Universidad de las Ciencias Informáticas
“Facultad de Entornos Virtuales"

Título: “Estrategia de pruebas de software: SCADA Nacional”

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor(es): Marelis Virgen Pérez García.

Junior Muñoz Treto.

Tutor(es): Ing. Jandrich Domínguez Fortún.

Ing. Gerandys Hernández Casanova.

Ciudad de La Habana, junio del 2008

Año 50 de la Revolución.

...”No hay secretos para el éxito. Este se alcanza preparándose, trabajando arduamente y aprendiendo del fracaso…”

“Colin Pawell”

DECLARACION DE AUTORIA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

"[Insertar nombre(s) de autor(es)]"

_____

"[Insertar nombre(s) de tutor(es)]"

_____

"[Insertar nombre(s) de autor(es)]"

_____

"[Insertar nombre(s) de tutor(es)]"

_____

Autor(es):

Nombre: Marelis Virgen Pérez García.

Correo Electrónico: mvperez@estudiantes.uci.cu

Nombre: Junior Muñoz Treto.

Correo Electrónico: jmunoz@estudiantes.uci.cu

Tutor(es):

Nombre: Jandrich Domínguez Fortún.

Correo Electrónico: jandrich@uci.cu

Nombre: Gerandys Hernández Casanova.

Correo Electrónico: ghernandez@uci.cu

AGRADECIMIENTOS

Muchas gracias a todos los que de una forma u otra confiaron y me apoyaron siempre, familiares, vecinos, amigos.

Una vez alguien me dio un manual, “Manual para subir montañas” se llamaba, nunca supe bien lo que decía hasta hoy que entendí que la montaña de la que se hablaba era el obstáculo que tenía enfrente y debía vencer. Mil gracias a Dios, sin él, creo que nunca hubiera salido de todo esto. A mi madre, mami un día te darás cuenta que todo lo que he hecho en mi vida es por y para ti. Siempre serás mi fuente de inspiración, sin tu apoyo y tu seguridad creo que no lo hubiera logrado. Tu esfuerzo no será en vano.

A mi padre agradecerle por su preocupación y apoyo. A toda mi familia muchas gracias por su preocupación y en especial a mi tía China.

Pepe muchas gracias a ti también por tu ayuda, que aunque no eras mi tutor siempre estuviste ahí para nosotros. A Jandrich Domínguez y Gerandys Hernández, mis tutores, gracias por su apoyo y horas de dedicación.

Marelis

Son tantas personas a las cuales debo parte de este triunfo.

Primero y antes que nada, dar gracias a **Dios**, por estar presente en cada paso que doy, por fortalecer mi corazón y mi fe, por iluminar mi mente. Por haber puesto en mi camino a aquellas personas que han sido mi soporte y compañía durante toda mi vida de estudiante.

Agradecer hoy y siempre a mi madre, la mujer que me apoyó todos estos años de mi vida, por su infinito amor, comprensión y ternura. Madre, serás siempre mi inspiración para alcanzar mis metas, por enseñarme que todo se aprende y que todo esfuerzo es al final recompensa. Tu esfuerzo, se convirtió en tu triunfo y el mío, TE AMO.

Gracias doy a mi padre por darme siempre el ánimo y apoyo que necesito, me da la fortaleza necesaria para seguir adelante.

A mi hermano al cual le debo mi alegría cotidiana y mis deseos de triunfar en la vida.

A mi novia Geidys quien es mi estabilidad emocional y sentimental. Por acompañarme en los malos y buenos momentos. Por ayudarme a que este momento llegara. ¡GRACIAS MI NIÑA!

Un agradecimiento especial a mis tutores Jandrich Domínguez y Gerandys Hernández por la colaboración, paciencia y sobre todo por esa gran amistad que supieron cultivar.

Y en especial agradecer a aquella persona que una vez dijo: “No hay nada que el hombre no pueda hacer si lo quiere con suficiente determinación”.

Junior

Este trabajo va dedicado a mi mamá que siempre me ha guiado por el mejor de los caminos y ha sabido ser paciente y optimista. A mi abuelita Hortensia y a Gloria, donde quiera que estén se que se sienten orgullosas de verme.

A los que pusieron su granito de arena para que esto que un día fue un sueño hoy se hiciera realidad. A Junior, mi compañero de tesis, por ser tan paciente y confiar en que saldríamos adelante.

Marelis

Dedico este trabajo de diploma en primer lugar a mi madre, quien a pesar de mis defectos me ha hecho la persona quien soy y me ha guiado paso a paso por la vida. A mi querido padre quien me ha extendido su mano en cada momento. A mi hermano que es mi razón de vivir y motivación. A mi abuelo Julio que donde quiera que esté ya puede sentirse orgulloso por tener su primer nieto en la meta soñada por él. A mi novia Geidys quien ha sido paciente a mi lado y me ha brindado seguridad y confianza.

A todos aquellos que han hecho posible esta victoria lograda, en especial a mis tutores Jandrich y Gerandys por soportarnos en cada instante que acudimos a ellos. A mi gran amigo Pepito que siempre atendió a mis lamentos de estudiante. A mi compañera de tesis Marelis que sin ella no habría logrado este triunfo ¡¡¡ya logramos nuestro primer sueño!!!

Junior

En la actualidad, el campo de la informática se ha impuesto en el desarrollo de la sociedad, buscando cada día la mejora y calidad con que se produce el software. A esta tarea se ha sumado Cuba, firmando un contrato con Venezuela para desarrollar un software para el Control, Supervisión y Adquisición de Datos (SCADA), específicamente implementado en la Universidad de Ciencias Informáticas (UCI).

Se hacía presente la necesidad de controlar la calidad con que se producía dicho sistema, por lo que surge la idea de trazar una estrategia de pruebas desarrollada por el grupo de calidad interno de la UCI, la cual tuviera como objetivos principales la planificación de las pruebas, diseño de casos de prueba y ejecución de las mismas para entregar los anexos pactados que lo requerían con una mayor calidad.

PALABRAS CLAVES: Prueba, Estrategia de Pruebas, SCADA.

TABLA DE CONTENIDOS

INTRODUCCION

El mundo de la informática últimamente ha tenido un gran avance en la producción de software, creciendo su desarrollo en gran medida y con ello la demanda de los clientes. Las empresas dedicadas al desarrollo de software intentan aumentar la calidad de sus productos para evitar los problemas inherentes a sus procesos: plazos y presupuestos incumplidos, insatisfacción del usuario, escasa productividad y la baja calidad en el software producido. Las empresas son conscientes de que el mercado valora cada día más, la calidad. Por lo tanto, las compañías exigen la disminución de errores y penalizan los retrasos en entregas y las cancelaciones de proyectos.

Cuba se ha visto inmersa en esta carrera de producción de software teniendo como uno de sus principales exponentes a la Universidad de Ciencias Informáticas (UCI), la cual surge al calor de la batalla de ideas y tiene como misión fundamental crear software con una mejor calidad en la industria cubana.

Cuba con su política de internacionalismo se ha vinculado estrechamente con el país hermano Venezuela, al cual se le ha brindado apoyo tanto en la rama de la salud como de la informática. La UCI ha dado su aporte en estas tareas, dando su paso al frente con la misión Milagro y en proyectos productivos, uno de estos es el vinculado con Petróleos de Venezuela Sociedad Anónima (PDVSA).

PDVSA cuenta en sus instalaciones con Sistemas de Supervisión y Adquisición de Datos (SCADA) suministrados por compañías extranjeras, que son imprescindibles para la confiabilidad y eficiencia de sus Procesos Tecnológicos. Pero el costo de mantenerlos en funcionamiento es elevado e imponen una total dependencia de los proveedores. (1)

A raíz de PDVSA surge el Proyecto SCADA Nacional (PSN), el cual tiene como objetivo el de “Desarrollar un Sistema que supervise, controle, optimice y gestione los procesos industriales de PDVSA sirviendo de base para la implantación en otras Industrias y Organismos del país.” (2)

La necesidad de la calidad en los productos se ve presente también en el PSN, ya que tanto en la industria como en el comercio y en las organizaciones de servicios y nuevas tecnologías, es hoy en día un claro factor diferencial competitivo. Evidentemente en el software SCADA no es una excepción.

La situación estriba en que a menudo los profesionales no saben cómo aumentar la calidad del software de una forma eficaz.

Una herramienta clave para conseguir una aplicación de alta calidad y libre de errores es contar con un proceso de pruebas efectivo. Lamentablemente y aunque es una de las técnicas fundamentales de aseguramiento de calidad, al proceso de pruebas no se le suele dar la importancia que merece, lo que significa que las empresas no cuentan con una buena metodología y/o con las herramientas adecuadas para diseñar e implementar un proceso de pruebas adecuado a sus necesidades.

Debido a la complejidad de probar una aplicación, la etapa de pruebas puede llegar a ser de las más lentas del proceso de desarrollo de software. Sin embargo, si un proceso de pruebas se desarrolla siguiendo una planificación, una metodología y unas herramientas adecuadas se pueden conseguir importantes disminuciones en los costos de desarrollo y mantenimiento del software, así como una reducción en el número de errores.

Desde la década del 70, el tema de las pruebas de software ha sido motivo de preocupación para especialistas, ingenieros, investigadores y comercializadores de software, los cuales han realizado gran cantidad de investigaciones al respecto. El 50 por ciento del tiempo empleado en el proyecto y más del 50 por ciento del costo total, era utilizado en las pruebas o en desarrollo del sistema. (3)

El desarrollo de la industria de software implica también que los productos realizados deben de ser confiables, precisos, rápidos de utilizar, flexibles y además deben de estar bien documentados. Un producto cumple con estos requisitos cuando se le han aplicado las técnicas de Ingeniería de Software adecuadas, se han utilizado los roles apropiados para el desarrollo de las tareas en la empresa de software y en su etapa de comprobación se le han aplicado las pruebas necesarias para lograr el nivel de calidad requerido**.** (4)

La UCI, constituye en la industria cubana del software uno de los pilares de mayor importancia para la economía del país. Las ideas de convertir nuestros servicios informáticos en un rubro exportable se ven frenadas por diversos problemas que atentan contra la calidad del proceso productivo, y por consiguiente, contra los resultados del producto final. Muchos de estos problemas tienen su raíz en malas prácticas de ingeniería y gestión de software durante el proceso de desarrollo. (5)

Este es un tema nuevo del que no se tienen conocimientos ni precedentes en la universidad. Se habían desarrollado productos pero todos encaminados a software de gestión, educativos y de simulación pero nunca se había incursionado en software críticos como el SCADA. Para desarrollarlo, se dividió el trabajo en varias líneas para llevar a cabo por Cuba y Venezuela, dos de las cuales correspondían a la parte venezolana como es el caso de Calidad y Prueba, pero como el personal que debía trabajar en estas dos líneas aún no estaba conformado, Cuba se vio en la necesidad de crear un equipo de calidad por su parte para desempeñar dicha tarea en ese momento. Por tal motivo es que se realiza esta propuesta de estrategia de prueba para la rápida detección y corrección de errores en la entrega de la primera versión del producto SCADA Nacional en uno de los proyectos productivos en la facultad 5 de la UCI.

Surge entonces el siguiente problema científico, ¿Cómo aplicar el proceso de pruebas de software en el proyecto SCADA Nacional?

Como objeto de estudio se plantea el proceso de producción de software del proyecto SCADA Nacional y el campo de acción, el proceso de pruebas del proyecto SCADA Nacional.

El objetivo general establecido es diseñar y aplicar una estrategia de prueba en el proyecto SCADA Nacional.

Como objetivos específicos de esta investigación se tienen:

1. Elaborar el marco teórico de la investigación.
2. Realizar el plan de prueba de cada uno de los módulos del proyecto SCADA Nacional.
3. Diseñar casos de prueba.
4. Ejecutar las pruebas planificadas.
5. Documentar los resultados alcanzados con las pruebas.

En la hipótesis se plantea que la aplicación de la estrategia de pruebas de software elaborada permitirá que los productos del trabajo entregados a los clientes cumplan con los requisitos establecidos por estos.

CAPITULO 1: FUNDAMENTACION TEORICA.

1.1- Introducción

El proceso de desarrollo de software consta de 4 fases: inicio, elaboración, construcción y transición. En cada una de ellas se van desarrollando y ensamblando partes que conformarán el producto final. Al igual que ocurre en un proceso productivo normal, cada una de estas partes debe ser "probada". El tipo de pruebas a realizar con el software varía a medida que el desarrollo avanza.

Es una realidad que los fallos de software ocasionan graves pérdidas económicas, sin embargo, el nivel de concienciación en cuanto a la necesidad de incorporar mecanismos de calidad es muy pequeño.

En el presente capítulo se lleva a cabo el estudio de las tendencias actuales en cuanto a calidad de software, abordando específicamente en el tema de las pruebas de software en los sistemas SCADA. Además se exponen conceptos relacionados con este proceso, los diferentes tipos de pruebas existentes y cuáles son aplicables para estos sistemas.

1.2- Calidad del software

Desde tiempos remotos el ser humano ha tratado de realizar las tareas que desempeñaba en su vida cotidiana lo mejor que podía, para poder obtener resultados satisfactorios. En la actualidad se sigue este patrón bien de cerca buscando realizar las tareas con una mejor calidad.

El término calidad tiene muchas acepciones, pero la básica es aquella que dice que aquel producto o servicio que se adquiere satisfaga las expectativas sobradamente, es decir, que el producto o servicio funcione tal y como se quiere. La palabra "Calidad" siempre será entendida de manera diferente por cada persona, ya que para unos la calidad residirá en un producto y en otros en su servicio posventa. Lo cierto es que nunca se llegará a definir exactamente lo que representa el término “Calidad” a pesar de que últimamente se haya puesto de moda.

Hoy en día se le han dotado de diferentes definiciones entre las que se pueden mencionar:

-**Calidad:** Conjunto de propiedades y características de un producto o servicio que le confieren su aptitud para satisfacer necesidades explícitas o implícitas. (6)

-Calidad: Grado en el que un conjunto de características inherentes cumple con los requisitos. (7)

Uno de los problemas que se afrontan actualmente en la esfera de la informática es la calidad del software. Desde la década del 70 este tema ha sido motivo de preocupación para ingenieros e investigadores de software, los cuales han realizado gran cantidad de investigaciones respecto a cómo obtener un software con calidad y cómo evaluar la calidad del software.

“La calidad del software es el grado con el que un sistema, componente o proceso cumple con los requerimientos especificados y las necesidades o expectativas del cliente o usuario”. (8)

La calidad del software es el conjunto de cualidades que lo caracterizan y que determinan su utilidad y existencia. El modelo de calidad establecido en la primera parte del estándar ISO 9126-1 (Organización Internacional de Estandarización), clasifica la calidad del software en un conjunto estructurado de características y subcaracterísticas de la siguiente manera: funcionalidad, fiabilidad, usabilidad, eficiencia, mantenimiento y movilidad.

La calidad del software puede medirse después de elaborado el producto, pero esto puede resultar muy costoso si se detectan problemas derivados de imperfecciones en el diseño, por lo que es imprescindible tener en cuenta tanto la obtención de la calidad como su control durante todas las etapas del ciclo de vida del software.

Cuando se hace referencia a la calidad no se habla de una medida en específico, sino que se trata de hacer una comparación para ver cuán bueno o cuán malo es un producto respecto a otro, o simplemente se hace referencia a los atributos que debe tener un servicio o producto para que sea bueno. Además se puede hablar de calidad cuando el cliente para el que estaba destinado el servicio o producto quedó totalmente satisfecho.

Ajustando todo esto a un producto software sería entonces la manera de asegurar todo el ciclo de vida del producto para cuando se ponga en práctica no ocurran fallos que den al traste con las expectativas del usuario.

1.2.1- Cómo obtener un software con calidad

La obtención de un software con calidad implica la utilización de metodologías y estándares para el análisis, diseño, implementación y prueba del software que permitan uniformar la filosofía de trabajo en aras de lograr una mayor confiabilidad, mantenibilidad y facilidad de prueba, a la vez que eleven la productividad, tanto para la labor de desarrollo como para el control de la calidad del software.

Según Pressman los factores que determinan la calidad del software se pueden clasificar en dos grupos:

- Los que pueden ser medidos directamente.
- Los que pueden ser medidos indirectamente. (9)

McCall se refiere a los factores de calidad del software de la siguiente manera:

- **Factor de Calidad:** Corrección, Fiabilidad, Eficiencia, Integridad, Facilidad de uso.
  - Aspecto que trata: Operación del producto.
- **Factor de Calidad:** Facilidad de mantenimiento, Flexibilidad, Facilidad de prueba.
  - Aspecto que trata: Revisión del producto.
- **Factor de Calidad:** Portabilidad, Reusabilidad, Facilidad de interoperación.
  - Aspecto que trata: Transición del producto. (10)

La adopción de buenos factores contribuye en gran medida a lograr la calidad del software pero no la asegura. Para el aseguramiento de la calidad es necesario su control o evaluación.

1.2.2- Cómo controlar la calidad del software

El control de la calidad es la parte de la gestión de la calidad orientada al cumplimiento de los requisitos de la calidad. (7)

Al Control de la calidad del Software lo acompañan las técnicas y actividades de carácter operativo, utilizadas para satisfacer los requisitos relativos a la calidad, centrados en 2 objetivos fundamentales:

- Mantener bajo control un proceso.
- Eliminar las causas de los defectos en las diferentes fases del ciclo de vida. (9)

Para controlar la calidad del software es necesario ante todo, definir los parámetros, indicadores o criterios de medición.

Las cualidades para medir la calidad del software son definidas por innumerables autores, los cuales las denominan y agrupan de formas diferentes. Por ejemplo, John Wiley define métricas de calidad y criterios, donde cada métrica se obtiene a partir de combinaciones de los diferentes criterios. Otros autores identifican la calidad con el nivel de complejidad del software y definen dos categorías de métricas: de complejidad de programa o código y de complejidad de sistema o estructura.

Se puede decir entonces de acuerdo a las numerosas definiciones expuestas por los distintos autores, que la cualidad más completa para medir la calidad del software es la definida por John Wiley ya que el mismo define métricas de calidad y criterios, ayudando esto a lograr un mayor éxito en la producción de software.

