Universidad de las Ciencias Informáticas

Facultad 7

Título: Proceso de Pruebas del Registro de Áreas de Salud de la Atención Primaria del Sistema de Información para la Salud

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor(es): Yosbel Ernesto González Alonso

Yoan Manuel Cabrera Arribas

Tutor(es): Lic. Claribel Lucy Cruz Águila

Co-tutor: Ing. Keidy García Lira

Asesor: Lic. Anett García Montkowski

DECLARACIÓN DE AUTORÍA

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 firmo la presente a los 22 días del mes de junio del año 2007

Yosbel Ernesto González Alonso
Firma

Yoan Manuel Cabrera Arribas.
Firma

Lic. Claribel Lucy Cruz Águila
Firma

DATOS DE CONTACTO

Tutor: Lic. Lucy Cruz Águila

Especialista de la dirección de Desarrollo de la Empresa SOFTEL, graduada de Licenciatura en Cibernética-Matemática en el año 1988.
Ha desarrollado diferentes proyectos de gestión en la esfera de la salud, turismo, empresarial, ministerial, ha dirigido técnicamente proyectos de gestión en estas áreas.
Ha participado en diferentes eventos científicos-técnicos en temas de calidad del desarrollo de software.
Ha recibido diferentes cursos de post-grados en ingeniería de software, programación y calidad.
Correo electrónico: lucycruz@softel.cu

Ubicación: SOFTEL, UCI, Cuba

Co-tutor: Ing. Keidy García Lira.

Graduada en el año 2004 de Ingeniería Informática en ISPJAE

Ha impartido clases de Ingeniería del Software en la facultad 3 de la Universidad de Ciencias Informáticas.

Correo electrónico: keidy@uci.cu

Ubicación: Facultad regional de Artemisa (UCI)

Asesora: Lic. *Anett Garcia Montkowski*
Graduada en el año 2005 de Licenciada en Economía en la Universidad de Pinar del Río.
Ha impartido clases de Comercio electrónico, Administración de Empresas y Metodología de la Investigación en la facultad 7 de la universidad de ciencias Informáticas.
Correo electrónico: anettgm@uci.cu
Ubicación: Facultad 7 (UCI)

PENSAMIENTO

La fama es premio justo de quien tiene
el valor de sacrificar el grato sigilo de
su persona a la idea que defiende.
José Martí

DEDICATORIA

A:

Mi madre, mi padre y mis hermanos.

Yoan Manuel Cabrera Arribas

DEDICATORIA

A:

Mi madre, mi padre, mi hermana y mis sobrinos Heriberto y Kisandra.

Yosbel Ernesto González Alonso

AGRADECIMIENTOS

Al Comandante en Jefe Fidel Castro Ruz por permitirnos ser parte de este gran sueño hoy hecho realidad, "La Universidad de Ciencias Informáticas".

A la Revolución cubana y a su pueblo trabajador.

A nuestra tutora Lucy, por poner a nuestra disposición sus conocimientos, tiempo y entrega, por ser una certera guía y una exelente educadora capaz de transmitirnos sus experiencias a partir de consejos útiles.

A nuestra cotutora Keidy García Lira por su apoyo incondicional.

A los amigos de la FEU, que nos apoyaron en todo momento.

A Ariannes Milagros Cruz Pérez por todo lo que nos enseñó, para poder triunfar ante las adversidades.

A Humberto Brito Santana por su constante preocupacion para con nosotros.

A Danieski Rodrigues Osoria (Kiki) por confiar en nosotros cuando muchos pensaron que no...

A Yoanky Pérez Vega por cooperar siempre con nosotros.

A Yanet Poza por su integración a nuestro equipo y su ayuda para la conclusión del trabajo.

A Yaimí Márquez Alpizar por aportar parte de su tiempo y conocimientos.

A Eikel Inda Ramos por alentarnos siempre y confiar en nosotros.

A Mirna Cabrera por sabernos entender, guiar y demostrarnos con su ejemplo, que cuando se persevera, se triunfa.

A Caridad Guzmán, Carmenchu y al Dr. Denis Derivet por la ayuda durante las incontables consultas que nos brindaron.

A los factores de nuestra facultad.

A todos los que de una forma u otra se preocuparon y nos preguntaban ¿y la tesis qué?

En fin a todos aquellos que influyeron en nuestra formación como ingenieros y hombres de bien.

AGRADECIMIENTOS

A mi mamá y mi papá por su preocupación constante por mis estudios y deberes, por hacer de mí un hombre de bien inculcándome modales y valores muy útiles para la vida, por sus útiles y oportunos consejos, por la educación que supieron darme, por su ejemplo, y por todas las cosas lindas que heredé de ellos.

A mi madrastra Rebeca, por siempre comprenderme y ayudarme, por sus útiles consejos y apoyo, por hacer de ella un hijo he influir en mi formación y guiarme por el buen camino.

A mi padrastro Diosdado porque a pesar de los pesares cuenta con un gran corazón que no todos logran entender.

A mis hermanos Manolito y Yordan por los cuales me he esforzado para que vean en mi no ejemplo a seguir, sino una meta a superar por mucho.

A mis maestros, maestras, profesores y profesoras que aportaron su grano de arena para mi formación.

En fin, todos los que de una forma u otra influyeron en mi para para que hoy yo pudiese escribir estas lineas.

A todos muchas gracias.

Yoan Manuel

AGRADECIMIENTOS

A mi mamá y mi papá que me han apoyado siempre en todos mis proyectos personales. A mi hermana Lisy por todo el apoyo que siempre he recibido de ella, a Mairienis por soportar mis peleas, mis alteraciones y por sobre todas las cosas, por estar siempre a mi lado. A mi tía Cary, a abuelo Oscar por siempre preocuparse y preguntarme por los estudios, a mi abuela Luisa esté donde esté.

A Tuto, a mis hermanos Osbel y Cachy, a Lauria, a María, Rolando, Rosita, Cheo y a mis queridos suegros Cidalisis y Alfredo.

A los buenos amigos de la FEU, a Velmour por ser tan especial conmigo.

A los mejores amigos de estos años Maylen, María Isabel, Yeni, Yoanky y Mara.

A Yoan Manuel ahora mi compañero de tesis, antes y para siempre mi hermano de mil batallas y mi amigo.

A Liesner Acevedo y Joel Arencibia por su amistad y por todo lo que me ayudaron.

A todos mis profesores, a los vecinos del barrio, y tantos otros que durante todos estos años han influido en mi formación como hombre de bien, como revolucionario y como profesional.

A todos muchas gracias por hacerme lo que soy.

Yosbel Ernesto

RESUMEN

El presente trabajo, tiene como objetivo ejecutar el proceso de pruebas al Registro de Áreas de Salud, del proyecto SISalud, siguiendo el flujo de trabajo de pruebas propuesto. Lo que resulta necesario, debido a que en algunos casos este proceso, solo se tiene en cuenta en la etapa de liberación, por lo que es de baja calidad. En otras ocasiones, la premura en la entrega de los productos provoca que se omitan pasos necesarios, en flujos anteriores y el tiempo puede extenderse incumpliéndose los cronogramas de trabajo.

El flujo de trabajo de pruebas se elaboró a traves del estudio de las estrategias, niveles y tipos de pruebas de software recomendadas en la bibliografía especializada. Se tomaron como guía fundamental los flujos de trabajos de pruebas propuestos en la Metodología RUP propuesta por el Proceso Unificado de Desarrollo de Software y en la propuesta por la versión más actualizada de RUP en soporte digital.

Este trabajo propone un procedimiento que garantizará la correcta ejecución del proceso de pruebas durante el ciclo de vida del producto; se ejecutó sobre el módulo Registro de Áreas de Salud, y se prevee su posterior generalización. Su valor social se expresa en la contribución al mejoramiento de las condiciones de trabajo, desempeño y calidad de los servicios en el sector de la salud, al poder entregar un producto depurado de errores a la empresa Softel, con el fin de realizar pruebas finales al producto antes de ser liberado al cliente final.

PALABRAS CLAVE: Pruebas de software, proceso de prueba, flujo de trabajo de pruebas, plan de prueba.

Índice

Introducción

Con el desarrollo de la informática y el aumento de su impacto social, cada vez son más las instituciones u organizaciones que optan por incorporar aplicaciones que gestionen su información, logrando así una mayor dinámica en sus procesos de negocio. A propuesta de la dirección del país, desde el año 2000 comienza el proceso para informatizar el Sistema Nacional de Salud (SNS), tomando la atención primaria como eje fundamental.

El SNS en su constante renovación ha asumido el diseño y puesta en marcha de manera gradual de un Sistema de Gestión de la Información y el Conocimiento de nuevas dimensiones: el Sistema Integral de Salud.

La medicina como ciencia, la prestación de servicios de salud, la docencia, la formación de recursos humanos y la investigación están en un continuo cambio; cada vez más rápido. Un factor clave de este cambio se sustenta en las nuevas tecnologías de la información y las comunicaciones (TIC) a través del uso de las herramientas de comunicación y de la informática médica, potenciando la mayor inmediatez posible, la seguridad, la calidad y el control de las acciones del SNS.

Desde hace varias décadas la informática ha encontrado en la medicina una de sus aplicaciones más comunes e importantes que ha permitido al sector de la salud, contar con métodos novedosos, sencillos y eficaces de gestión administrativa en consultas, hospitales y centros de investigación y servicios; facilitando el trabajo de las estadísticas médicas y aumentando la confiabilidad de la misma y siendo de gran apoyo en la toma de decisiones. También ofrecen una gran ayuda en el campo de la investigación epidemiológica, farmacéutica, biológica, química, etc. Aspectos relacionados con la lucha para conseguir un mejor nivel de salud en las personas y la excelencia en los servicios.

Dentro de la informatización de la sociedad, en el sector de la salud, se propone una solución informática integral que dote al sistema de mayor grado de acceso a información unificada y confiable en tiempo real, y que aporte la rapidez y fiabilidad necesarias para las modernas técnicas de administración; lo que facilitará la toma de decisiones a los diferentes niveles. Esta información, además de llegar con la puntualidad necesaria a los niveles requeridos, sirve de referencia a otras áreas con similares problemas para la solución de sus necesidades y persigue el mejoramiento de la calidad de los servicios médicos a la población en el país.

El proceso de informatización del SNS tiene como objetivo acercar eficientemente y con calidad la prestación de los servicios de salud a la población, por lo que se pretende implementar un programa general de informatización del SNS, que apoye las

estrategias y políticas trazadas por la dirección del país y del Ministerio de Salud Pública (MINSAP); de manera que se logre la incorporación progresiva y sistemática de las nuevas tecnologías de la información y las comunicaciones en función de la adquisición y gestión del conocimiento y los servicios de salud.

El proceso de informatización modificará este sistema de manera que sus programas se adecuen a las necesidades y características de cada área de salud, territorio o municipio. Permitiendo la participación de todos los involucrados en su ejecución, según sus propios problemas de salud, así como la captación de información en los registros primarios. Estos inciden en el flujo, periodicidad, forma y contenido de los datos, procesos y presentación de la información adecuada y necesaria para la dirección que se encontrará disponible en la red para el acceso que se requiera con la actualidad y oportunidad que esto consigue. Lo que aportará beneficios intangibles que brindan las condiciones necesarias para un salto en la eficiencia de la atención y la optimización en los servicios que se ofrecen al pueblo.

La dirección del MINSAP y el Ministerio de Informática y las Comunicaciones (MIC) encomiendan la tarea de la informatización del SNS tomando a la Atención Primaria de Salud (APS) y al policlínico como eje fundamental; a la empresa dedicada a la ejecución de soluciones informáticas para la salud (Softel), en conjunto con la Universidad de las Ciencias Informáticas (UCI) y médicos especialistas en medicina general integral, en calidad de expertos funcionales. Estas instituciones tienen la misión en el marco del Proyecto Sistema de Información para la Salud (SISalud), de elaborar un producto de software que facilite la gestión de la información y la toma de decisiones en este nivel de atención.

La informatización de la APS facilitará la gestión de la información tanto información médica como administrativa que redundará en beneficio de todo el SNS, además de cada una de las áreas de salud o incluso para un departamento o consultorio del médico y enfermera de la familia.

Uno de los módulos que gestiona información del nivel primario de salud es el Registro de Áreas de Salud (RAS), el cual debe gestionar y administrar la información relacionada con las características de las unidades de salud que brindan atención a la población, como por ejemplo: dónde están ubicadas, qué áreas geográficas atiende, qué servicios brinda, cuántas especialidades ofrecen sus servicios.

El proceso de pruebas del proyecto de SISalud, se ha desarrollado de forma artesanal, ejecutándose en la etapa final de desarrollo antes de ser liberados los componentes un grupo de pruebas funcionales donde solamente se validan y verifican los requerimientos más significativos que fueron definidos, sin seguir un procedimiento que sirva de guía durante todo el ciclo de desarrollo.

Luego es necesario definir para el proyecto SISalud un flujo de trabajo para el proceso de pruebas con el fin de garantizar su correcta administración a través de planes de pruebas que definan las estrategias de pruebas en cada una de las fases del desarrollo de software, así como definir los recursos necesarios y especificar los casos de pruebas.

Este proceso no garantiza que el producto tenga toda la calidad requerida. Pues en el desarrollo del software intervienen personas que deben dar cumplimientos a varias actividades de producción, en las que las posibilidades de aparición del fallo humano son notables. Los errores pueden empezar a darse desde el primer momento del proceso, donde los requerimientos pueden estar especificados de forma errónea o imperfecta, así como en posteriores pasos de diseño y desarrollo.

Desde el mismo comienzo del proyecto se pueden empezar a planear las pruebas porque se tienen los siguientes artefactos: lista de ideas de pruebas, la cual permite escoger pruebas potencialmente útiles; estrategia de pruebas por la que se pueden dirigir los esfuerzos a uno o más aspectos del software. Además de ejecutar algunos casos de prueba al prototipo que se le presente al cliente. No solo es producir, sino aplicar buenas prácticas, de controlar para así detectar los errores a tiempo y poner en manos del cliente un producto depurado.[1]

El software pasa por diferentes etapas de vida, para así ir evolucionando y obtener un producto final, pero como la perfección humana no existe, una vez generado el código fuente, este debe ser probado para descubrir y corregir la mayor cantidad de errores posibles, antes de entregarlo al cliente final. Además, de eso depende el cumplimiento de los objetivos propuestos en tiempo, presupuesto y la satisfacción del cliente.

Por todo lo que hasta aquí se ha expuesto el esfuerzo estará encaminado en resolver el siguiente **problema**: ¿Cómo llevar a cabo el proceso de pruebas en el proyecto SISalud, aplicando buenas prácticas al módulo RAS?

Se propone como **objetivo general** del presente trabajo: ejecutar el proceso de pruebas al Registro de Áreas de Salud del proyecto SISalud siguiendo el flujo de trabajo de prueba propuesto. El **objeto de estudio** es: el flujo de pruebas del proceso de software. Delimitando como el **campo de acción**: el proceso de prueba en el RAS.

Para cumplir el objetivo del trabajo se proponen las siguientes **tareas de investigación**:

✓ Elaborar un marco teórico sobre el tema de pruebas analizando las mejores prácticas aplicadas en el flujo de trabajo de prueba y los niveles, tipos, estrategias y métodos de pruebas para proyectos de gestión.

[1] Dra. A. Febles, 2005 ¨Calidad del software y la empresa, enseñansa de un tema imprescindible para el ingeniero informático¨.

- ✓ Proponer un flujo de trabajo de prueba generalizable en el proyecto SISalud.
- ✓ Ejecutar en el módulo RAS el flujo de trabajo propuesto.
- ✓ Evaluar los resultados de las pruebas.
- ✓ Crear script y componentes de pruebas.

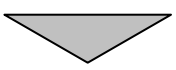
Este trabajo ha sido estructurado de la siguiente manera:

Capítulo 1: Se elabora un marco teórico sobre el tema de las pruebas, a partir de un análisis de la situación actual de este flujo de trabajo, en el proyecto SISalud. Se explica como se comporta el equipo de prueba a medida que transcurre el desarrollo del proceso de software, encontrará además, conceptos del flujo de trabajo de pruebas, en las cuales, se describen las actividades, trabajadores y artefactos que son generados. Además se describen los principios de las pruebas, el ciclo de vida de las pruebas, estrategias de las pruebas, los distintos tipos de niveles de pruebas y los métodos que se utilizan.

Capítulo 2: Describe la propuesta de flujo de trabajo de prueba para el grupo de desarrollo del proyecto SISalud, donde se muestran las actividades que se proponen para una iteración de pruebas, con los trabajadores, sus misiones y los artefactos que intervienen como entradas y salidas respectivamente.

Capítulo 3: Comprende la ejecución del Plan de Prueba del flujo propuesto. De una forma práctica, al aplicarlo en el proyecto SISalud, específicamente al RAS, en la primera iteración de la fase de construcción del primer ciclo de desarrollo del módulo. Se describirán los epígrafes más significativos del Plan de prueba. Se definirán los casos de prueba diseñados, scripts de datos de prueba y los procedimientos de prueba más significativos.

Capítulo 1: Fundamentación Teórica



Capítulo 1: Fundamentación Teórica

Introducción

El objetivo de este capítulo es abordar la elaboración de un flujo de trabajo de prueba con el fin de mejorar el proceso de pruebas, que en la actualidad lleva el proyecto SISalud, donde se desempeñan estudiantes y profesores de la Facultad 7 de la UCI, con la guía y asesoría técnica de la empresa Softel; dos entidades que persiguen un objetivo común: informatizar el SNS cubano, en especial la rama de la APS.

1.1 Situación actual de las pruebas

El proceso de pruebas, durante el ciclo de desarrollo del software es de baja calidad. Lo que incide en el análisis preliminar de nuevos requerimientos, y puede provocar la insatisfacción de los clientes por errores cometidos en el proceso. Se debe a la premura de entrega de los productos, lo que implica la omisión de pasos necesarios dentro del proceso de desarrollo.

Estas deficiencias consumen el escaso tiempo de los especialistas para el desarrollo de nuevos productos, evidenciándose que no está definido un Plan de pruebas, ni procedimientos que dicten los pasos a seguir en este flujo de trabajo, a partir de las técnicas que define la Ingeniería de Software para ello.

Con un grupo de entrevistas personales a estudiantes y profesores miembros de proyectos productivos de la UCI, así como la experiencia del propio proyecto SISalud se obtuvieron las ideas anteriormente planteadas.

La calidad se hace imprescindible para que los productos de la industria del software cubano avancen, ésta lleva implícito el respeto de los distintos flujos por los que debe atravesar un producto informático; como es el caso de las pruebas bien procedimentadas asegurando indiscutiblemente una correspondencia con los requerimientos acordados y disminuyendo el coste del tiempo dedicado para esta actividad.[2]

1.2 El ciclo de vida de prueba

El software es refinado a través de iteraciones en el ciclo de vida. El ciclo de vida de prueba se beneficia siguiendo un proceso iterativo equivalente. En cada iteración el equipo de desarrollo produce uno o más builds, cada builds es un candidato potencial para probar.[3]

Los objetivos del equipo de desarrollo difieren de una iteración a otra. El equipo de prueba estructura su prueba de acuerdo a los objetivos de la iteración.

[2] Ídem a la referencia 1
[3] UCI 2006, Flujo de trabajo de prueba en la fase de elaboración

En una iteración X, se puede implementar y ejecutar determinadas pruebas. Algunas de estas pruebas son conservadas y acumuladas, las cuales son usadas para pruebas de regresión. Cualquiera de las pruebas desarrolladas en la iteración X son candidatas para pruebas de X+1, cuando hay pruebas que son repetidas varias veces, vale la pena considerar automatizarlas. [Ver figura 1.1]

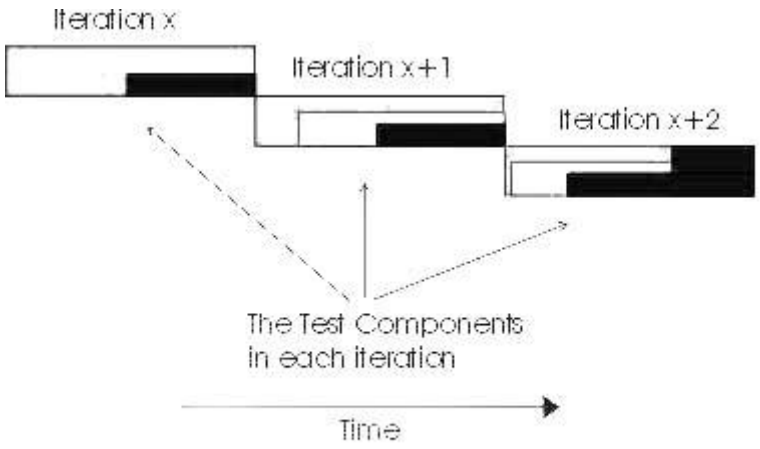


Figura 1.1 Ciclo de vida de prueba

1.3 Principios de las pruebas

Antes de aplicar los métodos para el diseño de casos de prueba es importante entender los principios que guían las pruebas del software. [4]

- ✓ A las pruebas se les debería dar seguimiento hasta los requisitos del cliente. Como se puede ver el objetivo de las pruebas del software es descubrir errores. Los defectos más graves son aquellos que impiden al programa cumplir sus requisitos.
- ✓ Las pruebas deberían planificarse mucho antes de su comienzo, cuando el modelo de requisitos y el modelo de diseño estén consolidado sin ningún tipo de problemas se podrá empezar a planificar las actividades de configuración de componentes de pruebas; antes de que se genere código alguno.
- ✓ El principio de Pareto es aplicable a la prueba del software. Este principio implica que al 80% de todos los errores encontrados se les puede dar seguimiento hasta un 20% de todos los módulos del programa, el problema sería separar los módulos sospechosos y probarlos concienzudamente.

[4] Pressman, R, S, (1997). "Ingeniería del software un enfoque práctico".

- ✓ Las pruebas deberían empezar por lo pequeño y de ahí progresar hacia lo grande. Las pruebas planeadas por lo general empiezan por los módulos individuales del programa y a medida que avanzan las pruebas se desplazan hacia grupos integrados de módulos hasta llegar finalmente al sistema completo.
- ✓ Las pruebas exhaustivas no son posibles, esto se debe a que el número de caminos o permutaciones en un programa es excepcionalmente grande, es imposible ejecutar todas las combinaciones posibles; pero cubrir adecuadamente la lógica del programa y asegurarse de que se han aplicado todas las condiciones en el diseño a nivel de componente sería una forma de resolver el problema.
- ✓ Para ser más eficaces, las pruebas deberían ser realizadas por un equipo independiente, el objetivo es encontrar errores y corregirlos, por tanto el ingeniero que creó el sistema no es el más indicado para hacer las pruebas de software.
- ✓ Una buena prueba, no puede ser redundante ya que el tiempo y los recursos son limitados. Las pruebas deben tener un propósito diferente.
- ✓ Una buena prueba, no debería ser ni demasiado sencilla ni demasiado compleja, aunque es posible a veces combinar una serie de pruebas en un sólo caso de prueba.
- ✓ Los ingenieros de pruebas deben prestarle más atención a las pruebas y crear casos de pruebas con una alta garantía de encontrar errores.

1.4 Las pruebas en el ciclo de vida del software

Desde la fase de inicio se puede comenzar a planificar y diseñar casos de prueba. Sin embargo, las pruebas fundamentalmente se desarrollan cuando una construcción es sometida a pruebas de integración y sistema.

Debido al esfuerzo iterativo en el desarrollo de software algunos de los casos de pruebas diseñados y ejecutados se pueden volver a ejecutar como consecuencia de las pruebas de regresión, por lo tanto en el desarrollo del ciclo de vida estas pruebas de regresión irán creciendo, siendo en la última iteración donde mayor será el esfuerzo para realizarlas.

Por tanto, es natural que el proceso de prueba se mantenga en todo el ciclo de vida del software, aunque el modelo de prueba cambia constantemente debido a:

- ✓ La eliminación de casos de pruebas obsoletos.
- ✓ El refinamiento de casos de pruebas en casos de pruebas de regresión.

- ✓ La creación de nuevos casos de uso para cada nueva construcción.

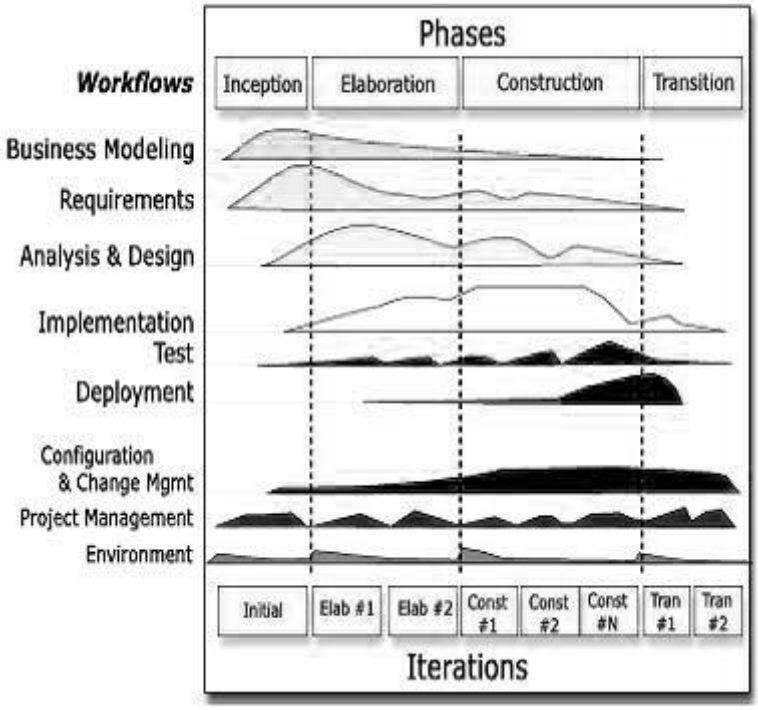


Figura 1.2 Rational Unified Process (RUP) en dos dimensiones

1.4.1 Fase de inicio

Los ingenieros de pruebas se van poniendo al corriente de la naturaleza general del sistema propuesto, van considerando que pruebas requerirá y van desarrollando algunos planes provisionales de prueba.

No se realiza en esta etapa un trabajo significativo de pruebas ya que el prototipo exploratorio de demostración tiene por lo general carácter ilustrativo más que operativo. El jefe de proyecto puede considerar útil el dedicar un pequeño esfuerzo a pruebas.

1.4.2 Fase de elaboración

El objetivo es asegurarse que los subsistemas de todos los niveles (subsistemas de servicio y subsistemas del diseño) y de todas las capas (desde la capa del sistema hasta las capas específicas de la aplicación) funcionen. Sólo se pueden probar los componentes ejecutables.

Al empezar por las capas más bajas de la arquitectura se prueban los mecanismos de distribución, almacenamiento, recuperación (persistencia) y concurrencias de objetos, así como otros mecanismos de las capas inferiores del sistema. Todas las capas no

son necesarias de probar sino es necesario probar cómo las capas superiores hacen uso de las inferiores.

Al planificar las pruebas el ingeniero de pruebas selecciona los objetivos a evaluar por la línea base de la arquitectura.

Al diseñar las pruebas los ingenieros de pruebas toman como base estos objetivos para identificar los casos de pruebas necesarios y prepararán procedimientos de pruebas para comprobar la sucesiva integración de subsistemas hasta completar la línea base.

1.4.3 Fase de construcción

En esta fase las pruebas son una actividad fundamental.

Al planificar las pruebas los ingenieros de pruebas seleccionarán los objetivos que comprueben las sucesivas construcciones, y por último el propio sistema.

Al diseñar las pruebas los ingenieros de pruebas determinarán cómo probar los requisitos en el conjunto de construcciones. Prepararán casos y procedimientos de pruebas con este fin.

Se realizan las pruebas de integración informando los resultados para tomar las medidas necesarias en casos de errores.

Se realizan también pruebas del sistema al alcanzarse el status de versión parcial del sistema, informando los resultados para tomar las medidas necesarias en casos de errores.

En esta fase se deben evaluar las pruebas a medida que transcurren las pruebas de integración y del sistema comprobando que éstas alcancen los objetivos del Plan de pruebas. Si una prueba no alcanza sus objetivos, los casos y procedimientos de pruebas deberán ser modificados para lograrlos.

1.4.4 Fase de transición

Se pueden realizar pruebas Beta (prueba realizadas en organizaciones representativas “clientes beta”) y/o pruebas Alfa (se realizan en la empresa que desarrolla el software y/o validaciones por terceros (una empresa especializada en pruebas realiza pruebas de aceptación por encargo del cliente). Se recopilan y analizan los resultados de estas pruebas con el objetivo de llevar a cabo acciones.

En esta fase se buscan pequeñas deficiencias que pasaron desapercibidas durante la fase de construcción y que pueden ser corregidas en el marco de la línea base de la arquitectura existente.

1.5 El Flujo de trabajo de RUP

El diagrama que se representa en la figura 1.3 es el flujo de trabajo predefinido para la disciplina de Prueba en el curso de una iteración típica en RUP, el cual es útil para mostrar su comportamiento dinámico.

