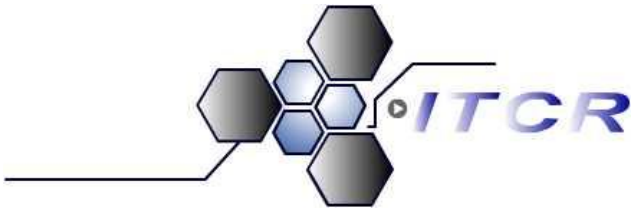
Instituto Tecnológico de Costa Rica

Sede San Carlos

Ingeniería en Computación



Northek Software S.A.

Mejoramiento en las pruebas del Software

Informe Final del Proyecto de Graduación para optar por el grado de Bachiller Ingeniería en Computación.

Alejandro Alfaro Quesada
[200116397]
aalfaro@northek.com

Ciudad Quesada Junio, 2006

Resumen Ejecutivo

El objetivo del siguiente informe es el de proporcionar una amplia información a los lectores sobre la metodología de Automatización de las Pruebas de Software en el área de Aseguramiento de la Calidad de Northek Software. En él se presenta información introductoria, diseño, y análisis de las situaciones más relevantes para un mejor desempeño en dicha automatización.

En el ciclo de vida de desarrollo para soluciones informáticas en Northek Software, se establecen etapas de pruebas, las cuales tienen como objetivo certificar que el producto final llegará a manos del cliente con una funcionalidad apegada a los requerimientos y que está libre de defectos que causen molestias de uso para el usuario final.

En los procesos de certificación constantemente se presenta la repetición de las mismas tareas, tanto en actividad como en entradas, y a pesar de ser tareas tan limitadas deben ser realizadas por personas, las cuales ocasionalmente dejan de hacer alguna prueba u observación. Significando esto que pueden existir defectos no reportados en el software y que las pruebas serán repetidas en tanto como la capacidad de las personas que realizan las pruebas lo permita.

El proyecto actual busca establecer mecanismos automáticos para realizar estás pruebas logrando de esta manera asegurarse de ejecutar todos los escenarios establecidos para un sistema y minimizar los tiempos y costos de certificación. La creación e implementación de estos mecanismos no interferirá con los procesos actuales de desarrollo, tampoco ocasionará que las responsabilidades actuales del departamento de Aseguramiento de la Calidad (QA) se vean aumentadas o disminuidas. Tan solo tienen el propósito de venir a ser herramientas que mejorarán el desempeño de las tareas realizadas en el departamento.

El presente informe abarca en una sección el diseño de la metodología sobre la Automatización de los Procesos de Software en el departamento de Aseguramiento de la Calidad (QA) de Northek Software.

Dicho diseño se ve especificado con el análisis detallado en primera instancia sobre el esquema actual de la metodología sobre los procesos de software utilizada por Northek Software. La especificación y análisis de los principales tipos de pruebas que conforman la estructura básica del Aseguramiento de la Calidad.

Por su parte, se examina la metodología deseada sobre los procesos de software, considerando la forma de trabajo de la herramienta de uso para la automatización de las pruebas; además de sus características que distinguen su apoyo profesional en el área de Aseguramiento de la Calidad (QA).

Una vez especificados los dos diseños de la metodología actual y deseada de las pruebas de software para el área de QA de Northek Software, se analiza la estructura diseñada para la grabación de las pruebas sobre la aplicación de Modularización de CMI (aplicación de prueba para garantizar el desempeño de la metodología de pruebas a implementar).

Obteniendo dicha información de antecedencia, se procede a la especificación y análisis de los procesos grabados sobre la Modularización de CMI, con el fin de detallar el diseño base de la metodología implementada en el área de QA de Northek Software.

Como conclusión del informe se presenta un análisis de las situaciones relevantes sucedidas a lo largo de la implantación de la metodología de Automatización de las Pruebas de Software en el Departamento de Aseguramiento de la Calidad de Northek Software.

Se presenta un análisis sobre los objetivos alcanzados en el proyecto, situaciones de mejoras sobre la automatización de los procesos realizados con el fin de informar al personal futuro sobre las características más relevantes que ayuden a la mejor automatización de las pruebas.

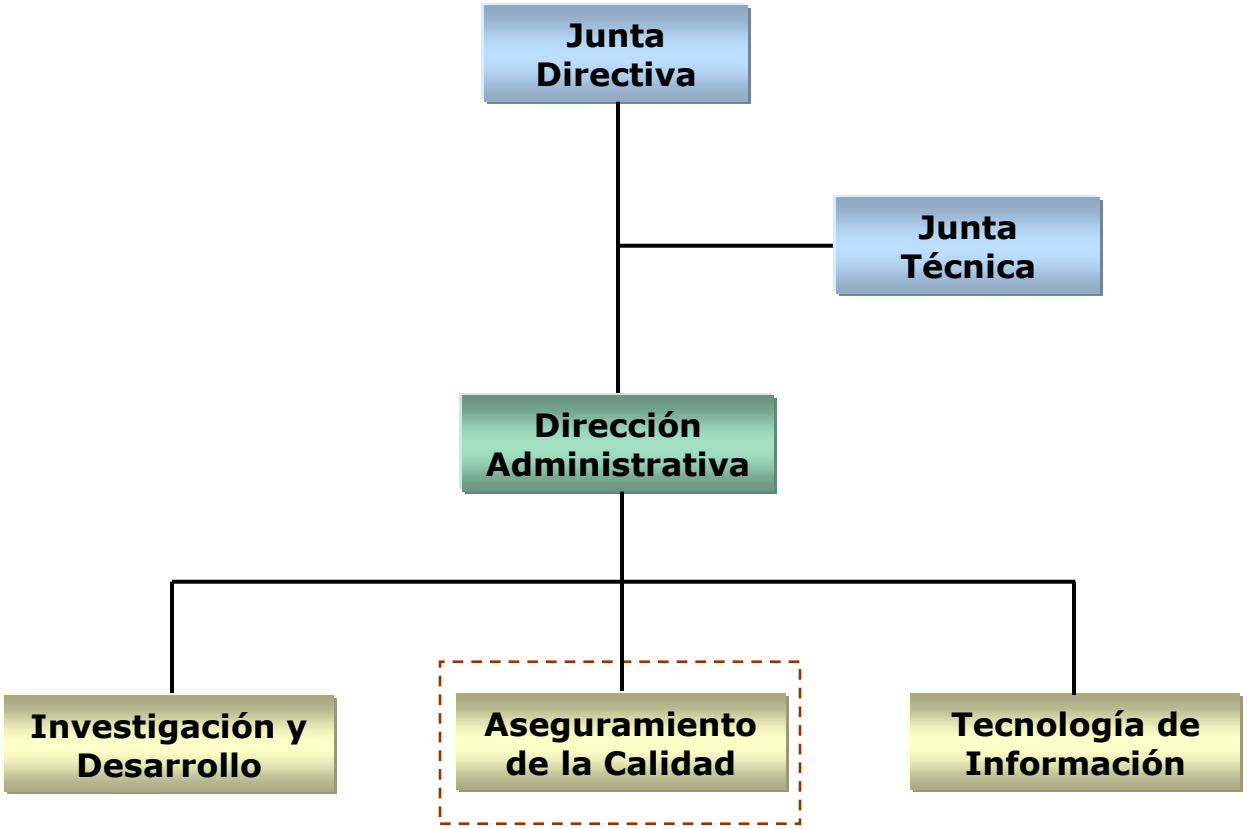
Tabla de Contenido

Indice de Figuras

Contexto del Proyecto

Organigrama Funcional del Proyecto

El organigrama que se presenta a continuación describe la organización básica del proyecto. Donde se pretende dar ubicación sobre el área donde se realizará el nuevo desarrollo, esta área será el departamento de Aseguramiento de la Calidad (QA) de Northek Software [Área demarcada con líneas consecutivas rojas].



Antecedentes del proyecto

En este espacio del documento, el proyecto como tal no cuenta con antecedentes específicos con lo que se pretende llegar.
Ahora por su parte si es factible denotar cierta información que ayuda a aclarar de donde surge la idea del actual proyecto.

- El proceso actual de Aseguramiento de la Calidad (QA) de Northek Software, es un proceso realizado en forma manual, es decir, las pruebas de los casos de prueba de los productos realizados, son realizadas una y otra vez (pruebas repetitivas) por el probador, hasta el punto en que el producto se encuentre libre de errores, para que dicho producto sea entregado al cliente, o en ciertas ocasiones cuando se da mantenimiento al producto una vez implantado al cliente.

- Las pruebas repetitivas en el departamento de Aseguramiento de la Calidad de Northek Software, es una tarea que depende del tipo de pulga y sus repercusiones sobre el producto, son pruebas que son exhaustivas para el probador, ya que, requiere de gran análisis y mucha precaución con el fin de que el error no vuelva a surgir o que no siga generando impactos considerables en el producto. A parte, de que esta tarea repetitiva consume otros factores de consideración como tiempo y costo para la empresa.

- Con la metodología del actual proyecto, lo que se pretende es la automatización de esas tareas repetitivas (pruebas repetitivas), con el fin de ahorrar muchos factores que son considerados de gran impacto para la empresa (tiempo, coste, etc.). Dicha metodología se encuentra en análisis desde mediados del año 2005; siendo hasta esta fecha donde se analiza la situación actual de la empresa, y es considerado dar un paso hacia esta nueva implantación en Northek Software.

Descripción del Problema

En el siguiente apartado, se pretende que el lector se entere sobre la necesidad actual de la empresa, con el fin de agilizar el trabajo del departamento de Aseguramiento de la Calidad de la empresa, y las armas con las que se intenta llenar las expectativas sobre tal necesidad. En términos generales del apartado, se analiza la visión que envuelve el proyecto.

Enunciado del Problema

En el desarrollo de software, ya sean paquetes comerciales, sistemas a la medida o creación de componentes se deben de realizar validaciones en los distintos niveles del software que van desde la interfaz de usuario hasta el almacenamiento de información, con diferentes tipos de pruebas como tipo de datos, seguridad, estrés, diseño, documentaciones técnicas, pruebas de unidad, usabilidad de interfaz entre otros. Estas pruebas no deben ser llevadas a cabo por los desarrolladores del programa ya que ellos como creadores saben cuales son las funcionalidades del sistema y las entradas que darán generalmente estarán sesgadas a buscar una respuesta exitosa en cada prueba. Paralelamente al desarrollo se establece un equipo de pruebas, el cual durante el tiempo de desarrollo se dedica a establecer una serie de distintos escenarios y grupos de entradas, aplicables sobre el software a certificar.

Desde la primera liberación de una versión del software por parte de desarrollo, el equipo de pruebas debe ejecutar en su totalidad las pruebas preestablecidas para el producto liberado, registrando los resultados de las pruebas, como parte de este registro se deben realizar informes a desarrollo sobre los problemas encontrados y las posibles soluciones que se pueden implementar. Desarrollo debe de resolver los problemas encontrados y entregar un nuevo producto al equipo de pruebas quien en primera instancia validará directamente que los

problemas reportados han sido corregidos y luego de esto se debe proceder a ejecutar nuevamente en su totalidad las pruebas preestablecidas para el producto liberado. Estos pasos se repiten en las actividades de pruebas hasta que el responsable de las pruebas emita su aprobación e informe a los involucrados que el software se da por certificado. Lo que significa que está en condiciones aceptables para ser entregado al cliente, además que desarrollo no debe modificar más código y que no se realizaran más pruebas.

Es por ello, que el procedimiento del trabajo del departamento de Aseguramiento de la Calidad de Northek Software, en ciertas oportunidades se vuelve exhaustivo generando gran responsabilidad por parte de los integrantes de dicho departamento, en términos de poder realizar las pruebas necesarias sobre el producto en un tiempo prudencial para la entrega del sistema al cliente. El hecho, es que las pruebas pueden llegar a ser tareas repetitivas que impliquen coste y tiempo para la empresa, que puedan traer repercusiones en la entrega final del producto; llevando lo anterior a un impacto, que en buenos términos a una culminación del producto aceptable tanto para Northek Software como al cliente al que se le entrega el producto; o por su parte, a una situación que puede llegar a ser crítica de entrega de producto tanto a nivel de tiempo y coste, como de que el producto sea lo esperado por el cliente y que tenga la funcionalidad deseada por él.

La situación anterior, tanto a una entrega en buenos términos como una entrega no deseada del producto al cliente, afecta en muchos ámbitos a Northek Software como empresa creadora del producto, como al cliente que le dará funcionalidad al producto. A continuación se especifican algunas de las razones que afecta dicha situación:

- Es evidente que el atraso de entrega de un producto afecta tanto a Northek Software como desarrollador de la aplicación, como al cliente que manipula el producto. Siendo estos los mayores afectados por el impacto

que pueda ocasionar, el análisis y ejecución de pruebas repetitivas del producto, las cuales consumen gran cantidad de tiempo y coste, en donde el impacto será mayor conforme avancen situaciones no deseadas que ocurran con el producto.

- Por su parte, se ven afectados los encargados del departamento de QA, ya que, al ser los responsables de realizar las pruebas necesarias sobre el producto, recae sobre ellos una responsabilidad muy seria, que puede llegar a ocasionar que se caiga en una situación de omitir el análisis de algunas pruebas o suposiciones de algunas situaciones de evaluación de pruebas, que al final generan más coste y tiempo para la empresa, si es que dichas omisiones producen alguna repercusión mayor sobre el producto.

- Otro departamento que se ve afectado, por el impacto de dicho problema, es el departamento de desarrollo, que aunque éstos no están relacionados directamente con el análisis de las pruebas sobre el producto, si son los encargados de solucionar todas aquellas pulgas que surjan del análisis de las pruebas por parte del departamento de QA. Esto implica, que los desarrolladores sean responsables de resolver las pulgas en un tiempo prudencial, obligándolos a acelerar el paso de resolución de las mismas hasta el punto de que desarrollo tome en consideración tiempo extra para la solución de las pulgas, y siempre esperando que esa solución sea aceptable, ya que dichas soluciones de nuevo serán analizadas por el departamento de QA, y que se podría recaer de nuevo al mismo ciclo con no sería deseado para ninguna de las dos partes (Northek Software y el cliente).

Enunciado de la Solución

El proyecto actual se enfocará en identificar y esquematizar las tareas que se deben de realizar repetidamente dentro del proceso de certificación. Una vez estructurada esta información se procederá al diseño e implantación de una solución informática que sea capaz de llevar acabo la ejecución de un grupo de pruebas sobre un software, obteniendo a la vez retroalimentación de estas pruebas, y que sea capaz de generar estados de resultados obtenidos de las pruebas realizadas.

Se pretende tener una visión en considerar metodologías basadas en modelos para pruebas de Software, implementados junto a estas, procesos automáticos que se encarguen de realizar aquellas tareas que deben ser ejecutadas en más de una ocasión para obtener la certificación del Software.

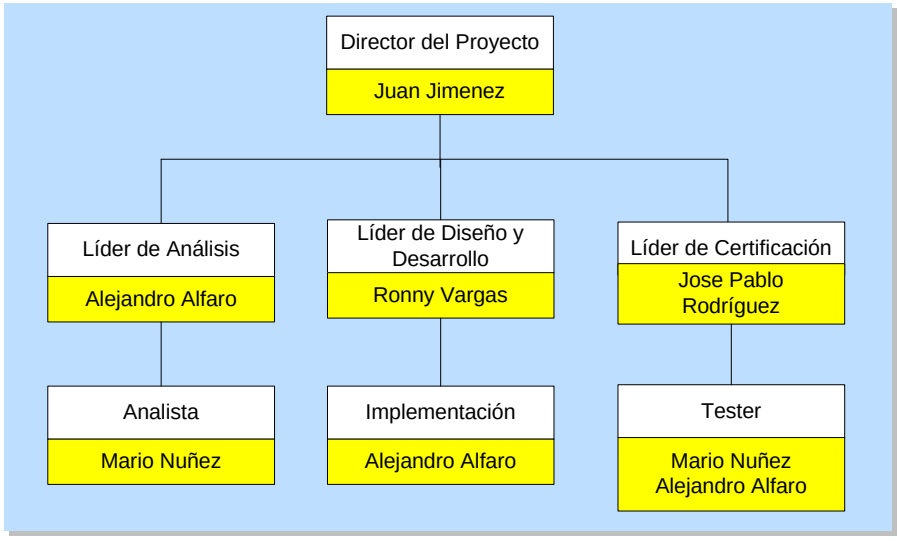
Se realizarán métricas que determinen los puntos críticos, donde el trabajo se vuelve repetitivo dentro de las tareas del departamento de Aseguramiento de la Calidad (QA), procediendo a tabular la información y generar un documento con los resultados.