1.2.3-Normas y estándares de calidad

La industria de software a diferencia de otras industrias, tiene muy poco tiempo de existir. Lo que ha llamado la atención del mercado hacia ella han sido dos factores esenciales: la velocidad con que ha crecido y su alcance. El software se convirtió con el tiempo en un negocio rentable al haber tanta demanda en ese campo. Muchas personas empezaron a desarrollar software en diferentes países naciendo las primeras grandes empresas de software, trayendo consigo un problema natural.

Había tantos desarrolladores en distintos países y trabajando en distintas aplicaciones que comenzó a haber diversidad de estilos así como la calidad del producto final variaba mucho. Es por ello la necesidad de crear un estándar que permitiera a los consumidores de software decidir si el producto que estaban recibiendo era de calidad y si cumplía ciertos requisitos de funcionalidad.

Según ISO, un estándar es “un conjunto de acuerdos documentados que contienen especificaciones técnicas u otros criterios precisos para ser usados constantemente, como reglas, lineamientos, o definiciones de características. Todo esto con la finalidad de asegurar que los materiales, productos, procesos y servicios son óptimos para su propósito”.

Los estándares de calidad determinan el nivel mínimo y máximo aceptable para un indicador. Si el valor del indicador se encuentra dentro del rango significa que se cumple con el criterio de calidad que se había definido y que las cosas transcurren conforme a lo previsto.

1.2.3.1-Norma ISO 9001:2000

Uno de los propósitos principales de un estándar es promover un intercambio de productos en base a ciertos lineamientos comunes.

La '**Norma ISO 9001'** ha sido elaborada por el Comité Técnico ISO/TC176 de ISO y especifica los requisitos para un sistema de gestión de la calidad que pueden utilizarse para su aplicación interna por las organizaciones, para certificación o con fines contractuales.

La norma ISO 9001:2000 contiene la especificación del modelo de gestión y los requisitos del Modelo, los requisitos que han de cumplir los sistemas de la calidad a efectos de confianza interna, contractuales o de certificación.

Dicha norma está estructurada en ocho apartados, refiriéndose los cuatro primeros a declaraciones de principios, estructura y descripción de la empresa, requisitos generales y requisitos de documentación, es decir, son de carácter introductorio. Los apartados cinco a ocho están orientados a procesos y en ellos se agrupan los requisitos para la implantación del sistema de calidad. Los mismos son:

- **Apartado del 1 al 3**: Guías y descripciones generales, no se enuncia ningún requisito.

  1.1 Generalidades.

  1.2 Reducción en el alcance.

  2 Normativas de referencia.

  3 Términos y definiciones.

- **Apartado 4. Sistema de gestión**: contiene los requisitos generales y los requisitos para gestionar la documentación.

4.1 Requisitos generales.

4.2 Requisitos de documentación.

- **Apartado 5. Responsabilidades de la dirección**: contiene los requisitos que debe cumplir la dirección de la organización, tales como definir la política, asegurar que las responsabilidades y autoridades están definidas, aprobar objetivos, el compromiso de la dirección con la calidad, etc.

  5.1 Requisitos generales.

  5.2 Requisitos del cliente.

  5.3 Política de calidad.

  5.4 Planeación.

  5.5 Responsabilidad, autoridad y comunicación.

  5.6 Revisión gerencial.

- **Apartado 6. Gestión de los recursos**: la norma distingue 3 tipos de recursos sobre los cuales se debe actuar: RRHH, infraestructura, y ambiente de trabajo. Aquí se contienen los requisitos exigidos en su gestión.

  6.1 Requisitos generales.

  6.2 Recursos humanos.

  6.3 Infraestructura.

  6.4 Ambiente de trabajo.

- **Apartado 7. Realización del producto**: aquí están contenidos los requisitos puramente productivos, desde la atención al cliente, hasta la entrega del producto o el servicio.

7.1 Planeación de la realización del producto y/o servicio.

7.2 Procesos relacionados con el cliente.

7.3 Diseño y desarrollo.

7.4 Compras.

7.5 Operaciones de producción y servicio

7.6 Control de dispositivos de medición, inspección y monitoreo

- **Apartado 8. Medición, análisis y mejora**: aquí se sitúan los requisitos para los procesos que recopilan información, la analizan, y que actúan en consecuencia. El objetivo es mejorar continuamente la capacidad de la organización para suministrar productos que cumplan los requisitos. El objetivo declarado en la norma, es que la organización busque sin descanso la satisfacción del cliente a través del cumplimiento de los requisitos.

  8.1 Requisitos generales.

  8.2 Seguimiento y medición.

  8.3 Control de producto no conforme.

  8.4 Análisis de los datos para mejorar el desempeño.

  8.5 Mejora.

El propósito de ISO 9001 es asegurar a los clientes que los proveedores pueden brindar productos y servicios de calidad. Está pensado para llenar las necesidades del cliente, las del proveedor son secundarias. Una organización debe de alcanzar y sostener la calidad de un producto o servicio producido para seguir en la búsqueda continua de las necesidades explícitas o implícitas del cliente.

1.2.3.2- Modelo de Capacidad y Madurez (CMMI)

CMMI es un modelo para la mejora o evaluación de los procesos de desarrollo y mantenimiento de sistemas y productos de software. Fue desarrollado por el Instituto de Ingeniería del Software de la Universidad Carnegie Mellon (SEI) y publicado en su primera versión en enero del 2002.

Un nivel de madurez es un sistema evolutivo y bien definido para alcanzar el proceso de madurez del software. Cada nivel de madurez tiene dentro de si mismo parámetros que permitan la mejora continua. Alcanzar un nivel dentro de la escala de CMMI significa seguir en busca de mejores prácticas y a la vez mantener los logros alcanzados.

Los niveles de madurez tienen varios objetivos:

- Definir un orden para medir la calidad de software gracias a la madurez de la compañía.
- Ayudar a la organización a ver que procesos debe de mejorar en forma gradual para alcanzar el nivel óptimo.
- Mantener un proceso bien documentado.
- Lograr un producto controlado, verificable, validado y medido.

Los niveles son: inicial, repetible, definido, administrado y óptimo. Estos se describirán a continuación.

Nivel 1: Inicial

Este nivel es el primer estado en la evolución de las organizaciones que desarrollan software. En éste nivel se encuentran todas las empresas que no han logrado implementar las prácticas básicas de administración de proyectos e ingeniería de software definidas a partir del nivel 2 o superiores.

Nivel 2: Repetible

En el nivel repetible se establecen prácticas básicas de administración de proyecto que permiten establecer control de requerimientos, calendarizaciones y costos. El equipo que desarrolló el proyecto puede aprovechar su experiencia e inversión en procesos para aplicarla en un nuevo proyecto.

Nivel 3: Definido

En este nivel la empresa ha definido un conjunto de procesos, metodologías y herramientas usados por todos los niveles involucrados en el proceso, tanto administrativos como desarrolladores. Existen pautas y criterios definidos para adaptar un proceso estándar a las necesidades y características propias de cada proyecto. El nivel de definición es detallado y completo. En el proceso ya no existe dependencia de esfuerzos individuales, pues todos conocen el proceso.

Dentro de este nivel las áreas de procesos que más aportan al proceso de pruebas son las de Validación y Verificación.

Área de Proceso Validación

El objetivo que se persigue con esta área de proceso es demostrar que todo producto puede satisfacer las necesidades por las que fue creado. Como metas a implementar y prácticas a desarrollar se encuentran las siguientes:

- **Objetivo Específico 1:** Preparar la validación**.**
  - Práctica Específica 1.1: Seleccionar los productos a validar.
  - Práctica Específica 1.2: Establecer el entorno de validación.
  - Práctica Específica 1.3: Establecer los procedimientos y criterios de validación.
- **Objetivo Específico 2:** Validar los productos o componentes de los productos**.**
  - Práctica Específica 2.1. Realizar la validación.
  - Práctica Específica 2.2. Analizar los resultados de la validación.

Área de Proceso Verificación

Como objetivo de esta área se encuentra el asegurar que los productos de trabajo responden a los requerimientos planteados. Como metas a implementar y prácticas a desarrollar se encuentran las siguientes:

- **Objetivo Específico 1: Preparar la verificación**
  - Práctica Específica 1.1: Seleccionar los productos de trabajo para la verificación
  - Práctica Específica 1.2: Establecer el entorno de verificación.
  - Práctica Específica 1.3: Establecer los procedimientos y criterios de verificación.

- **Objetivo Específico 2: Realizar revisiones por terceros**
  - Práctica Específica 2.1: Preparar revisiones por terceros.
  - Práctica Específica 2.2: Realizar revisiones por terceros.
  - Práctica Específica 2.3: Analizar resultados de revisiones por terceros.
- **Objetivo Específico 3: Verificar los productos de trabajo seleccionados**
  - Práctica Específica 3.1: Realizar la verificación.
  - Práctica Específica 3.2: Analizar los resultados de la verificación.

Nivel 4: Administrado

En este nivel la organización mide la calidad del producto y del proceso de software. Ambas partes son seguidas en forma cuantitativa y se controlan mediante métricas detalladas. La capacidad de rendimiento del proceso es previsible.

Nivel 5: Óptimo

La característica principal aquí es que la organización entera lleva a cabo mejoras continuas en el proceso, en base a la retroalimentación cuantitativa y al ensayo de ideas y tecnologías innovadoras. Todos los cambios realizados en cualquier parte del proceso son vigilados y controlados con el sistema de métricas. Durante todo el desarrollo no suelen suceder errores.

1.2.4- Metodologías de Desarrollo de Software

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos software.

En estas se van indicando paso a paso todas las actividades a realizar para lograr el producto informático deseado, indicando además qué personas deben participar en el desarrollo de las

actividades y qué papel deben de tener. Además detallan la información que se debe producir como resultado de una actividad y la información necesaria para comenzarla.

Actualmente es imprescindible considerar los riesgos, aunque habitualmente las empresas no han sido concienciadas de los riesgos inherentes al procesamiento de la información mediante ordenadores, a lo que han contribuido, a veces, los propios responsables de informática, que no han sabido explicar con la suficiente claridad las consecuencias de una política de seguridad insuficiente o incluso inexistente. Por otro lado, debido a una cierta deformación profesional en la aplicación de los criterios de coste/beneficio, el directivo desconocedor de la informática no acostumbra a autorizar inversiones que no lleven implícito un beneficio demostrable, tangible y mensurable.

Las técnicas indican cómo debe ser realizada una actividad técnica determinada, identificada en la metodología, combinando el empleo de unos modelos o representaciones gráficas junto con el empleo de unos procedimientos detallados. Se debe tener en consideración que una técnica determinada puede ser utilizada en una o más actividades de la metodología de desarrollo de software, teniendo en cuenta cuando cambiar una técnica por otra.

La evolución de la disciplina de ingeniería de software ha traído consigo propuestas diferentes para mejorar los resultados del proceso de construcción. Las metodologías tradicionales haciendo énfasis en la planeación y las metodologías ágiles haciendo énfasis en la adaptabilidad del proceso, delinean las principales propuestas presentes en la literatura. De manera paralela, el tema de modelos para el mejoramiento de los procesos de desarrollo ocupa un lugar importante en la búsqueda de la metodología adecuada para producir software de calidad en cualquier contexto de desarrollo.

1.2.4.1-Metodologías tradicionales en el desarrollo

Las metodologías tradicionales se caracterizan por exponer procesos basados en planeación exhaustiva. Esta planeación se realiza esperando que el resultado de cada proceso sea determinante y predecible. La experiencia ha mostrado que como consecuencia de las características del software, los resultados de los procesos no son siempre predecibles y sobre todo, es difícil predecir desde el comienzo del proyecto cada resultado. Sin embargo, es posible por medio de la recolección y estudio de métricas de desarrollo lograr realizar estimaciones acertadas en contextos de desarrollo repetibles.

Remontándose a la historia, el modelo de cascada fue uno de los primeros modelos de ciclo de vida (MCV) que formalizó un conjunto de procesos de desarrollo de software. Este MCV describe un orden secuencial en la ejecución de los procesos asociados. El modelo espiral se postuló como una alternativa al modelo de cascada. La ventaja de este modelo radica en el perfeccionamiento de las soluciones encontradas con cada ciclo de desarrollo, en términos de dar respuesta a los requerimientos inicialmente analizados. El modelo de cascada y el modelo espiral suponen, de manera general, que los requerimientos del cliente no cambian radicalmente en el transcurso del desarrollo del sistema.

Por otro lado, la realización de prototipos es una herramienta en la que se apoyan diferentes MCV. Un prototipo debe tener el objetivo de mostrar al cliente o a la gerencia del proyecto el resultado que se obtendrá de la implementación de cada uno de los requerimientos del cliente una vez terminado el desarrollo. Con los prototipos se tiene la posibilidad de obtener retroalimentación de manera temprana.

La solución a algunos de los problemas presentados por las metodologías tradicionales se logra con una gran evolución del modelo espiral. El proceso unificado propone la elaboración de varios ciclos de desarrollo, donde cada uno finaliza con la entrega al cliente de un producto terminado. Este se enmarca entre los conocidos modelos iterativo-incremental, el cual es nombrado como Proceso Unificado de Desarrollo de Software (RUP).

1.2.4.2-Metodologías ágiles

Algunos grupos de desarrollo han experimentado soluciones que basan su fundamento en la adaptabilidad de los procesos de desarrollo, en lugar de seguir esperando lograr resultados predecibles de un proceso que no evoluciona. Esta comunidad de desarrolladores e investigadores han nombrado su trabajo bajo lo que conocemos como metodologías ágiles. Las metodologías ágiles no están en contra de administrar procesos de desarrollo, por el contrario, promueven la formalización de procesos adaptables.

La compilación de los principios y valores que resaltan las metodologías ágiles fue formalizada en el manifiesto para el desarrollo de software ágil. Este documento desarrollado por los representantes de cada una de las metodologías que en el momento se presentaban como ágiles, logra resumir en un conjunto de ideas las prácticas que una metodología de este estilo debe llevar a cabo. Como

característica fundamental, la habilidad de responder al cambio es la principal característica de las metodologías ágiles.

Programación Extrema (XP), una de las más difundidas, es una metodología de desarrollo de software ágil que define pocas reglas y pocas prácticas. XP promueve la adaptabilidad de los procesos de desarrollo basándose en los principios y prácticas que presenta. Quienes trabajan usando XP deben seguir procesos disciplinados, pero más que eso, deben combinar la disciplina con la adaptabilidad necesaria del proceso.

Las metodologías de Cristal se basan en el principio de que tipos diferentes de proyectos requieren tipos diferentes de metodologías. La metodología escogida debe depender de dos factores: el número de personas en el proyecto y las consecuencias de los errores. Conforme al principio de las metodologías ágiles, Scrum recalca la imposibilidad de encontrar procesos definidos y repetibles cuando no existen problemas, personas ni ambientes definidos y repetibles.

Gracias a lo planteado anteriormente, se llega a la conclusión de que entre los tipos de metodologías existentes, las más completas son las metodologías tradicionales ya que a través de las mismas es posible lograr realizar estimaciones acertadas en contextos de desarrollo repetibles por medio de la recolección y estudio de métricas de desarrollo.

Dentro de este tipo de metodología resalta RUP, la cual tiene gran importancia por su forma disciplinada de asignar tareas y responsabilidades, llevar a cabo un desarrollo iterativo, administrar los requisitos y en especial verificar la calidad de software.

1.2.4.3 – Metodología RUP

Los autores de RUP destacan que el proceso de software propuesto por RUP tiene tres características esenciales: está dirigido por Casos de Uso, está centrado en la arquitectura y es iterativo e incremental.

Los Casos de Uso son una técnica de captura de requisitos que fuerza a pensar en términos de importancia para el usuario y no sólo en términos de funciones que sería bueno contemplar. Se define

un Caso de Uso como un fragmento de funcionalidad del sistema que proporciona al usuario un valor añadido. Los Casos de Uso representan los requisitos funcionales del sistema. (11)

En RUP los Casos de Uso no son sólo una herramienta para especificar los requisitos del sistema. También guían su diseño, implementación y prueba. Los Casos de Uso constituyen un elemento integrador y una guía del trabajo como se muestra en el Anexo2.

En el caso de RUP además de utilizar los Casos de Uso para guiar el proceso, presta especial atención al establecimiento temprano de una buena arquitectura que no se vea fuertemente impactada ante cambios posteriores durante la construcción y el mantenimiento.

La arquitectura de un sistema es la organización o estructura de sus partes más relevantes, lo que permite tener una visión común entre todos los involucrados (desarrolladores y usuarios) y una perspectiva clara del sistema completo, necesaria para controlar el desarrollo. (11)

Cada producto tiene tanto una función como una forma. La función corresponde a la funcionalidad reflejada en los Casos de Uso y la forma la proporciona la arquitectura. Existe una interacción entre los Casos de Uso y la arquitectura. Los Casos de Uso deben encajar en la arquitectura cuando se llevan a cabo y la arquitectura debe permitir el desarrollo de todos los Casos de Uso requeridos actualmente y en el futuro. Esto provoca que tanto arquitectura como Casos de Uso deban evolucionar en paralelo durante todo el proceso de desarrollo de software.

Es conveniente ver el sistema desde diferentes perspectivas para comprender mejor el diseño por lo que la arquitectura se representa mediante varias vistas que se centran en aspectos concretos del sistema, abstrayéndose de los demás. Estas vistas son: la vista lógica, de implementación, de proceso y de despliegue, más la de Casos de Uso que es la que da cohesión a todas.

La estrategia que se propone en RUP para el equilibrio correcto entre los Casos de Uso y la arquitectura es tener un proceso iterativo e incremental en donde el trabajo se divide en partes más pequeñas o mini proyectos permitiendo que el equilibrio entre Casos de Uso y arquitectura se vaya logrando durante cada mini proyecto y así durante todo el proceso de desarrollo. Cada mini proyecto se puede ver como una iteración (un recorrido más o menos completo a lo largo de todos los flujos de

trabajo fundamentales) del cual se obtiene un incremento que produce un crecimiento en el producto. (12)

Una iteración puede realizarse por medio de una cascada como se muestra en el Anexo3. Se pasa por los flujos fundamentales (Requisitos, Análisis, Diseño, Implementación y Pruebas). También existe una planificación de la iteración, un análisis de la iteración y algunas actividades específicas de la iteración. Al finalizar se realiza una integración de los resultados con lo obtenido de las iteraciones anteriores.