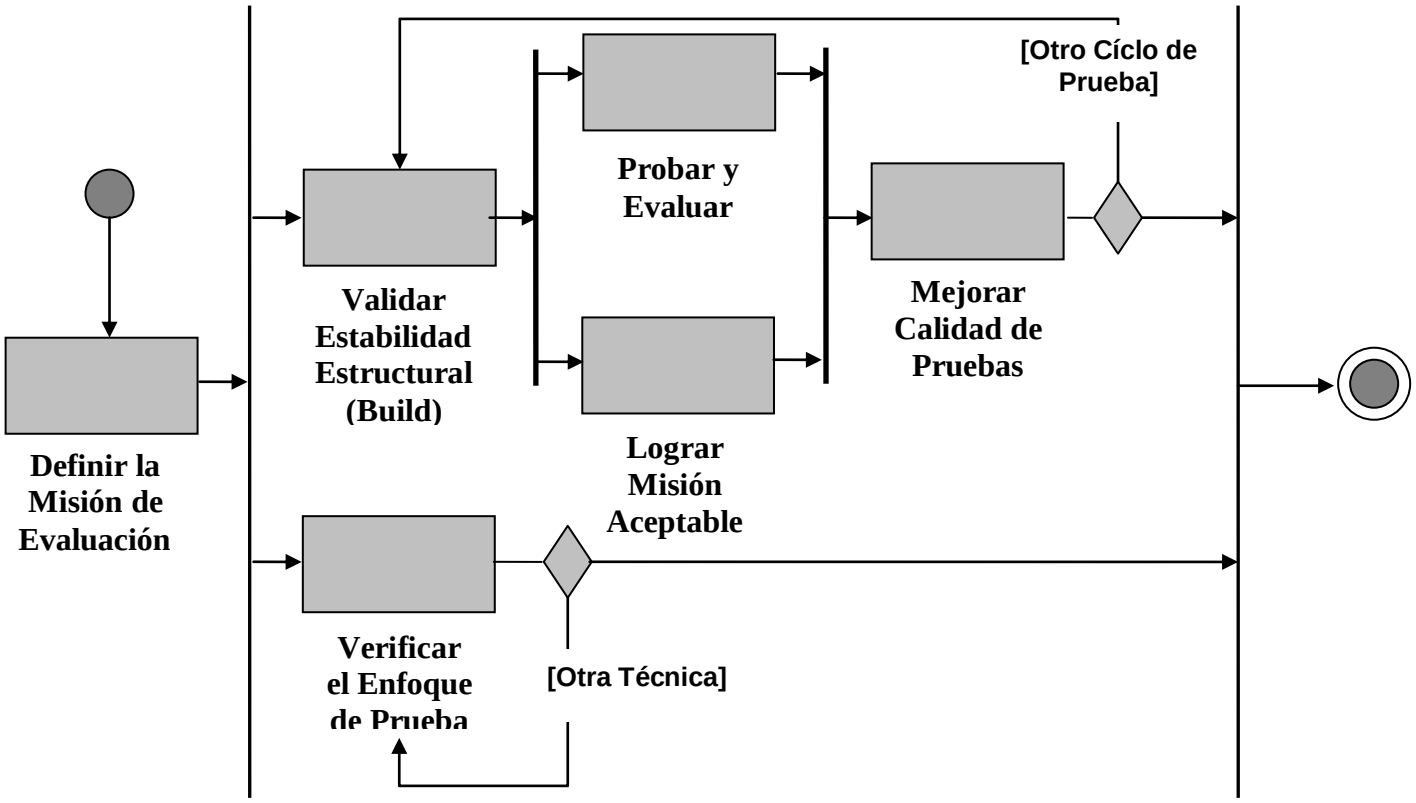


Figura 1.3 Ciclo de prueba de RUP

1.5.1 Actividades del Flujo de Trabajo

- ✓ Definir la misión de evaluación.
- ✓ Verificar el enfoque de prueba.
- ✓ Validar estabilidad estructural.
- ✓ Probar y evaluar.
- ✓ Lograr la misión aceptable.
- ✓ Mejorar la calidad de las pruebas.[5]

1.5.1.1 Definir la misión de evaluación

Esta actividad se centra en identificar el método apropiado de prueba para las iteraciones y estar de acuerdo con los stakeholder de las metas correspondientes que dirigirán las pruebas. En otras palabras, el propósito de la actividad definir misión de

[5] RUP C, 2003, Ayuda del Rational

evaluación es, planificar las pruebas, la cual se define al planificar los esfuerzos de prueba en una iteración llevando a cabo las tareas:

- ✓ Describir una estrategia de prueba, ejemplo cuántos flujos básicos y alternativos deben ser probados, cuántas pruebas se realizarán automatizadas y cuántas manuales
- ✓ Estimar los requisitos para el esfuerzo de la prueba, ejemplo recursos humanos, sistemas necesarios, etc.
- ✓ Planificar el esfuerzo de prueba.[6]

Por lo antes expuesto esta actividad se centra en definir el Plan de prueba, identificando los objetivos y estrategias de las pruebas. Se define además cómo monitorear y evaluar el progreso de las pruebas.

Para obtener el Plan de prueba, la estrategia de pruebas y los requisitos necesarios de pruebas de una iteración los ingenieros de prueba deben partir de los artefactos de entrada: requisitos adicionales, modelo de casos de uso, modelo de análisis, modelo de diseño, modelo de implementación, descripción de la arquitectura (vistas arquitectónicas de los modelos).

1.5.1.2 Verificar el enfoque de prueba

La actividad se centra fundamentalmente en verificar los enfoques de las pruebas, si estas no brindan resultados precisos y es apropiado para los recursos disponibles. Además su objetivo es lograr una solución de implementación apropiada para cada técnica o encontrar técnicas alternativas que puedan usarse. Esto ayuda a mitigar el riesgo de descubrir muy tarde técnicas que no son factibles aplicarlas en el proyecto.[7]

1.5.1.3 Validar estabilidad estructural (Build)

Su función fundamental es validar que la estructura (build) es suficientemente estable para la prueba detallada y para comenzar la evaluación. Es importante definir las descripciones de las pruebas, implementarlas, ejecutarlas, analizar los defectos y determinar los resultados de la prueba. De otro punto de vista, el objetivo de la actividad validar estabilidad estructural, es diseñar las pruebas pues esta actividad tiene como propósito:

- ✓ Identificar y construir los casos de pruebas para cada iteración.
- ✓ Identificar y ejecutar los procedimientos de pruebas especificando como ejecutar los casos de pruebas.[8]

[6] I Jacobson, G Booh, J Rumbaugh, 2002 ¨El proceso unificado de desarrollo de software¨
[7] Ídem a la referencia 5
[8] Ídem a la referencia 6

Como esta actividad se centra, entre otras cosas en diseñar casos de pruebas, por lo anteriormente expuesto, se le incluye a la actividad validar estabilidad estructural, los siguientes puntos:

- ✓ Deben ser diseñados los casos de prueba de integración para verificar que los componentes interaccionan entre sí de la forma apropiada después de ser integrados en una construcción.

  Estos casos de prueba pueden ser derivados de las realizaciones de los casos de uso de diseño (un diagrama de secuencia es parte de la realización de caso de uso diseño) o los diagramas de interacción de las realizaciones de casos de uso ya que estas realizaciones describen cómo interaccionan las clases y los objetos, y por tanto cómo interaccionan los componentes. Los diseñadores de pruebas buscan combinaciones de entradas, salida y estado inicial de sistema que den lugar a escenarios interesantes que empleen las clases (y por tanto los componentes) que participan en los diagramas.
- ✓ Deben ser diseñado casos de prueba de sistema, estas prueba principalmente combinaciones de casos de uso bajo condiciones diferente. Estas condiciones incluyen diferentes configuraciones de hardware, así como diferentes niveles de carga del sistema, diferentes tipos de actores y diferentes tamaños de bases de datos.[9]
- ✓ Deben ser diseñados casos de prueba de regresión. Algunos de los casos de pruebas de construcciones anteriores pueden ser usados para pruebas de regresión en construcciones subsiguientes, aunque no todos pueden ser utilizados para pruebas de regresión.

  Los casos de pruebas, para ser usados en pruebas de regresión, deben ser suficientemente flexibles para ser resistentes a cambios, esta flexibilidad debe ser cuidadosa ya que la conversión de un caso de prueba a caso de prueba de regresión supone un esfuerzo de desarrollo extra, luego se debe convertir casos de prueba a casos de prueba de regresión sólo cuando el esfuerzo merezca la pena.
- ✓ Se debe reutilizar procedimientos de prueba existentes tanto como sea posible, realizando las modificaciones adecuadas. Los diseñadores de prueba deben crear procedimientos de pruebas que puedan ser reutilizados en varios casos de prueba.[10]

[9] Ídem a la referencia 6
[10] Ídem a la referencia 6

1.5.1.4 Probar y evaluar

Realizar pruebas adecuadas a lo ancho y en profundidad para permitir una evaluación suficiente de los elementos que son objetivo de las pruebas. Esta evaluación suficiente es dirigida por la misión de evaluación actual y por los elementos motivadores de prueba.[11]

El propósito de la evolución de la prueba es el de evaluar los esfuerzos de la prueba en una iteración. Para evaluar los resultados de pruebas estos deben ser comparados con los objetivos esbozados en el Plan de pruebas. Los diseñadores de pruebas preparan métricas que les permiten determinar el nivel de calidad de software y qué cantidad de pruebas es necesario hacer.

El ingeniero de pruebas observa dos métricas:

- ✓ Compleción de la prueba: obtenida a partir de la cobertura de los casos de prueba y de la cobertura de los componentes probados. Esta métrica indica el porcentaje de casos de prueba que han sido ejecutados y el porcentaje de código que ha sido probado.
- ✓ Fiabilidad: la cual se basa en el análisis de las tendencias en los defectos detectados y en las tendencias en las pruebas que se ejecutan con el resultado esperado.[12]

1.5.1.5 Lograr la misión aceptable

Se centra fundamentalmente en entregar un resultado útil de la evaluación de las pruebas para los stakeholder, estos resultados están determinados en términos de la misión de evaluación. En la mayoría de los casos significarían enfocar el esfuerzo en ayudar al equipo de proyecto en lograr los objetivos del Plan de Iteración que se aplica al ciclo de prueba actual.[13]

1.5.1.6 Mejorar la calidad de las pruebas

Esta actividad se centra fundamentalmente en mantener y mejorar la calidad de las pruebas. Esto es especialmente importante si la intención es re-usar las ventajas desarrolladas en el actual ciclo de prueba en ciclos de pruebas posteriores.[14]

1.5.2 Trabajadores

- ✓ Administrador de prueba.
- ✓ Analista de prueba.
- ✓ Diseñador de prueba.

[11] Ídem a la referencia 5
[12] Ídem a la referencia 6
[13] Ídem a la referencia 5
[14] Ídem a la referencia 5

- ✓ Verificador o Probador.

1.5.2.1 Administrador de prueba

Tiene la responsabilidad global por el éxito del esfuerzo de la prueba. El rol involucra calidad de la prueba, recurso que planea y dirección, así como la solución de problemas que impiden el esfuerzo de la prueba. Es por ello que este debe ser seleccionado por sus aptitudes y capacidad para realizar esta tarea, ya que será la cabeza de la prueba.

Actividades o tareas que realiza:

- ✓ Trabajar acorde a la misión.
- ✓ Identificar los elementos que motiven o faciliten las pruebas.
- ✓ Comprometerse con las pruebas.
- ✓ Evaluar y auditar la calidad.
- ✓ Evaluar y mejorar las pruebas.

Es responsable de:

- ✓ Plan de pruebas.
- ✓ Resumen de los resultados de las pruebas.
- ✓ Negociar o acordar las pruebas.
- ✓ Asegurarse de una correcta planificación y administración de recursos.
- ✓ Verificar el progreso y efectividad de las pruebas.
- ✓ Evaluar los resultados de cada ciclo de prueba.

Modifica:

- ✓ Lista de defectos.
- ✓ Solicitudes de cambio.[15]

1.5.2.2 Analista de prueba

Es responsable de identificar y definir las pruebas requeridas, supervisando el progreso de las pruebas, así como el resultado por cada ciclo de las pruebas y evalúa la calidad global experimentada, como resultado de la actividad de prueba. El rol de manera general tiene la responsabilidad de representar las necesidades de los clientes que no se relacionan directamente, o no tienen un conocimiento profundo acerca de la representación del proyecto. Este rol es considerado una especialización del administrador de prueba.

Asociamos a este rol, al rol de diseñador de prueba ya que planean las pruebas lo que significa que deciden los objetivos de las pruebas apropiados y planifican las pruebas.[16]

[15] Ídem a la referencia 5

Las tareas o actividades que debe realizar son:

- ✓ Identificar los objetivos de la prueba.
- ✓ Identificar las ideas de la prueba.
- ✓ Definir los detalles de las pruebas.
- ✓ Definir necesidades de evaluación y trazabilidad.
- ✓ Determinar los resultados de las pruebas.
- ✓ Verificar los cambios en la construcción.

Es responsable de:

- ✓ Lista de ideas de las pruebas.
- ✓ Modelo de prueba.
- ✓ Casos de prueba.
- ✓ Descripciones de los casos de pruebas.
- ✓ Procedimientos de pruebas.
- ✓ Colección y administración de datos de pruebas.
- ✓ Colección de los resultados de las pruebas.
- ✓ Evaluar los resultados de cada ciclo de prueba.

Modifica:

- ✓ Plan de pruebas.
- ✓ Resumen de los resultados de las pruebas.
- ✓ Requisitos de cambio.
- ✓ Pautas para las pruebas.[17]

1.5.2.3 Diseñador de prueba

Es responsable de definir el acercamiento de la prueba para así asegurar su aplicación exitosa. El rol involucra identificar las técnicas apropiadas, herramientas y pautas para llevar a cabo las pruebas requeridas, y dar guía en los requisitos de los recursos correspondientes para el esfuerzo de la pruebas.[18]

Este rol es llamado por arquitecto de prueba o arquitecto de automatización de prueba. Por lo que le asociamos a este rol, el rol de ingeniero de componente ya que son responsables de automatizar algunos de los procedimientos de prueba.

Actividades o tareas que realiza:

- ✓ Definir enfoque de las pruebas.
- ✓ Definir la configuración del ambiente de las pruebas.
- ✓ Identificar los mecanismos de pruebas.

---

[16] Ídem a la referencia 6
[17] Ídem a la referencia 5
[18] Ídem a la referencia 5

- ✓ Estructurar la implementación de las pruebas.
- ✓ Definir los elementos de las pruebas.
- ✓ Desarrollar las pautas de las pruebas.

Es responsable de:

- ✓ Documento de prueba de arquitectura.
- ✓ Componente de prueba.
- ✓ Especificación de interfaz de prueba.
- ✓ Configuración del ambiente de prueba.
- ✓ Definir la suite de pruebas.
- ✓ Identificar y describir apropiadas técnicas de prueba.
- ✓ Identificar herramientas apropiadas de soporte.
- ✓ Definir y mantener la arquitectura de automatización de pruebas.
- ✓ Especificar y verificar las condiciones de las configuraciones de prueba.

Modifica:

- ✓ Plan de pruebas.
- ✓ Script de pruebas.[19]

1.5.2.4 Verificador o Probador

Este rol es responsable de las principales actividades y el mayor esfuerzo de las pruebas. Ejecuta o produce las principales pruebas del software y es quien obtiene directamente los resultados.

Se asocia al rol ingeniero de prueba de sistema ya que un ingeniero de prueba de sistema es responsable de realizar las pruebas de sistema necesarias sobre una construcción que muestra el resultado ejecutable de una iteración completa. Este se encarga de documentar los defectos como resultados de la ejecución de las pruebas de sistema.[20]

Se le asocia el rol de ingeniero de prueba de integración ya que es responsable de ejecutar todas las pruebas de integración planificadas para cada construcción de la iteración realizada en el flujo de implementación, además se encarga de documentar los defectos descubiertos en la ejecución de las pruebas de integración.

Debe realizar las siguientes actividades o tareas:

- ✓ Implementar pruebas.
- ✓ Implementar la suite de pruebas.
- ✓ Ejecutar la suite de pruebas.

[19] Ídem a la referencia 6
[20] Ídem a la referencia 6

- ✓ Analizar fallos de prueba.

Es responsable de:

- ✓ Test log (salida de ejecución de las pruebas).
- ✓ Test Script (Pasos o instrucciones para realizar la prueba).
- ✓ Identificar la implementación más apropiada para la prueba.
- ✓ Implementar pruebas individuales.
- ✓ Defecto.
- ✓ Verificar la forma en que se ejecutan las pruebas.
- ✓ Analizar y recuperarse de los errores de ejecución.

Modifica:

Requisitos de cambio confeccionado por el jefe de control de cambios.

1.5.3 Artefactos

- ✓ Modelo de prueba.
- ✓ Caso de prueba.
- ✓ Procedimiento de prueba.
- ✓ Componente de prueba.
- ✓ Plan de prueba.
- ✓ Defecto.
- ✓ Evaluación de las Pruebas.

1.5.3.1 Modelo de prueba

Describe principalmente como se pueden probar los componentes ejecutables, en el nivel de integración y sistema. También puede describir como probar aspectos específicos del sistema como por ejemplo si la interfaz de usuario es utilizable o el manual de usuario cumple su cometido. El modelo de pruebas es una colección de casos de pruebas.

1.5.3.2 Caso de prueba

El caso de prueba específica la forma de probar un sistema incluyendo las entradas, salidas y resultados esperados, así como bajo que condiciones debe probarse el sistema. Un caso pude derivarse de las descripciones de los casos de uso del diseño.[21]

1.5.3.3 Procedimiento de prueba

Un procedimiento de prueba especifica como realizar uno o varios casos de pruebas, así como puede servir de guía para un individuo de cómo realizar un caso de prueba

[21] Ídem a la referencia 6

manualmente. Además puede ser una especificación de cómo interactuar con una herramienta para ejecutar una prueba.

1.5.3.4 Componente de prueba

Un componente de prueba automatiza uno o varios procedimientos de prueba, proporcionando entradas de pruebas, controlando y monitoreando los componentes a probar y de ser posible informa el resultado de las pruebas.[22]

1.5.3.5 Plan de prueba

El Plan de prueba describe las estrategias, recursos y planificación de las pruebas. La estrategia de prueba incluye la definición del tipo de prueba a ejecutar en cada iteración y sus objetivos.[23]

1.5.3.6 Defecto

Un defecto es una anomalía del sistema, descubierto en una revisión o un fallo del software.[24]

1.5.3.7 Evaluación de las pruebas

Es una evaluación de los resultados de los esfuerzos de prueba.

1.6 Niveles de Prueba

Existen diferentes niveles de pruebas, cada una de ellas se realiza en determinados momentos del ciclo de vida del software, la siguiente tabla resume las mismas:

- ✓ Unidad.
- ✓ Integración.
- ✓ Sistema.
- ✓ Aceptación.

Este modelo describe a un nivel alto de abstracción las fases del ciclo de desarrollo en las que se involucran las pruebas y los niveles de las mismas, la figura 1.4 muestra este modelo.[25]

[22] Ídem a la referencia 6
[23] Ídem a la referencia 6
[24] Humphrey 2001,¨ Introducción al proceso personal de software ¨, Madrid
[25] MSc.Y, Fernández, Lic. L Cruz, Ing M González, Ing D, Acosta, 2006, ¨Proceso de pruebas de aceptación para un software de gestión ¨

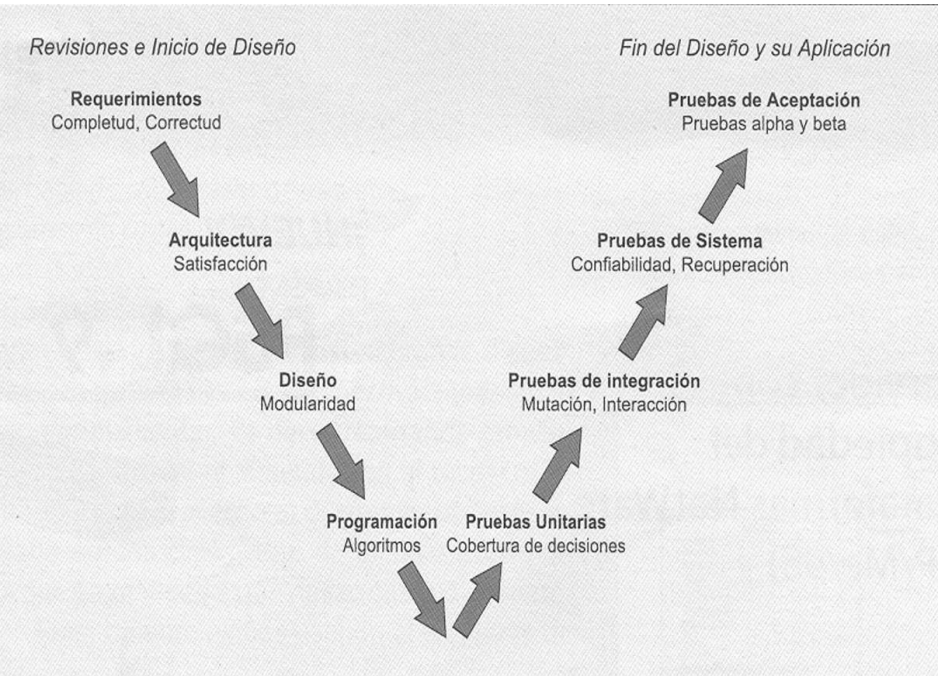


Figura 1.4 Niveles de prueba con las fases de desarrollo de software

1.6.1 Nivel de Unidad

En este nivel de prueba fundamentalmente se ejecutan casos de pruebas de caja blanca diseñados para:

- ✓ Probar las estructuras de datos locales para asegurar que los datos que se mantienen temporalmente, conservan su integridad durante todos los pasos de ejecución del algoritmo.
- ✓ Probar las condiciones límites para asegurar que el módulo funciona correctamente en los límites establecidos como restricciones de procesamiento.
- ✓ Ejercitar todos los caminos independientes (caminos básicos) de la estructura de control con el fin de asegurar que todas las sentencias del módulo se ejecutan por lo menos una vez. Y, finalmente, se prueban todos los caminos de manejo de errores.[26]

Los casos de pruebas diseñados para ejecutar a nivel de unidad deben descubrir errores como:

- ✓ Comparaciones entre tipos de datos distintos.
- ✓ Operadores lógicos o de procedencia incorrectos.
- ✓ Igualdad esperada cuando los errores de precisión la hacen poco probable.
- ✓ Variables o comparaciones incorrectas.

[26] Ídem a la referencia 25

- ✓ Terminación de bucles inapropiada o inexistente.
- ✓ Fallo de salida cuando se encuentra una iteración divergente.
- ✓ Variables de bucles modificadas de forma inapropiada.

La prueba de límites es probablemente la más importante, es una tarea del paso de la prueba de unidad. El software falla con frecuencia en sus condiciones límites. Las pruebas que ejercitan las estructuras de datos, el flujo de control y los valores de los datos por debajo y por encima de los máximos y los mínimos son muy apropiadas para descubrir estos errores.

1.6.2 Nivel de Integración

La prueba de integración es una técnica sistemática para construir la estructura del programa mientras que, al mismo tiempo, se llevan a cabo pruebas para detectar errores asociados con la interacción.

El objetivo es coger los módulos probados mediante la prueba de unidad y construir una estructura de programa que esté de acuerdo con lo que dicta el diseño. En el proceso de integración puede derivarse la realización de los casos de uso del sistema, estos describen como interactúan las clases y los objetos y por lo tanto la interacción de los componentes.

Los encargados de diseñar los casos de pruebas de integración consideran como entrada a los diagramas de secuencia de las realizaciones de los casos de uso, pues ahí buscan las combinaciones de entradas, salidas y estado inicial del sistema que dan lugar a escenarios que usan las clases y por lo tanto los componentes que participan en los diagramas. Un caso de prueba se deriva de un diagrama de secuencia, este describe una o varias secuencias del negocio.

Después se procede a ejecutar el caso de prueba de integración diseñado, creando trazas de ejecución o ejecutándolo paso a paso, a continuación se compara las interacciones actuales con los diagramas de secuencia y de no ser igual se trata de un defecto que puede dar al traste con un error.[27]

1.6.3 Nivel de Sistema

Las pruebas de sistemas caen fuera del ámbito del proceso de software y no las realiza únicamente el desarrollador del software. Sin embargo los pasos durante el diseño del software y durante la prueba pueden mejorar enormemente la posibilidad de éxito de las pruebas de sistema.

Las pruebas de sistema principalmente se centran en verificar la interacción de los actores con el sistema, por lo que a menudo los casos de pruebas se obtienen a partir

[27] Ídem a la referencia 4

de las descripciones de los casos de uso. Aunque también se le aplican pruebas al sistema como un todo.

1.6.4 Nivel de Aceptación

La mayoría de los desarrolladores de productos de software llevan a cabo un proceso denominado pruebas Alfa o Beta para descubrir errores considerando que solo el usuario final puede descubrir.

La prueba Alfa es llevada a cabo por el cliente en el lugar de desarrollo. Se usa el software de forma natural con el desarrollador de observador registrando los errores y los problemas de uso. Estas pruebas se llevan a acabo en un entorno controlado.

Las pruebas Betas se llevan a acabo por los usuarios finales en sus puestos de trabajo, y a diferencia de la prueba Alfa el desarrollador no está presente, así que esta prueba no puede ser controlada por el desarrollador, por lo que el cliente debe registrar todas las inconformidades y tramitárselas al desarrollador.[28]

1.7 Tipos de prueba

1.7.1 Pruebas de Regresión

Cuando se procede a añadir un nuevo módulo a un software como parte de la integración, el software cambia, pues se establecen nuevos caminos lógicos de flujos de datos, por lo que esto puede traer consigo errores en funciones que antes funcionaban correctamente. La prueba de regresión es volver a ejecutar un subconjunto de casos de pruebas que fueron ejecutados anteriormente con el objetivo de encontrar estos errores o asegurarse que la integración del nuevo módulo no introdujo defectos no deseados en el software.

En un contexto más amplio, las pruebas de cualquier tipo dan como resultado el descubrimiento de cualquier error, por lo que este hay que corregirlo, esto de algún modo cambia la configuración del software (el programa, su documentación, o los datos que soportan) por lo que es necesario aplicar pruebas de regresión pues son una actividad que ayudan a asegurar que los cambios debidos a las pruebas u otros motivos no introducen un comportamiento no deseado.

El conjunto de pruebas de regresión (el subconjunto de pruebas a realizar) contiene tres clases diferentes de prueba:

- ✓ Una muestra representativa de pruebas que ejercite todas las funciones del software.
- ✓ Pruebas adicionales que se centran en las funciones del software que se van a ver probablemente afectadas por el cambio.

[28] Ídem a la referencia 4

- ✓ Pruebas que se centran en los componentes del software que ha cambiado.

1.7.2 Criterios de la prueba de validación

La validación del software se consigue mediante una serie de pruebas de caja negra que demuestran la conformidad con los requisitos. Un Plan de prueba traza la clase de pruebas que se han de llevar a cabo, y un procedimiento de prueba define los casos de prueba específicos en un intento por descubrir errores de acuerdo con los requisitos.

Tanto el plan como el procedimiento estarán diseñados para asegurar que se satisfacen todos los requisitos funcionales, que se alcanzan todos los requisitos de rendimiento, que la documentación es correcta e inteligible y que se alcanzan otros requisitos (por ejemplo, portabilidad, compatibilidad, recuperación de errores, facilidad de mantenimiento).

Después que se procede a la prueba de validación, puede darse con una de las dos condiciones siguientes:

- ✓ Las características de funcionamiento o de rendimiento están de acuerdo con las especificaciones y son aceptables.
- ✓ Se descubre una desviación de las especificaciones y se crea una lista de deficiencias.

Las desviaciones o errores descubiertos en esta fase del proyecto raramente se pueden corregir antes de la terminación planificada.[29]

1.7.3 Revisión de la configuración

En el proceso de validación de un elemento, es importante es la revisión de la configuración, esto está dirigido con la intención de revisar y asegurarse que todos los elementos de configuración del software se han desarrollado adecuadamente, se han catalogado y están suficientemente detallados para soportar la fase de mantenimiento durante el ciclo de vida del software. La revisión de la configuración a veces se denomina auditoría.[30]

1.7.4 Prueba de recuperación

La prueba de recuperación es una prueba de sistema que forza al software de muchas formas y verifica que se llevó a cabo apropiadamente. Si la recuperación es automática (llevada a cabo por el propio sistema) hay que evaluar la corrección de la inicialización, de los mecanismos de recuperación del estado del sistema, de la recuperación de datos y del proceso de rearranque. Si la recuperación requiere la

[29] Ídem a la referencia 4
[30] Ídem a la referencia 4

intervención humana, hay que evaluar los tiempos medios de reparación para determinar si están dentro de unos límites aceptables.

1.7.5 Prueba de Seguridad

Todo sistema basado en computadora que maneja información sensible y se quiera proteger del mundo exterior se requiere de un sistema de seguridad guiado por computadora que permitirá que la información no sea robada o saboteada por piratas informáticos ya sea por deporte o por llevar a cabo una destrucción de la información.

Por lo que la prueba de seguridad intenta verificar que los mecanismos de seguridad del software son válidos y confiables luego esto le brinda al software garantía y robustez.

Por supuesto, la seguridad del sistema debe ser probada en su invulnerabilidad frente a un ataque frontal, pero también debe probarse en su invulnerabilidad a ataques por los flancos o por la retaguardia.

En la ejecución de esta prueba el responsable de pruebas de software desempeña el papel de un individuo que desea entrar al sistema sin importar, su móvil de entrada, para ello deberá tratar de conseguir claves de acceso por cualquier medio. Con tiempo y recursos suficientes, una buena prueba de seguridad terminará por acceder al sistema. El papel del diseñador del sistema es hacer que el coste de la entrada ilegal sea mayor que el valor de la información.

1.7.6 Prueba de resistencia (Stress)

Es un tipo de prueba de fiablidad que se enfoca en evaluar como el sistema responde bajo condiciones anormales. Stress en el sistema pudiera incluir sobrecarga extrema, insuficiente memoria, servicios y hardware no disponible, o recursos compartidos limitados. Estas pruebas son usualmente realizadas para ganar u obtener una mejor comprensión de cómo y que áreas del sistema se pudieran dañar, de esta forma saber qué plan de contingencia y actualización en el mantenimiento puede ser planificado y qué presupuesto para bien se puede planificar de antemano.

1.8 Estrategia de Pruebas de Software

La estrategia de prueba, integra las técnicas de diseño de casos de pruebas a través de pasos planificados y elaborados, con el fin de construir de forma correcta un software. La estrategia proporciona una línea a seguir, describe como hacer las pruebas, cuando se deben planificar y ejecutar, así como el tiempo y el esfuerzo que se va a requerir en el desarrollo de las mismas. La estrategia de prueba que se quiera usar debe incorporar: La planificación de las pruebas, el diseño de casos de pruebas, la ejecución de las pruebas y evaluación de los datos resultados.

Una estrategia de pruebas de software debe incluir pruebas de bajo nivel que ejecuten todos los pequeños segmentos de código fuente para probar si se han implementado correctamente, a través de la aplicación de técnicas de diseño de métodos de casos de prueba de caja blanca. Así como pruebas de alto nivel que validen las principales funcionalidades del sistema frente a los requisitos del cliente, a través de la aplicación de técnicas de diseño de métodos de casos de prueba de caja negra.

Una estrategia debe proporcionar una guía al profesional y un conjunto de hitos para el jefe de proyecto. De esa manera facilita el proceso de administración del proyecto en particular del proceso de prueba.

Debido a que los pasos de la estrategia de pruebas se deben dar a la vez cuando aumenta la presión de los plazos fijados, se debe medir el progreso y los problemas que deben aparecer lo antes posible. [31]

1.9 Los defectos de software

El término defecto se refiere a algo que está equivocado en un programa, tal como un error sintáctico, una falta tipográfica, un error de puntuación o una sentencia incorrecta del programa. Los defectos pueden estar en los programas, en los diseños o incluso en los requisitos, las especificaciones o en otra documentación. Los defectos pueden ser sentencias extra o redundantes, sentencias incorrectas o secciones del programa omitidas. Un defecto, es cualquier cosa que reduce la capacidad de los programas para cumplir completa y efectivamente las necesidades de los usuarios. Un defecto es una cosa objetiva. Es algo que se puede identificar, describir y contabilizar.

Los errores triviales de implementación son cosas incorrectas que cometen las personas, y sin tener en cuenta cuándo y quién los comete, pueden causar serios problemas en el sistema.

