Universidad Autónoma del Estado de México

Facultad de Ingeniería

“Desarrollo de un plan de pruebas para un data warehouse de una empresa del sector financiero”

TESINA

Que para obtener el título de

Ingeniera en Computación

Presenta

Gabriela Nataly Trujillo Hernández

Asesor

Dr. Marcelo Romero Huertas

Toluca de Lerdo, Noviembre de 2019

Resumen

En esta tesina se presentan conceptos relacionados al proceso de pruebas como niveles y tipos de pruebas los cuales son importantes al desarrollar un plan de pruebas.

Por otro lado, se define la estructura de un plan de pruebas para un *data warehouse* con base en el estándar *IEEE 829*, incluyendo las secciones: introducción, elementos de las pruebas, riesgos, características a probar, características que no se prueban, alcance, criterios de aprobación, tareas, ambientes, equipo de trabajo, responsabilidades, calendario y plan de riesgos.

Además, se exponen un caso de estudio del desarrollo de un repositorio *data warehouse* de una empresa del sector financiero, para la cual se crea un plan de pruebas. Como parte del caso de estudio se mencionan los principales componentes y procesos involucrados. Adicionalmente, se presentan conceptos de negocio relacionados a la institución bancaria, los cuales se utilizan para crear el modelo de datos bancario del *data warehouse*.

Finalmente, se exponen las conclusiones del trabajo realizado.

Índice

Índice de Figuras

Índice de tablas

Introducción

En la actualidad, la mayoría de las personas son conscientes de los sistemas de software, los cuales son utilizados en los hogares, centros de trabajo, y lugares públicos. Debido a la globalización, dichos sistemas cada vez se integran más a la vida cotidiana.

Sin embargo, la mayoría de las personas han tenido una experiencia con software que no funciona como se espera: un error en una factura, un retraso al esperar una tarjeta de crédito que está siendo procesada en un cajero y un sitio web que no se carga correctamente, son ejemplos comunes de problemas que pueden ocurrir debido a problemas de software (Soares, 2011).

No todos los sistemas de software tienen el mismo nivel de riesgo y no todos los problemas tienen el mismo impacto cuando ocurren, así que existe cierta preocupación por estos problemas potenciales ya que si uno de ellos ocurriera habría un impacto negativo (Graham, Veenendaal, Evans, & Black, 2008). La calidad es una característica crítica de los sistemas de *TI* que se asegura a través de las medidas de pruebas correspondientes.

El presente proyecto expone el desarrollo de un plan de pruebas para un *data warehouse*, en el cual se describe la planificación, ejecución y planes de gestión en una institución bancaria, cuyo objetivo final es garantizar la calidad de los datos a nivel integridad referencial y consistencia. El plan consiste en definir niveles y fases de prueba, dependencias y supuestos, ambientes de prueba, calendario, análisis de riesgos, roles y responsabilidades del equipo de pruebas, así como procesos de pruebas a seguir.

Objetivo general:

Documentar el desarrollo de un plan de pruebas para un *data warehouse* de una empresa del sector financiero, incluyendo las fases de análisis, planificación y ejecución, para procurar la integridad referencial y consistencia de los datos.

La implementación de un plan de pruebas en la construcción de un data warehouse es importante para lograr una liberación exitosa pues el principal

objetivo es tener datos de calidad en el *data warehouse* y cumplir los requerimientos solicitados.

Para el caso de estudio de este tema se toma como referencia una empresa del sector financiero, que por confidencialidad no se menciona el nombre de dicha compañía, así como tampoco se menciona como tal nombre de las bases o sistemas complementarios, se hace mención de ellos con un nombre clave.

Por motivos técnicos algunos de los términos o nombres se presentan en este documento en inglés debido a que en la industria comúnmente se usan de esta forma.

Alcances:

En el caso de uso documentado en esta tesina, la implementación del *data warehouse* en la institución bancaria se divide en 4 fases. Sin embargo, en este documento solamente se describe la fase 1 de la implementación del *data warehouse*, la cual incluye extracción de datos, integración y administración de requerimientos. Para lo cual, en la fase 1 se contemplan como fuente de datos 4 bases de datos de la institución bancaria para poblar el *data warehouse, las cuales proporcionan detalle de la información del cliente*.

El mapeo de datos de las bases de datos fuente de la institución bancaria son un desafío conocido en la industria bancaria, para este caso de estudio el requerimiento de mapeo son 5,417 atributos lo cual se considera un esfuerzo grande, las actividades adicionales que ayudan a la reducción de este objetivo fueron: comenzar con las fuentes bien documentadas y priorizar el mapeo de entidades, por tal motivo la fase 1 contempla solo 1,992 atributos destino.