El proceso iterativo e incremental consta de una secuencia de iteraciones. Cada una aborda una parte de la funcionalidad total pasando por todos los flujos de trabajo relevantes y refinando la arquitectura. Cada iteración se analiza cuando termina. Se puede determinar si han aparecido nuevos requisitos o han cambiado los existentes, afectando a las iteraciones siguientes. Durante la planificación de los detalles de la siguiente iteración, el equipo también examina cómo afectarán los riesgos que aún quedan al trabajo en curso. Toda la retroalimentación de la iteración pasada permite reajustar los objetivos para las siguientes iteraciones. Se continúa con esta dinámica hasta que se haya finalizado por completo con la versión actual del producto.

RUP divide el proceso en cuatro fases (Inicio, Elaboración, Construcción y Transición), dentro de las cuales se realizan varias iteraciones en número variable según el proyecto. Ver Anexo4.

Las primeras iteraciones (en las fases de Inicio y Elaboración) se enfocan hacia la comprensión del problema y la tecnología, la delimitación del ámbito del proyecto, la eliminación de los riesgos críticos y al establecimiento de una línea base de la arquitectura.

Durante la fase de inicio las iteraciones ponen mayor énfasis en actividades de captura de requisitos y modelado del negocio.

En la fase de elaboración, las iteraciones se orientan al desarrollo de la línea base de la arquitectura, abarcan más los flujos de trabajo de requerimientos, modelo de negocios (refinamiento), análisis, diseño y una parte de implementación orientado a la línea base de la arquitectura.

En la fase de construcción, se lleva a cabo la construcción del producto por medio de una serie de iteraciones.

Para cada iteración se seleccionan los Casos de Uso, se refina su análisis y diseño y se procede a su implementación y pruebas. Se realizan tantas iteraciones como sean necesarias hasta que se termine con la implementación de la nueva versión del producto.

En la fase de transición se pretende garantizar que se tiene un producto preparado para su entrega a la comunidad de usuarios.

En cada fase participan todas las disciplinas, pero dependiendo de la fase, el esfuerzo dedicado a una disciplina varía.

1.3- Roles de pruebas según RUP

Un rol define el comportamiento y responsabilidades de un individuo, o de un grupo de individuos trabajando juntos como un equipo. (12)

Una misma persona puede desempeñar varios roles y un mismo rol puede ser representado por varias personas, en dependencia de la cantidad de personas que hayan disponibles en el proyecto para desempeñarlo.

El papel del probador es organizar, conducir y realizar una serie de actividades que permitan determinar la calidad del producto. Las pruebas pueden ser realizadas en conjunto con los responsables de los distintos roles de RUP.

La Metodología RUP define distintos tipos de roles de pruebas, los cuales están presentes durante el Flujo de Trabajo de Pruebas. (12)

El rol de **Ingeniero de componentes** es el responsable de los componentes de prueba que automatizan algunos de los procedimientos de prueba.

Así como el **Ingeniero de pruebas de integración** es el encargado de realizar las pruebas de integración que se necesitan para cada construcción producida en el flujo de trabajo de implementación. Se encarga también de documentar los defectos en los resultados de las pruebas de integración.

Se encuentra también el **Ingeniero de pruebas de sistema** que es el encargado de realizar pruebas al software, necesarias sobre una construcción que muestra y evalúa el resultado de una iteración completa.

Por último está el **Diseñador de Pruebas** el cual planea las pruebas, es el responsable de la integridad del modelo de pruebas, asegurando que el modelo cumpla con su propósito. Selecciona y describe los casos de prueba y procedimientos de prueba. Además define el enfoque de prueba, las configuraciones del ambiente de prueba y los elementos de las pruebas. También identifica los mecanismos de pruebas y estructura la implementación de cada una de ellas.

Las responsabilidades de un rol son tanto el llevar a cabo un conjunto de actividades como el ser el dueño de un conjunto de artefactos.

1.4- Pruebas de Software

Bajo el nombre de pruebas de software se agrupan un conjunto de prácticas correctivas (frente a las prácticas preventivas que se aplican durante el proceso de construcción de software) cuyo objetivo es determinar la calidad de los sistemas software. (13)

Un concepto más específico dado por algunos desarrolladores de software es que las pruebas son: “Cualquier intento de demostrar que el software tiene propiedades por debajo de la calidad requerida”. (14)

Las pruebas requieren que se descarten ideas preconcebidas sobre la corrección del software que se acaba de desarrollar y se supere cualquier conflicto de intereses que aparezcan cuando se descubran errores.

La prueba es el proceso de ejecución de un programa con la intención de descubrir un error. (9)

Según lo anteriormente planteado las pruebas de software son la actividad más común de control de calidad realizada en los proyectos de desarrollo o mantenimiento de aplicaciones y sistemas. Aunque un aseguramiento de calidad de software más eficaz debería incluir otras técnicas como por ejemplo, inspecciones y revisiones (automatizadas o no) de modelos y documentos no ejecutables de las primeras fases de desarrollo.

Las pruebas de software no garantizan que un software esté libre de errores, sino que se detecten la mayor cantidad de defectos posibles en el mismo para su debida corrección. Si al ejecutar las pruebas no se encuentran errores esto indica que las pruebas realizadas no fueron lo suficientemente eficientes.

1.4.1- Objetivos de las pruebas de software

Dentro de los objetivos principales que se persiguen con las pruebas de software en RUP están los siguientes:

- Planificar las pruebas necesarias en cada iteración, incluyendo las pruebas de integración y las pruebas de sistema. Las pruebas de integración son necesarias para cada construcción dentro de la interacción, mientras que las pruebas de sistema son necesarias sólo al final de la iteración.
- Diseñar e implementar las pruebas creando los casos de prueba que especifican qué probar, creando los procedimientos de prueba que especifican cómo realizar las pruebas y creando, si es posible, componentes de prueba ejecutables para automatizar las pruebas.
- Realizar las diferentes pruebas y manejar los resultados de cada prueba sistemáticamente. Las construcciones en las que se detectan defectos son probadas de nuevo y posiblemente devueltas a otro flujo de trabajo, como diseño o implementación, de forma que los defectos importantes puedan ser arreglados. (12)

Según Pressman: “Las pruebas deberían empezar por lo pequeño y progresar hacia lo grande”. (9)

Las pruebas de software se llevan a cabo para brindar un mayor nivel de confiabilidad en los productos que se van generando, detectar fallas o errores y aumentar la calidad del producto final.

1.4.2 -Tipos de pruebas de software que existen

Según RUP, existen cuatro niveles de prueba: unidad, integración, sistema y aceptación. Un software se va perfeccionando mediante las iteraciones que se van realizando durante su ciclo de vida. Durante este proceso, las pruebas de software son fundamentales, permitiendo un fuerte acercamiento a los requerimientos establecidos con el cliente. En cada iteración, el equipo de desarrollo obtiene un resultado, siendo éste el principal candidato para las pruebas.

Basados en esta metodología, existen diferentes tipos de pruebas que se le pueden aplicar al software.

Para saber que un sistema una vez lanzado funcionará correctamente, es necesario realizar una comprobación crítica del mismo. Muchos grupos de trabajo se limitan a dedicar una o dos personas a probar el sistema. Para lograr una mayor calidad en el sistema se llevan a cabo las ***Pruebas de Unidad*** durante la fase de construcción, específicamente en el flujo de trabajo de implementación; las cuales se basan en probar los componentes implementados como unidades individuales.

Aplicar pruebas de unidad a un sistema existente será difícil. Si el sistema es mediano o grande, es conveniente planear sus pruebas y aplicar los cambios necesarios a través de varias versiones futuras.

Las pruebas unitarias aíslan cada parte del programa y muestran que las partes individuales son correctas. A pesar de esto no descubrirán todos los errores del código, no descubren errores de integración, de rendimiento ni otros problemas que afectan a todo el sistema en su conjunto.

Las pruebas de unidad están divididas en dos grupos, en Pruebas de Caja Blanca y Pruebas de Caja Negra.

En las pruebas de Caja Blanca se observa siempre el código, las mismas se realizan para probarlo todo. Estas pruebas se aplican mediante diferentes métodos.

-Prueba de Condición: Es un método de diseño de casos de prueba que ejercita las condiciones lógicas contenidas en el módulo de un programa.

-Prueba de Flujo de Datos: Es un método en el que se seleccionan caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa.

-Prueba de Bucles: Es una técnica de prueba de caja blanca que se centra exclusivamente en la validez de las construcciones de bucles.

-Prueba del Camino Básico: Esta permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución.

Según RUP, las pruebas de Caja Blanca verifican la implementación interna de la unidad. Para cada componente se estudiará su implementación interna y se tratará de verificar su correcto comportamiento algorítmico. Se comprobarán los caminos comunes, los críticos, los menos conocidos y otros asociados con riesgos altos.

Lograr un buen método con pruebas de caja blanca es un objetivo deseable pero no suficiente a todos los efectos. Un programa puede estar perfecto en todos sus términos y sin embargo no servir a la función que se pretende.

Por su parte, las pruebas de caja negra se centran en lo que se espera de un módulo, es decir, intentan encontrar casos en que el módulo no se atiene a su especificación. Por ello se denominan pruebas funcionales y el probador se limita a suministrarle datos como entrada y estudiar la salida, sin preocuparse de lo que pueda estar haciendo el módulo por dentro.

Las pruebas de caja negra están especialmente indicadas en aquellos módulos que van a ser interfaz con el usuario (ejemplo: canales de comunicaciones, etc.).

Según RUP estas pruebas verifican el comportamiento de la unidad observable externamente. Cuando el número de entradas y salidas es grande, se dividen estas en clases de equivalencias. Una clase de equivalencia es un conjunto de valores de entradas o salidas para los que se supone que un componente se comporta de forma similar.

El problema con las pruebas de caja negra no suele estar en el número de funciones proporcionadas por el módulo (que siempre es un número muy limitado en diseños razonables) sino en los datos que se le pasan a estas funciones. El conjunto de datos posibles suele ser muy amplio (por ejemplo, un entero).

Durante la lectura de los requisitos del sistema, se pueden encontrar una serie de valores singulares que marcan diferencias de comportamiento. Estos valores son claros candidatos a marcar clases de equivalencia: por abajo y por arriba.

Una vez identificadas las clases de equivalencia significativas en dicho módulo, se procede a coger un valor de cada clase, que no esté justamente al límite de la clase. Este valor aleatorio, sustituye a cualquier valor normal que se le pueda pasar en la ejecución real.

La experiencia muestra que un buen número de errores aparecen en torno a los puntos de cambio de clase de equivalencia. Hay una serie de valores denominados "frontera" (o valores límite) que conviene probar, además de los elegidos en el párrafo anterior. Usualmente se necesitan 2 valores por frontera, uno justo abajo y otro justo encima.

Las pruebas de caja negra convencen de que un programa hace lo que se quiere pero no de que haga (además) otras cosas menos aceptables.

Al realizar pruebas funcionales lo que se pretende es ponerse en los pies del usuario, usar el sistema como él lo usaría, sin embargo el analista de pruebas debe ir más allá que cualquier usuario, generalmente se requiere apoyo de los usuarios finales ya que ellos pueden aportar mucho en el desarrollo de casos de prueba complejos, enfocados básicamente al negocio, posibles particularidades que no se hayan contemplado adecuadamente en el diseño funcional.

El analista de pruebas debería dar fuerza a las pruebas funcionales y más aún a las de robustez, generalmente los usuarios realizan las pruebas con la idea de que todo debería funcionar, a diferencia del analista de pruebas que tiene más bien una misión destructiva, su objetivo será encontrar alguna posible debilidad y si la llega a ubicar se esforzará porque deje de ser pequeña y posiblemente se convertirá en un gran error, cada error encontrado por el analista de pruebas es un éxito.

Otro tipo de prueba que se llevan a cabo durante esta fase (Construcción) son las ***Pruebas de Integración***, las mismas involucran a un número creciente de módulos y terminan probando el sistema como conjunto.

Estas pruebas se pueden plantear desde un punto de vista estructural o funcional.

Las pruebas estructurales de integración son similares a las pruebas de caja blanca pero trabajan a un nivel conceptual superior. No se refieren a sentencias del lenguaje sino a llamadas entre módulos. Se trata pues de identificar todos los posibles esquemas de llamadas y ejercitarlos para lograr una buena cobertura de segmentos o de ramas.

Las pruebas funcionales de integración son similares a las pruebas de caja negra. Con estas se trata de encontrar fallos en la respuesta de un módulo cuando su operación depende de los servicios

prestados por otro(s) módulo(s). Según se van acercando al sistema total, estas pruebas se basan cada vez más en la especificación de requisitos del usuario.

En todas estas pruebas funcionales se siguen utilizando las técnicas de partición en clases de equivalencia y análisis de casos límite (fronteras).

Las Pruebas de Integración según RUP verifican que los componentes interaccionan entre sí de un modo apropiado después de haber sido integrados en el sistema. Se toman como Casos de Prueba los casos de uso del diseño. Para ello se utiliza el Diagrama de Secuencia correspondiente y se diseñan combinaciones de entrada y salida del sistema que lleven a distintas utilizaciones de las clases y en consecuencia de los componentes, que participan en el diagrama.

Estas pruebas pueden ser realizadas en forma ascendente, esto evita tener que crear módulos emuladores ya que a medida que se va creando la pirámide va siendo probada de abajo hacia arriba, todo esto acarrea un trabajo simétricamente mayor lo que equipara o supera el tiempo que podría tomar el crear módulos para prueba.

Actualmente, los casos de uso tienen una gran aceptación en el desarrollo de sistemas software. Para el éxito de un sistema software es imprescindible garantizar que los casos de uso estén correctamente implementados. Por este motivo, es muy habitual en la industria del software desarrollar pruebas del sistema a partir de los casos de uso. Estas pruebas verifican que se ha implementado todo el caso de uso correctamente.

Durante esta fase, el sistema se emplea de manera experimental para asegurarse que el software no tenga fallas, es decir, que funciona de acuerdo con las especificaciones y en la forma en que los usuarios esperan que lo haga. Se alimentan como entradas conjuntos de datos de prueba para su procesamiento y después se examinan los resultados. En ocasiones se permite que varios usuarios utilicen el sistema, para que los analistas observen si tratan de emplearlo en formas no previstas, antes de que la organización implante el sistema y dependa de él.

Las ***Pruebas del Sistema*** propuestas por la Metodología RUP, prueban que el sistema funciona globalmente de forma correcta. Cada prueba del sistema prueba combinaciones de casos de uso bajo

condiciones diferentes. Se prueba el sistema como un todo probando casos de uso unos detrás de otros y si es posible, en paralelo.

Se trata de ver que cada caso de uso funciona adecuadamente en distintas configuraciones hardware, con varios actores a la vez, en distinto orden. Estas pruebas verifican que el sistema software ofrece a los actores humanos la funcionalidad recogida en su especificación.

Cuando se construye software a la medida para un cliente, se llevan a cabo una serie de ***Pruebas de Aceptación*** para permitir que el cliente valide y verifique todos los requisitos pactados. Estas pruebas las realiza el usuario final en lugar del responsable del desarrollo del sistema.

Es virtualmente imposible que un desarrollador de software pueda prever como utilizará el usuario realmente el programa. Se pueden malinterpretar las instrucciones de uso, se pueden utilizar habitualmente extrañas combinaciones de datos y una salida que puede parecer clara para el responsable de las pruebas y puede ser no entendible para el usuario, hay errores que sólo el cliente puede detectar.

El cliente es quien impone los requisitos, quien mejor que él para dar fe de su satisfacción, además no se debe dejar de la mano la existencia de la calidad percibida que por cruel que parezca, determina en cómo el software será aceptado.

Partiendo de la Metodología RUP, las Pruebas de Aceptación son las que hará el cliente, se determina que el sistema cumple con lo deseado y se obtiene la conformidad del cliente.

En esta fase de pruebas se determina que el sistema cumple con el objetivo deseado, determina la conformidad del cliente antes de que el programa le sea entregado como una versión final. El usuario comprueba en su propio entorno de explotación si acepta el software como está o precisa ser necesario aplicar nuevas optimizaciones y soluciones de fallas.

Se pueden especificar otros casos de prueba para probar el sistema como un todo. Por ejemplo:

Pruebas Negativas: Intentan provocar que el sistema falle para poder así revelar sus debilidades. Los ingenieros de pruebas identifican los casos de prueba que intentan utilizar el sistema en formas para

los que no ha sido diseñado, por ejemplo, utilizando configuraciones de red incorrectas, capacidad de hardware insuficiente o una carga de trabajo imposible. (12)

Por lo planteado anteriormente de las Pruebas Negativas, se puede inferir que las Pruebas de Rendimiento y Penetración o Ataque, se pueden ubicar dentro de la misma.

-Prueba de Penetración: son aquellas pruebas que se encargan de que el sistema revele sus debilidades para posteriormente encontrar soluciones a estos errores.

-Prueba de Rendimiento: consiste en determinar que los tiempos de respuesta están dentro de los intervalos establecidos en las especificaciones del sistema.

1.4.3- Estrategia de Prueba de software

Para obtener un producto de alta calidad, algunos autores como Pressman, plantean que las pruebas de software se deben realizar durante todo el ciclo de vida del mismo. Además para lograr el éxito total se deberá dar seguimiento a través de una estrategia de prueba.

La Estrategia de Prueba de software integra un conjunto de actividades que describen los pasos que hay que llevar a cabo en un proceso de prueba: la planificación, el diseño de casos de prueba, la ejecución y los resultados, tomando en consideración cuánto esfuerzo y recursos se van a requerir, con el fin de obtener como resultado una correcta construcción del software. (9)

Cada prueba que se realice (ya sea unitaria, integración, sistema, validación y verificación, estrés, configuración y/o instalación, aceptación) debe ser planificada con antelación, para de esta manera lograr que los resultados alcanzados sean los esperados y evitar la improvisación a la hora de ejecutarla.

Para esto, es necesario llevar a cabo un Plan de Prueba, en el cual se prevean los recursos necesarios (humanos, materiales y/o financieros), se definen los procedimientos y plantillas que serán utilizados, así como quién debe aplicar las pruebas y cuándo estas deben aplicarse.

