para dar lugar a una interacción más profunda. Con frecuencia ofrecen una funcionalidad que permite supervisar el comportamiento del sistema como un todo y no una prueba individual cualquiera.

   Cuando la carga colocada en el sistema se eleva más allá de los patrones de uso normal, para poner a prueba la respuesta del sistema a los

habituales o los picos de demanda, se le conoce como pruebas de estrés. La carga suele ser tan grande que las condiciones de error son el resultado esperado, aunque no existe una frontera clara cuando una actividad deja de ser una prueba de carga y se convierte en una prueba de esfuerzo.

Hay poco acuerdo sobre cuáles son los objetivos específicos de las pruebas de carga. El término se utiliza a menudo como sinónimo de software de pruebas de rendimiento, pruebas de fiabilidad, y volumen de las pruebas.

Las herramientas más comunes para automatización de pruebas de carga son:

- JMeter: es una herramienta Open Source realizada en java (es un proyecto de Apache) que se utiliza para realizar tests de rendimiento, normalmente contra aplicaciones web. Jmeter nos permite realizar simulaciones de gran carga en el servidor, red o aplicación para comprobar su “fuerza” y para analizar el rendimiento ante diferentes tipos de sobrecarga.

- JCrawler: Aplicación Opensource para realizar test de estrés a aplicaciones web. Le pasas una URL y puedes realizar una navegación. Admite redirecciones HTTP y cookies. Es independiente de la plataforma, posee un modo consola y es sencillo de configurar.

- SOLEX: Es una herramienta Open Source para testeo creado para ser implantado como un plugin en Eclipse. Solex permite grabar la sesión de un usuario, permitiéndolo configurar en función a diferentes parámetros, para poder ser utilizado posteriormente y ser repetida, de manera que

aseguremos la no regresión de la aplicación. Asimismo, también nos permite realizar pruebas de estrés y rendimiento contra la aplicación web.

c. **Las herramientas de supervisión:** (conocidas también como registradores de vuelo) también miden las propiedades generales del sistema y su ambiente durante las pruebas, y permiten realizar una comparación de la historia de las medidas que toman. Pueden dar a un sistema información de afinación y notificaciones tempranas acerca de los problemas sistemáticos inesperados, por eso, se usan con frecuencia en las pruebas unitarias y en el funcionamiento en vivo.

d. **Las herramientas de inyección de fallas:** simulan problemas en el ambiente del sistema. Permiten realizar pruebas que, de otro modo, serían difíciles de reproducir y pueden dar una ruta rápida y efectiva para descubrir las debilidades que normalmente no serían evidentes en un ambiente de pruebas no equipado (especialmente las fallas que podrían ser el resultado de un uso malintencionado). Algunas de estas herramientas son:

- Exhaustif/SWIFI: es una herramienta de inyección de fallos, efectiva en costes, que sirve para mejorar la fiabilidad de sistemas intensivos en software. La herramienta puede emplearse durante las fases de integración o pruebas de sistema de cualquier ciclo de vida de desarrollo Software. (10Ju).

- Holodeck: una herramienta de testeo de seguridad que le permite analizar la interacción de la aplicación con su entorno, y la fuerza para manejar condiciones de error que puede conducir al fracaso o incumplimiento. Se trata de un filemon, Regmon, netmon, processmon,

libmon y apimon todo en una interfaz de usuario, para que pueda ámbito de aplicación de proceso o procesos y subprocesos.

Para las pruebas de seguridad, Holodeck expone la superficie de ataque de las aplicaciones y la utilización de sofisticadas técnicas de fuzzing y bloqueo, le permite analizar y bloquear las áreas en donde su aplicación es vulnerable a explotar. Version 2.8 includes support for Windows Vista and .NET Framework 3.5. Versión 2.8 incluye soporte para Windows Vista y. NET Framework 3.5. (SecurityInnovation).

e. Las herramientas de pruebas funcionales: Las pruebas funcionales están desarrolladas bajo la perspectiva del usuario, confirmando que el sistema hace lo que los usuarios esperan que haga. El objetivo de la prueba funcional es validar cuando el comportamiento observado del software probado cumple o no con sus especificaciones. La prueba funcional toma el punto de vista del usuario. Las funciones son probadas ingresando las entradas y examinando las salidas. La estructura interna del programa raramente es considerada. Las Herramientas más comunes son:

- Selenium: HQ es un framework que permite automatizar las pruebas funcionales sobre aplicaciones web. Está formado por varias herramientas que permiten definir estas pruebas y automatizarlas. En concreto, vamos a ver:

  Selenium IDE: Se trata de un plugin de Firefox que permite definir una prueba de una aplicación web guardando en un script las acciones que ejecuta el usuario sobre el navegador. Este script puede volver a ejecutarse después todas las veces que se quiera.

Selenium RC (Remote Control): Permite automatizar las pruebas sobre las aplicaciones web a partir de los scripts definidos mediante Selenium IDE (Róldan).

- IBM Rational Functional Tester: Proporciona a los testers con poca experiencia funciones automatizadas para actividades como, por ejemplo, la generación de pruebas y el testing dirigido por datos.
  Incorpora soporte para el control de la versión para permitir un desarrollo paralelo de los scripts de verificación y el uso simultáneo por parte de equipos distribuidos por el mundo.
  Permite la realización de pruebas de aplicaciones creadas con VS.NET Winforms, J2SE/J2EE, HTML/DHTML, XML, JavaScript y applets de Java e incluye soporte exclusivo para la biblioteca SWT de Java asociada con el shell de Eclipse (10Ju1).

f **Las herramientas de pruebas de regresión:** Se denominan Pruebas de regresión a cualquier tipo de pruebas de software que intentan descubrir las causas de nuevos errores (bugs), carencias de funcionalidad, o divergencias funcionales con respecto al comportamiento esperado del software, inducidos por cambios recientemente realizados en partes de la aplicación que anteriormente al citado cambio no eran propensas a este tipo de error. Esto implica que el error tratado se reproduce como consecuencia inesperada del citado cambio en el programa.
  Este tipo de cambio puede ser debido a prácticas no adecuadas de control de versiones, falta de consideración acerca del ámbito o contexto de producción final y extensibilidad del error que fue corregido (fragilidad de la corrección), o simplemente una consecuencia del rediseño de la aplicación.

  Herramientas más comunes para pruebas de regresión son:

- Rational Robot: permite a los equipos de pruebas ('testers') automatizar las pruebas de regresión de aplicaciones .NET, Java, web y otras aplicaciones basadas en GUI. Automatiza el testing de regresión, funcional y de configuración para aplicaciones e-commerce, cliente/servidor y ERP. Se usa para testear aplicaciones basadas en una gran variedad de tecnologías de interfaz de usuario y está integrado a la solución Rational TestManager para proporcionar soporte para administrar todas las actividades de testing.
- ManageEngine QEngine: es una característica rica herramienta de pruebas de regresión. La capacidad de pruebas de regresión ayuda a automatizar las pruebas de funcionalidad, rendimiento y capacidad de manejo de carga de aplicaciones web como nuevas funcionalidades se han añadido al software. Además de ser fácil de instalar y mantener, QEngine soporta una arquitectura distribuida, que puede escalar a las necesidades de su organización y ayudar a su equipo de desarrollo para producir software de alta calidad.
- vTest: es una herramienta para automatizar pruebas funcionales y pruebas de regresión para aplicaciones web. Se le autoriza para verificar a fondo y validar las aplicaciones web en una variedad de ambientes. Usted puede mejorar la productividad mediante la generación de scripts automatizados de prueba, se repiten tanto automatizados y personalizados scripts de prueba, generación de informes de ensayo y detectar los errores de software al principio del ciclo de desarrollo.

  VTest Verisium mejora la eficiencia de las pruebas ingeniero a través de la capacidad de crear guiones de prueba automatizados. Estos scripts de prueba se pueden ejecutar de forma automática con mínima intervención humana. También pueden programarse en base a un plan establecido. Esto suele desembocar en una mejora de la productividad y permite a los grandes departamentos de pruebas para llevar a cabo más trabajo de manera significativa en la misma cantidad de tiempo.

g. **Las herramientas de pruebas de integración y aceptación:** Las pruebas de integración se llevan a cabo durante la construcción del sistema, involucran a un número creciente de módulos y terminan probando el sistema como un conjunto.

Estas pruebas se pueden plantear desde un punto de vista estructural o funcional. Las pruebas estructurales de integración son similares a las pruebas de caja blanca; pero trabajan a un nivel conceptual superior. En lugar de referirnos a sentencias del lenguaje, nos referiremos a llamadas entre módulos. Se trata pues de identificar todos los posibles esquemas de llamadas y ejercitarlos para lograr una buena cobertura de segmentos o de ramas.

Las pruebas funcionales de integración son similares a las pruebas de caja negra. Aquí trataremos de encontrar fallos en la respuesta de un módulo cuando su operación depende de los servicios prestados por otro(s) módulo(s). Según nos vamos acercando al sistema total, estas pruebas se van basando más y más en la especificación de requisitos del usuario.

Las pruebas finales de integración cubren todo el sistema y pretenden cubrir plenamente la especificación de requisitos del usuario. Además, a estas alturas ya suele estar disponible el manual de usuario, que también se utiliza para realizar pruebas hasta lograr una cobertura aceptable.

Pruebas de Aceptación. Estas pruebas las realiza el cliente. Son básicamente pruebas funcionales, sobre el sistema completo, y buscan una cobertura de la especificación de requisitos y del manual del usuario. Estas pruebas no se realizan durante el desarrollo, pues sería impresentable de cara al cliente; sino una vez pasada todas las pruebas de integración por parte del desarrollador. La experiencia muestra que aún después del más

cuidadoso proceso de pruebas por parte del desarrollador, quedan una serie de errores que sólo aparecen cuando el cliente se pone a usarlo.

Herramientas más comunes:

- FitNesse: Es una herramienta de colaboración de desarrollo de Software, que mejora la colaboración en el desarrollo de Software, permite a los Clientes, Tester y desarrolladores comparar lo que debe hacer el software con lo que realmente hace. Se comprar las expectativas del cliente con los resultados reales. Se pueden realizar pruebas de aceptación y pruebas de reglas de negocio (10Ju2).

- Avignon: permite a los usuarios expresar pruebas de aceptación de una forma no ambigua antes que comience el desarrollo. Trabaja en conjunto con JUnit, HTTPUnit, etc (10Ma).

h. Las herramientas de pruebas de seguridad. El desarrollar una aplicación segura, es otro apartado imprescindible, ya que posteriormente pueden surgir problemas muy serios. A continuación se verán algunas herramientas para la comprobación de la seguridad.

Herramientas más comunes:

- Powerfuzzer: es totalmente gratuito y permite a los administradores crear tests personalizados para sus aplicaciones Web con la intención de detectar agujeros de seguridad. En esencia, se trata de un simple escaner de aplicaciones. Algunos de los tests que ofrece son los siguientes: Código de páginas cruzadas (XSS), Inyecciones (SQL, LDAP, código, comandos, y XPATh), CRLF y Estados HTTP 500.

- Netsparker: es un producto de análisis de vulnerabilidades web. Entre sus principales características se anuncia que está libre de reportar falsos positivos, o lo que es lo mismo, identificar como vulnerabilidades peticiones que realmente no lo son. Aunque esta característica tiene truco, ya que las que no puede garantizar como vulnerables las etiqueta como "posibles", para que posteriormente el auditor haga las comprobaciones oportunas.

  Los motores soportan la detección de los riesgos más comunes: Inyección SQL, XSS, inclusión local y remota de ficheros, inyección de comandos, CRLF, archivos obsoletos, código fuente, recursos ocultos, listado de directorios, vulnerabilidades de configuración de los distintos servidores web, etc. Pero sin duda el que más destaca y funciona mejor es el de inyección SQL, que además permite la ejecución de comandos y de sentencias una vez se detecta un parámetro vulnerable.

- Nessus: es un programa de escaneo de vulnerabilidades en diversos sistemas operativos. Consiste en nessusd, el daemon Nessus, que realiza el escaneo en el sistema objetivo, y nessus, el cliente (basado en consola o gráfico) que muestra el avance y reporte de los escaneos. Desde consola nessus puede ser programado para hacer escaneos programados con cron.

  En operación normal, nessus comienza escaneando los puertos con nmap o con su propio escaneador de puertos para buscar puertos abiertos y después intentar varios exploits para atacarlo. Las pruebas de vulnerabilidad, disponibles como una larga lista de plugins, son escritos en NASL (Nessus Attack Scripting Language, Lenguaje de Scripting de Ataque Nessus por sus siglas en inglés), un lenguaje scripting optimizado para interacciones personalizadas en redes.

  Opcionalmente, los resultados del escaneo pueden ser exportados en reportes en varios formatos, como texto plano, XML, HTML, y LaTeX. Los

resultados también pueden ser guardados en una base de conocimiento para referencia en futuros escaneos de vulnerabilidades.

Algunas de las pruebas de vulnerabilidades de Nessus pueden causar que los servicios o sistemas operativos se corrompan y caigan. El usuario puede evitar esto desactivando "unsafe test" (pruebas no seguras) antes de escanear.

i. **Comprobadores de Links (Link Checkers):** Si el proyecto que estamos desarrollando se trata de una aplicación web, es imprescindible que todos los links del portal funcionen perfectamente. Para comprobarlo, a continuación se pueden ver 2 herramientas interesantes.

Herramientas más comunes:

- W3CLink Checker: Al igual que el “W3C Validator” para comprobar que la web cumple los estándares del W3C, W3C también nos permiten analizar una web en busca de enlaces rotos.
  Se pueden comprobar los enlaces de una web en la siguiente dirección (10Ag1):

- LinkChecker: Plugin para firefox mediante el que podremos validar si los links de una web funcionan correctamente.

j. **Validadores de HTML (HTML Validators).**Tener un HTML correcto es otro factor muy importante a la hora de realizar una aplicación web. Por ello, a continuación podremos ver 2 herramientas para poder revisar el HTML de una web.

- W3C Validator: Es uno de los validadores más conocidos en la actualidad. Es muy exigente y en ocasiones es difícil cumplirlo al 100%. Aun así, es recomendable seguir sus directrices (10Ag).

- HTML Validator: Plugin para firefox mediante el que podremos validar el HTML de nuestra web.

2.2.1.2 Las herramientas para la administración de pruebas

Para realizar pruebas es necesario tener muchas listas, es decir, listas de las pruebas y listas de los problemas. Las herramientas que ayudan a administrar dichas listas pueden mejorar la comunicación y la documentación y ayudar a que el proyecto funcione mejor. Estas herramientas no sólo pueden usarse para administrar las listas de pruebas y de problemas, sino que pueden configurarse para dar soporte a las listas de requisitos, de mejoras, de asignaciones de trabajo, de contactos de los clientes, etc. Las herramientas específicas para las pruebas tienen funcionalidades que soportan directamente los requisitos de los equipos de pruebas y control de calidad. Dentro de esta categoría encontramos herramientas para:

- **Las herramientas de generación de pruebas**, como su nombre lo indica, generan pruebas dentro de ciertos parámetros para que se adapten a un modelo del sistema que se está evaluando. A pesar de que funcionan mejor cuando se integran con una herramienta de automatización de pruebas y una herramienta de gestión de pruebas, pueden ser herramientas analíticas útiles para las pruebas manuales.
- **Rastreador de defectos** son usados por la mayor parte de equipos para mantener una lista centralizada de los problemas que se registran. Las herramientas permiten clasificar y priorizar los problemas para que se

asignen a las personas a medida que van de ser detectados a ser resueltos.

- **Las herramientas de manipulación de datos** trabajan con conjuntos y bases de datos en masa. Permiten que un equipo analice los datos de salida y simplifican la construcción precisa y completa de datos de las pruebas.
- **Las herramientas de gestión de ambientes** se usan normalmente con las máquinas en el laboratorio de pruebas. Permiten configurar el laboratorio para que sea supervisado y pueden ayudar a que los evaluadores reconfiguren su ambiente de pruebas de forma activa.
- **Las herramientas de gestión de pruebas** permiten agrupar, ordenar, priorizar y asignar una lista de pruebas, ayudando a que el equipo administre el trabajo relacionado con las pruebas. Las herramientas más enfocadas crearán reportes acerca del progreso de las pruebas y podrán capturar y resumir los resultados y las historias de las mismas, permitiendo comparar y analizar las tendencias.

Es de buena práctica cuando se va a realizar una prueba a la aplicación, sin importar el tipo de prueba que se va a llevar a cabo, ni la fase de desarrollo en el que se encuentre la aplicación, diseñar un documento con las pruebas a realizar y documentar el resultado de esos casos de prueba, lo anterior da pie a que surjan un sin número de documentos asociados a cada caso, diseño de prueba y su resultado, de allí la necesidad de contar con una herramienta que permita generar, controlar estos documentos de modo organizado, evitando la pérdida de documentación y agilizando el proceso de diseño en la prueba del software y que facilite la adecuada gestión de todos estos documentos y la disponibilidad de la información relacionada, de modo que se pueda organizar las pruebas por proyectos o por módulos, según la empresa lo requiera.

Las herramientas más comunes para la gestión de pruebas son:

- T-Plan Professional
- Testlink
- RTH
- QaTraq
- TestDirector
- Case Maker
- Qualify 2.4

En el estado actual de la empresa Choucair Testing, desde la planeación se consideran los tipos de pruebas que se van a realizar dependiendo de las modificaciones que haya tenido el aplicativo o dependiendo de la etapa del desarrollo de software en que se encuentre, y es responsabilidad del tester encargado de hacer el diseño de las pruebas, algunos de los tipos de test utilizados allí son:

- Smoke Test.
- Regresión
- Funcionalidad – integración: hay de 3 tipos

  - Test Funcionales Unitarios
  - Test funcionales- integración con el resto del aplicativo
  - Test de Integración (con otros aplicativos)

- Peformance Básico

La planeación de los casos de prueba se realizan con base en los riesgos del producto software, el alcance que se le ha dado y los recursos, de allí se seleccionan los tipos de pruebas que se van a realizar y se hacen los diseños de los casos de pruebas y los escenarios de las pruebas.

Una vez definidos estos casos de prueba, se procede a la ejecución manual de los casos de prueba, con la ayuda de plantillas en Excel en el cual se ingresa la

información básica de la prueba como el proyecto, el cliente, el tipo de prueba, el responsable, la fecha y demás información general, y luego de una forma más detallada se coloca el paso a paso del caso de prueba con su resultado esperado, el cual servirá como referencia para indicar si el resultado de la prueba para ese paso fue OK o no, se coloca un comentario y el # de Bug según sea el caso.

A partir de esta información se sacan estadísticas como el número de pasos de la prueba, pasos OK, Pasos no OK, número de pasos ejecutados y número de casos por ejecutar.

Dado el volumen de información que maneja esta empresa por el tamaño de sus pruebas se puede dificultar el manejo y control de estas plantillas, y la disponibilidad de estas para todos los integrantes del equipo de Testing. Con el fin de mejorar o agilizar este proceso y evitar el uso de herramientas adicionales para comunicarse (correo electrónico o unidades compartidas) y poner a disposición de todos estas plantillas, además de mejorar la ejecución de reportes que brinden información veraz y oportuna, se realizará un estudio detallado de las 4 herramientas de gestión de pruebas más comunes en nuestro medio y de las cuales se encuentra más documentación y permiten un mejor análisis, ellas son:

- Testlink
- QaTraq
- Case Maker
- Qualify 2.4

Para cada una de estas herramientas se definirá una descripción, prerrequisitos, tipo de licenciamiento y un análisis de la herramienta como tal, como está estructurada, como es la creación y administración de los

proyectos, casos de pruebas, roles, reportes y demás actividades que se pueden realizar con este tipo de herramientas y cuál será el flujo de trabajo.