Estructura de la tesina:

Esta tesina consta de 3 capítulos:

El *Capítulo I. Generalidades,* describe conceptos relacionados a calidad en un proyecto de tecnologías de la información, así como la fase de pruebas, niveles, tipos y técnicas de pruebas.

El *Capítulo II. Plan de pruebas,* presenta el desarrollo del plan de pruebas, incluyendo las secciones que debe contener de acuerdo al estándar *IEEE 829* para validar un *data warehouse, destacando: objetivos, alcance, ambientes de pruebas, niveles de pruebas, roles y responsabilidades.*

El *Capítulo III. Caso de estudio,* documenta la implementación del plan de pruebas para validar un *DWH* en una empresa del sector financiero, se presentan generalidades del proyecto.

Al final del documento se exponen las conclusiones y sugerencias obtenidas en esta tesina.

CAPÍTULO I. Generalidades

Este capítulo describe conceptos relacionados a calidad en un proyecto de tecnologías de la información, así como la fase de pruebas, niveles, tipos y técnicas de pruebas.

1.1 Calidad de software

Todo proyecto tiene como objetivo producir software de la mejor calidad posible, que cubra las expectativas de los usuarios.

La definición de calidad según la organización *ISO* es "La totalidad de características de un producto, proceso o servicio que cuenta con la habilidad de satisfacer necesidades explícitas o implícitas".

Dentro del contexto de Ingeniería de software, la definición de calidad de software es un conjunto de actividades que se realizan durante el proceso de desarrollo que influencia la calidad del producto de software final. La medición de la calidad del producto está en términos del conteo de defectos encontrados en el software durante su desarrollo y operación. Un proceso de mayor calidad, es decir, un proceso de desarrollo maduro, generará productos de software con menos defectos. (Garcia Mireles, 2016).

1.2 Pruebas de software

El término pruebas de software o *software testing* en inglés se define como un conjunto de actividades, tareas, técnicas y herramientas que buscan asegurar que la aplicación o software funcione de manera apropiada, con el fin de encontrar todos los errores o fallos que al final serán corregidos. Es decir, es el proceso de ejecutar una serie de pasos con la intención de encontrar fallos (Valentin Pozo & Veliz Ticse , 2014).

Las pruebas de software se han convertido en un factor determinante para logar el éxito de sistemas, de ahí la importancia de establecer un plan que nos guíe el camino para llegar a nuestro objetivo, un software con la mayor calidad posible.

Las ventajas de establecer un proceso de calidad son:

- Detectar la mayor cantidad de errores posibles antes de salir a producción.
- Ayudar a los administradores a la toma de decisiones.
- Buscar los escenarios seguros para el uso del producto.
- Evaluar la calidad.
- Verificar la corrección del producto.
- Asegurar la calidad (Campos Chiu, 2015).

1.3 Proceso de pruebas

El proceso fundamental de pruebas y actividades se describen en esta sección. Dicho proceso empieza con la planificación de las pruebas y continua hasta el cierre, para cada parte del proceso de pruebas hay actividades definidas.

Las actividades dentro del proceso de pruebas pueden dividirse en pasos básicos como lo son:

- Planeación y control: Durante la planificación de las pruebas, se debe de asegurar la comprensión de las metas y objetivos de los clientes, las partes interesadas y del proyecto, también los riesgos que pueda haber. Esto conlleva a lo que se llama misión de las pruebas o la asignación de pruebas. Con base a esta comprensión, se establece las metas y objetivos para las pruebas en sí, y se deriva un enfoque y un plan de pruebas, incluida la especificación de las actividades de prueba.

  El control de las pruebas es una actividad continua. Se necesita comparar el progreso real con el progreso planificado e informar al gerente del proyecto y al cliente sobre el estado actual de las pruebas, incluidos los cambios o desviaciones del plan.

- Análisis y diseño: El análisis y diseño de las pruebas es la actividad donde los objetivos generales de prueba se transforman en condiciones de prueba tangibles y diseños de prueba. Durante el análisis y diseño de la prueba, se toman los objetivos de prueba

generales identificados durante la planificación y se construye diseños y procedimientos de prueba.

- Implementación y ejecución: Durante la implementación y ejecución de las pruebas, se toman las condiciones de estas y se convierten en casos de prueba, *testware* y configuración del ambiente de pruebas. Esto significa que, después de armar un diseño de alto nivel de pruebas se comienza a construir.

  Las condiciones de prueba se transforman en casos de prueba, procedimientos de prueba y *testware,* tal es el caso de *scripts* para automatización. También se necesita configurar un entorno donde se ejecuten los casos de prueba y construir los datos de prueba.