Para lograr un mejor control del trabajo a realizar se deben confeccionar listas de chequeos, las cuales sirven para verificar el grado de cumplimiento de cada regla con un fin determinado.

El esfuerzo necesario para el desarrollo de las pruebas adecuadas, puede reducir el tiempo de realización y ejecución de las mismas y disminuir los costos que se generan, siguiendo una adecuada planificación de la Estrategia de Prueba.

1.5 –Sistema SCADA

Los sistemas SCADA utilizan la computadora y tecnologías de comunicación para automatizar el monitoreo y control de procesos industriales. Estos sistemas son partes integrales de la mayoría de los ambientes industriales complejos o muy geográficamente dispersos, ya que pueden recoger la información de una gran cantidad de fuentes muy rápidamente, y la presentan a un operador en una forma amigable. Los sistemas SCADA mejoran la eficacia del proceso de monitoreo y control proporcionando la información oportuna para poder tomar decisiones operacionales apropiadas. (15)

Los sistemas SCADA han eliminado la necesidad de estar físicamente vigilando y ajustando los componentes del proceso: una red de sensores transmite información del estado de los componentes a una sala de operadores que deciden si hay que realizar alguna modificación sobre el proceso. Muchas veces esta toma de decisiones está apoyada por una unidad central que descarga al operario de tareas repetitivas, dejándole actuar sobre el sistema a muy alto nivel.

Las ventajas que aportan los sistemas SCADA los han llevado a ser el corazón informático en la operación de muchas de las infraestructuras que consideramos clave en nuestro día a día:

- Medios de transportes (Control ferroviario, aéreo, tráfico.)
- Utilidades (suministros de electricidad, agua, gas, etc.)
- Centrales Nucleares.
- Sistemas Industriales (Químicas, refinerías, etc.)

Los sistemas SCADA se utilizan en lugares mayormente industrializados en los que sea una necesidad automatizar y controlar los procesos. En estos sistemas los datos son manejados de forma tal que no existan errores fatales, pues como un sistema crítico, estos errores podrían equivaler a la pérdida de vidas humanas.

A grandes rasgos, un sistema SCADA recoge información a través de una red de sensores distribuidos, datos que sirven para crear una representación visual para que los operadores evalúen el estado del proceso y decidan si es necesario tomar algún tipo de acción, que es trasladada de nuevo a las Múltiples Unidades de Terminal Remota (RTU) en forma de comandos de control.

Un sistema SCADA está compuesto por 3 elementos fundamentales (16):

- Múltiples Unidades de Terminal Remota (conocidas como RTU)
- Estación Maestra y ordenador con interfaz hombre-máquina (HMI)
- Infraestructura de comunicación (Ver Anexo 1)

En estos sistemas la adquisición de datos se lleva a cabo a partir de terminales remotas y la información que se recopila es formateada de tal forma que los operadores puedan supervisarla.

Estos sistemas son utilizados para controlar los datos que se generan en los distintos procesos automatizados de la industria en tiempo real, logrando así el ajuste de los valores en el campo y manipular las alarmas que se generen en el sistema general.

Cada uno de estos componentes cumple con funciones específicas que se describen a continuación.

Las Estaciones Remotas (RTU): se conectan al equipo físicamente y leen los datos de estado como los estados abierto/cerrado desde una válvula o un intercambiador, leen las medidas como presión, flujo, voltaje o corriente. Por el equipo las RTU pueden enviar señales que pueden controlarlo: abrirlo, cerrarlo, intercambiar la válvula o configurar la velocidad de la bomba.

Las RTU pueden leer el estado de los datos digitales o medidas de datos analógicos y envía comandos digitales de salida o puntos de ajuste analógicos. Una de las partes más importantes de la implementación de un SCADA son las alarmas. Una alarma es un punto de estado digital que tiene cada valor NORMAL o ALARMA. La alarma se puede crear en cada paso que los requerimientos lo necesiten. El operador de SCADA pone atención a la parte del sistema que lo requiera, por la alarma. Pueden enviarse por correo electrónico o mensajes de texto con la activación de una alarma, alertando al administrador o incluso al operador de SCADA.

La Estación Maestra y ordenador con interfaz hombre-máquina o HMI ("Human Machine Interface"): es el aparato que presenta los datos a un operador (humano) y a través del cual éste controla el proceso.

La industria de HMI nació esencialmente de la necesidad de estandarizar la manera de monitorear y de controlar múltiples sistemas remotos, Controladores Lógicos Programables (PLCs) y otros mecanismos de control. Aunque un PLC realiza automáticamente un control pre-programado sobre un proceso, normalmente se distribuyen a lo largo de toda la planta, haciendo difícil recoger los datos de manera manual, los sistemas SCADA lo hacen de manera automática. Históricamente los PLC no tienen una manera estándar de presentar la información al operador.

La obtención de los datos por el sistema SCADA parte desde el PLC o desde otros controladores y se realiza por medio de algún tipo de red, posteriormente esta información es combinada y formateada. Un HMI puede tener también vínculos con una base de datos para proporcionar las tendencias, los datos de diagnóstico y manejo de la información así como un cronograma de procedimientos de mantenimiento, información logística, esquemas detallados para un sensor o máquina en particular, incluso sistemas expertos con guía de resolución de problemas.

Desde cerca de 1998, virtualmente todos los productores principales de PLC ofrecen integración con sistemas HMI/SCADA, muchos de ellos usan protocolos de comunicaciones abiertos y no propietarios. Numerosos paquetes de HMI/SCADA de terceros ofrecen compatibilidad incorporada con la mayoría de PLCs, incluyendo la entrada al mercado de ingenieros mecánicos, eléctricos y técnicos para configurar estas interfaces por sí mismos, sin la necesidad de un programa hecho a medida escrito por un desarrollador de software.

SCADA es popular debido a esta compatibilidad y seguridad. Ésta se usa desde aplicaciones pequeñas, como controladores de temperatura en un espacio, hasta aplicaciones muy grandes como el control de plantas nucleares.

La Infraestructura de comunicación: no es más que el medio físico que conecta a las diferentes terminales remotas y las estaciona en el sistema. Los sistemas SCADA tienen tradicionalmente una combinación de radios y señales directas seriales o conexiones de modem para conocer los requerimientos de comunicaciones, incluso Ethernet e IP sobre SONET es también frecuentemente usada en sitios muy grandes como ferrocarriles y estaciones de poder (energía eléctrica).

1.5.1- Componentes de hardware en un sistema SCADA

Un SCADA está formado por:

- Ordenador Central o MTU (unidades de terminal central)
- Ordenadores Remotos o RTU´s (unidades terminal remotas)
- Red de comunicación
- Instrumentación de campo

La Unidad Central (MTU) es el ordenador principal del sistema, generalmente es una computadora personal que contiene la HMI. La MTU actúa como intermediaria con el operador e incluye la presentación de los datos y la administración de las alarmas en tiempo real. Se encuentra en el nivel superior o nivel de gerencia en un SCADA. (17)

Las Estaciones Remotas (RTU) están situadas en los nodos estratégicos del sistema gestionando y controlando las sub-estaciones, reciben los datos de los sensores de campo y le envían a los mismos los comandos que llegan desde la estación central. Se encuentran en el nivel intermedio o nivel de automatización. (17)

La Red de Comunicación gestiona la información que se envía a la red de ordenadores. El tipo de implementación es variada y depende de las necesidades y del software escogido para el sistema. Debido a que los componentes pueden estar localizados en diversas áreas, se utilizan sistemas de tipo Red de Área Amplia (Wide Area Network, WAN). (17)

Los instrumentos de campo son todos aquellos dispositivos que permiten tanto la supervisión y el control (PLC (Controlador Lógico Programable), controladores y actuadores) así como la adquisición de los datos (sensores) en el campo. Para la selección de un sistema SCADA es necesario tener en cuenta aspectos como: (17)

- Número de variables del proceso a automatizar
- Si el proceso está distribuido geográficamente
- Si se requiere o no la capacidad de operar en tiempo real

1.5.2- Sistemas SCADA en la actualidad

Hoy en día se han desarrollado múltiples sistemas SCADA por la gran necesidad de automatizar y controlar los procesos en las industrias complejas. Varios de estos sistemas son reconocidos a nivel mundial. A continuación se ejemplifican algunos de ellos:

- Paradym-31, de Advantech
- HMI/SCADA Paragon, de Nematron
- WizFactory, de eMation
- HYBREX(Hybrex Expert System), de SIEMENS
- Genesis32, de Iconics
- LabView, de National Instruments

Todos estos SCADA anteriormente mencionados son propietarios, lo cual no indica que no se han desarrollado alguno de estos sistemas en tecnologías libres, ya que en la actualidad ha tenido un gran avance el código libre. Algunos de estos sistemas se mencionan a continuación:

- FreeSCADA, de Raditex AB (18)
- Visual (19)
- Qscada (20)

Cuba no está exenta de estos grandes avances. Como resultado del estudio realizado se obtuvo información de los distintos sistemas SCADA implantados en el país, entre los que se encuentran Movicon, OASyS, EROS, entre otros.

A pesar de existir a nivel mundial SCADAs que se desarrollaron en tecnologías libres, PDVSA necesitaba desarrollar un nuevo SCADA que cumpliera con la satisfacción de todas sus necesidades. Necesitaban uno lo más apropiado posible para tener un producto con control y una comunidad de desarrollo, debido a esta necesidad es que deciden crear un nuevo SCADA completamente en Linux.

1.6- Estrategias de Prueba en los Sistemas SCADAS

Hoy en día la automatización de procesos es un tema de gran importancia en la actualidad en diversas partes del mundo y junto a esta la presentación y aplicación de estrategias de prueba a herramientas

informáticas ya que se hace necesario el hecho de probarlos buscando un mayor y eficaz funcionamiento. Por otro lado, la importancia de conocer los fundamentos de la automatización de procesos radica en que permitirá a los desarrolladores tener una visión más amplia sobre qué proceso o qué parte del proceso poder automatizar, dándole a su vez la opción de probar cada uno de estos procesos automatizados.

A continuación se ejemplifica una estrategia de prueba aplicada al sistema SCADA Minden, realizada en la actualidad.

Sistema SCADA Minden: Minden es un sistema SCADA utilizado en presas para el control de aguas navegables. En dicho SCADA se llevó a cabo un test de software para realizar pruebas de funcionalidad, para el cual se desarrolló un formulario. Para llevar a cabo la comprobación de dicho formulario y su funcionalidad, se creó un plan de prueba que estaba conformado por cinco categorías de prueba:

- **Test de la superficie gráfica:** verifica si el diálogo contiene todos los objetos necesarios, si éstos pueden ser activados, si funcionan adecuadamente y si su apariencia y composición han sido realizadas correctamente.
- **Test de funcionalidad contextual:** verifica si los elementos existentes satisfacen plena y correctamente sus correspondientes funcionalidades.
- **Test de estrés:** debe proporcionar información acerca de la estabilidad y comprobar si los recursos del sistema (memoria) son liberados una vez finalizada cada acción.
- **Test de entrada errónea:** se pretende comprobar la reacción de la aplicación ante la introducción de datos erróneos, realizando para ello entradas de datos permitidos y no permitidos.
- **Test de operación errónea:** se simulan diversos errores de operación y se comprueban las reacciones del sistema ante las mismas.

Todas estas pruebas mencionadas anteriormente quedaron planteadas en un plan de pruebas realizado, con el que se fija la estructura del procedimiento de prueba.

Lo planteado anteriormente demuestra la gran importancia que tiene el hecho de confeccionar un Plan de Prueba ya que en este quedan plasmadas todas las pruebas por las cuales se va a regir el

probador para probar dicho software, además de la gran importancia que tienen las pruebas de funcionalidad, las cuales demuestran como está realmente la aplicación por la parte funcional.

CAPITULO 2: CARACTERIZACION GENERAL DEL PROYECTO SCADA NACIONAL.

2.1- Introducción

En el año 2004 se declaró la migración de software propietario a software libre en Venezuela, surgiendo en este mismo año el SCADA Nacional.

Debido al papel protagónico que desempeña la UCI en el desarrollo de software en Cuba, es que se asume la tarea de contribuir a la implementación del SCADA, integrando un equipo en el cual trabajan de manera continua 100 estudiantes, 22 profesores y especialistas de todo el país con experiencia en automatización de procesos y desarrollo de Sistemas SCADA; donde había que llevar a cabo el desarrollo de componentes en software libre.

El proyecto SCADA Nacional PDVSA comienza sus labores como proyecto de la facultad 5 a inicios del curso 2007-2008. Su objetivo, desarrollar un Sistema Supervisor de Procesos Automatizados únicamente con tecnología libre, el cual está compuesto por diferentes módulos, los cuales se dividieron entre los dos países que desarrollarían dicho sistema.

2.2- Líneas a desarrollar por la parte cubana y venezolana

Para lograr una mejor distribución de las diferentes líneas de trabajo por las que está compuesto dicho SCADA, se decidió dividir el trabajo entre las dos partes que lo desarrollarían para su mejor funcionamiento. Las líneas asignadas a Cuba para desarrollar el SCADA se eligieron porque Cuba contaba con el personal calificado para desempeñar esta tarea.

El trabajo fue pactado inicialmente para desarrollar 5 líneas por la parte cubana, luego a raíz de los resultados del buen trabajo se pactó el acuerdo para desarrollar 3 líneas más.

- **Base de Datos a Tiempo Real (en conjunto con Venezuela)**

  Problema: Uno de los factores determinantes en la eficiencia y posibilidades de aplicación de un SCADA radica en su capacidad de gestionar la data que se colecta en el campo para su visualización y posterior envío a la base de datos histórica. ¿Cómo puede lograrse la

adquisición eficiente de los datos desde los sensores para el almacenamiento instantáneo, su visualización y envío posterior a la base de datos histórica de modo que se garantice una mayor aplicabilidad del SCADA en desarrollo?

Objetivo: Diseñar una estructura en memoria que permita el almacenamiento instantáneo de los últimos datos, obtenidos de los sensores y dispositivos de campo, para su visualización y envío posterior a la base de datos histórica y el servicio al resto de los módulos del sistema. Comprende tanto los datos puntuales como las tendencias.

- **Middleware**

Problema: En todo sistema distribuido existe una capa de comunicación capaz de interconectar los distintos módulos o componentes que forman parte del sistema, logrando la comunicación y la interoperabilidad, esa capa es el Middleware. Debido a esto surge el por qué el SCADA como sistema distribuido necesita un sistema que comunique las aplicaciones de alto y mediano nivel que forman parte del mismo, dígase la comunicación entre los módulos de BDH (Base de Datos Histórica), Configuración, Seguridad, HMI.

Objetivo: Desarrollar un Middleware basado en TAO que es el ORB de ACE y una implementación de CORBA.

- **Gráficos-HMI (Interfaz Hombre Máquina)**

Problema: No se dispone de una tecnología adecuada para el desarrollo de los componentes que requiere un sistema SCADA sobre Linux para el tratamiento de gráficos vectoriales que resulta imprescindible en la configuración de los sinópticos y su despliegue para la visualización de los operarios. ¿Qué elementos deben conformar el componente de visualización de animaciones vectoriales del sistema SCADA y cómo este debe implementarse para que garantice la gestión de gráficos y animaciones vectoriales en el sistema SCADA y sus despliegues?

Objetivo: Desarrollar una tecnología que permita la gestión de gráficos y animaciones vectoriales sobre Linux, necesarios para la visualización de los sinópticos durante su diseño y explotación.

- **Drivers**

Problema: La comunicación del SCADA con los dispositivos del campo se debe realizar a través de drivers que creen una capa de abstracción del hardware.

Objetivo: Diseño y programación de los drivers que se necesitan para la primera implementación.

- **Reportes**

Problema: El SCADA requiere de un sistema de Reportes que facilite obtener información rápida sobre valores y estadísticas de las variables y que a la vez sea de fácil configuración.

Objetivo: Diseño y programación del submódulo de reportes con el uso del CALC de OpenOffice para la creación de un sistema de reportes sencillo, potente y fiable.

- **Base de Datos Histórica (BDH)**

Problema: El Sistema SCADA requiere de gestores de bases de datos que soporten un alto número de lecturas y escrituras concurrentes.

Objetivo: Evaluar o desarrollar un Sistema de Base de Datos en disco para el almacenamiento persistente de los datos que garantice la posterior visualización en tendencias con la inclusión Mapeo Objeto-Relacional, Bases de Datos de Objetos nativas, y evaluar las ventajas de bases de datos basadas en XML, algoritmos de compresión, mecanismos de almacenamiento acelerados.

- **OPC (OLE (Objeto de Enlace e Incrustación) del Control de Procesos)**

Problema: En PDVSA utilizan tecnologías y software, que implementan este estándar en sus sistemas de control, ya que es muy utilizado en el mundo del control automático por sistemas SCADAS. Por su utilización y debido al propósito de PDVSA de migrar a Software Libre tenía que en este proceso de migración y aún después, tener interacción con este estándar.

Objetivo: Dotar al SCADA Nacional para la interacción con clientes OPC para el acceso a datos publicados de este estándar y dotarlo también de la posibilidad de publicar los datos del SCADA en este estándar mediante la implementación de servidores OPC.

- **Seguridad**

Problema: La seguridad de los sistemas SCADA es uno de los requerimientos de mayor relevancia, no sólo para evitar incidentes fortuitos sino para los intencionados que pueden poner en peligro las operaciones de sistemas de alto riesgo e importancia económica. ¿Qué tecnología de seguridad puede garantizar la inviolabilidad de las normativas de seguridad del SCADA bajo las condiciones de desarrollo con software libre?

Objetivo: Establecer mecanismos de seguridad que garanticen en la mayor medida la inviolabilidad del sistema SCADA bajo software libre. Analizar el uso de sistemas AAA para la seguridad del sistema: Autenticación, Autorización y Auditoría, a todos los niveles incluyendo prevención y recuperación ante desastres.

Por la parte venezolana se le asignaron 4 líneas de trabajo:

- Calidad.
- Pruebas.
- Recolección.
- Configuración.

A pesar de que las líneas de Calidad y Pruebas son desarrolladas por Venezuela, se hacía necesario controlar la calidad de forma interna en las líneas que se llevaban a cabo por la parte cubana ya que la calidad de un software se debe medir desde el comienzo del mismo.