Qualify 2.4

- **Descripción general:** Sistema de gestión avanzado de casos de prueba, diseñado por profesionales de QA.

  La metodología de descomposición de aplicaciones y herramientas de apoyo proporcionadas por Qualify están diseñados para proporcionar un enfoque claro y pragmático de las pruebas del software mientras que proporciona un alto grado de flexibilidad, utiliza un enfoque modular en la construcción de los casos de prueba. Este enfoque permite la reutilización de artefactos de prueba y presta al mantenimiento de sus pruebas. Proporciona los medios para asignar las pruebas directamente a la interfaz de usuario. Esto permite una evaluación cuantificable de la cobertura de la prueba de interfaz de usuario.

  Las características de informes en Qualify ofrecen a los gestores y los probadores la información que necesitan para evaluar el rendimiento pasado y actual. Está diseñado para que los equipos a trabajen de modo eficiente con un proceso de instalación fácil e interfaz de usuario intuitiva.

- **Licenciamiento:** El licenciamiento de esta herramienta es no es gratuita, se cobra por precio licencia a nodo cerrado y precio licencia flotante. Sin embargo, tiene un trial por 30 días.

- **Prerrequisitos:** El servidor administrador Qualify debe estar instalado en una de 32 bits de SQL \ SQL Server Express. Los requisitos del sistema son los siguientes:

Tabla 2.2 Requisitos del sistema para instalación Qualify 2.4

Procesador	600-megahertz (MHz) Pentium III-compatible o procesador más alto; 1-gigahertz (GHz) recomendado o más alto.
Framework	.NET Framework 3.0
Sistema operativo	• Windows XP with Service Pack 2 or later • Microsoft Windows 2000 Professional with SP4 • Microsoft Windows 2000 Server with Service Pack 4 or later • Windows Server 2003 Standard, Enterprise, or Datacenter editions with Service Pack 1 or later • Windows Server 2003 Web Edition SP1 • Windows Small Business Server 2003 with Service Pack 1 or later • Vista Home Basic and above (SQL Express SP1 and SQL Express Advanced SP2)
Memoria	192 megabytes (MB) of RAM or more; 512 megabytes (MB) recomendado o mas alto.
Disco Duro	• Aproximadamente 350 MB de espacio disponible en disco, recomendado para la instalación. • Aproximadamente 425 MB de espacio adicional en disco duro disponible para SQL Books Online Server, SQL Server Mobile Libros en pantalla, y bases de datos como ejemplo.

Tabla 2.3 Requisitos del sistema para el cliente Qualify 2.4

Procesador	400-megahertz (MHz) Pentium III-compatible o procesador más alto; 1-gigahertz (GHz) o mas alto (recomendado).
Framework	.NET Framework 3.0
Sistema Operativo	• Windows XP with Service Pack 2 or later • Microsoft Windows 2000 Professional with SP4 • Microsoft Windows 2000 Server with Service Pack 4 or later • Windows Server 2003 Standard, Enterprise, or Datacenter editions with Service Pack 1 or later • Windows Server 2003 Web Edition SP1 • Windows Small Business Server 2003 with Service Pack 1 or later • Vista Home Basic and above (SQL Express SP1 and SQL Express Advanced SP2)
Memoria	512 megabytes (MB) o más alto (recomendado)
Disco duro	• The Qualify Client takes only ~ 10-15MB of hard drive space. The .NET Framework v.3.0 that is required takes an additional 500MB.

Antes de su instalación es necesario tener instalado SQL Express Server 2005 como base de datos. Para el uso de la herramienta es necesario instalar el servidor administrador de Qualify y el cliente.

Qualify ha sido diseñado para su reutilización y la eficiencia para reducir el costo y el tiempo necesario para mantener sus planes de prueba. Como parte de este, Qualify utiliza una jerarquía de elementos de prueba para construir pruebas. La jerarquía permite crear elementos que puedan ser reutilizados, y grupos de elementos que pueden ser reutilizados.

- **Funcionalidades y estudio de la herramienta:**

La jerarquía de prueba se compone de cuatro elementos:

Plan de Pruebas. Es una colección ordenada de escenarios, casos de prueba, y pasos de prueba. Los planes de pruebas están en la cima de la jerarquía de la prueba y representan un conjunto completo de acciones necesarias para verificar una aplicación completa o una porción grande de una aplicación como un módulo. Los planes de pruebas se asignan a los usuarios para su ejecución.

Escenarios Un escenario es una colección ordenada de casos de prueba o de pasos de prueba. Los escenarios representan flujo de trabajo en una aplicación y generalmente reflejan cómo una aplicación se utiliza para lograr los objetivos de negocio. Se recomienda para crear escenarios de alto nivel que no incluyen ningún tipo de casos de prueba para la realización de pruebas exploratorias. Al mismo tiempo, usted querrá crear escenarios y agregar todos los casos de prueba necesarios para crear un script de prueba más rígida. Mediante la combinación de ambos métodos, se puede asegurar una cantidad conocida de cobertura de la prueba, mientras que trabajan fuera de los límites de un plan de pruebas "scripts"

para encontrar errores que no se puede encontrar siguiendo a través de una prueba de regresión rígidamente definida.

Casos de Prueba. Un caso de prueba es un conjunto ordenado de pasos de prueba. Los casos de prueba representan una sola actividad que se puede lograr en una pantalla. Por ejemplo, una pantalla de administración de usuario puede incluir casos de prueba para crear, editar y eliminar un usuario. Los casos de prueba se pueden asociar con capturas de pantalla que le permite determinar la cantidad de cobertura de las pruebas que tiene para una pantalla determinada en su aplicación. Los casos de prueba son opcionales y no están obligados a crear un plan ejecutable de prueba. Mediante la creación y el uso de casos de prueba, usted será capaz de crear los componentes necesarios para ensamblar todos los escenarios soportados por su aplicación.

Pasos de Prueba. Estos son la parte inferior de la jerarquía de la prueba. Representan las acciones más básicas que se pueden realizar en una aplicación como hacer clic en los botones o enlaces y seleccionar o introducir valores. Pasos de prueba se puede asignar a la interfaz de usuario utilizando la herramienta de Touchpoint. Esto proporciona en última instancia la capacidad de determinar cuánto cobertura de la prueba usted tiene de su interfaz utilizado en un plan de prueba dado. Con todo esto dicho, los pasos de prueba son opcionales. No es necesario que se creen medidas o pasos de prueba a fin de crear un plan ejecutable de prueba. Mediante el uso de pasos, puede crear una base sólida desde la que puede construir cualquier caso de prueba necesaria para apoyar sus esfuerzos de prueba.

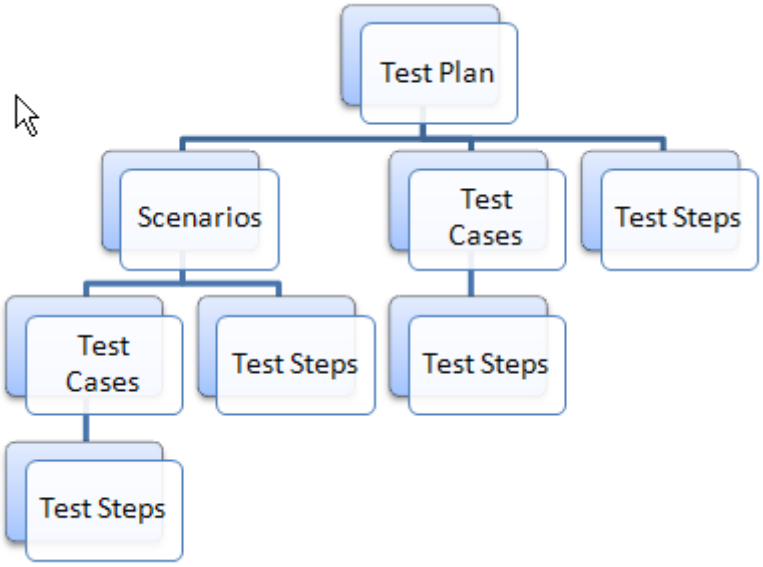


Figura 2.5 Jerarquía pruebas en Qualify 2.4.

Cómo define los roles de usuario Qualify?.

Los roles de usuario se definen en el menú principal por Admin.

Para crear y administrar proyectos, click en Admin → Manage Projects ítem en el menú principal de la barra de herramientas.

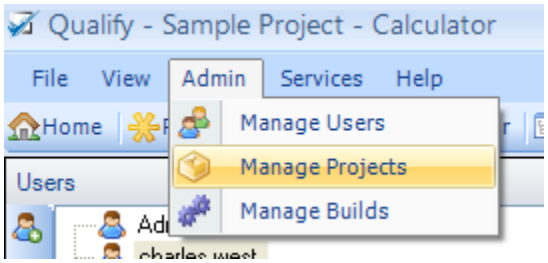


Figura 2.6 Roles de usuario.

Si no existe la opción Admin en el menú, su usuario no tiene permisos para administrar proyectos y usuarios. Si éste es el caso, cualquiera al crear el repositorio tendrá un usuario administrativo con este fin.

Pantalla para la administración de usuarios. Esta permite crear, actualizar y borrar usuarios, a su vez darle permisos. Los permisos se asignan a un proyecto específico y los tipos de permisos están predeterminados. (Sqa Design).

Ver las graficas siguientes:

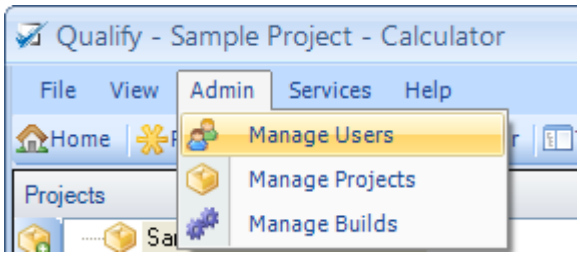


Figura 2.7 Administración de usuario.

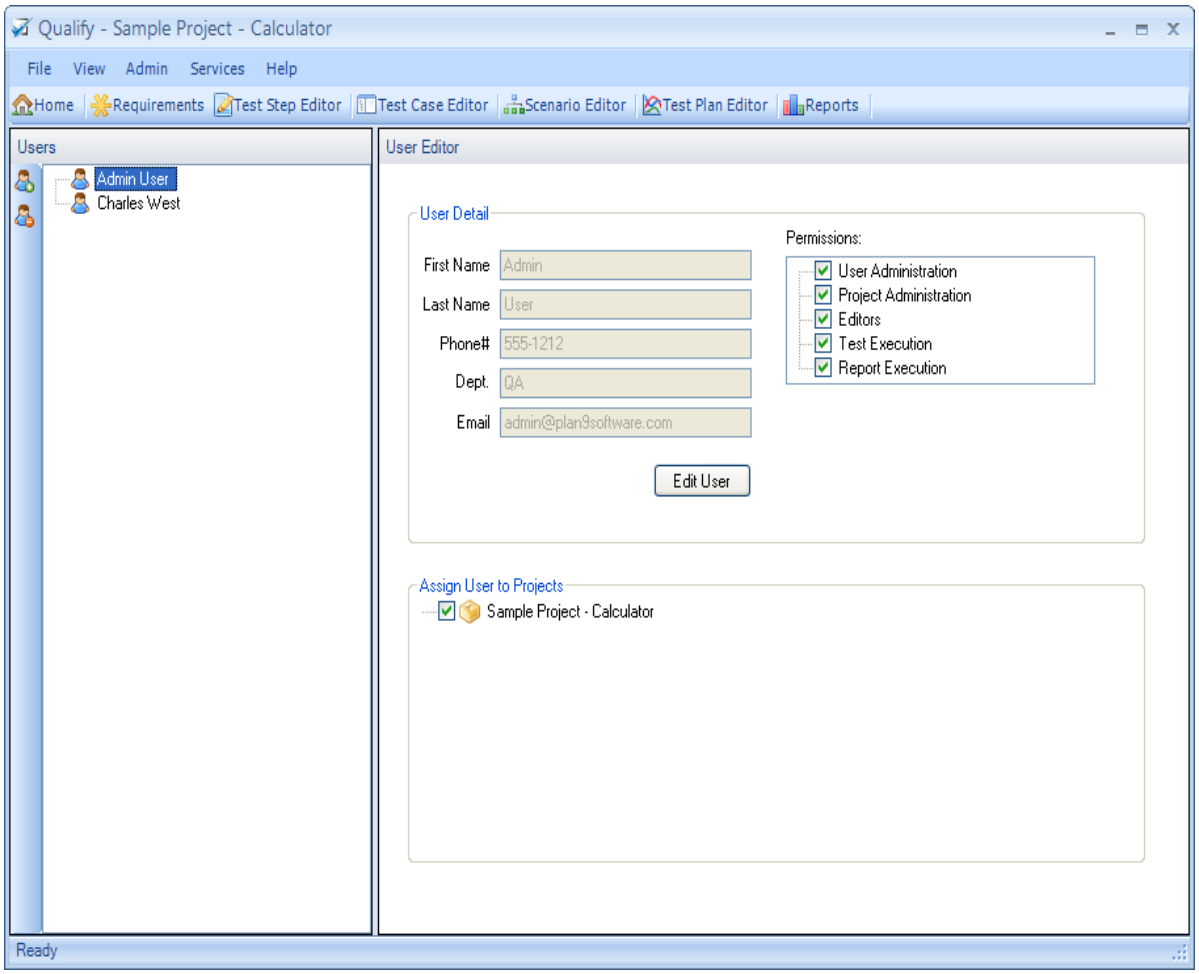


Figura 2.8 Administración de usuario.

Pasos para implementar un caso de prueba en la herramienta Qualify:

Para tal efecto vamos a seguir un ejemplo de un proyecto Calculadora.

1. Debemos Crear un proyecto en el menú Admin, Manage Projects

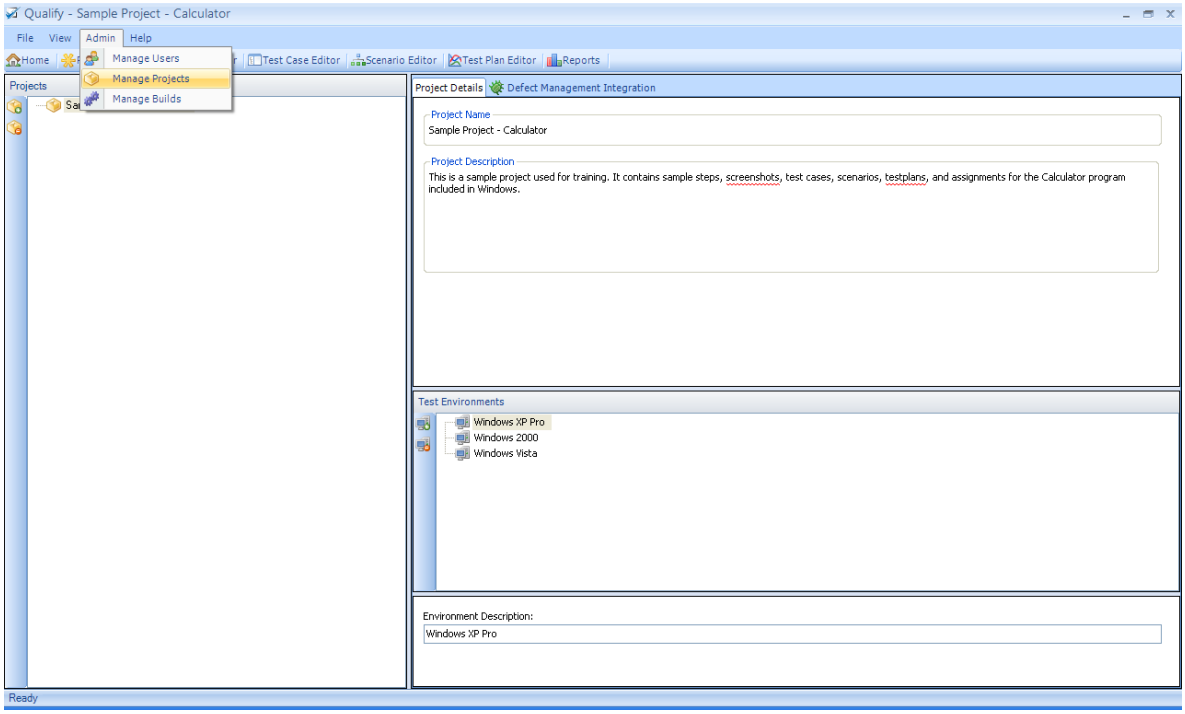


Figura 2.9 Crear un proyecto Qualify 2.4.

Esta vista se compone de varias sesiones que podemos apreciar, el lado izquierdo los proyectos, y al lado derecho el nombre de el proyecto, con su descripción y detalles de el ambiente en donde se probará

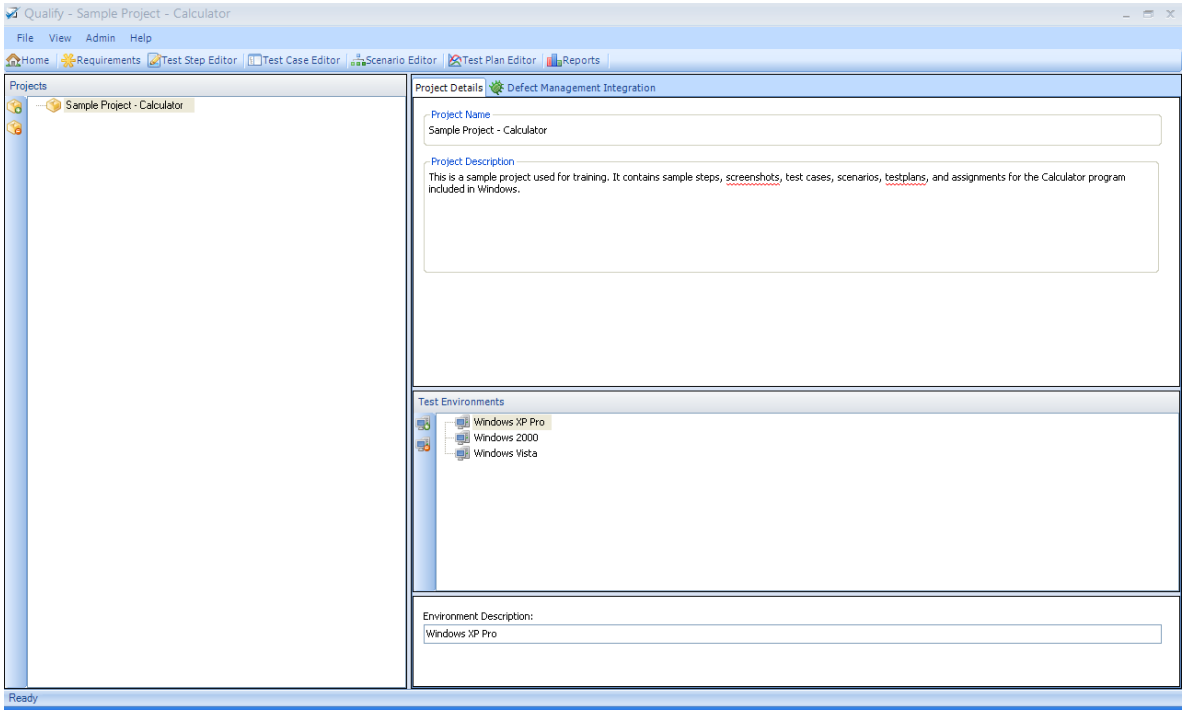


Figura 2.10. Detalle pantalla crear proyecto

2. Requerimientos a tenerse en cuenta para llevar a cabo el caso de prueba de la Calculadora. Se enumeran los requerimientos, que en este caso se encuentran al lado izquierdo. Estos requerimientos se tendrán en cuenta para asociárselos a un caso de prueba, en el editor Test Case.