- Evaluación de criterios de salida y reporteo: La evaluación de los criterios de salida es la actividad en la que la ejecución de las pruebas se evalúa en relación con los objetivos definidos. Esto debe hacerse para cada nivel de prueba, ya que para cada uno se necesita saber si se han realizado suficientes pruebas.

  Basándose en la evaluación de riesgos definidos, se habrán establecido criterios a través de los cuales hay una medición (suficiente). Estos criterios varían para cada proyecto y se conocen como criterios de salida. Permiten saber si se puede declarar una actividad de prueba terminada o un nivel completado.

- Cierre de actividades de pruebas: Durante las actividades de cierre de pruebas, se recopilan datos de las actividades de prueba completadas para consolidar los resultados, incluida la verificación y la presentación de pruebas, el análisis de hechos y números. Es posible que se necesite hacer esto cuando se entregue el software (Graham, Veenendaal, Evans, & Black, 2008).

1.4 Niveles de prueba

Nivel de prueba o *test level* en inglés, es el grupo de actividades que están organizadas y gestionadas de forma conjunta. Un nivel de prueba está vinculado con las responsabilidades en un proyecto.

En esta sección se analizan y mencionan las características clave para cada nivel, existen cuatro tipos de nivel de pruebas: pruebas de componentes, pruebas de integración, pruebas de sistema y pruebas de aceptación, cada uno de ellos tiene sus propios objetivos.

La decisión acerca de la elección de niveles de prueba, depende del sistema o contexto del software y del nivel de riesgo asociado con él.

1.4.1 Pruebas de componentes

También conocido como *component testing* en inglés, tiene por objeto localizar defectos y comprobar el funcionamiento de módulos software, programas, objetos, clases, etc., que puedan probarse por separado. Es decir, se pueden realizar de manera independiente al resto del sistema (Mera Paz J. A., 2016).

1.4.2 Pruebas de integración

Se encargan de probar las interfaces entre los componentes o módulos; por ejemplo, el componente validación de usuario con el sistema operativo, el sistema de archivos en integración con el hardware, etc. Diseño de casos de pruebas: "Diseño de software, arquitectura, flujos de trabajo, casos de uso, se deben tener en cuenta los objetos de prueba típicos: base de datos de subsistemas, infraestructura, interfaces, configuración del sistema, datos de configuración (Graham, Veenendaal, Evans, & Black, 2008).

1.4.3 Pruebas de sistema

Hacen referencia al sistema como un todo; se debe elaborar un plan de pruebas de forma clara y bien estructurada. Diseño de casos de pruebas: "Requisitos del usuario, requisitos del sistema, casos de uso, procesos de negocio, informes de análisis de riesgo. Se deben tener en cuenta los objetos de prueba típicos: procesos de negocio en sistema completamente integrado, procesos operativos y de mantenimiento, procedimientos de usuario, formularios, informes, datos de configuración" (Mera Paz J. , 2016).

1.4.4 Pruebas de aceptación

Se enfocan en la aceptación de los criterios previstos en un contrato de desarrollo de software, acordado entre la fábrica de software y el cliente. “Las pruebas de aceptación del sistema por parte de los administradores del sistema, entre las que se incluyen” (Graham, Veenendaal, Evans, & Black, 2008).

1.5 Tipos de pruebas

Existen diferentes tipos de prueba de software, las que tienen como objetivo probar funcionalidad de software, las que prueban características no funcionales, las que buscan probar la estructura del software y las que aseguran que no se ha introducido un defecto después de los cambios realizados al software.

Tomando en cuenta lo arriba descrito se tienen cuatro tipos de pruebas:

- Pruebas funcionales.
- Pruebas no funcionales.
- Pruebas estructurales.
- Pruebas de regresión.

1.5.1 Pruebas funcionales

Este tipo de pruebas se basa en las funcionalidades de un sistema que se describen en la especificación de requisitos, es decir, lo que hace el sistema. En la *ISO 25010* indica que “la funcionalidad representa la capacidad del producto de software para proporcionar funciones que satisfacen las necesidades declaradas e implícitas, cuando el producto se usa en las condiciones específicas”.

Estas pruebas pueden llevarse a cabo en todos los niveles de prueba y suelen estar asociadas a las técnicas de diseño de pruebas de caja negra, ya que tienen en cuenta el comportamiento externo del software (Sánchez Peño, 2015).

1.5.2 Pruebas no funcionales

Las pruebas no funcionales se interesan en qué tan bien o que tan rápido se hace algo. Se prueba algo que se necesita medir en una escala, por ejemplo, tiempo de respuesta. Este tipo de prueba se realiza en todos los niveles de prueba.