Es por esto que se creó un equipo de calidad por la parte cubana para controlar la calidad con la que contaban los anexos pactados. De ahí el surgimiento de un grupo de apoyo de calidad de la facultad 5 vinculado al proyecto SCADA Nacional, el cual tenía como principal objetivo asegurar la calidad de dicho proyecto, con la aplicación de estándares y normas de calidad requeridas, y a su vez ir desempeñando el rol de probador según los distintos anexos que lo exigían.

2.3- Anexos pactados durante el desarrollo del proyecto

Durante el desarrollo del proyecto, se fueron pactando anexos a entregar por la parte cubana a la parte venezolana, dichos anexos variaron con el transcurso del tiempo de acuerdo a la necesidad del ciclo de vida del software.

Un anexo es sinónimo de entregable, el cual es un producto medible y verificable que se elabora para completar un proyecto o parte de un proyecto. Existen entregables intermedios (internos), que se utilizan para producir los entregables finales que validará el cliente del proyecto.

Entre los anexos pactados para la entrega de las diferentes líneas, se hacía presente la realización de pruebas y análisis de resultados para cada una de ellas. Ejemplo de esto es el caso del Subsistema de Reportes el cual exigía la entrega de código fuente para pruebas de unidad, plantilla de casos de prueba y plan de pruebas. Otro de los anexos pactados fue para el Subsistema de Seguridad, el cual exigía de la realización de pruebas de ataques o pruebas de penetración a dicho módulo y al sistema en general.

2.4- Estándares y Normas de Calidad aplicados al Sistema SCADA

Desde los inicios del SCADA, se decidió basarse en diferentes estándares y normas de calidad para su mejor desarrollo. Las normas y estándares se implantan en las organizaciones de manera voluntaria u obligatoria. Invariablemente, una vez tomada la decisión, se obtienen beneficios para el proyecto.

Un estándar o norma es aplicable a muchos tipos de organizaciones pequeñas, medianas, grandes, locales o internacionales. Por esta razón, la implantación se debe iniciar con un análisis de brecha entre lo que existe y lo que se requiere y además conocer las circunstancias internas y el entorno de la organización de tal manera que la norma o estándar se implante de manera ágil, que sea útil, de bajo costo y acorde a la estrategia de su líder.

Para ello se establecieron un conjunto de prácticas y actividades desde el punto de vista del cumplimiento de RUP al Modelo de Capacidad y Madurez Integrado (CMMI) en los niveles Repetible y Definido, aplicando algunas prácticas de dicho modelo, además del apoyo de algunos estándares como es el caso de la ISO 9001.

2.5- Dificultades detectadas y que tuvieron que ser asumidas

Dos de las líneas asignadas a Venezuela, Calidad y Prueba, en sus inicios no tenían el personal asignado para desempeñarlas. Debido a este inconveniente, Cuba se vio en la tarea de asumirlas en un inicio para que los anexos a entregar por la parte cubana marcharan con la calidad requerida.

Como bien se ha explicado, las líneas a desarrollar se separaron entre Cuba y Venezuela, implementándolas cada país en su respectivo lugar de origen. Dicho proyecto comenzó a funcionar bajo estas condiciones y sigue así en la actualidad.

Todos estos aspectos planteados anteriormente trajeron consigo que no existiera una total relación y una integración plena por parte de los desarrolladores de ambos países, ya que cada cual trabajaba por separado y a larga distancia.

Debido a todo esto se trazó la estrategia de llevar a Venezuela, basándose en la necesidad primaria que hubiese en el momento, a los desarrolladores de las líneas, para que se fuera logrando la integración. Aún con esta idea seguía creada una brecha entre los desarrolladores que se encontraban en Cuba y Venezuela por la parte cubana, ya que nunca el trabajo se llegaba a realizar en paralelo, perdiendo los desarrolladores que se encontraban en Cuba la posibilidad de trabajar sobre la última versión del entregable.

Todas estas dificultades tuvieron que ser asumidas para comenzar con el trabajo, ya que se necesitaba obtener resultados en el tiempo pactado. Aún así nunca el equipo de desarrollo cubano se dio por vencido, y se mantuvo planeando una estrategia para dar solución a dichas dificultades, la cual está basada en desarrollar un canal dedicado a interconectar dos ambientes de trabajo o dos redes. Con una Red Privada Virtual (VPN) se pueden conectar dos redes locales aisladas socialmente, convirtiéndolas luego en una sola red.

Los servicios que prestan normalmente Internet e Intranet se exponen a quien esté conectado del otro lado a través de una VPN con seguridad, o sea, se traza un canal seguro y por ahí se puede enviar toda la información de la forma más segura posible. Es decir, se tiene un servidor y un cliente en cada punta y se intercambian los paquetes con un alto nivel de seguridad.

Con esto se resolvería el problema de no sólo trabajar con la última versión, sino que se pueda realmente salvar o actualizar la información en la que se esté trabajando.

En el proyecto SCADA no se ha implantado aún este sistema, puesto que existe un rango de IP muy grande y cambiante, el cual oscila de 255 a 255. Estas 55 mil posibilidades de direcciones IP que hay, posibilitan a cualquiera tener acceso a la información y recursos del SCADA, es por ello que se sigue trabajando sobre la base de poder reducir dicho rango y utilizar este canal conocido a nivel mundial.

2.6- Condiciones bajo las cuales funciona el proyecto, dentro del polo y de la facultad

El Polo Productivo de Hardware y Automática de la facultad 5 surge a raíz del SCADA, quien le da vida al polo. Dicho proyecto por su magnitud e impacto, tiene buenas perspectivas de comercialización, de desarrollo y de mercado. Todos estos aspectos mencionados en conjunto con los subproductos que se pueden obtener del SCADA son los que le dan espacio y razón de ser al Polo Productivo.

A diferencia de otros proyectos u otros polos, dicho polo realmente se ha constituido de una asociación temática de proyectos y personas. Este proyecto por la magnitud, la importancia de los resultados y las perspectivas de desarrollo que tiene, fueron los que permitieron asociar y crear un polo alrededor de él.

Dentro de este polo ya existe un proyecto de Información Virtual, uno de Supervisión Energética y otros proyectos más en análisis, además de la propia evolución del SCADA que sigue siendo importante.

Todas estas líneas de investigación conforman un gran equipo investigativo con el propósito de fundamentar los conocimientos de cada uno de sus miembros; teniendo como objetivo principal el de delimitar la habilidad investigativa y orientar dichas investigaciones con el propósito de obtener resultados que satisfagan las necesidades deseadas y que sirvan a su vez al desarrollo tecnológico de la facultad en esa rama.

Del SCADA se derivarán productos para Demótica, Supervisión Energética entre otros, es decir, se conformarán nuevos productos y servicios.

2.7- Equipo de Aseguramiento de la Calidad

El equipo de aseguramiento de la calidad del SCADA está conformado por 8 integrantes, cada uno de los cuales está distribuido por los diferentes módulos. Entre las tareas a realizar por el equipo de calidad se encuentran no sólo las de diseñar casos de prueba y desarrollar pruebas, sino que también se trabajó sobre la base de fundamentar en la aplicación existente para el control de los desarrolladores de dicho proyecto (Trac), introduciendo para esto los aspectos de calidad aplicados en el proyecto como son:

- Documentación referente a la Gestión de Configuración.
- El control de los cuadernos de los desarrolladores en su desempeño dentro del proyecto.
- Publicación de los resultados de auditorías realizadas a los distintos módulos con el objetivo de plasmar el avance de cada uno de estos en la realización de las tareas asignadas por el grupo de calidad.

Cada uno de los integrantes de este equipo desempeñó un rol específico, como es el caso del responsable de calidad, diseñador de casos de prueba y probador.

Uno de los principales objetivos trazados por el equipo fue lograr un alto nivel de conocimiento acerca del tema de calidad de software, centrándose principalmente en la Gestión de Configuración, Pruebas de Software y la Planificación y Gestión de Tiempo mediante el uso del Cuaderno del Ingeniero.

Para ello se impartían capacitaciones a los integrantes de los diferentes módulos del proyecto buscando lograr cumplir estos objetivos. Se llevaron a cabo auditorías con el propósito de darle un seguimiento a las tareas que se le asignaban a cada miembro de las diferentes líneas por el jefe de las mismas.

Las plantillas que se utilizaron para el diseño de los Casos de Prueba y listado de No Conformidades detectadas fueron las propuestas por PDVSA, pues en esos momentos en la UCI se trabajaba sobre una nueva filosofía para estandarizar toda la documentación de los proyectos, pero aún no se había oficializado.

La primera versión funcional se encuentra actualmente en fase de pruebas, en Barinas, Venezuela, en tan solo un año y unos meses de desarrollo. Constituye el primer SCADA desarrollado con software libre tanto en Cuba como en Venezuela. Los resultados obtenidos hasta ahora y el prestigio que ha ganado el equipo de trabajo en Mérida han asegurado la posición de los desarrolladores por la parte cubana como equipo de desarrollo para la solución contratada y ha ayudado a alcanzar la Soberanía Tecnológica de Venezuela. En la actualidad lleva el nombre de Guardián del Alba, decisión tomada por la dirección de PDVSA para quitarle el anglicismo.

CAPITULO 3: DESCRIPCION DE LA SOLUCION PROUESTA.

3.1- Introducción

En el siguiente capítulo se detalla la estrategia general que se llevó a cabo para el aseguramiento de la calidad del SCADA mediante las pruebas de software. Esta estrategia apunta fundamentalmente a las pruebas que se realizaron, quiénes participaron en las mismas, además, qué artefactos y resultados se obtuvieron de dichas pruebas.

3.2- La calidad en el Sistema SCADA

El objetivo del Aseguramiento de la Calidad es proporcionar un marco de referencia para el control de la calidad en el desarrollo del SCADA.

Para asegurar la calidad del producto resultante se realizaron una serie de actividades las cuales en su conjunto sirvieron para reducir, eliminar y prevenir las deficiencias de calidad de los productos a obtener, además de satisfacer las necesidades del cliente o usuario en un inicio.

El grupo de aseguramiento de la calidad de SCADA se encargó de determinar que las normas o criterios seleccionados para aplicar en el software estuviesen acordes a los procedimientos planteados en el Plan de Calidad. Dentro de las principales tareas que se desarrollaron por el equipo están las revisiones, las que posibilitaron eliminar o corregir errores a pequeña escala.

Durante el desarrollo del proyecto se elaboraron planes de prueba un poco más detallados, en dependencia del producto a entregar para controlar su calidad en ese momento.

3.3- Identificación de las propiedades de Calidad

Para llevar a cabo la estrategia de pruebas en el Sistema SCADA, se tuvo primeramente presente las propiedades que permiten evaluar la Calidad, dentro de las que se enmarcan las siguientes:

Funcionalidad: El grado en que el software satisface las necesidades indicadas por los siguientes subatributos: idoneidad, corrección, interoperatividad, conformidad y seguridad.

Confiabilidad: Cantidad de tiempo que el software está disponible para su uso. Está referido por los siguientes subatributos: madurez, tolerancia a fallos y facilidad de recuperación.

Usabilidad: Grado en que el software es fácil de usar. Viene reflejado por los siguientes atributos: facilidad de comprensión, facilidad de aprendizaje y operatividad.

Facilidad de mantenimiento: La facilidad con que una modificación puede ser realizada. Está indicada por los siguientes subatributos: facilidad de análisis, facilidad de cambio, estabilidad y facilidad de prueba. (21)

La mayor parte de las tareas y actividades se enfocaron a realizar pruebas de software para alcanzar la calidad pre-establecida y la funcionalidad deseada del sistema.

3.4- Resumen de las Pruebas

Entre los objetivos que se persiguieron para desarrollar las pruebas al Sistema SCADA, se encuentran los siguientes:

- El correcto funcionamiento de los componentes del sistema y detección de errores.
- El funcionamiento correcto de las interfaces de los distintos subsistemas que lo componen.
- La validación de cada uno de los datos y robustez de contraseñas en el sistema, además de la seguridad de la Base de Datos.
- Verificación del envío y recibo correcto de información por parte de los subsistemas.

Primeramente para darle solución a los objetivos perseguidos se elaboró un Plan de Prueba por cada uno de los módulos, donde se recogieron todas las pruebas necesarias que se podían llevar a cabo en ese momento.

Los mismos se encuentran en la siguiente dirección URL:

http://192.168.11.200/svn/scadanac/branches/reportes/pruebas/Plan de Pruebas Reportes/.

http://192.168.11.200/svn/scadanac/branches/drivers/pruebas/Plan de Pruebas Drivers/.

http://192.168.11.200/svn/scadanac/branches/middleware/pruebas/Plan de Pruebas Middleware/.

http://192.168.11.200/svn/scadanac/branches/seguridad/pruebas/Plan de Pruebas Seguridad/.

Se planteó en el mismo además la estrategia de pruebas que sería puesta en práctica donde se abordaron los diferentes tipos de pruebas a realizar y las diferentes categorías de errores que se podrían producir, entre las que se encuentran las enfocadas a:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y terminación.

Se describieron también las técnicas que serían utilizadas para llevar a cabo las pruebas así como el proceso de las mismas, el cual constó de varios pasos:

- Preparación para la ejecución de las pruebas.
- Medición.
- Evaluación.

Se elaboró en conjunto al Plan de Prueba una Lista de Chequeo, la misma aborda aspectos relacionados fundamentalmente a la evaluación de la interfaz de usuario del sistema, se trató de abarcar con esto todos los aspectos de usabilidad necesarios, así como los de confiabilidad, eficiencia y portabilidad.

Dicha Lista de Chequeo consta de un formato en forma de cuadro, lo que permite un llenado rápido de los distintos casilleros, verificando en los mismos la regla en cuestión. Se pueden contestar con un Sí o un No, o bien tildar los casilleros para los casos en que se verifica la regla, dejando el espacio en blanco si no se cumple.

Esta Lista de Chequeo se encuentra publicada en la siguiente URL:

http://192.168.11.200/svn/scadanac/branches/calidad/Lista de Chequeo/.

Parte de esta Lista de Chequeo se muestra en el Anexo9.

Los tipos de pruebas aplicados en el Sistema SCADA son los siguientes:

- Pruebas Unitarias.
  - Pruebas de Funcionalidad.
  - Pruebas de Caja Blanca.
- Pruebas Negativas.
  - Pruebas de Rendimiento.
  - Pruebas de Penetración.

Cada una de estas pruebas se realizó debido a la necesidad de cumplir con los anexos pactados que lo exigían. Algunos de estos anexos incluían pruebas unitarias para algunos módulos en particular, como es el caso del Módulo de Reportes, debido a la importancia que tiene la misma, ya que mediante estas se logra garantizar la ejecución de al menos una vez todos los caminos independientes de cada módulo y se ejerciten todas las estructuras de datos para garantizar su validez.

Entre los aspectos que incluían los anexos pactados se encontraban además las pruebas de penetración al Módulo de Seguridad. Dichas pruebas exigían de una base de la arquitectura del sistema por la cual estaba constituido el SCADA, basándose además en el estudio de la estructura de la red del sistema.

Se decidió aplicar las pruebas de rendimiento al Módulo de Middleware debido a que en un sistema de Supervisión, Control y Adquisición de Datos, el Middleware es la capa de software que se encuentra por encima de los niveles físicos y de transporte y por debajo de las capas de gestión y aplicaciones de usuario. Su función radica en establecer un método de comunicación entre las aplicaciones que se encuentran ejecutándose en cualquier punto del sistema, siendo su localización transparente a las demás. Se necesitaba verificar el correcto envío y recibo de información por dicho canal de comunicación, y a su vez la eficiencia del tiempo con que debía realizar esta acción según los requerimientos pactados.

Nunca se realizaron las pruebas de integración debido a que para las mismas se necesitaba de un gran número de módulos los cuales terminan probando el sistema en conjunto, lo cual no estaba

presente en el SCADA debido a la lejanía existente entre ambos grupos de desarrollo. Este tipo de prueba hubiese sido de gran ayuda ya que la misma contribuye a conocer si los módulos interaccionan entre sí de una forma correcta después de haber sido integrados en el sistema.

Es por ello que tampoco se pudieron llevar a cabo las pruebas de sistema ya que para las mismas se necesitaba que el sistema funcionara como un todo, así como tampoco se efectuaron las pruebas de aceptación por la parte cubana.

Todo lo planteado anteriormente constituye el pilar fundamental de la estrategia de pruebas del SCADA Nacional.

3.5- Plantillas

La estructura general de la plantilla a usar para el diseño de casos de pruebas emitida por PDVSA se muestra en el Anexo5.

Pasos a seguir para su llenado.

Propósito de los diseños de prueba: Realizar una descripción detallada de los pasos a seguir por el probador para probar un caso de uso.

Dicho formulario se genera por el diseñador de casos de prueba y es usado por el probador.

Partes de la plantilla:

Encabezado

Esta parte contiene la información del nombre del proyecto, nombre del caso de prueba, versión, revisiones históricas y descripción general donde se da una breve descripción del número de casos de pruebas que se le desarrollarán al caso de uso correspondiente.

Detalle:

En esta parte de la plantilla se encuentra contenido todo el proceso paso por paso de cómo el probador debe ejecutar cada caso de prueba, contando este proceso con un flujo central, un flujo

alterno si fuese necesario, condiciones de ejecución bajo las que se ejecutó el caso de prueba y las iteraciones donde se registran las clases válidas e inválidas según los datos suministrados, resultado esperado, el resultado obtenido y las observaciones necesarias.

La plantilla utilizada para describir las no conformidades detectadas se muestra en el Anexo6.

Pasos a seguir para su llenado.

Propósito del listado de no conformidades: Realizar una descripción detallada de los problemas detectados en los casos de prueba.

Dicho formulario es usado por el probador.

Partes de la plantilla:

Encabezado

Esta parte contiene la información del nombre del proyecto, nombre del módulo, versión, revisiones históricas y aspectos generales donde se da una breve descripción de los elementos probados y los no probados y las causas por las cuales no fueron probados.

Detalle:

En esta parte de la plantilla se encuentra contenido todo el proceso paso por paso de cómo el probador expone los elementos que fueron probados, las no conformidades detectadas, el aspecto correspondiente, la etapa de detección, la importancia y la recomendación.