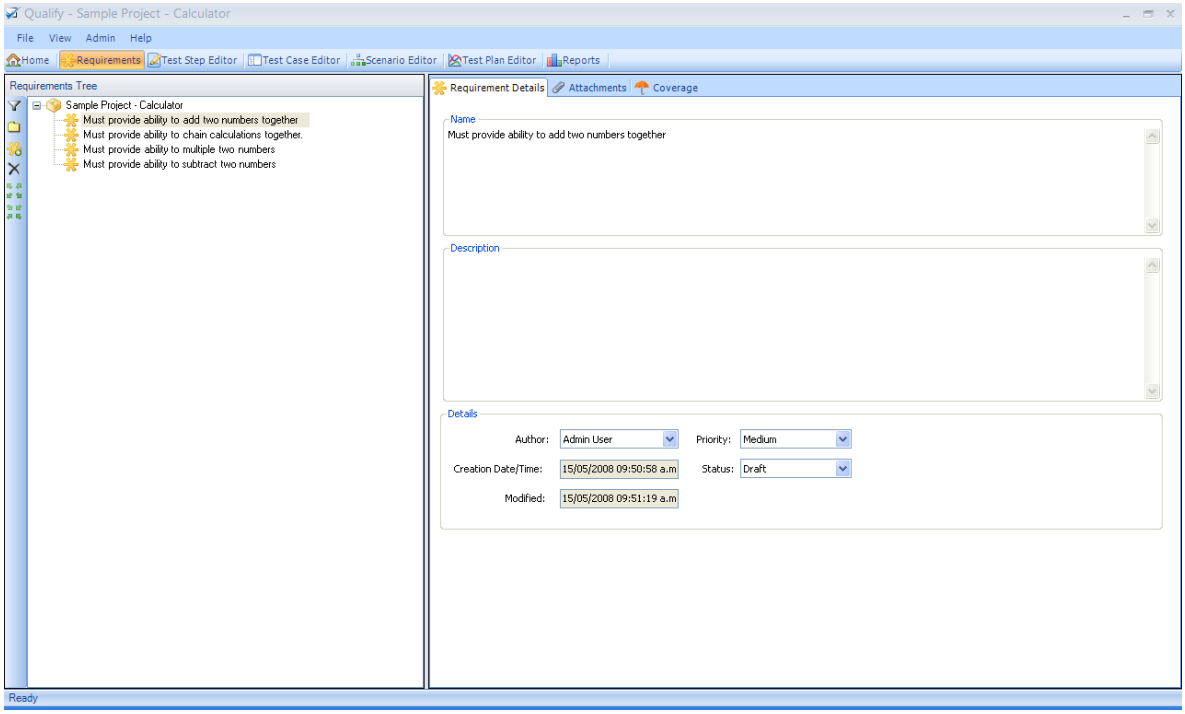


Figura 2.11 Requerimientos para el caso de prueba

Después de ser asociados los Test Case a los requerimientos, podremos ver en la opción de Requirements, no solo los requerimientos sino los casos de prueba asociados al mismo.

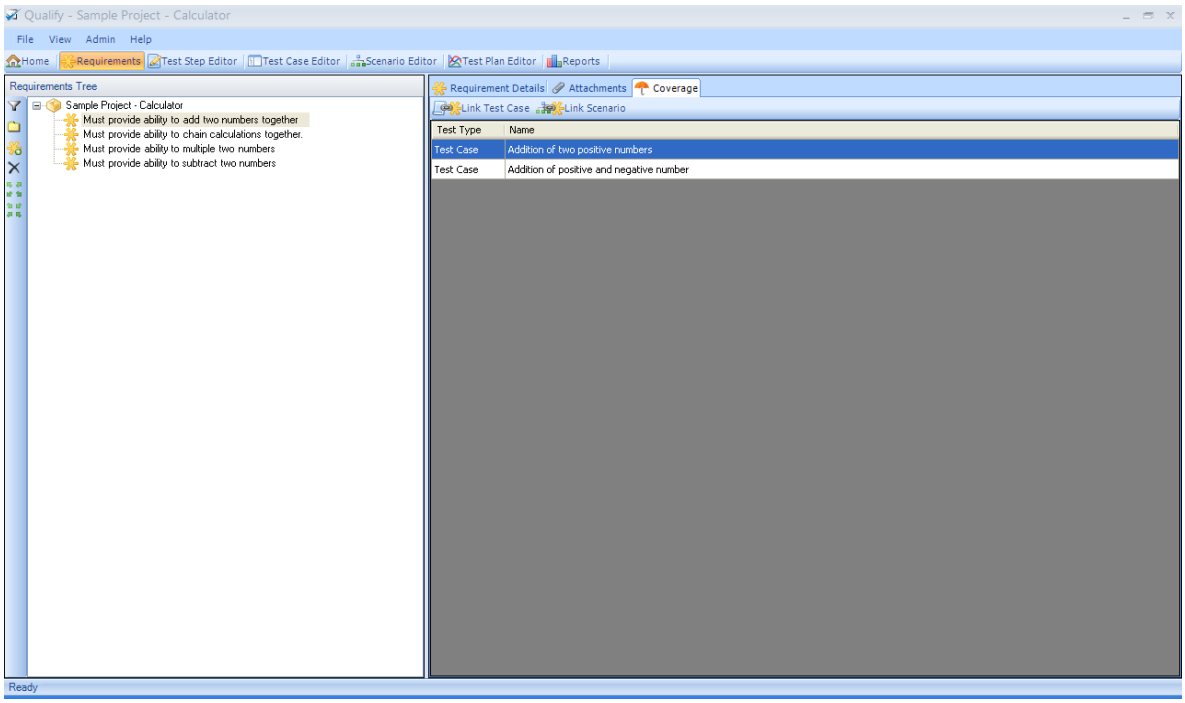


Figura 2.12 Casos de prueba asociados a los requerimientos

3. En este paso usaremos la opción Test Step Editor, esta opción nos permite incluir una imagen de la pantalla o las pantallas donde vamos a ejecutar el requerimiento y a su vez describir los pasos o los clicks que debemos dar en la o las pantallas que aquí se adjuntan. Para este caso en particular, vemos que primero se creó una carpeta de nombre Global ítems, la cual contiene 3 imágenes de las opciones de menú que hay para operar la calculadora. Bajo el mismo nivel de la carpeta Global ítems esta la imagen estándar Calculator, que señala cada ítem contenido en esta imagen.

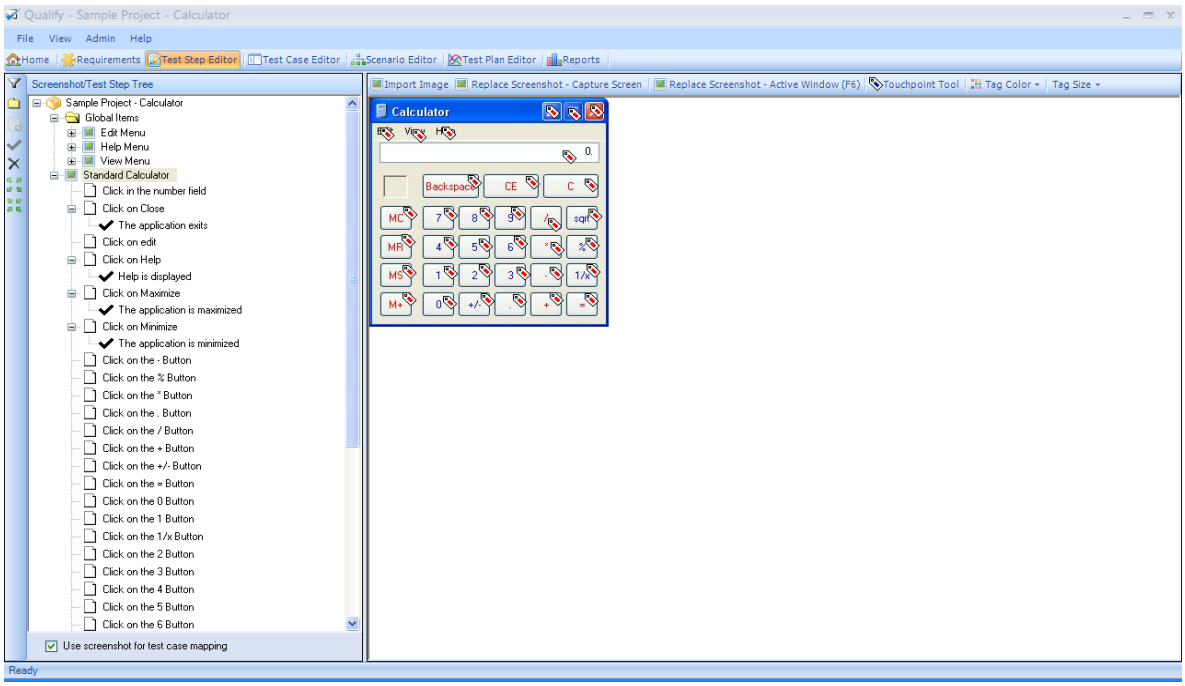


Figura 2.13 Test Step Editor

Como vimos en la pantalla anterior, se importa una imagen y se guarda. Para hacer esto, estando en el Test step Editor, posicionamos el cursor en la carpeta donde deseamos exportar nuestra imagen, le decimos importar imagen y la pantalla lucirá como esta:

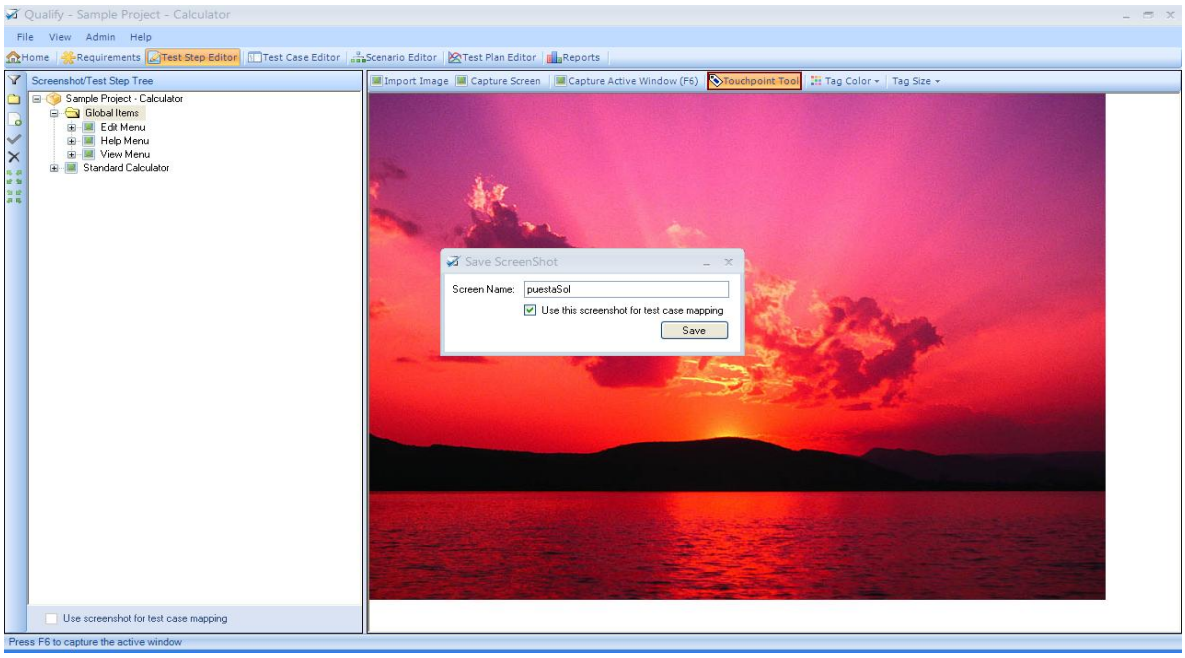


Figura 2.14 Cargar imagen Touchpoint Tool

La cual se grabará. Para hacer referencia a algún punto de esta imagen, seleccionamos Touchpoint Tool y damos click sobre el punto especifico en la imagen, el cual deseamos describir como paso (step).

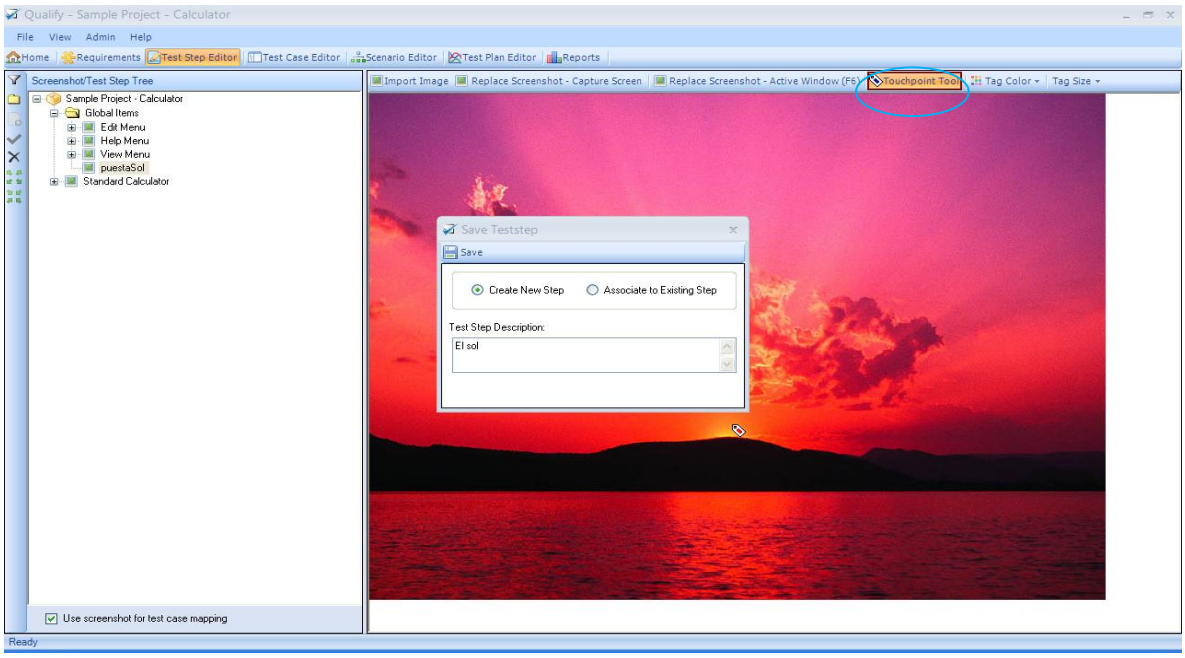


Figura 2.15 Definir el step en Touchpoint Tool

4. El editor de casos de prueba, el cual se va asociar a un requerimiento. En este caso es la suma de un número positivo y otro negativo, en la segunda vista del cuadro derecho aparecen los ítems que se deben dar para llevar a cabo este caso de prueba.

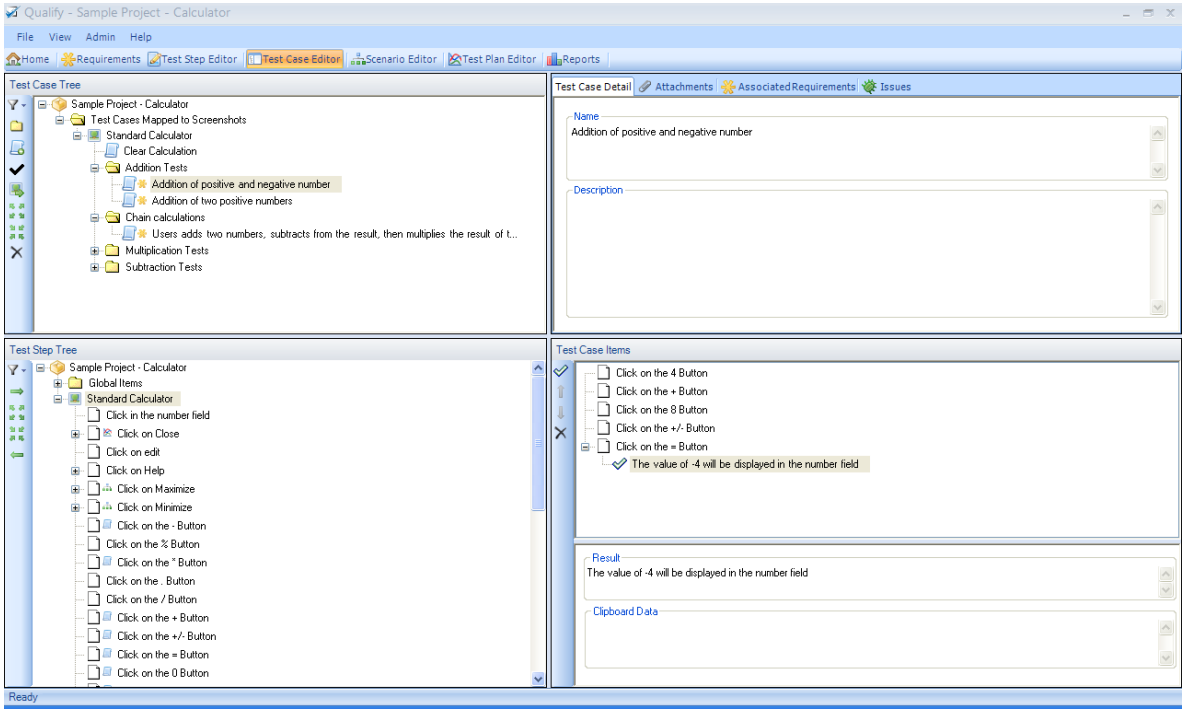


Figura 2.16 Editor Casos de prueba

5. Continuamos con el editor de escenarios, los escenarios son sinónimos de los casos de uso, en este se plantean los ítems de los escenarios con sus resultados esperados si es necesario (véase en la pantalla Scenario items).