El producto deberá ser documentado en sus procesos que apliquen a los trabajos y necesidades de Northek Software, además se entregará junto con un ejemplo práctico de producción. Para aprovechar este esfuerzo se decide montar este ejemplo práctico sobre otro proyecto del departamento de Desarrollo, que se está ejecutando paralelamente a las investigaciones y tareas de nuestro proyecto. El proyecto elegido para tal caso será CMI. Es por esto que en el presente proyecto, se pretende realizar grabaciones en la herramienta sobre la modulación de la aplicación CMI, con el fin de eliminar el factor humano de las tareas repetitivas dentro del departamento de QA obteniendo un tiempo de certificación más adecuado para la entrega del producto al cliente, llevando esto a minimizar los costes de certificación y por ende de producción del software ocasionando una elevación de la calidad del software que se le entrega

al cliente. En este caso para automatizar los escenarios que se encuentran hechos para dicho proyecto, automatizaciones que facilitarán el trabajo para el departamento de QA de Northek Software. Las grabaciones de las pruebas sobre la modulación de CMI se pretenden automatizar con el fin de que cuando ocurran revisiones posteriores a la modulación éstas faciliten el trabajo de los probadores. Además, de que dichas grabaciones de pruebas sobre la modulación de CMI vendrán a certificar a la nueva herramienta, garantizando la capacidad de automatizar las pruebas con esta nueva metodología.

Dicha metodología genera un aporte bastante beneficioso tanto para el departamento de QA como para Northek Software, en su situación de empresa como responsable del desarrollo del producto. Donde, la metodología proporciona un manejo más adecuado del tiempo y coste, en cuanto al análisis de las pruebas sobre el producto, y ayudando al crecimiento de la capacidad de respuesta de la empresa en la certificación de sus productos hacia sus clientes.

Clasificación de los Involucrados



El proyecto tiene el siguiente inventario de involucrados, a diferentes niveles:

Involucrados	Puesto	Responsabilidad en el proyecto
José Pablo Rodríguez	Gerente general	Líder de certificación
Ronny Vargas	Director de Investigación y Desarrollo	Líder de diseño y desarrollo
Juan Antonio Jiménez	Director de QA	Director del proyecto
Mario Núñez	Ejecutor de pruebas	Analista y tester
Alejandro Alfaro	Investigación, desarrollo y ejecutor de pruebas.	Líder de análisis, implementación y tester.

Objetivo por involucrado en el proyecto, para que éste sea exitoso:

Involucrados	Objetivo
José Pablo Rodríguez	Certificación del proyecto. Control sobre el procedimiento a seguir para la solución del proyecto.
Ronny Vargas	Control sobre el desarrollo del proyecto CMI, el cual será utilizado como caso de prueba en el desarrollo del proyecto del Mejoramiento en las Pruebas de Software.
Juan Antonio Jiménez	Dirección y control directo sobre el proyecto.
Mario Núñez	Soporte sobre la metodología de prueba de Northek Software.
Alejandro Alfaro	Análisis, investigación, documentación sobre la metodología a desarrollar.

Necesidades y Expectativas

La necesidad se basa en poder dar soporte al factor humano del departamento de Aseguramiento de la Calidad de Northek Software en la ardua tarea de verificación del software que se desarrolla en la empresa, donde la prioridad de la necesidad es dar un fuerte soporte en tareas repetitivas con las cuales los probadores tienen que lidiar. Las cuales en determinados momentos se vuelven exhaustivas y pueden provocar consecuencias considerables a lo largo del desarrollo del proyecto, inclusive también una vez que se haya entregado el producto al cliente.

El problema actual que conlleva al estudio de esta necesidad es que dichas tareas repetitivas que invaden el trabajo del departamento de Aseguramiento de la Calidad en Northek Software, conllevan en muchas ocasiones a influir en los factores de coste y tiempo de la empresa, que depende de la situación que se presente puede tener un impacto bastante considerable. Generando impactos en el departamento de Aseguramiento de la Calidad (QA), en el departamento de Desarrollo, la empresa como tal (Northek Software) y el usuario final (cliente) quien es el que manipulará la aplicación.

Actualmente dentro de Northek Software, específicamente en el departamento de QA, el análisis de las pruebas sobre los productos son realizadas manualmente con el apoyo de un software (Bugzilla), que en términos generales las soluciones que actualmente se obtienen de esta forma, son aceptables, y hasta el momento se han podido cumplir las expectativas del trabajo del departamento de QA. Pero por su parte, siempre Northek Software tiene el deseo de superación en todos sus ámbitos es por lo cual se intenta dar un soporte mayor al departamento de QA de la empresa.

Por lo tanto, se pretende automatizar las pruebas repetitivas sobre los productos desarrollados en la organización con la idea fundamental de apoyar ese sector de la empresa. Por su parte, la automatización de las pruebas lleva a poder reducir los tiempos en que se corren actualmente los grupos de pruebas sobre cada software a certificar, poder evitar la no detección de errores por descuidos humanos, minimizar los costes de certificación y por ende de producción de software, elevar la calidad de software que se le entrega al cliente, entre muchas otras situaciones que se pueden derivar de los casos descritos anteriormente.

Perspectivas, supuestos y dependencias

Perspectivas

- Se pretende automatizar en al menos 40% las tareas realizadas por el departamento de Aseguramiento de la Calidad (QA) de Northek Software. En donde el porcentaje representa lo que se pretende alcanzar de automatización en la empresa, ya que Northek Software cuenta con varias aplicaciones para el mercado, y en donde para el caso de esta Práctica de Especialidad se toma en cuenta la automatización de la modulación de la aplicación CMI.

- Realizar las pruebas automatizadas sobre la aplicación CMI.

- Evitar la no detección de errores por descuidos humanos.

- Ahorrar tiempo y coste, en la detección rápida y temprana de pulgas evitando pruebas adicionales de las regresiones.

- Proporcionar entregas de alta calidad del producto al cliente.

Beneficios Esperados

- Reducir los tiempos requeridos para certificar el software.

- Construir pruebas de una forma rápida y confiable.

- Grabaciones de pruebas de carga, estrés y escalabilidad.

- Evitar el costo de reanudación y de pruebas adicionales de regresiones, disminuyendo la persistencia de éstas después del lanzamiento del producto.

- Evitar la no detección de errores por descuidos humanos.

- Minimizar los costos de certificación y por ende de producción.

- Elevar la calidad del software que se le entrega al cliente.

Supuestos

- Se cuenta con los documentos de estándares que aplican al desarrollo.
- El departamento de Investigación y Desarrollo realizará la implantación del software a probar (CMI).
- Se cumplirán con las políticas, normas y estándares propios en el desarrollo del software.
- El software será utilizado por los usuarios del departamento de Aseguramiento de la Calidad (QA) de Northek Software.

Dependencias

- El departamento de Investigación y Desarrollo proporciona la aplicación para las pruebas (CMI).
- Utilización adecuada y aceptable de la herramienta, por parte de los encargados del departamento de Aseguramiento de la Calidad (QA) de Northek Software.
- Explicación del manejo exacto de la aplicación a probar (CMI) por parte de Desarrollo.

Requerimientos no funcionales

- Eliminar el factor humano de las tareas repetitivas dentro del departamento de Aseguramiento de la Calidad (QA) de Northek Software.
- Reducir los tiempos en que se corren actualmente un grupo de pruebas sobre cada software a certificar.
- Asegurarse que una vez establecido un escenario para pruebas este no deje de ejecutarse involuntariamente.
- Aumentar la experiencia de los encargados de realizar las pruebas en el departamento de QA de Northek Software en cuanto al uso de la herramienta para la automatización de pruebas.
- Existencia de documentación clara sobre el uso de la herramienta, sus objetivos y características que desempeña dentro del departamento de QA de Northek Software.

- Asegurarse de compatibilidad tanto con otro hardware como existencia de otros sistemas que no vayan a influir en el desempeño de la herramienta.

- Por seguridad el derecho de administración de la herramienta será realizada por los encargados del departamento de QA de Northek Software.

- Se pretende que la metodología implementada sea de fácil acceso y manipulación para nuevos miembros del departamento de QA de Northek Software.

Características Generales

Medidas

- Todas las pruebas planeadas sobre el sistema son ejecutadas.

- Todos los defectos encontrados y reportados, son válidos.

- Los tiempos de ejecución de pruebas deben ser menores a los reportados actualmente.

- El rendimiento y desempeño del proceso de certificación del software deben ser mejor que el actual.

Exclusiones

- Las metodologías y sistemas propuestos para las pruebas no vendrán a establecer normas para el desarrollo del software.

- El trabajo final no contemplará todos los tipos de pruebas que se realizan sobre un software.

- El trabajo final no establecerá límites en los tipos de pruebas que pueden realizarse sobre el software.

Restricciones

- No es posible abarcar todos los escenarios.

- No es posible abarcar todas las entradas.

- No es posible abarcar todas las permutaciones de secuencias.

Análisis de Riesgos

Nombre	Demora en el aprendizaje sobre el uso de la herramienta por parte del departamento de Aseguramiento de la Calidad de Northek Software.
Categoría	Personas
Causa	Herramienta con características nuevas con una forma diferente de trabajar.
Impacto	Medio
Probabilidad de Ocurrencia	30%
Exposición	30%
Estrategia de Evasión	Tratamiento de la mejor manera de la herramienta, trabajar con orden.
Estrategia de Contingencia	Capacitar al departamento sobre las nuevas características.

Nombre	La calidad de la especificación de la aplicación CMI se encuentra bien, o se encuentran pobremente definida.
Categoría	Personas
Causa	Documentación apta o pobre de la aplicación, para facilitar o complicar el entendimiento de la misma por parte de los encargados de testing, para realizar las pruebas.
Impacto	Bajo
Probabilidad de Ocurrencia	15%
Exposición	15%
Estrategia de Evasión	Verificar con tiempo el estado de especificación de la aplicación, para analizar con detalle las consecuencias.
Estrategia de Contingencia	Estudiar la situación de la especificación, y en caso de dudas, apoyarse en los desarrolladores del proyecto.

Nombre	Falta de orden en la automatización de las pruebas de una aplicación, que conlleve a un pobre entendimiento conforme avancen las grabaciones.
Categoría	Personas
Causa	Pobre análisis de la estructura de prueba que debe seguir una determinada aplicación.
Impacto	Bajo
Probabilidad de Ocurrencia	20%
Exposición	20%
Estrategia de Evasión	Análisis con anterioridad de forma detallada de la aplicación a probar, especificando el orden adecuado de prueba.
Estrategia de Contingencia	Conocer de una forma aceptable el manejo adecuado de la aplicación a la cual se le realizará las pruebas automatizadas.

Nombre	Extravío o pérdida de información de importancia, tal como: reportes, grabaciones de pruebas, plan de pruebas de algún software a probar, entre otros.
Categoría	Tecnológico
Causa	Fallo de Hardware
Impacto	Medio
Probabilidad de Ocurrencia	15%
Exposición	15%
Estrategia de Evasión	Respaldo de archivos valiosos periódicamente.
Estrategia de Contingencia	Informar a los demás miembros sobre el fallo ocurrido para rescatar de ellos todo el material que se pueda.

Nombre	Los requerimientos técnicos de uso de la herramienta son nuevos y complejos.
Categoría	Tecnológico
Causa	Implantación de una herramienta nueva, con aspectos y conceptos innovadores para realizar la automatización de pruebas a una determinada aplicación.
Impacto	Medio
Probabilidad de Ocurrencia	20%
Exposición	20%
Estrategia de Evasión	Trabajar en forma ordena y conocer al menos las características más relevantes de la herramienta a implantar.
Estrategia de Contingencia	Capacitación al personal sobre los usos más relevantes de la herramienta por implantar.

Nombre	Ingreso de forma no autorizada sobre la herramienta con el fin de insertar, modificar o eliminar datos sobre las pruebas, sin el consentimiento de los encargados.
Categoría	Políticas empresariales.
Causa	No poseer los permisos necesarios para acceder a la información.
Impacto	Bajo
Probabilidad de Ocurrencia	15%
Exposición	15%
Estrategia de Evasión	Establecer la(s) persona(s) encargadas de otorgar permisos para el acceso a la información.
Estrategia de Contingencia	Utilización de objetos de seguridad, que nieguen el acceso a usuarios sin permisos.

Objetivos del Sistema

Objetivo General

Automatizar en al menos 40% de las tareas realizadas por el departamento de Aseguramiento de la Calidad (QA), antes del 1 de julio del 2006.

Objetivos Específicos

- Se contará con la modularización de la aplicación CMI, con el fin de realizar las pruebas automatizadas sobre dicha aplicación.

- Se pretende realizar una detección rápida y temprana de pulgas evitando pruebas adicionales de regresiones, que conlleve a un aumento de coste y tiempo en la certificación del software.

- Mantener un orden en la automatización de las pruebas para realizar entregas de alta calidad del producto al usuario final.

Alcance del Proyecto

Se analizarán todas las pruebas que se realizan en el departamento de Aseguramiento de la Calidad (QA) de Northek Software, con el propósito de definir un grupo de éstas y establecer metodologías que aprovechen los recursos informáticos, definiendo tareas que se lleven a cabo automáticamente por el sistema, las nuevas metodologías no deberán alterar las actividades de desarrollo.

Como ejecución de pruebas de la herramienta a implementar, se automatizarán pruebas sobre la modularización de la aplicación CMI. Garantizando el desempeño de la metodología a implementar en el departamento de QA de Northek Software, y fijando la automatización a la aplicación CMI, como inicio del proceso de automatización de las anteriores y futuras aplicaciones desarrolladas por Northek Software.

Proceso Actual

La definición de entradas, configuraciones e inicio de las pruebas automáticas serán administradas por el departamento de QA de Northek Software.

En esta sección se introduce al lector al proceso de pruebas (Testing) del software. Se Aclararán conceptos y se explicará el ciclo de vida de una pulga (Bug), así como la forma correcta de interactuar con las pulgas y saber identificar las transiciones pertinentes de una Pulga.

Dicho análisis es de los procesos actuales de aseguramiento de la calidad de Northek Software, el cual esta implementado con la herramienta Bugzilla; es por ello que la información presente en esta sección del informe será enfocada a la herramienta Bugzilla.

Por su parte, cabe destacar que el lector tome en consideración el glosario de términos que se encuentra especificado en la sección de anexos del documento, con el fin de determinar la importancia de varios conceptos que son contemplados en todo el proceso de pruebas y análisis del ciclo de vida de una pulga.

Sobre Bugzilla

Bugzilla es una aplicación que permite dar seguimiento a fallos dentro de cualquier tipo de aplicación que se este analizando su calidad como software. Permite llevar un control sobre los fallos ocurridos o la documentación de las

solicitudes de mejoras. El desarrollo del trato de los fallos dentro de bugzilla es en forma manual, indicando paso por paso el proceso de la pulga y su posible solución.

Características

- Estructura optimizada de la base de datos para un funcionamiento y escalabilidad más alto.
- Excelente seguridad para contener la información registrada del proceso de las pulgas de una determinada aplicación.
- Sistema de búsqueda de pulgas registradas.
- Sistema de envíos de correos, a las personas de desarrollo involucradas con la pulga que surge en una determinada aplicación.
- Permite determinar permisos de uso, dependiendo del tipo de usuario registrado al sistema.

Beneficios

- Permite una amplia comunicación entre el área de aseguramiento de la calidad con el área de desarrollo dentro de una empresa.
- Permite dar un aumento en la calidad del producto, por su orden de controlar la información de las pruebas.
- Mejora la satisfacción del cliente, al poderse entregar un producto más certificado por parte de la empresa.
- Permite a los encargados del área de Aseguramiento de la Calidad llevar un mayor control sobre las pruebas incentivándoles responsabilidad hacia tal tarea.

Ciclo de vida de las pulgas en Bugzilla

Dentro del área de Aseguramiento de la Calidad de Northek Software se utiliza la herramienta Bugzilla, la cual en secciones anteriores se explico las principales

características de ella, y la cual determina un proceso de registro de documentación de pruebas y de la aparición o no de las pulgas provenientes de la revisión de la calidad de dichas pruebas. Es por ello, que dichas pulgas que aparecen en la revisión de una aplicación cumplen con un estándar sobre el proceso de registro en Bugzilla, a esto es lo que llamamos el ciclo de vida de las pulgas en Bugzilla, el cual se define su proceso en la tabla siguiente, indicando el posible estado (estado mientras es resuelta por el desarrollador o mientras se define que hacer con la pulga) que se puede encontrar una pulga dentro de Bugzilla y la resolución (estado después de un análisis o resolución de alguna de las pulgas) que puede tomar la pulga dependiendo del estado en que se encuentre.