3.6- Procedimientos para realizar las pruebas

El diseño de las pruebas y los resultados de las mismas, quedaron establecidos en el documento “Diseño de Casos de Prueba” por los diferentes módulos y se almacenaron en el repositorio del SCADA Nacional; por lo que se podrá acceder a ellos por las siguientes URL:

http://192.168.11.200/svn/scadanac/branches/reportes/pruebas/.

http://192.168.11.200/svn/scadanac/branches/drivers/pruebas/.

http://192.168.11.200/svn/scadanac/branches/middleware/pruebas/.

http://192.168.11.200/svn/scadanac/branches/seguridad/pruebas/.

No obstante a continuación se muestran algunos de los ejemplos de casos de prueba diseñados para formalizar las pruebas del software.

Procedimiento de Generación de Casos de Prueba

Para los casos de prueba de las ***pruebas funcionales***, la mayor información se obtuvo de la especificación de casos de uso. Para diseñar los casos de pruebas, se tomaron los requerimientos funcionales a implementar.

A continuación se muestran algunos ejemplos de casos de pruebas diseñados.

- Módulo: Reportes.

  Caso de Uso: Diseñar Reportes.

  Caso de Prueba:

  - ✓ Diseñar partiendo de plantillas pre-elaboradas.

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
Marcar cada una de las agrupaciones dando clic sobre las pestañas.		El sistema debe mostrar cada una de las agrupaciones con sus respectivas plantillas pre-elaboradas enunciadas en la especificación de requisitos del Subsistema		

		Reportes.		
Seleccionar inicialmente del grupo "Sistema" la plantilla pre-elaborada "Reportes de falla" y seguidamente la plantilla pre-elaborada “Reporte de eventos del sistema”.		El sistema desmarca automáticamente la plantilla pre-elaborada seleccionada inicialmente.		
	El mantenedor intenta tocar otra funcionalidad del sistema, cuando está abierta la ventana “Nuevo diseño”.	No permite realizar dicha operación.		
	Presionar el botón “Aceptar” sin seleccionar ningún tipo de plantilla pre-elaboradas.	No permite realizar dicha operación.		

Caso de Prueba:

- ✓ Abandonar proceso de diseño.

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
Cuando la ventana “Abrir” esté abierta se cierra por el		El sistema debe cerrar la ventana “Abrir”		

botón “Cerrar” que se encuentra en la parte superior derecha de dicha ventana.		y volver a la ventana principal.		
Cuando la ventana “Abrir” esté abierta se da clic en el botón cancelar.		El sistema debe cancelar la acción y volver a la ventana principal.		
En el escaque “Nombre de Archivo” teclear el nombre del diseño “Producción Turno-Poso1-Extracción-Maracaibo” y presionar el botón “Abrir”.		El sistema carga y muestra el diseño.		
	En el escaque “Nombre de Archivo” teclear el nombre del diseño “Producción Turno-poso3-Extracción-Maracaibo” y presionar el botón “Abrir”.	El sistema no debe ejecutar ninguna acción.		
	Presionar el botón “Abrir” sin haber seleccionado ningún diseño.	El sistema no debe ejecutar ninguna acción.		

Caso de Uso: Salvar Diseño.

Caso de Prueba:

- ✓ Salvar Diseño.

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
Después de haber guardado el diseño		El sistema muestra una interfaz que		

“Producción Turno-Poso1-Extracción-Maracaibo”, cambiar la propiedad de uno de los componentes y realiza el Flujo alterno 2.3.1		permite al mantenedor entrar los datos de configuración del diseño, dándole la opción de “Salvar” o “Cancelar”.		
Seleccionar la opción “Salvar” de la clase válida anterior.		El sistema muestra un mensaje diciendo que ese nombre ya existe y si está seguro que desea modificarlo.		
Cuando la ventana “Guardar” esté abierta se cierra por el botón “Cerrar” que se encuentra en la parte superior derecha de dicha ventana.		El sistema debe cerrar la ventana “Guardar” y volver a la ventana principal donde está el diseño creado.		
Cuando la ventana “Guardar” esté abierta se da clic en el botón cancelar.		El sistema debe cancelar la acción y volver a la ventana principal donde está el diseño creado.		
Después de haber guardado el diseño “Fallas diciembre 2007”, cambiar la propiedad de uno de los componentes y realiza el flujo alterno 2.3.2		Se guarda el diseño automáticamente con el mismo nombre.		
Después de haber guardado el diseño “Producción Turno-Poso2-Extracción-Maracaibo”, cambiar la propiedad de uno de los componentes y presionar el icono “Guardar” que esta en la barra de herramientas.		Se guarda el diseño automáticamente.		
	Al intentar salvar no indicar ningún	Debe aparecer inactiva la opción “Salvar”.		

	nombre al diseño.			

Caso de Uso: Insertar Componentes.

Caso de Prueba:

- ✓ Insertar Texto.

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
El mantenedor mueve el componente Texto fuera del área de diseño.		El puntero del Mouse cambia su apariencia a su paso por fuera del área permitida para su inserción.		
Seleccionar la tecla "ESC" para elegir otra opción u otro componente cuando se este arrastrando al componente “Texto”.		El puntero del Mouse vuelve a tomar la apariencia normal, mostrando en el sistema las propiedades del anterior componente insertado o en caso de que este sea el primer componente insertado, se mostrarán las propiedades del área de diseño.		
Se escribe el texto “Alarmas Día 20 de Diciembre”.		Se escribe el texto en el componente insertado.		
	El mantenedor mueve el componente Texto fuera del área de diseño, e intenta	El sistema indica un error de que en esa área no se puede insertar el componente, con esta acción se desmarca la opción de insertar el componente.		

	colocarlo.			
	Se desmarca el componente y se intenta escribir el texto “No se puede escribir”	No se escribe el texto.		

Caso de Prueba:

- ✓ Editar Sección.

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
Marcar las siguientes opciones de la ventana del “Editor Secciones”: -Encabezado de Reporte -Pie de Reporte -Encabezado de Página -Pie de Página		El sistema muestra en el diseño dichas secciones según se van marcando.		
Estando abierta la ventana “Propiedades Detalles Sección”, se introducen los siguientes valores: -Nombre: “Detalles_Alarmas” -Seleccionar Consulta “Alarmas_Activas” y acto seguido presionar el botón “Cerrar”.		El sistema agrega el detalle “Detalles_Alarmas” a la lista de detalles de la sección satisfactoriamente.		
Seleccionar un detalle de la lista de detalles y presionar el botón “Eliminar”		El sistema elimina dicho detalle y todos los posibles agrupamientos creados y asociados al mismo satisfactoriamente.		

Seleccionar un detalle de la lista de detalles habiendo más de uno y presionar el botón “Mover Arriba”		El sistema mueve el detalle una posición por encima de donde se encontraba en la lista de detalles. Probar esto con grupos creados.		
Seleccionar un detalle de la lista de detalles habiendo más de uno y presionar el botón “Mover Abajo”.		El sistema mueve el detalle una posición por debajo de donde se encontraba en la lista de detalles. Probar esto con grupos creados.		
Estando abierta la ventana “Editor Grupo Secciones”, entrar los datos: -Nombre Grupo: “Alarmas_Por_Areas” -Columna: “Valida” -Seleccionar “Mostrar Encabezamiento de Grupo” -Seleccionar “Mostrar Pie de Grupo” y presiona el botón “Aceptar”.		El sistema agrega el grupo “Alarmas_Por_Area” al detalle “Detalles_Alarmas” satisfactoriamente.		
Estando abierta la ventana “Editor Grupo Secciones”, entrar los datos: -Nombre Grupo: “Alarmas_Por_Area” -Columna: “Valida” -No seleccionar “Mostrar Encabezamiento de Grupo” -Seleccionar “Mostrar Pie de Grupo“ y presiona el botón		El sistema agrega el grupo “Alarmas_Por_Area” al detalle “Detalles_Alarmas” satisfactoriamente.		

"Aceptar"				
	El mantenedor intenta tocar otra funcionalidad del sistema, cuando está abierta la ventana "Editor Secciones".	El sistema no realiza la operación deseada por el diseñador y se mantiene con la ventana abierta.		
	Estando abierta la ventana "Editor Grupo Secciones", entrar los datos: -Nombre Grupo: "Alarma$_Activas" -Columna: "Valida" -Seleccionar "Mostrar Encabezamiento de Grupo" -Seleccionar "Mostrar Pie de Grupo" y presiona el botón "Aceptar"	El sistema muestra un mensaje de error indicando que el campo "Nombre" no es correcto.		
	Estando abierta la ventana "Propiedades Detalles Sección", se introducen los siguientes valores: -Nombre: " " -Seleccionar Consulta "Alarmas_Activas" y acto seguido presionar el	El sistema muestra un mensaje de error indicando que el campo "Nombre" es obligatorio llenarlo.		

	botón “Cerrar”.			
	Estando abierta la ventana “Propiedades Detalles Sección”, se introducen los siguientes valores: -Nombre: “Detalle$_Alarmas” -Seleccionar Consulta “Alarmas_Activas” y acto seguido presionar el botón “Cerrar”.	El sistema muestra un mensaje de error indicando que el campo “Nombre” es inválido.		
	El mantenedor presiona los botones “Editar”, “Eliminar”, “Mover Arriba” y “Mover Abajo” sin seleccionar ningún detalle de la lista de detalles.	El sistema no realiza ninguna operación.		

Para los casos de prueba de las ***pruebas caja blanca***, la mayor información se obtuvo de las clases o funciones del sistema a probar. Para diseñar los casos de pruebas, se tomaron los métodos que serían probados de una clase en específico.

- Módulo: Drivers.

Casos de prueba de la clase “QtSerialTransport ”

Clase de prueba: “unitTestQtSerialTransport ”

Casos de prueba del método: “connect”

Utils::DWord connects (Utils::DWord timeout);

Nota: Las pruebas que siguen corresponden a las particiones de equivalencia identificadas para los valores de entrada del método en cuestión:

Particiones de equivalencia:

1- 0 <= timeOut

2- timeOut > 0

Casos de Prueba:

1- void testConnect_EP1()
2- void testConnect_EP2()

Formato de pruebas unitarias del método "connect".

# Caso de prueba	Objetivo de la prueba	Datos de entrada	Salida esperada	Salida obtenida	# Clases de equivalencia cubiertas	Observaciones
1	Verificar que cuando el valor del parámetro timeOut esta en el rango admisible es decir que es mayor que 0. Entonces el valor de retorno es 0.	800	0	0	1	--
2	Verificar que cuando el parámetro timeout es menor que 0 el valor de retorno es el errBind.	-20	Utils::errBind = 50	Utils::errBind = 50	2	--

Casos de prueba del método: “read”.

Utils::DWord read();

Nota: Las pruebas que siguen corresponden a las particiones de equivalencia identificadas para los valores de entrada del método en cuestión:

Particiones de equivalencia:

1- 0 <= timeOut

2- timeOut > 0

3- buffer != 0

4- buffer = 0

5- size > 0

6- size <= 0

Casos de prueba:

1- void testRead_PE1()

2- void testRead_PE2()

3- void testRead_PE3()

4- void testRead_PE4()

5- void testRead_PE5()

6- void testRead_PE6()

Para los casos de prueba de las ***pruebas de rendimiento,*** la mayor información se obtuvo de la especificación de casos de uso. Para diseñar los casos de pruebas, se tomaron los requerimientos funcionales a implementar.

- Módulo: Middleware.

  Caso de Uso: Receive Information of Data.

  Caso de Prueba:

  ✓ Recibir datos complejos del tipo punto.

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
Recibir 50000 datos de tipo punto. -Se espera recibir desde un modulo o consola de envío.		Se reciben todos los puntos especificados en el envío.		
Poner a recibir 10 módulos o consolas para el envío de datos. -Se está esperando recibir desde 2 módulos o consolas.		Se reciben en cada una de las consolas la cantidad que se especifica en el envío.		
Recibir 20000 datos de tipo punto con una		Se esperan recibir la cantidad total de datos que se especificó en el envío.		

consola o módulo. -Se está esperando recibir desde 2 módulos o consolas de envío. Recibir datos de tipo punto en 3 consolas o módulos. -Se espera recibir desde 3 módulos o consolas.		Se esperan recibir la cantidad total de datos que se especificó en el envío.		

Caso de Prueba:

- ✓ Recibir datos complejos del tipo alarma.

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
Recibir 50000 datos de tipo alarma. -Se espera recibir desde un módulo o consola de envío. Poner a recibir 10 módulos o consolas para el envío de datos. -Se está esperando recibir desde 2 módulos o consolas.		Se reciben todos los puntos especificados en el envío. Se reciben en cada una de las consolas la cantidad que se especifica en el envío.		

Recibir 20000 datos de tipo alarma con una consola o módulo. -Se está esperando recibir desde 2 módulos o consolas de envío.		Se esperan recibir la cantidad total de datos que se especificó en el envío.		
Recibir datos de tipo alarma en 3 consolas o módulos. -Se espera recibir desde 3 módulos o consolas de envío.		Se esperan recibir la cantidad total de datos que se especificó en el envío.		

Caso de Uso: Send Events.

Caso de Prueba:

- ✓ Enviar Evento.

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
Se envían 100 eventos desde un módulo o una consola de envío.		Se envían la cantidad de eventos especificados desde el módulo o consola de envío.		
Se envían 30000 eventos desde 4 módulos o consolas de envío.		Se envían la cantidad de eventos especificados desde cada uno de los módulos o consolas de envío.		

Caso de Uso: Receive UserLog.

Caso de Prueba:

- ✓ Recibir log de usuario.

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
Poner a recibir 1 módulo o una consola para la recepción de userlog. -Se está esperando recibir desde 1 módulo o consola.		Se recibe en el módulo o consola la cantidad que se especifica en el envío.		
Recibir 40000 userlog con 1 módulo o consola. -Se está esperando recibir desde 5 módulos o consolas de envío.		Se esperan recibir la cantidad total de userlog que se especifico en el envío.		

Para los casos de prueba de las ***pruebas de penetración.*** Para diseñar los casos de pruebas, se tomaron los requerimientos funcionales y no funcionales a implementar, además de basarse de la arquitectura del SCADA.

- Módulo: Seguridad

  Casos de Pruebas:

  - ✓ Reconocimiento de la red.
  - ✓ Inyección a la autenticación.

Reconocimiento de la red

Prueba 1: Enumeración de host.

Herramienta usada: SuperScan.

Sistema Operativo: Windows.

Ver Anexo7.

Prueba 2: Enumeración de puertos.

Se conocen los puertos de escucha de los servicios activos.

Fueron encuestados 13 host para un total de 106 puertos abiertos.

Ver Anexo8.

Prueba 3: Enumeración de host.

Herramienta usada: NMap.

Sistema Operativo: Linux

nmap -sP 192.168.10.1-255

-sP Scanea host activos sin salir del rango especificado.

Prueba 4: Enumeración de puertos.

nmap -O -sX -T4 -D 1.2.3.4,5.6.7.8 192.168.10.11
-O Detecta el SO.
-sX
-T4 Realizar una ejecución más rápida.

-D Engañar haciendo creer que el testeo a la red se está realizando desde los IP 1.2.3.4 y 5.6.7.8

Inyección a la autenticación

Prueba 1: Inyección con comilla simple (‘).

Prueba 2: Inyección con doble guión (--).

Prueba 3: Colapso del sistema.

Para cada uno de los tipos de prueba se suministra la siguiente información:

- Caso de prueba.
- Si se aplica, documento contra el cual debe basarse la prueba.
- Breve descripción del criterio de entrada y salida.

3.7- Procedimientos de Ejecución y Reporte de Errores

Después que los casos de pruebas han sido desarrollados y el ambiente de pruebas ha sido preparado, se está listo para iniciar la ejecución de las mismas.

Para ello se tuvo en cuenta como primer paso a seguir, la elaboración de un equipo de trabajo que no fueran los propios desarrolladores del sistema, poniendo en práctica los principios básicos en la realización de las pruebas, ya que de esta manera se evitaba una simple verificación del correcto funcionamiento del programa.

Para el reporte claro y preciso de problemas en el proceso de pruebas, se elaboró un formulario nombrado Documento de No Conformidades, el cual está asociado con la documentación de los problemas identificados durante las pruebas, quedando registrados en éste todos los resultados de las mismas.

El Documento de No Conformidades describe los problemas detectados y las recomendaciones para dar solución a cada uno de éstos.

El probador analiza el caso de prueba, siguiendo las especificaciones paso a paso descritas en el mismo.

Si la ejecución de un caso de prueba produce resultados incorrectos o inesperados, se llena la plantilla del Documento de No Conformidades. La siguiente información debe ser registrada en dicha plantilla:

- Fecha en que se ejecutó el caso.
- Versión del caso de prueba.
- Nombre del probador.
- Descripción del problema o error.
- Elementos probados.

- Elementos no probados y causas.
- No Conformidades Detectadas.
- Recomendaciones.
- Anexos.

Al terminar el caso de prueba, se adjuntaron todos los documentos producidos durante la prueba, como informes, imágenes de errores, etc.

Se dice que una prueba ha llegado a su fin cuando se ejecutaron todos los casos de prueba exitosamente y no quedan problemas a resolver.

3.8- Resultados Obtenidos de las Pruebas

Una de las etapas más importantes en el desarrollo de las pruebas la constituye los resultados que se obtienen en las mismas, los cuales permiten medir la calidad con la que se encuentra el producto probado.

En cada uno de los módulos del sistema, se realizaron según la necesidad, una serie de iteraciones en cada una de las pruebas, debido a que hubo pruebas que tuvieron que ser diseñadas sin tener aún la aplicación existente terminada.

Como resultados de las pruebas quedó conformado el Documento de No Conformidades, donde se ponen de manifiesto todos los resultados obtenidos en la realización de las mismas, como es el caso de la evaluación del rendimiento del Middleware donde se mostraron resultados negativos en cuanto al recibo de puntos, ya que se recibieron más puntos de los enviados.

Se evaluaron también los resultados del Módulo de Seguridad en el cual no se tenía aún implementado un Firewall o Corta Fuegos con el objetivo de negar la entrada de intrusos al sistema. En los resultados de las pruebas de funcionalidad en el Módulo de Reportes se detectaron errores en cuanto al funcionamiento de algunos requerimientos que se exigían. Debido a todas las iteraciones realizadas a los diseño de las pruebas, es que se obtuvo la calidad requerida en los anexos pactados.