Las pruebas no funcionales incluyen entre otras, pruebas de rendimiento, pruebas de carga, pruebas de estrés, pruebas de usabilidad, pruebas de mantenimiento, pruebas confiabilidad y pruebas de portabilidad (Graham, Veenendaal, Evans, & Black, 2008).

1.5.3 Pruebas estructurales

Las pruebas estructurales permiten medir la totalidad de las pruebas mediante la evaluación de tipo estructura.

Las pruebas estructurales se utilizan con mayor frecuencia como una forma de medir la exhaustividad de las pruebas a través de la cobertura de un conjunto de elementos estructurales o elementos de cobertura. Puede ocurrir en cualquier nivel de prueba, aunque tiende a aplicarse principalmente en componentes e integración y, en general, es menos probable en niveles de prueba más altos. A nivel de integración de componentes, puede basarse en la arquitectura del sistema, como una jerarquía de llamadas. Una base de prueba de sistema, integración de sistema o prueba de aceptación podría ser un modelo de negocio o una estructura de menú.

Las técnicas utilizadas para las pruebas estructurales son técnicas basadas en estructuras, también conocidas como técnicas de caja blanca (Graham, Veenendaal, Evans, & Black, 2008).

1.5.4 Pruebas de regresión

Las pruebas de regresión consisten en probar un sistema que ha sido analizado previamente para asegurar que no se haya introducido algún tipo de defecto como resultado de cambios realizados (Chavarria, 2018).

Las pruebas de regresión se ejecutan cada vez que el software cambia, ya sea como resultado de correcciones o funcionalidades nuevas o modificadas.

También es una buena idea ejecutarlos cuando cambia algún aspecto del entorno, por ejemplo, cuando se introduce una nueva versión de un sistema de administración de bases de datos o se usa una nueva versión de un compilador de código fuente.

1.6 Testing outline

La definición de *testing outline* describe los objetivos que se relacionan con la evaluación, revelando defectos y calidad. Como se indica en la definición, se pueden utilizar dos enfoques para lograr estos objetivos, pruebas estáticas y pruebas dinámicas.

Las pruebas dinámicas y las pruebas estáticas son métodos complementarios, ya que tienden a encontrar diferentes tipos de defectos de manera efectiva y eficiente. Los tipos de defectos que son más fáciles de encontrar durante las pruebas estáticas son: desviaciones de los estándares, requisitos faltantes, defectos de diseño, código no mantenible y especificaciones de interfaz inconsistentes (Graham, Veenendaal, Evans, & Black, 2008).

1.6.1 Pruebas estáticas

Durante las pruebas estáticas, los productos de trabajo de software se examinan manualmente o con un conjunto de herramientas, pero no se ejecutan. Entre las preguntas que surgen están: ¿Cómo podemos evaluar o analizar un documento de requisitos, un documento de diseño, un plan de prueba o un manual de usuario? ¿Cómo podemos examinar efectivamente el código fuente antes de la ejecución? Una técnica poderosa que se puede usar es la prueba estática.

Las pruebas estáticas son un método muy adecuado para mejorar la calidad de los productos de trabajo de software. Esto se aplica principalmente a los productos evaluados, pero también es importante que la mejora de la calidad no se logre una vez, sino que tenga un carácter más estructural. La retroalimentación del proceso de pruebas estáticas al proceso de desarrollo permite la mejora del proceso, lo que respalda la prevención de errores similares en el futuro.

Algunas de las técnicas de pruebas estáticas son: revisiones informales, revisiones técnicas y *walkthroughs*.

1.6.2 Pruebas dinámicas

Con los métodos de prueba dinámicos, el software se ejecuta utilizando un conjunto de valores de entrada y su salida se examina y compara con lo que se espera. Las pruebas dinámicas solo pueden aplicarse al código de software. La ejecución dinámica se aplica como una técnica para detectar defectos y para determinar los atributos de calidad del código. Esta opción de prueba no es aplicable para la mayoría de los productos de trabajo de software.

Las técnicas dinámicas se subdividen en tres categorías: basadas en especificaciones (*black-box* también conocido como técnicas de comportamiento), basadas en estructuras (*white-box* o técnicas estructurales) y basadas en la experiencia.

CAPÍTULO II. Plan de pruebas

En este capítulo se describe el plan de pruebas y las secciones que lo comprenden con base en el estándar *IEEE 829.*

2.1 Definición de plan de pruebas

Aunque ejecutar pruebas es importante, también se necesita un plan de acción y un informe sobre el resultado de las pruebas.