Los defectos deberían ser importante para cada ingeniero de software no sólo porque afectan a los usuarios, sino también porque más de la mitad del esfuerzo de las organizaciones de software está dedicado a encontrar y corregir los defectos. Como el tiempo de prueba es difícil de predecir, los defectos son la causa principal de los problemas de costes y programaciones.[32]

1.9.1 Tipos de defectos

Para analizar los defectos, es útil dividirlos en categorías.[33]

Defecto	Descripción
Documentación	Comentarios, mensajes

[31] Ídem a la referencia 4
[32] Ídem a la referencia 24
[33] Ídem a la referencia 24

Sintaxis	Ortografía, erratas, puntuación, formato de las instrucciones
Construir paquetes	Gestión del cambio, librerías, control de versión
Asignación	Declaración, nombres duplicados, ámbito, límites
Interfaz	Llamadas a procedimientos y referencias, E/S, formatos de usuario
Chequeo	Mensajes de error, chequeos inadecuados
Datos	Estructura, contenido
Función	Lógica, punteros, bucles, recursión, computación, defectos de la función
Sistema	Configuración, temporización, memoria
Entorno	Diseño, compilación, pruebas y otros problemas que soporta el sistema

Tabla 1.1 Tabla de defectos estándares

1.9.2 Pasos para encontrar defectos

Aunque no hay forma de acabar con la introducción de defectos, es posible encontrar y eliminar casi todos los defectos al principio.

Eliminar los defectos al principio ahorrará tiempo y esfuerzo y se realizarán mejores productos; si se pudiera encontrar un defecto de diseño antes de hacer el código no se gastaría tiempo implementando un diseño incorrecto, de forma similar cuando se encuentra y corrige defectos de codificación antes de compilar se ahorraría tiempo de encontrar esos defectos en el flujo de prueba.

Hay varios métodos de encontrar defectos en un programa. En esencia, todos estos métodos implican los siguientes pasos:

- ✓ Identificar los síntomas del defecto.
- ✓ Deducir de estos síntomas la localización del defecto.
- ✓ Entender lo que es erróneo en el programa.
- ✓ Decidir cómo corregir el defecto.
- ✓ Hacer la corrección.
- ✓ Verificar que el arreglo ha resuelto el problema.

Existen varias herramientas de detección de errores una de ellas es el compilador, aunque estos no detectarán todos los errores tipográficos, ni de puntuación u otro defecto tipográfico.

Una segunda forma de encontrar defectos son las pruebas, aunque existen muchas clases de métodos de pruebas todas requieren de los examinadores que proporcionan datos de pruebas y condiciones de pruebas; muchas veces llamado casos de pruebas o escenarios de pruebas.

Una última forma para la detección y corrección de defectos sería revisar personalmente el código fuente del programa.[34]

1.9.3 ¿Por qué revisar el código?

Revisar el código es una forma rápida y sencilla para la detección y corrección de defectos. Pues se debe estudiar el código fuente para encontrar los errores. Esto es mejor hacerlo después de escrito el código fuente y antes de probarlo o compilarlo. Muchos defectos son simples descuidos fáciles de encontrar después de hacer el diseño o el código. Aunque la revisión del código consume mucho tiempo es mejor que hacer las pruebas de unidad.

La causa de que la revisión del código sea tan eficiente es porque cuando revisas ves los problemas y no los síntomas. Es decir cuando revisas el código piensas en lo que el programa debe hacer y cuando ves un defecto ves el problema. Este método tiene la desventaja, que consume mucho tiempo en las revisiones y son difícil de hacer.[35]

1.10 Una estrategia de Prueba

Se puede ver el proceso de la ingeniería del software como una espiral. [Ver figura 1.5]. En el inicio la ingeniería define los papeles del software que conduce al análisis de los requerimientos de este, se establece el dominio de información así como la función, comportamiento, rendimiento, restricciones y los criterios de validación del software.[36]

Al moverse hacia el interior de la espiral se llega al diseño y por último a la codificación, para desarrollar un software se da vueltas en la espiral a través de los flujos, transitando por diferentes niveles de pruebas Unidad, Integración, Sistema, Implantación y Aceptación.

El nivel de Unidad comienza en el vértice de la espiral y se centra en cada unidad del software según como esté implementada en el código fuente. En este nivel se prueba el comportamiento de cada uno de los componentes de forma independiente, estas

[34] Ídem a la referencia 24
[35] Ídem a la referencia 24
[36] Ídem a la referencia 4

pruebas se ejecutan una vez realizada la implementación en el componente. Se prueba la funcionalidad de una clase o conjunto de clases correlacionadas, es decir aquí se ejecutan métodos de casos de pruebas de caja blanca.

La prueba avanza moviéndose hacia fuera de la espiral y llega al nivel de Integración. Este nivel de prueba se centra en la ejecución de métodos de caja negra aunque se pueden llevar a cabo métodos de pruebas de caja blanca para asegurar que se cumplen los principios de caminos de control.

Además de la construcción de la arquitectura del software, su objetivo es probar la integración de los módulos del sistema, una vez que a estos se les hayan aplicado casos de pruebas de unidad, se deberá verificar que estos interactúan correctamente a través de las funcionalidades expuestas en sus interfaces, se verifican las interfaces entre las partes de una arquitectura. Este tipo de prueba deberá realizarse durante la fase de construcción, una vez sea implementado el componente.

El proceso continúa dando otra vuelta hacia fuera y se encuentra el nivel de Sistema el cual se centra en elementos de sistema (por ejemplo hardware, personas, bases de datos), en comprobar la funcionalidad del sistema como un todo, así como verificar fundamentalmente los requisitos no funcionales definidos para el producto, por ejemplo, que el sistema puede gestionar los volúmenes de información requeridos, se ajusta a los tiempos de respuesta deseados y que los procedimientos de respaldo, seguridad e interfaces con otros sistemas funcionan correctamente. Se debe verificar también el comportamiento del sistema bajo las condiciones más extremas.

Al avanzar una vuelta más hacia fuera se encuentra el nivel de Aceptación, el cual se centra en verificar formalmente con el cliente que el sistema satisface todas sus necesidades.

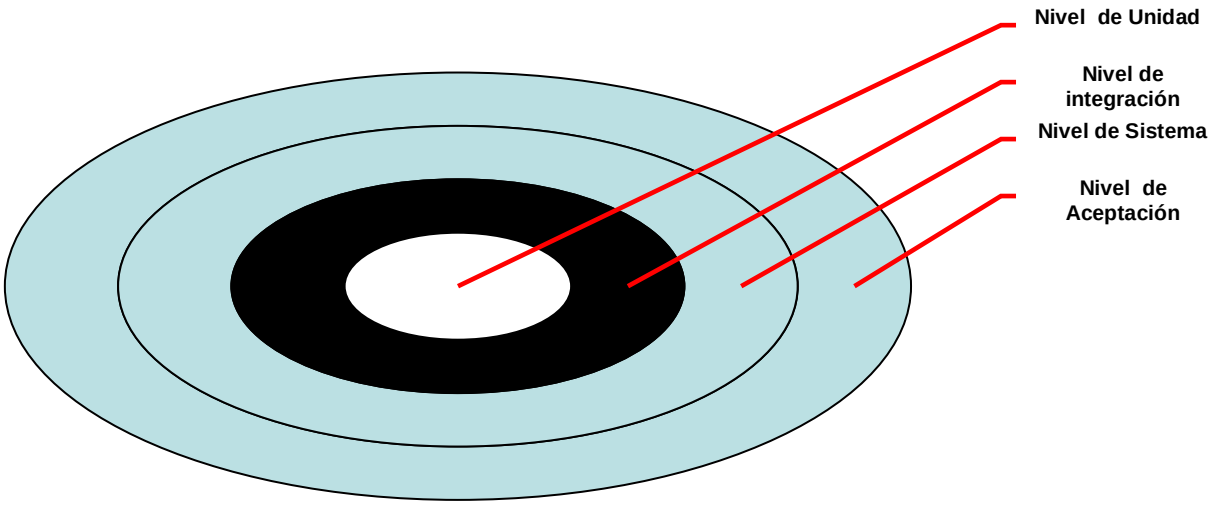


Figura 1.5 Niveles de pruebas

1.10.1 Aspectos Estratégicos para las Pruebas

Pressman plantea que se deben abordar los siguientes puntos si se desea implementar con éxito una estrategia de prueba del software: [37]

- ✓ Especificar los requisitos del producto de manera confiable mucho antes de que comiencen las pruebas. Aunque el objeto principal de una buena estrategia también evalúa otras características de la calidad tales como la portabilidad, facilidad de mantenimiento y facilidad de uso.
- ✓ Establecer los objetivos de la prueba de manera explicita: Se deberían establecer en términos medibles los objetivos específicos de la prueba. Por ejemplo, la efectividad de la prueba, la cobertura de la prueba, tiempo medio de fallo, el costo para encontrar y arreglar errores, densidad de fallos remanente o frecuencia de ocurrencia, y horas de trabajo por prueba de regresión que deberían establecerse dentro de la planificación de la prueba
- ✓ Comprender que usuarios van a manejar el software y desarrollar un perfil para cada categoría de usuario. Usar casos de prueba que describan el escenario de Interacción para cada clase de usuario pudiendo reducir el esfuerzo general de prueba concentrando la prueba en el empleo real del producto.
- ✓ Desarrollar un Plan de prueba que haga hincapié en las pruebas de ciclo rápido recomienda que un equipo de ingeniería del software aprenda a probar en ciclos rápidos (2 por 100 del esfuerzo del proyecto) de incrementos de funcionalidad y/o mejora de la calidad útil para el cliente, y que se puedan

[37] Ídem a la referencia 4

probar sobre el terreno. La realimentación generada por estas pruebas de ciclo rápido puede usarse para controlar los niveles de calidad y las correspondientes estrategias de prueba.

- ✓ Construir un software Robusto. El software deberá diseñarse de manera que use técnicas de depuración es decir, el software debería ser capaz de diagnosticar ciertas clases de errores. Además, el diseño debería incluir pruebas automatizadas y pruebas de regresión.
- ✓ Usar Revisiones Técnicas Formales Efectivas. Las revisiones técnicas formales pueden ser tan efectivas como las pruebas en el descubrimiento de errores. Por este motivo, las revisiones pueden reducir la cantidad de esfuerzo de prueba necesaria para producir software de alta calidad. Llevar a cabo revisiones técnicas formales para evaluar la estrategia de prueba y los propios casos de prueba. Las revisiones técnicas formales pueden descubrir inconsistencias, omisiones y errores claros en el enfoque de la prueba. Esto ahorra tiempo y también mejora la calidad del producto.
- ✓ Desarrollar un enfoque de mejora continua al proceso de prueba. Debería medirse la estrategia de prueba. Las métricas agrupadas durante la prueba deberían usarse como parte de un enfoque estadístico de control del proceso para la prueba del software.

1.11 Métodos de Pruebas

Existen diversos métodos para realizar las pruebas de software, entre las más importantes se encuentran la prueba de caja blanca y prueba de caja negra.

La prueba de caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. O sea, los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene.

La prueba de la caja blanca del software comprueba los caminos lógicos del software, proponiendo casos de prueba que se ejerciten conjuntos específicos de condiciones y/o bucles. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coinciden con el esperado o mencionado.[38]

[38] Ídem a la referencia 3

1.11.1 Método de Caja Blanca

La prueba de caja blanca es un método de diseño de casos de pruebas, a través de las estructuras de control de diseño procedimental la cual se ilustra en la Figura 1.6. Mediante los métodos de pruebas de caja blanca el ingeniero de prueba puede obtener casos de pruebas que:

- ✓ Se ejerciten por lo menos una vez todos los caminos independientes de cada módulo.
- ✓ Se ejerciten todas las decisiones lógicas verdaderas o falsas.
- ✓ Se ejecuten los bucles en sus límites y con sus límites operacionales.
- ✓ Se ejerciten las estructuras internas de datos para asegurar su validez.

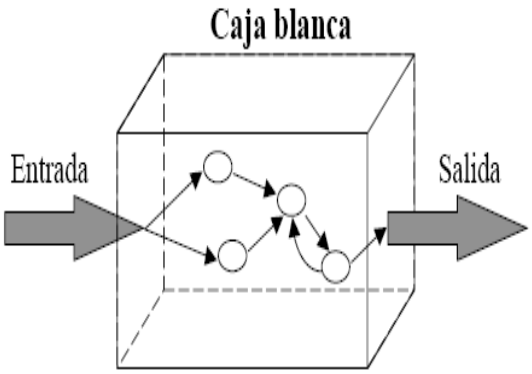


Figura 1.6 Prueba de caja blanca

1.11.1.1 Técnica de Prueba Camino Básico

Este método fue propuesto por Tom McCabe y 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 de caminos de ejecución. Los casos de prueba obtenidos del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia.[39]

Para empezar a trabajar en el método de camino básico se debe introducir la notación de grafo de flujo, notación que representa el flujo de control lógico ilustrada en la Figura 1.7

[39] Ídem a la referencia 4

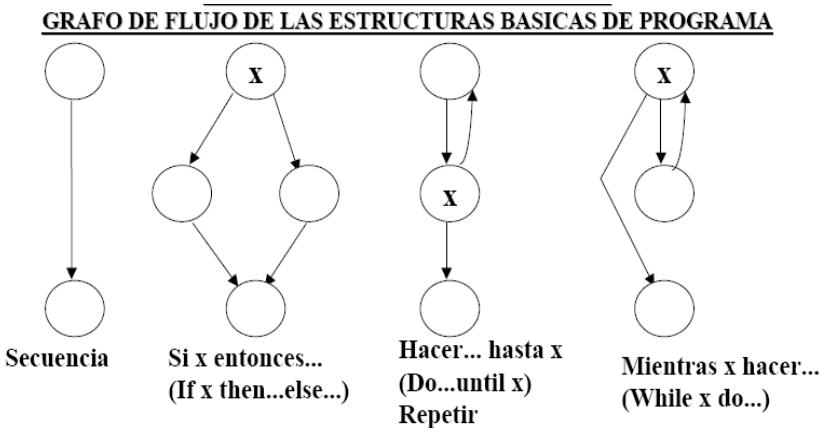


Figura 1.7 Grafo de las estructuras básicas del programa

Desarrollo del método.

- ✓ Primero se procede a dibujar el grafo de flujo asociado a partir del código fuente lo cual se ilustra en las Figuras 1.8 y Figura 1.9.

```
1<?php
  2 if(!function_exists('plaser_handler')) 3die('Violacion de acceso.');
  4 require_once 'PLASER/dbz_class.php';
  4 require_once '../general/validar.php';
  4 require_once '../general/validacion.php';
  4 $methodname = "ListarTipoEstructura ";
  4 global $espacio;
  4 $tipo = intval($args['tipo_salida']);
  //SELECCIONANDO EL TIPO DE SALIDA
  5 switch($tipo)
  {
  6 case 1:
    7 $tipo = '.pdf';
    8 break;
  9 case 2:
    10 $tipo = '.xls';
    break;
  11 default:
    12 tipo = '';
  }

?> 13
```

Figura1.8 Código

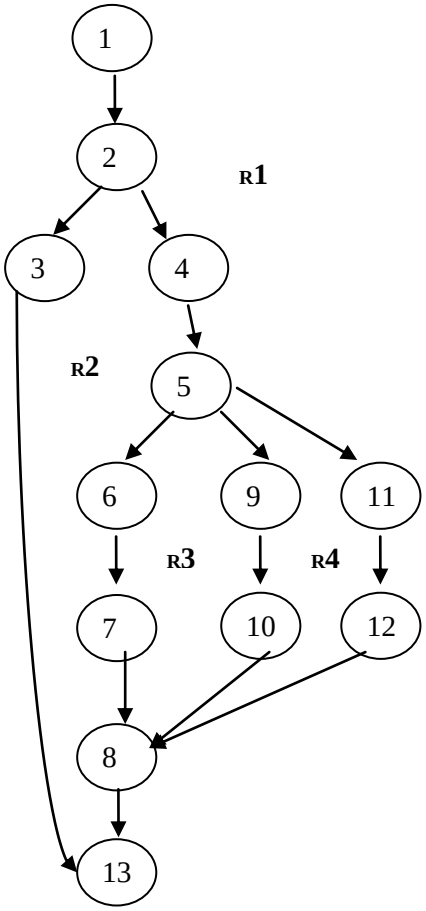


Figura 1.9 Grafo asociado

Se procede a calcular la complejidad ciclomática. La misma brinda la información de: Número de caminos independientes del conjunto básico de un programa y da un límite superior para el número de pruebas que se debe realizar, ejecutar cada sentencia al menos una vez. Se calcula de la siguiente forma:

V(G)= A – N +2 y es equivalente a la cantidad de regiones cerradas en el grafo

A: son las aristas (→), N: Son los nodos (◯). Además se calcula V (G) = NV + 2 NV=> El nodo que tenga una bifurcación y V(G) = Cantidad de regiones cerradas del grafo.

- ✓ Se determina un conjunto básico de caminos independientes.
- ✓ Se separan los casos de pruebas que obliguen a la ejecución de cada camino del conjunto básico.[40]

1.11.2 Método de caja Negra

Esta prueba se denomina prueba de comportamiento y básicamente se centra en los requisitos funcionales del software y se ilustra en la figura 1.10, lo que le permite al

[40] UCI 2006, Caso de prueba.

ingeniero de software obtener un conjunto de entradas que ejerciten todos los requisitos funcionales de este y así obtener un conjunto de errores para poder corregirlos. Esta técnica no es una alternativa a la prueba de caja blanca, más bien se trata de un nuevo enfoque que trata de descubrir diferentes errores de los métodos de caja blanca. Esta técnica intenta descubrir errores de las siguientes categorías.

- ✓ Funciones incorrectas o ausentes.
- ✓ Errores de interfaz.
- ✓ Errores de estructura de datos o de conexión a bases de datos externas.
- ✓ Errores de rendimiento.
- ✓ Errores de inicialización o terminación.[41]

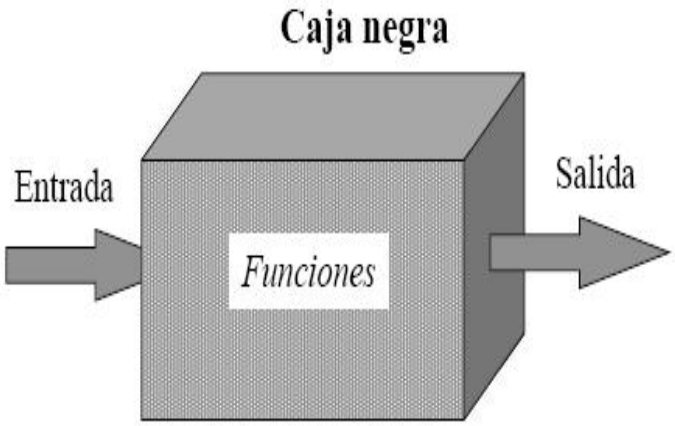


Figura 1.10 Método de prueba de caja Negra

1.11.2.1 Técnica de Prueba Partición Equivalente

El método partición equivalente divide los campos de entrada de un programa en clases de datos de los que se pueden derivar casos de pruebas, pues un caso ideal de pruebas descubre de forma inmediata una clase de errores. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar.

El diseño de estos casos de prueba para la partición equivalente se basa en la evaluación de las clases de equivalencia y estas clases se puede definir como: Una clase de equivalencia representa un conjunto de estados válidos o no válidos para condiciones de entrada.[42]

[41] Ídem a la referencia 4
[42] Ídem a la referencia 4

Si un conjunto de objetos puede unirse por medio de relaciones simétricas, transitivas y reflexivas, entonces existe una clase de equivalencia.

Las clases de equivalencia se pueden describir de acuerdo con las siguientes directrices:

- ✓ Si una condición de entrada especifica un rango, se define una clase de equivalencia válida y dos no válidas.
- ✓ Si una condición de entrada requiere un valor específico se requiere de una clase de equivalencia valida y dos no validas
- ✓ Si una condición de entrada especifica un miembro de un conjunto, se define una clase de equivalencia válida y una no válida.
- ✓ Si una condición de entrada es lógica, se define una clase de equivalencia válida y una no válida.[43]

Los mismos criterios se aplican a las salidas esperadas: hay que intentar generar resultados en todas y cada una de las clases.

Aplicando estas directrices se ejecutan casos de pruebas para cada elemento de datos del campo de entrada a desarrollar. Los casos se seleccionan de forma que ejerciten el mayor número de atributos de cada clase de equivalencia a la vez.

Para aplicar esta técnica de prueba se tienen en cuenta los siguientes pasos:

Primeramente se deben identificar las clases de equivalencia lo cual se hace tomando cada condición de entrada y aplicándole las directrices antes expuestas, luego se identifican los casos de pruebas de partición equivalente.

Condición externa	Clases de Equivalencia Válidas	Clases de Equivalencia Inválidas
Condición de entrada	Clases válidas para esa condición de entrada	Clases inválidas para esa condición de entrada

Tabla 1.2 Caracteristicas de los Grafos

1.11.2.1.1 Pasos para identificar Clases de Equivalencia

1. Identificación de las condiciones de las entradas del programa, es decir, restricciones de formato o contenido de los datos de entrada.
2. A partir de ellas, se identifican clases de equivalencia que pueden ser:
   a) De datos válidos.
   b) De datos no válidos o erróneos.

[43] Ídem a la referencia 4

3. Existen algunas reglas que ayudan a identificar clases :
   a) Si se especifica un rango de valores para los datos de entrada, recreará una clase válida y dos clases no válidas
   b) Si se especifica un número finito y consecutivo de valores, se creará una clase válida y dos no válidas.
   c) Si se especifica una situación del tipo «debe ser» o booleana (por ejemplo, «el primer carácter debe ser una letra»), se identifican una clase válida («es una letra») y una no válida («no es una letra»)
   d) Si se especifica un conjunto de valores admitidos y se sabe que el programa trata de forma diferente cada uno de ellos, se identifica una clase válida por cada valor y una no válida.
   e) En cualquier caso, si se sospecha que ciertos elementos de una clase no se tratan igual que el resto de la misma, deben dividirse en clases menores.[44]

1.11.2.1.2 Pasos para Identificar Casos de Pruebas de Partición Equivalente

El último paso del método es el uso de las clases de equivalencia para identificar los casos de prueba correspondientes. Este proceso consta de las siguientes fases:

4. Asignación de un número único a cada clase de equivalencia.
5. Hasta que todas las clases de equivalencia válidas hayan sido cubiertas por (incorporadas a) casos de prueba, se tratará de escribir un caso que cubra tantas clases válidas no incorporadas como sea posible.
6. Hasta que todas las clases de equivalencia no válidas hayan sido cubiertas por casos de prueba, escribir un caso para una única clase no válida sin cubrir.[45]

1.11.2.1.3 Un ejemplo

Ejemplo: Aplicación bancaria en la que el operador debe proporcionar un código, un nombre y una operación. [Ver Tabla 1.3].

Condición de entrada	Clases válidas	Clases inválidas
Código área Nº de 3 dígitos que no empieza con 0 ni 1)	(1) 200 <= código <=999	(2) código <200 (3) código >999 (4) no es número
Nombre para identificar la operación	(5) seis caracteres	(6) menos de 6 caracteres (7) más de 6 caracteres
Orden Una de las siguientes	(8) <<cheque>> (9) <<depósito>> (10) <<pago factura>>	(12) ninguna orden válida

[44] Ídem a la referencia 4
[45] Ídem a la referencia 4

(11)<<retirada de fondos>>

Tabla 1.3 Ejemplo de clases válidas inválidas usando la técnica de particiones equivalentes

Entonces queda diseñar casos de prueba que cubran todas las clases de equivalencia, tanto válida como inválida, y para las inválidas en casos de prueba distintos.[46]

1.11.2.2 Técnica Análisis de los Valores Límites

Por razones no del todo claras los errores tienden a darse en los mismos valores límites de entrada, por ejemplo si se tiene que entrar un número y en los requerimientos plantean que tiene que ser de tres dígitos y no puede empezar por 1, por lo que el análisis de los valores límites lleva a una elección de casos de pruebas que ejerciten los valores límites.

El análisis de valores límite es una técnica de diseño de casos de prueba que complementa a la partición equivalente. En lugar de seleccionar cualquier elemento de una clase de equivalencia, el Análisis de Valores Límites (AVL) lleva a la elección de casos de prueba en los extremos de la clase.

Las directrices de AVL son similares en muchos aspectos a las que proporciona la partición equivalente:

- ✓ Si una condición de entrada especifica un rango delimitado por los valores a y b, se deben diseñar casos de prueba para los valores a y b, y para los valores justo por debajo y justo por encima de a y b, respectivamente.
- ✓ Si una condición de entrada especifica un número de valores, se deben desarrollar casos de prueba que ejerciten los valores máximo y mínimo. También se deben probar los valores justo por encima y justo por debajo del máximo y del mínimo.
- ✓ Aplicar las directrices 1 y 2 a las condiciones de salida. Por ejemplo, supongamos que se requiere una tabla de Temperatura / Presión como salida de un programa de análisis de ingeniería. Se deben diseñar casos de prueba que creen un informe de salida que produzca el máximo (y el mínimo) número permitido de entradas en la tabla.
- ✓ Si las estructuras de datos internas tienen límites preestablecidos (por ejemplo, una matriz que tenga un límite definido de 100 entradas), hay que asegurarse

[46] I. Brito, 2003, ¨Las pruebas de software al configurar CASE¨

de diseñar un caso de prueba que ejercite la estructura de datos en sus límites.

La mayoría de Los ingenieros del software llevan a cabo de forma intuitiva alguna forma de AVL. Aplicando las directrices que se acaban de exponer, la prueba de límites será más completa y, por tanto, tendrá una mayor probabilidad de detectar errores.[47]

1.12 Diseñar casos de Pruebas a partir de los Casos de Uso

Las técnicas de pruebas de software asumen una serie de principios que condicionan su aplicación, entre los que se pueden encontrar las técnicas de caja blanca y de caja negra sobre las cuales se ha hablado en apartados anteriores. Resulta difícil tratar de llevar las pruebas exhaustivas del software porque no se pueden probar todas las posibles combinaciones de entradas o todos los caminos lógicos del mismo, por lo que no queda otro remedio de no poder demostrar la ausencia de errores.

En consecuencia de esto resulta imprescindible que el diseño de las pruebas se apoye en la selección de algunas de las entradas o situaciones que ejercitar dentro del dominio de valores, siempre en función de que sean representativas en su comportamiento de otros casos similares.

Nada impide que el diseño y la planificación de las pruebas se inicien en paralelo al trabajo de especificación. No sólo es algo posible sino se trata de una práctica recomendada para lograr mayor calidad, productividad y disminución de riesgos. Claro que con el aumento de la complejidad de las tecnologías de software ha supuesto la evolución de diseño sobre todo las técnicas de especificación y de diseño de software. Así como la llegada de las interfaces de usuario gráficas y las aplicaciones, cuyo funcionamiento está basado en eventos que complica el manejo en las pruebas de las combinaciones de entradas y de las opciones de funcionamiento.

Para generar casos de prueba se debe generar todos los posibles escenarios, o caminos de ejecución [Ver figura 1.11], de cada caso de uso. En el dibujo siguiente se muestra un ejemplo genérico donde se aprecia un camino de ejecución principal, flecha de color oscuro, y cuatro caminos de ejecución alternativos, uno que provoca la finalización de la ejecución y tres que hacen retroceder la ejecución a un paso anterior.

[47] Ídem a la referencia 4

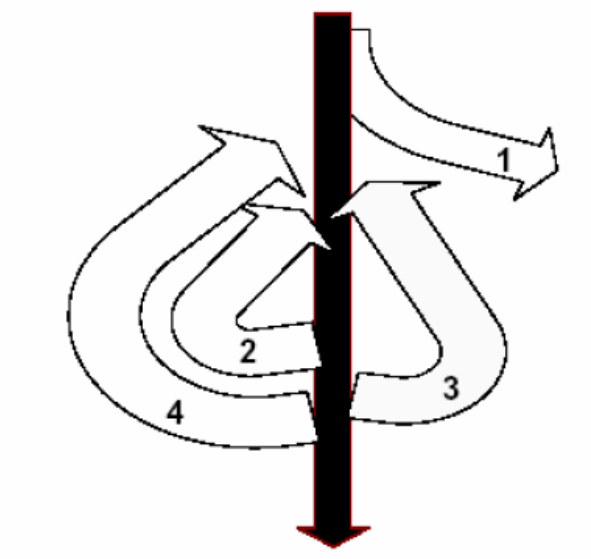


Figura 1.11 Camino de ejecucion principal y sus caminos alternativos

Después se identifican los casos de prueba a partir de esos escenarios y, por último, se identifican los valores a probar de cada caso de prueba. Tomando como punto de partida los casos de uso y su descripción no formal se obtendrá al final una lista de casos de prueba, con los valores que deben probar y los resultados esperados para cada caso.[48]

1.12.1 Descubrir los requisitos del Cliente

El objetivo de los desarrolladores de software es complacer al cliente, es decir satisfacer los requisitos del usuario para el sistema. Estos requisitos pueden ser requisitos de software, requisitos de productos, o requisitos de pruebas. Por lo tanto la meta debe ser capturar y comprobar los requisitos del usuario para así asegurar que todos los requisitos son completados por el diseño, y que el diseño es acorde con los requisitos especificados.[49]

Muchas veces los requisitos del sistema ya existen en forma de documentos de requisitos. Los casos de uso se utilizan para correlacionar cada escenario con los requisitos que completa. Si los requisitos no existen, modelar el sistema a través de los casos de uso, permite el descubrimiento de estos requisitos.

El modelo de casos de uso [Ver figura 1.12] contiene el actor y caso de uso del sistema. Los actores representan usuarios y otros sistemas que interaccionan con el sistema. Se dibujan como “muñecos” de palo. Representan el tipo de usuario, no una instancia de usuario. Los casos de uso representan el comportamiento del sistema, los

[48] F.P, López, ¨Caso de uso: una guia para la definición de caso de prueba¨
[49] Ídem a la referencia 48

escenarios que el sistema atraviesa en respuesta a un estímulo desde un actor. Se dibujan como elipses.

Las líneas representan la comunicación entre un actor y un caso del uso. Cada caso de uso representa una parte de la funcionalidad que se puede implementar, y cada actor representa a alguien o a algo fuera del sistema que interactúa con él. Cada caso de uso se documenta con una descripción del escenario. La descripción puede ser escrita en modo de texto o en un formato paso a paso. Los diagramas de actividad ofrecen una herramienta gráfica para modelar el proceso de un caso de uso.

Figura 1.12 Modelo de caso de uso configurar área de salud

1.12.2 Pasos para generar las pruebas a partir de los Casos de Uso

Algo importante de un caso de uso para la generación automática de un caso de prueba son los caminos de ejecución. Este camino se divide en dos: el camino principal o secuencia normal y los caminos alternativos o excepciones.

Todos los escenarios de caso de uso posibles serán utilizados como base para crear las pruebas. Un caso de prueba será un conjunto de entradas con datos de prueba, unas condiciones de ejecución, y unos resultados esperados.