Para más información se puede acceder a la siguiente URL donde se encuentra publicado el listado de No Conformidades Detectadas:

http://192.168.11.200/svn/scadanac/branches/calidad/Lista No Conformidades/.

En cada iteración se fueron obteniendo resultados reales del estado en el que se encontraba el sistema hasta ese momento, los que fueron evolucionando positivamente en el transcurso de vida del software. Finalmente los resultados obtenidos por parte del equipo de calidad fueron todos satisfactorios cumpliendo con los objetivos trazados, obteniendo de esta forma la principal meta del equipo de calidad, la cual era lograr el perfeccionamiento iterativo del software para su entrega.

CONCLUSIONES

Mediante el presente trabajo se logró dar cumplimiento a su principal objetivo el cual consistía en diseñar y aplicar una estrategia de prueba en el proyecto SCADA Nacional. Para darle inicio a dicha investigación se planificaron las pruebas de caja blanca, funcionalidad, rendimiento y pruebas de penetración, las cuales quedaron plasmadas en el plan de pruebas por cada módulo del proyecto.

La estrategia de prueba se planteó siguiendo la metodología que ya se había adoptado en el proyecto, es decir, la metodología tradicional RUP, haciendo un estudio detallado de los diferentes roles que plantea la misma para el flujo de trabajo de prueba. Los casos de prueba diseñados fueron acorde a lo exigido en los anexos pactados, utilizando para ello plantillas para el diseño de los casos de prueba y lista de chequeo, siendo el pilar fundamental de todo el estudio, la realización del plan de pruebas.

Luego de varias iteraciones de la ejecución de las pruebas, cada uno de los defectos encontrados se reflejó en el listado de no conformidades, persiguiendo con esto obtener los entregables finales con la calidad requerida.

RECOMENDACIONES

Por lo planteado en dicha investigación y los resultados obtenidos se recomienda:

- Seguir aplicando la estrategia de prueba desarrollada en el proyecto SCADA Nacional.
- Ampliar más en la investigación referente a estrategias de pruebas en Sistemas SCADAs a nivel mundial y nacional.
- Utilizar la estrategia propuesta para la aplicación de las pruebas del proyecto SCADA Nacional en otros proyectos productivos del polo de automática de la facultad de entornos virtuales.

BIBLIOGRAFÍA

1. **Martínez, Ana Silvia Tellería.** *Diseño de un Middleware básico para el sistema SCADA de PDVSA.* Ciudad de la Habana : s.n., 2007.

2. *Anexo 5 Diagnóstico del Proyecto SCADA Nacional del Convenio Marco PDVSA – ALBET, S.A.* Caracas : s.n., 2006.

3. **Myers, Glenford J.** *The Art of Software Testing.* 2004.

4. **Brito, Irina y Napal, Irina.** *Las pruebas de software, su aplicación al Config. Case.* Ciudad de la Habana : s.n., 2003.

5. **Delgado, Ramsés y González, Emilio.** *Confiscase 3.0 Herramienta de apoyo a la Gestión de Configuración. Propuesta arquitectónica.* Ciudad de la Habana : s.n., 2006.

6. *ISO 8402.* 1994.

7. *ISO 9001.* 2000.

8. *IEEE Std 610.* 1990.

9. **Pressman, R.S.** *Ingeniería del Software. Un enfoque práctico.* 2002.

10. *Calidad del software (I).* **Arechavala, Yolanda González y García, Fernando de Cuadra.** País Vasco : s.n., 2001, Vol. 1.

11. **Kruchten, P.** *The Rational Unified Process: An Introduction.* 2000.

12. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software.* Madrid : Pearson Educacion S.A, 2000. 84-7829-036-2.

13. **Juristo, N., Moreno, A. y Vegas, S.** *Limitations of Empirical Testing Technique Knowledge.* New Jersey : s.n., 2003.

14. cigitallabs. *cigitallabs.* [En línea] 05 de 11 de 2002. [Citado el: 15 de 04 de 2008.] http://:www.cigitallabs.com/resources/definitions/software_testing.html.

15. **Montero, Dagoberto, Barrantes, David B. y Quirós, Jorge M.** *Introducción a los sistemas de control supervisor y adquisición de datos (SCADA).* Costa Rica : s.n., 2004.

16. autómatas. *autómatas.* [En línea] 2005. [Citado el: 18 de 03 de 2008.] http://www.autómatas.org/redes/scadas.htm.

17. **Videaux, Leonel Salazar y Neyra, Raúl Pérez-Alejo.** *Módulo de Visualización de Gráficos Vectoriales para Aplicaciones SCADA.* Ciudad de la Habana : s.n., 2007.

18. freescada. *freescada.* [En línea] 2005. [Citado el: 25 de 03 de 2008.] http://www.freescada.com/freescada.php.

19. visual. *visual.* [En línea] 2004. [Citado el: 06 de 04 de 2008.] http://visual.sourceforge.net/new/index.php.

20. qscada. *qscada.* [En línea] 2006. [Citado el: 12 de 04 de 2008.] http://qscada.sourceforge.net/.

21. *ISO 9126.* 2001.

GLOSARIO

C

CMMI: Modelo de Capacidad y Madurez.

D

Documento de No Conformidades: Documento en el que se recogen los incumplimientos de los procedimientos definidos formalmente para el software y que pueden repercutir directa o indirectamente en la satisfacción del cliente y en el cumplimiento de las metas del proyecto.

E

Estrategia de prueba: un conjunto de actividades que describen los pasos que hay que llevar a cabo en un proceso de prueba.

H

HMI: Estación Maestra y ordenador con interfaz Hombre-Maquina.

I

ISO: Organización Internacional para la Estandarización.

L

Lista de Chequeo: listado de preguntas en forma de cuestionario que sirve para verificar el grado de cumplimiento de determinadas reglas establecidas a priori con un fin determinado.

M

Middleware: Capa de comunicación que interconecta los diferentes módulos que forman partes de un sistema, logrando la comunicación y la interoperabilidad.

Módulos: partes de un sistema.

MTU: Unidades de Terminal Central.

O

OLE: Es un estándar que permite la incrustación y vinculación de objetos (imágenes, clips de vídeo, sonido MIDI, animaciones, etc.) dentro de ficheros (documentos, bases de datos, hojas de cálculo, etc.).

P

PDVSA: Petróleos de Venezuela Sociedad Anónima.

Plan de Prueba: informe en el cual se recogen todas las pruebas necesarias que se llevan a cabo en un proyecto.

Plantilla: forma de dispositivo que proporciona una separación entre la forma o estructura y el contenido.

PLC: Controlador Lógico Programable.

PSN: Proyecto SCADA Nacional.

R

RTU: Múltiples Unidades de Terminal Remota.

S

SCADA: Sistemas de Supervisión y Adquisición de Datos.

U

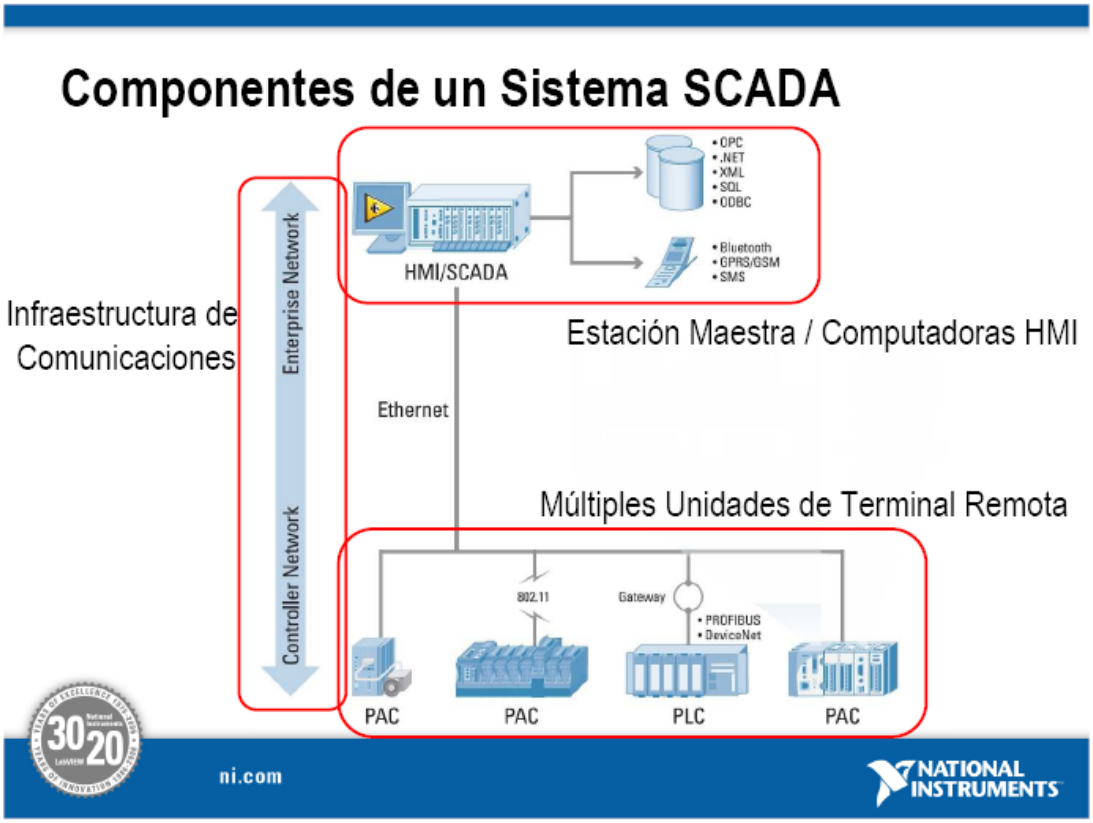
URL: es un localizador uniforme de recursos.

V

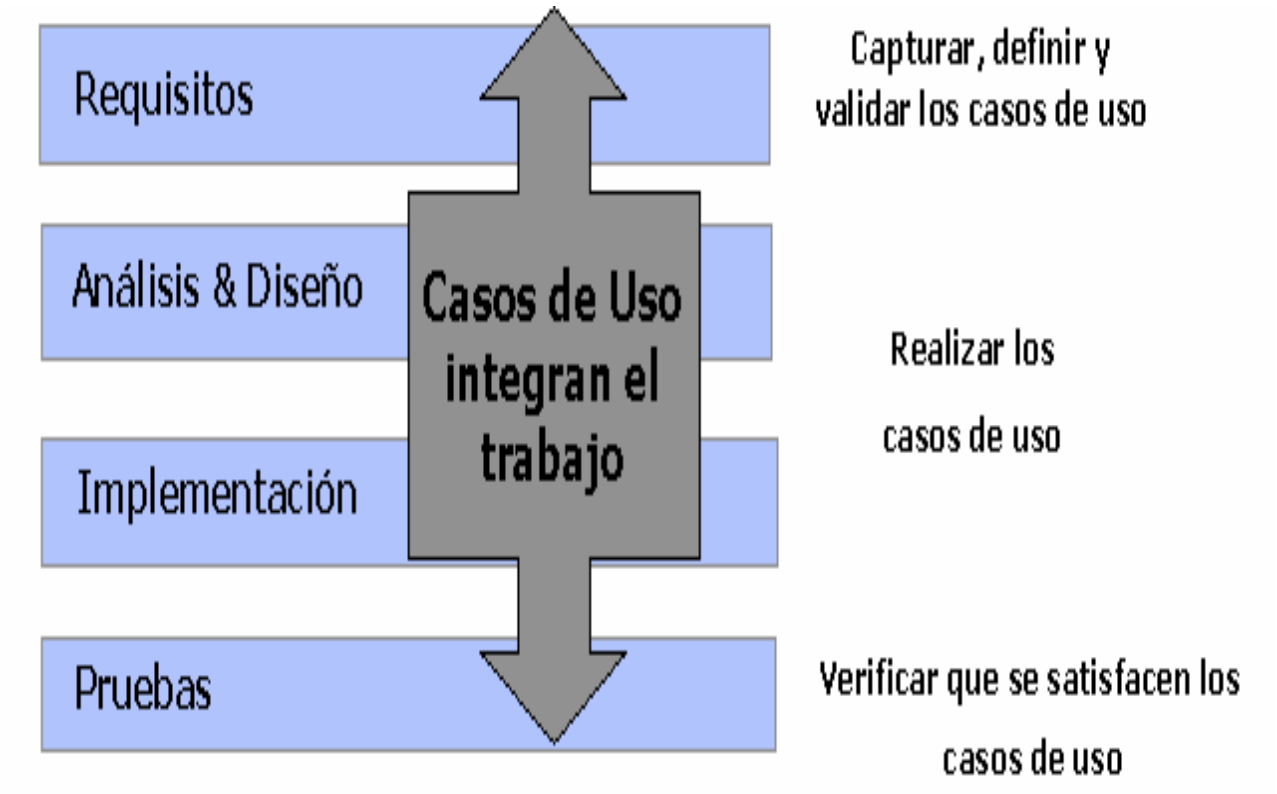
VPN: Red Privada Virtual.

ANEXOS

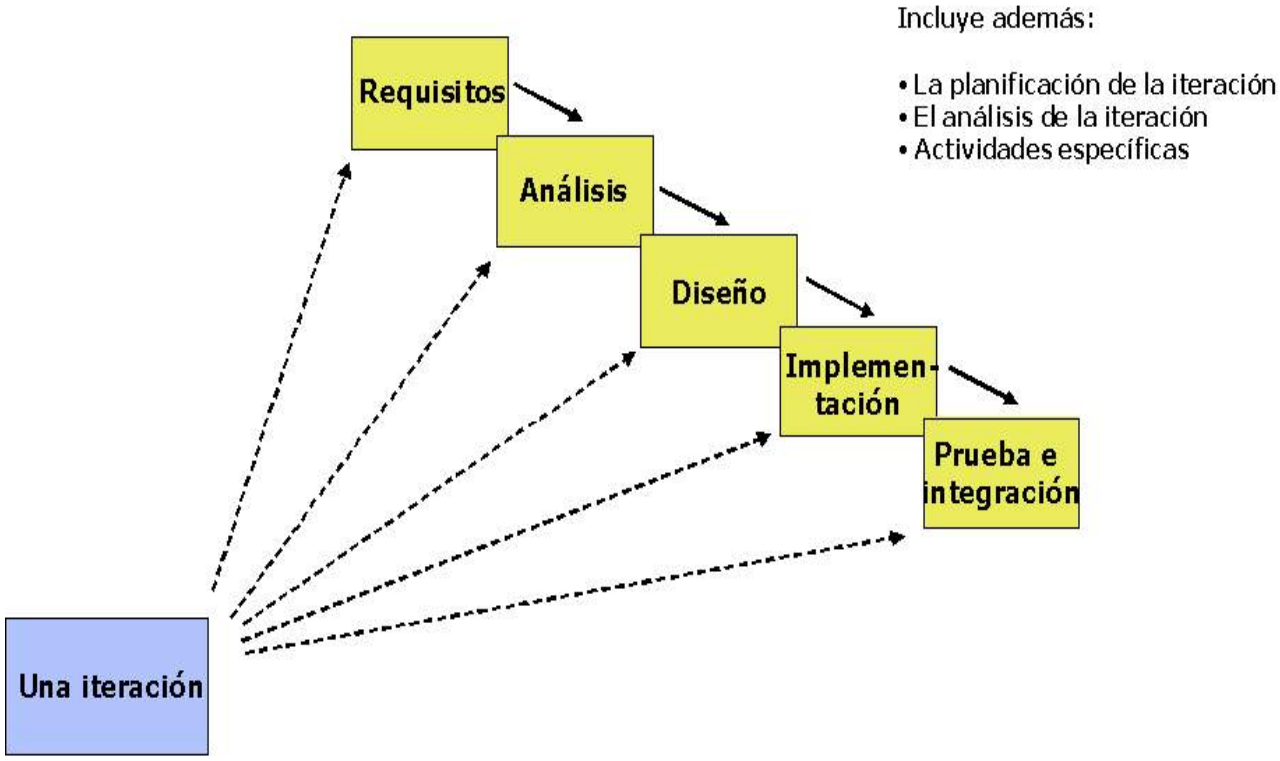
Anexo1



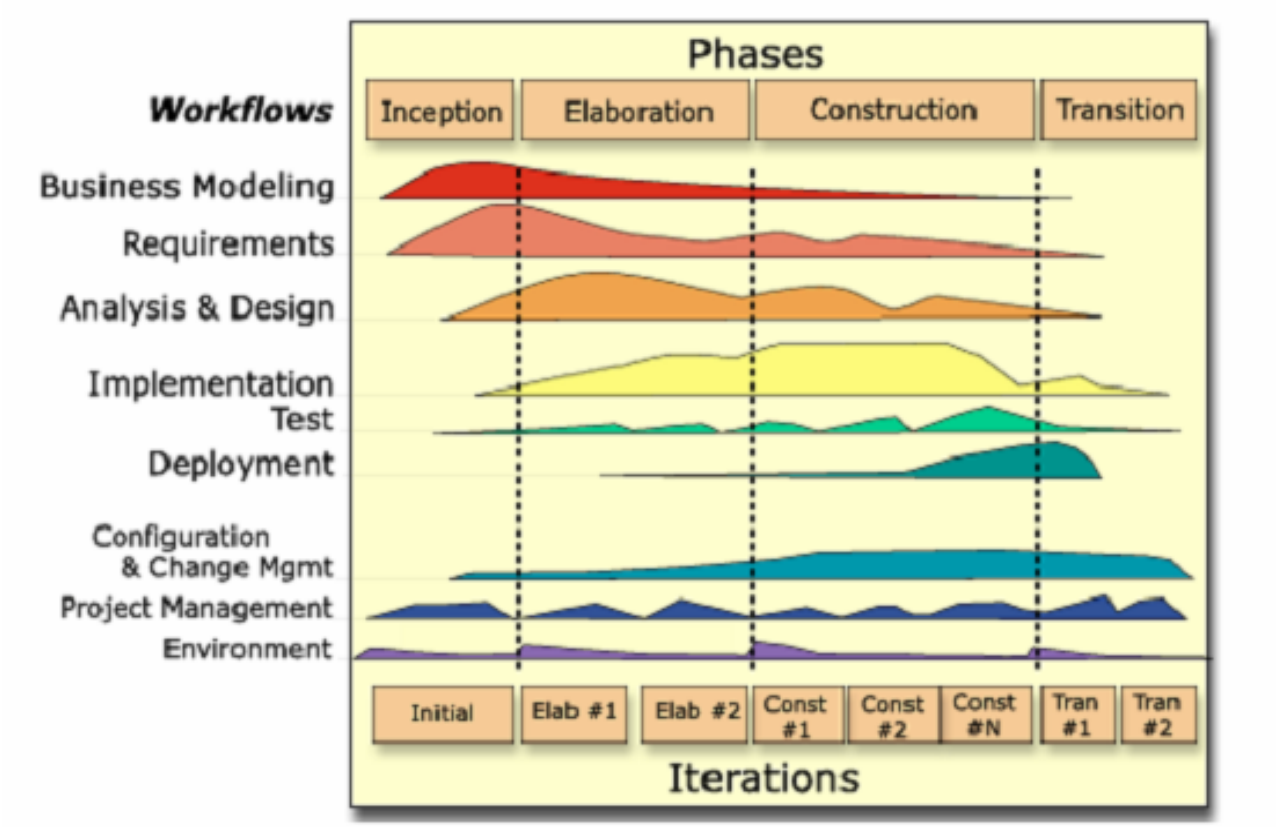
Anexo2


Requisitos
Análisis & Diseño
Implementación
Pruebas
Casos de Uso integran el trabajo
Capturar, definir y validar los casos de uso
Realizar los casos de uso
Verificar que se satisfacen los casos de uso

Anexo3


Incluye además:
• La planificación de la iteración
• El análisis de la iteración
• Actividades específicas
Requisitos
Análisis
Diseño
Implemen-
tación
Prueba e
integración
Una iteración

Anexo4


Phases
Workflows
Inception
Elaboration
Construction
Transition
Business Modeling
Requirements
Analysis & Design
Implementation
Test
Deployment
Configuration
& Change Mgmt
Project Management
Environment
Initial
Elab #1
Elab #2
Const #1
Const #2
Const #N
Tran #1
Tran #2
Iterations

Anexo5

Nombre del Proyecto

Diseño Caso Prueba

Nombre Caso Prueba

Versión

Revisiones Históricas

Fecha	Versión	Descripción	Autor (es)

Descripción General

-Número de Casos de Pruebas

Caso Prueba 1.