La planificación de las pruebas es uno de los puntos más importantes a la hora de probar un proyecto, se debe tener controlados todos los aspectos, desde la manera de la que se va probar, cómo se a probar, los recursos de los que se van a disponer, las pruebas de regresión, la documentación del proyecto, etc. (Sánchez Peño, 2015).

El plan de pruebas es un producto formal que define los objetivos de la prueba de un sistema, establece y coordina una estrategia de trabajo, y provee del marco adecuado para elaborar una planificación paso a paso de las actividades de prueba. El plan se inicia en el proceso de análisis del sistema de información, definiendo el marco general, y estableciendo los requisitos de prueba de aceptación, relacionados directamente con la especificación de requisitos (Cillero, 2001).

Dicho plan se va completando y detallando a medida que se avanza en los restantes procesos del ciclo de vida del software, se debe tener en cuenta que desde el punto de vista general el ciclo de vida del software tiene tres etapas: planificación, implementación y puesta a producción (Villagómez, 2017).

Existen otras razones que hacen que el plan de prueba sea importante:

- Ayuda a determinar el esfuerzo que se requiere para validar la calidad de la aplicación bajo prueba.
- Ayuda a las personas externas al equipo de pruebas, como desarrolladores, gerentes de negocios, clientes y otras partes interesadas, a comprender los detalles de las diversas pruebas ejecutadas en la aplicación.

- El plan de prueba guía el pensamiento del probador, ya que actúa como un libro de reglas que debe seguirse sin interrupción.
- Los probadores pueden documentar los aspectos importantes relacionados con las pruebas de software, como la estimación de la prueba, el alcance de la prueba, la estrategia de prueba y más, que luego puede ser revisado por el equipo de gestión y reutilizado para otros proyectos (QA, 2018).

El plan se inicia en el proceso de análisis del sistema de información, definiendo el marco general, y estableciendo los requisitos de prueba de aceptación, relacionados directamente con la especificación de requisitos.

Los planes de proyecto y prueba deben incluir el tiempo que se debe dedicar a planificar las pruebas, diseñar casos de prueba, prepararse para la ejecución y evaluar el estado.

El equipo de probadores sigue el estándar *IEEE 829 (que se explica en la Sección 2.2)* de documentación del plan de prueba, ya que puede haber una diferencia de opinión con respecto a lo que debe incluirse en un documento del plan de prueba.

2.2 Estándar IEEE-829

IEEE es una institución internacional que define estándares y plantillas que son universalmente reconocidas y aceptadas. Tiene estándares definidos para la documentación del plan de prueba del sistema y el software, que se conoce globalmente como estándar *IEEE 829*. Este estándar especifica el formato para un conjunto de documentos que se requieren para las pruebas de software y sistema. El estándar *IEEE 829* especifica las diversas etapas que se encuentran en el proceso de documentación, cada una de las cuales produce un documento separado para cada etapa de prueba (Ramos, 2018).

De acuerdo con el estándar *IEEE 829* un plan de pruebas debe contener la siguiente información:

- Identificador del plan de pruebas.
- Referencias.
- Introducción.

- Elementos sujetos a pruebas.
- Riesgos.
- Características que se probarán.
- Características que no se probarán.
- Alcance.
- Criterios de aprobación/fallo.
- Criterios de suspensión y requisitos de reanudación.
- Entregables de prueba.
- Tareas de pruebas restantes.
- Ambientes.
- Equipo de trabajo y capacitación.
- Responsabilidades.
- Calendario.
- Plan de riesgos y contingencias.
- Aprobaciones.
- Glosario (IEEE, 2014).

2.3 Identificador del plan de pruebas

El identificador es un número único generado por la compañía para distinguir el plan de pruebas y el nivel de software con el que está relacionado. Esto es para ayudar a coordinar las versiones de software y software de prueba dentro de la administración de la configuración, incluso la versión del plan de pruebas (IEEE, 2014).

2.4 Referencias

Las referencias son la lista de los documentos que dan soporte al plan de pruebas. Los documentos que pueden estar relacionados o son punto de apoyo para la creación del plan de pruebas son: plan del proyecto, especificación de los requerimientos, documento de diseño de alto nivel, documentos de diseño de software, estándares de la empresa y guías, entre otros.

2.5 Introducción

En esta sección se expone el propósito del plan de pruebas, esto es el resumen ejecutivo del plan. Es posible que se desee incluir referencias a otros

planes, documentos o elementos que contengan información relevante para este proyecto o proceso. De preferencia crear una sección de referencias para contener dicha información.