El propósito de un caso de prueba es identificar y comunicar las condiciones que se llevarán a cabo en la prueba. Los casos de la prueba son necesarios para verificar la aplicación exitosa y aceptable de los requisitos del producto (casos de uso). Se describirá un proceso de tres-pasos para generar los casos de prueba a partir de un caso de uso totalmente detallado:

- ✓ Para cada caso de uso, generar un sistema completo de escenarios.
- ✓ Para cada escenario, identificar por lo menos un caso de prueba y las condiciones que hagan que se ejecute.
- ✓ Para cada caso de prueba, identificar los valores de los datos con los cuales se hará la prueba.

Paso 1:
Se procede a leer todas las descripciones textuales de los casos de uso con el fin de identificar todos los posibles caminos de ejecución del mismo, es decir todas las combinaciones posibles entre el camino principal y los caminos alternativos y se le asigna un nombre. Cada combinación será un escenario de uso.

Paso 2:
Después de generado todos los escenarios, el próximo paso es identificar los casos de pruebas, esto se puede lograr analizando los escenarios y repasando también las descripciones textuales de los casos de uso. Debe haber un caso de prueba por lo menos para cada escenario, pero probablemente habrá más.

Paso 3:
Una vez identificados todos los casos de prueba todos ellos deben repasarse y validarse para así asegurar exactitud e identificar los casos de prueba redundantes o que faltan. Entonces una vez que sean aprobados, el paso final es sustituir los valores reales de los datos para las pruebas. Sin los datos, los casos de prueba (o métodos de prueba) no pueden ser implementados ni ejecutados; son sólo descripciones de condiciones, escenarios, y caminos. Por lo tanto, es necesario identificar valores reales que se utilizarán en la implementación de las pruebas finales.

1.12.3 Ventajas e inconvenientes

Se pueden obtener rápidamente los casos de pruebas en solo tres pasos a través del lenguaje natural que se interpreta de las descripciones de los casos de uso. La dificultad de implantación es baja, porque no requiere ningún formalismo sino que se basa directamente en el lenguaje natural y solo necesita tres pasos para obtener un conjunto de casos de prueba a partir de un caso de uso.

La planificación de las pruebas del sistema en las primeras fases del desarrollo ayuda a validar los requisitos funcionales. Esta planificación debe empezar tan pronto como se comience a disponer de los requisitos funcionales. La obtención de los casos de prueba del sistema puede sistematizarse e integrarse en el proceso de desarrollo.

1.13 Conclusiones

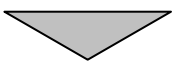
Si se encuentran y corrigen defectos de codificación antes de compilar, se ahorraría tiempo de encontrar esos defectos en el flujo de prueba, por lo que se proponen dos premisas fundamentales:

1. Lograr que los programadores revisen y prueben su código antes de compilarlo y entregarlos. [Ver Anexo A]
2. Mejorar el proceso de revisiones técnicas formales llevadas a cabo en el proyecto. Tratando de definir claramente los objetivos de la revisión,

documentarlos y darle seguimiento a los errores encontrados. Las revisiones técnicas formales pueden ser tan efectivas como las pruebas en el descubrimiento de errores.

Después de analizar las debilidades y fortalezas del flujo de trabajo estudiado, se decide, conformar la propuesta de flujo de trabajo de prueba. Tratando que se ajuste a las necesidades reales del proyecto SISalud, esta será guiada por una estrategia de prueba por niveles. Además se aplicarán pruebas de caja blanca, por el método de camino básico y pruebas de caja negra a partir de las descripciones de los casos de uso utilizando la técnica de particiones equivalentes.

Capítulo 2: Propuesta del Flujo de Trabajo de Prueba para el Proyecto SISalud



Capítulo 2: Propuesta del Flujo de Trabajo de Prueba para el Proyecto SISalud

Introducción

En el presente capítulo se describe la propuesta del flujo de trabajo de prueba para el grupo de desarrollo del proyecto SISalud, partiendo de las dos premisas fundamentales definidas en las conclusiones del capítulo 1. Después de elaborar un marco teórico sobre el tema de las pruebas, este permitió conocer los procesos o actividades a desarrollar y los métodos para llevar a cabo dichas actividades con el objetivo de mejorar el desarrollo de este flujo de trabajo en el proyecto.

2.1 Propuesta de Flujo de Trabajo de Prueba

La figura 2.1 muestra el flujo de trabajo de prueba propuesto para cada iteración del desarrollo de los componentes o módulos del proyecto SISalud.

Esta propuesta cuenta con las siguientes actividades:

Primero se planifican las pruebas, después se analizan y diseñan, se implementan, se ejecutan y se evalúan, donde si el resultado de la evaluación no es positivo se debe regresar a la actividad planificar pruebas, con el fin de volver a repetir el ciclo de vida del flujo de trabajo propuesto; de ser el resultado de la evaluación positivo, se pasa a la última actividad: aceptar las pruebas, donde quedarán cumplidos los objetivos de prueba de la iteración pasando al proceso de liberación correspondiente por parte del grupo de desarrollo de conjunto con su líder de proyecto, al grupo de desarrollo del proyecto SISalud de la empresa Softel, donde se realizarán las pruebas de aceptación según las estrategias definidas por la empresa.

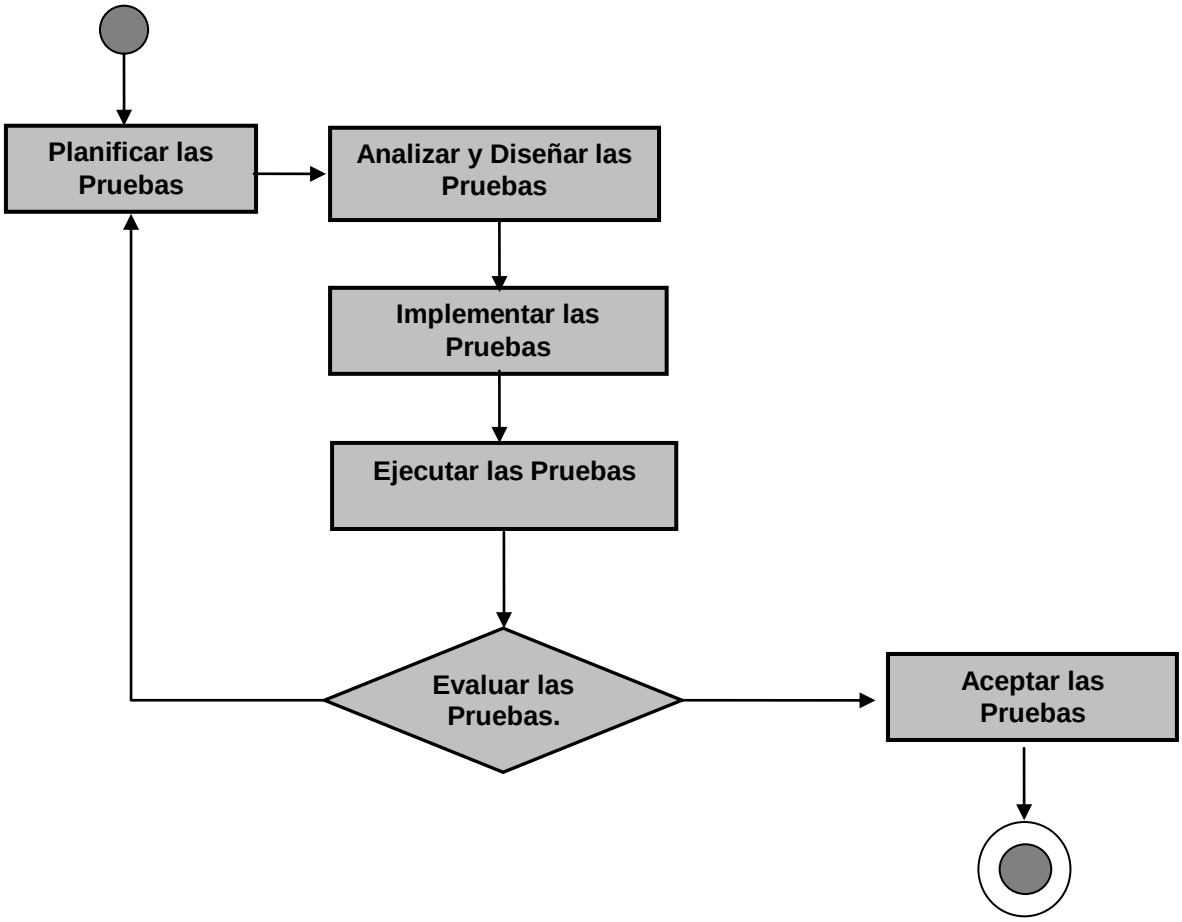


Figura 2.1 Propuesta de flujo de trabajo de prueba para el proyecto SISalud

2.2 Actividades de la Propuesta de Flujo de trabajo de Prueba

- ✓ Planificar las Pruebas.
- ✓ Analizar y Diseñar las Pruebas.
- ✓ Implementar las Pruebas.
- ✓ Ejecutar las Pruebas.
- ✓ Evaluar las Pruebas.
- ✓ Aceptar las Pruebas.

2.2.1 Planificar pruebas

En la actividad planificar pruebas [Ver figura 2.2], la función fundamental es describir la estrategia de prueba de la iteración definiendo claramente el método apropiado para ejecutarlas, los objetivos, las posibles pruebas a realizar, las pautas a seguir y el enfoque de las mismas. El objetivo de esta actividad es realizar el Plan de prueba que describe los recursos y la planificación de las mismas. [Ver Anexo B]. El cual posee los siguientes epígrafes más significativos:

- ✓ Introducción.
- ✓ Propósito.
- ✓ Alcance.
- ✓ Estrategia de evolución del Plan.
- ✓ Requerimientos a Probar.
- ✓ Estrategia de prueba.
- ✓ Infraestructura de prueba.
- ✓ Recursos.

Los trabajadores: administrador de prueba, analista de prueba y diseñador de prueba, mediante las actividades que realizan en la iteración participan en la elaboración de este plan, siendo el administrador de prueba el principal responsable de elaborar el Plan de prueba, que es el artefacto de salida de esta actividad y debe poseer como artefactos de entrada, el modelo de análisis, modelo de diseño, modelo de caso de uso, descripción de los casos de uso, descripción de la Arquitectura y documento visión. Además puede tener una lista de ideas de prueba, documentos o artefactos que le faciliten la confección del plan.

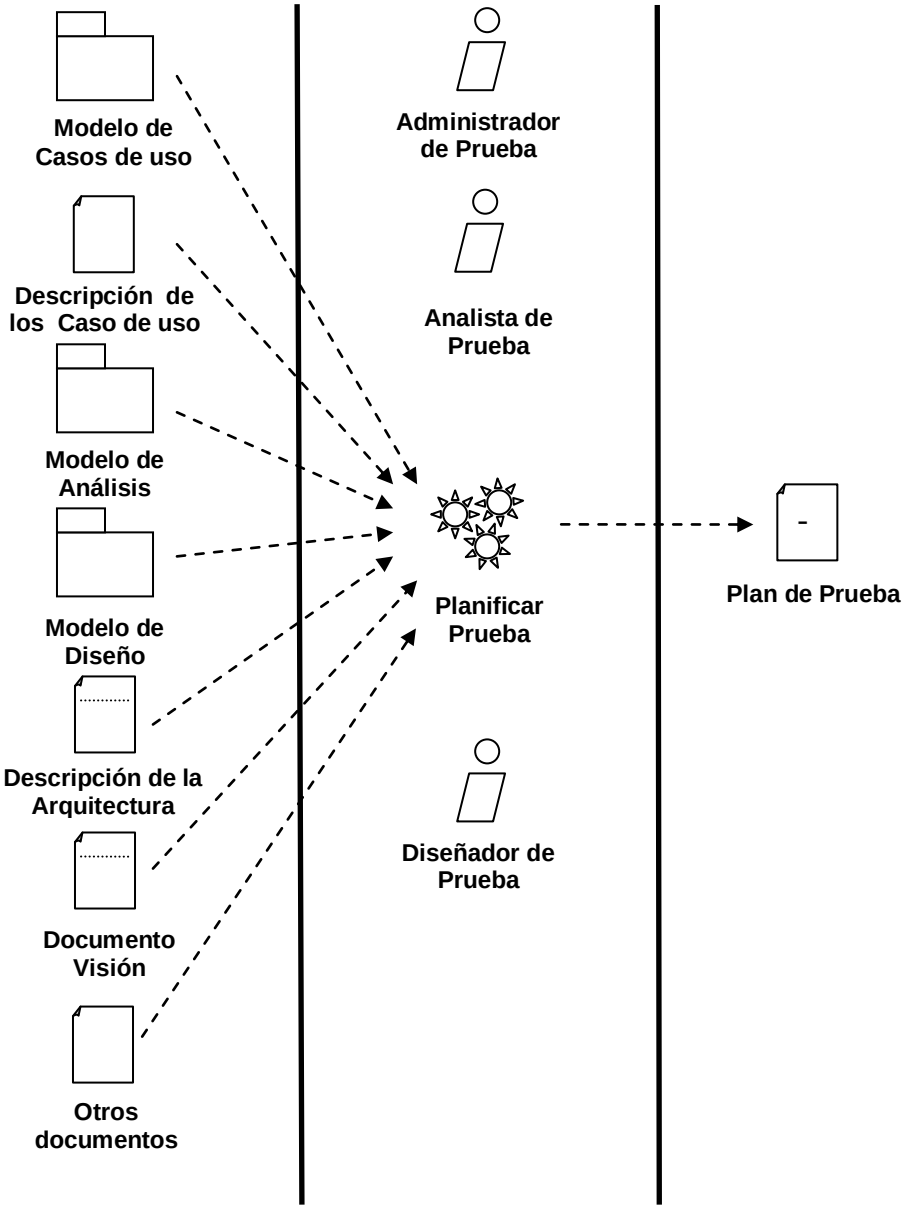


Figura 2.2 Entrada y salida de la actividad planificar prueba

2.2.2 Analizar y Diseñar las pruebas

La actividad diseñar prueba [Ver figura 2.3], fluye a partir de las principales ideas de las pruebas que se van a realizar, al llevar a cabo un diseño de las pruebas y los diferentes casos de pruebas a implementar. Guiándose siempre por los siguientes epígrafes del Plan de prueba:

- ✓ Alcance.
- ✓ Identificación del proyecto.
- ✓ Requerimientos a Probar.
- ✓ Estrategia de prueba.

- ✓ Tipos de pruebas.
- ✓ Recursos.

Esta actividad verifica los enfoques de la pruebas, por ejemplo: si estas nos pueden brindar resultados precisos y es apropiado para los recursos disponibles. Además su objetivo es lograr una solución de implementación apropiada de cada técnica de pruebas o encontrar técnicas alternativas que puedan usarse; es decir se analizan, se identifican y se construyen los procedimientos de prueba que especifican como realizar uno o varios casos de prueba, sirven de guía para que un individuo sepa cómo realizar casos de prueba manualmente o una especificación de cómo interactuar con una herramienta para ejecutar una prueba.

Los casos de pruebas de la iteración, también se coleccionan y se administran los datos de prueba generándose los scripts de datos de prueba, así como se realizan las descripciones de los casos de pruebas los cuales describen la forma de probar un sistema incluyendo las entradas, salidas y resultados esperados.

Los casos de prueba describen bajo que condiciones debe probarse el sistema y ellos pueden derivarse de las descripciones de los casos de uso del diseño. Siendo los artefactos de salida de esta actividad los casos de prueba, los scripts de datos de prueba y los procedimientos de prueba. [Ver Anexo C].

En esta actividad intervienen los trabajadores analista de prueba y diseñador de prueba y tiene como artefactos de entrada, el modelo de implementación, el modelo de caso de uso, la descripción de los casos de uso, el plan de prueba, la descripción de la arquitectura, el documento visión y el modelo de diseño.

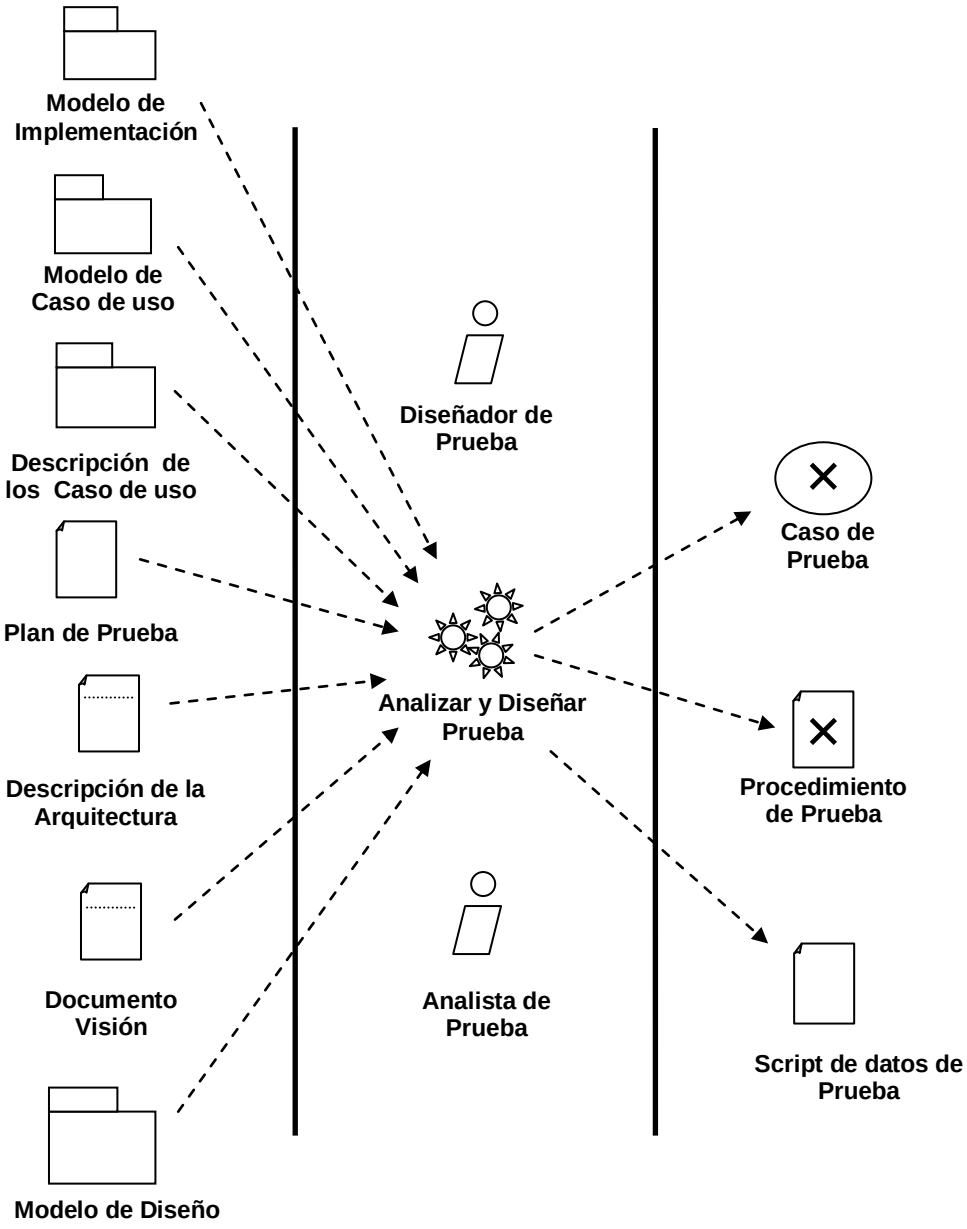


Figura 2.3 Entrada y salida de la actividad analizar y diseñar prueba

Al describir los casos de pruebas deben quedar definidos:

- ✓ Descripción General.
- ✓ Nombre de la Prueba.
- ✓ Descripción.
- ✓ Condiciones de ejecución.
- ✓ Entrada.
- ✓ Resultado esperado.
- ✓ Evaluación de la Prueba.

2.2.3 Implementar las Pruebas

El propósito de la actividad [Ver figura 2.4], implementar prueba consiste en automatizar uno o varios procedimientos de pruebas, pues esto ahorraría tiempo y esfuerzo a la hora de ejecutar las pruebas, así como a la hora de llevar a cabo pruebas de regresión sería bueno tener una herramienta que lo hiciese.

Existen herramientas que cuando se ejecuta una prueba con ella, esta graba los pasos de la prueba y los guarda, lo cual es algo muy útil a la hora de ejecutar pruebas de regresión. Esta actividad la deben ejecutar los trabajadores analista de prueba y diseñador de pruebas, los mismos son responsables de actividades que tributan al objetivo de la actividad. Los artefactos de entrada son los procedimientos de prueba y los artefactos de salida los componentes de prueba.

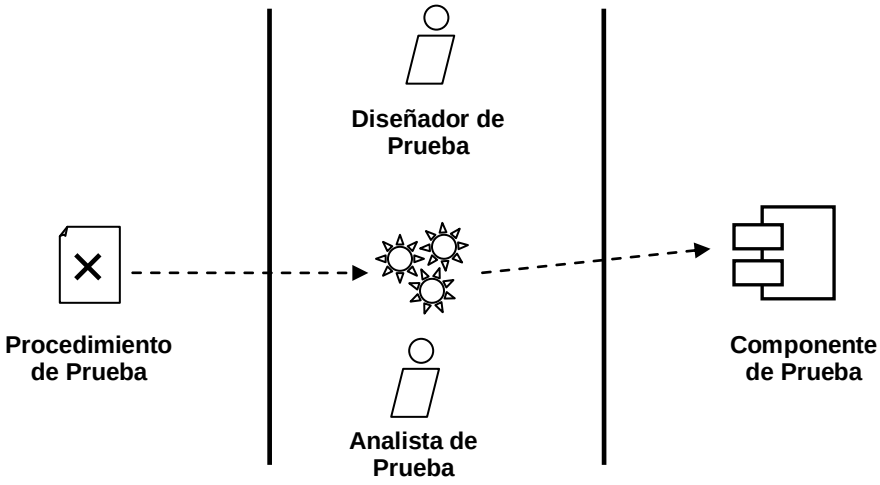


Figura 2.4 Entradas y salidas de la actividad implementar pruebas

2.2.4 Ejecutar las Pruebas

La actividad ejecutar pruebas [Ver figura 2.5] tiene como objetivo realizar correctamente las pruebas planificadas y diseñadas. Se van a realizar las pruebas unitarias, de integración y de sistema según los objetivos de la iteración.

Es decir se centra en la ejecución del artefacto casos de pruebas, a partir de las descripciones de los casos de prueba, apoyándose de los componentes de pruebas que automatizan uno o varios procedimientos de prueba proporcionando entradas de pruebas, controlando y monitoreando los componentes a probar. El uso de herramientas de prueba facilita y mejora el proceso de ejecución de las pruebas,

además se ejecutarán algunas pruebas manualmente, siempre guiadas por los epígrafes del Plan de pruebas:

- ✓ Estrategia de prueba.
- ✓ Tipos de pruebas.
- ✓ Recursos.

En la actividad, interviene el trabajador probador y la misma tiene como entrada los artefactos casos de prueba, procedimiento de prueba, componente de prueba, Plan de prueba y como artefacto de salida la lista de defectos, que no son más que las anomalías del sistema, que se detectan durante la ejecución de las pruebas.

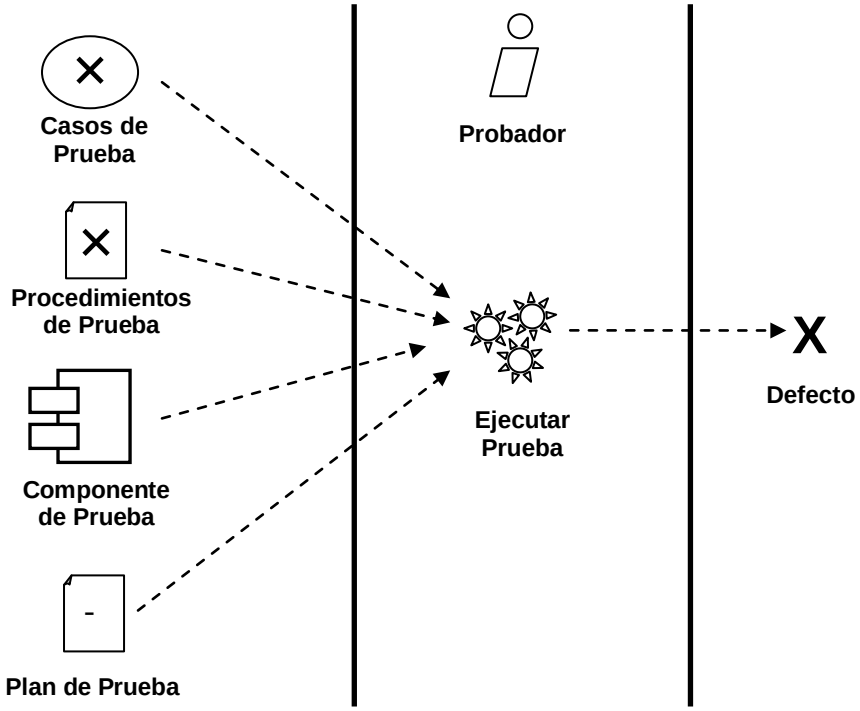


Figura 2.5 Entradas y salidas de la actividad ejecutar pruebas

2.2.5 Evaluación de las Pruebas

Esta actividad [ver figura 2.6] se centra en evaluar los esfuerzos de las pruebas en una iteración. Para evaluar los resultados de pruebas estos deben ser comparados con los objetivos esbozados en el Plan de prueba. Esta actividad reúne fundamentalmente mantener y mejorar la calidad de las pruebas, pues permite re-usar las anteriores actividades del actual ciclo de prueba, en ciclo de pruebas posteriores. La misma presenta como entrada los artefactos Plan de prueba, modelo de prueba, los defectos y como artefacto de salida la evaluación de las pruebas. En esta actividad intervienen

todos los trabajadores del flujo de prueba: administrador de prueba, analista de prueba, diseñador de prueba y probador.

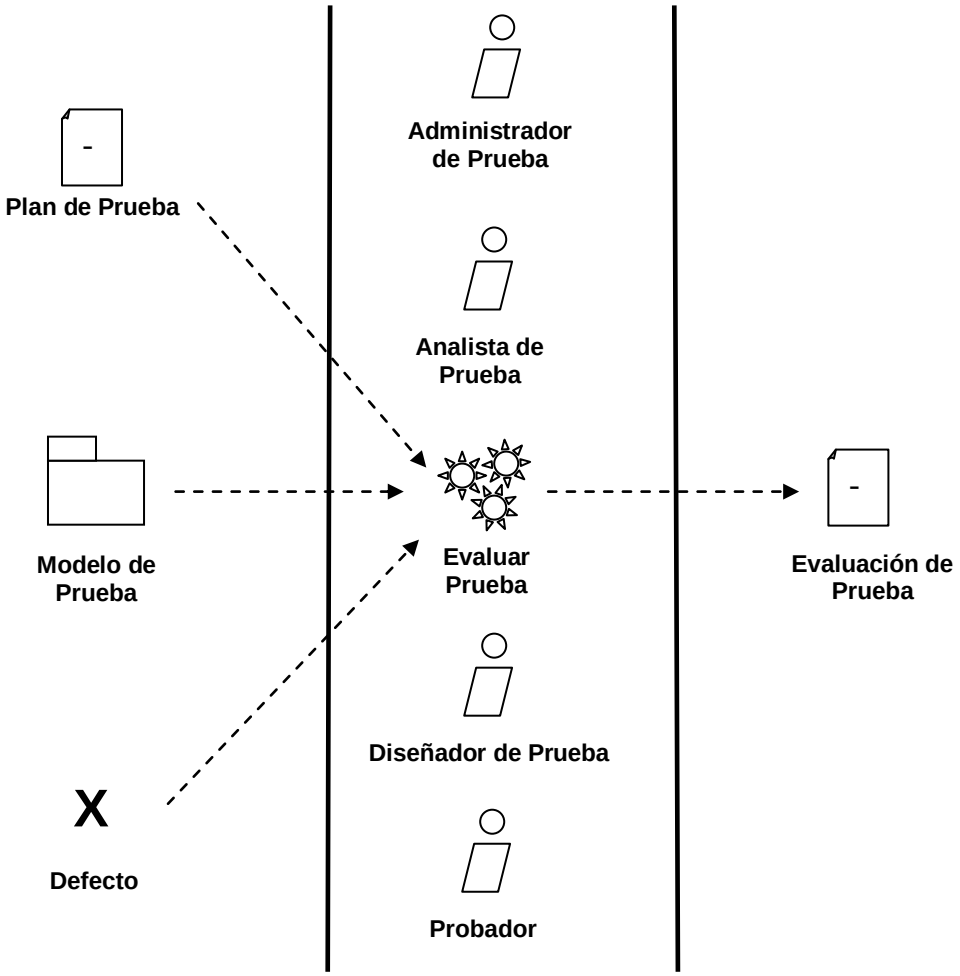


Figura 2.6 Entradas y salidas de la actividad evaluar pruebas

2.2.6 Aceptar las Pruebas

Esta actividad se centra en la aceptación o no de las pruebas a nivel de proyecto por el líder de proyecto de la universidad teniendo en cuenta la evaluación de las mismas, empezando entonces los procesos de gestión de cambios y/o liberación del grupo de proyecto de Softel. En esta actividad intervienen todos los trabajadores del flujo de trabajo de prueba: administrador de prueba, analista de prueba, diseñador de prueba y probador.

2.3 Trabajadores propuestos

- ✓ Administrador de Prueba.
- ✓ Analista de Prueba.
- ✓ Diseñador de Prueba.

- ✓ Probador.

2.3.1 Administrador de Prueba

El administrador de prueba, es responsable del éxito global de las pruebas, dirige y organiza el proceso.

Subactividades que debe cumplir:

- ✓ Identificar los elementos que motiven o faciliten las pruebas.
- ✓ Evaluar y auditar la calidad.
- ✓ Evaluar y mejorar las pruebas.
- ✓ Informar al líder de proyecto de los resultados del proceso de aceptación de la pruebas.
- ✓ Evaluar los resultados de las pruebas.

2.3.2 Analista de Prueba

Es responsable de identificar y definir las pruebas requeridas en una iteración, supervisando el progreso de las pruebas, así como el resultado para cada ciclo de prueba. El rol de manera general tiene la responsabilidad de representar las necesidades de los clientes que no se relacionan directamente con el proyecto. Este rol es una especialización del administrador de prueba.

Subactividades que debe cumplir:

- ✓ Identificar los objetivos de las pruebas.
- ✓ Identificar las ideas de las pruebas.
- ✓ Generar los scripts de datos de prueba.

2.3.3 Diseñador de Prueba

El diseñador de prueba, se centra en identificar las técnicas apropiadas, herramientas y pautas para llevar a cabo las pruebas y usar los recursos correspondientes para el esfuerzo de las mismas.