-Flujo Central

-Flujo Alterno.

-Condiciones de Ejecución.

-Iteraciones

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado Obtenido	Observaciones

Registro de Defectos y Dificultades Detectadas

Elemento	No	No Conformidad	Aspecto Correspondiente	Etapa Detección	Importancia	Recomendaciones

Anexos

Anexo6

Nombre del Proyecto

Nombre del Módulo

Plantilla de No Conformidades

Versión

Revisiones Históricas

Fecha	Versión	Descripción	Autor (es)

1. Aspectos generales.

[Descripción de Aspectos Generales a tener en cuenta a la hora de analizar el resultado de las pruebas, incidencias en el momento de su desarrollo y otros aspectos relevantes.]

1.1 Elementos probados.

[Descripción general o lista de los Elementos Probados, y otros aspectos importantes a tener en cuenta a la hora analizar las No Conformidades Detectadas.]

1.2 Elementos No Probados y Causas.

[Descripción general o lista de los Elementos no Probados, la causa de que no se hayan podido realizar las Pruebas y cualquier otro elemento importante que aporte la información necesaria para que sean analizadas estas causas y resueltas para la siguiente iteración.]

2. Tabla de No Conformidades Detectadas

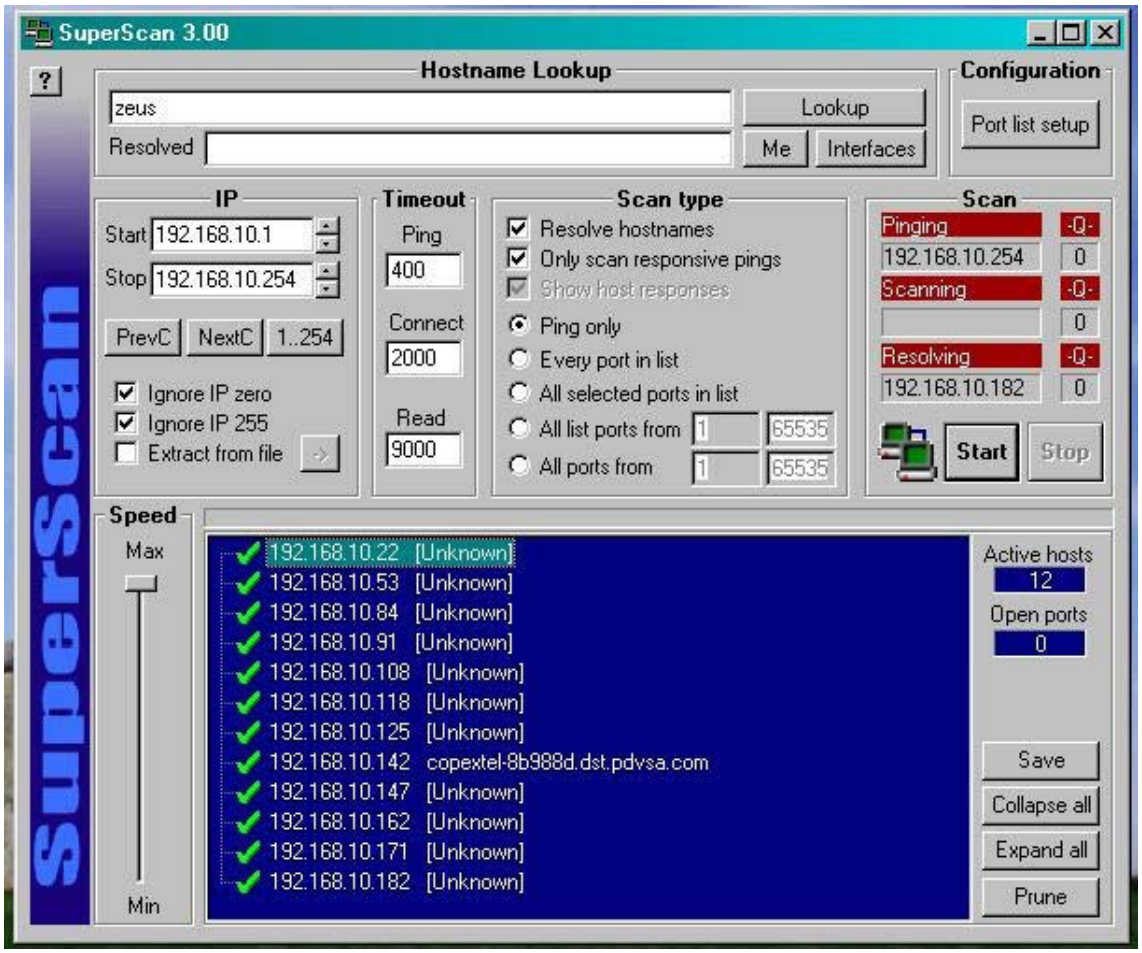
Elemento	No	No Conformidad	Aspecto Correspondiente	Etapa Detección	Importancia	Recomendaciones
<Nombre del Elemento>	< 1>	<Descripción de la No Conformidad>	<Descripción del Aspecto correspondiente>	<Etapa de detección del error>	<X>	<X>

3. Recomendaciones

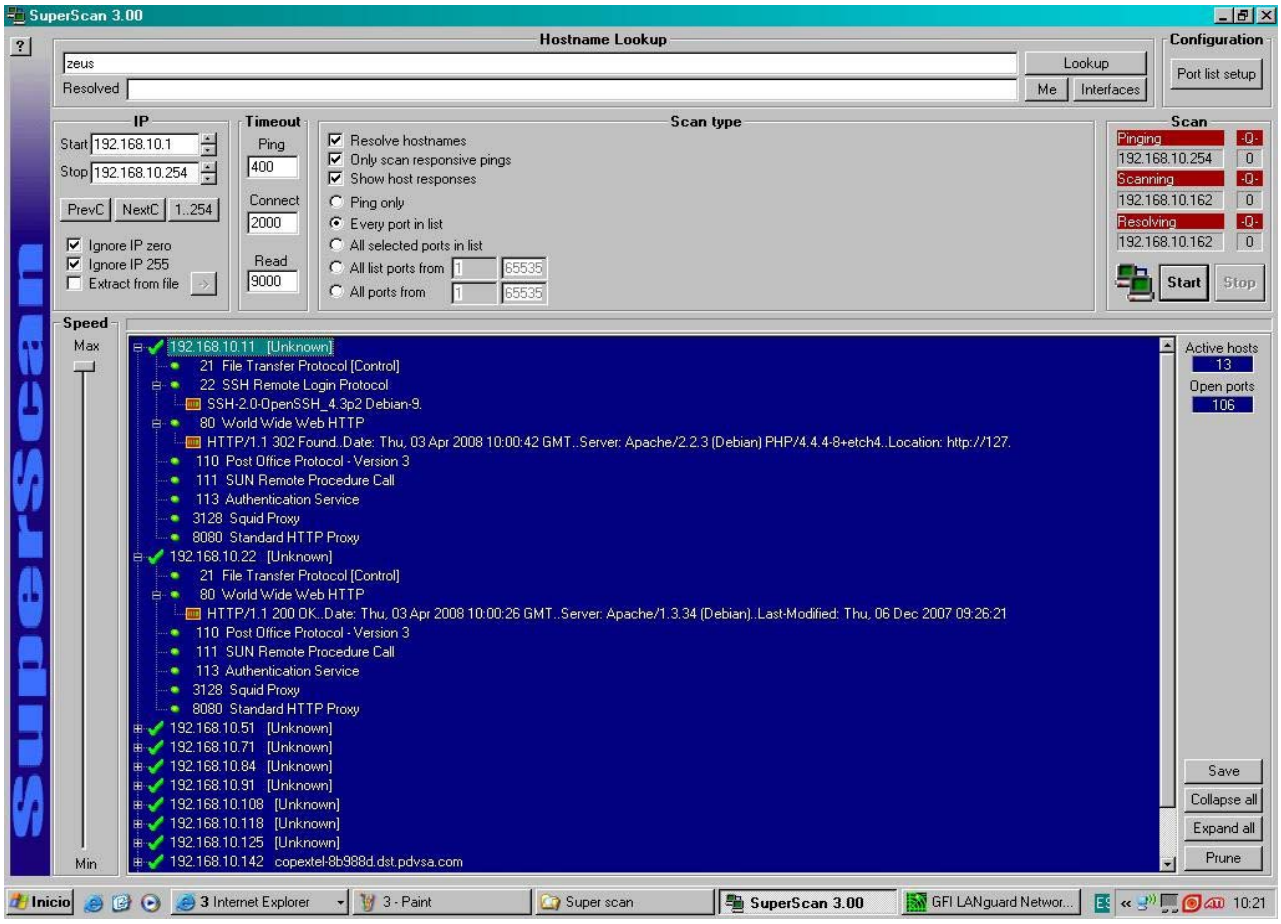
Elemento	No	No Conformidad	Aspecto Correspondiente	Etapa Detección
<Nombre del Elemento>	< 1>	<Descripción de la No Conformidad>	<Descripción del Aspecto correspondiente>	<Etapa de la detección del error>

4. Anexos

Anexo7


SuperScan 3.00
Hostname Lookup
zeus
Lookup
Resolved
Me
Interfaces
Configuration
Port list setup
IP
Start 192.168.10.1
Stop 192.168.10.254
PrevC
NextC
1..254
Ignore IP zero
Ignore IP 255
Extract from file
Timeout
Ping
400
Connect
2000
Read
9000
Scan type
Resolve hostnames
Only scan responsive pings
Show host responses
Ping only
Every port in list
All selected ports in list
All list ports from 1 65535
All ports from 1 65535
Scan
Pinging -Q-
192.168.10.254 0
Scanning -Q-
0
Resolving -Q-
192.168.10.182 0
Start
Stop
Speed
Max
Min
192.168.10.22 [Unknown]
192.168.10.53 [Unknown]
192.168.10.84 [Unknown]
192.168.10.91 [Unknown]
192.168.10.108 [Unknown]
192.168.10.118 [Unknown]
192.168.10.125 [Unknown]
192.168.10.142 copextel-8b988d.dst.pdvsa.com
192.168.10.147 [Unknown]
192.168.10.162 [Unknown]
192.168.10.171 [Unknown]
192.168.10.182 [Unknown]
Active hosts
12
Open ports
0
Save
Collapse all
Expand all
Prune
SuperScan

Anexo8


SuperScan 3.00
Hostname Lookup
zeus
Resolved
Lookup
Me
Interfaces
Configuration
Port list setup
IP
Start 192.168.10.1
Stop 192.168.10.254
PrevC
NextC
1..254
Ignore IP zero
Ignore IP 255
Extract from file
Timeout
Ping
400
Connect
2000
Read
9000
Scan type
Resolve hostnames
Only scan responsive pings
Show host responses
Ping only
Every port in list
All selected ports in list
All list ports from 1 65535
All ports from 1 65535
Scan
Pinging -0-
192.168.10.254 0
Scanning -0-
192.168.10.162 0
Resolving -0-
192.168.10.162 0
Start
Stop
Speed
Max
Min
192.168.10.11 [Unknown]
21 File Transfer Protocol [Control]
22 SSH Remote Login Protocol
SSH-2.0-OpenSSH_4.3p2 Debian-9.
80 World Wide Web HTTP
HTTP/1.1 302 Found..Date: Thu, 03 Apr 2008 10:00:42 GMT..Server: Apache/2.2.3 (Debian) PHP/4.4.4-8+etch4..Location: http://127.
110 Post Office Protocol - Version 3
111 SUN Remote Procedure Call
113 Authentication Service
3128 Squid Proxy
8080 Standard HTTP Proxy
192.168.10.22 [Unknown]
21 File Transfer Protocol [Control]
80 World Wide Web HTTP
HTTP/1.1 200 OK..Date: Thu, 03 Apr 2008 10:00:26 GMT..Server: Apache/1.3.34 (Debian)..Last-Modified: Thu, 06 Dec 2007 09:26:21
110 Post Office Protocol - Version 3
111 SUN Remote Procedure Call
113 Authentication Service
3128 Squid Proxy
8080 Standard HTTP Proxy
192.168.10.51 [Unknown]
192.168.10.71 [Unknown]
192.168.10.84 [Unknown]
192.168.10.91 [Unknown]
192.168.10.108 [Unknown]
192.168.10.118 [Unknown]
192.168.10.125 [Unknown]
192.168.10.142 copextel-8b988d.dst.pdvsa.com
Active hosts
13
Open ports
106
Save
Collapse all
Expand all
Prune
Inicio
3 Internet Explorer
3 - Paint
Super scan
SuperScan 3.00
GFI LANguard Networ...
10:21

Anexo9

EVALUACIÓN DE LA INTERFAZ DE USUARIO

Proyecto: SCADA Nacional

Control N°: _____ **Etapa:** _____

Fecha: _____

N°	Evaluación	Si	No	NP	Comentario
1.	¿Está la información libre de errores gramaticales, ortográficos y de los errores tipográficos?				
2.	¿Los íconos representan las acciones a las que están asociados?				
3.	¿Existe el menú de "archivo"?				
4.	¿Existe el menú de la "ayuda"?				
5.	¿Los comandos y las opciones apropiadas están en cada menú?				
6.	¿Cada una de las opciones da las respuestas apropiadas?				
7.	¿Se tiene ausencia de errores críticos al probar todas las opciones?				
8.	¿Los botones en la barra de herramientas se corresponden con comandos del menú?				
9.	¿Cada comando de menú tiene una secuencia hot-key la cuál lo invocará cuando sea apropiado?				
10.	En cajas de diálogo "tabuladas", ¿los nombres de la lengüeta o pestañas no son abreviaturas?				
11.	En cajas de diálogo "tabuladas", ¿las lengüetas o pestañas se pueden alcanzar vía combinaciones de hot-key apropiadas?				
12.	¿En las cajas dialogadas, no existen hot-key duplicadas?				
13.	¿Las lengüetas están colocadas horizontalmente?				
14.	¿La funcionalidad de la tecla escape es correcta? (la cuál debe cancelar cualquier cambio que se haya realizado).				
15.	¿La función del botón de cancelación es igual que la tecla escape?				
16.	¿El botón de cancelación se activa cuando se están realizando cambios?				
17.	Un botón de comando debe habilitarse cuando la acción se corresponda con su uso y no en otras				

	ocasiones, ¿el botón esta habilitado o no lo está cuando le corresponde?				
18.	¿Los botones de aceptación y de cancelación están agrupados de forma separada de los otros botones de comandos?				
19.	¿Los nombres de los botones de comandos no son abreviaturas?				
20.	¿Los nombres de los botones de comandos no son etiquetas técnicas, pero si nombres que tienen un significado para los usuarios del sistema?				
21.	¿Los botones de comando son de tamaño y forma similares?				
22.	¿Cada botón de comando se puede alcanzar vía combinación de hot-key?				
23.	¿Los botones de comandos en la misma caja de dialogo no tienen hot-key duplicadas?				
24.	¿Cada caja de dialogo tiene el valor prefijado claramente marcado?				
25.	¿El focus está fijado a un objeto?				
26.	¿Los nombres de los botones de opción (botón de radio) no son abreviaturas?				
27.	¿Los nombres de los botones de opción no son etiquetas técnicas?				
28.	¿Los nombres de la caja de opción no son abreviaturas?				
29.	¿Los nombres de la caja de opción no son etiquetas técnicas y son significativas a los usuarios?				
30.	¿Las hot-key se utilizan para tener acceso a las cajas de opción?				
31.	¿Las cajas de opción, los botones de opción, y los botones de comando están agrupados lógicamente juntos en áreas claramente demarcadas?				
32.	¿Cada área está demarcada?				
33.	¿La ventana del padre tiene una barra de estado?				
34.	¿La consistencia de acciones del ratón a través de las ventanas es la correcta?				
35.	¿Cuando se abre la aplicación, esta ocupará la posición que ocupaba cuando se cerró anteriormente?				
36.	¿El doble click en la barra de "Título" es equivalente a maximizar y volver a la posición anterior?				