Se identifica el alcance del plan de pruebas en relación con el plan del proyecto de software al que se refiere. Otros elementos que se pueden incluir son: restricciones de recursos y presupuesto, alcance del esfuerzo de prueba, cómo se relacionan las pruebas con otras actividades de evaluación (análisis y revisiones), y es posible que el proceso sea utilizado para el control de cambios y la comunicación y coordinación de actividades clave.

Como este es el "Resumen Ejecutivo", se debe mantener la información breve y concisa.

2.6 Elementos de las pruebas

Se refiere a los elementos que se desea probar dentro del alcance del plan de pruebas. Esencialmente algo que se probará, una lista de lo que se va a probar.

Los elementos pueden definirse analizando los documentos de desarrollo, así como otras fuentes de documentación e información. Esto puede ser controlado y definido por el proceso local de *Configuration Management* (*CM*) en caso de contar con él. Esta información incluye números de versión, requisitos de configuración donde necesario (especialmente si se admiten varias versiones del producto).

Se debe recordar, que lo que se está probando es lo que se pretende entregar al Cliente.

2.7 Riesgos

Para la sección de riesgos se identifica qué software se va a probar y cuáles son las áreas críticas, como:

- Entrega de un producto de terceros.
- Nueva versión de la interfaz del software.
- Capacidad para usar y comprender una nueva herramienta, etc.

- Funciones extremadamente complejas.
- Modificaciones a componentes con antecedentes de fallas.
- Módulos mal documentados o solicitudes de cambio.

Existen algunos riesgos inherentes al software, como la complejidad; estos necesitan ser identificados:

- Seguridad.
- Múltiples interfaces.
- Impactos en el cliente.
- Regulaciones y normas gubernamentales.

Otra área clave de riesgo es no entender el requerimiento original. Esto puede ocurrir en los niveles de gestión, usuario y desarrollador. Hay que tener en cuenta los requisitos vagos o poco claros no pueden ser probados correctamente.

El historial de defectos (errores) en el pasado descubiertos durante las pruebas unitarias ayudará a identificar posibles áreas dentro del software que son riesgosas. Si el *unit test* o pruebas unitarias en español descubriera una gran cantidad de defectos o una tendencia hacia defectos en un área particular del software, esto es una indicación de posibles problemas futuros.

Es la naturaleza de los defectos agruparse, un buen enfoque para definir dónde están los riesgos es tener varias sesiones de lluvia de ideas.

2.8 Características que se probarán

Esta es una lista de lo que se probará desde el punto de vista de usuarios de lo que hace el sistema, esta no es una descripción técnica del software, sino una vista de usuario de las funciones.

Establecer el nivel de riesgo para cada característica y usar una escala de calificación simple como (A, M, B): alta, media y baja, estos tipos de niveles son entendibles para un usuario. Se debería de estar preparado para discutir por qué se eligió un nivel en particular.

Cabe señalar que la Sección 2.6 y la Sección 2.8 son muy similares, la única diferencia verdadera es el punto de vista. La Sección 2.6 es una descripción de tipo técnico que incluye números de versión y otra información técnica y la Sección 2.8 son desde el punto de vista del usuario. Los usuarios no entienden la terminología técnica del software; entienden las funciones y procesos en relación con sus trabajos.

2.9 Características que no se probarán

Esta es una lista de lo que no se debe probar desde el punto de vista de los usuarios, de lo que el sistema hace y una vista de control de versiones o gestión de la configuración. Esto no es una descripción técnica del software, pero si una vista de usuario de las funciones.

Identificar por qué la característica no se va a probar, puede haber varias razones:

- No se incluirá en esta versión del software.
- Bajo riesgo, se ha usado antes y se considera estable.
- Se liberará, pero no se probará ni documentará como parte funcional del lanzamiento de esta versión del software.

Las Secciones 2.8 y 2.9 están directamente relacionadas con las Secciones 2.7 y 2.19. Lo que será y no será probado es directamente afectado por los niveles de riesgo aceptables dentro del proyecto y lo que no se prueba afecta el nivel de riesgo del proyecto.

2.10 Alcance (estrategia)

Esta es la estrategia general de prueba para el plan de pruebas, en la cual se deben identificar las reglas y procesos generales.

- ¿Se deben utilizar herramientas especiales y cuáles son?
- ¿La herramienta requerirá entrenamiento especial?
- ¿Qué métricas se recopilarán?
- ¿En qué nivel se debe recopilar cada métrica?
- ¿Cómo se maneja la gestión de la configuración?
- ¿Cuántas configuraciones diferentes se probarán?