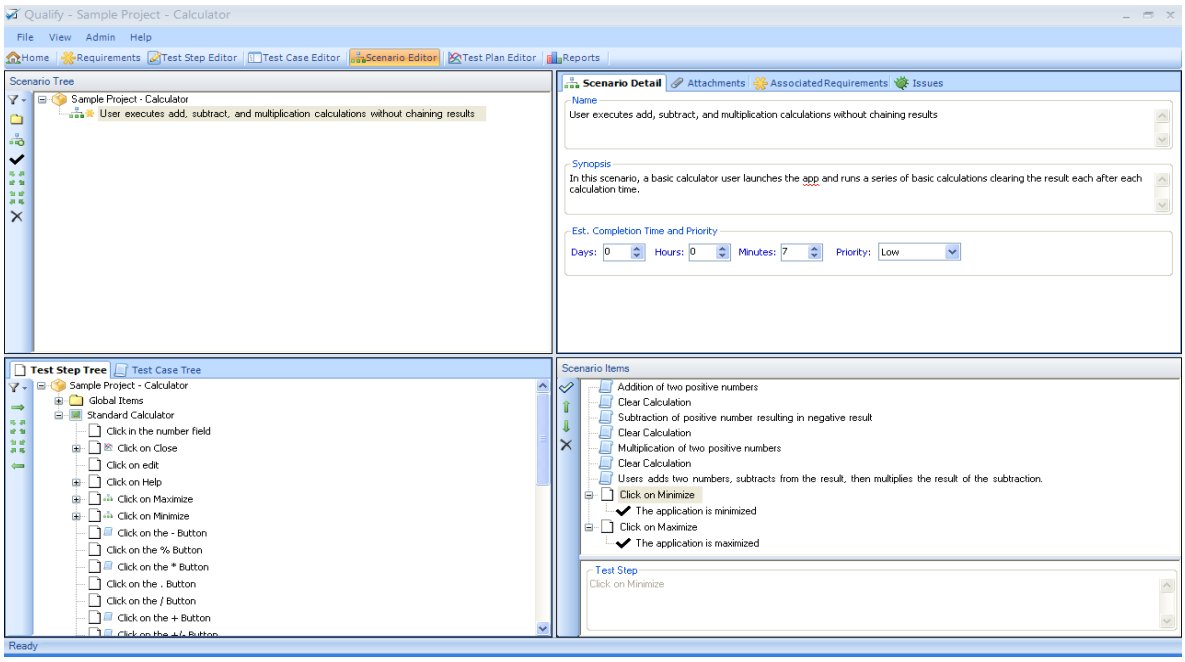


Figura 2.17 Editor escenario

6. En la vista o tab, Test Plan Editor, podemos realizar las siguientes acciones:

- Crear, corregir, organizar y suprimir planes de prueba.
- Agregar escenarios, los casos de prueba, y o pasos de la prueba a los planes de prueba.
- Agregar resultados dinámicos a los escenarios, los casos de prueba o los pasos de la prueba.
- Sincronizar planes de prueba.
- Asociar planes de prueba a las ediciones.
- Asignar planes de pruebas a los usuarios.
- Determinar tiempo de ejecución de la prueba.

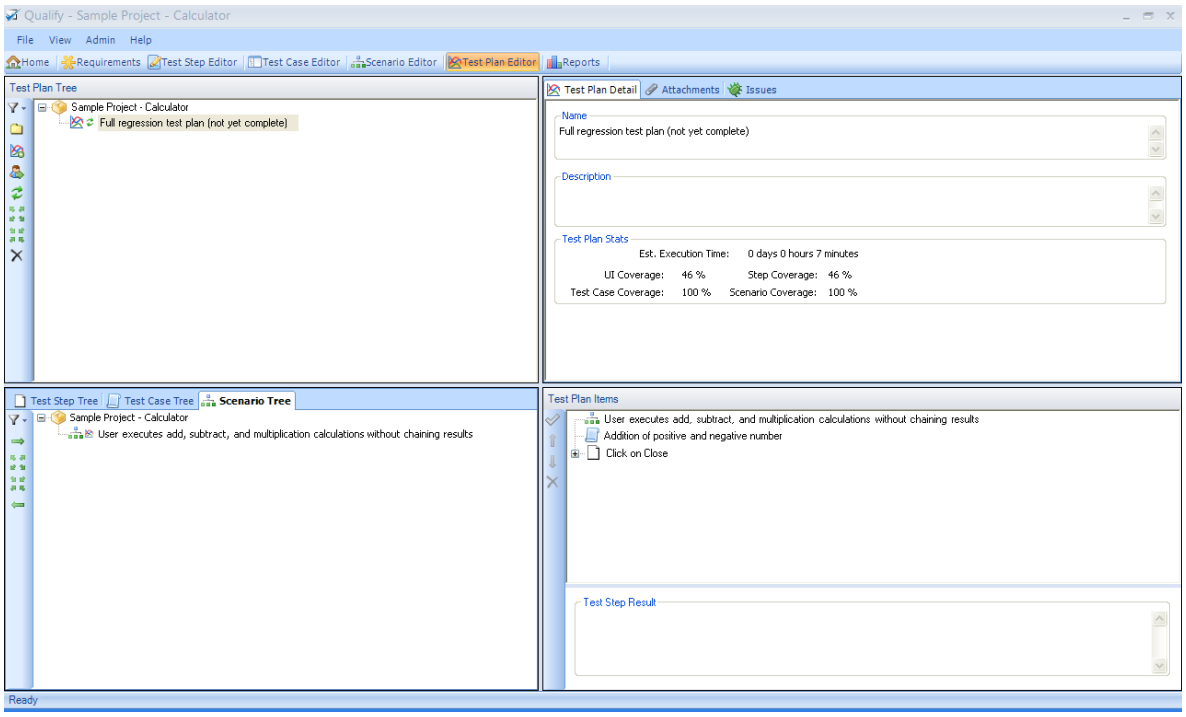


Figura 2.18 Test Plan Editor

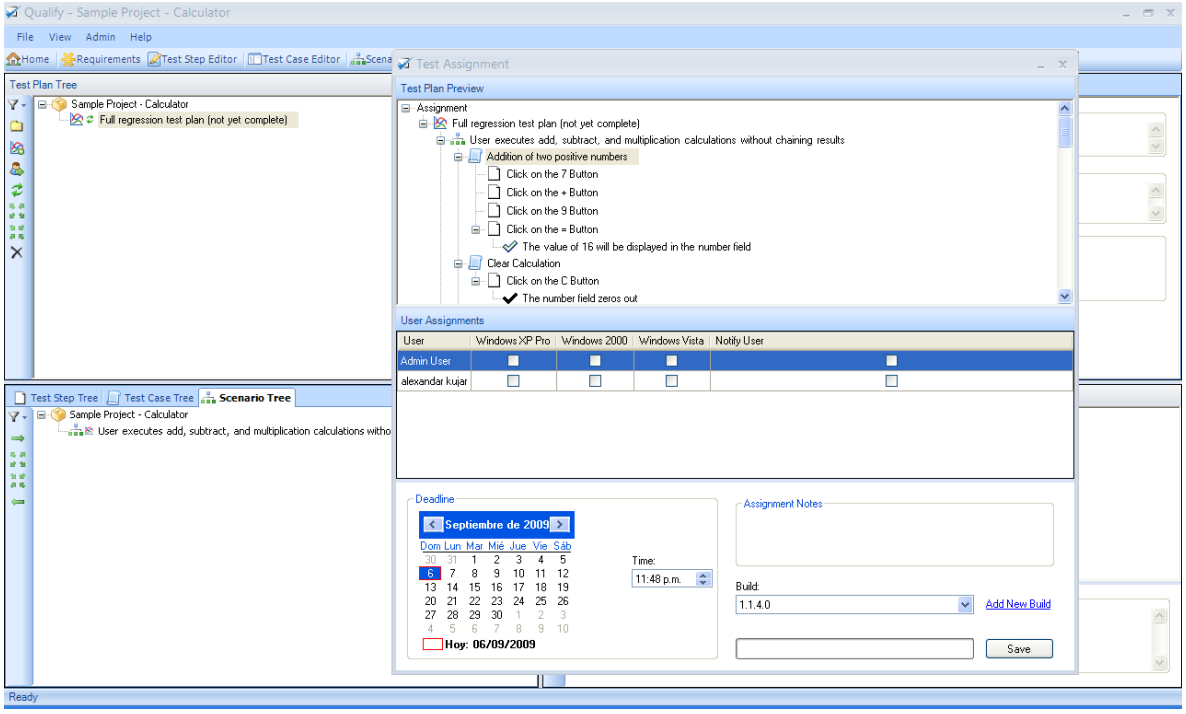


Figura 2.19 Tab Test Plan Editor

QATRAQ

- **Descripción general:** Permite a consolidar el proceso de prueba durante todo el ciclo de vida de todas las pruebas con un conjunto de herramientas de gestión de prueba. Desde comunicar sus planes de prueba, a la gestión de los resultados y la elaboración de informes. Proporcionando una ubicación central para la documentación de prueba con el control de revisión total. Los documentos están vinculados a través del proceso, por lo tanto los scripts de prueba están relacionados con el plan de pruebas pertinentes, y los casos de prueba relacionados a los scripts de prueba adecuados.

- **Licenciamiento:** El costo de esta herramienta varía dependiendo del número de usuarios concurrentes que va a soportar que va desde 1 hasta ilimitado, lo cual incluye un año de mantenimiento y actualizaciones, cada año se puede renovar este contrato de mantenimiento que tendrá un costo dependiendo del plan que se adquirió inicialmente.

- **Prerrequisitos:** Esta herramienta corre bajo las plataformas Linux/Unix y Windows.

  Pasos de instalación:

  Como QaTraq profesional se basa en Apache, Mysql y PHP estos debe estar instalados antes de instalar QaTraq profesional, si no lo están, estas serán instaladas cuando se instala Qatraq.

  Instalación propia de la herramienta
  Ejecutar el instalador Qatraq_pro-7_0_0 e instalar en la ruta seleccionada.

En la ruta C:\Program Files\QaTraq\htdocs\qatraq\licence se copia el archivo de la licencia.

- **Funcionalidades y estudio de la herramienta:**

En la figura 3.14 y 3.15 se ilustra los diferentes componentes de la herramienta Qatraq

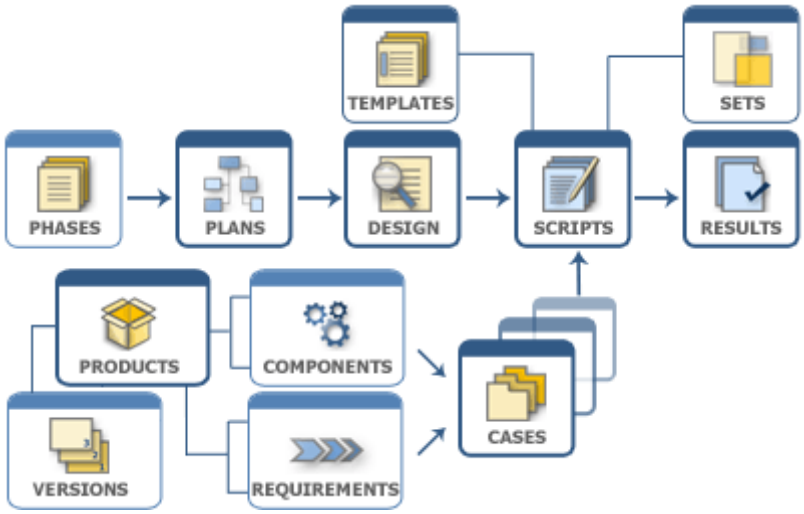


Figura 2.20 Componentes de Qatraq Professional

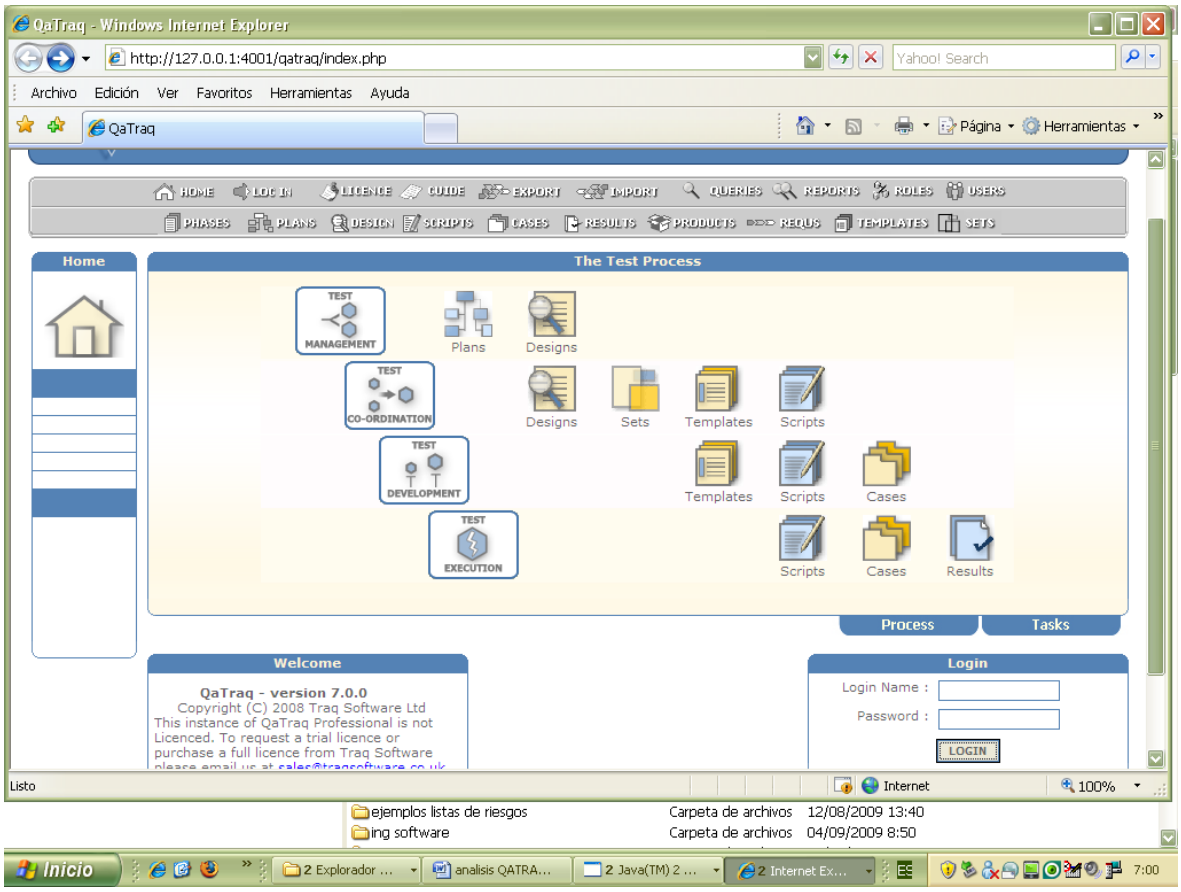


Figura 2.21 Componentes de Qatraq Professional

Estos componentes son descritos a continuación:

ROLES: Se define el nombre del rol, se asignan privilegios que tendrán sobre cada actividad que se puede desarrollar como proyectos, fases, planes, diseños, Queries y los permisos sobre estos (ver, modificar, crear, borrar, copiar, incluir, remover)

USER: Se define el LOGIN, el USER NAME, PASSWORD y el rol al que pertenece.

REPORTES: Permite ejecutar reportes existentes o Crear nuevos reportes a partir de unos prediseñados y con las especificaciones necesarias.

Ventana principal opción PROJECT: Para crear un nuevo Proyecto o Buscar uno existente

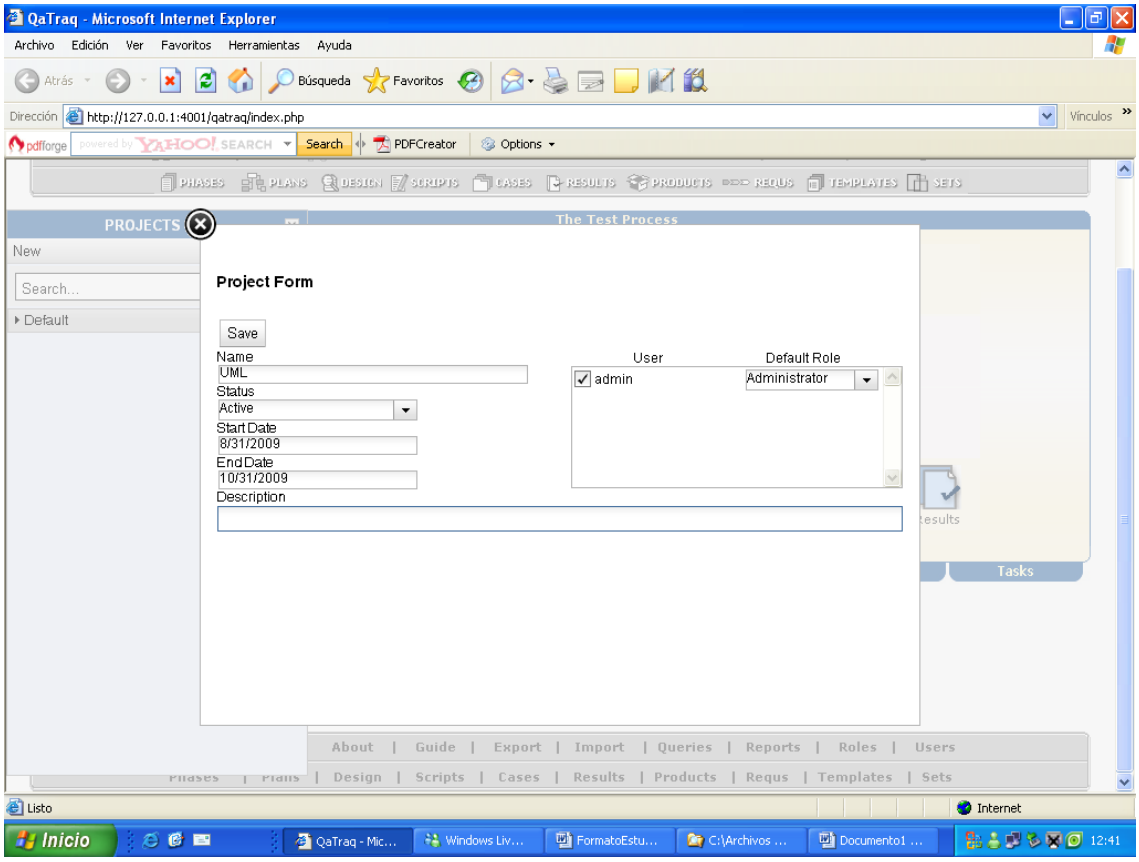


Figura 2.22 Ventana principal crear Project

Grabar y se podrá crear otro o visualizar el creado.

PHASES: Acá se define los diferentes tipo de fases de prueba (p. ej. pruebas de unidad, pruebas de sistema, etc.). Permite definir sus fases de prueba y registrar la información como la entrada y criterios de salida aplicables a cada fase de

prueba. Una vez definidas es posible relacionar la prueba planifica a ciertas fases de prueba.

Se puede crear un nuevo documento de fase de prueba o buscar uno ya existente para ver, modificar, suprimir o copiar.

PLANS: El Plan de prueba proporciona un depósito para todas las actividades de pruebas que se llevarán a cabo y la documentación que se relaciona con la planificación de un proyecto de pruebas (todos los equipos, recursos necesarios, calendario, resultados). También debe especificar todas las responsabilidades, incluidas las de otros que necesitan para entregar en el equipo de pruebas. Podría contener diagramas de Gantt , detalles de cómo los errores serán reportados y cómo la construcción será estructurada y pruebas de regresión.

Cada plan de prueba puede ser relacionado con una o varias fases paternales de prueba y puede contener uno o vario diseño de prueba áreas.