Subactividades que debe cumplir:

- ✓ Definir enfoque de las pruebas.
- ✓ Diseñar los casos de prueba.
- ✓ Describir los casos de prueba
- ✓ Identificar herramientas apropiadas para la ejecución de las pruebas.
- ✓ Estructurar la implementación de las pruebas.
- ✓ Identificar y definir técnicas apropiadas de prueba.

2.3.4 Probador

El probador es encargado de la actividad de mayor esfuerzo del flujo de trabajo de prueba, las pruebas se realizan ejecutando los casos de pruebas a partir de de sus

procedimientos. Estas pruebas se pueden ejecutar de forma manual, o a través de componentes de pruebas o herramientas de pruebas, las cuales nos ayudan a aminorar el esfuerzo de las mismas para una iteración así como nos facilitan ejecutar pruebas de sistema, como son las de carga, rendimiento, stress entre otras.

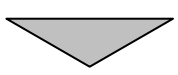
Subactividades que debe cumplir:

- ✓ Analizar y evaluar fallos como resultado de las pruebas.
- ✓ Identificar y listar los defectos encontrados.

2.4 Conclusiones

En este capitulo se ha descrito la propuesta de flujo de trabajo de prueba para el grupo de desarrollo del proyecto SISalud, en el cual se proponen 4 trabajadores con las respectivas actividades que deben cumplir. Además se propone que el flujo esté en constante evaluación, con el objetivo de valorar su funcionamiento e ir agregando nuevas ideas para ir mejorándolo y obtener un flujo de trabajo sostenido en el proyecto.

Capítulo 3: Ejecución del Flujo de Trabajo Propuesto



Capítulo 3: Ejecución del Flujo de Trabajo Propuesto

Introducción

El presente capítulo se basará en ejecutar el flujo de trabajo de prueba propuesto, de una forma práctica, al aplicarlo en el proyecto SISalud, en específico al módulo RAS, en la primera iteración de la fase de construcción del primer ciclo de desarrollo del módulo. Se describirán los epígrafes más significativos del Plan de prueba. Se definirán los casos de prueba diseñados, el scripts de datos de prueba, los procedimientos de prueba más significativos, indicando cuales fueron implementados a través de un componente de prueba, la lista de defectos identificados en la ejecución de los casos de prueba más significativos y la evaluación de las pruebas.

3.1 Plan de Prueba del módulo RAS

3.1.1 Introducción

Este documento se confecciona con el objetivo de definir el Plan de prueba para la aplicación RAS del Proyecto SISalud en la primera iteración de la fase de construcción del primer ciclo de desarrollo del módulo. El proposito de este plan es el siguiente:

1. Definir el alcance por etapas de las pruebas de la iteración.
2. Definir la estrategia de evolución de chequeo de este plan.
3. Definir la estrategia de pruebas a llevar a cabo en la iteración.
4. Definir la infraestructura de pruebas.
5. Definir los recursos necesarios.
6. Definir los roles y las responsabilidades.
7. Definir el cronograma de trabajo.

3.1.2 Alcance

Etapa I

Pruebas de unidad a los métodos de la capa de negocio: Configurar Sistema

Pruebas de unidad a los métodos de la capa de negocio: Configurar Área de salud.

Etapa II

Pruebas de integración de los métodos que cubren las funcionalidades de Configurar Sistema con los métodos de la capa de presentación.

Pruebas de integración de los métodos que cubren las funcionalidades de Configurar Área de salud con los métodos de la capa de presentación.

Etapa III

Pruebas de integración de todos los métodos probados en la Etapa II, cubriendo las funcionalidades de Configurar Sistema y Configurar Área de salud, con la capa de presentación del módulo.

Pruebas de integración con el componente de seguridad (SAAA) cubriendo las funcionalidades de seguridad.

3.1.3 Estrategia de Evolución del Plan

El plan será chequeado diariamente para realizar los ajustes necesarios. Al finalizar el día se reúne el equipo de prueba para verificar el cumplimiento del plan, analizar los ajustes necesarios al plan, al cronograma y retroalimentar al equipo de desarrollo.

3.1.4 Requerimientos a Probar

En el expediente del componente RAS del proyecto SISalud se describen los requerimientos funcionales y no funcionales a probar atendiendo al alcance de este plan.

3.1.5 Estrategia de Prueba

A continuación se describe el flujo que será implementado durante todo el período de ejecución de las pruebas de la iteración así como se detallan las diferentes estrategias a seguir en cada una de las etapas definidas en el alcance de este plan.

El flujo de trabajo se inicia elaborando este plan e iniciando la ejecución de las actividades de análisis y diseño, implementación, ejecución y aceptación de las pruebas.

Para la ejecución de la etapa I se deben diseñar casos de prueba de unidad utilizando el método de caja blanca con el objetivo de probar los caminos básicos más significativos, se implementará un componente de prueba en el que se tenga la posibilidad de introducir los datos de entrada y se obtenga visualmente en la pantalla el resultado. Estos resultados serán evaluados teniendo en cuenta los resultados esperados y al final del día se hará un análisis de los errores detectados para informar al grupo de desarrollo para que paralelamente al proceso de prueba se desarrolle el proceso de gestión de cambios a través del líder de proyecto de la universidad.

Para la generación de la etapa II se deben diseñar casos de prueba de caja negra utilizando la interfaz de usuario implementada con el objetivo de probar la integración de Configurar Sistema con los componentes correspondientes a la capa de presentación y las funcionalidades que le corresponden respectivamente. De la misma forma se probará Configurar Área de salud. Se deben introducir los datos de entrada descritos en los procedimientos de cada caso de prueba, registrar los resultados obtenidos para concluir evaluándose de esta manera el caso de prueba

a partir de los resultados obtenidos y los resultados esperados. Al final del día se hará un análisis de los errores detectados para informar al grupo de desarrollo para que paralelamente al proceso de prueba se desarrolle el proceso de gestión de cambios a través del líder de proyecto de la universidad.

Para la ejecución de la etapa III se deben diseñar y ejecutar casos de prueba de caja negra utilizando la interfaz de usuario implementada con el objetivo de validar la integración de las funcionalidades de Configurar Sistema, Configurar Área de salud y el componente de seguridad (SAAA). Al final del día se hará un análisis de los errores detectados para informar a través del líder de proyecto de la universidad al grupo de desarrollo para que paralelamente al proceso de prueba se desarrolle el proceso de gestión de cambios.

En el desarrollo de cada etapa debe tenerse en cuenta también las estrategias de evolución del Plan descrito en el epígrafe correspondiente.

Después de ser positivo el resultado de evolución de las 3 etapas se debe realizar el proceso de aceptación de las pruebas, el objetivo de esta actividad debe ser, teniendo en cuenta que corresponde a la primera iteración de la etapa de construcción del ciclo de desarrollo, evaluar el cumplimiento de los objetivos de la iteración y si son satisfactorios los 3, entonces el resultado del proceso de aceptación es positivo de lo contrario no debe aceptarse informando en ambos casos los resultados al líder del proyecto de estos resultados para determinar la continuidad del proceso de gestión de cambio y/o la liberación del grupo de desarrollo al grupo de proyecto de Softel.

En la figura 3.1 se muestra gráficamente la estrategia de prueba descrita en el plan.

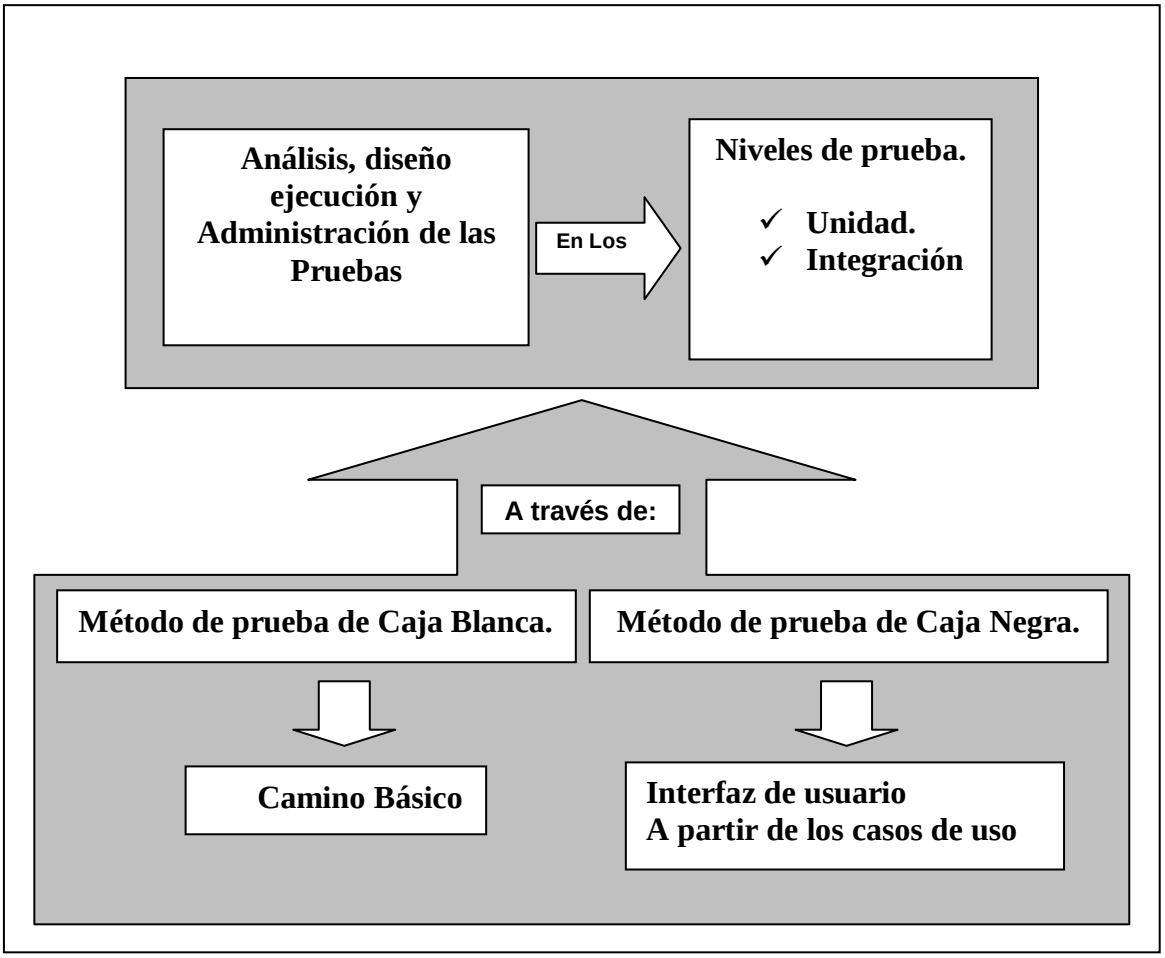


Figura 3.1 Eestrategia de prueba del plan de prueba

3.1.6 Prueba de integridad de los datos en la Base de datos

3.1.6.1 Objetivo de la Prueba

El objetivo de esta prueba es verificar que las inserciones, actualizaciones, y eliminaciones de los datos de la BD se ejecutan debidamente, es decir si los métodos de estas funcionalidades funcionan bien.

3.1.6.2 Técnica

Se utilizarán métodos de prueba de caja negra a partir de la descripción de los casos de uso de inserción, modificación y eliminación de datos.

3.1.6.3 Criterio de Evaluación

Todos los métodos y procesos de acceso a la base de datos funcionan como fueron diseñados y sin datos corruptos.

3.1.6.4 Consideraciones Especiales

Se ejecutarán los métodos a través de la capa de presentación y con usuarios editores y visualizador en los niveles, nacional, provincial, municipal y unidad de salud, con el

debido acceso para facilitar la prueba, las mismas deberán devolver datos de pruebas insertados en la BD con anterioridad.

3.1.7 Prueba de Funcionalidad

La prueba de funcionalidad se enfoca en requerimientos para verificar que se corresponden directamente a casos de usos o funciones y reglas del negocio. Los objetivos de estas pruebas son verificar la aceptación de los datos, el proceso y la implementación de los requerimientos funcionales correspondientes a ser cubiertos en la iteración. Este tipo de prueba se basa en técnicas de caja negra que consisten en verificar la aplicación y sus procesos interactuando por medio de la interfaz de usuario y analizar los resultados obtenidos para asegurar la funcionalidad apropiada de las opciones Configurar Sistema y Configurar Área de salud del RAS.

3.1.7.1 Técnica

Ejecución de cada proceso o función usando datos válidos y no válidos, para verificar lo siguiente:

- ✓ Se obtienen los resultados esperados cuando se usan datos válidos.
- ✓ Cuando se usan datos no válidos se despliegan los mensajes de error o de advertencia que corresponden.
- ✓ Se aplica apropiadamente cada regla del negocio.

3.1.7.2 Criterio de Evaluación

Todas las pruebas planificadas se realizaron. Todos los defectos encontrados han sido debidamente identificados.

3.1.7.3 Consideraciones Especiales

Se ejecutarán los métodos a través de la capa de presentación y con usuarios con el debido acceso para facilitar la prueba.

3.1.8 Prueba de Interfaz de Usuario

Esta prueba verifica que la interfaz de la aplicación proporcione al usuario el acceso y navegación a través de las funciones apropiadas. Además asegura que los objetos presentes en la interfaz de usuario se muestren como se espera y conforme a los estándares establecidos por el proyecto.

Se verificará lo siguiente:

- ✓ Navegación a través de todas las funcionalidades, verificar que cada interfaz es amigable al usuario.
- ✓ Verificar las funciones de Ayuda Online.
- ✓ Validaciones de los campos.

- ✓ Métodos de acceso como el menú configurar sistema donde aparecen los listar.

3.1.8.1 Técnica

En los procedimientos de cada caso de prueba de funcionalidad de caja negra especificar en los en los resultados esperados los estados de los objetos para cada interfaz y cada objeto contemplado dentro de la misma.

3.1.8.2 Criterio de Evaluación

Cada ventana ha sido verificada exitosamente siendo consistente con una versión de referencia o estándar establecido.

3.1.8.3 Consideraciones Especiales

Se ejecutarán los casos de pruebas a través de la capa de presentación y con usuarios con el debido acceso para facilitar la prueba.

3.1.9 Pruebas de seguridad y control de acceso

La prueba de seguridad verifica el correcto funcionamiento del componente de seguridad SAAA a través del cual se valída que, solo los usuarios con derecho a acceder al sistema son capaces de acceder a las aplicaciones y solo a través de los niveles de acceso apropiados. Ademas de verificar lo siguiente:

- ✓ Que un usuario pueda acceder solo a las funciones o datos para los cuales su tipo de usuario tiene permiso con las funcionalidades editar y visualizar según el nivel en que este se encuentre.
- ✓ Verificar que solo los usuarios con acceso al sistema y a las aplicaciones, puedan acceder a ellos según los niveles definidos. (Nacional, Provincial, Municipal o Unidad de Salud)

3.1.9.1 Técnica

Seguridad en el ámbito de aplicación: Identificar y hacer una lista de cada tipo de usuario y las funciones y datos sobre las que cada tipo tiene permiso.

Crear pruebas para cada tipo de usuario y verificar cada permiso creando operaciones específicas para cada uno.

Modificar el tipo de usuario y volver a ejecutar las pruebas para los mismos. En cada caso, verificar que las funciones o datos adicionales están correctamente disponibles o son denegados.

Acceso en el ámbito de sistema: Ver consideraciones especiales más abajo.

3.1.9.2 Criterio de Evaluación

Para cada tipo de actor conocido las funciones y datos apropiados están disponibles, y todas las operaciones funcionan como se espera y ejecutan las pruebas de funcionalidad de la aplicación.

3.1.9.3 Consideraciones Especiales

El acceso al sistema debe ser garantizado por el componente de seguridad SAAA, mediante el cual valida los usuarios según los niveles de acceso que tienen. Se verifica si el usuario que se autentica está registrado y de no estarlo se reporta un error de acceso, en caso contrario se le da acceso al RAS.

3.1.10 Infraestructura de Pruebas

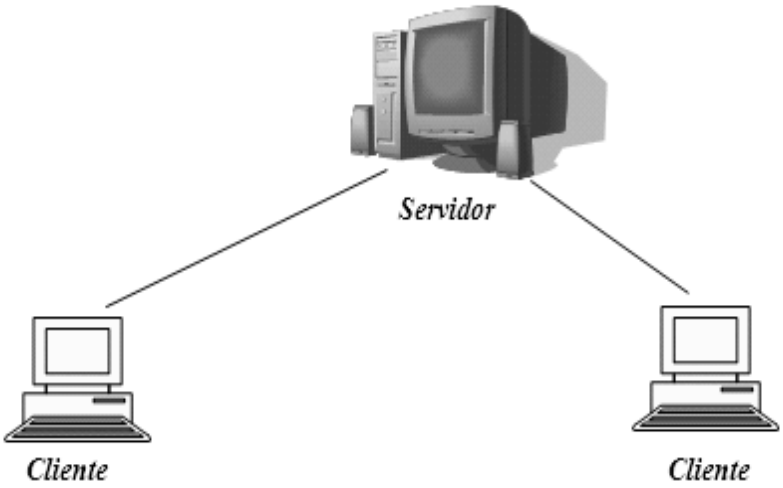


Figura 3.2 Infraestructura de pruebas

Escenario de Pruebas	
Servidor	Pentium IV 3,00 GHz, 2 Gb RAM y 250 Gb de Disco Duro.
Cliente	Pentium IV 2.40 GHz, 248 MB RAM y 80 Gb de Disco Duro.
Cliente	Pentium IV 2.40 GHz, 248 MB RAM y 80 Gb de Disco Duro.

Tabla 3.1 Requerimientos de hardware

Escenario de Pruebas	
Servidor	✓ MySQL 4 ✓ Sistema Operativo GNU/Linux, Distribución Red Hat. ✓ Servidor HTTP (preferiblemente

	Apache) que soporte VitualHosts Navegador web (web browser) que soporte DHTML y CSS2.
Cliente	✓ PHP 4.3.4 ✓ Especificaciones de PHP: PEAR de PHP Librería PEAR-SOAP 0.8RC3 Módulo XSLT (Sablotron) en PHP Módulo DBX en PHP Habilitar 'ctype' (desde el 4.3.0 esta builtin) Habilitar '—enable-memory-limit' en compilación, ya que algunos scripts consumen memoria y tienen puesto ini_set('memory_limit', '12M'); o también, aunque no recomendable, poner en el php.ini: memory_limit = 12M ✓ MySQL 4 ✓ MySQL-Front 3.2 ✓ Servidor HTTP (preferiblemente Apache) que soporte VitualHosts ✓ Navegador web (web browser) que soporte DHTML y CSS2. (Mozilla 1.5) ✓ Dreamweaver 8 ✓ Stylus Studio 5 ✓ Nusphere 4.0
Cliente	✓ PHP 4.3.4 ✓ Especificaciones de PHP: PEAR de PHP Librería PEAR-SOAP 0.8RC3 Módulo XSLT (Sablotron) en PHP Módulo DBX en PHP Habilitar 'ctype' (desde el 4.3.0

	esta builtin) Habilitar '—enable-memory-limit' en compilación, ya que algunos scripts consumen memoria y tienen puesto ini_set('memory_limit', '12M'); o también, aunque no recomendable, poner en el php.ini: memory_limit = 12M ✓ MySQL 4 ✓ MySQL-Front 3.2 ✓ Servidor HTTP (preferiblemente Apache) que soporte VitualHosts ✓ Navegador web (web browser) que soporte DHTML y CSS2. (Internet Explorer 5.0 o superior) ✓ Dreamweaver 8 ✓ Stylus Studio 5 ✓ Nusphere 4.0

Tabla 3.2 Requerimienros de software

3.1.11 Recursos

En el proyecto SISalud los recursos sobre los cuales se ejecutarán las pruebas se propone que sean 3 PC, una que funcionará como servidor y 2 como clientes, con un equipo de pruebas integrado por 5 estudiantes de la universidad que cubrirán los 4 roles propuestos, 1 administrador de prueba, 1 una analista, 1 diseñador y 2 probadores, guiados por el líder del proyecto de la UCI y supervisados por un especialista de la empresa Softel.

3.1.12 Roles y responsabilidades

Rol	Cantidad	Responsabilidades
Administrador de Prueba	1	✓ Identifica, prioriza e implementa los casos de prueba. ✓ Genera el Plan de prueba para la iteración. ✓ Negociar o acordar las pruebas. ✓ Genera resumen de resultado de prueba. ✓ Verifica el progreso y efectividad de las pruebas

		✓ Genera informe de no conformidades.
Analista de Prueba	1	✓ Genera procedimientos de prueba. ✓ Identifica los casos prueba. ✓ Confecciona la lista de ideas para las pruebas. ✓ Genera documento con datos de pruebas.
Diseñador de Prueba	1	✓ Confecciona casos de pruebas. ✓ Genera las descripciones de los casos de pruebas. ✓ Genera modelo de prueba. ✓ Componente de prueba. ✓ Especificación de interfaz de prueba. ✓ Configuración del ambiente de prueba. ✓ Definir la suite de pruebas. ✓ Identificar y describir apropiadas técnicas de prueba. ✓ Identificar herramientas apropiadas de soporte. ✓ Definir y mantener la arquitectura de automatización de pruebas. ✓ Especificar y verificar las condiciones de las configuraciones de prueba.
Probador	2	✓ Ejecuta las pruebas ✓ Registra los resultados de las pruebas. ✓ Documenta los pedidos de cambio. ✓ Implementar pruebas individuales. ✓ Defecto. ✓ Verificar la forma en que se ejecutan las pruebas.
Líder de proyecto por la UCI	1	✓ Responsable del desarrollo de las pruebas. ✓ Responsable de revisiones técnicas formales al desarrollo de las pruebas. ✓ Responsable de aceptar o no el producto después de la fase de prueba.

Líder de Proyecto por Softel	1	✓ Responsable de revisiones técnicas formales al desarrollo de las pruebas. ✓ Responsable de aceptar o no el producto después de la fase de prueba.

Tabla 3.3 Roles y responsabilidades del flujo de trabajo pruebas

3.2 Pruebas Estructurales (Caja Blanca)

3.2.1 Método Buscar Plantilla

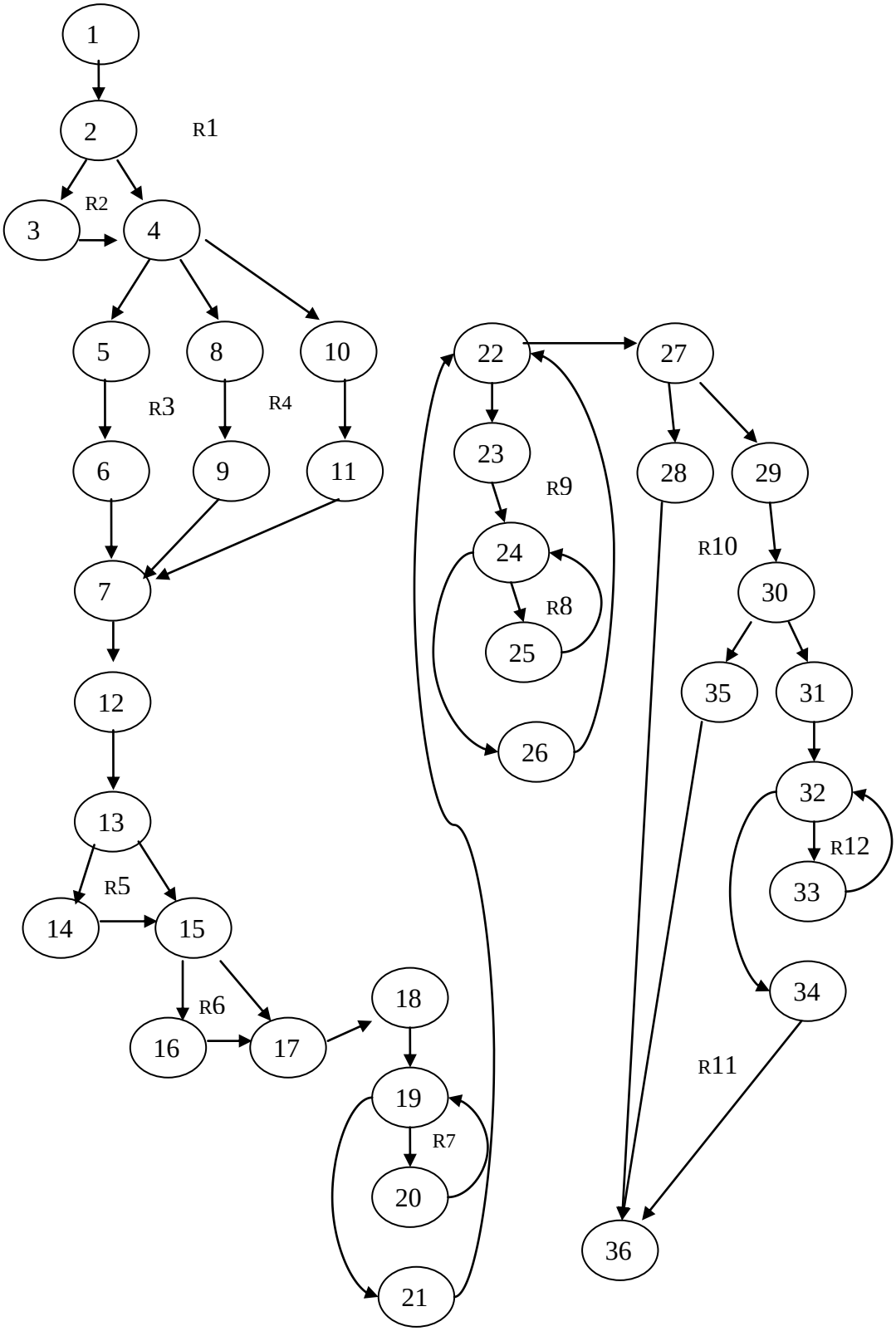


Figura 3.3 Representacion de un grafo de flujo

3.2.1.1 Complejidad Ciclomática

V(G) = A – N + 2 A => Arista, N=> Nodos

V(G) = 46 – 36 + 2

V(G) = 12

V(G) = NB + 1 NB=> El nodo que tenga una bifurcación.

V(G) = 11 + 1

V(G) = 12.

V(G) = Cantidad de regiones cerradas.

V(G) = 12.

Posibles caminos del método

Camino 1: 1-2-3-4-5-6-7-12-13-14-15-16-17-18-19-20-19-21-22-23-24-25-24-26-22-27-28-36.

Camino 2: 1-2-3-4-8-9-7-12-13-14-15-16-17-18-19-20-19-21-22-23-24-25-24-26-22-27-28-36.

Camino 3: 1-2-3-4-10-11-9-7-12-13-14-15-16-17-18-19-20-19-21-22-23-24-25-24-26-22-27-28-36.

Camino 4: 1-2-4-5-6-7-12-13-14-15-16-17-18-19-20-19-21-22-23-24-25-24-26-22-27-28-36.

Camino 5: 1-2-4-8-9-7-12-13-14-15-16-17-18-19-20-19-21-22-23-24-25-24-26-22-27-28-36.

Camino 6: 1-2-4-10-11-9-7-12-13-14-15-16-17-18-19-20-19-21-22-23-24-25-24-26-22-27-28-36.

Camino 7: 1-2-4-5-6-7-12-13-15-17-18-19-20-19-21-22-23-24-25-24-26-22-27-29-30-31-32-33-32-34-36.

Camino 8: 1-2-4-8-9-13-15-17-18-19-20-19-21-22-23-24-25-24-26-22-27-29-30-31-32-33-32-34-36.

Camino 9: 1-2-4-10-11-13-15-17-18-19-20-19-21-22-23-24-25-24-26-22-27-29-30-31-32-33-32-34-36.

Camino 10: 1-2-3-4-5-6-7-12-13-15-17-18-19-20-19-21-22-23-24-25-24-26-22-27-29-30-31-32-33-32-34-36.

Camino 11: 1-2-3-4-8-9-7-12-13-15-17-18-19-20-19-21-22-23-24-25-24-26-22-27-29-30-31-32-33-32-34-36.

Camino 12: 1-2-3-4-10-11-7-12-13-15-17-18-19-20-19-21-22-23-24-25-24-26-22-27-29-30-31-32-33-32-34-36.

3.2.1.2 Procedimiento de prueba Método Buscar Plantilla

Descripción General

El caso de prueba consiste en probar las condiciones lógicas del método buscar plantilla, para el diseño del mismo se realizo por la técnica de caja blanca camino básico, mediante la cual se calculo la complejidad ciclomática la cual nos brinda la información de: el número máximo de caminos lógicos a probar. Para la ejecución del mismo se creo un componente de prueba mediante el cual se probaran 4 caminos lógicos los cuales son los caminos 1, 2, 4,11.

Las pruebas realizadas son:

1. Buscar Plantilla por el criterio de búsqueda Id_cargo.
2. Buscar Plantilla por el criterio de búsqueda Descripción.
3. Buscar Plantilla por el criterio de búsqueda Id_cargo y Descripción.
4. Buscar Todas las plantillas.

3.2.1.2.1 CPR 1 Buscar Plantilla por el criterio de búsqueda por el id cargo

Descripción

Para la ejecución de esta prueba se ejecuta el componente de prueba buscar plantilla como resultado de la actividad implementar prueba.

Flujo central

- ✓ El componente de prueba muestra la interfaz con los diferentes criterios de búsquedas.
- ✓ En el label id Cargo se introduce el id de cargo correspondiente a la plantilla que se quiere buscar.
- ✓ El componente de prueba debe mostar la plantilla.

Condiciones de Ejecución

Tienen que estar en la BD los id_plantillas, id_cargos, y las plantillas y los cargos.

id_Plantilla	Plantilla	Id_cargo	Cargo
1	PLANTILLA 1	2	cargo 1
		6	Jefe Grupo Basico
19	PLANTILLA 10	2	cargo 1
15	PLANTILLA 15	2	cargo 1
		12	Ginecostreticia
		13	Enfermera Supervisora
22	PLANTILLA 16	2	cargo 1
		6	Jefe Grupo Basico
		12	Ginecostreticia
		13	Enfermera Supervisora
23	PLANTILLA 17	8	Medicina Interna
		9	Pediatria
		14	Técnico en higiene y epidemiología
		15	Técnico en en trabajo Social

2	PLANTILLA 2	2	cargo 1
		4	Licenciado en psicologia
		6	Jefe Grupo Basico
9	PLANTILLA 3	2	cargo 1
		6	Jefe Grupo Basico
		12	Ginecostreticia
		13	Enfermera Supervisora
13	PLANTILLA 4	6	Jefe Grupo Basico
14	PLANTILLA 5	14	Técnico en higiene y epidemiología
15	PLANTILLA 6	9	Pediatria
16	PLANTILLA 7	15	Técnico en en trabajo Social
17	PLANTILLA 8	4	Licenciado en psicologia
		6	Jefe Grupo Basico
		13	Enfermera Supervisora
18	PLANTILLA 9	2	cargo 1

Tabla 3.4 Condiciones de ejecucion para el procediminero buscar plantilla por id_cargo a través del componente de prueba

Clases válidas	Clases inválidas	Resultado Esperado	Resultado de la prueba	Observaciones	Cumplimiento %
Se intruduce el Id Cargo: 6		PLANTILLA 1	La prueba se realizó satisfactoriamente.	.	100%

Tabla 3.5 Resultados del CP buscar plantilla por el criterio de busqueda por id_cargo a través del componente de prueba

3.2.1.2.2 CPR2: Buscar Plantilla por el criterio de búsqueda Descripción

Descripción

Para la ejecución de esta prueba se ejecuta el componente de prueba buscar plantilla como resultado de la actividad implementar prueba.