- Hardware.
- Software.
- Combinaciones de hardware, software y otros paquetes de proveedores.
- ¿Qué niveles de pruebas de regresión se realizarán y cuánto en cada nivel de prueba?
- ¿Las pruebas de regresión se basarán en la gravedad de los defectos detectados?
- ¿Cómo serán los elementos en el requerimiento y el diseño que no hacen sentido o no se puede procesar?

El enfoque general de pruebas del proyecto y los requisitos de cobertura también debe ser identificados. Especificar si hay requisitos especiales para la prueba.

- Solo se probará el componente completo.
- Un segmento específico de agrupación de características o componentes debe probarse juntos.

2.11 Criterios de aprobación/fallo

Se debe definir cuáles son los criterios de finalización de este plan, este es un aspecto crítico del plan de prueba.

- En el nivel de ejecución de prueba, esto podría ser elementos como:
    - Todos los casos de prueba completados.
    - Un porcentaje especificado de casos completados con un porcentaje que contiene algún número de defectos menores.
    - La herramienta de cobertura de código indica todo el código cubierto.

- En el nivel del plan de pruebas, esto podría ser elementos como:
    - Todos los planes de nivel inferior completados.

- Un número especificado de planes completados sin errores y un porcentaje con menor defectos.

2.12 Criterios de suspensión y requisitos de reanudación

Es importante saber cuándo detener la ejecución de las pruebas, si el número o tipo de defectos alcanza un punto en el que el seguimiento de las pruebas no tiene valor, no tiene sentido continuar la prueba; solo están desperdiciando recursos.

Especificar qué constituye la interrupción de la prueba o serie de pruebas y cuál es el nivel aceptable de defectos que permitirán que la prueba continúe más allá de los defectos.

Las pruebas después de un error verdaderamente fatal generarán condiciones que pueden identificarse como defectos, pero, son errores fantasmas causados por los defectos anteriores que se ignoraron.

2.13 Entregables de prueba

Entregables de prueba se refiere a lo que se entregara como parte del plan, entre ellos están:

- Documento del plan de prueba.
- Casos de prueba.
- Especificaciones del diseño de prueba.
- Herramientas y sus resultados.
- Simuladores.
- Generadores estáticos y dinámicos.
- Registros de errores y registros de ejecución.
- Informes de problemas y acciones correctivas.

Una cosa que no se puede entregar en una prueba es el software en sí, que se enumera en los elementos de prueba y se entrega por desarrollo.

2.14 Tareas de pruebas restantes

Si este es un proceso de múltiples fases o si la aplicación se va a lanzar incrementalmente, puede haber partes de la aplicación que este plan no aborda. Estas áreas necesitan ser identificadas para evitar cualquier confusión en caso de que se reporten defectos en esas funciones futuras. Esto también permite a los usuarios y evaluadores evitar funciones incompletas y evitar el desperdicio de recursos persiguiendo falsos defectos.

Si el proyecto se desarrolla como un proceso de múltiples partes, este plan solo puede cubrir una parte del total de funciones o características. Este estado necesita ser identificado para que esas otras áreas tengan planes desarrollados para ellos y para evitar desperdiciar recursos rastreando defectos que no se relacionan con este plan.

Cuando un tercero está desarrollando el software, esta sección puede contener descripciones de aquellos probar tareas pertenecientes a los grupos internos y externos.

2.15 Ambientes

Existen requerimientos especiales para el plan de prueba, como:

- Hardware especial como simuladores, generadores estáticos, etc.
- ¿Cómo se proporcionarán los datos de prueba?
- ¿Existen requisitos de recopilación especiales o rangos específicos de datos que se deben proporcionar?
- ¿Cuántas pruebas se realizarán en cada componente de una función de varias partes?
- Requisitos especiales de potencia.
- Versiones específicas de otro software de soporte.
- Uso restringido del sistema durante las pruebas.

2.16 Equipo de trabajo y capacitación

Formación sobre la aplicación o sistema, capacitación para las herramientas de prueba que se utilizarán. La Sección 2.6 y la Sección 2.17 también afectan esta sección.

Qué se debe evaluar y quién es responsable para las pruebas y el entrenamiento, estas preguntas ayudan a definir el tipo de capacitación que se requiere y también definir el equipo de trabajo.

2.17 Responsabilidades

En el desarrollo de un plan de plan de pruebas, es importante definir desde el inicio, quién está a cargo, incluyendo todas las áreas del plan. Aquí hay unos ejemplos:

- Establecer riesgos.
- Seleccionando características para ser probadas y no probadas.
- Establecer una estrategia general para este nivel de plan.
- Asegurarse de que todos los elementos necesarios estén en su lugar para la prueba.
- Proporciona la resolución de conflictos de programación, especialmente, si las pruebas se realizan en el sistema de producción.
- ¿Quién proporciona la capacitación requerida?
- ¿Quién toma las decisiones críticas de ir o no ir para los elementos que no están cubiertos en los planes de prueba?