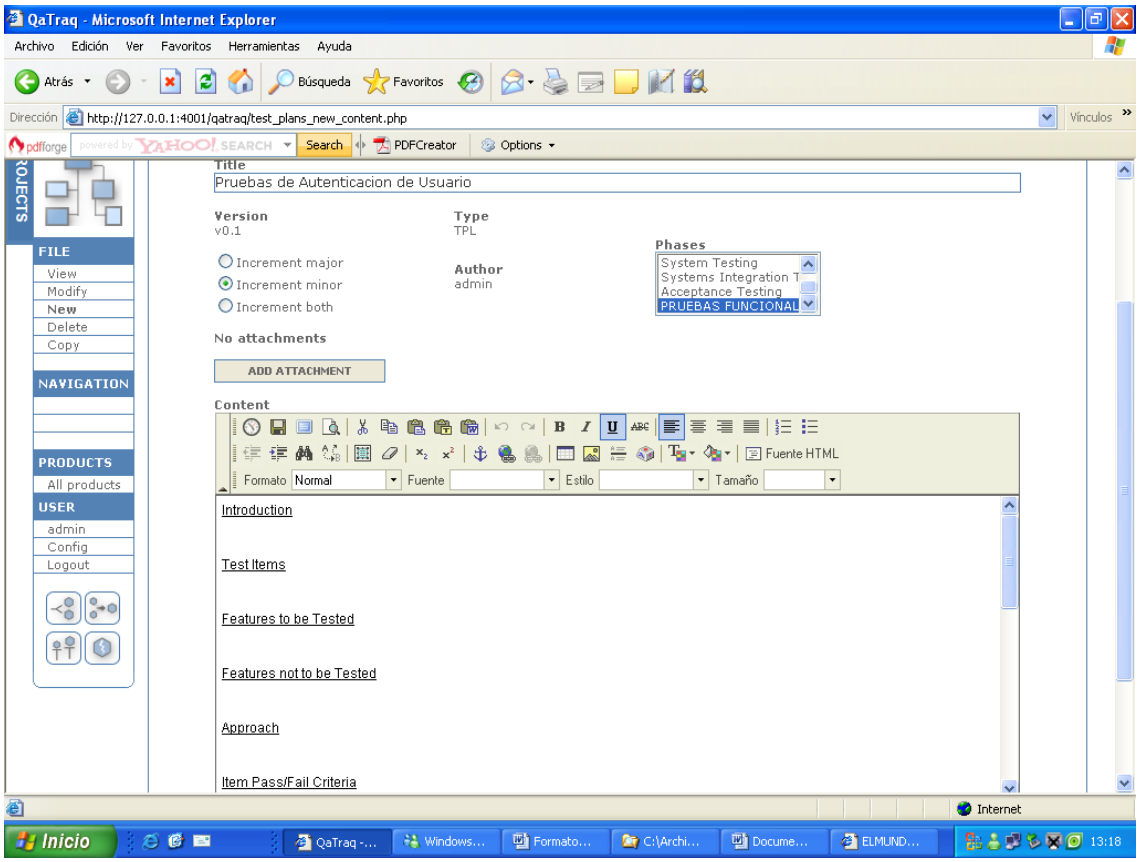


Figura 2.23 Plans

Se diligencia el título dado a este registro de plan de prueba, la fecha de creación/modificación, la Versión, el Documento ID y el Autor (o la última persona que modifica el documento). Documentos Relacionados - dentro de cada documento de plan de prueba usted puede catalogar los documentos de diseño de prueba ya realizados.

Introducción

Artículos De prueba

Rasgos para ser Probados

Rasgos para no ser Probados

Acercamiento

Artículo Pasa/Falla Criterios

Criterios de Suspensión y Exigencias de Reanudación

Prueba Deliverables

Tareas de prueba

Necesidades Ambientales

Responsabilidades

Necesidades de Staffing y training

Lista(Programa)

Riesgos y Contingencias

Aprobaciones

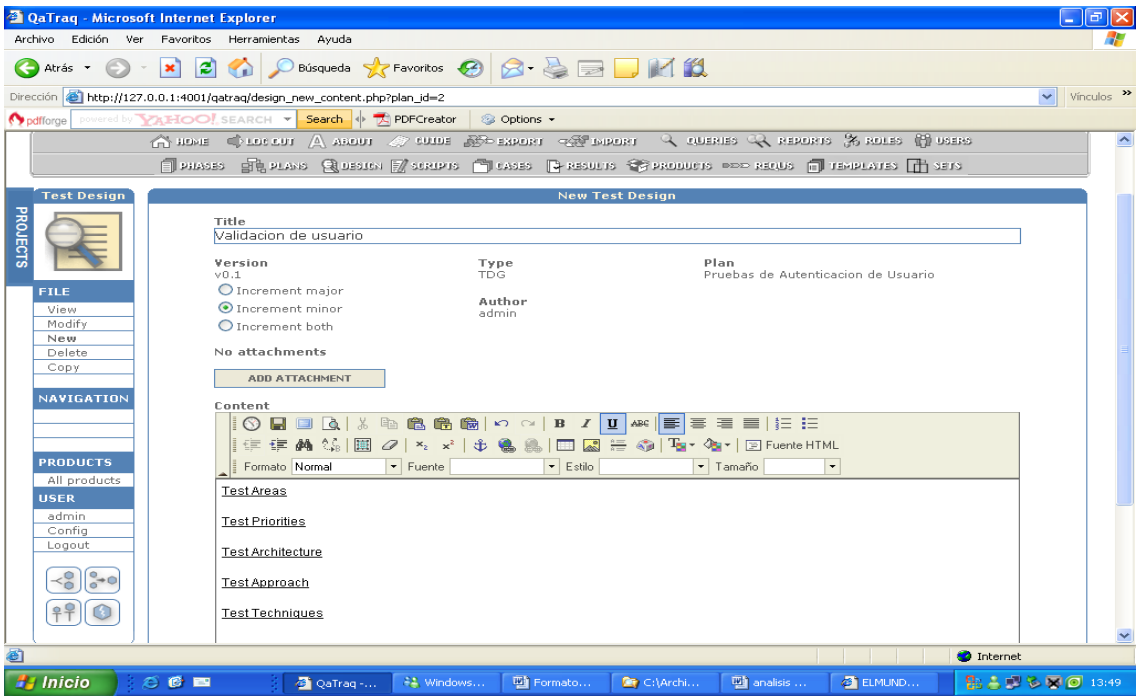


Figura 2.24 Design

DESING Cada plan de prueba puede contener uno o vario aéreas diseño de prueba. Y cada área de diseño de prueba puede incluir un número de escrituras de prueba. Por ejemplo usted podría usar un documento de diseño de prueba para

agrupar un número de escrituras de prueba que fueron asociadas con un aspecto particular de un producto.

Inicialmente se selecciona el plan de pruebas: y se crea un nuevo diseño.

Se diligencia el título dado a este registro de Diseño de prueba, la fecha de creación/modificación, la Versión, el Documento ID y el Autor (o la última persona que modifica el documento). Documentos Relacionados - dentro de cada documento de plan de prueba usted puede catalogar los documentos de diseño de prueba ya realizados. Y se documenta:

Test Areas, Test Priorities, Test Architecture, Test Approach, Test Techniques

SCRIPTS: es un documento que describe detalladamente cómo debe ser conducida una prueba, recoge los casos de prueba en grupos lógicos y los incluyen en los scripts de prueba para facilitar el seguimiento, la coordinación y la agregación de resultados de las pruebas.
Contiene información que perfila la prueba, define los prerrequisitos y un número de casos de prueba. El usuario entrará en el contorno y el texto previamente necesario y luego unirá al script de prueba un número de casos de prueba que tienen que estar contenidos.

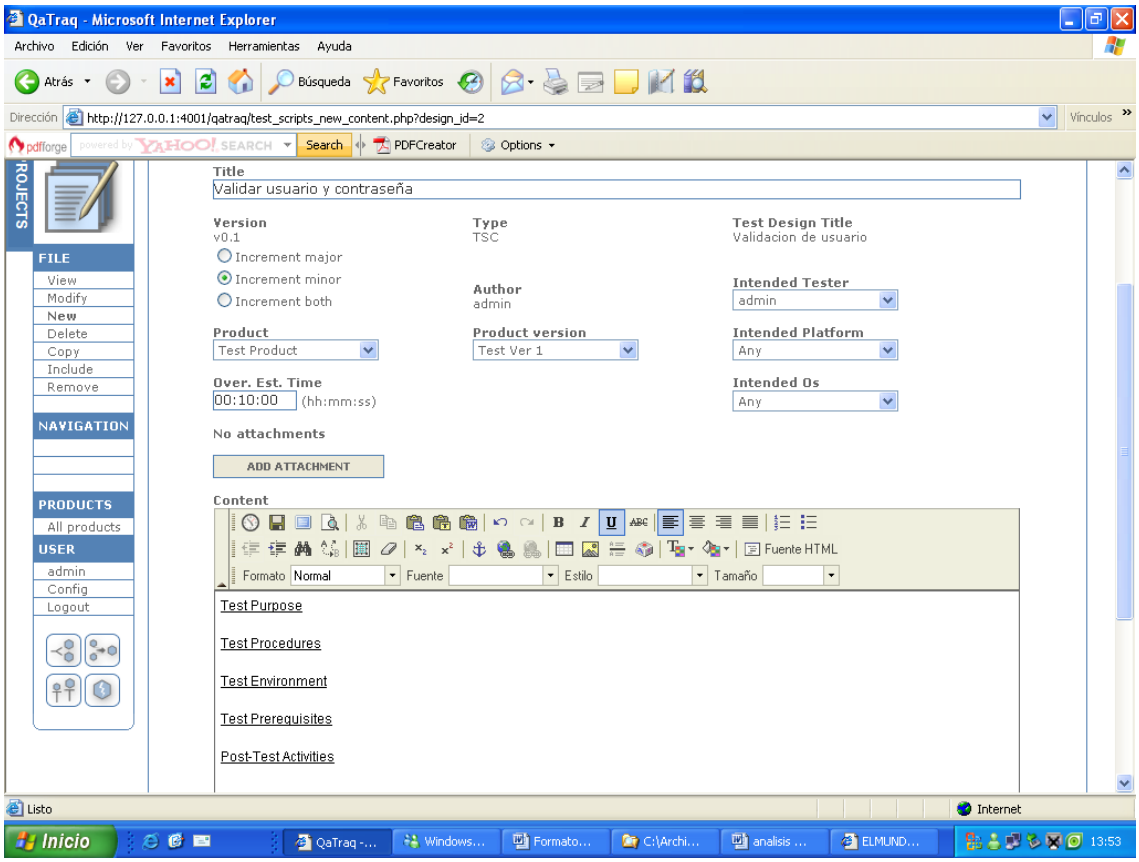


Figura 2.25 Scripts

SCRIPTS: Se selecciona el Diseño al que va a pertenecer, y definimos

1. El Título de Documento y la Información de Versión, la fecha de creación/modificación, el Documento ID y el Autor (o la última persona para modificar el documento).
2. Definir El Tester Asignado, La plataforma sobre la que se probara y el Sistema Operativo sobre el que se probara
3. Especificar: Objetivo De prueba

   Procedimientos de prueba

   Ambiente de prueba

   Requisitos previos de prueba

   Actividades Post de prueba

Listado de funcionalidades

CASOS DE PRUEBA

Un caso de prueba es una prueba específica destinada a verificar un aspecto particular de un producto bajo prueba. Cada caso de prueba debe estar relacionados con un solo producto, sino que también estar relacionados con uno o más componentes y requisitos de ese producto. Uno o más casos de prueba se incluyen dentro de scripts de prueba que se describe cómo una serie de pruebas se deben ejecutar.

Para crear un nuevo caso de prueba debe utilizar primero el servicio de búsqueda de arriba para encontrar un producto para el que desea crear el caso de prueba para. De lo contrario, puede buscar un caso de prueba existentes para ver, modificar, borrar o copia.

RESULTADO DE LA PRUEBA

Para cada caso de prueba que se incluye dentro de un script, se crea un registro de resultados de la prueba para esa combinación script/caso.

Cada página de resultados contiene:
Detalles de secuencias de comandos de pruebas
Los vínculos a cada caso de prueba
Detalles específicos de casos de prueba
Registro de resultados de pruebas.
Comentarios para cada resultado
Enlaces a los defectos asociados con cada resultado

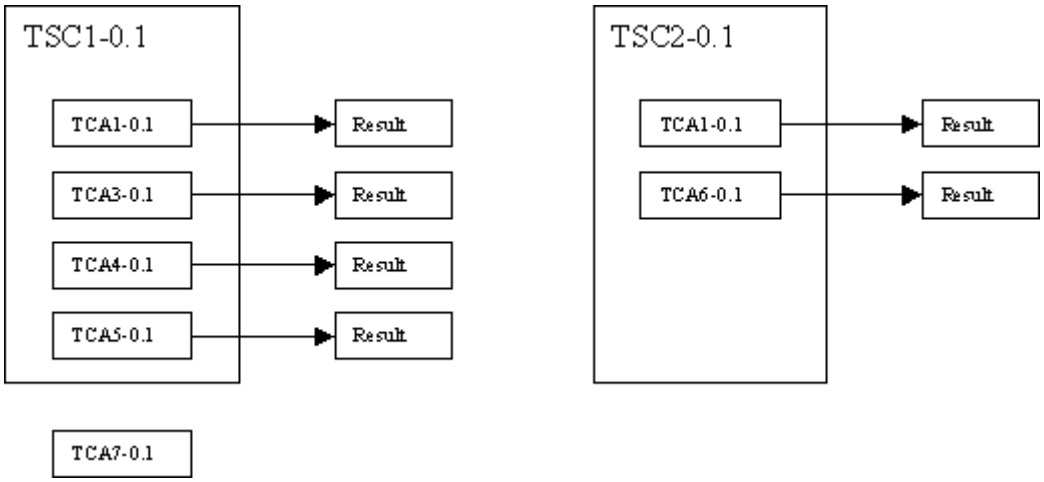


Figura 2.26 Diagrama caso prueba vs script de prueba

De esto se debe anotar:

1. Un registro de resultados de la prueba sólo se crea cuando un caso de prueba se incluye en un script de prueba (por ejemplo, de casos de prueba TCA5-0.1 que se incluye en el script de prueba TSC1-0.1 tiene un resultado de la prueba específica).
2. El mismo caso de prueba se puede incluir en scripts de prueba diferentes (por ejemplo, TCA1-0.1 que se incluye en tanto TSC1 y TSC2). Cada caso de prueba en este caso tendrá un registro de resultado de prueba único.

3. Si un caso de prueba no está incluido en un script de prueba, no habrá un resultado de prueba asociado a este (EEG TCA7-0.1).

REPORTES DE PRUEBAS Y CONSULTAS (QUERIES)

La funcionalidad de consultas (QUERIES) en QaTraq de pruebas de software de herramientas permite al usuario escribir, almacenar y ejecutar instrucciones SQL puro (lo que requiere conocimientos de SQL y la estructura QaTraq base de

datos). La otra funcionalidad de Informe ofrece a los usuarios ejecutar consultas SQL con formas predefinidas (por lo que los informes no requieren conocimientos de SQL).
La funcionalidad de Reportes se presta a la adición de nuevos informes a medida que van siendo creado por desarrolladores y usuarios.

PRODUCTOS

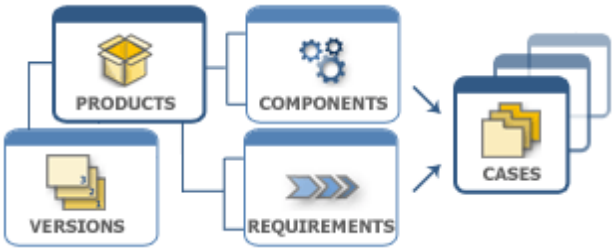


Figura 2.27 Productos

Se debe definir los Productos y sus Versiones asociadas a los resultados de las pruebas de acuerdo a la versión del producto que está bajo prueba.

Se definen los componentes y los requisitos para cada producto, lo que permite crear casos de prueba que están vinculados a los componentes y requisitos adecuados. Esto le permite elaborar informes con exactitud en los ámbitos señalados en los planes de la prueba.

Como seguimiento de defectos, la definición de sus productos, componentes, los requisitos y las versiones es la clave para ofrecerle la información que necesita para tener éxito con los proyectos de prueba (Testmanagement, 2008).

TESTLINK

- **Descripción general:** Permite fácilmente crear y gestionar casos de prueba, así como organizar los planes de prueba. Estos planes permiten a los miembros del equipo ejecutar casos de prueba y pistas de los resultados de las pruebas de forma dinámica, generar informes, traza los requisitos de software, priorizar y asignar tareas.

- **Licenciamiento:** TestLink es una herramienta basada en web bajo la licencia GPL (gratuito para su uso). El proyecto es mantenido por la comunidad abierta de Testers. TestLink hace los proceso de pruebas hace fácil y organizados.

- **Prerrequisitos:**

  La herramienta tiene una interfaz basada en web con PHP y base de datos MySQL, Postgres o MS -SQL. Colabora con sistemas de rastreo de errores conocidos como Bugzilla, Mantis, etc

- **Funcionalidades y estudio de la herramienta:**

  La estructura de esta herramienta está dada por 3 pilares: Proyecto de prueba, Plan de prueba y Usuarios

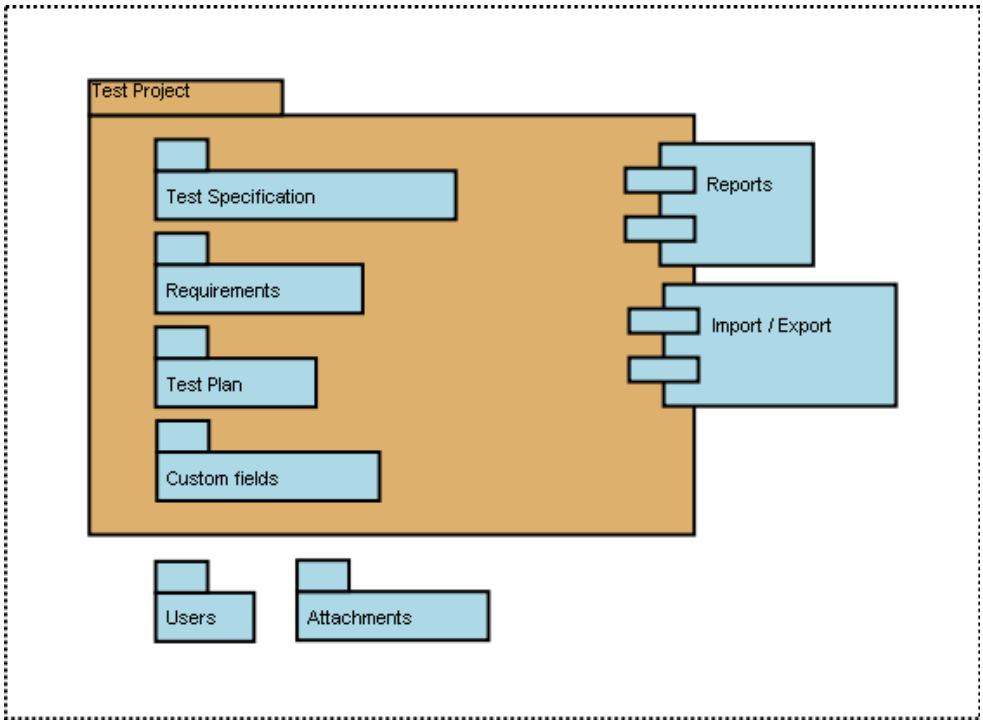


Figura 2.28 Estructura de TestLink

Casos de prueba: son la pieza fundamental en Testlink, estos describen una tarea de prueba a través de los pasos (acciones y escenarios) y los resultados esperados.

Test Suite: organiza los casos de prueba en unidades. Esta estructura de prueba específica en partes lógicas.

Plan de prueba: Es creado cuando se desea ejecutar un caso de prueba, pueden estar compuestos por los casos de prueba de proyecto actual, incluye hitos, asignaciones de usuarios, y resultados de la prueba.