A continuación se define el ciclo de vida de las pulgas en Bugzilla:

Estado	Resolución
El campo del estado indica la salud general de una pulga. Solamente se permiten ciertas transiciones del estado.	El campo de la resolución indica qué le sucedió a la pulga
Unconfirmed Esta pulga se ha agregado recientemente a la base de datos. Nadie ha validado que esta pulga es verdad. Usuarios que tienen el "canconfirm" el sistema del permiso puede confirmar esta pulga, cambiando su estado a NUEVO o, puede ser resuelto y ser marcado directamente RESUELTO.	Ninguna resolución todavía. Todas las pulgas que son en uno de estos "open" los estados tienen la resolución fijada al espacio en blanco. El resto de las pulgas serán marcadas con una de las resoluciones siguientes.
New Esta pulga se ha agregado a la lista de assignee's de pulgas y debe recientemente ser procesado. Las pulgas en este estado pueden ser aceptados, y ASIGNARSE, pasado encendido algún otro, y siguen siendo NUEVOS, o resuelto y marcado RESUELTO.	
Assigned	

Esta pulga todavía no se resuelve, sino se asigna a la persona apropiada. Aquí la pulga puede ser dada a otra persona y llegar a ser NUEVO, o ser resuelto y RESOLVERSE.

Reopened

Esta pulga fue resuelta una vez, pero la resolución era juzgada incorrecta. Por ejemplo, SE ABRE DE NUEVO un insecto de WORKSFORME cuando más información demuestra para arriba y el insecto es reproductivo ahora. Aquí de insectos están marcados ASIGNADO o RESUELTO.

Resolved

La resolución de se ha tomado, y está aguardando la verificación de QA. Aquí de insectos se abren de nuevo y SE ABREN DE NUEVO, están marcados VERIFICÓ, o son cerrado para CERRADO bueno y marcado.

Verified

El QA ha mirado la pulga y la resolución y conviene que se ha tomado la resolución apropiada. Sigue habiendo las pulga en este estado hasta el producto que fueron divulgados contra realmente las naves, en que punto llegan a ser CERRADOS.

Closed

La pulga se considera muerta, la resolución está correcta.

Fixed

Se comprobó que la pulga se resolvió correctamente.

Invalid

El problema descrito no es la pulga

Wontfix

El problema descrito es una pulga que nunca será fixed.

Duplicate

El problema es una pulga duplicada. Marcar un duplicado de la pulga requiere el # de la pulga que esta duplicada y pondrá por lo menos ese número del insecto en el campo de descripción.

Worksforme

Que todas las tentativas en la reproducción de esta pulga eran vano, leyendo el código no produce ninguna pista en cuanto a porqué ocurriría este comportamiento. Si aparece más información más adelante, reasigne por

	favor la pulga, para ahora, lo archivan. **Moved** El problema era específico a un producto relacionado que la pulga se sigue en otra base de datos. La pulga se ha movido a esa base de datos.

Una vez estudiado el ciclo de vida de las pulgas en Bugzilla, existen conceptos importantes que deben ser tomados en consideración y que son parte complementaria del proceso del ciclo de vida de las pulgas; dicho situación se refiere a la severidad y prioridad de una pulga en su tratamiento. A continuación se presenta una gama de conceptos que definen dicha severidad y prioridad de las pulgas:

Severidad
Aquí se describe el impacto de una pulga en un sistema.
Blocker: bloquea el trabajo del desarrollo y de prueba, es decir, el sistema no puede continuar con otras funciones porque la pulga no deja que el sistema siga su trabajo normal, y por el lado de las pruebas éstas no se pueden continuar por el mismo hecho ya que no se puede continuar probando el resto del sistema porque la pulga no se lo permite, está bloqueando el sistema.
Critical: son todas aquellas pulgas que son consideradas como críticas, las cuales no pueden suceder en una aplicación desarrollada, en donde dichas pulgas no dan consistencia a la aplicación tanto dentro de la empresa desarrolladora cuando la prueban, como la aplicación implantada donde el cliente, algunas de este tipo son: caídas del sistema, pérdida de datos, escape severo de la memoria.
Major: pérdida importante de función, es decir, la pulga proporciona situaciones donde se continúa con el proceso de la aplicación pero llega el punto donde hay inconsistencias porque la funcionalidad no fue esperada en secciones anteriores.
Minor: pérdida de menor importancia de la función, es decir, se puede continuar con el proceso de la aplicación porque la situación funcional que ocurrió, no es tan determinante en el producto final, en donde no quiere decir que no es tomada en cuenta ni que debe ser resuelta.
Trivial: son todas aquellas pulgas que se basan sus errores en situaciones meramente insignificantes, pero que marcan una situación de estética importante en el producto final, algunas de esas situaciones por ejemplo son: problemas cosméticos, como palabras deletreadas mal o texto mal alineado.
Enhancement: son todas aquellas pulgas que no son de gran prioridad el tomarlas en cuenta pero que son importante tomarlas en cuenta.

Prioridad
Este campo se describe la importancia y el orden de una pulga. Este campo es utilizado por el programmers/engineers para dar la prioridad a su trabajo que realizará. Las prioridades disponibles se extienden de P1 (más importante) a P5 (lo más menos posible importante).

Test Runner

Test Plan

Para esto es necesario entrar a la página principal del BTR donde se enumera todos los productos definidos en su base de datos de Bugzilla. Los testplan se asocian a los productos. Cuando se da clic en un nombre del producto, el BTR presenta en forma de una lista los testplan definidos para ese producto dado. Inicialmente, no hay testplan entonces se debe identificar la opción de agregar un nuevo testplan. Luego se puede ver que el producto elegido tiene un testplan definido. Esta página es el punto de comienzo al trabajar con testplan.

Documento del plan

El documento del plan es donde se describe lo que se suponen son las pruebas para hacer en general. La información que se incluirá son los requisitos del propósito, del alcance, del software y de hardware del plan, entre otros.

El BTR en sí mismo no hace cumplir ninguna estructura definida para un documento del plan. El documento se almacena y se exhibe en el HTML. Usted puede insertar acoplamientos a la documentación externa, o a Bugzilla, u otras páginas del BTR.

Pruebe los casos

La base de cada testplan es la lista de sus casos de la prueba. En el BTR, las cajas de la prueba se concentran en grupos funcionales.

Los grupos funcionales responden a muchos propósitos. Primero, dividen los testcases de un testplan en sistemas más pequeños, más fáciles digerir. En

segundo lugar, proporcionan una cierta "forma básica" para su testplan. Creado el testcase le agrupa específicamente los aspectos del software que usted desea probar.

Algunos ejemplos de grupos funcionales son:

- Pruebas De la Instalación
- Pruebas De Tensión
- Pruebas del GUI (Guía de Interfaz de Usuario)

Se pueden crear grupos, o utilizar plantillas predefinidas provistas del BTR.

Después de que se crean los grupos funcionales para el producto, se está listo para comenzar a crear casos de la prueba.

Un caso de la prueba se puede definir como el sistema de operaciones que ejercitan una cierta funcionalidad en su producto, un sistema de entradas, y los resultados previstos. Un testcase puede o no puede ser automatizado. El BTR considera los elementos siguientes en un caso de la prueba:

- Autor
- Versión
- Identificación del requisito
- Resumen
- Acción
- Efecto
- Requerimiento

Requerimiento: En este campo va el número del requerimiento, de acuerdo con el documento de alcances del proyecto. Este documento ya elaborado con anterioridad en base a los requerimientos del sistema es indispensable para el trabajo del departamento de QA.

Se almacenan los campos del resumen, de la acción y del efecto, exhibido en HTML. En algunos casos un testcase puede ser bastante complejo y los elementos del HTML tales como tablas o listas pueden venir muy prácticos presentarlo de manera legible.

En los testcases también se puede compartir información. Imagínese, que se han fijado los testcases, que requieren las mismas condiciones iniciales, e.g. "módem desconectado de la línea telefónica". Se puede repetir esa oración en *el campo* de la acción para cada caso de la prueba. Esto es redundante, en lugar de esto, se puede utilizar algo llamado "etiquetas comunes". La etiqueta es algo que se puede unir a más de un caso y a las ayudas de la prueba evitando repetirse.

Yendo de nuevo al ejemplo del módem, se puede definir una etiqueta nombrada "módem desconectado" y agregarla a todos los casos de la prueba que la necesitan.

Una etiqueta tiene un nombre y un valor. El nombre se limita a 240 caracteres, mientras que su valor puede ser mucho más largo. Algunas etiquetas pueden consistir en un nombre solamente, como en ejemplo del módem.

Una etiqueta puede definir las condiciones que son iguales para un número de casos de la prueba. Puede también ser tratada como cualidad adicional de un caso de la prueba.

Los campos de la acción y del efecto, se ligan a un testcase solamente, no a una entrada del registro del testcase. De esta manera, si se corre un testcase y después se cambian sus etiquetas, se verá la etiqueta actualizada en la entrada del registro del testcase.

Cada testplan tiene su propia lista de etiquetas.

Lista de testers para los testplan

El BTR no solo define que hacer en cada caso de la prueba, también quien debe hacerlo (correr). Para esto es la lista del testers.

Cuando se corre un testplan, BTR crea copias de cada caso de la prueba para cada tester definido. Por el defecto, todos los casos de la prueba son asignados a todos los testers. Se puede cambiar esto asignando testers a los componentes.

Correr un TestPlan

Cuando se ha definido el documento del testplan, creando sus testcases y asignando los tester a los componentes, se está listo correr el testplan.

El BTR distingue entre el testplan y la corrida del testplan. Un testplan se define una vez y se puede correr múltiples ocasiones. Mientras que los productos cambian, los testplan cambian también. Se agregan, se borran, se cambian algunos testcases y listo.

Un testplan corriendo por otra parte, se refiere a una versión específica de un plan del producto y de prueba. Cuando se corre un testplan, las entradas del registro del testcase se crean para cada tester definido. Los tester comprueban los test cases uno por uno que marcan su entrada del registro según lo "pasada" o "fallada." Cada tester tiene sus entradas separadas del registro.

Cuando un testcase falla, tenemos una nueva pulga, y es aquí donde el encargado del aseguramiento de la calidad junto con la herramienta Bugzilla se encargan de reportar el análisis de lo sucedido con la falla, al encargado de esa sección o del sistema como tal en el área de desarrollo. Se le indica mediante un correo la severidad y prioridad de la pulga, la situación que ocurrió cuando se probó y la situación que se esperaba; todo lo anterior es documentado por el encargado del departamento de QA.

Ciclo de Vida del Desarrollo

Toma de Requerimientos

Esta es la primera etapa en todo sistema, esta consiste en una serie de reuniones con el cliente, cuantas sean necesarias, para determinar que es lo que el cliente necesita que se le haga.

Desarrollo y Testing

El departamento de QA debe ir paso a paso con el departamento de desarrollo, mientras ellos empiezan a desarrollar, QA entra a la etapa de desarrollo, en la cual se debe ir creando los testplans y testcases de los requerimientos, así como desarrollo vaya creciendo el tester podrá ir creando los testcases de los testplans del launcher.

El líder del proyecto deberá ir creando builds ya con los procesos terminados, en este momento esta persona deberá enviar un correo al tester informándole que se encuentra un build nuevo, así se podrá continuar con las revisiones.

Las pulgas que se están reproduciendo durante la revisión del nuevo build, se deberán agregar al bugzilla, claro teniendo en cuenta que existe un encargado por modulo o un encargado llamado Triage, donde la pulga será revisada por el líder del proyecto y el coordinador de QA; si existen pulgas corregidas en el bugzilla, lo cual significa que la pulga se encuentra como fixed, se debe verificar que este correctamente arreglada, si fuera el caso, se debe cambiar el status a close o si no esta correctamente arreglada se debe cambiar el status a reopened y poner un comentario del porque se esta reabriendo.

Los desarrolladores deben mantenerse en el bug hell definido para el proyecto, por este motivo el tester deberá estar constantemente revisando bugzilla; y si fuera el caso de que existan anomalías con el bug hell, deberá informarle al encargado de QA para que este le informe al líder del proyecto.

Después de la etapa de desarrollo se entrara a la de estabilización, en esta parte desarrollo se encargara a resolver pulgas pendientes y nuevas solamente. El proceso de revisiones de pulgas es igual al de la etapa anterior.

Terminando estabilización, desarrollo liberará un build definitivo, el departamento de testing debe tomarlo y aplicarle las pruebas finales para asegurar la calidad del sistema, las pulgas que se reproduzcan en esta etapa las revisaran los coordinadores de QA y Desarrollo, y definirán si se deben resolver, en esta etapa los desarrolladores no deben tocar el código, estos solamente esperan a que se reproduzca una pulga que se deba resolver.

Durante el proyecto puede llegar el momento de que se estén realizando giras adonde el cliente, esto para verificar que la aplicación corra de la mejor manera en el ambiente del cliente, todas las pulgas que se reproduzcan durante estas revisiones se deberán agregar al bugzilla, pero se debe mencionar que fueron reproducidas donde el cliente para que no existan confusiones o para averiguar si el problema es en el ambiente propio del cliente.

Estrategias de QA

Actividad	Frecuencia	Descripción
Revisión del status de CMI en Bugzilla.	Semanales Antes de cada entrega	Tendremos reuniones de evaluación donde los desarrolladores y los tester deberán revisar el estado de los proyectos en bugzilla
Pruebas al Sistema	100% de los campos y pantallas de la interfaz del usuario 100% de los requerimientos especificados	El equipo de QA deberá generar y mantener un conjunto detallado de documentos de pruebas manuales para probar el sistema completo a través de la interfaz del usuario. Este plan será suficientemente detallado para que una persona pueda realizar repetidamente las pruebas desde la documentación de prueba y otros documentos

		asociados.
Pruebas de regresión	Correr todas las pruebas unitarias antes de cada reunión con el cliente Añadir nuevas pruebas cuando se añadan correcciones	Adoptaremos una política de volver a correr todas las pruebas superadas a un tiempo especificado. Esto ayudará a detectar regresiones (bugs que se pensaba ya reparados, pero que podrían aparecer otra vez).
Reportes de pulgas	El tester debe reportar las fallas de cada revisión	Cada pulga que se reproduzca en una aplicación se deberá reportar.

Metas de QA

Funcionalidad > Corrección

Corrección es el objetivo de calidad más básico. Significa que, cuando una entrada válida es dada y el sistema se encuentra en un estado válido y bajo una carga razonable, el comportamiento del sistema y sus resultados serán correctos.

Funcionalidad > Robustez

La robustez es la habilidad del sistema para manejar elegantemente entradas inválidas. No debería ser posible para ninguna entrada del usuario abortar el sistema o corromper la información, incluso si la entrada del usuario en anormal, inesperada, o maliciosa.

Funcionalidad > Exactitud

La exactitud se refiere a la precisión matemática de los cálculos hechos por el sistema. Cualquier sistema que realice cálculos numéricos debe considerar la exactitud; por ejemplo, aplicaciones financieras o científicas.

Funcionalidad > Compatibilidad

Los sistemas que aseguran que siguen estándares o proclaman cierta compatibilidad con sistemas existentes deberán adherirse a los protocolos relevantes de formatos de archivos y APIs. Se encontrarán ligas a los estándares relevantes en la parte de arriba de este documento.

Usabilidad > Comprensibilidad e Legibilidad

Los usuarios necesitan entender el sistema para usarlo. La metáfora básica deberá ser comprensible y apropiada para las necesidades del usuario. Algunas fallas en la comprensibilidad incluyen metáforas poco claras, etiquetas pobres o difíciles de ver, falta de retroalimentación para confirmar los efectos de las acciones del usuario, y falta de ayuda o ayuda inadecuada en línea.

Usabilidad > Intituividad y Memorabilidad

Cada interfaz de usuario contiene algunos detalles que los usuarios necesitarán aprender y recordar. Por ejemplo, Alt-A para abrir el menú "Archivo". Las reglas de la interfaz del usuario pueden hacer estos detalles más fáciles de aprender y recordar. Por ejemplo, la "A" está subrayada y, como regla, la primera letra es normalmente la tecla de acceso rápido.

Usabilidad > Eficiencia

Los usuarios deberían ser capaces de realizar tareas comunes con un esfuerzo razonable. Las tareas comunes deberían ser posibles con solo uno o dos pasos. La dificultad de cada paso debería ser también considerada Por ejemplo, ¿el usuario tiene que recordar una clave larga o dar click en un botón muy pequeño?

Usabilidad > Seguridad

Las personas somos propensos a equivocarnos, pero los efectos negativos de errores comunes debería se limitado. Por ejemplo, los usuarios deberían darse cuenta que un comando dado borrará su información, y debería ser consultado para confirmar acción, o tener una opción para deshacer.