Flujo Central

- ✓ El componente de prueba muestra la interfaz con los diferentes criterios de búsquedas.
- ✓ En el label Descripción se introduce el nombre de la plantilla que se quiere buscar.
- ✓ El componente de prueba debe mostrar el id_ plantilla, la plantilla y el id y los cargos correspondientes a esta plantilla.

Condiciones de ejecución

Tienen que estar en la BD los id_plantillas, id_cargos, y las plantillas y los cargos.

id_Plantilla	Plantilla	Id_cargo	Cargo
1	PLANTILLA 1	2	cargo 1
		6	Jefe Grupo Basico
19	PLANTILLA 10	2	cargo 1
15	PLANTILLA 15	2	cargo 1

		12	Ginecostreticia
		13	Enfermera Supervisora
22	PLANTILLA 16	2	cargo 1
		6	Jefe Grupo Basico
		12	Ginecostreticia
		13	Enfermera Supervisora
23	PLANTILLA 17	8	Medicina Interna
		9	Pediatria
		14	Técnico en higiene y epidemiología
		15	Técnico en en trabajo Social
2	PLANTILLA 2	2	cargo 1
		4	Licenciado en psicologia
		6	Jefe Grupo Basico
9	PLANTILLA 3	2	cargo 1
		6	Jefe Grupo Basico
		12	Ginecostreticia
		13	Enfermera Supervisora
13	PLANTILLA 4	6	Jefe Grupo Basico
14	PLANTILLA 5	14	Técnico en higiene y epidemiología
15	PLANTILLA 6	9	Pediatria
16	PLANTILLA 7	15	Técnico en en trabajo Social
17	PLANTILLA 8	4	Licenciado en psicologia
		6	Jefe Grupo Basico
		13	Enfermera Supervisora
18	PLANTILLA 9	2	cargo 1

Tabla 3.6 Condiciones de ejecucion para el procedimiento buscar Plantilla por el criterio de búsqueda descripción a través del componente de prueba

Clases válidas	Clases inválidas	Resultado Esperado	Resultado de la prueba	Observaciones	Cumplimiento %
PLANTILLA 2		Id plantilla: 2 PLANTILLA 2 Id cargo: 2 Cargo 1	La prueba se realizo satisfactoriamente		100%

Tabla 3.7 Resultados del CP buscar plantilla por el criterio de busqueda descripción a través del componente de prueba

3.2.1.2.3 CPR3: Buscar Plantilla por el criterio de búsqueda Id_cargo y Descripción

Descripción

Para la ejecución de esta prueba se ejecuta el componente de prueba buscar plantilla como resultado de la actividad implementar prueba.

Flujo Central

- ✓ El componente de prueba muestra la interfaz con los diferentes criterios de búsquedas.
- ✓ En el label Descripción se introduce el nombre de la plantilla que se quiere buscar.
- ✓ En el label Id Cargo se introduce el ID de cargo correspondiente a la plantilla que se quiere buscar otro id.
- ✓ El componente de prueba debe mostrar el id_ plantilla, la plantilla y el id y los cargos correspondientes a esta plantilla.

Condiciones de ejecución

id_Plantilla	Plantilla	Id_cargo	Cargo
1	PLANTILLA 1	2	cargo 1
		6	Jefe Grupo Basico
19	PLANTILLA 10	2	cargo 1
15	PLANTILLA 15	2	cargo 1
		12	Ginecostreticia
		13	Enfermera Supervisora
22	PLANTILLA 16	2	cargo 1
		6	Jefe Grupo Basico
		12	Ginecostreticia
		13	Enfermera Supervisora
23	PLANTILLA 17	8	Medicina Interna
		9	Pediatria
		14	Técnico en higiene y epidemiología
		15	Técnico en en trabajo Social
2	PLANTILLA 2	2	cargo 1
		4	Licenciado en psicologia
		6	Jefe Grupo Basico
9	PLANTILLA 3	2	cargo 1
		6	Jefe Grupo Basico
		12	Ginecostreticia
		13	Enfermera Supervisora
13	PLANTILLA 4	6	Jefe Grupo Basico
14	PLANTILLA 5	14	Técnico en higiene y epidemiología
15	PLANTILLA 6	9	Pediatria
16	PLANTILLA 7	15	Técnico en en trabajo Social
17	PLANTILLA 8	4	Licenciado en psicologia
		6	Jefe Grupo Basico
		13	Enfermera Supervisora
18	PLANTILLA 9	2	cargo 1

Tabla 3.8 Condiciones de ejecución para el procediminero buscar plantilla por los criterios de búsquedas id_cargo y descripción a través del componente de prueba

Clases válidas	Clases inválidas	Resultado Esperado	Resultado de la prueba	Observaciones	Cumplimiento %
PLANTILLA 10 Id_cargo: 2		Id plantilla: 19 PLANTILLA 10 Id cargo: 2 Cargo 1	La prueba se ejecutó correctamente		100%

Tabla 3.9 Resultados del CP buscar plantilla por los criterios de búsqueda id_cargo y descripción a través del componente de prueba

3.2.1.2.4 CPR4: Buscar Todas las plantillas

Descripción

Para la ejecución de esta prueba se ejecuta el componente de prueba buscar plantilla como resultado de la actividad implementar prueba.

Flujo Central

- ✓ El componente de prueba muestra la interfaz con los diferentes criterios de búsquedas.
- ✓ No se introducen valores en los label, id_cargo , id_descipcion.
- ✓ El componente de prueba debe mostrar todos los id_plantilla, id_cargo y las plantillas y los cargos.

Condiciones de ejecución

id_Plantilla	Plantilla	Id_cargo	Cargo
1	PLANTILLA 1	2	cargo 1
		6	Jefe Grupo Basico
19	PLANTILLA 10	2	cargo 1
15	PLANTILLA 15	2	cargo 1
		12	Ginecostreticia
		13	Enfermera Supervisora
22	PLANTILLA 16	2	cargo 1
		6	Jefe Grupo Basico
		12	Ginecostreticia
		13	Enfermera Supervisora
23	PLANTILLA 17	8	Medicina Interna
		9	Pediatria
		14	Técnico en higiene y epidemiología
		15	Técnico en en trabajo Social
2	PLANTILLA 2	2	cargo 1
		4	Licenciado en psicologia
		6	Jefe Grupo Basico
9	PLANTILLA 3	2	cargo 1
		6	Jefe Grupo Basico
		12	Ginecostreticia
		13	Enfermera Supervisora

13	PLANTILLA 4	6	Jefe Grupo Basico
14	PLANTILLA 5	14	Técnico en higiene y epidemiología
15	PLANTILLA 6	9	Pediatria
16	PLANTILLA 7	15	Técnico en en trabajo Social
17	PLANTILLA 8	4	Licenciado en psicologia
		6	Jefe Grupo Basico
		13	Enfermera Supervisora
18	PLANTILLA 9	2	cargo 1

Tabla 3.10 Condiciones de ejecución para el procediminero buscar plantilla sin criterios de búsquedas a través del componente de prueba

Clases válidas	Clases inválidas	Resultado Esperado	Resultado de la prueba	Observaciones	Cumplimiento %
No se introducen valores en los label id_cargo y descripción		Se debe mostrar todos los datos descritos en la condiciones de ejecución.	La prueba se ejecutó correctamente		100%

Tabla 3.11 Resultados del CP buscar plantilla sin criterios de búsquedas, a través del componente de prueba

3.3 Pruebas funcionales (Caja Negra)

En el siguiente diseño, se derivan un conjunto de casos de prueba funcionales que ejercitan algunos de los escenarios del RAS.

3.3.1 Descripción General Listar Plantilla

El caso de uso inicia cuando el usuario eiditor o visualizador, para los niveles de unidad de salud municipal, provincial y nacional, de Configurar Sistema accede al RAS para listar los Tipos de Estructuras existentes; el mismo selecciona en la opción de Configurar Sistema, específicamente Tipo de Estructuras, el caso de uso consiste en listar los tipos de estructuras y si el usuario desea puede también imprimir dicho listado en los formatos de PDF o XSL.

Las pruebas realizadas a este caso de uso son:

1. Buscar plantillas.
2. Agregar plantilla.
3. Editar plantilla.
4. Eliminar Plantilla.
5. Revisión de la ayuda.

3.3.1.1 CPR 1: Buscar Plantillas

Descripción

Se ejecuta la opción configurar sistema, plantillas y el sistema muestra en una pantalla las plantillas existentes.

Flujo central

- ✓ El sistema muestra la interfaz que le permite al usuario autenticarse.
- ✓ Si el usuario se autentica correctamente con el usuario domingo y contraseña domingo2007 el cual tiene permiso de editor Nacional, el sistema muestra la interfaz principal de la aplicación, el usuario ejecuta la opción RAS y elige en el menú principal Configurar Sistema, plantillas.

Condiciones de Ejecución

Tienen que estar presente en la base de datos los tipos de plantillas.

- ✓ PLANTILLA 1
- ✓ PLANTILLA 2
- ✓ PLANTILLA 3
- ✓ PLANTILLA 4
- ✓ PLANTILLA 5
- ✓ PLANTILLA 6
- ✓ PLANTILLA 7
- ✓ PLANTILLA 8
- ✓ PLANTILLA GBT

Clases válidas	Clases inválidas	Resultado Esperado	Resultado de la prueba	Observaciones	Cumplimiento %
Mostrar todos		PLANTILLA 1 PLANTILLA 2 PLANTILLA 3 PLANTILLA 4 PLANTILLA 5 PLANTILLA 6 PLANTILLA 7 PLANTILLA 8 PLANTILLA GBT	La prueba se realizó satisfactoriamente.	.	100%

	PLANTILLA 1 ¡”.	No introduzca caracteres extraños	La prueba no se realizó correctamente		100%

Tabla 3.12 Resultados del CP Buscar plantilla

3.3.1.2 CPR2: Agregar Plantilla

Descripción

Este caso de prueba consiste en probar la opción agregar plantilla.

Flujo Central

- ✓ El sistema muestra la interfaz que le permite al usuario autenticarse.
- ✓ Si el usuario se autentica correctamente con el usuario domingo y la contraseña domingo2007 el cual tiene permiso de editor nacional, el sistema muestra la interfaz principal de la aplicación el usuario ejecuta la opción RAS y el usuario elige en el menú principal Configurar Sistema, plantillas, después de realizar la búsqueda de las plantilla por cualquier criterio.
- ✓ El sistema muestra una pantalla donde existe la opción agregar.
- ✓ El sistema muestra una pantalla donde se llenan los datos de la plantilla.
- ✓ En el label nombre se pone la nueva plantilla, se selecciona el cargo, y se pone la cantidad después se da en la opción agregar.

Condiciones de ejecución

El sistema debe mostrar una pantalla donde existe la opción agregar.

En el label nombre se pone el nombre de la nueva plantilla, se selecciona el cargo, y se pone la cantidad después se da clic en la opción agregar.

Clases válidas	Clases inválidas	Resultado Esperado	Resultado de la prueba	Observaciones	Cumplimiento %
Plantilla A		Inserción de la plantilla A	La prueba se realizó satisfactoriamente		100%
	Plantilla a!”.	Se inserta la plantilla Plantilla a!”.	La plantilla ¡”. se inserta sin problemas.	No se validan los caracteres de entrada.	100%

Tabla 3.13 Resultados del CP Agregar plantilla

3.3.1.3 CPR2: Editar Plantilla

Descripción

Se ejecuta la opción configurar sistema, plantillas y el sistema muestra en una pantalla las plantillas existentes, se da clic en la que se desea editar y se modifican los campos según se desee.

Flujo Central

- ✓ El sistema muestra la interfaz que le permite al usuario autenticarse.
- ✓ Si el usuario se autentica correctamente con el usuario domingo y la contraseña domingo2007 el cual tiene permiso de editor nacional, el sistema muestra la interfaz principal de la aplicación el usuario ejecuta la opción RAS y el usuario elige en el menú principal Configurar Sistema, plantillas,
- ✓ El sistema muestra todas las plantillas existentes.
- ✓ Se selecciona la plantilla que se desea editar.
- ✓ Se modifican los campos que se desee para editar la plantilla.

Condiciones de ejecución

El sistema debe mostrar una pantalla con la plantilla que se desea editar.

En el label nombre se pone la plantilla a actualizar, se selecciona el cargo a actualizar, y se pone la cantidad a actualizar o puede ser uno de ellos, después se da en la opción agregar.

Por la PLANTILLA A se va a poner la PLANTILLA AAA

Clases válidas	Clases inválidas	Resultado Esperado	Resultado de la prueba	Observaciones	Cumplimiento %
PLANTILLA AAA		Se introdujeron sin problemas los cambios en la plantilla	La prueba se ejecutó correctamente	Intercambiar el label cargo por el de cantidad, para que primero se ponga la cantidad y después se seleccione el cargo.	100%
	PLANTILLA AAA ¡”.	No se ejecute la actualización de la plantilla	La prueba no se ejecutó correctamente	No se validan los caracteres de entrada.	100%

Tabla 3.14 Resultados del CP Editar plantilla

3.3.1.4 CPR2: Eliminar Plantilla

Descripción

Se ejecuta la opción configurar sistema, plantillas y el sistema muestra en una pantalla las plantillas existentes, se da clic en la que se desea editar y se modifican los campos según se desee.

Flujo Central

- ✓ El sistema muestra la interfaz que le permite al usuario autenticarse.

- ✓ Si el usuario se autentica correctamente con el usuario domingo y la contraseña domingo2007 el cual tiene permiso de editor nacional, el sistema muestra la interfaz principal de la aplicación el usuario ejecuta la opción RAS y el usuario elige en el menú principal Configurar Sistema, plantillas,
- ✓ El sistema muestra todas las plantillas existentes.
- ✓ Se selecciona la plantilla que se desea eliminar.
- ✓ Se da clic en la opcion aceptar.

Condiciones de ejecución

El sistema debe mostrar una pantalla con la opción eliminar.

Se escoge la plantilla a eliminar PLANTILLA AAA, se muestra una plantilla donde se selecciona la opción eliminar, el sistema muestra una advertencia de que si en realidad desea eliminar la plantilla y se da aceptar.

Clases válidas	Clases inválidas	Resultado Esperado	Resultado de la prueba	Observaciones	Cumplimiento %
PLANTILLA AAA		Se elimina la plantilla PLANTILLA AAA	La prueba se ejecutó correctamente		100%

Tabla 3.15 Resultados del CP Eliminar plantilla

3.3.1.5 CPR2: Revisión de la Ayuda

Descripción

Este caso de prueba consiste en probar la veracidad de la ayuda así como su descripción y estructura.

Flujo Central

- ✓ El sistema muestra la interfaz que le permite al usuario autenticarse.
- ✓ Si el usuario se autentica correctamente con el usuario domingo y la contraseña domingo2007 el cual tiene permiso de editor nacional, el sistema muestra la interfaz principal de la aplicación el usuario ejecuta la opción RAS.
- ✓ Se ejecuta el hipervínculo ayuda o se oprime la tecla F1.
- ✓ Se muestra la ventana con la ayuda correspondiente a la pantalla donde se esté trabajando.

Condiciones de ejecución

Se debe mostrar la ayuda cuando se presione la tecla del teclado F1.

Se debe mostrar la ayuda al ejecutar el hipervínculo ayuda de la pantalla del módulo RAS.

Clases válidas	Clases inválidas	Resultado Esperado	Resultado de la prueba	Observaciones	Cumplimiento %
Se presiona la tecla F 1		Se muestra la ayuda correspondiente a la pantalla usada.	No se ejecuta la prueba correctamente.	Se muestra la ayuda del navegador.	100%
Se ejecuta el hipervínculo ayuda		Se muestra la ayuda correspondiente a la pantalla usada.	La prueba se ejecutó correctamente.	Ver documento Prueba A	100%

Tabla 3.16 Resultados del CP revisión de la ayuda

3.3.1.6 Registro de Defectos y Dificultades Detectadas

Elemento	N°	Descripción de la No conformidad	Aspecto correspondiente	Etapa de la detección	Importante	Recomendaci s
Interfaz	1	No se validan los caracteres extraños	Esto permite que a la BD entren caracteres no necesarios.	Una Iteración de Prueba	X	Revisar la clas validaciones.
Interfaz	2	No se va a la ayuda cuando se presiona F1	No brinda la facilidad para los usuarios de encontrar la ayuda de una forma rápida y sencilla.	Una Iteración de Prueba	X	Revisar hacerlo.
Interfaz	3	Faltan funcionalidades en la ayuda	Ver documento Prueba A	Una iteración de prueba	X	Revisar págin ayuda

Tabla 3.17 Resumen de defectos y dificultades del procedimiento de prueba Listar plantilla

3.3.2 Descripción General Listar Tipos de Estructura

El caso de uso inicia cuando el usuario para los niveles de unidad de salud, municipal, provincial y nacional, de Configurar Sistema accede al Registro de Áreas de Salud para listar los Tipos de Estructuras existentes; él mismo selecciona en la opción de Configurar Sistema, específicamente Tipo de Estructuras El caso de uso consiste en listar los tipos de estructuras y si el usuario desea puede también imprimir dicho listado en los formatos de PDF o XSL.

Las pruebas realizadas a este caso de uso son:

1. Listar tipos de estructura.
2. Exportar a PDF.

3. Exportar a XSL.

3.3.2.1 CPR 1: Listar Tipos de Estructura

Descripción

Se ejecuta la opción configurar sistema, tipo de estructuras y el sistema muestra en una pantalla los tipos de estructuras existentes.

Flujo central

- ✓ El sistema muestra la interfaz que le permite al usuario autenticarse.
- ✓ Si el usuario se autentica correctamente con el usuario domingo y la contraseña domingo2007 él cual tiene permiso de editor nacional, el sistema muestra la interfaz principal de la aplicación el usuario ejecuta la opción RAS y el usuario elige en el menú principal Configurar Sistema, tipos de estructuras.

Condiciones de Ejecución

Tienen que estar presente en la BD los tipos de estructuras.

- ✓ Equipo Básico de Salud.
- ✓ Grupo Básico de Trabajo.

Clases válidas	Clases inválidas	Resultado Esperado	Resultado de la prueba	Observaciones	Cumplimiento %
		✓ Equipo Básico de Salud. ✓ Grupo Básico de Trabajo.	La prueba se realizó satisfactoriamente.	.	100 %

Tabla 3.184 Resulatados del CP Listar tipos de estructuras

3.3.2.2 CPR2: Exportar a PDF

Descripción

Este caso de prueba consiste en probar que el listado de estructuras se exporte a un documento PDF

Flujo Central

- ✓ El sistema muestra la interfaz que le permite al usuario autenticarse.
- ✓ Si el usuario se autentica correctamente con el usuario domingo y la contraseña domingo2007 el cual tiene permiso de editor nacional, el sistema

muestra la interfaz principal de la aplicación el usuario ejecuta la opción RAS y el usuario elige en el menú principal Configurar Sistema, tipos de estructuras.

- ✓ El sistema muestra todas las estructuras existentes en la BD.
- ✓ El usuario ejecuta la opción exportar a PDF.

Condiciones de ejecución

El sistema debe mostrar una pantalla en formato PDF con e las estructuras existentes en la BD.

Clases válidas	Clases inválidas	Resultado Esperado	Resultado de la prueba	Observaciones	Métrica externa	Submétrica externa	Cumplimiento %
		Documento PDF	La prueba no se realizó satisfactoriamente				

Tabla 3.19 Resultados del CP Exportar a pdf

3.3.2.3 CPR2: Exportar a XSL

Descripción

Este caso de prueba consiste en probar que el listado de estructuras se exporte a un documento XSL

Flujo Central

- ✓ El sistema muestra la interfaz que le permite al usuario autenticarse.
- ✓ Si el usuario se autentica correctamente con el usuario domingo y la contraseña domingo2007 el cual tiene permiso de editor nacional, el sistema muestra la interfaz principal de la aplicación el usuario ejecuta la opción RAS y el usuario elige en el menú principal Configurar Sistema, tipos de estructuras.
- ✓ El sistema muestra todas las estructuras existentes en la BD.
- ✓ El usuario ejecuta la opción exportar a XSL.

Condiciones de ejecución

El sistema debe mostrar una pantalla en formato PDF con e las estructuras existentes en la BD.

Clases válidas	Clases inválidas	Resultado Esperado	Resultado de la prueba	Observaciones	Métrica externa	Submétrica externa	Cumplimiento %
		Documento XSL	La prueba no se realizó satisfactoriamente				

Tabla 5 Resultados del CP Exportar a xsl

3.3.2.4 Registro de Defectos y Dificultades Detectadas

Elemento	N°	Descripción de la No conformidad	Aspecto correspondiente	Etapa de la detección	Importante	Recomendaciones
Interfaz	1	No se exporta a XSL los resultados de la consulta Listar Tipo de estructuras	En este requisito no cumple realmente la funcionalidad con respecto a la presentación de los Datos	prueba	X	Revisar en la clase exportar a XSL.
Interfaz	2	No se exporta a PDF los resultados de la consulta Listar Tipo de estructuras	En este requisito no cumple realmente la funcionalidad con respecto a la presentación de los Datos	prueba	X	Revisar en la clase exportar a PDF.

Tabla 3.21 Registro de defectos y dificultades del procedimiento de prueba listar tipo de estructuras

3.3.3 Descripción General Listar Áreas de Salud

El caso de uso inicia cuando el Usuario para los niveles de unidad de Salud municipal, provincial y nacional, de Configurar Sistema accede al Registro de Áreas de Salud para listar las áreas de salud existentes; el mismo selecciona en la opción de Configurar Sistema, específicamente Áreas de Salud, el caso de uso consiste en listar las áreas de salud y si el Usuario desea puede también imprimir dicho listado en los formatos de PDF o XSL.

Las pruebas realizadas a este caso de uso son:

1. Listar Áreas de Salud.
2. Exportar a PDF.
3. Exportar a XSL.
4. Editar Área de salud.

3.3.3.1 CPR 1: Listar Áreas de Salud

Descripción

Se ejecuta la opción configurar sistema, áreas de salud y el sistema muestra en una pantalla los tipos de áreas de salud existentes.

Flujo central

- ✓ El sistema muestra la interfaz que permite loguearse al usuario.
- ✓ Si el usuario se loguea correctamente con el login ypoza13 y pass yanet2007 el cual tiene permiso de editor unidad de salud, el sistema muestra la interfaz principal de la aplicación el usuario ejecuta la opción RAS y el usuario elige en el menú principal Configurar Sistema, Áreas de Salud.

Condiciones de Ejecución

Se ejecutó el scritp de datos de prueba el cual garantiza que en la BD estén insertados los datos de pruebas.

1. Policlínico Área VI.

Clases válidas	Clases inválidas	Resultado Esperado	Resultado de la prueba	Observaciones	Cumplimiento %
		1. Policlínico Área VI	La prueba se realizó satisfactoriamente.		

Tabla 6 Resultados del CP listar áreas de salud

3.3.3.2 CPR2: Exportar a PDF

Descripción

Este caso de prueba consiste en probar que el listado de las áreas de salud se exporte a un documento PDF

Flujo Central

- ✓ El sistema muestra la interfaz que permite loguearse al usuario.
- ✓ Si el usuario se loguea correctamente con el login ypoza13 y pass yanet2007 el cual tiene permiso de editor unidad de salud, el sistema muestra la interfaz principal de la aplicación el usuario ejecuta la opción RAS y el usuario elige en el menú principal Configurar Sistema, áreas de salud.
- ✓ El sistema muestra las áreas de salud existentes en la BD.
- ✓ El usuario ejecuta la opción exportar a PDF.

Condiciones de ejecución

El sistema debe mostrar una pantalla en formato PDF con las áreas de salud existente en la BD.

Clases válidas	Clases inválidas	Resultado Esperado	Resultado de la prueba	Observaciones	Cumplimiento %

		Documento PDF	La prueba no se realizó satisfactoriamente		

Tabla 3.23 Resultados del CP Exportar a pdf

3.3.3.3 CPR3: Exportar a XSL

Descripción

Este caso de prueba consiste en probar que el listado de las áreas de salud se exporte a un documento XSL

Flujo Central

- ✓ El sistema muestra la interfaz que permite loguearse al usuario.
- ✓ Si el usuario se loguea correctamente con el login ypoza13 y pass yanet2007 el cual tiene permiso de editor unidad de salud, el sistema muestra la interfaz principal de la aplicación el usuario ejecuta la opción RAS y el usuario elige en el menú principal Configurar Sistema, áreas de salud.
- ✓ El sistema muestra todas las áreas de salud existentes en la BD.
- ✓ El usuario ejecuta la opción exportar a XSL.

Condiciones de ejecución

El sistema muestra en formato PDF las áreas de salud existente en la BD.

Clases válidas	Clases inválidas	Resultado Esperado	Resultado de la prueba	Observaciones	Cumplimiento %
		Documento XSL	La prueba no se realizó satisfactoriamente		

Tabla 3.24 Resultados del CP Exportar a xsl

3.3.3.4 CPR4: Editar Área de Salud

Descripción

Este caso de prueba consiste en probar que el listado de las áreas de salud se exporte a un documento XSL.

Flujo Central

- ✓ El sistema muestra la interfaz que permite loguearse al usuario.
- ✓ Si el usuario se loguea correctamente con el login ypoza13 y pass yanet2007 el cual tiene permiso de editor unidad de salud, el sistema muestra la interfaz principal de la aplicación el usuario ejecuta la opción RAS y el usuario elige en el menú principal Configurar Sistema, áreas de salud.
- ✓ El sistema muestra todas las áreas de salud existentes en la BD.
- ✓ El usuario selecciona la opción editar dándole clic encima del nombre.

- ✓ El sistema muestra todos los datos introducidos anteriormente de esa área de salud
- ✓ El usuario modifica los datos q desee cambiar.

Condiciones de ejecución

El sistema debe mostrar una pantalla con los datos insertados con anterioridad por el usuario.

Clases válidas	Clases inválidas	Resultado Esperado	Resultado de la prueba	Observaciones	Cumplimiento %
	- En Código del Área y en Extensión introduzco: 5j89* Nkdsfn$45	Debe mostrar un mensaje de error indicando que solamente se pueden introducir números	La prueba se realizo satisfactoriamente		100%
- En Código del Área y en Extensión introduzco: 597 94812		Se deben de actualizar los datos en la base de datos	La prueba se realizo satisfactoriamente		100%
	En la opción Configurar Áreas de Salud->Hospitales Base cuando voy a pasar de Hospitales Base por Área a Hospitales Base por segunda vez y no hay ningún elemento seleccionado, el botón  se mantiene activado	Que el botón este desactivado o que muestre un mensaje de error indicando que se debe seleccionar un elemento.	La prueba no se realizo satisfactoriamente		100%

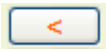
	En la opción Configurar Áreas de Salud->Localidades cuando no tengo ninguna localidad en Localidades del Área el botón  se mantiene activado	Que el botón este desactivado o que muestre un mensaje de error indicando que se debe seleccionar un elemento.	La prueba no se realizo satisfactoriamente		100%

Tabla 3.25 Resultados del CP Editar área de salud

3.3.3.5 Registro de Defectos y Dificultades Detectadas

Elemento	N°	Descripción de la No conformidad	Aspecto correspondiente	Etapa de la detección	Importante	Recomendaciones
Interfaz	**1**	No se muestra la información en un PDF sino que muestra la pantalla con un mensaje de error	En este requisito no se cumple con la funcionalidad respecto a la presentación de los datos.	prueba	X	Revisar en la clase exportar a PDF
Interfaz	**2**	No se exporta a XSL los resultados de la consulta Listar Áreas de Salud.	En este requisito no se cumple con la funcionalidad respecto a la presentación de los datos	prueba	X	Revisar en la clase exportar a XSL
Interfaz	3, 4	No se muestra un mensaje de error ni se desactiva el botón.	En este requisito no se cumple con la funcionalidad	prueba	X	Revisar en la clase donde se debe mostrar un error o desactivar ese botón.

Tabla 3.26 Registro de defectos y dificultades del procedimiento de prueba listar áreas de salud

3.3.4 Descripción General Configurar Área de Salud

El caso de uso inicia cuando el Usuario para los niveles de unidad de Salud, municipal, provincial y nacional, de Configurar Sistema accede al Registro de Áreas de Salud para buscar un área de salud y editarla en caso que tenga permiso para hacerlo; el

mismo selecciona en la opción de Configurar área de salud, el cual le aparecen varios criterios de búsqueda se llena uno o varios y se ejecuta la opción buscar, o se ejecuta la opción todos la cual muestra todas la áreas de salud existentes.

Las pruebas realizadas a este caso de uso son:

- ✓ Buscar Área de Salud (Nivel Nacional).
- ✓ Buscar Área de Salud (Nivel Área de Salud).
- ✓ Editar Área de Salud.

3.3.4.1 CPR 1: Buscar Área de salud (Nivel Nacional)

Descripción

Se ejecuta la opción configurar Área de salud, aparecen varios criterios de búsqueda, se escoge uno o varios y se ejecuta la opción buscar, además se puede ejecutar la opción buscar todos, que muestra todas las áreas de salud existentes.

Flujo central

- ✓ El sistema muestra la interfaz que permite loguearse al usuario.
- ✓ Si el usuario se loguea correctamente con el login domingo y pass domingo 2007 el cual tiene permiso de editor nacional, el sistema muestra la interfaz principal de la aplicación el usuario ejecuta la opción RAS y el usuario elige en el menú principal Configurar Áreas de salud.

Condiciones de Ejecución

Tienen que estar presente en la BD los tipos de áreas de salud.

- ✓ Principal de Urgencia Lajas
- ✓ Área I I I Octavio de la Concepcion de la Pedraja
- ✓ Policlinico Área VI
- ✓ Área V (Piti Fajardo)
- ✓ Área V I I I (Fabio Di Celmo)
- ✓ Policlínico La Sierpe I.

Clases válidas	Clases inválidas	Resultado Esperado	Resultado de la prueba	Observaciones	Cumplimiento %
Mostrar todos		incipal de Urgencia ijas ea I I I Octavio de la oncepcion de la edraja oliclinico Área VI ea V (Piti Fajardo) ea V I I I (Fabio Di elmo) oliclínico La Sierpe I.	La prueba se realizó satisfactoriamente.	.	100%
	Código unidad: AAAAAAAAAAAAAA	Se introducen el código sin problemas	La prueba no se realizo correctamente	El label Código Unidad debe validar la entrada de letras y mostrar un cartel de alerta.	100%

Tabla 3.27 Resultados del CP Buscar área de salud

3.3.4.2 CPR2: Buscar Área de Salud (Nivel Área de Salud)

Descripción

Se ejecuta la opción configurar Área de salud, aparecen varios criterios de búsqueda, se escoge uno o varios y se ejecuta la opción buscar, además se puede ejecutar la opción buscar todos, que muestra todas las áreas de salud existentes.