Proyecto de prueba: es algo que existirá siempre en testlink, el proyecto se someterá a muchas versiones diferentes durante su vida útil. Este incluye pruebas

con especificación de casos de prueba, requisitos y palabras clave. Los usuarios dentro del proyecto deben tener definidas sus funciones.

Usuario: cada uno tiene un rol que define las características disponibles en TestLink (Testlink, 2008).

Flujo de trabajo

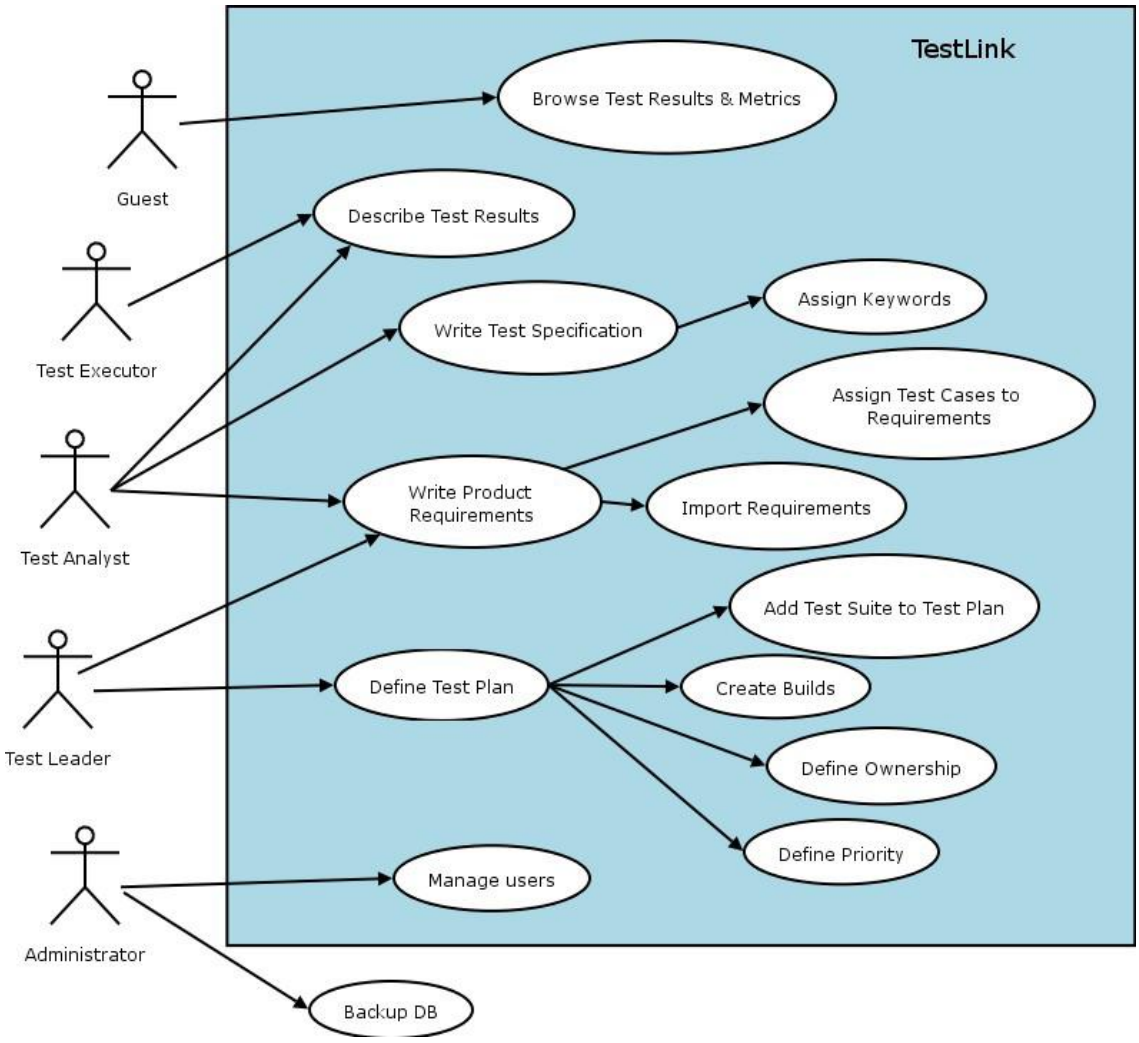


Figura 2.29 Flujo trabajo TestLink

Ejemplo de un flujo de trabajo sencillo de TestLink

1. El Administrador crea un proyecto de prueba "comidas rápidas" y dos usuarios, a Adam con los derechos de "Líder" y Bela con los derechos de "Senior Tester".

2. El Líder Adam importa los Requisitos de software y para una parte de estos requisitos genera de casos de prueba vacíos. Los reorganizarse en dos Test Suites: "Peces" y "Chips".

3. El Tester Bela describe un escenario de prueba (crear un contenido de casos de prueba vacíos) con estas especificación de prueba que se organiza en Test Suites.

4. Adam crea la Keyword "Las pruebas de regresión" y asigna a esta Keyword a diez de estos casos de prueba.

5. Adam crea un plan de pruebas "Peces & Chips 1", construye "Pescado 0,1" y vincula a todos los Casos de Prueba en el Test Suite "Peces" a este plan de pruebas. Se asigna a sí mismo y a Bela este plan de pruebas también.

6. Ahora, los desarrolladores desarrollan una primera construcción. Adam y Bela ejecutan y registran los resultados de las pruebas: 5 pasados, 1 fracasado y 4 bloqueado.

7. Los desarrolladores hacer una nueva versión "peces 0,2" y Bela prueba únicamente el caso de prueba fallido y bloqueado. Esta vez, todos estos casos bloqueados y fallidos pasan. Se reprueba también todos los Casos de Prueba con la Keyword "Las pruebas de regresión".

8. Un administrador de este equipo le gustaría ver los resultados. Administrador le explica que él puede crear una cuenta a sí mismo en la página de inicio de sesión. Manager lo hace. Tiene por defecto "Invitado" los derechos y pueden ver resultados de la prueba y casos de prueba. Él puede ver que todo lo que pasa en el informe global, y problemas en la construcción "de peces 0,1" en un informe para ese desarrollo en particular.

9. Más tarde, los desarrolladores por último añaden también la funcionalidad "Chips". Adam crea un plan de pruebas "Peces & Chips 2". Se puede reutilizar el primer plan de pruebas como plantilla. Todos los casos de pruebas “Peces" y las funciones se añadirán automáticamente. El crear una nueva "Peces 1,1" y vincula a todos los casos de prueba "chips" a este plan de pruebas también.

10. Ahora comiencen las pruebas como de costumbre.

11. Más tarde, el Administrador crea un nuevo proyecto de prueba para otro producto "Hot Dog". Pero este es otro equipo de pruebas y una historia diferente.

CASE MAKER

- **Descripción general:** Es una herramienta de diseño de caso de prueba y de generación de datos de prueba que cabe en un lugar único entre las necesidades y herramientas de gestión de pruebas y herramientas ejecución de la prueba. Gracias a su interfaz abierta, CaseMaker puede trabajar con toda la gestión de gran prueba y herramientas de generación automática ejecución de la prueba.

  CaseMaker le ahorrará tiempo, dinero y esfuerzo, ya que utiliza metodologías reconocidas de prueba y un enfoque basado en el riesgo de generar pruebas menos numerosas y más eficaces. CaseMaker mejorará

su cobertura de la prueba y le dará confianza en la calidad de las soluciones de software que ofrecen.

Está disponible en 3 idiomas: Inglés, Alemán y Español.

- **Licenciamiento:** El licenciamiento tiene un costo dependiendo del número de licencias y si son empresa a personas independientes, adicional se debe de pagar anualmente un porcentaje del valor de la licencia por mantenimiento y actualizaciones.

- **Prerrequisitos:** Soporta sistemas operativos Windows y Linux/Unix, está desarrollado en Java requiriendo JDK 1.4.2 o superior.

- **Funcionalidades y estudio de la herramienta:**

Esta herramienta está compuesta de 5 módulos:

– Business Rules: En este módulo se definen las reglas de transacciones de negocios. Las normas siguen la descripción del Grupo de Reglas de Negocio. Apoyo a las otras descripciones de reglas de negocio se está desarrollando actualmente. Los casos de prueba, así como los datos de prueba pueden generarse a partir de las reglas de negocio.

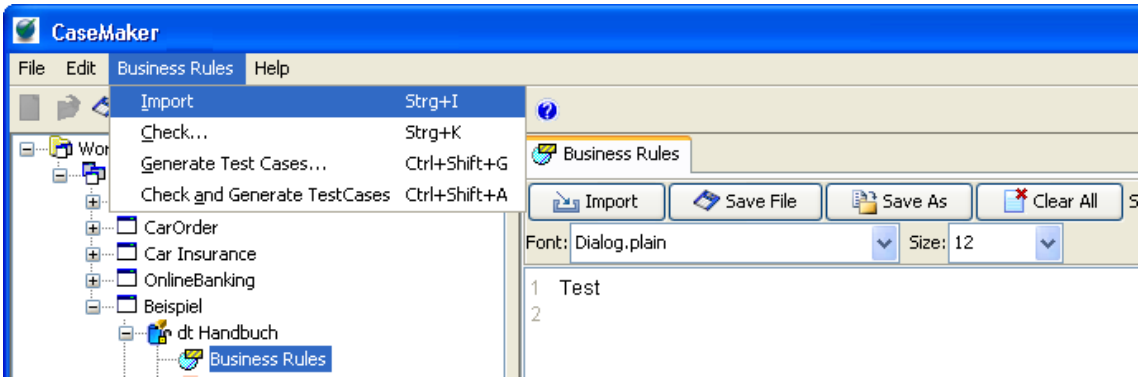


Figura 2.30 Business Rules

- Casos de pruebas: este es el modulo básico de CaseMaker. Soporta métodos probados para reducir y priorizar el número de casos de prueba al mismo tiempo alcanzar una cobertura alta.
- Los casos de prueba representa el núcleo principal del programa. Aquí es donde los elementos y clases de equivalencia se crean, las dependencias y combinaciones formadas y casos de prueba generados.

  El área de Casos de prueba se divide en cinco áreas de trabajo diferentes, cada uno de los cuales tiene sus propios comandos de menú. Las cinco áreas comprenden: Elementos y clases de equivalencia, Causa efectos, Dependencias y combinaciones, Casos de prueba, Norma combinaciones.

  La lista de casos de prueba se genera en HTML y se pueden ordenar por casos de prueba o por clases de riesgo.

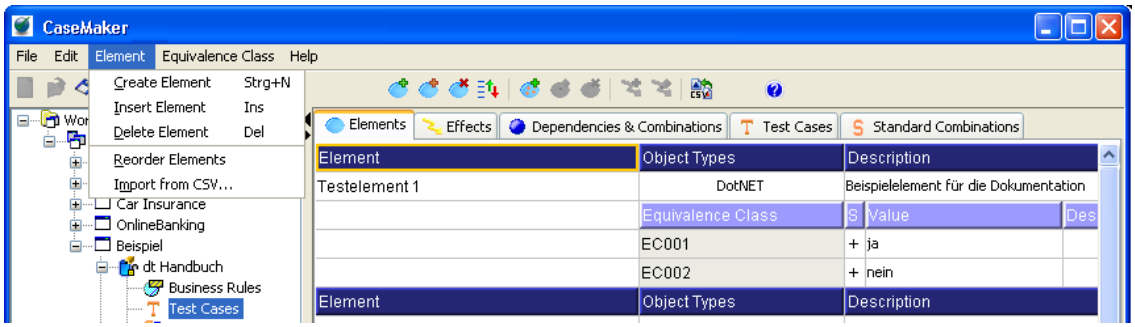


Figura 2.31 Test Cases Case Maker

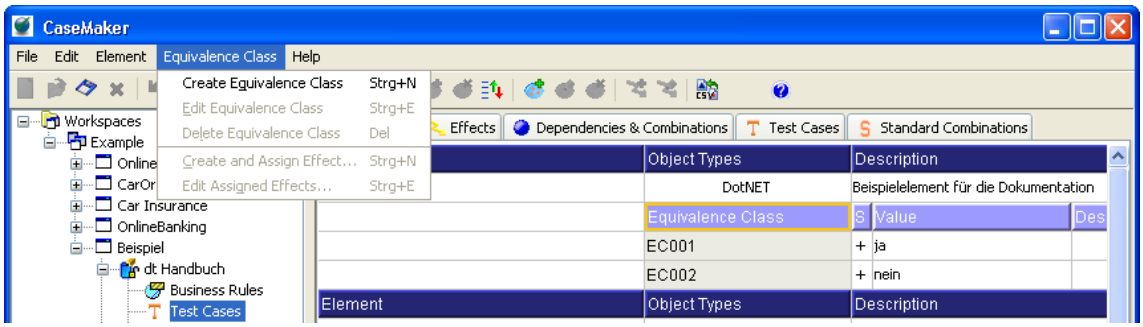


Figura 2.32 Clases de equivalencia Case Maker

- Datos de pruebas: Las estructuras de datos se pueden generar automáticamente a partir de los casos de prueba, o de forma manual. Estas estructuras forman la base para la definición de datos de prueba y la definición de resultados actual/esperados. Fórmulas o variables se pueden asignar a los campos de las estructuras para la generación automática de datos de prueba.

  Los datos requeridos para las pruebas y para comparación actual/esperada pueden ser importados a partir de datos existentes a través de una interfaz de CSV o generados por CaseMaker.

  Los datos son exportados a través de los archivos XSL, con el resultado que prácticamente cualquier formato de datos se pueden generar. Los archivos XSL puede ser asignado al proyecto respectivo.

  Este modulo se subdivide en 4 áreas:

  Structure

  Test Data

  Test Data Set

  Variables

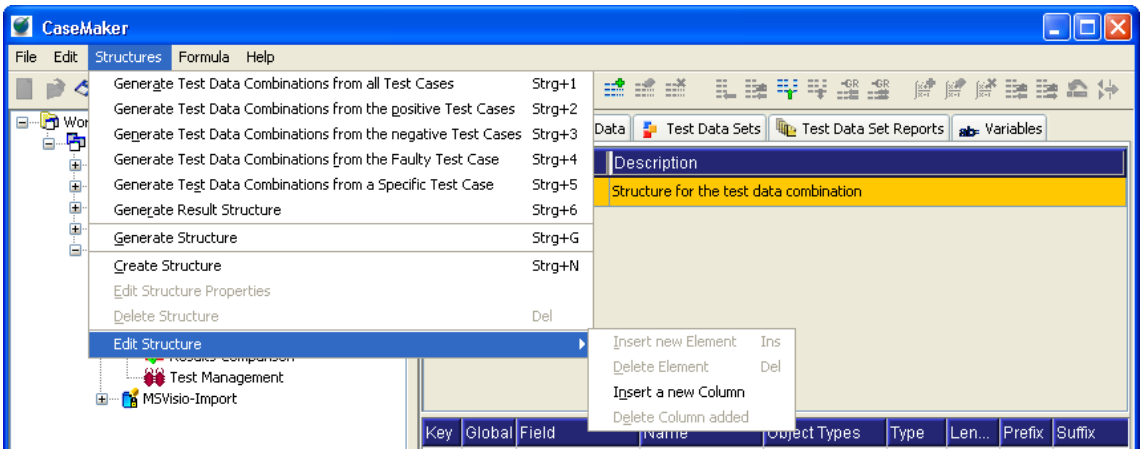


Figura 2.33 Editar estructura Case Maker

- Resultados de la comparación: Este módulo se utiliza para comparar los resultados reales y esperados, que se definen de forma manual o mediante la interfaz de importación establecida en CaseMaker. La comparación se lleva a cabo a nivel de campo de datos.
- El informe de comparación se genera como un archivo HTML

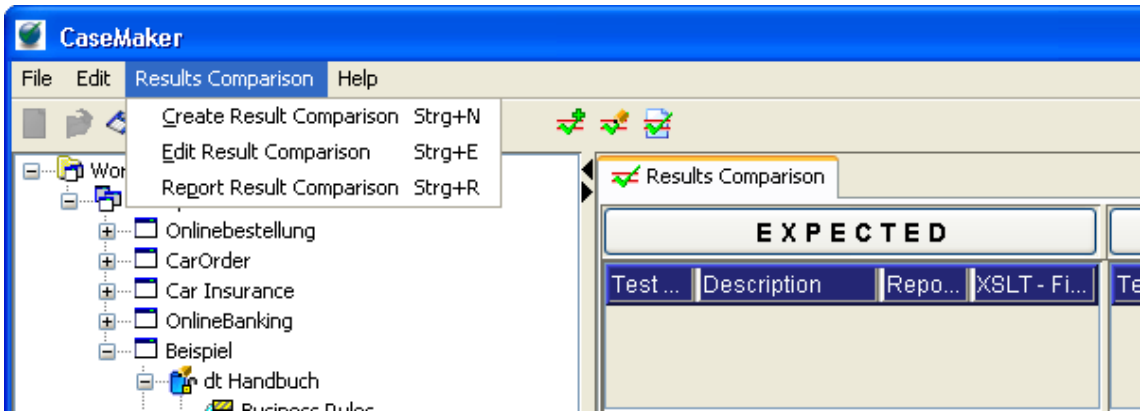


Figura 2.34 Results Comparison Case Maker

- Administrador de pruebas: Esto permite el registro y gestión de errores. Cada fallo se registra de forma individual y contiene la información necesaria para darles seguimiento. Varios de los casos de prueba también

se pueden asignar a los errores individuales. Esto da lugar a un manejo más eficiente y manipulación de errores, y permite prueba especificas que se aplicaron tras la depuración (CaseMaker International Ltd, a Diaz & Hilterscheid Group Company, Incorporated in England & Wales, Registered No. 5464015.).

3. SOLUCIÓN PROPUESTA

A continuación se presenta una evaluación de las herramientas analizadas, de acuerdo a si cumplen o no con unos criterios preestablecidos de selección, los cuales debería soportar la herramienta que se va a seleccionar para la propuesta. Los criterios enunciados en la tabla 4.2 se establecieron teniendo en cuenta las necesidades de la empresa Choucair Testing, la cual requiere de una herramienta que facilite la gestión de las pruebas funcionales, mediante la creación de múltiples proyectos, el acceso a los proyectos de acuerdo al rol, la vinculación de documentos externos y la posibilidad de realizar seguimiento a cada ítem especificado en el caso de pruebas. Así como tener la información de cada proyecto de modo centralizado y a su vez poder extraer información que sea relevante para la toma de decisiones.

Criterios	QATRAQ	TESTLINK	QUALIFY 2.4	CASEMAKER
Creación de múltiples proyectos	SI	SI	SI	SI
Gestión de usuarios	SI	SI	SI	
Creación de nuevos roles	SI	SI	SI	NO
Soporte para diversos tipos de prueba (unidad, integración, sistema, aceptación)	SI	SI	SI	SI
Facilidad de importación de requerimientos	SI	SI	SI	SI
Enlace con requisitos	SI	NO	SI	SI
Herramienta gratuita	NO	SI	NO	NO
Manejo de reportes	SI	SI	SI	SI
Modificación de Reportes	SI	SI	NO	NO
Facilidad de instalación	SI	SI	NO	SI
Evaluación de los pasos de prueba independiente de los casos de prueba.	NO	NO	SI	NO

Tabla 3.1 Criterios de comparación herramientas funcionales

La mayoría de los criterios de selección son soportados por todas la herramientas, lo que indica que son muy homogéneas en cuanto a su funcionalidad como la posibilidad de crear y administrar diversos proyectos a la vez, soportan diversos tipos de pruebas en diferentes etapas de desarrollo, posibilitan la generación de reportes para ver los resultados de las pruebas. Sus diferencias se encuentran básicamente en el flujo de trabajo normal de cada una de las herramientas y la terminología que utilizan para sus funcionalidades, otras diferencias entre una y otra herramienta que es importante tener en cuenta a la hora de elegirla, es el tipo de licenciamiento y la facilidad del manejo de la herramienta. Sin embargo, para nuestro caso de estudio y la necesidad que tiene la empresa de Testing para la cual se hace la propuesta, es que la herramienta permita evaluar de forma independiente los pasos que forman los casos de prueba.