Usabilidad > Consistencia y Familiaridad

Los usuarios deberían de ser capaces de utilizar su experiencia previa en otros sistemas similares. Es significa que los estándares para interfaces de usuario deberían ser seguidos, y las convenciones más comunes deberían ser usados cuando sea posible. Además, los elementos que aparecen en varias partes de la interfaz deberían ser usados de forma consistente a menos que otra convención

para UI tenga prioridad. Por ejemplo si la mayoría de los campos de entrada para moneda no requieren un signo de moneda, entonces el que lo necesita es un error de consistencia, a menos que exista una oportunidad real de que el usuario este trabajando con otro tipo de cambio en ese paso de la tarea.

Usabilidad > Satisfacción Subjetiva

Los usuarios deberían sentirse generalmente satisfechos con la interfaz del usuario. Esta calidad subjetiva se suma a las otras cualidades de la interfaz así como su estética.

Seguridad

El sistema debería permitir se usado solo por usuarios autorizados, y restringir el uso basado en permisos. El sistema no debería permitir a los usuarios saltarse las reglas de seguridad o utilizar agujeros en la seguridad. Por ejemplo, todas las entradas de los usuarios deberían ser validadas y cualquiera entrada maliciosa debería ser rechazada.

Confiabilidad > Consistencia sobre carga

Todo sistema tiene límites de capacidad. ¿Qué pasa cuando estos límites son excedidos? El sistema no debería jamás perder o corromper la información.

Confiabilidad > Consistencia bajo concurrencia

Los sistemas que permiten accesos simultáneos por usuarios múltiples, o los que usan concurrencia interna debería estar libres de de condiciones de carrera y bloqueo.

Confiabilidad > Disponibilidad bajo carga

Todo sistema tiene límites de capacidad. ¿Qué pasa cuando estos límites son excedidos? El sistema debería de continuar sirviendo aquellas solicitudes que es capaz de manejas. No debería caerse o detener el proceso de todas las solicitudes.

Confiabilidad > Longevidad

El sistema debería continuar operando tanto como lo necesite. No debería gradualmente terminarse los recursos disponibles. Ejemplos de defectos de

longevidad incluyen fugas de memoria o el agotamiento de espacio en discos con archivos de registro.

Eficiencia

Las superaciones del sistema deberían ejecutarse rápidamente, con un uso razonable de los recursos del equipo y de la red. Por ejemplo, si un usuario realiza una operación, esta debería ejecutarse eficientemente.

Escalabilidad

Escalabilidad es una cualidad general que se mantiene cuando el sistema continúa satisfaciendo sus requerimientos en situaciones en que sus parámetros se han incrementado. Por ejemplo, un servidor de archivos debería ser escalable a un gran número de usuarios, a archivos muy grandes, o a discos de muy alta capacidad. Algunos objetivos específicos de escalabilidad se enumeran abajo.

Escalabilidad > Desempeño bajo carga

Este es un tipo específico de objetivo de escalabilidad involucrado con el desempeño del sistema en momentos en que sus servicios son tienen muchas solicitudes de muchos usuarios.

Escalabilidad > Volúmenes grandes de datos

Este es un tipo específico de objetivo de escalabilidad involucrado con la habilidad del sistema para manejar grandes volúmenes de datos. Las operaciones deberían de continuar correcta y eficientemente aunque el tamaño del volumen de datos aumente. Más aún, la interfaz del usuario debería ser aún utilizable según la información presentada a los usuarios incremente en longitud.

Operabilidad

Las necesidades a largo plazo de los administradores del sistema deberían ser apoyadas confiablemente. Por ejemplo, ¿es el sistema fácil de instalar? ¿Puede el administrador recuperarse de una caída? ¿Existen suficientes datos en el registro para diagnosticar problemas en el campo? ¿Pueden los datos del

sistema ser respaldados sin bajas en el desempeño? ¿Puede el sistema ser actualizado de forma práctica?

Capacidad de mantenimiento > Comprensibilidad

¿Sería fácil para los (futuros) desarrolladores entender cómo el sistema funciona?

Capacidad de mantenimiento > Evolutividad

¿Puede el sistema ser fácilmente modificado y extendido en el futuro?

Capacidad de mantenimiento > Capacidad de prueba

¿Puede el sistema ser probado? ¿Los requerimientos especifican de forma precisa posibles entradas y resultados deseados? ¿Puede el sistema ser probado por partes? ¿Cuando se observan fallas, pueden ser rastreadas hasta las fallas en los componentes específicos (depuración)? ¿La realización de pruebas son prácticas con las herramientas de prueba disponibles?

Tipo de Pruebas

En el área de Aseguramiento de la Calidad del software, en cualquier empresa incursionada en éste ámbito de desarrollo de aplicaciones de software, cuenta con una serie de tipos de pruebas que los encargados de dicho departamento deben tomar en consideración y tener muy presente la estructura y especificación de cada una de las pruebas, con el fin de que en el momento de aplicar los tipos de prueba, los tester estén identificados sobre la forma de actuar en cada una de ellas.

Es por ello, que a continuación se determinan algunos de los más relevantes tipos de pruebas que son aplicados en el área de Aseguramiento de la Calidad en las empresas de desarrollo de software.

Se visualiza una breve descripción sobre el objetivo y la especificación detallada sobre cada una de las pruebas, los tipos de pruebas que se analizan a continuación son: las pruebas de unidad, pruebas de volumen, pruebas funcionales, pruebas de transacciones, pruebas de estrés, pruebas de carga y pruebas de regresión.

Prueba Unitaria

Objetivo de la Prueba:
Se focaliza en ejecutar cada módulo (o unidad mínima a ser probada, ej = una clase) lo que provee un mejor modo de manejar la integración de las unidades en componentes mayores. Busca asegurar que el código funciona de acuerdo con las especificaciones y que el módulo lógico es válido.

Descripción de la Prueba:

- Particionar los módulos en pruebas en unidades lógicas fáciles de probar.
- Por cada unidad hay que definir los casos de prueba (pruebas de caja blanca).
- Para esto los casos de prueba deben diseñarse de forma tal que se recorran todos los caminos de ejecución posibles dentro del código bajo prueba; por lo tanto el diseñador debe construirlos con acceso al código fuente de la unidad a probar.
- Los aspectos a considerar son los siguientes: Rutinas de excepción, Rutinas de error, Manejo de parámetros, Validaciones, Valores válidos, Valores límites, Rangos, Mensajes posibles.

Técnica:

- Comparar el resultado esperado con el resultado obtenido.
- Si existen errores, reportarlos.

Criterio de Completitud:

- Todas las pruebas planeadas han sido ejecutadas.
- Todos los defectos que se identificaron han sido tenidos en cuenta.

Pruebas de Volumen

Objetivo de la prueba:
Verificar que la aplicación funciona adecuadamente bajo los siguientes escenarios de volumen:

- Máximo (actual o físicamente posible) número de clientes conectados (o simulados), todos ejecutando la misma función por un periodo extendido.
- Máximo tamaño de base de datos y múltiples consultas ejecutadas simultáneamente.

Descripción de la prueba:
Las pruebas de volumen hacen referencia a grandes cantidades de datos para determinar los límites en que se causa que el Sistema falle. También identifican la carga máxima o volumen que el sistema puede manejar en un período dado. Por ejemplo, si el sistema está procesando un conjunto de registros de base de datos para generar un reporte, una prueba de volumen podría usar una base de datos de prueba grande y verificar que el sistema se comporta normalmente y produce el reporte correctamente. El objetivo de esta prueba es someter al sistema a grandes volúmenes de datos para determinar si el mismo puede manejar el volumen de datos especificado en sus requisitos.

Criterio de completitud:
Todas las pruebas planeadas han sido ejecutadas y los límites especificados en el sistema se han conseguido o excedido sin que el sistema falle.

Pruebas Funcionales

Están basadas en técnicas de caja negra, que es verificar la aplicación mediante la interacción con la aplicación vía GUI y analizar las salidas (resultados).

Objetivo de la prueba:
Se asegura el trabajo apropiado de los requisitos funcionales, incluyendo la navegación, entrada de datos, procesamiento y obtención de resultados.

Descripción de la prueba:
Las pruebas Funcionales deben enfocarse en los requisitos funcionales, las pruebas pueden estar basadas directamente en los Casos de Uso (o funciones de negocio), y las reglas del negocio. Las metas de estas pruebas son:

- Verificar la apropiada aceptación de datos.
- Verificar el procesamiento, recuperación y la implementación adecuada de las reglas del negocio.

Criterio de completitud:
Todas las pruebas planeadas han sido ejecutadas. Todos los defectos que se identificaron han sido resueltos.

Pruebas de Transacciones

Determina una unidad de la operación; o se terminan en su totalidad la acciones de las transacciones o ocurren fallos para que una determinada acción no concluya satisfactoriamente.

Objetivo de la prueba:
Representa cómo evoluciona el sistema. En su nivel más bajo consiste en definir cómo se insertan, borran y modifican ocurrencias de todos los objetos del sistema.

Descripción de la prueba:
Las transacciones se dan por una secuencia de operaciones de acceso a los datos que constituye una unidad lógica de ejecución. Constituyendo lo siguiente:

- Atomicidad: es considerado en varios puntos dentro del Aseguramiento de la Calidad, por ejemplo:
  - En sistemas de base de datos [todos los pasos en una transacción tiene éxito o no se pudo concluir la transacción].
  - En sistemas de programación y sistemas operativos [toma efecto en un cierto instante durante la ejecución].
  - Esta presente en fallas del CPU.
- Consistencia: se asegura de que cualquier cambio a los valores en un caso sea constante con los cambios a otros valores en el mismo caso.
- Aislamiento: hacer que las operaciones en una transacción aparecen aisladas del resto de las operaciones.
- Persistencia: garantiza que una vez que se haya notificado al usuario el éxito, la transacción va a persistir.

Criterio de completitud:
Todas las transacciones realizadas concluyeron con éxito. Se identificaron las transacciones que fallaron y se resolvió tal falla.

Pruebas de Stress

El acto de asegurar que el sistema funciona como se espera bajo grandes volúmenes de transacciones, usuarios, carga, etc., además de identificar y documentar las condiciones bajo las cuales el sistema falla.

En ciertos sistemas es conveniente saber hasta dónde aguantan, bien por razones internas (ej: ¿hasta cuantos datos podrá procesar?), bien por externas (ej: ¿es capaz de trabajar con un disco al 90%?, ¿aguanta una carga del CPU al 90%?, etc.). Por tanto, es necesario el análisis de las Pruebas de Stress.

Objetivo de la prueba:
Verificar que el sistema funciona apropiadamente y sin errores, bajo las siguientes condiciones de stress:

- Memoria baja o no disponible en el servidor.
- Máximo número de clientes conectados o simulados (actuales o físicamente posibles).
- Múltiples usuarios desempeñando la misma transacción con los mismos datos.
- El peor caso de volumen de transacciones.

Descripción de la prueba:
Las pruebas de stress se proponen encontrar errores debidos a recursos bajos o completitud de recursos. Poca memoria o espacio en disco puede revelar defectos en el sistema que no son aparentes bajo condiciones normales. Otros defectos pueden resultar de incluir recursos compartidos, como bloqueos de base de datos o ancho de banda de la red. Las pruebas de stress identifican la carga máxima que el sistema puede manejar. El objetivo de esta prueba es investigar el comportamiento del sistema bajo condiciones que sobrecargan sus recursos. No debe confundirse con las pruebas de volumen: un esfuerzo grande es un pico de volumen de datos que se presenta en un corto período de tiempo. Puesto que la prueba de esfuerzo involucra un elemento de tiempo, no resulta aplicable a muchos programas, por ejemplo a un compilador o a una rutina de pagos. Es aplicable, sin embargo, a programas que trabajan bajo cargas variables, interactivos, de tiempo real y de control de proceso. Aunque muchas pruebas de esfuerzo representan condiciones que el programa encontrará realmente durante su utilización, muchas otras serán en verdad situaciones que nunca ocurrirán en la realidad. Esto no implica, sin embargo, que estas pruebas no sean útiles. Si se detectan errores durante estas condiciones "imposibles", la prueba es valiosa porque es de esperar que los mismos errores puedan presentarse en situaciones reales, algo menos exigentes.

Criterio de completitud:
Todas las pruebas planeadas ha sido ejecutadas y excedidas sin que el sistema falle o si las condiciones en que el sistema falle ocurren por fuera de las condiciones especificadas.

Pruebas de Carga

Objetivo de la Prueba:

Verificar el tiempo de respuesta del sistema para transacciones o casos de uso de negocios, bajo diferentes condiciones de carga.

Descripción de la Prueba:

Las pruebas de carga miden la capacidad del sistema para continuar funcionando apropiadamente bajo diferentes condiciones de carga. La meta de las pruebas de carga es determinar y asegurar que el sistema funciona apropiadamente aún más allá de la carga de trabajo máxima esperada. Adicionalmente, las pruebas de carga evalúan las características de desempeño (tiempos de respuesta, tasas de transacciones y otros aspectos sensibles al tiempo).

Técnica:

- Use los scripts desarrolladas para Pruebas del Negocio.
- Modifique archivos de datos (para incrementar el número de transacciones o veces que cada transacción ocurre).

Criterio de Completitud:

Múltiples transacciones, múltiples usuarios. Se completaron las pruebas de los scripts sin ninguna falla y dentro del tiempo esperado.

Consideraciones Especiales:

- Las pruebas de carga deben ser ejecutadas en una máquina dedicada o en un tiempo dedicado. Esto permite control total y medidas precisas.
- La Base de datos utilizada para pruebas de desempeño debe ser de un tamaño real o proporcionalmente más grande que la diseñada.

Prueba de Regresión

Objetivo de la Prueba:
Determinar si los cambios recientes en una parte de la aplicación tienen efecto adverso en otras partes.

Descripción de la Prueba:
En esta prueba se vuelve a probar el sistema a la luz de los cambios realizados durante el debugging, mantenimiento o desarrollo de la nueva versión del sistema buscando efectos adversos en otras partes.

Técnica:

- La prueba de regresión es una nueva corrida de casos de prueba previos.
- Se requiere de políticas para ser creada la prueba de regresión y decidir qué casos de prueba incluir, para probar eficientemente.
- La prueba de regresión es un buen candidato para automatización. Desde que estas pruebas se repiten una y otra vez, las herramientas para minimizar el esfuerzo del trabajo son útiles.
- La prueba de viejas funcionalidades es más importante que la de nuevas funcionalidades.
- Aquellos casos de uso (y los casos de prueba asociados) que descubren defectos tempranamente deben ser incluidos en la prueba de regresión.

Criterio de Completitud:

- Todas las pruebas planeadas han sido ejecutadas.
- Todos los defectos que se identificaron han sido tenidos en cuenta.

Proceso Deseado

En esta sección se introduce al lector a la información del proceso deseado con la metodología a implantar, dando a conocer los aspectos y conceptos relevantes de la herramienta TestComplete para la automatización de las pruebas de software.

TestComplete – Black Box Testing

La prueba de la caja negra maneja su estructura basada en los requisitos. Por tanto, TestComplete no escapa de la metodología que contempla las pruebas de caja negra, ya que su análisis de las pruebas se rige bajo este tipo de estructuras.

La prueba de caja negra, no asume ningún conocimiento del código o del diseño del sistema para poder responder a los estímulos de las entradas. Lo que hace es probar todas las combinaciones posibles de las acciones del usuario final. Implica probar las interfaces externas para asegurarse de que el código resuelve los requisitos funcionales y no funcionales. La caja negra se puede empezar a utilizar inmediatamente después de que están disponibles los requisitos y las especificaciones funcionales.

Los tipos de pruebas que caben dentro de la estrategia de la caja negra son:

- Pruebas funcionales [comprobar si el uso de los requisitos funcionales se comporta según lo esperado].
- Pruebas de estrés [pruebas de carga pesada, como valores numéricos complejos, número grande de entradas].
- Pruebas de recuperación [comprobar que tan rápido el proceso puede recuperarse contra cualquier tipo de desplome o falta del hardware].

- Pruebas de volumen [comprobar la cantidad de datos que se pueden procesar, para conocer las limitaciones extremas del sistema].
- Pruebas de aceptación de usuarios [comprobar que el software que se entrega al usuario cumple con las expectativas].
- Pruebas de humo [probar el sistema constantemente].
- Pruebas alfa [se lleva a los usuarios al centro de desarrollo para que analicen el producto, si existe algún error, se arregla el problema].
- Pruebas beta [se le lleva el producto al sitio de trabajo del usuario, si existe algún error, se arregla el problema].

Proceso de prueba de la caja negra

- Crear los planes de prueba [especificando la prioridad para la prueba].
- Probar las interfaces externas [probar las entradas usando escenarios automatizados de la prueba].
- Realizar la prueba de la carga [cargar la prueba en el bloque, para asegurarse de que resuelva todos los objetivos del funcionamiento que se indiquen como requisitos].
- Realizar la prueba de stress [realizar las pruebas de carga pesada].
- Realizar la prueba de seguridad [mantener al margen todo lo que se refiere a seguridad del sistema].