Flujo Central

- ✓ El sistema muestra la interfaz que permite loguearse al usuario.
- ✓ Si el usuario se loguea correctamente con el login ypoza13 y pass yanet2007 el cual tiene permiso de editor Área de Salud, el sistema muestra la interfaz principal de la aplicación el usuario ejecuta la opción RAS y el usuario elige en el menú principal Configurar Áreas de salud.
- ✓ Se listan todas las áreas de salud a través de la opción mostrar todas o a través de los criterios de búsquedas y la opción buscar.
- ✓ Se muestran las unidades de salud correspondiente a la localidad que pertenece el usuario logeado.

Condiciones de ejecución

Policlínico La Sierpe I.

Clases válidas	Clases inválidas	Resultado Esperado	Resultado de la prueba	Observaciones	Cumplimiento %
Mostrar todas		✓ Policlínico La Sierpe I.	La prueba se realizó satisfactoriamente		100%
	Código de Área: AAAAAAAA	Se actualiza el código del área con los datos AAAAAAAA	La prueba no se realizó satisfactoriamente	No se validan los caracteres de entrada.	100%

3.3.4.3 CPR3: Editar Área de Salud

Tabla 3.28 Resultados del CP Buscar área de salud por el nivel de área de salud

Descripción

Este caso de uso consiste en probar los datos de entrada para editar un área de salud.

Flujo Central

- ✓ El sistema muestra la interfaz que permite loguearse al usuario.
- ✓ Si el usuario se loguea correctamente con el login ypoza13 y pass yanet2007 el cual tiene permiso de editor Unidad de Salud, el sistema muestra la interfaz principal de la aplicación el usuario ejecuta la opción RAS y el usuario elige en el menú principal Configurar Áreas de salud.
- ✓ Se listan todas las áreas de salud a través de la opción mostrar todas o a través de los criterios de búsquedas y la opción buscar.
- ✓ Se muestran las unidades de salud correspondiente a la localidad que pertenece el usuario logeado.
- ✓ Se muestra los datos generales.

Condiciones de ejecución

El sistema muestra los datos generales, los hospitales bases, las localidades, los departamentos y los servicios (el código de la unidad y el nombre no pueden ser modificados).

Clases válidas	Clases inválidas	Resultado Esperado	Resultado de la prueba	Observaciones	Cumplimiento %
	Código de la Unidad: AAAAAAAAA	Se introdujeron sin problemas los cambios en el Unidad de salud	La prueba no se ejecutó correctamente	No se valida la entrada de letras.	100%
Se ejecuta el link ayuda		Se muestra la ayuda correspondiente a la pantalla usada.	La prueba se ejecutó correctamente.	Ver documento Prueba A	100%

Tabla 3.29 Resultados del CP Editar área de salud por el nivel de Unidad de salud

3.3.4.4 Registro de Defectos y Dificultades Detectadas

Elemento	N°	Descripción de las No conformidades	Aspecto correspondiente	Etapa de la detección	Importante	Recomendaciones
Validación	1	No se validan el código de la unidad, pues entran caracteres extraños como letras.	Esto permite que a la BD entren caracteres no necesarios.	Una Iteración de Prueba	X	Revisar la clase de validaciones.
Validación	2	No se validan el código de la área, pues entran caracteres extraños como letras.	Esto permite que a la BD entren caracteres no necesarios.	Una Iteración de Prueba	X	Revisar la clase de validaciones.
Validación	3	No se validan el código de la unidad, pues entran caracteres extraños como letras.	Esto permite que a la BD entren caracteres no necesarios.	Una iteración de prueba	X	Revisar la clase de validaciones.

Tabla 3.30 Registro de defectos y dificultades del procedimiento de prueba configurar áreas de salud

3.4 Componente de prueba

El componente de prueba fue creado con el fin de ejecutar los caminos del grafo de flujo 1, 2, 3, 11 [Ver figura 2.4], del procedimiento de prueba Buscar Plantilla. [Ver Anexo D]

3.5 Resumen de defectos

Se muestra un resumen de los defectos encontrados como resultado de la ejecución de la actividad ejecutar pruebas del flujo de trabajo de prueba propuesto, al módulo RAS, del proyecto SISalud. [Ver tabla 3.31]

Error	Nombre Caso Prueba	Descripción
No valida caracteres extraños ni los caracteres tipo letra.	✓ Configurar Plantilla CP 2 ✓ Configurar Área de Salud CP 1 ✓ Configurar Área de Salud CP 2 ✓ Configurar Área de Salud CP 3	No se validan los caracteres extraños, ejemplo: @!#$%, ni los caracteres tipo letra ejemplo : AAAAAA, en los niveles, nacional, área de salud y unidad de salud a la hora de buscar o editar las áreas de salud.
El botón está desactivado y debe mostrar un mensaje de error indicando que se debe seleccionar un elemento.	✓ Editar Área de Salud CP4	
No muestra la ayuda al presionar la tecla F1	✓ Listar Plantilla CP5	Al presionar la tecla F-1 se muestra la ayuda del navegador y no la del sistema.
Al presionar el link de la ayuda se muestra la misma.	✓ Configurar Plantilla CP3 ✓ Listar Tipo de Estructura CP2 ✓ Listar Áreas de Salud CP2	Esta ayuda tiene varios defectos descritos en el documento de texto **Prueba A**.
No exporta a PDF	✓ Listar Tipo de Estructura CP2 ✓ Listar Áreas de Salud CP2	La opción exportar a PDF no funciona.
No exporta a XSL	✓ Listar Tipo de Estructura CP3 ✓ Listar Áreas de Salud CP3	La opción exportar a XSL no funciona

Tabla 3.31 Registro de defectos como resultado de la actividad ejecutar pruebas

3.6 Evaluación de las pruebas de la iteración

Los objetivos trazados en el Plan de prueba fueron cumplidos un 100%, vale destacar la presencia de errores de validación y de interfaz; específicamente en la ayuda. Reconociendo que no todas las pruebas cumplieron el propósito de encontrar la mayor cantidad posible de errores. Se propone crear otros procedimientos de pruebas para las futuras fases e iteraciones del ciclo de vida de un proyecto.

3.7 Aceptación de las pruebas de la iteración

Esta actividad se centra en la aceptación o no de las pruebas a nivel de proyecto por el líder de proyecto de la universidad teniendo en cuenta la evaluación de las mismas,

empezando entonces los procesos de gestión de cambios y/o liberación del grupo de proyecto de Softel.

3.8 Conclusiones

En el presente capítulo se ejecutó el flujo de trabajo de prueba propuesto, de forma práctica, en el proyecto SISalud, específicamente al módulo RAS, en la primera iteración de la fase de construcción del ciclo de desarrollo. Se elaboró el Plan de prueba para dicha iteración mediante el cual se definieron el alcance y los objetivos de las pruebas.

Se describieron y ejecutaron los procedimientos de prueba de forma manual y automática a través de un componente, para llegar a obtener una lista de defectos, los cuales validan el éxito de las pruebas. Se elaboró un script de datos de pruebas con el objetivo de llenar la BD con datos válidos para la ejecución de las pruebas, lo que sirvió como constancia a la hora de evaluar los resultados de salida.

Conclusiones

Mediante este trabajo, se ha logrado implementar un flujo de trabajo de prueba al módulo RAS. Facilitándole al grupo de desarrollo del Proyecto SISalud un procedimiento, guiado por actividades, artefactos y trabajadores bien definidos, que permitirá la detección y corrección de errores a lo largo del ciclo de vida del software. Durante su realización se arribaron a las siguientes conclusiones:

- ✓ A través del estudio, se detectaron las deficiencias en la planificación y ejecución de las pruebas del Proyecto SISalud, lo que constituyó un punto de partida importante para la investigación.
- ✓ Producto de la mala planificación de las pruebas, no se logró la detección de errores, en tiempo. Lo que ha provocado el atraso en la liberación de algunos módulos del proyecto. Evidenciando que no está definido un flujo de trabajo de prueba, por el cual se guíe el equipo que debe garantizar la calidad de los productos.
- ✓ Para la creación y ejecución de este flujo, se elaboró un marco teórico sobre las pruebas, donde se analizaron las estrategias, tipos y flujo de trabajo, propuesto por el Proceso Unificado de Desarrollo de Software y el propuesto por la versión más actualizada de RUP en soporte digital.
- ✓ El valor social de esta investigación se expresa en la contribución al mejoramiento de las condiciones de trabajo, desempeño y calidad de los servicios en el sector de la salud, al poder entregar un producto depurado de errores a la empresa Softel, con el fin de realizar pruebas finales al producto antes de ser liberado al cliente final.

Con el estudio realizado se cumple con el objetivo propuesto: ya que se ejecutó el proceso de pruebas al RAS del proyecto SISalud siguiendo el flujo de trabajo de prueba propuesto.

Recomendaciones

De forma general los objetivos propuestos al inicio de este trabajo fueron cumplidos. No obstante, durante el transcurso de su desarrollo, han surgido una serie de ideas y recomendaciones que podrían implementarse en futuras iteraciones. De manera que pueda lograrse un flujo de trabajo de prueba más útil, completo y efectivo, para lo cual se recomienda:

- ✓ Diseñar un plan de pruebas principal del proyecto, al que se haga referencia en los planes de cada iteración, donde se describan las estrategias de forma general para cada nivel de prueba y técnicas, etc.
- ✓ Extender el flujo de trabajo de pruebas al resto de los módulos del proyecto, con el fin de probarlo, evaluarlo e incorporarle mejoras.
- ✓ Para próximas iteraciones, profundizar en el trabajo con herramientas para la automatización de las pruebas.

Bibliografía

1. DRA. A. FEBLES, I. H. F. *Calidad de Software y la empresa, enseñanza de un tema imprescindible para el Ingeniero Informático*, 2005.
2. HUMPHREY, W. *Introducción al Proceso Software Personal*. Madrid, 2001. 328 p. 84-7829-052-4
3. BRITO, I. N. *Las pruebas de software, su aplicación al Config. CASE*. CEIS. Habana, ISPJAE, 2003. 121. p.
4. JACOBSON, G. B., J. RUMBAUGH. *El proceso unificado de desarrollo de software*. Quinta edición. Madrid, 2002. 464 p. 84-7829-036-2
5. K. GARCÍA, N. R. *Portal de Servicios Postales*. CEIS. Habana, ISPJAE, 2004. 145. p.
6. LÓPEZ, F. P. Casos de uso: una guía para la definición de los casos de prueba, 2004.
7. MSC. Y. FERNÁNDEZ , L. L. C., ING. M. GONZÁLEZ, ING. D. ACOSTA *Proceso de pruebas de aceptación para un software de gestión*, 2006. 10 p.
8. PRESSMAN, R. S. *Ingeniería del Software un enfoque práctico*. Quinta edición. Madrid, 1997. p.
9. RUP, C. *Ayuda del Rational*, 2003.
10. UCI, C. D. P. D. D. M. C. D. I. Y. G. D. S. D. L. U. D. L. C. I. *Casos de prueba*, 2006. 8 p.
11. ---. *Flujo de trabajo de prueba en la fase de elaboración*, http://docencia/, 2006. 17 p.
12. Beizer, B, Software Testing Techiniques, 1990
13. Davis, A, Principles of Software Development, 1995.
14. Kaner, C. Testing Computer Software, 1993.
15. McCabe, T. Software Complenxity Measure, 1976.
16. Myers, G. The Art of Software Testing 1979.
17. Phadke, M.S. Planing Efficient Software tests, 1997.
18. Howden, W.E, Weak Mutation Testing and the Completeness of test Cases, 1982.
19. Hetzel, W. The Complete Guide to software testing, !984.
20. Tai, K, C. Test Generation for Boolean expressions. 1987.

Glosario de Términos y Siglas

Actor

Alguien o algo, fuera del sistema o negocio que interactúa con el sistema o negocio.

Apache

Apache es programa de servidor HTTP Web de código abierto (open source). Fue desarrollado en 1995 y actualmente es uno de los servidores web más utilizados en la red. Usualmente corre en UNIX, Linux, BSD y Windows. Es un poderoso paquete de servidor web con muchos módulos que se le pueden agregar y que se consiguen gratuitamente en el Internet. Uno de sus competidores es Microsoft IIS.

Aplicación

Cualquier programa que corra en un sistema operativo y que haga una función específica para un usuario. Por ejemplo, procesadores de palabras, bases de datos, agendas electrónicas, etc.

Artefactos

Una parte de la información que (1) es producida, modificada, o usada por un proceso, (2) define un área de responsabilidad, y (3) está sujeta al control de versión. Un artefacto puede ser un modelo, un elemento del modelo, o un documento. Un documento puede adjuntar otros documentos. Una parte de la información que es usada o producida por un proceso de desarrollo del software. Un artefacto puede ser un modelo, una descripción, o un software.

BD

Conjunto de datos interrelacionados, almacenados con carácter más o menos permanente en la computadora, puede ser considerado una colección de datos variables en el tiempo.

Build

Pedazo de código construido.

Browser

(Hojeador, Navegador, Visor, Visualizador) Aplicación para visualizar documentos WWW y navegar por el espacio Internet. En su forma más básicas son aplicaciones hipertexto que facilitan la navegación por los servidores de información Internet; cuentan con funcionalidades plenamente multimedia que permite indistintamente la navegación por servidores WWW, FTP, Gopher, el acceso a grupos de noticias, la gestión del correo electrónico, etc.

Caso de prueba

Especificación de un caso para probar el sistema, incluyendo qué probar, con qué entrar y resultado bajo qué condiciones.

Caso de uso

Una descripción de un conjunto de secuencias de acciones, incluyendo variaciones, que un sistema lleva a cabo y conduce a un resultado observable de interés para un actor determinado.

Ciclo de vida del software

Ciclo que cubre cuatro fases en le siguiente orden: inicio, elaboración, construcción y transición.

Cliente

Una persona u organización, interna o externa a la organización productora que toma responsabilidad financiera por el sistema. El cliente es el último destinatario del producto desarrollado y sus artefactos.

Código Fuente

En ingles Source Code. Conjunto de instrucciones que componen un programa, escrito en cualquier lenguaje. Hay programas de código abierto que se pueden modificar si uno tiene el conocimiento (por lo general estos programas están bajo licencia GPL), por ejemplo Linux, Openoffice, etc. Hay programas “de código cerrado” como por ejemplo Windows, Photoshop, y la mayoría de los programas comerciales, en donde el código es inaccesible y por lo tanto no se puede alterar la estructura del programa. En estos casos uno compra el programa, el programa es de uno, pero el código fuente o instrucciones del programa son del fabricante.

Contraseña

Password. Código utilizado para acceder un sistema restringido. Pueden contener caracteres alfanuméricos e incluso algunos otros símbolos. Se destaca que la contraseña no es visible en la pantalla al momento de ser tecleada con el propósito de que sólo pueda ser conocida por el usuario.

Defecto

Anomalía del sistema, por ejemplo un síntoma de error en el software descubierto durante las pruebas, o un problema descubierto durante una reunión de revisión.

Empresa

En términos estrictamente económico, es una unidad económica que reúne una serie de factores de producción: recursos naturales, humanos, tecnológicos (o de capital) y financieros (que posibilitan la adquisición de los anteriores), y los utiliza para producir

bienes y/o servicios, que vende a personas individuales, a otras empresas y/ o a las administraciones públicas.

Fase

Periodo de tiempo entre dos hitos principales de un proceso de desarrollo.

Flujo de trabajo

Realización de un caso de uso de negocio parte de él. Puede describirse en términos de diagrama de actividad, que incluye a los trabajadores participantes, las actividades que realizan y los artefactos que producen.

Gigabytes

Unidad de medida de una memoria. 1 gigabyte = 1024 Megabytes = 1.073.741.824 bytes.

Hipervínculo

Vínculo existente en un documento hipertexto que apunta o enlaza a otro documento que puede ser o no otro documento hipertexto.

HTTP

HTTP o *HiperText Transfer Protocol* (protocolo de transferencia de hipertexto) Es el grupo de reglas, o protocolos, que gobiernan la transferencia de hipertexto entre dos o más computadoras. Es muy cómodo y fácil de usar para transferir texto, imágenes, sonido, etc.

Interfaz

Frontera convencional entre dos sistemas o dos unidades, que permite intercambio de informaciones.

IIS

Microsoft Internet Information Services. Servicios de Información de Internet de Microsoft. IIS es un conjunto de servicios basados en Internet, para máquinas con Windows. Originalmente se proporcionaba como opcional en Windows NT, pero posteriormente fue integrado a Windows 2000 y Windows Server 2003. Incluye servidores para FTP, SMTP, NNTP y HTTP/HTTPS. Compite con Apache en el área de servidores web.

Internet Explorer

Conocido también como IE es el browser web de Microsoft, creado en 1995 para Windows y mucho después para Mac. No fue el primero en el mercado y Netscape le sacó la delantera por muchos años, pero la penetración de Windows en el mercado es muy fuerte. Microsoft empezó a distribuir Windows junto con IE. Poco a poco las personas simplemente preferían usar lo que venía en la computadora a tener que

descargar una aplicación de gran tamaño como era Netscape. En la actualidad navegadores como Firefox están ganando terreno.

Iteración

Conjunto de actividades llevadas a cabo de acuerdo a un plan (de iteración) y unos criterios de evaluación, que lleva a producir una versión, ya sea interna o externa.

MySQL

MySQL es uno de los Sistemas Gestores de Bases de Datos más populares. Su ingeniosa arquitectura lo hace extremadamente rápido y fácil de personalizar. La extensiva reutilización del código dentro del software y una aproximación minimalística para producir características altamente funcionales, ha dado lugar a un sistema de administración de base de datos de alta velocidad, compactación, estabilidad y facilidad de despliegue. La exclusiva separación del core server del manejador de tablas, permite funcionar a MyQSL bajo control estricto de transacciones o con acceso a disco no transaccional ultrarrápido. Lo mantiene la empresa sueca MySQL bajo la licencia GPL (GNU Public License); puede utilizarse gratuitamente y su código fuente está disponible.

Mega Byte

Agrupación de 1024 Bytes.

Navegador

Un navegador es una aplicación cliente de software para Internet que sirve como interface para navegar a través del mundo de información de la web.

PC

Computadora Personal

PDF

(*Portable Document Format*) es un formato de archivo que captura todos los elementos de un documento impreso como por ejemplo una imagen electrónica la cual se puede ver, navegar, imprimir, o enviar a otra persona. Los archivos pdf se crean usando el Adobe Acrobat, Acrobat Capture, o productos similares. Para visiualizar y utilizar estos ficheros, se necesita el Acrobat Reader.

PHP

Hypertext Preprocessor. Lenguaje de script diseñado para la creación de páginas web activas (similares a ".asp" de Microsoft), muy popular en Linux, aunque existe también versión para sistemas Microsoft. Concebido en el tercer trimestre de 1994 por Rasmus Lerdorf, es usado principalmente para la programación de CGIs para páginas web, destaca por su capacidad de ser embebido en el código HTML.

Plan de pruebas

Plan que describe las estrategias, recursos y programación de las pruebas.

Proceso

Secuencia de actividades invocadas para producir un producto de software.

Procedimiento de pruebas

Especificación de cómo llevar a cabo uno o varios casos de prueba o parte de ellos.

Pruebas

Flujo de trabajo fundamental cuyo propósito esencial es comprobar el resultado de la implementación mediante pruebas de cada construcción, incluyendo tanto construcciones internas como intermedias, así como las versiones finales del sistema que van a ser entregadas a terceras partes.

Proyecto

Esfuerzo de desarrollo para llevar un sistema a lo largo del ciclo de vida.

Requerimiento

Condición o capacidad que debe cumplir un sistema.

RAM

'Random Access Memory'. La RAM se usa para mantener los programas mientras se están ejecutando, y los datos mientras se los procesa. La RAM es volátil, lo que significa que la información escrita en la RAM desaparecerá cuando se apague la alimentación de energía del ordenador.

RAS

Registro de Áreas de Salud

RIS

Registro Informatizado de Salud

Rol

Papel, cometido o función que tiene o desempeña que interpreta un actor.

RUP

El Proceso Unificado Rational (RUP) es una metodología de desarrollo para la programación orientada a objetos. Según Rational (diseñadores de Rose Rational y el Idioma Modelado Unificado), RUP está como un mentor en línea que mantiene pautas, plantillas, y ejemplos de todos los aspectos y fases de desarrollo del programa. RUP y los productos similares—como el Proceso del Software Objeto-orientado (OOSP), y el Proceso ABIERTO—es software comprensivo que diseña herramientas que combinan los aspectos procesales de desarrollo (como las fases definidas, técnicas, y prácticas) con otros componentes de desarrollo (como los documentos, modelos, manuales, el código, y así sucesivamente) dentro de un armazón unificándose.

SAAA

Componente de seguridad Modelo de Autenticación, Autorización y Auditoria (AAA).

Servidor

Es un computador potente o un software que provee una clase especial de servicio a los software clientes que están corriendo en otros computadores y que lo accedan para realizar una función determinada. Un computador funcionando como servidor puede tener operando varios software servidores para prestar servicios, por ejemplo: servidor de www, servidor de FTP, de Mail, etc.

SISalud

Sistema de información para la Salud.

Script

(Guión) Conjunto de caracteres formados por mandatos y secuencias de tecleo, que se utiliza muy a menudo en Internet para automatizar tareas muy habituales como, por ejemplo: la conexión a la red (login).

SGBD

Es el software que permite la utilización y/o la actualización de los datos almacenados en una (o varias) base(s) de datos por uno o varios usuarios desde diferentes puntos de vista y a la vez.

SOAP

SOAP es un protocolo elaborado para facilitar la llamada remota de funciones a través de Internet, permitiendo que dos programas se comuniquen de una manera muy similar técnicamente a la invocación de páginas Web. El protocolo SOAP tiene diversas ventajas sobre otras maneras de llamar funciones de manera remota como DCOM, CORBA o directamente en TCP/IP.

Software

Palabra en inglés utilizada para indicar a los programas de computadoras, a las aplicaciones.

Stakeholder

Involucrados en el proceso de desarrollo de software, especialistas en el campo que se informatiza.

UML

“Unified Modeling Language” Lenguaje gráfico que brinda un vocabulario y reglas para especificar, construir, visualizar y documentar los artefactos de un sistema utilizando el enfoque orientado a objetos.

Usuario

Persona que usa ordinariamente una cosa.

Web Services

Servicios Web, aplicaciones web que permiten ser usadas desde cualquier aplicación web.

XML

Lenguaje extensible de marcado (*eXtensible Markup Language*), con un formato basado en el etiquetado textual para documentos y datos. Proviene del SGML (*Standard Generalized Mark-up Language*). Fue aceptado por la *World Wide Web* (W3C) en Febrero de 1998. XML no es realmente un nuevo lenguaje, es un metalenguaje usado para definir a otros lenguajes, actuando como protocolo integrador entre aplicaciones. Permite crear documentos bien estructurados y como resultado todo lenguaje basado en XML también será bien estructurado, lo que significa que los datos en XML son más fáciles de usar; así XML es una completamente nueva manera de comunicarse a través de Internet, por que permite a los negocios y sus sistemas de computadoras comunicarse más fácilmente.

Anexos

Anexo A: Guía para Revisar el Código

1 Criterios de entrabada para la revisión.

- ✓ Especificación de los requisitos**.**
- ✓ El diseño del programa.
- ✓ El código fuente del programa.
- ✓ Estándares de codificación.

1.1 Procedimiento de Revisión.

- ✓ Primero escribe el código fuente del programa completo.
- ✓ Antes de compilar o probar el programa, de ser posible imprime un listado del código fuente.
- ✓ Después procedes a hacer una revisión del código.
- ✓ Durante la revisión del código chequea cuidadosamente cada línea de código fuente para encontrar y corregir tantos defectos como puedas.

1.2 Corregir todos los defectos encontrados.

- ✓ Corregir todos los defectos encontrados.
- ✓ Comprobar las correcciones para asegurar que son correctas.
- ✓ Llevar un registro de los defectos encontrados.

1.3 Revisar el Ámbito.

- ✓ Verificar que el diseño del programa satisface todas las funciones descritas en la especificación.
- ✓ Verifica que el código fuente implementa todo el diseño.

1.4 Revisar la lógica del programa.

- ✓ Verificar que el diseño lógico del programa es correcto.
- ✓ Verificar que el programa implementa correctamente el diseño lógico.

1.5 Comprobar los nombres y los Tipos.

- ✓ Verificar que todos los nombres y los tipos son correctamente declarados y utilizados.
- ✓ Chequea la correcta declaración de los Tipos.

1.6 Comprobar todas la Variables.

- ✓ Asegúrate de que cada variable esta inicializada.
- ✓ Chequea los problemas de desbordamiento o de fuera de rango.

1.7 Comprobar la sintaxis del programa.

- ✓ Verificar que el código fuente cumpla con todas las especificaciones del lenguaje.

1.8 Criterios de Salidas.

Al finalizar debe tener:

- ✓ El código fuente terminado y corregido.
- ✓ Un registro de defectos completo.

Anexo B: Plantilla de plan de prueba

<Versión 1.0>

<Nombre Proyecto>

Historia de revisiones

Fecha	Versión	Descripción	Autor
	1.0		

Contenido

Introducción

[Se explica brevemente como se aplicará el plan de prueba sobre el componente]

Propósito

[Se plantea el objetivo del plan de prueba que se va a ejecutar]

Punto de partida.

[Descripción del Componente a probar]

1.3 **Alcance**

[Se propone por etapas para la realización de las pruebas según el nivel en que se vayan a desarrollar y los objetivos de la iteración.]

Pruebas de Unidad: Consiste en hacer pruebas estructurales, utilizando técnicas de caja Blanca.

Pruebas de Integración: Se prueban los diferentes componentes como un todo, en esta etapa se pueden Aplicar pruebas de caja Blanca y en algunos casos se utiliza la caja Negra.

Pruebas de Sistema: Consiste en realizarle pruebas al sistema cuando ya esta integrado y consiste en probar los requisitos funcionales definidos por el cliente, en este nivel se realizan principalmente pruebas por el método de prueba de caja negra.

Pruebas de aceptación: Consiste en las denominada pruebas alfa o beta.

Identificación del proyecto

[Se numeran los artefactos que serán utilizados por ejemplo:

Las descripciones de los casos de uso, documento visión, etc.]

Estrategia de evolución del Plan

[Se detallan las actividades a realizar para ir chequeando la evolución del plan y la peridicidad de cada una.

Además se relacionan los representantes de cada una de las partes que tendrán la responsabilidad de aprobar el Plan de Pruebas]

Es decir este acápite debe contener:

Quien es responsable de monitorear el Plan de prueba.

Con cuanta frecuencia se realizarán modificaciones al Plan.

Como serán evaluados y aprobados los cambios al Plan.

Como serán realizados y comunicados los cambios al Plan.]

Requerimientos para probar

[En la lista a continuación se presentan los elementos, casos de uso, requerimientos funcionales y requerimientos no funcionales, que serán probados.

[Lista de los requerimientos más importantes a ser verificados.]

Estrategia de Pruebas

[Esta sección presenta el enfoque recomendado para la verificación. Describe como se verificarán los elementos.

Para cada tipo de prueba, proporcione una descripción de la prueba y por qué será implementada y ejecutada.

Se indicarán las técnicas usadas y el criterio para saber cuando una prueba se completó (criterio de aceptación).

Las pruebas se deben ejecutar usando bases de datos conocidas y controladas en un ambiente seguro.]

Tipos de pruebas

[Se deben diseñar pruebas de los siguientes tipos, donde se especifique: técnica, criterio de aceptación y consideraciones especiales.]

1. **Prueba de integridad de los datos y la base de datos**

Objetivo de la prueba, técnica, criterio de aceptación y consideraciones especiales

[Asegurar que los métodos y procesos de acceso a la base de datos funcionan correctamente y sin corromper datos.]

Técnica

[Invoque cada método o proceso de acceso a la base de datos con datos válidos y no válidos.]

[Inspeccione la base de datos para asegurarse de que se han guardado los datos correctos, que todos los eventos de la base de datos ocurrieron correctamente, o repase los datos devueltos para asegurar que se recuperaron datos correctos por la vía correcta.]

Criterio de aceptación

[Todos los métodos y procesos de acceso a la base de datos funcionan como fueron diseñados y sin datos corruptos.]

Consideraciones especiales

[La prueba requiere un entorno de administración de DBMS o controladores para ingresar o modificar información directamente en la base de datos.

Se deben usar bases de datos pequeñas para aumentar la facilidad de inspección de los datos para verificar que no sucedan eventos no aceptables.]

2. **Prueba de Funcionalidad**

[La prueba de funcionalidad se enfoca en requerimientos para verificar que se corresponden directamente a casos de usos o funciones y reglas del negocio. Los objetivos de estas pruebas son verificar la aceptación de los datos, el proceso, la recuperación y la implementación correcta de las reglas del negocio. Este tipo de prueba se basa en técnicas de caja negra, que consisten en verificar la aplicación y sus procesos interactuando con la aplicación por medio de la interfase de usuario y analizar los resultados obtenidos.]

Objetivo de la prueba, técnica, criterio de aceptación y consideraciones especiales

[Asegurar la funcionalidad apropiada del objeto de prueba, incluyendo la navegación, entrada de datos, proceso y recuperación.]

Técnica

[Ejecute cada caso de uso, flujo de caso de uso, o función usando datos válidos y no válidos, para verificar lo siguiente:

Se obtienen los resultados esperados cuando se usan datos válidos.

Cuando se usan datos no válidos se despliegan los mensajes de error o advertencia apropiados.

Se aplica apropiadamente cada regla del negocio.]

Criterio de aceptación

[Todas las pruebas planificadas se realizaron. Todos los defectos encontrados han sido debidamente identificados.]

Consideraciones especiales

[Identificar o describir aquellos elementos o problemas (internos o externos) que impactaron en la implementación y ejecución de las pruebas de funcionalidad.]

3. Prueba de Interfase de Usuario

[Esta prueba verifica que la interfase de usuario proporcione al usuario el acceso y navegación a través de las funciones apropiada. Además asegura que los objetos presentes en la interfase de usuario se muestren como se espera y conforme a los estándares establecidos por la empresa o de la industria.]

Objetivo de la prueba, técnica, criterio de aceptación y consideraciones especiales

[Verificar que: la navegación a través de los elementos que se están probando reflejen las funciones del negocio y los requerimientos, incluyendo manejo de ventanas, campos y métodos de acceso; los objetos de las ventanas y características, como menúes, tamaño, posición, estado funcionen de acuerdo a los estándares.]