Teniendo en cuenta que de las herramientas estudiadas, la que cumple con los criterios más relevantes para la empresa Choucair Testing es Qualify 2.4. Esta será la herramienta que se propone en esta investigación para el apoyo en la documentación de los casos de prueba funcionales. Algunos de los criterios que llevo a tomar la decisión de elegir esta herramienta son:

- Permite la evaluación de cada paso descrito en los casos de prueba.

- Si los reportes que arroja la herramienta necesitan ser modificados, esto no es posible, sin embargo, se puede crear cualquier informe con el uso de SQL Server Reporting Services, que es una herramienta que viene libre con todas las versiones de SQL Server.

- La herramienta Qualify 2.4, cuenta con un equipo de soporte que dentro de sus servicios profesionales ofrecen el diseño de reportes personalizados y algunas personalizaciones de la herramienta, con el fin de adaptar la herramienta según las necesidades del cliente.

- Por ser una herramienta licenciada tiene el apoyo de todo un equipo que brindan ayuda desde la instalación, hasta la puesta en marcha y ejecución de la misma.

Esta herramienta que se recomienda en esta propuesta ejecuta los siguientes reportes:

- Informe de Prueba. Este informe muestra en forma de barras la cobertura de de los pasos en una prueba, cobertura de los casos de prueba, escenarios de cubiertos, ver figura 3.1.

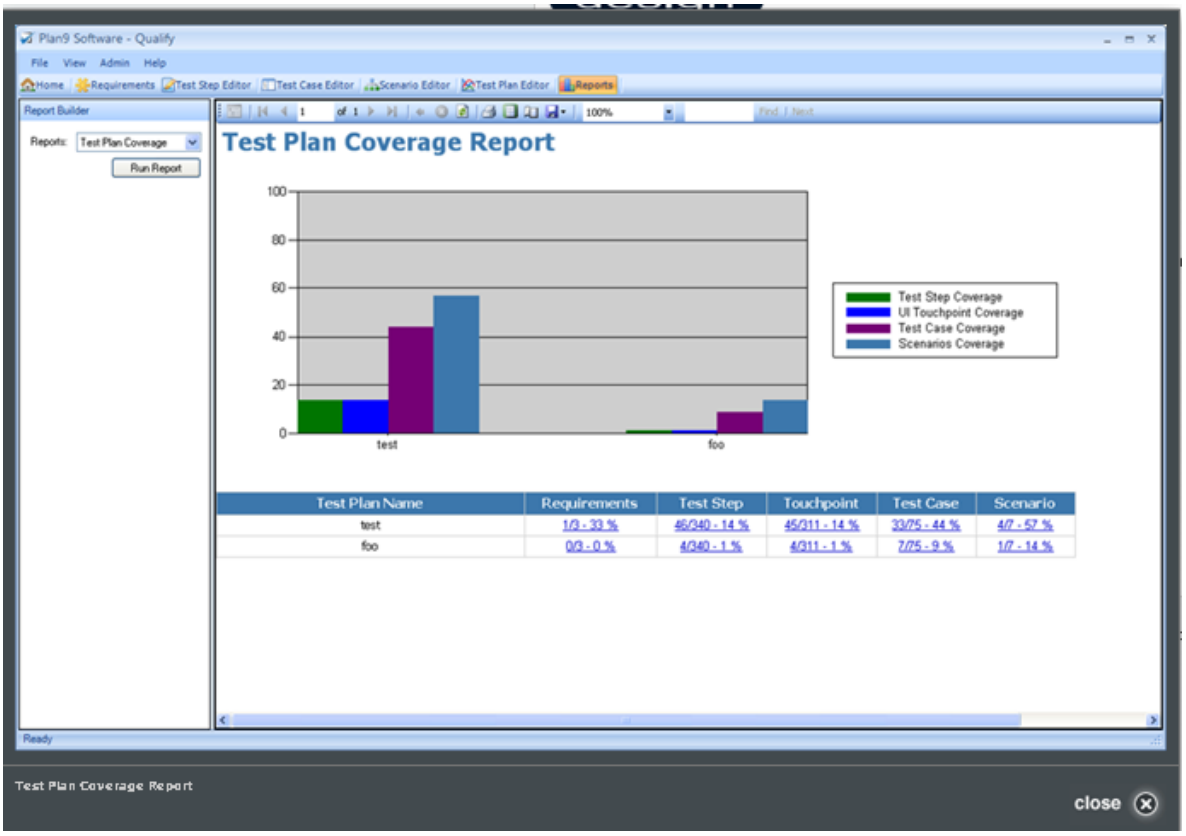


Figura 3.1 Reporte Cobertura Plan Pruebas

- Cobertura del plan de pruebas por Touchpoint (Característica que tiene la herramienta Qualify 2.4 para realizar pruebas por puntos específicos en una imagen, pantalla o interfaz). Este informe permite visualizar el porcentaje de cobertura de por plan. Ver figura 3.2.

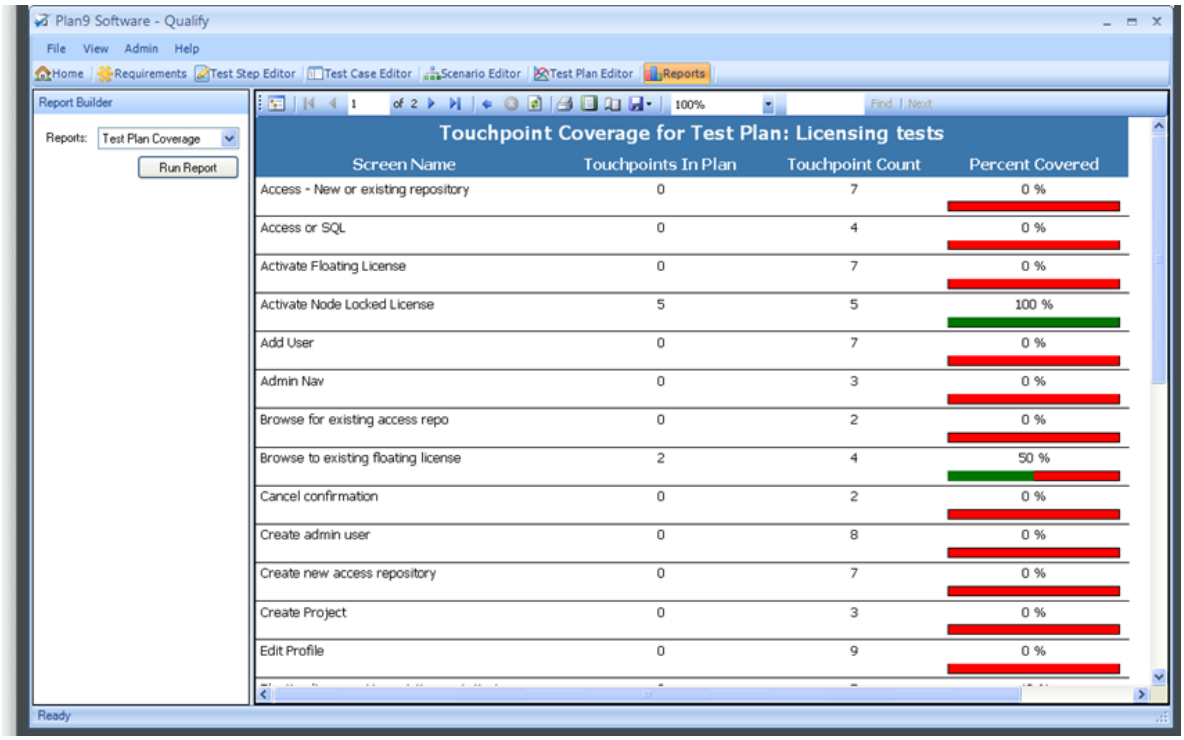
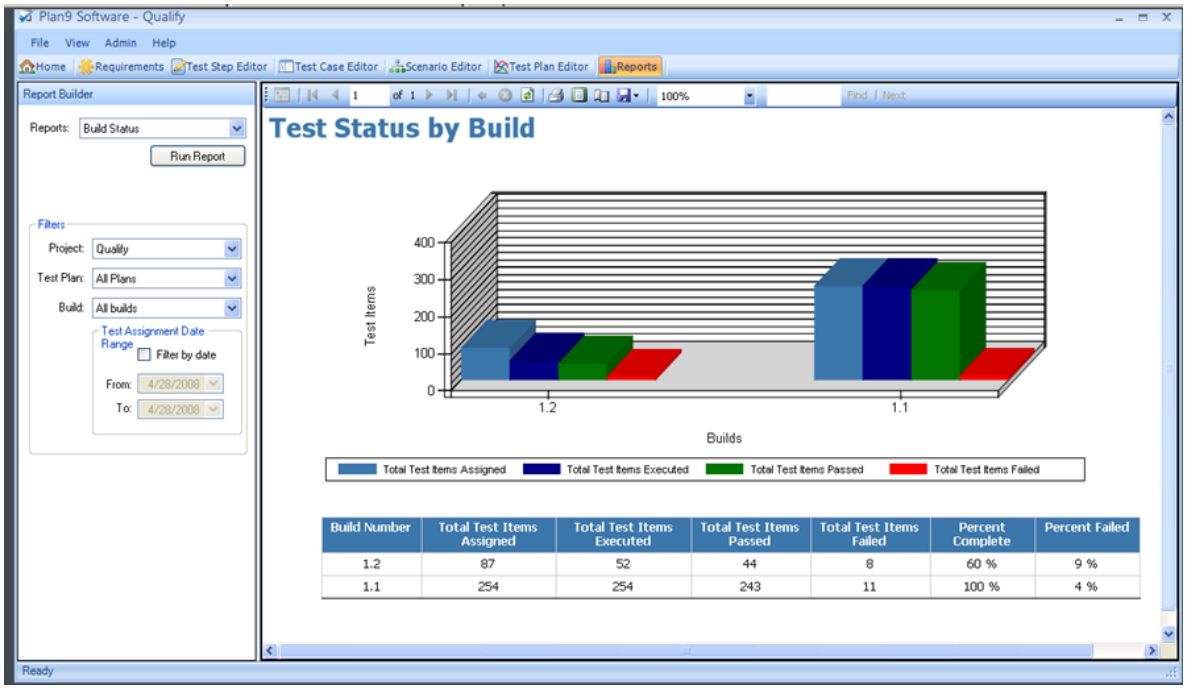


Figura 3.2 Cobertura del plan de pruebas por Touchpoint

- Estado del plan de pruebas por construir. Este informe de modo global muestra el total de ítems asignados, ejecutados, exitosos y fallidos en una prueba. Estos resultados globales los arroja de acuerdo a unos filtros como podemos observar en la figura 3.3.



Build Number	Total Test Items Assigned	Total Test Items Executed	Total Test Items Passed	Total Test Items Failed	Percent Complete	Percent Failed
1.2	87	52	44	8	60 %	9 %
1.1	254	254	243	11	100 %	4 %

Figura 3.3 Estado del plan de pruebas por construir

- Informe para imprimir plan de pruebas. El informe describe los casos de prueba, los pasos por cada prueba de acuerdo a cada escenario. Ver figura 3.4.

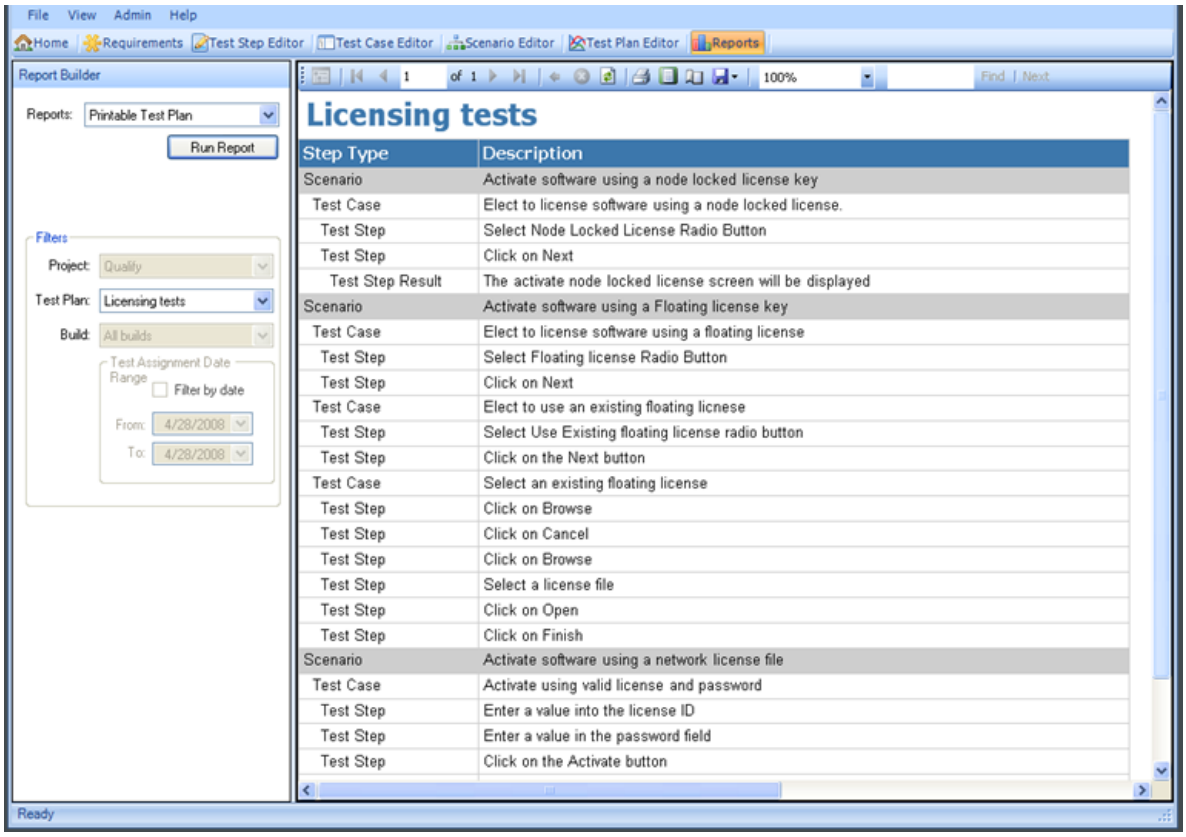


Figura 3.4 Informe plan pruebas

Actualmente la empresa Choucair Testing, con base en la información recolectada de la creación y ejecución de los casos de prueba, genera una serie de reportes muy personalizados que le ayuda a tener información a cerca de:

- La cantidad de los casos de prueba por cada proyecto.
- Cantidad de pasos de prueba.
- La cantidad de pasos de pruebas ejecutados y por ejecutar.
- Número de pasos de prueba fallados y número sin fallos.
- Porcentaje de esfuerzo, el cual se calcula teniendo en cuenta el numero de pasos ejecutados/Número total de pasos por corrida.
- Avance de los casos, que se calcula así:
  Número de casos OK / número total de casos
- Porcentaje ejecutado ok en cada corrida.
- Número de bugs encontrados.

Teniendo en cuenta que actualmente la empresa trabaja sobre plantillas en Excel en las cuales se almacena la información de un caso de prueba en particular, es relativamente sencillo tomar información de ese caso de prueba de manera independiente, pero al momento de sacar un consolidado podría tornarse más complejo al tener que reunir todos los datos; en un momento dado que se pueda

contar con una herramienta como Qualify 2.4, en la cual todos los datos están centralizados en una base de datos y con el fin de sacar un máximo provecho a la herramienta, se recomienda una serie de reportes que se puedan ejecutar de manera particular y consolidado. A continuación se hace una descripción del reporte y como quedaría gráficamente:

- **Reporte número 1: Estadístico de pruebas por Tester:** este reporte permitiría visualizar como es el avance de los tester de forma individual y en comparación con el global de la compañía en relación a los casos de prueba que le han sido asignados, cuántos de estos ha ejecutado, cuantos pasos de prueba están involucrados en estos casos de prueba, cuántos de estos ya están ejecutados y cuantos no, y cuál ha sido el resultado encontrado por cada uno. Y al final se imprimirá un consolidado de la empresa con cada uno de los ítems anteriores y un grafico que muestre cual ha sido el desempeño del tester con relación a sus demás compañeros y a la empresa en general, el cual estará filtrado por un periodo de tiempo determinado.

Reporte Número 1. Estadístico de pruebas por Tester
Periodo comprendido entre 01 de Julio de 2010 y 31 de Julio de 2010

Tester A

	CRITERIO	VALOR
1	NÚMERO DE CASOS DE PRUEBA ASIGNADOS	20
2	NÚMERO DE CASOS DE PRUEBA EJECUTADOS	15
3	TOTAL DE PASOS DE PRUEBAS	80
4	PASOS DE PRUEBAS EJECUTADOS	62
5	PASOS DE PRUEBA NO EJECUTADOS	18
6	PASOS DE PRUEBA QUE FALLARON	10
7	PASOS DE PRUEBAS QUE NO FALLARON	52
8	NUMERO DE BUGS ENCONTRADOS	5
9	AVANCE POR CASOS DE PRUEBA (CRITERIO 2/ CRITERIO 1)	75%
10	AVANCE POR PASOS DE PRUEBA (CRITERIO 4/CRITERIO 3)	77.5%

Tabla 3.2 Estadístico pruebas por Tester A

Tester B

	CRITERIO	VALOR
1	NÚMERO DE CASOS DE PRUEBA ASIGNADOS	18
2	NÚMERO DE CASOS DE PRUEBA EJECUTADOS	12
3	TOTAL DE PASOS DE PRUEBAS	90
4	PASOS DE PRUEBAS EJECUTADOS	75
5	PASOS DE PRUEBAS NO EJECUTADOS	15
6	PASOS DE PRUEBA QUE FALLARON	7
7	PASOS DE PRUEBAS QUE NO FALLARON	68
8	NUMERO DE BUGS ENCONTRADOS	8
9	AVANCE POR CASOS DE PRUEBA (CRITERIO 2/ CRITERIO 1)	66.66%
10	AVANCE POR PASOS DE PRUEBA (CRITERIO 4/CRITERIO 3)	83.33%

Tabla 3.3 Estadístico pruebas por Tester B

Total general

	CRITERIO	VALOR
1	NUMERO DE TESTER	2
2	NÚMERO DE CASOS DE PRUEBA ASIGNADOS	38
3	NÚMERO DE CASOS DE PRUEBA EJECUTADOS	27
4	TOTAL DE PASOS DE PRUEBAS	170
5	PASOS DE PRUEBAS EJECUTADOS	137
6	PASOS DE PRUEBAS NO EJECUTADOS	33
7	PASOS DE PRUEBA QUE FALLARON	17
8	PASOS DE PRUEBAS QUE NO FALLARON	120
9	NUMERO DE BUGS ENCONTRADOS	13
10	AVANCE POR CASOS DE PRUEBA (CRITERIO 2/ CRITERIO 1)	71.05%
11	AVANCE POR PASOS DE PRUEBA (CRITERIO 4/CRITERIO 3)	80.58%

Tabla 3.4 Estadístico pruebas consolidado

Gráfico

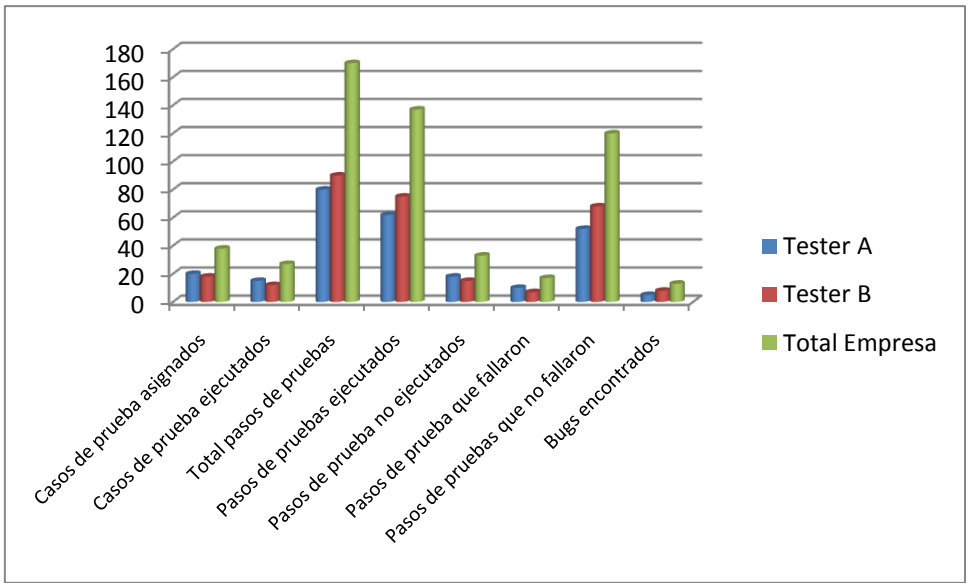


Figura 3.5 Gráfico estadístico pruebas consolidado

- **Reporte Número 2. Estadístico de pruebas por Proyecto**: este reporte tiene los mismos ítems del reporte anterior pero a nivel de los proyectos, permitiendo tener una visión de cuál es el estado de cada uno de los proyectos y poder compararlos con los cronogramas y de esta manera si se amerita, poder aplicar acciones que permitan cumplir con el cronograma establecido, así como determinar que proyectos están necesitando mayor recursos dado la cantidad de casos de pruebas y pasos de estos, cuales han presentado mayor número de fallas con respecto al global de la empresa, igualmente en un periodo de tiempo determinado.