Sobre TestComplete

TestComplete es un ambiente de prueba automatizado para Win32 y aplicaciones .NET. Proporciona extensa ayuda para pruebas de páginas Web y proyectos creados in Microsoft Visual C++, Visual Basic, Borland Delphi, C++Builder, Java y herramientas de desarrollo .NET.

TestComplete maneja scripts para probar usos externamente. Es orientado a pruebas de unidad y a pruebas funcionales, además de proporcionar una ayuda excelente para las pruebas de regresiones, evitando un gasto de coste y tiempo en departamentos de Aseguramientos de la Calidad.

Por su parte TestComplete, maneja el uso de grabaciones de pruebas sobre páginas Web y sobre pruebas de carga, estrés y escalabilidad en aplicaciones http. Facilitando una expresión excelente de análisis de resultados en cuanto este tipo de pruebas, graficando y llevando control sobre el uso de peticiones hacia el servidor en la automatización de este tipo de aplicaciones.

TestComplete [Enterprise Edition]

Permite la definición y ejecución de:

- Pruebas Unitarias
- Pruebas Funcionales
- Pruebas Distribuídas
- Pruebas de Regresión
- Pruebas de rendimiento HTTP

Para cualquier tipo de aplicación:

- Win32
- .Net
- Java
- Web

Funcionalidades:

HTTP Load Testing [Prueba de Carga]

- Pruebas de carga, stress y escalabilidad en aplicaciones Web
- Graba peticiones HTTP y permite reproducirlas varias veces a través de usuarios virtuales.

- Con HTTP Load Testing Remote Agent se pueden simular múltiples usuarios virtuales en múltiples PC's.

Distributed Testing [Prueba Distribuida]

- Gestión de la sincronización de proyectos de pruebas ejecutándose en múltiples PC's.
- Los test se pueden ejecutar: simultáneamente o sincronizados de manera que podamos crear sofisticados escenarios que representen la realidad.

Object-Driven Testing

- Permite la definición y prueba de clases sin necesidad de escribir código.

Compressing Test Results [Resultados de la Prueba de Comprensión]

- Permite comprimir y archivar los resultados de los test según sea necesario.
- Utilizando su herramienta de compresión favorita, desde la UI.

Llamadas a funciones desde .NET assemblies

- Permite llamar a funciones que residan en aplicaciones .NET.

Testing CORBA Objects [Objetos de prueba de CORBA]

- TestComplete puede acceder a objetos CORBA situados en una máquina local o en una red.
- Los objetos se pueden explorar en el panel Object Browser, y trabajar con ellos en los scripts.

Soporte de múltiples compiladores

- Visual Basic, Visual C++, Delphi, C++ Builder, Java, Visual Studio .NET, Visual FoxPro.

Scripting por grabación o por codificación

- Se puede construir test rápidamente y probarlos.
- O usar una mezcla de: acciones grabadas, código, llamadas a librerías de TestComplete.

Web Page Testing

- TestComplete usa el modelo de objetos HTML.
- Genera scripts de test similares a VBS (Visual Basic Scripts) O JS (Java Script).
- Añade métodos, propiedades y eventos específicos a los procesos de Internet Explorer.
- Estos métodos permiten navegar a la página deseada, retrasar la ejecución del script hasta que la página se ha cargado, etc.

Características de las últimas versiones

Migración de Proyectos

TestComplete 4 permite ejecutar proyectos grabados en versiones anteriores de él mismo. Por ejemplo si existe ya una grabación en TestComplete 3, TestComplete 4 lo carga y automáticamente realiza los cambios necesarios para que el proyecto tenga la posibilidad de ejecutarse en TestComplete 4.

Migración de Sintaxis

Comprueba automáticamente la sintaxis de los scripts y recolecta las declaraciones erróneas que se encuentran en la sintaxis de los scripts de las versiones anteriores de TestComplete.

Mientras se va convirtiendo una grabación en una versión anterior de TestComplete, TestComplete 4 va analizando la sintaxis de los scripts que se encuentran en dicha grabación. Si existen declaraciones erróneas,

TestComplete 4 las guarda y una vez que se convirtió la grabación, la herramienta envía en forma de mensajes la información de las declaraciones que se encuentran con errores, para que el probador las verifique y realice los cambios necesarios, con el fin de obtener la grabación tal y como estaba grabada en la versión anterior de TestComplete.

Migración de Ítems

Cuando se abre un proyecto creado en una versión anterior de TestComplete en TestComplete 4, el proyecto se convierte en un nuevo formato. Al convertirse, TestComplete crea automáticamente Ítems necesarios del proyecto y cambia algunas declaraciones del script.

TestComplete convierte automáticamente las llamadas de las funciones siguientes [se indican únicamente 2 ejemplos para denotar los posibles cambios]:

Antes	Después	Descripción
Sys.Delay	Delay	El método Delay se ha movido del BuiltIn.
Sys.GetOleObject	Sys.OleObject	Se ha quitado el método de Sys.GetOleObject.

Compatibilidad de uso de Plug-In

La ventaja principal de la compatibilidad de los Plug-In es que deja funcionar los proyectos de TestComplete 3 con cambios mínimos en TestComplete 4. Se puede realizar los cambios deseados en cualquier código o en las rutinas de los scripts creados en TestComplete 3.

Detalles en TestComplete 4

Panel	Descripción
Project Explorer	Se visualiza el contenido de la habitación del proyecto, de los proyectos dentro de él, y los ítems del proyecto.
Workspace	Contiene un sistema de las páginas tabuladas, en las cuales se pueden modificar características de los ítems del proyecto.
Log	Filtros y resultados de las pruebas.
Object Browser	Se visualizan todos los usos bajo prueba que funcionan en la máquina. Todo lo que esta en el browser se puede utilizar en una prueba.
Bookmarks	Muestra una lista del sistema de los bookmarks en el código del script. Utilizados para corregir los scripts.
Breakpoints	Utilizado para eliminar errores de los scripts.
Call Stack	Utilizado para eliminar errores de los scripts, durante la ejecución de los scripts.
Code Explorer	Navega a través de los scripts que se guardan dentro del proyecto. Utilizado para corregir y eliminar errores de los scripts.
Watch List	Utilizado para eliminar errores de los scripts. Variable de las demostraciones y valores de característica durante el funcionamiento de las pruebas.

Creación de un Nuevo Proyecto

Creación del proyecto

La *figura 1* muestra el inicio de la creación de un proyecto en TestComplete 4. En donde se eligen los siguientes pasos para su elaboración:

- Se elige el tipo de template que se utilizará para las grabaciones.
- Seguido se especifica el nombre del proyecto.
- Se especifica el nombre de la habitación del proyecto.
- Se escoge el tipo de lenguaje del cual se requiera por parte del usuario, preferible para realizar las grabaciones.

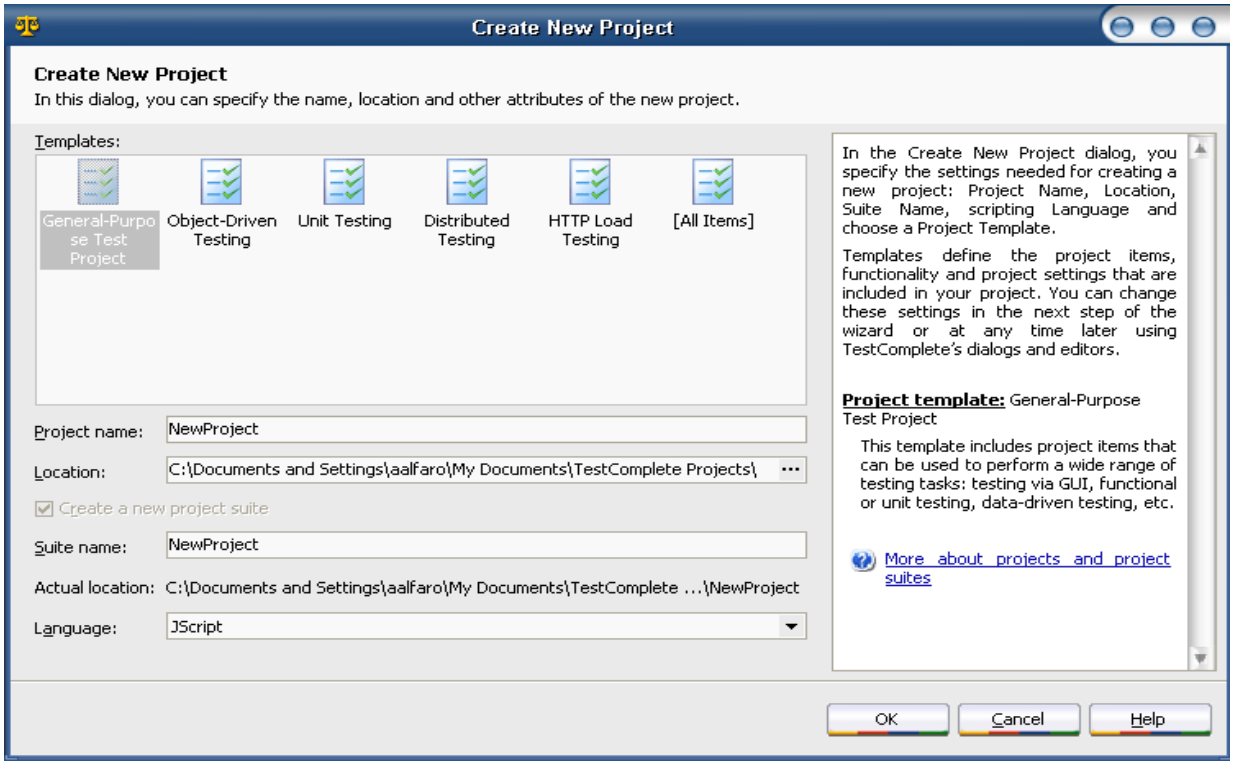


FIGURA 1. TIPO DE PROYECTO A ESCOGER.

En la *figura 2* se especifican los ítems que serán incluidos en el proyecto.

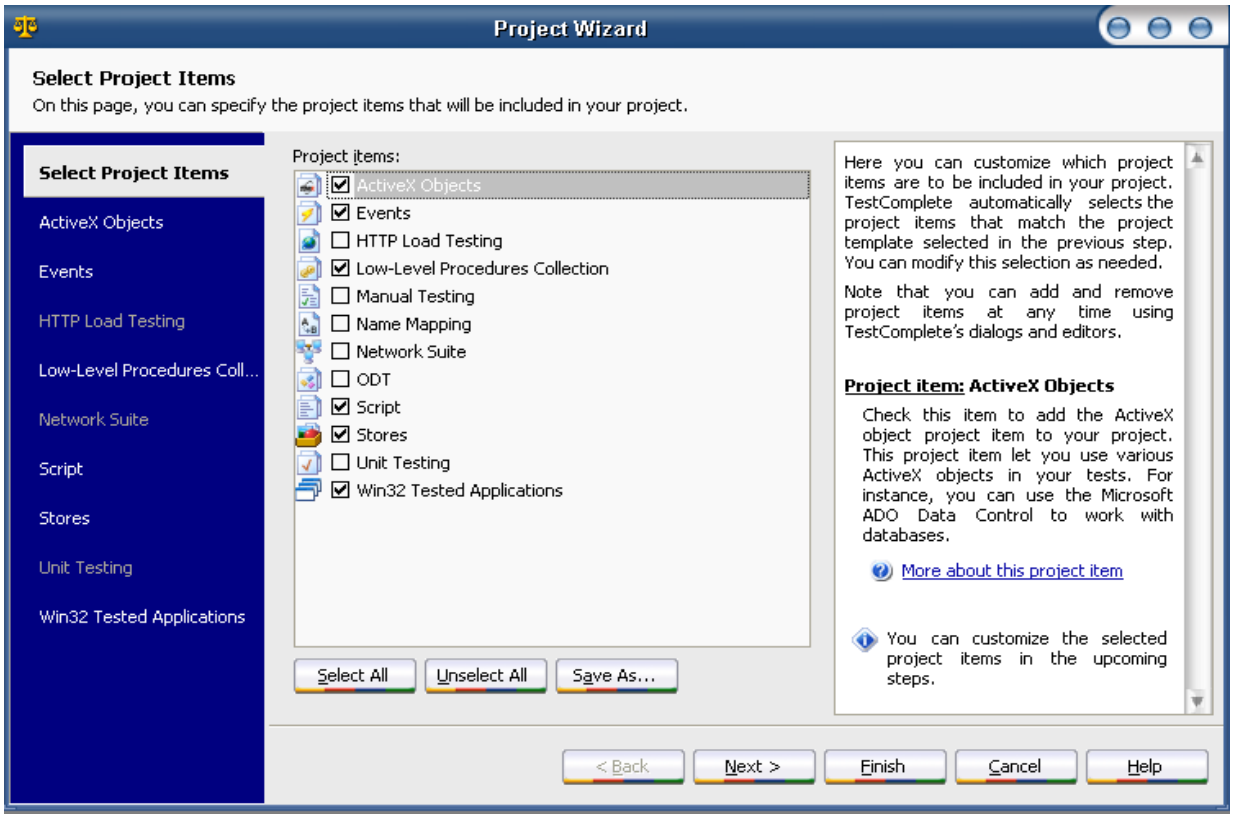


Figura 2. Ítems para un nuevo proyecto.

Definir usos de la prueba

Una vez que se ha creado el proyecto en el Project Explorer aparece todo el árbol que indica todas las acciones que se crearon y que se escogieron. Donde dichas acciones pueden ser manipuladas para que realicen acciones que el usuario quiera ejecutar; acciones como renombrar, borrar, correr, salvar, etc. Se puede determinar en la *figura 3* dichas acciones.

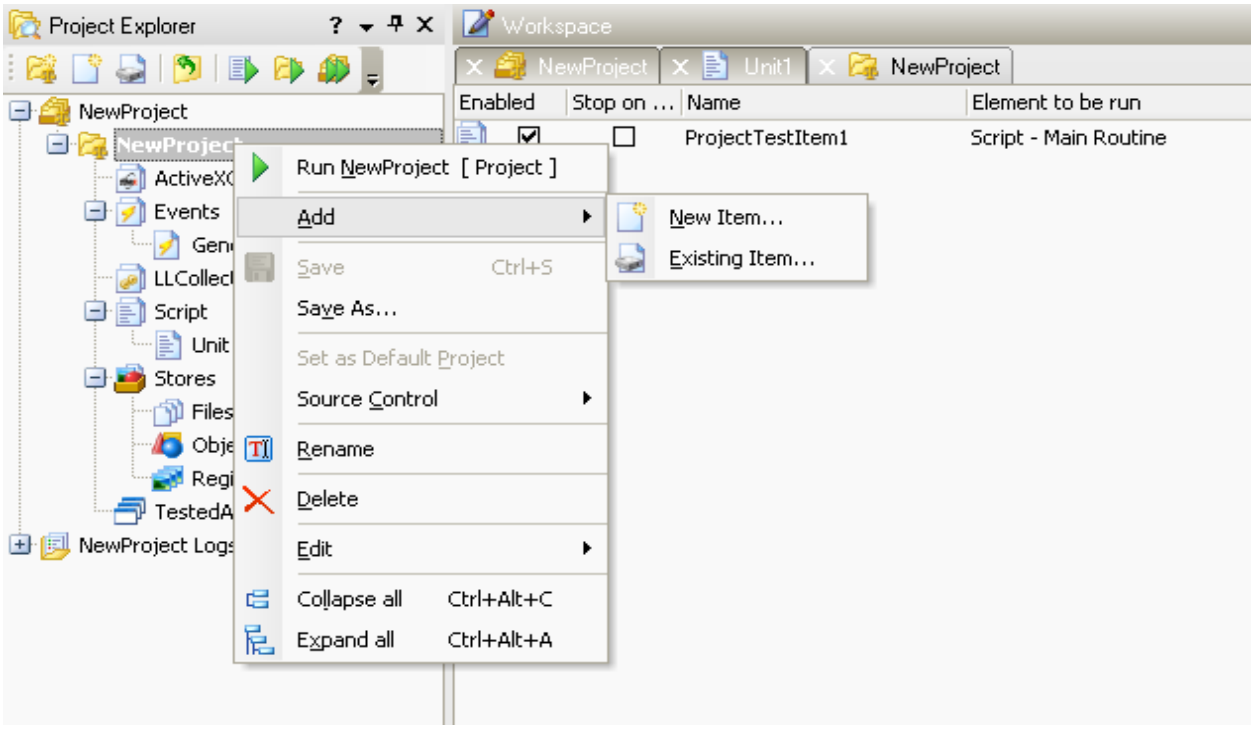


FIGURA 3. ACCIONES DE USO EN UN PROYECTO.