Técnica

[Crear o modificar pruebas para cada ventana verificando la navegación y los estados de los objetos para cada ventana de la aplicación y cada objeto dentro de la ventana.]

Criterio de aceptación

[Cada ventana ha sido verificada exitosamente siendo consistente con una versión de referencia o estándar establecido.]

Consideraciones especiales

[No todas las propiedades de los objetos se pueden acceder.]

4. Prueba de Performance

[En esta prueba se miden y evalúan los tiempos de respuesta, los tiempos de transacción y otros requerimientos sensitivos al tiempo. El objetivo de la prueba es verificar que se logren los requerimientos de performance. La prueba de performance es implementada y ejecutada para poner a punto los destinos de pruebas de performance como función de condiciones de trabajo o configuraciones de hardware.]

Objetivo de la prueba, técnica, criterio de aceptación y consideraciones especiales

[Verificar la performance de determinadas transacciones o funciones de negocio bajo ciertas condiciones:

condiciones de trabajo normales conocidas.

peores casos de condiciones de trabajo conocidas.]

Técnica

[Usar procedimientos de prueba desarrollados para verificar funciones o ciclos de negocio.

Modificar archivos de datos para aumentar el número de transacciones o los procedimientos de prueba para aumentar el número de iteraciones de ocurrencia de transacciones.

Las pruebas se deben ejecutar en una máquina (mejor caso de prueba un solo usuario, una sola transacción) y se debe repetir con múltiples usuarios (virtuales o reales).]

Criterio de aceptación

[Con una transacción o un usuario: Éxito completo de la prueba sin fallas y dentro del tiempo esperado o requerido.

Con múltiples transacciones y varios usuarios: Éxito completo de la prueba sin fallas y dentro de un tiempo aceptable.]

Consideraciones especiales

[Las pruebas de performance deben incluir un trabajo de fondo en el servidor. Esto se puede realizar de distintas formas:

Enviar transacciones directamente al servidor, generalmente en la forma de consultas (SQL).

Crear usuarios virtuales para simular muchos clientes, generalmente varios cientos. Se pueden usar herramientas de Emulación de Terminar Remota para lograr este objetivo. Esta técnica también se usa para cargar la red con “trafico”.

Usar muchos clientes físicos, cada uno corriendo procedimientos de prueba.

La prueba de performance se debe realizar en una máquina dedicada para permitir control total y medición exacta.

Las bases de datos usadas para las pruebas de performance deben tener un tamaño similar a las reales.]

5. Prueba de Carga

[La prueba de carga somete los objetos a verificar a diferentes cargas de trabajo para medir y evaluar los comportamientos de performance y la habilidad de los objetos de continuar funcionando apropiadamente bajo diferentes cargas de trabajo. El objetivo es determinar y asegurar que el sistema funciona apropiadamente en circunstancias de máxima carga de trabajo esperada. Además evaluar las características de performance, como tiempos de respuesta, tiempos de transacciones y otros elementos sensitivos al tiempo.]

Objetivo de la prueba, técnica, criterio de aceptación y consideraciones especiales

[Verificar el comportamiento de performance de determinados componentes del software bajo condiciones de trabajo diferentes.]

Técnica

[Usar pruebas desarrolladas para funciones o ciclos de negocios y modificar archivos de datos para aumentar el número de transacciones o las pruebas para aumentar la cantidad de ocurrencia de transacciones.]

Criterio de aceptación

[Para múltiples transacciones y múltiples usuarios: Realización exitosa de las pruebas sin fallas y dentro del tiempo aceptable.]

Consideraciones especiales

[La prueba de carga debe realizarse en una máquina dedicada para tener control total y exactitud de mediciones.

Las bases de datos usadas para la prueba deben tener un tamaño similar a las reales.]

6. **Prueba de Esfuerzo (stress, competencia por recursos, bajos recursos)**

[La prueba de esfuerzo en un tipo de prueba de performance implementada y ejecutada para encontrar errores cuando hay pocos recursos o cuando hay competencia por recursos. Poca memoria o poco espacio de disco pueden revelar fallas en el software que no aparecen bajo condiciones normales de cantidad de recursos. Otras fallas pueden resultar al competir por recursos compartidos como bloqueos de bases de datos o ancho de banda de red. La prueba de esfuerzo también puede usarse para identificar el trabajo máximo que el software puede manejar.]

Objetivo de la prueba, técnica, criterio de aceptación y consideraciones especiales

[Verificar que el software funciona apropiadamente y sin error bajo condiciones de esfuerzo, como son:

- poca memoria o sin disponibilidad de memoria en el servidor
- cantidad máxima de clientes conectados
- múltiples usuarios realizando la misma operación sobre los mismos datos
- peor caso de volumen de operaciones.

El objetivo de la prueba de esfuerzo es también identificar y documentar las condiciones bajo las cuales el sistema falla y no continua funcionando apropiadamente.]

Técnica

[Usar las pruebas desarrolladas para Performance y Prueba de Carga.

Para probar recursos limitados, las pruebas se deben ejecutar en una sola máquina, y se debe reducir o limitar la memoria en el servidor.

Para las pruebas de esfuerzo restantes, deber usarse múltiples clientes, cualquiera que ejecute las mismas pruebas o pruebas complementarias para producir el peor caso de volumen de operaciones.]

Criterio de aceptación

[Todas las pruebas planeadas se ejecutaron y se alcanzaron o excedieron los límites del sistema sin que el software fallara o las condiciones bajo las que ocurre una falla en el software están fuera de las condiciones especificadas.]

Consideraciones especiales

[Las pruebas de esfuerzo de red pueden requerir herramientas de red para cargar la red con mensajes o paquetes.

La cantidad de disco del servidor usada por el sistema debe ser reducida temporalmente para restringir el espacio disponible para crecimiento de la base de datos.

Sincronizar el acceso simultáneo de varios clientes accediendo a los mismos datos.]

7. Prueba de Volumen

[La Prueba de Volumen somete el software a grandes cantidades de datos para determinar si se alcanzan límites que causen la falla del software. La Prueba de Volumen identifica la carga máxima continua que puede manejar el software a prueba en un período dado.]

Objetivo de la prueba, técnica, criterio de aceptación y consideraciones especiales

[Verificar que el software funciona correctamente con volúmenes de datos grandes:

Máximo (real o físicamente posible) número de clientes conectados, o simulados, todos realizando la misma operación (peor caso de operación) por un período de tiempo extenso.

Máximo tamaño de base de datos y múltiples consultas ejecutadas simultáneamente.]

Técnica

[Usar pruebas desarrolladas para Prueba de Performance y Prueba de Carga.

Se deben usar múltiples clientes, ejecutando las mismas pruebas o pruebas complementarias para producir el peor caso de volumen de operaciones o mezcla en un período de tiempo extenso.

Se debe crear el tamaño máximo de base de datos (real, escalado o con datos representativos) y múltiples clientes ejecutando consultas simultáneamente por un período de tiempo extenso.]

Criterio de aceptación

[Todas las pruebas planificadas se ejecutaron y se han alcanzado o excedido los límites especificados sin que el software falle.]

Consideraciones especiales

[¿Qué período de tiempo se considera aceptable para condiciones de gran volumen?]

8. Prueba de Seguridad y Control de Acceso

[La Prueba de Seguridad y Control de Acceso se enfoca en dos áreas de seguridad:

Seguridad en el ámbito de aplicación, incluyendo el acceso a los datos y a las funciones de negocios.

Seguridad en el ámbito de sistema, incluyendo conexión, o acceso remoto al sistema.

La seguridad en el ámbito de aplicación asegura que, basado en la seguridad deseada los actores están restringidos a funciones o casos de uso específicos o limitados en los datos que están disponibles para ellos.

La seguridad en el ámbito de sistema asegura que, solo los usuarios con derecho a acceder al sistema son capaces de acceder a las aplicaciones y solo a través de los puntos de ingresos apropiados.]

Objetivo de la prueba, técnica, criterio de aceptación y consideraciones especiales

Seguridad en el ámbito de aplicación:

[Verificar que un actor pueda acceder solo a las funciones o datos para los cuales su tipo de usuario tiene permiso.]

Seguridad en el ámbito de sistema: [Verificar que solo los actores con acceso al sistema y a las aplicaciones, puedan acceder a ellos.]

Técnica

Seguridad en el ámbito de aplicación: [Identificar y hacer una lista de cada tipo de usuario y las funciones y datos sobre las que cada tipo tiene permiso.]

[Crear pruebas para cada tipo de usuario y verificar cada permiso creando operaciones específicas para cada tipo de usuario.]

[Modificar el tipo de usuario y volver a ejecutar las pruebas para los mismos usuarios. En cada caso, verificar que las funciones o datos adicionales están correctamente disponibles o son denegados.

Acceso en el ámbito de sistema: [Ver consideraciones especiales más abajo.]

Criterio de aceptación

[Para cada tipo de actor conocido las funciones y datos apropiados están disponibles, y todas las operaciones funcionan como se espera y ejecutan las pruebas de Funcionalidad de la aplicación.]

Consideraciones especiales

[El acceso al sistema debe ser discutido con el administrador del sistema o la red. Esta prueba no puede requerirse como tal, es una función del administrador del sistema o de la red.]

9. Prueba de Fallas y Recuperación

[Las Pruebas de Fallas y Recuperación aseguran que el software puede recuperarse de fallas de hardware, software o mal funcionamiento de la red sin pérdida de datos o de integridad de los datos.

La Prueba de Recuperación es un proceso en el cual la aplicación o sistema se expone a condiciones extremas, o condiciones simuladas, para causar falla, como fallas en dispositivos de Entrada/Salida o punteros a la base de datos inválidos. Los procedimientos de recuperación se invocan y la aplicación o sistema es monitoreado e inspeccionado para verificar que se recupera apropiadamente la aplicación o sistema y se logre la recuperación de datos.]

Objetivo de la prueba, técnica, criterio de aceptación y consideraciones especiales

[Verificar que los procesos de recuperación (manual o automáticos) recuperen apropiadamente la base de datos, aplicaciones y sistema a un estado conocido y deseado. En la prueba se incluyen los siguientes tipos de condiciones:

- interrupción de energía al cliente
- interrupción de energía al servidor
- interrupción de comunicaciones mediante los servidores de la red
- interrupción de comunicación o pérdida de energía de los discos del servidor o con los controladores
- ciclos incompletos (procesos de filtro de datos interrumpidos, procesos de sincronización de datos interrumpidos)
- punteros a la base de datos o claves inválidos
- elementos de datos en la base de datos inválidos o corruptos.]

Técnica

[Se deben usar las pruebas creadas para probar Funcionalidad y Ciclos de negocio para crear una serie de operaciones. Una vez logrado el punto de comienzo deseado, se deben realizar o simular las siguientes acciones, individualmente:

Interrumpir la energía del cliente: apagar el PC.

Interrumpir la energía del servidor: simular o iniciar el proceso de apagado del servidor.

Interrupción por medio de los servidores de red: simular o iniciar la pérdida de comunicación con la red (desconectar físicamente la comunicación o apagar el servidor de red o router

Interrumpir la comunicación o quitar la energía de los discos del servidor o sus controladores: simular o eliminar físicamente al comunicación con uno o más controladores de disco o los discos.)

Una vez que se lograron o simularon estas condiciones, se deben invocar los procedimientos de recuperación.

Las pruebas de ciclos incompletos utilizan la misma técnica excepto que los procesos de bases de datos deben ser abortados a sí mismos o terminados prematuramente.

Las últimas dos pruebas requieren que se logre un estado conocido de la base de datos. Se deben corromper manualmente campos de la base de datos, punteros y claves trabajando directamente sobre la base de datos (utilizando herramientas para la base de datos). Se deben ejecutar las pruebas de Funcionalidad y Ciclo de negocio y verificar que los ciclos se completen.]

Criterio de aceptación

[En todos los casos, la aplicación, la base de datos y el sistema deben, en la realización procedimientos de recuperación, volver a un estado conocido y deseable. Este estado incluye corrupción de datos limitada al los campos, punteros o claves corruptos conocidos, y reportes indicando los procesos u operaciones que no se completaron debido a las interrupciones.]

Consideraciones especiales

[Los procedimientos para desconectar cables (simulando falta de energía o pérdida de comunicación) no son deseables o factibles. Se pueden requerir métodos alternativos, como software de diagnóstico. Se requieren los grupos de recursos de Sistemas, Bases de datos y Red.

Estas pruebas deben ejecutarse fuera del horario de trabajo normal o en una máquina aislada.]

10. Prueba de Configuración

[La Prueba de Configuración verifica el funcionamiento del software con diferentes configuraciones de software y hardware.]

Objetivo de la prueba, técnica, criterio de aceptación y consideraciones especiales

[Verificar que el software funcione apropiadamente en las configuraciones requeridas de hardware y software.]

Técnica

[Usar las pruebas de Funcionalidad.

Abrir y cerrar varias sesiones de software que no son objeto de prueba, como parte de la prueba o antes de comenzar la prueba.

Ejecutar operaciones seleccionadas para simular la interacción del actor con el software objeto de prueba y con el software que no es objeto de prueba.

Repetir los procedimientos anteriores minimizando la memoria convencional disponible en la máquina cliente.]

Criterio de aceptación

[Por cada combinación de software objeto de prueba y software que no es objeto de prueba, todas las operaciones son completadas exitosamente sin fallas.]

Consideraciones especiales

[Todo el software que no es objeto de prueba que es necesario y debe estar accesible.

¿Qué aplicaciones se usan normalmente?

¿Qué información se maneja en las aplicaciones que se usan normalmente, y que tamaño de información?

Los sistemas, red, servidores de red, bases de datos, etc., deben ser documentados como parte de esta prueba.]

11. Prueba de Instalación

[La Prueba de Instalación tiene dos propósitos. Uno es asegurar que el software puede ser instalado en diferentes condiciones (como una nueva instalación, una actualización, y una instalación completa o personalizada) bajo condiciones normales y anormales. Condiciones anormales pueden ser insuficiente espacio en disco, falta de privilegios para crear directorios, etc. El otro propósito es verificar que, una vez instalado, el software opera correctamente. Esto significa normalmente ejecutar un conjunto de pruebas que fueron desarrolladas para Prueba de Funcionalidad.]

Objetivo de la prueba, técnica, criterio de aceptación y consideraciones especiales

[Verificar que el software objeto de prueba se instala correctamente en cada configuración de hardware requerida bajo las siguientes condiciones:

- Instalación nueva, una nueva máquina, nunca instalada previamente con [Nombre del proyecto]
- Actualización, máquina previamente instalada con [Nombre del proyecto], con la misma versión
- actualización, máquina previamente instalada con [Nombre del proyecto], con una versión anterior.]

Técnica

[Manualmente o desarrollando programas, para validar la condición de la máquina destino (nueva, nunca instalado, misma versión, versión anterior ya instalada).

Realizar la instalación.

Ejecutar un conjunto de pruebas funcionales ya implementadas para la Prueba de Funcionalidad.]

Criterio de aceptación

[Las pruebas de funcionalidad de [Nombre de proyecto] se ejecutan exitosamente sin fallas.]

Consideraciones especiales

[¿Qué operaciones se deben ser seleccionar para realizar una prueba confiable de que la aplicación [Nombre del proyecto] ha sido exitosamente instalada sin dejar fuera ningún componente importante?]

12. Prueba de Documentos

[La Prueba de Documentos debe asegurar que los documentos relacionados al software que se generen en el proceso sean correctos, consistentes y entendible. Se incluyen como documentos los Materiales para Soporte al Usuario, Documentación Técnica, Ayuda en Línea y todo tipo de documento que forme parte del paquete de software.]

Objetivo de la prueba, técnica, criterio de aceptación y consideraciones especiales

[Verificar que el documento objeto de prueba sea:

Correcto, esto es, que cumpla con el formato y organización para el documento establecido en el proyecto.

Consistente, esto es, que el contenido del documento sea fiel a lo que hace referencia. Si el documento es Documentación de Usuario, que la explicación de un procedimiento sea exactamente como se realiza el procedimiento en el software, si se muestran pantallas que sean las correctas.

Entendible, esto es, que al leer el documento se entienda correctamente lo que expresa y sin ambigüedades, además que sea fácil de leer.]

Técnica

[Para verificar que el documento es correcto se debe comparar con el estándar definido si existe o con las pautas de documentación y ver que el documento cumple con ellas.

Para verificar que el documento es Consistente se debe ejecutar el programa siguiendo el documento en caso de los Materiales de Soporte al Usuario y comprobar que lo que se explica en estos documentos es exactamente lo que se ejecuta en el programa. En caso de Documentación Técnica se debe revisar el código al cual corresponde la documentación y comprobar que dicha describe el código.

Para verificar que el documento es entendible, debe comprobar que se entiende correctamente, que no tiene ambigüedades y que sea fácil de leer.]

Criterio de aceptación

[El documento expresa exactamente lo que debe expresar, no hay diferencias entre lo que está escrito y el objeto de la descripción (operación de software, código de programa, decisiones técnicas) y se entiende fácilmente.]

Consideraciones especiales

[Enumere las consideraciones que considere importantes para la verificación de documentos]

Herramientas

[Ingrese una lista con las herramientas usadas en el proyecto, como son herramientas de gestión de sistema de bases de datos (DBMS), herramientas para gestión de proyecto, seguimiento de errores, monitoreo de cubrimiento de pruebas, etc. Para cada herramienta indique la tarea para la que se usa, el nombre de la herramienta, el origen (vendedor o software hecho en la empresa) y la versión.]

Recursos

[En esta sección se presentan los recursos recomendados para el proyecto [Nombre de proyecto], sus principales responsabilidades y su conocimiento o habilidades.]

Roles

En la tabla a continuación se muestra la composición de personal para el proyecto [Nombre del proyecto] en el área Verificación del Software.

Rol	Cantidad mínima de recursos recomendada	Responsabilidades
Responsable de verificación		Identifica, prioriza e implementa los casos de prueba. Genera el Plan de Verificación. Genera el Modelo de Prueba. Evalúa el esfuerzo necesario para verificar. Proporciona la dirección técnica. Adquiere los recursos apropiados. Proporciona informes sobre la verificación.
Asistente de verificación		Ejecuta las pruebas Registra los resultados de las pruebas. Recuperar el software de errores. Documenta los pedidos de cambio.
Administrador de Base de Datos		Realiza la gestión y mantenimiento del entorno de los datos (base de datos) de prueba y los recursos. Administra la base de datos de prueba.

Sistema

En la siguiente tabla se establecen los recursos de sistema necesarios para realizar la verificación.

[Es recomendable que el sistema simule el entorno de producción, reduciendo los accesos y los tamaños de bases de datos si fuera apropiado.]

[Borre o agregue elementos a la lista]

Recurso	Nombre/Tipo
Servidor de base de datos	
Red o subred	
Nombre del servidor	
Nombre de la base de datos	
PC Cliente para pruebas	
Requerimientos especiales	
Repositorio de pruebas	
Red o subred	
Nombre del servidor	

Hitos del proyecto de Verificación

[La verificación del [Nombre de proyecto] debe incorporar actividades de prueba para cada verificación identificada en las secciones anteriores. Se deben identificar los hitos del proyecto de verificación separados para comunicar los logros de estado de proyecto.]

Actividad que determina el hito	Esfuerzo	Fecha de comienzo	Fecha de finalización
Planificar la verificación			
Elaborar casos de prueba			
Ajuste y Control de Verificación			
Ejecutar la verificación			
Evaluar la verificación			

[Las últimas tres actividades se repiten en cada iteración. Debería incluir en esta tabla los datos de todas las iteraciones del proyecto.]

Entregables

[En esta sección enumere los documentos, herramientas e informes que se crearán, por quien, para quien y cuándo serán liberados.

Para cada entregable deberá indicar las fechas en que son liberadas todas las versiones del mismo.]

Modelo de Casos de Prueba

Documento	Modelo de Casos de Prueba
Creado por	El Responsable de verificación, [nombre del responsable de verificación].
Para quien	Es la guía para realizar las pruebas del sistema y lo usarán los Asistentes de verificación y el Responsable de verificación cuando se ejecuten las pruebas del sistema.
Fecha de liberación	Será liberado el [fecha de primera liberación].

Informes de Verificación

Documento	Se genera un documento Informe de Verificación Unitaria por cada prueba unitaria que se realice al sistema.
Creado por	Las personas que ejecutan las pruebas.
Para quien	Es el retorno para los implementadores de la tarea de verificación, que detalla los errores encontrados para que puedan ser corregidos.
Fecha de liberación	Será liberado luego de cada verificación unitaria. [Indique la versión y la fecha de liberación de todas las versiones de este informe.]

Documento	Se genera un documento Informe Consolidación por cada consolidación que se realice al sistema.

Creado por	Las personas que ejecutan las pruebas.
Para quien	Es el retorno para los implementadores de la tarea de consolidación, que detalla los errores encontrados para que puedan ser corregidos.
Fecha de liberación	Será liberado luego de cada consolidación. [Indique la versión y la fecha de liberación de todas las versiones de este informe.]

Documento	Se genera un documento Informe de Verificación de Integración por cada prueba de integración que se realice al sistema.
Creado por	Las personas que ejecutan las pruebas.
Para quien	Es el retorno para los implementadores de la tarea de verificación, que detalla los errores encontrados para que puedan ser corregidos.
Fecha de liberación	Será liberado luego de cada verificación de integración. [Indique la versión y la fecha de liberación de todas las versiones de este informe.]

Documento	Se genera un documento Informe de Verificación de Sistema por cada prueba de sistema que se realice.
Creado por	Las personas que ejecutan las pruebas.
Para quien	Es el retorno para los implementadores de la tarea de verificación, que detalla los errores encontrados para que puedan ser corregidos.
Fecha de liberación	Será liberado luego de cada verificación de sistema. [Indique la fecha de liberación de este informe.]

Evaluación de la verificación

Documento	Se genera un documento Evaluación de la verificación por cada prueba que se realice al sistema. Este

	documento contiene las fallas encontradas en el sistema, la cobertura de la verificación realizada y el estado del sistema.
Creado por	El Responsable de verificación, que toma como fuente de su trabajo los Informes de verificación.
Para quien	Es el resumen de la tarea de verificación y es el retorno para todo el equipo de trabajo del estado del sistema.
Fecha de liberación	Será liberado luego de cada verificación, unitaria, de integración y de sistema. [Indique la versión y la fecha de liberación de todas las versiones de este informe.]

Informe final de verificación

Documento	El documento Informe final de verificación es el resumen de la verificación final del sistema antes de que sea liberado al entorno del usuario.
Creado por	El Responsable de verificación, que toma como fuente de su trabajo los Informes de verificación.
Para quien	Indica el estado del sistema.
Fecha de liberación	Será liberado luego de la verificación final del sistema.

Dependencias [opcional]

[En esta sección se detallan las dependencias, si existen, de las actividades de verificación respecto a otros elementos del sistema.]

Dependencia de personal [opcional]

[En esta sección se detallan las necesidades de personal para el equipo de verificación, la cantidad y habilidades.]

Dependencia de software [opcional]

[En esta sección se detallan los requisitos que debe cumplir el software a ser verificado para que se realice la verificación. Por ejemplo, que debe tener una

verificación previa por parte del implementador, que debe estar en fecha disponible para verificar.]

Dependencia de hardware [opcional]

[En esta sección se detallan las necesidades de disponibilidad de hardware para realizar las tareas de verificación. Por ejemplo, horario, cantidad de horas que debe estar disponible para que las tareas de verificación se puedan realizar dentro del cronograma establecido.]

Dependencia de datos y base de datos de prueba [opcional]

[En esta sección se detallan las necesidades de disponibilidad de dato y de la base de datos para realizar las tareas de verificación. Por ejemplo, conjuntos de datos necesarios para la verificación, horarios de acceso a la base de datos que permitan que las tareas de verificación se puedan realizar dentro del cronograma establecido.]

Riesgos [opcional]

[En esta sección se detallan los riesgos detectados que puedan afectar la normal realización de las tareas de verificación.]

Planificación [opcional]

[En esta sección se plantean los riesgos relativos a la planificación. Por ejemplo, si una cronograma es muy ajustado un pequeño retraso en la liberación del software para verificar atrasa la verificación y por consiguiente las actividades que dependen de esta, provocando un retraso en el cronograma de todo el proyecto.]

Técnico [opcional]

[En esta sección se plantean los riesgos técnicos que afectan a la verificación. Por ejemplo, si existe un sistema anterior se deberá hacer una verificación en paralelo con el anterior en lugar de liberar la versión en un punto de trabajo a modo de prueba del sistema.]

Gestión [opcional]

[En esta sección se plantea como mediante la gestión se pueden mitigar los riesgos en las tareas de verificación.]

Apéndice

Niveles de gravedad de error

En muchas actividades del proceso de verificación se deben clasificar los errores según su nivel de gravedad. Se asigna un nivel de gravedad a los errores para poder capturar de alguna manera su impacto en el sistema. Además para poder evaluar la verificación y el sistema.

A continuación se da una sugerencia de cuatro niveles diferentes de gravedad de error:

Catastrófico: un error cuya presencia impide el uso del sistema.

Crítico: un error cuya presencia causa la pérdida de una funcionalidad crítica del sistema. Si no se corrige el sistema no satisfará las necesidades del cliente.

Marginal: un error que causa un daño menor, produciendo pérdida de efectividad, pérdida de disponibilidad o degradación de una funcionalidad que no se realiza fácilmente de otra manera.

Menor: un error que no causa perjuicio al sistema, pero que requiere mantenimiento o reparación. No causa pérdida de funcionalidades que no se puedan realizar de otra manera.

Niveles de aceptación para lo elementos verificados

[Se debe establecer un nivel de aceptación para los elementos verificados para poder establecer el estado en el que se encuentra el proyecto.

En esta sección defina niveles de aceptación y los criterios de pertenencia a cada nivel.

Como ejemplo de niveles de aceptación:

No aprobado: el elemento verificado tiene errores catastróficos (uno o varios) que impiden su uso o tiene errores críticos (uno o varios) que hacen que el elemento verificado no sea confiable. El usuario no puede depender de él para realizar el trabajo.

Aprobado con Observaciones: el elemento verificado no tiene errores catastróficos, ni errores críticos, pero tiene errores marginales (uno o varios) que hacen que el elemento de software se degrade en algunas situaciones.

Aprobado: el elemento verificado no tiene errores o tiene errores menores que no afectan el normal funcionamiento del elemento.]

Anexo C: Descripción de los Casos de pruebas

[Lugar donde se realizan las pruebas]

[Casos de Pruebas.]

[Nombre del Proyecto], [Nombre del módulo]

[Nombre del Caso de uso que cubre el caso de prueba]

1 Descripción General

[Se describe de forma general las pruebas que se ejecutaran al caso a probar, así como se describe brevemente el caso de uso]
[Las pruebas realizadas a este caso de uso son:]

- ✓ Prueba 1
- ✓ Prueba 2
- ✓ Prueba n

1.1 CPR 1: Prueba 1

1.2 Descripción.

[Se describe brevemente como se debe ejecutar el caso de prueba]

1.3 Flujo Central.

[Se debe describir paso por paso el como se debe ejecutar el caso de prueba, con el fin de que el encargado de ejecutarla la realice sin problemas y esta pueda cumplir el objetivo para la cual fue creada. Se recomienda que se describa en forma de viñetas]

1.4 Condiciones de Ejecución

[Se describen todos los elementos que deben estar presente para lograr una correcta ejecución del caso de prueba.]

Clases válidas	Clases inválidas	Resultado Esperado	Resultado de la prueba	Observaciones	Cumplimiento %
[Se describen las clases válidas para la prueba]	*[Se describen las clases no válidas para la prueba]*	*[Se describen los resultados que se sabe que debe devolver el caso de prueba]*	*[Se describe como se ejecutó la prueba si fue correcta o no]*	*[Se describe las observaciones que sean pertinentes]*	*[Se describe el cumpliemto de la prueba en %]*

Registro de Defectos y Dificultades Detectadas

Elemento	Nº	Descripción de la No conformidad	Aspecto correspondiente	Etapa de la detección	Importante	Recomendaciones
[Se describe la naturaleza del error]	*[número]*	*[Se describe la no conformidad]*	*[Se describe el error]*			

[Nombre del que diseño la prueba]
[Fecha]
[Nombre del que ejecutó la prueba]
[Fecha]

Anexo D: Componente de prueba

1 Interfaz del componente de prueba.

En la figura 1 se ilustra la interfaz del componente de prueba creado para ejecutar el procedimiento de prueba del método Buscar Plantilla.

Figura 1

2 Cliente php.

```
<?php
$NUMERO = $_POST['numero'];
$ID_CARGO = intval($_POST['id_cargo']);
$DESCRIPCION = utf8_encode($_POST['descripcion']);
require 'PLASER/Client.php';
$datos = array();
$client= new PLASER_Client();
//$client->plaser_debug = 2   ;
$q=array();
$q['id_cargo']=$ID_CARGO;
$q['descripcion']=$DESCRIPCION;


$q['offset']=0;
```

```
$q['cantidad']=-1;
$res = $client->callB('RAS.BuscarPlantilla',$q);
//print_r($res->BusquedaPlantilla);exit;
$tabla='<table width="50%" border="0">';
   for($i=0;$i<=count($res->BusquedaPlantilla)-1;$i++)
   {
   $tabla.='<tr><td>'.$res->BusquedaPlantilla[$i]->id.'</td><td>'.utf8_decode($res-
>BusquedaPlantilla[$i]-
>descripcion).'</td><td><strong>id_cargo</strong></td><td><strong>Cargos</strong>
</td></tr>';
    $tbody="";
    for($j=0;$j<=count($res->BusquedaPlantilla[$i]->cargos)-1;$j++)
                {
      $id_cargo=$res->BusquedaPlantilla[$i]->cargos[$j]->id_cargo;
                        $descripcion=utf8_decode($res->BusquedaPlantilla[$i]-
>cargos[$j]->descripcion);
        $tbody.='<tr><td> </td><td> </td><td>'.$id_cargo.'</td><td>'.$des
cripcion.'</td><tr>';
                }
        $tabla.='<tbody>'.$tbody.'</tbody>';
   }
$tabla.='</table>';
echo $tabla;exit;
$A="
[cargos] => Array
            (
      [0] => stdClass Object   (
    [id_cargo] => 2
    [descripcion] => cargo 1
    [cantidad_cargo] => 0
                )
      [1] => stdClass Object
                (
    [id_cargo] => 6
    [descripcion] => Jefe Grupo Basico
    [cantidad_cargo] => 0
                )
            )
";


?>
```

Prueba A: Errores de la ayuda

1. No muestra la ayuda correspondiente a la página cuando se oprime la tecla F-1.
2. Los siguientes link no funcionan.
   - ✓ Página de inicio.
   - ✓ Tabla de contenido.
   - ✓ Cerrar.
   - ✓ Exportar el manual a PDF.
   - ✓ Anterior.
   - ✓ Siguiente.
3. La imagen de buscar plantilla falta.