Reporte Número 2. Estadístico de pruebas por Proyectos
Periodo comprendido entre 01 de Julio de 2010 y 31 de Julio de 2010

Proyecto 1

	CRITERIO	VALOR
1	NÚMERO DE CASOS DE PRUEBA CREADOS	10
2	NÚMERO DE CASOS DE PRUEBA EJECUTADOS	10
3	TOTAL DE PASOS DE PRUEBAS	50
4	PASOS DE PRUEBAS EJECUTADOS	50
5	PASOS DE PRUEBA NO EJECUTADOS	0
6	PASOS DE PRUEBA QUE FALLARON	7
7	PASOS DE PRUEBAS QUE NO FALLARON	43
8	NUMERO DE BUGS ENCONTRADOS	2
9	AVANCE DEL PROYECTO POR CASOS DE PRUEBA (CRITERIO 2/ CRITERIO 1)	100%
10	AVANCE DEL PROYECTO POR PASOS DE PRUEBA (CRITERIO 4/CRITERIO 3)	100%

Tabla 3.5 Estadístico pruebas por proyecto 1

Proyecto 2

	CRITERIO	VALOR
1	NÚMERO DE CASOS DE PRUEBA CREADOS	15
2	NÚMERO DE CASOS DE PRUEBA EJECUTADOS	9
3	TOTAL DE PASOS DE PRUEBAS	74
4	PASOS DE PRUEBAS EJECUTADOS	58
5	PASOS DE PRUEBA NO EJECUTADOS	16
6	PASOS DE PRUEBA QUE FALLARON	4
7	PASOS DE PRUEBAS QUE NO FALLARON	54
8	NUMERO DE BUGS ENCONTRADOS	6
9	AVANCE DEL PROYECTO POR CASOS DE PRUEBA (CRITERIO 2/ CRITERIO 1)	60%
10	AVANCE DEL PROYECTO POR PASOS DE PRUEBA (CRITERIO 4/CRITERIO 3)	78.37%

Tabla 3.6 Estadístico pruebas por proyecto 2

Proyecto 3

	CRITERIO	VALOR
1	NÚMERO DE CASOS DE PRUEBA CREADOS	13
2	NÚMERO DE CASOS DE PRUEBA EJECUTADOS	8
3	TOTAL DE PASOS DE PRUEBAS	46
4	PASOS DE PRUEBAS EJECUTADOS	29
5	PASOS DE PRUEBA NO EJECUTADOS	17
6	PASOS DE PRUEBA QUE FALLARON	6
7	PASOS DE PRUEBAS QUE NO FALLARON	23
8	NUMERO DE BUGS ENCONTRADOS	5
9	AVANCE DEL PROYECTO POR CASOS DE PRUEBA (CRITERIO 2/ CRITERIO 1)	61.53%
10	AVANCE DEL PROYECTO POR PASOS DE PRUEBA (CRITERIO 4/CRITERIO 3)	63.04%

Tabla 3.7 Estadístico pruebas por proyecto 3

Total general

	CRITERIO	VALOR
1	NUMERO DE PROYECTOS EN EL PERIODO	3
2	NÚMERO DE CASOS DE PRUEBA CREADOS	38
3	NÚMERO DE CASOS DE PRUEBA EJECUTADOS	27
4	TOTAL DE PASOS DE PRUEBAS	170
5	PASOS DE PRUEBAS EJECUTADOS	137
6	PASOS DE PRUEBA NO EJECUTADOS	33
7	PASOS DE PRUEBA QUE FALLARON	17
8	PASOS DE PRUEBAS QUE NO FALLARON	120
9	NUMERO DE BUGS ENCONTRADOS	13
10	AVANCE DEL PROYECTO POR CASOS DE PRUEBA (CRITERIO 2/ CRITERIO 1)	71.05%
11	AVANCE DEL PROYECTO POR PASOS DE PRUEBA (CRITERIO 4/CRITERIO 3)	80.58%

Tabla 3.8 Estadístico pruebas consolidado proyectos

Gráfico

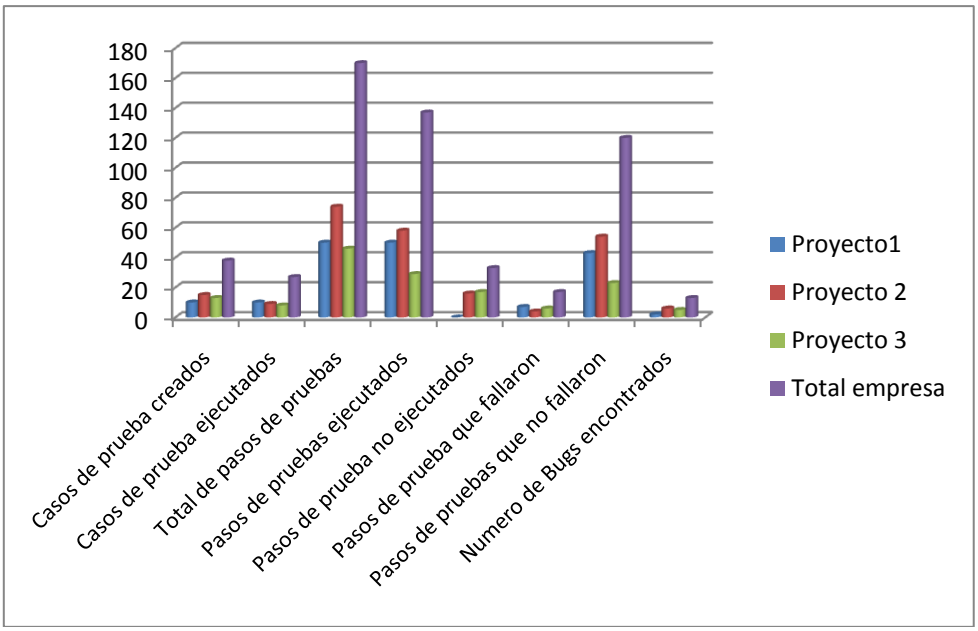


Figura 3.6 Gráfico estadístico pruebas consolidado proyectos

- Reporte Número 3

Dado que estos reportes propuestos son muy propios de la empresa, es necesarios realizar unas personalización que son posibles en esta herramienta Qualify 2.4, a través de SQL Server 2005 Reporting Services; este componente que hace parte del SQL Server 2005 proporciona funciones de generación de informes de ámbito empresarial habilitados para Web. En resumen podemos mencionar las principales funcionalidades que presenta el configurador de reportes: crear informes que extraen contenido de distintos orígenes de datos, publicar informes en diferentes formatos y administrar centralmente la seguridad y las suscripciones. Crea informes sencillos, agrega funciones de agrupación, ordenación y aplicación de formato, crea modelos de informes, usa consultas dinámicas, crea suscripciones controladas por datos, obtiene acceso a informes a través de servicios web y crea informes de forma programática a través del lenguaje de definición de informes (RDL, Report Definition Language) (msdn).

Para iniciar la herramienta de configuración de Reporting Services :

1. Haga clic en Inicio, elija Todos los programas, Microsoft SQL Server 2005, Herramientas de configuración y, finalmente, haga clic en Configuración de Reporting Services.

Se abrirá el cuadro de diálogo Selección de instancia de instalación del servidor de informes para que seleccione la instancia del servidor de informes que desee configurar.

2. En Nombre del equipo, especifique el nombre del equipo en el que está instalada la instancia del servidor de informes. De manera predeterminada aparece el nombre del equipo local, pero también puede escribir el nombre de una instancia de SQL Server remota.

   Si especifica un equipo remoto, haga clic en Buscar para establecer una conexión. Previamente, debe haber configurado el servidor de informes para la administración remota. Para obtener más información, vea Configurar un servidor de informes para la administración remota.

3. En Nombre de instancia, elija la instancia de SQL Server 2005 Reporting Services que desee configurar. En la lista sólo aparecen instancias del servidor de informes de SQL Server 2005. No es posible configurar versiones anteriores de Reporting Services.
4. Haga clic en Conectar.

5. Para comprobar que se haya iniciado la herramienta, compare sus resultados con los de la siguiente figura 4.1.

Figura 3.7 Administrador de configuración de Reporting Services

Ejemplo de cómo definir una consulta Transact-SQL para datos de informe (msdn):

1. Escriba, o copie y pegue, la siguiente consulta en el panel SQL del diseñador de consultas genérico. El panel SQL es el panel superior de la herramienta de diseño. El diagrama que aparece a continuación de estos pasos muestra el lugar donde se debe especificar la consulta.

   Copiar

   SELECT S.OrderDate, S.SalesOrderNumber, S.TotalDue, C.FirstName, C.LastName FROM HumanResources.Employee E INNER JOIN Person.Contact C ON E.ContactID = C.ContactID INNER JOIN Sales.SalesOrderHeader S ON E.EmployeeID = S.SalesPersonID

2. Para ver los resultados de la consulta, haga clic en el botón Ejecutar (!) de la barra de herramientas del diseñador de consultas.

La figura 4.2 muestra el diseñador de consultas genérico con la consulta copiada en el panel SQL. Observe que el botón de alternancia para habilitar el diseñador

de consultas genérico está seleccionado. Puede hacer clic en este botón si desea utilizar el diseñador de consultas gráfico.

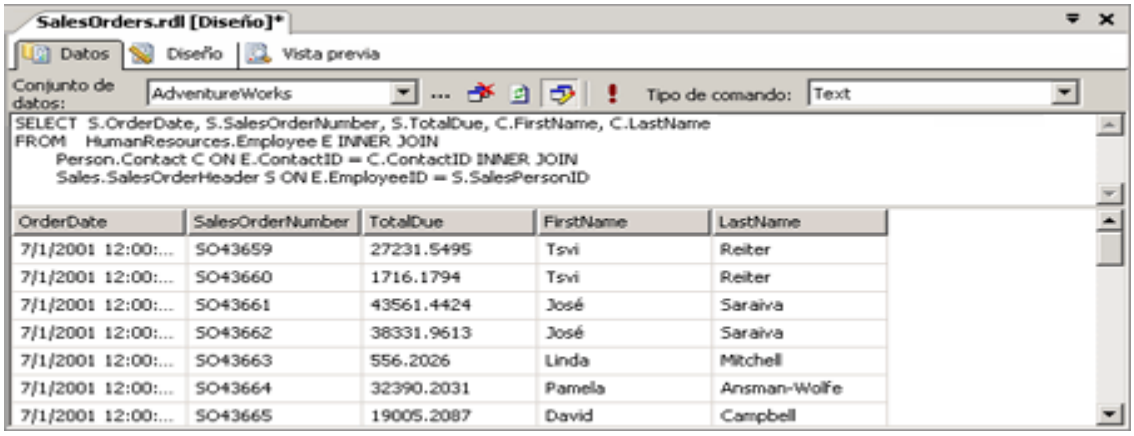


Figura 3.8 Ejemplo informe mediante consulta

4. CONCLUSIONES

- No existe una clasificación explicita sobre las herramientas para pruebas que hay actualmente.

- Las herramientas existentes son de acuerdo al tipo de prueba que se desea realizar, pruebas unitarias, de integración, de carga o stress etc.

- Por falta de conocimiento y experiencia en las mismas las empresas encargadas de realizar pruebas no cuentan con una herramienta especializada para documentar los casos de prueba.

- Las pruebas son una actividad continua durante las diversas etapas del ciclo de desarrollo de una aplicación.

- La planificación y diseño de las pruebas de sistema en las primeras fases de desarrollo, cuando aún se realiza la elicitación de requisitos, permite encontrar errores, omisiones, inconsistencias y sobreespecificaciones en los requisitos funcionales cuando aún es fácil y económico corregirlas, ya que el costo de eliminar defectos aumenta a medida que aumenta el tiempo qué transcurre entre la aparición del defecto y su detección (Jalote, Pankaj, 2002).

- La dificultad principal en el tema de pruebas de software no es la falta de estándares o investigación en el tema de las pruebas de software y aseguramiento de la calidad del producto, sino más bien la dispersión de dichos elementos y su difícil acceso. El trabajo futuro será la proposición de un estándar unificado que profundice y particularice los detalles que los estándares y modelos de calidad no contemplan. La difusión y

concientización a las empresas de desarrollo en la utilización de herramientas que apoyen el proceso de las pruebas de software. Y la posibilidad de adaptar una herramienta para documentación de pruebas open source, a las necesidades de las empresas que hay en el medio.

- La empresa Choucair Testing a través de la experiencia ya tienen un modelo o estándar definido para llevar a cabo la gestión de la documentación, sin embargo, el modo en que se implementa por ser manual, da pie a perdidas de documentos y reproceso. La herramienta en este caso jugaría un papel importante para la gestión de toda la documentación que se genera en el proceso de las pruebas del software.

5. BIBLIOGRAFÍA

Kaner, C. (2006). Instituto de la Florida de la tecnología, Conferencia de prueba del software anual mundial del instituto de la garantía de calidad. Orlando, FL.

Javier J. Gutiérrez, M. J. .Implementación de pruebas del sistema. Un caso práctico. Actas de Talleres de Ingeniería del Software y Bases de Datos, Vol. 1, No. 4.

Jokella, l. Usability and CMMI: Does A Higher Maturity Level in Product Development Mean Better Usability?.

IEEE - 0730, 2002, Standard for software quality assurance plans.

VELÁSQUEZ, C. D. (2009). PROPUESTA METODOLÓGICA PARA LA REALIZACIÓN DE PRUEBAS DE SOFTWARE EN UN AMBIENTE PRODUCTIVO. MEDELLIN: FACULTAD DE MINAS, UNIVERSIDAD NACIONAL.

F. Javier Zarazaga Soria, M. I. (2003). La ingeniería del software en el currículo del ingeniero de informática. Novática: Revista de la Asociación de Técnicos de Informática. Nº. 161, pags. 43-50.

F.L.Bauer. (1972). "Software Engineering", Information Processing. North Holland Publishing.
IEEE - 0730, 2002, Standard for software quality assurance plans.

Boehm, B. (1976). “Software Engineering”, IEEE Transactions on Computers.

M.V.Zelkovitz, A. J. (1979). Principles of Software Engineering and Design. Prentice Hall.

(1990). IEEE Std 610.12-1990 (R2002), IEEE Standard Glossary of Software Engineering Terminology.

IEEE Computer Society, S. G. (2004). Recuperado el Agosto de 2010, de http://www.swebok.org

G., M. (2004). The art of software testing. New Jersey: Editorial John Wiley & Sons.

Craig R, J. S. (2002). Systematic software testing. Londrés: Artech House Publishers.

Technology Evaluation Centers. (s.f.). Recuperado el JULIO de 2010, de http://test-tools.technologyevaluation.com/es/.

Lyndsay, J. (27 de Abril de 2005). *¿Qué son las herramientas de prueba de software?* . Recuperado el Julio de 2010, de http://www2.technologyevaluation.com/es/Research/ResearchHighlights/SoftwareTestTools/2005/04/research_notes/es/prn_TU_TT_XJL_04_27_05_1.asp

(s.f.). Recuperado el Julio de 2010, de http://www.mtp.es/content/view/160/198/lang,es/

SecurityInnovation. (s.f.). Recuperado el Agosto de 2010, de http://www.securityinnovation.com/holodeck/index.shtml
.

Róldan, J. (s.f.). Recuperado el Agosto de 2010, de http://tratandodeentenderlo.blogspot.com/2010/01/pruebas-funcionales-con-selenium.html

(s.f.). Recuperado el 17 de Junio de 2010, de http://www-306.ibm.com/software/info/ecatalog/ es_ES/ products/C108274I57640Y75.html.

(s.f.). Recuperado el Junio de 2010, de HTTP://FITNESSE.ORG/.

(s.f.). Recuperado el Mayo de 2010, de http://www.nolacom.com/avignon/index.asp.

(s.f.). Recuperado el Agosto de 2010, de HTTP://VALIDATOR.W3.ORG.

Sqa Design. (s.f.). Recuperado el Septiembre de 2009, de http://sqadesign.com/

CaseMaker International Ltd, a Diaz & Hilterscheid Group Company, Incorporated in England & Wales, Registered No. 5464015. . (s.f.). Recuperado el Agosto de 2010, de http://www.casemakerinternational.com/download.htm

IEEE. (1990). IEEE Standard Glossary of Software Engineering Terminology . *IEEE* , 74.

Ignacio, E. (2010). Marco de trabajo para la automatización de las pruebas funcionales usando herramientas open source . *Revista de Ingeniería Informatica PUCP* , 27-28.

NOSOLOUNIX. (2010, 02 28). *No solo Unix*. Retrieved 08 09, 2010, from www.nosolounix.com/2010/02/herramientas-test-software.html
Standards Coordinating Committee. (1990). IEEE Standard Glossary of Software Engineering Terminology . *IEEE* , 74.

msdn. (s.f.). Recuperado el 23 de AGOSTO de 2010, de http://social.msdn.microsoft.com/Search/es-ES?query=SQL%20Server%202005%20Reporting&ac=8