Explorar los usos en el Object Browser

Antes de una grabación o un funcionamiento de un script, se puede explorar qué características, campos, métodos y acontecimientos de uso de objetos de TestComplete se tiene acceso; esto se visualiza en la *figura 4.*

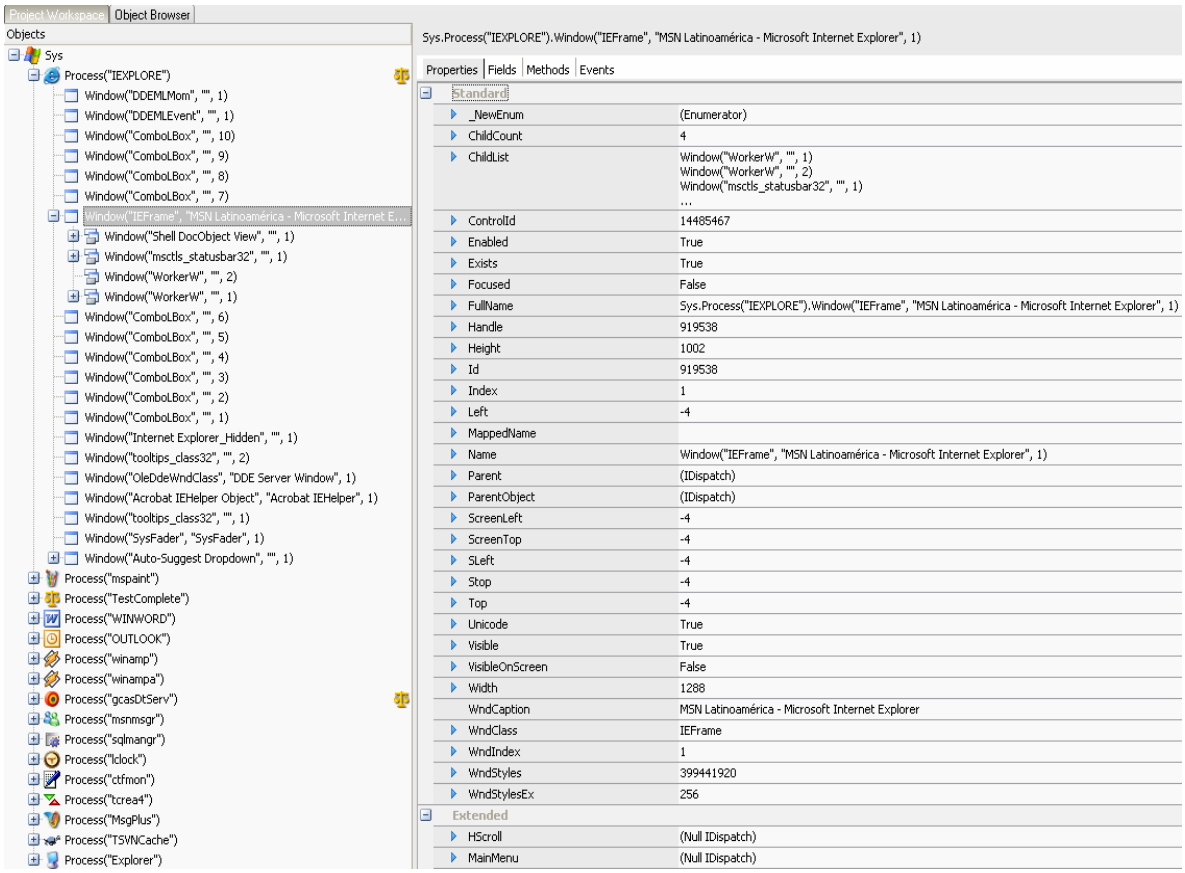


FIGURA 4. VISUALIZAR LOS USOS DE OBJETOS.

Crear las Pruebas

Antes de crear una prueba, se debe primero planearla; para esto se deben tomar varios puntos en consideración, los cuales se analizan a continuación:

- *Definir la meta de la prueba* [definir que funcionalidad de los usos se va a probar]: es mejor crear una prueba simple que se dirija a un único objetivo, así una vez que se tienen varias pruebas simples, se pueden organizar en una sola prueba grande.
- *Definir los pasos del plan de prueba* [decidir qué acciones realizará la prueba]: dependiendo del propósito de la prueba y de la naturaleza del uso bajo prueba.

- *Comprobar los resultados de la prueba* [determinar si la prueba pasó con éxito o falló]: después de que terminen las acciones de prueba, la prueba debe comprobar los resultados contra las salidas previstas y decidir si la prueba es acertada o no.
- *Registrar los resultados* [determine como registrar el resultado de la prueba].

Una vez planeada la prueba, se sigue con el paso de crearla, siguiendo lo siguiente:

- Definir las salidas previstas.
- Definir la entrada correspondiente que alimentará la prueba.
- Resultado de la prueba.
- Comparar el resultado con la salida prevista.
- Definir qué hacer si la comparación falla.

Grabación de Scripts

Cuando se empieza las grabaciones de scripts de una aplicación, existen dos puntos importantes, que no son necesarios realizar pero que si pueden ahorrar mucho trabajo más adelante:

- Existen botones para poder tomar fotografías a las pantallas, esto con el fin de que en algún determinado momento del análisis de las pruebas, poder comparar fotografías con nuevas pantallas para determinar si ha sufrido algún cambio (botones, áreas de texto, etc.) y ahorrarse el tiempo de no probar scripts que han sufrido cambios de interfaz.
- Existe la opción con botones dentro de la herramienta, de poder agregar comentarios a los scripts, comentarios que más adelante serán de gran ayuda, ya que pueden llegar a determinar el qué hace cierto sector de la grabación del script por ejemplo. Por eso se recomienda que estos

comentarios se hagan lo más serios que se pueda, ya que se hacen en el momento de las grabaciones de los scripts, y es el momento en el que se tienen las cosas claras de que es lo que esta sucediendo actualmente, más adelante entre tanta grabación se pueden olvidar situaciones de relevancia, que al estar documentadas pueden ser de gran ayuda para saber que es lo que esta sucediendo.

TestComplete genera las grabaciones de los scripts en lenguajes tales como: VBScript, JScript, DelphiScript, C++Script o C#Script.

TestComplete crea automáticamente un nombre para el nuevo script, ese nombre aparece como TestN, donde N es un número que aumenta conforme aumente la cantidad de grabaciones de scripts. Ese nombre puede ser editado por el usuario.

TestComplete no registra la trayectoria exacta del Mouse en ciertos lugares o el tiempo transcurrido en cada Tab dentro de la pantalla, con el fin de no llenar la grabación del script con datos que al final no serán necesarios para las pruebas.

TestComplete no registra los delay entre las acciones realizadas por el usuario, haciendo que más rápido la creación de los scripts y disminuyendo el tamaño total del mismo. Por otra parte, si el usuario lo desea la opción del delay puede ser registrada, se obtendrá un tamaño más grande del script pero se ejecutará a la misma velocidad en que fue grabada.

Editar los scripts

TestComplete proporciona la opción de poder editar los scripts que han sido grabados. Así, el usuario puede realizar los cambios necesarios en el código de los scripts grabados en el momento que lo crea necesario.

Especificar el orden de ejecución de las pruebas

Una vez creados los ítems deseados para las pruebas, se puede especificar el orden de ejecución, donde se presenta un árbol con todas las estructuras existentes de las pruebas y se pueden modificar según lo necesitado por el usuario. En la *figura 5* se denota la forma en que se puede especificar el orden de la ejecución de las pruebas.

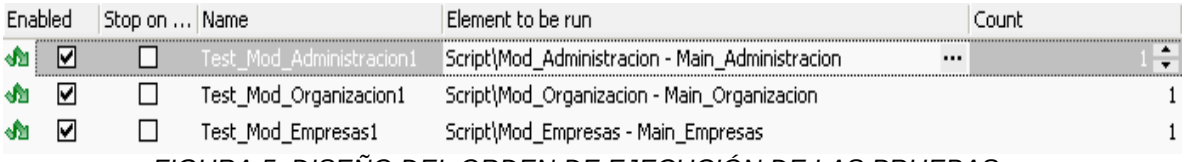


FIGURA 5. DISEÑO DEL ORDEN DE EJECUCIÓN DE LAS PRUEBAS.

Correr las Pruebas

Antes de poder correr las pruebas, se necesitan tener listos algunos pasos preliminares listos:

- Definir el propósito de la prueba.
- Crear los ítems de la prueba.
- Especificar el orden de ejecución de los ítems de la prueba.

Una vez que esos pasos están listos, se puede proceder a ejecutar la ejecución de la prueba, dentro de la herramienta se encuentran los botones que hacen posible tal acción. Después de que la prueba fue ejecutada, se puede visualizar en la sección de *Test Log* los resultados que TestComplete proporciona a partir de lo sucedido con la ejecución de la prueba.

Análisis de los resultados de la prueba

En la *figura 6* se puede notar todas las acciones que ocurrieron después de que se ejecuto una prueba. Por ejemplo se puede denotar las unidades que no tuvieron errores o las que si; los warnings, eventos, etc., después de una corrida. Una vez vistos, se analiza el tipo de causa que ocasiono el error para poder determinar un criterio sobre que se debe hacer después de corrida la prueba.

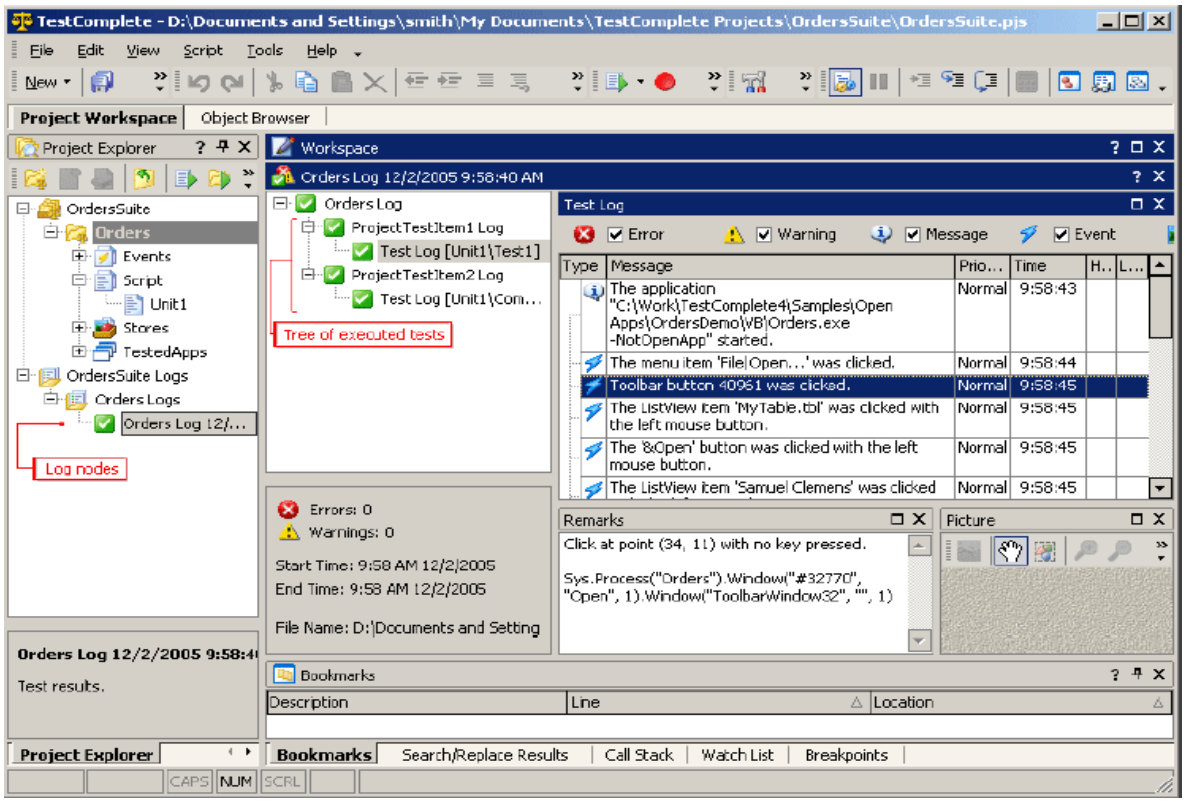


FIGURA 6. ANÁLISIS DE LOS RESULTAS DE LAS PRUEBAS.

Creación de un Nuevo Proyecto de Carga

A continuación se describen los pasos para crear una prueba de carga, y se tomará en cuenta algunas acciones que se deben realizar para llevar a cabo este tipo de pruebas.

Es importante antes de crear la prueba de carga, de configurar el explorador que se va a utilizar (Internet Explorer en este caso, puede ser cualquier otro explorador), y configurar TestComplete para las pruebas de carga.

Configuración del explorador:

- Accedemos a la siguiente opción del explorador *tools/Internet options/connections/LAN settings*, y configuramos lo siguiente:

    - Habilitar *use automatic configuration script* y en el campo de la dirección, digitamos el nombre del servidor donde se encuentra instalado TestComplete, en este caso buffalo.

    - Habilitar *use a proxy Server for your LAN.* En el campo de la dirección se digita el nombre del servidor donde se encuentra instalado TestComplete, es decir, buffalo. En el campo del puerto digitamos 8080.

    - Deshabilitar la opción *bypass proxy Server for local address.*

A continuación en la *figura 7* se presenta la configuración que debe tener el explorador a utilizar, en este caso es Internet Explorer. Se puede visualizar los campos que deben llenarse, las opciones que deben estar activadas y las que deben estar desactivadas.

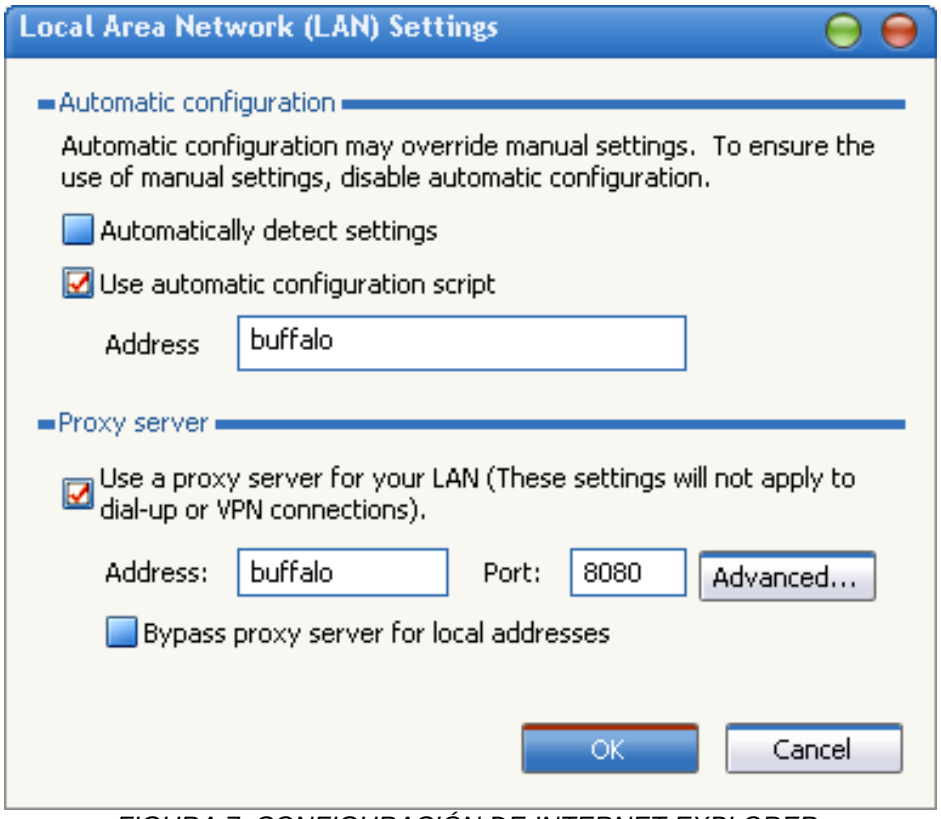


FIGURA 7. CONFIGURACIÓN DE INTERNET EXPLORER.

Configuración de TestComplete:

- Accedemos a la siguiente opción de TestComplete *tools/options/engines/http load testing*, y configuramos lo siguiente:

  - En el campo de Proxy Port digitamos el número 9999 (el Proxy Port viene configurado por defecto por TestComplete).

  - En el campo de RA Port digitamos el número 9091 (puerto que el agente remoto utiliza para comunicarse con TestComplete, configurado por defecto).

  - En el campo del http Proxy Host digitamos la dirección IP de la máquina en la que TestComplete se encuentra instalado, en este caso 192.168.0.103. Específica el servidor Proxy que el agente remoto utiliza para conectarse con el sitio de prueba Web y con la computadora donde TestComplete esta funcionando.

  - En el campo del http Proxy Port digitamos el puerto 8080, en donde especifica el puerto en el que el agente remoto utiliza para conectarse con el servidor Proxy especificado por el Proxy Host.

- El resto de campos dentro de esta configuración quedan a como están, por configuración de TestComplete.

La anterior configuración se ve representada en la *figura 8*, en donde se denotan los campos que deben ser configurados en la herramienta TestComplete, para lo que son las pruebas de carga, estrés y escalabilidad.

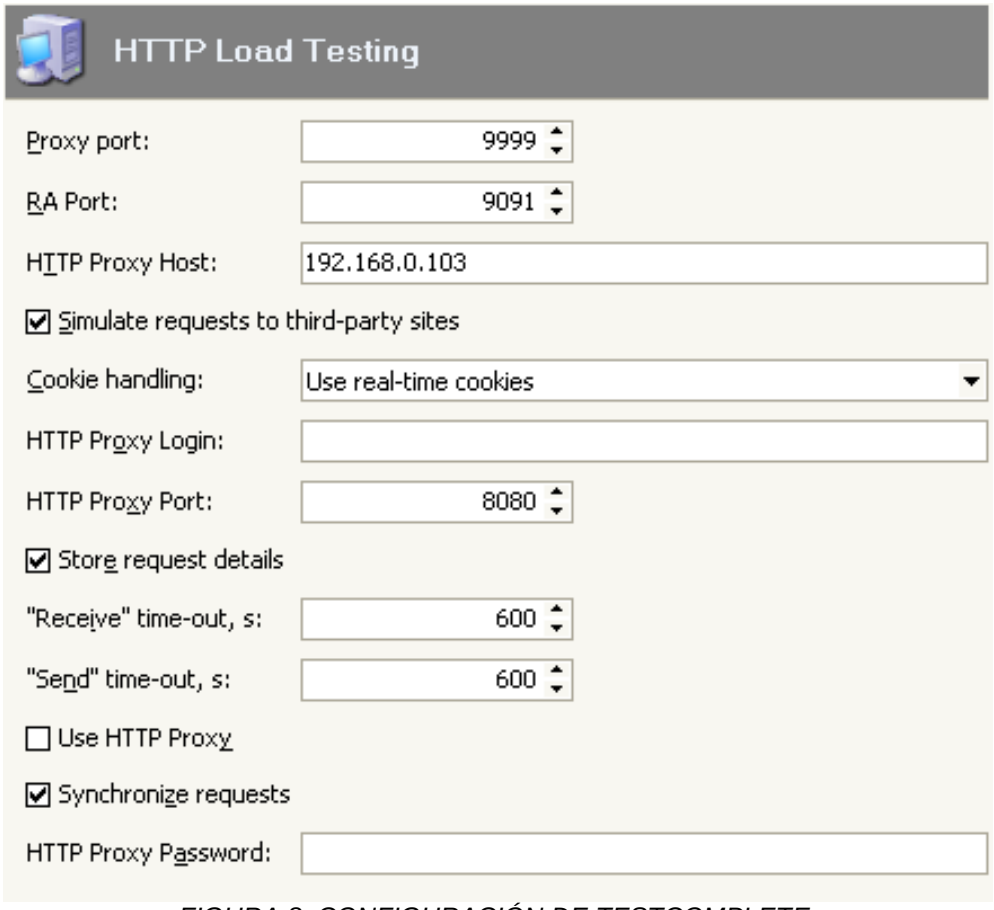


FIGURA 8. CONFIGURACIÓN DE TESTCOMPLETE.

Una vez configurado esas dos acciones, se prosigue con los siguientes pasos que van a permitir al usuario crear una prueba de carga, estrés y escalabilidad para una determinada aplicación.

A continuación se presentan los pasos principales para dicha creación del tipo de prueba.

- Se debe crear un proyecto http load testing. Donde el wizard nos presenta la opción para marcar el tipo de proyecto que deseamos, el espacio para asignar un nombre al proyecto, escoger la dirección donde queremos guardar el proyecto, opción si se quiere crear una habitación sobre ese proyecto (no necesario), y se escoge el tipo de lenguaje en que estarán escritas las grabaciones. *Ver Figura 9.*

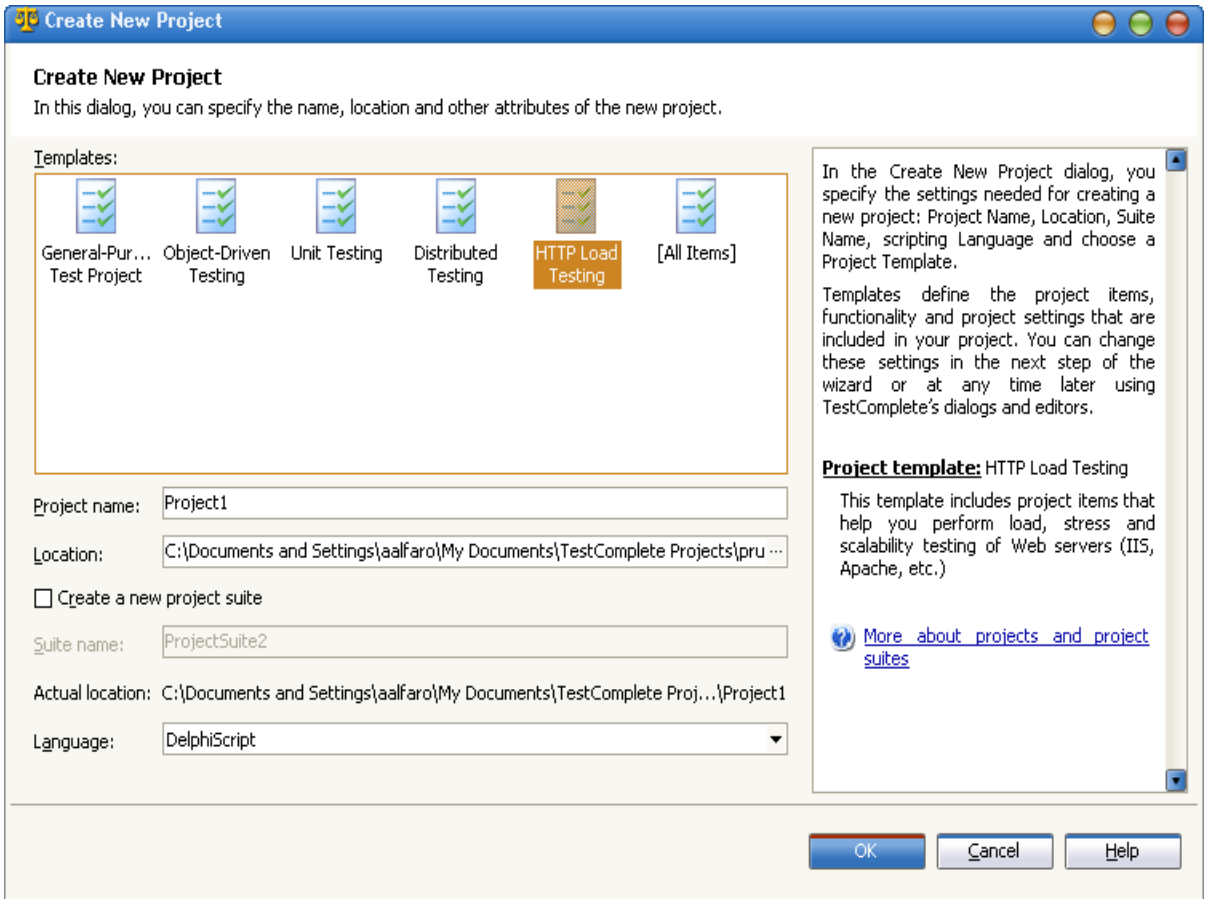


FIGURA 9. PROYECTO HTTP LOAD TESTING.

- Seguido de la creación de un proyecto, el siguiente wizard nos presenta los ítems que podemos escoger para que sean parte de nuestro proyecto. En este caso, por defecto se asigna los eventos (obligatoriamente deben aparecer), se asigna el http load testing el cual es fundamental para la grabación de las pruebas de carga, y la opción de script donde se registra en forma de código las pruebas de carga que se vayan haciendo sobre una aplicación. *Ver Figura 10.*

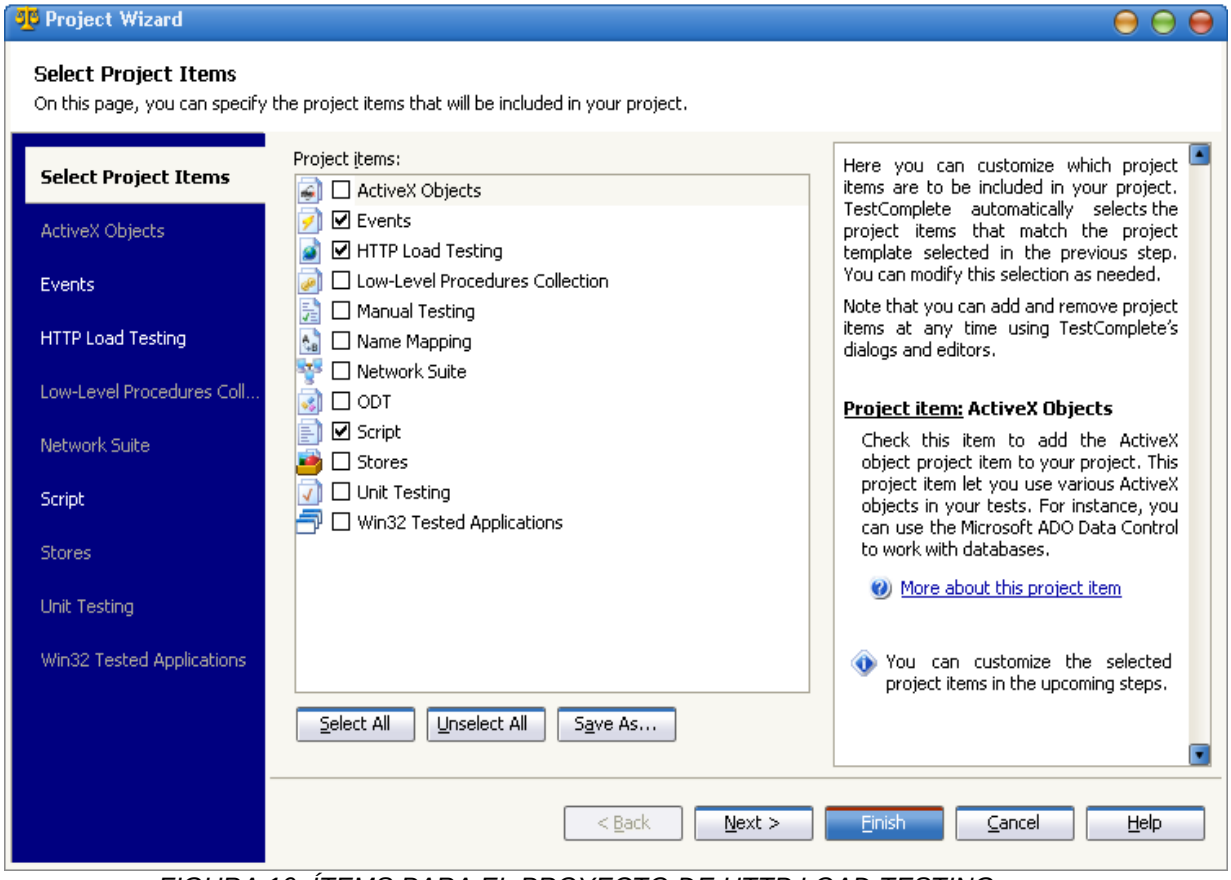


FIGURA 10. ÍTEMS PARA EL PROYECTO DE HTTP LOAD TESTING.

- Una vez creado el proyecto, dentro de TestComplete aparecerá toda la estructura que compone una prueba de carga. Aparece un árbol con carpetas que tiene diferentes cometidos:

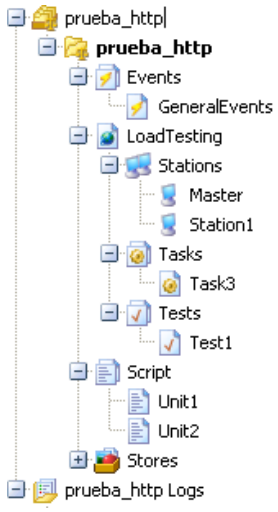


FIGURA 11. ESTRUCTURA DE UNA PRUEBA DE CARGA.

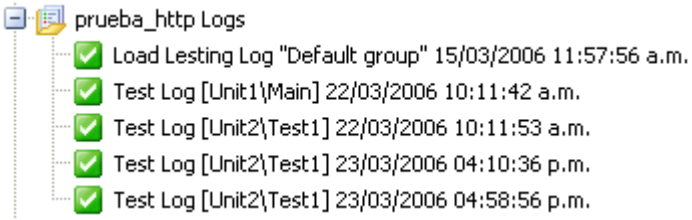


FIGURA 12. EJEMPLOS DE RESULTADOS DE PRUEBAS DE CARGA.

- Como raíces principales aparece el proyecto como tal (*prueba_http*) y la sección donde se lleva el registro de todo tipo de pruebas que se requieran hacer (*prueba_http logs*). *Ver Figura 11.*

- En la subraíz (***prueba_http***) se denotan los ítems que fueron escogidos en el wizard de la *figura 10.* Los cuales se describen a continuación:
    - Se encuentra la carpeta que contiene los eventos (events).
    - Se encuentra la carpeta que contiene todas las acciones de la prueba de carga, entre ellas: las estaciones [sitios de trabajo],

las tareas [especifica la tarea asignada a un usuario dado], y las pruebas [creación y asignación de los usuarios virtuales].

- Se encuentra la carpeta que contiene los scripts, los cuales se dividen en unidades, donde se podrán hacer grabaciones en el lenguaje escogido en esas unidades de una determinada aplicación.

- En la parte inferior del árbol de la *figura 11,* se puede denotar la carpeta que registra los distintos tipos de pruebas que se van realizando (*prueba_http logs*). En la *figura 12* se ven ejemplos de pruebas que fueron realizadas, en donde la primera es una prueba de carga (load testing), y las siguientes son pruebas de unidad de una aplicación. Dichas pruebas se encuentran bien identificadas con los nombres de la unidad y la prueba que fueron ejecutadas (ej. Unit2/Test1), y la fecha y hora de cuando se dio la ejecución de la prueba (ej. 23/03/2006 04:58:56 p.m.).

- Analizaremos ahora el contenido de las secciones que se encuentran dentro de la prueba de carga (LoadTesting: stations, task, test). *Ver Figura 11.*

- **Tests**

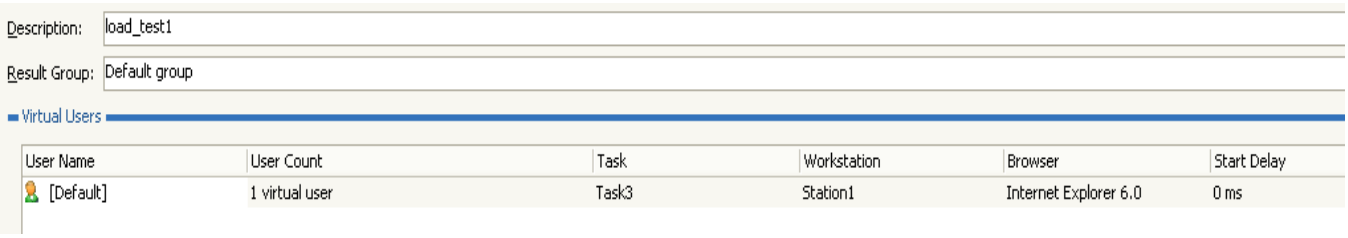


FIGURA 13. REPRESENTACIÓN DE UNA PRUEBA DE CARGA CON USUARIOS VIRTUALES.

El contenido presente dentro de una prueba (ej. Test1) es el siguiente: *ver Figura 13.*

- Contiene un campo de descripción con el fin de digitar el nombre que se le dará a esa prueba.
- El siguiente campo es para asignar al los usuarios virtuales a un grupo de trabajo, en este caso TestComplete genera un grupo por defecto (default group).
- En la parte inferior es donde se da la creación de usuarios virtuales, los cuales al ser creados se le asignan características importantes que definen propiedades para ser parte de las pruebas de carga, entre ellas se encuentran: el nombre de usuario (user name), se especifica el o los usuarios que formaran parte de esa tarea (user count), a el o los usuarios virtuales se les asocia una tarea específica (task), además debe asignarse la estación de trabajo a la cual van a pertenecer (workstation), por su parte, se escoge el explorador donde se están ejecutando las pruebas para la aplicación (browser; en este caso Internet Explorer, pero puede ser cualquier otro), y por ultimo asignar un delay en milisegundos para la conexión dada del HTTP.

- **Stations**

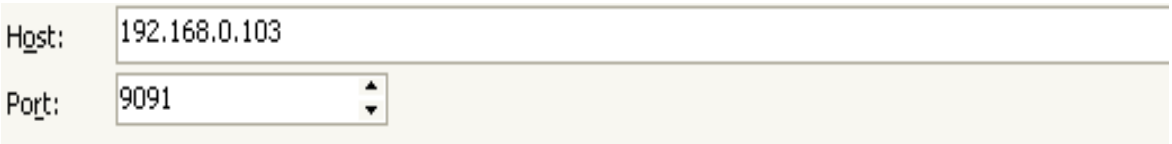


FIGURA 14. DESCRIPCIÓN DE LAS ESTACIONES DE TRABAJO.

En el caso de las estaciones de trabajo, se debe de llenar dos campos: uno identifica el host (dirección IP del sitio de trabajo). El otro es el puerto que especifica el número de acceso para comunicar las aplicaciones de TestComplete con un determinado sitio de trabajo (TestComplete por defecto asigna el puerto 9091). *Ver figura 14.*

- **Task**

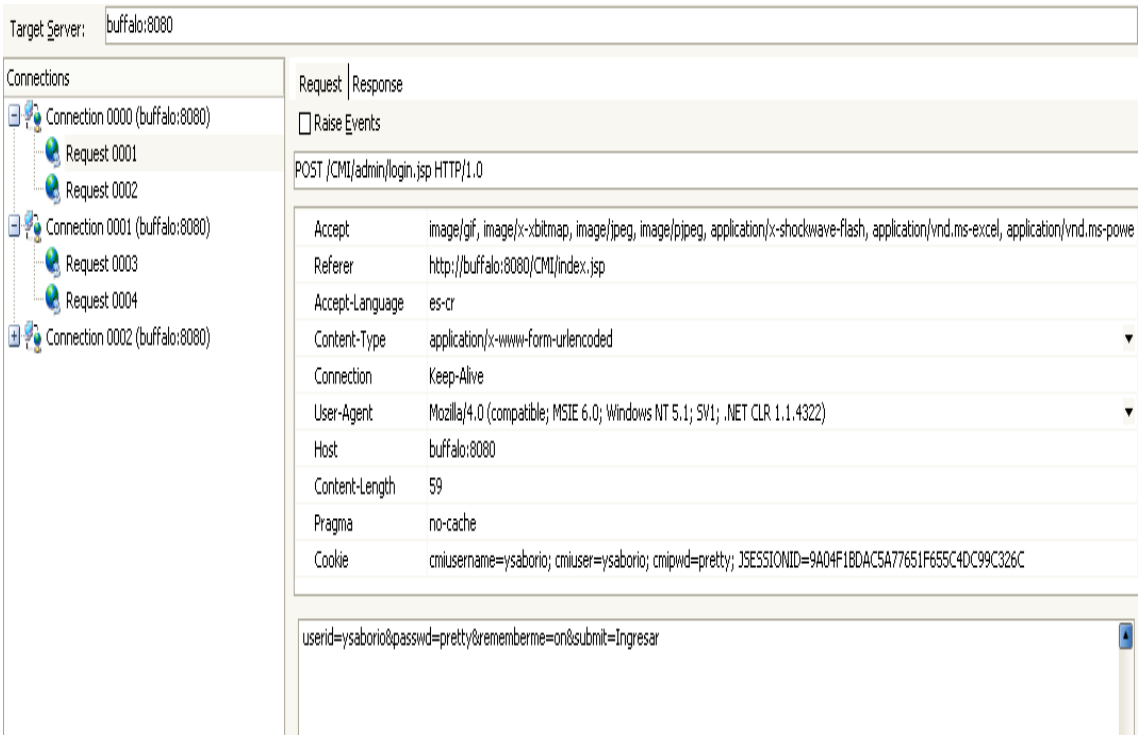


FIGURA 15. TRÁFICO DE LAS TAREAS CREADAS.

La *figura 15,* denota el seguimiento que se le puede dar al tráfico del HTTP registrado por TestComplete, el cual lo organiza en tareas; además que TestComplete especifica el host del cual proviene el tráfico que él mismo registra (target Server). En dichas tareas se puede modificar las características de las peticiones (request) y las respuestas (response) que fueron registradas.

Análisis de Resultados

Análisis de los resultados de las pruebas http load testing log, en donde se incluyen los usuarios, las conexiones, las peticiones y los funcionamientos de las peticiones (headers).

Usuarios

Este panel cuenta con las siguientes columnas que describen los resultados después de generar una prueba de carga, estrés y escalabilidad. Describiendo el comportamiento de los usuarios virtuales dentro de las pruebas.

Columna	Descripción
#	Índice del usuario virtual en la tabla
Virtual User	Nombre del usuario virtual que ejecutó la tarea
Task	Nombre de la tarea
Time (s)	Tiempo de ejecución de la tarea (en segundos)
Connection Count	Número de las conexiones al servidor que fueron creadas al ejecutar la tarea
Workstation	Estación donde la tarea fue ejecutada. Nombre de la estación o la dirección IP.
State	Indica el estado de las conexiones elegidas de la tarea ejecutada: Ok, warning, error.

Conexiones

Esta sección lleva a cabo la información sobre las conexiones que fueron creadas durante el funcionamiento de la tarea seleccionada en el panel de los usuarios. A continuación la descripción de las columnas del comportamiento de las conexiones.

Columna	Descripción
#	Índice de la conexión
Host	Nombre del servidor con el cual la conexión fue establecida
Port	Puerto usado para la conexión
Initialization Time (s)	Duración de la inicialización en segundos (cuanto mayor es el valor de la inicialización, más ocupada está la red)
Time (s)	Duración de la conexión en segundos

Total Bytes Sent (Kb)	Número total de kilobytes enviados al servidor por todas las peticiones hechas vía la conexión
Total Bytes Received (Kb)	Número total de kilobytes recibidos del servidor por todas las peticiones hechas vía la conexión
Start Delay (s)	Cantidad de segundos que pasaron desde el comienzo de la prueba de carga hasta el momento en que la conexión fue establecida
Request Count	Número de peticiones hechas vía la conexión
Error Description	Informe de si todas las peticiones de la conexión fueron simuladas con éxito o no
Performance (Kb/s)	Los valores en esta columna son calculados por la siguiente fórmula: Performance = (Total Bytes Sent + Total Bytes Received) / Time Estos valores indican el funcionamiento del servidor cuando trabaja con la conexión

Peticiones

El panel de las peticiones muestra los resultados para cada petición que fue hecha vía la conexión seleccionada en el panel de las conexiones. A continuación la descripción de las columnas del comportamiento de las peticiones.

Columna	Descripción
#	Índice de la petición en la tabla
Request Time (s)	Cantidad de segundos que la petición de usuario tomó para conseguir al servidor
Bytes Sent (Kb)	Cantidad de datos en kilobytes de la petición enviada al servidor
Bytes Received (Kb)	Cantidad de datos en kilobytes que el servidor envía detrás en respuesta
Response Time (s)	Número de segundos tomados por el servidor para procesar la petición y para enviar los datos al usuario, es decir, este valor es una suma del tiempo del proceso de la petición y del tiempo de la transferencia de los datos
Performance (Kb/s)	Los valores en esta columna son calculados por la siguiente fórmula:

	Performance = (Bytes Sent + Bytes Received) / (Request Time + Response Time) Estos valores indican el funcionamiento total del servidor cuando era de recepción y de proceso de la petición
Request Performance (Kb/s)	Los valores en esta columna son calculados por la siguiente fórmula: Request Performance = Bytes Sent / Request Time Estos valores indican el funcionamiento del servidor cuando recibe la petición
Response Performance (Kb/s)	Los valores en esta columna son calculados por la siguiente fórmula: Response Performance = Bytes Received / Response Time Estos valores indican el funcionamiento del servidor cuando procesa la información

Funcionamientos de las peticiones (headers)

Este panel representa las peticiones y las respuestas que proporciona el servidor a estas peticiones.

Graphs & Charts

Los datos que se tabulan en los usuarios, conexiones o peticiones, pueden ser representados de forma gráfica, con el fin de conseguir una mayor idea de lo que esta sucediendo con las pruebas de carga que se realizan. Los graphs & charts son las opciones que permiten visualizar los datos tabulados en forma gráfica.

En ambos elementos, la información que se desea mostrar gráficamente, puede ser manipulada por el usuario, es decir, éste puede decidir las acciones que quiere ver reflejadas gráficamente, realizando todas las combinaciones necesarias para obtener el resultado de la prueba que el usuario desee.

Por ejemplo en la pestaña Charts, se puede comparar el tiempo que se tomó para simular diversos usuarios virtuales. En cuanto a la pestaña Graphs, se puede visualizar gráficos que demuestran la relación o la dependencia entre dos o más valores.

Análisis del resumen de las pruebas de carga [Load Testing Summary]

Este es un resumen que TestComplete va realizando. Lo que hace es ir registrando todos los resultados de varios de los funcionamientos de la prueba de carga, lo va incluyendo todo en un panel, con sus respectivos gráficos, con el fin de analizar los resultados, y dar al usuario una explicación de lo que esta sucediendo con dichas pruebas de carga.

Este panel contiene información tabuladas en filas y columnas. En donde cada fila en la tabla corresponde a un solo funcionamiento de prueba; por su lado, en cuanto a las columnas se describe a continuación la especificación de cada una de ellas.

Columna	Descripción
N	Índice de la prueba en la tabla
Avg Bytes Received, Kb	Cantidad media de información (en kilobytes) recibida del servidor por la petición
Avg Bytes Sent, Kb	Cantidad media de información (en kilobytes) enviada al servidor por cada petición
Avg Process Time, ms	Número medio de milisegundos que pasa desde el comienzo de la petición (el tiempo en que la petición fue enviada al servidor) hasta el extremo de la petición (el tiempo en que la respuesta del servidor fue recibida)
Avg Request Size, Kb	Tamaño medio de las peticiones (en kilobytes) enviadas al servidor durante el funcionamiento de la prueba
Performance (Kb/s)	El tiempo de interacción de la computadora del cliente con el servidor se puede dividir en 3 fases: - Enviar la petición al servidor - Proceso de la petición en el servidor - Enviar respuesta al cliente

	La columna del funcionamiento demuestra el funcionamiento del sistema durante el funcionamiento de la prueba (en Kb/s). Es un valor sumario de las 3 fases, se calcula de la siguiente manera: (Total Bytes Sent + Total Bytes Received) / Total Process Time
Request Count	Número de peticiones que fueron enviadas al servidor durante el funcionamiento de la prueba
Status	Indica si algunas advertencias o errores fueron encontrados durante el funcionamiento de la prueba; o si el funcionamiento de la prueba fue un éxito (Ok, Warning, Error)
Task Count	Número de tareas que se realizaron durante el funcionamiento de una prueba
Test Date/Time	Fecha y hora del funcionamiento de la prueba
Total Bytes Received, Kb	Cantidad total de kilobytes recibidos del servidor
Total Bytes Sent, Kb	Cantidad total de kilobytes enviados por todas las peticiones al servidor
User Count	Número de los usuarios virtuales que trabajaron con el servidor simultáneamente durante el funcionamiento de una prueba

Extras

TestExecute

Permite ejecutar los scripts y los resultados de las pruebas realizados en TestComplete, en las máquinas que no tienen instalado TestComplete. Permite que los departamentos de QA tengan la capacidad de probar usos en ambientes de usuarios verdaderos, por ejemplo, en los sitios del cliente sin la necesidad de instalar TestComplete.

TestExecute se construye sobre la tecnología de TestComplete y apoya todos los objetos de TestComplete (registros, sistema, opciones, etc.). Es por ello, que no hay problema de cargar los scripts y resultados de las pruebas realizados en TestComplete, y poderlos simular en máquinas en donde no se encuentre instalado TestComplete.

TestRecorder

Es un sistema de biblioteca runtime que se distribuirá con sus usos. Incorporado una vez en sus app, detalla completamente acciones del usuario final en la forma de archivo binario que se puede entonces convertir con TestComplete a un código fácilmente legible por medio de un script (VBScript, JScript, DelphiScript, C++Script o C#Script). Estos scripts le dicen exactamente lo que hacía un usuario durante la ejecución de sus usos, permitiendo que se puedan repetir exactamente la secuencia de las acciones del usuario por medio de TestComplete o TestExecute.

Conclusiones

- Se realizaron las grabaciones de toda la aplicación de Modularización de CMI, tomando en consideración todos los módulos, para las posibles generaciones de aplicaciones. Se realizo la grabación de los siguientes módulos: CMI, evaluaciones, datos generales, incentivos, acciones, riesgos, encuestas, organización, administración, tablero de control, empresas y alertas.

- Es indispensable al momento de realizar las grabaciones, de mantener una secuencia lógica de las mismas, ya que algunos módulos dependen de acciones de módulos anteriores y éstos de módulos posteriores. Es por ello, que la aplicación fue grabada con una secuencia lógica que permite correr los módulos en su orden y obtener resultados favorables.

- Las grabaciones de una aplicación son recomendables realizarlas con una base de datos en blanco, en donde, con forme se avanza en la grabación de la funcionalidad, se van insertando los datos; esto con el fin, de que cada vez que se requiera correr las grabaciones se creen los datos en la base de datos correctamente.

- Un punto considerable de la funcionalidad de ciertos paneles, es donde existen calendarios con el fin de registrar un periodo, de escoger un día del año, etc., ya que en estas situaciones es recomendable cuando escoge una fecha para grabar, que sean fechas futuras con el fin de evitar que la posición grabada no afecte la cantidad y orden de los días en un determinado mes.

- Se debe documentar internamente, cada grabación sobre la aplicación, de una forma, clara y ordenada; ya que a revisiones futuras si la aplicación presenta errores, que el encargado del aseguramiento de la calidad, tenga facilidad de conocer qué pasos se deben realizar para una corrida normal de la aplicación.

- En el momento de cargar las grabaciones en TestComplete es necesario conocer la forma correcta del orden de cada uno de los módulos, para poder mantener la secuencia lógica que se explico anteriormente. Además, que la base de datos podrá ser llenada correctamente si se lleva un orden a la hora de la asignación de los módulos para su ejecución.

Aspectos del Departamento

- El departamento de Aseguramiento de la Calidad de Northek Software, pretende seguir un conjunto de acciones planificadas y sistemáticas, implantadas dentro del Sistema de Calidad de la empresa. Estas acciones deben ser demostrables para proporcionar confianza adecuada (tanto a la empresa como tal, como ha sus clientes) de que se cumplan los requisitos del Sistema de la Calidad.

- El departamento deben considerar siempre en sus planes que son los responsables por que el producto se certifique de la mejor manera hacia el mercado. Es por ello que los encargados de dicho departamento cumplen funciones básicas dentro de la empresa: se denotan los encargados como asesores con responsabilidad indirecta, porque proyecta, implanta, desarrolla y coordina el Sistema de la Calidad de todos los departamentos, siendo estos últimos responsables del cumplimiento de normas, especificaciones y procedimientos establecidos. Otro papel que juegan los encargados es de Ejecutivo, con responsabilidad directa, porque controla la calidad de las actividades de la empresa, por medio de los resultados del autocontrol, las inspecciones y auditorias de la calidad, exigiendo el cumplimiento de las normas, especificaciones y procedimientos establecidos.

- Northek Software cuenta con herramientas de alta proyección en el Área de Aseguramiento de la Calidad, apoyadas éstas, con personal altamente calificado para el buen desempeño de las tareas en esta área de la empresa.

Experiencia Laboral

Concluyo la Práctica de Especialidad con satisfacción ya que considero de que fui participe del progreso de una empresa, y más que eso de una organización de la Zona Norte.

Por su parte, el ámbito laboral es sinónimo de responsabilidad en la vida. La experiencia adquirida en este proceso proporciona un crecimiento personal tanto intelectualmente como ético y moral.

En cuanto al proyecto como tal, considero que es un área de mucha importancia en una empresa, al punto de que es indispensable que exista, ya que el aseguramiento de la calidad hace de una empresa ser reconocida y respetada porque proporciona productos altamente certificados al cliente.

Por tanto, me satisface el haber conocido sobre esta área ya que en el grado de Bachillerato de Ingeniería en Sistemas, no existe un enfoque sobre esta área, y he podido conocer sobre lo que hay detrás del desarrollo de un producto cuando éste ya esta certificado en su destino.

Se puede considerar que desempeñarse en cualquier área de una empresa, sea la que sea, uno es importante y mentalizarse de dar lo mejor, y que el trabajo que se realiza es de vital ayuda para la empresa.

En fin, de la experiencia laboral rescato la humildad, el respeto, responsabilidad, comunicación, compañerismo, sacrificio, ética, moral, empeño y el trabajo como tal; aspectos que forman parte de cada quien y que permiten crecer a una empresa, pero más importante a uno como persona.