2.18 Calendario

Debe basarse en estimaciones realistas y válidas, si las estimaciones para el desarrollo de la aplicación son inexactas, el plan completo del proyecto se deslizará y las pruebas son parte de plan general del proyecto.

En este punto, todos los hitos relevantes deben identificarse con su relación con el proceso de desarrollo identificado. Esto también ayudará a identificar y rastrear el potencial deslizamiento en el horario causado por el proceso de prueba.

Siempre es mejor vincular todas las fechas de prueba directamente a sus fechas de actividad de desarrollo relacionadas, esto evita que el equipo de prueba sea percibido como la causa de un retraso. Por ejemplo, si el sistema las pruebas deben comenzar después de la entrega de la compilación final, luego las pruebas del sistema comienzan el día después entrega. Si la entrega

se retrasa, las pruebas del sistema comienzan desde el día de la entrega, no en una fecha específica.

2.19 Plan de riesgos y contingencias

Otro punto importante es identificar los riesgos generales para el proyecto haciendo énfasis en el proceso de prueba, destacando:

- Falta de recursos de personal cuando se inician las pruebas.
- Falta de disponibilidad de hardware, software, datos o herramientas requeridos.
- Entrega tardía del software, hardware o herramientas.
- Retrasos en la capacitación sobre la aplicación y / o herramientas.
- Cambios en los requisitos o diseños originales.

Se debe especificar lo que se hará para varios eventos, por ejemplo:

- La definición de requisitos se completará antes del 1 de enero de 19XX y, si los requisitos cambiar después de esa fecha, se tomarán las siguientes acciones.
    - El cronograma de prueba y el cronograma de desarrollo se moverán un número apropiado de días. Esto rara vez ocurre, ya que la mayoría de los proyectos tienden a tener fechas de entrega fijas.
    - El número de pruebas realizadas se reducirá.
    - Se aumentará el número de defectos aceptables.
        - Estos dos elementos podrían reducir la calidad general del producto entregado.
    - Se agregarán recursos al equipo de prueba.
    - El equipo de prueba trabajará horas extra.
        - Esto podría afectar la moral del equipo.
    - El alcance del plan puede ser modificado.
    - Puede haber cierta optimización de los recursos. Esto debe evitarse, si es posible, por obvias razones.
    - Podrías simplemente SALIR. Una opción bastante extrema para decir lo menos.

La administración generalmente es complicada de aceptar escenarios como el anterior, aunque lo he visto suceder en el pasado.

Lo importante es recordar que, si no hace nada, el resultado habitual es que las pruebas se reducen o se omiten por completo, ninguna de las cuales debería ser una opción aceptable.

2.20 Aprobaciones

También se debe considerar la aprobación del proceso completo para permitir que el proyecto pase al siguiente nivel, en caso de existir. En el nivel del plan de prueba, esto puede ser todas las partes involucradas. Al determinar el proceso de aprobación, hay que tener en cuenta quién es la audiencia.

- Los niveles y el tipo de conocimiento en los distintos niveles también serán diferentes.
- Los programadores son muy técnicos, pero pueden no tener una comprensión clara de la situación general proceso empresarial que impulsa el proyecto.
- Los usuarios pueden tener diferentes niveles de perspicacia comercial y muy pocas habilidades técnicas.
- Siempre hay que tener cuidado con los usuarios que afirman tener altos niveles de habilidades técnicas y programadores que pretenden comprender completamente el proceso comercial. Estos tipos de individuos pueden causar más daño que bien si no tienen las habilidades que creen que poseen.

2.21 Glosario en el plan de pruebas

Se utiliza para definir términos y acrónimos utilizados en el documento, y pruebas en general, para eliminar confusión y promover comunicaciones consistentes.

CAPÍTULO III. Caso de estudio

En este capítulo se documenta el desarrollo de un plan de pruebas para un *data warehouse* de una empresa del sector financiero, incluyendo: componentes, procesos para la construcción y conceptos de negocio en los cuales se basa el modelo de datos financiero de la institución bancaria.

3.1 Antecedentes

Una empresa dedicada al sector financiero (Institución Bancaria) cuya oferta son productos tal como: cuentas de ahorro, transacciones, cuentas de crédito, entre otros, debe mantenerse a la vanguardia en este mundo competitivo, por ello se ve en la necesidad de ofrecer nuevos y mejores productos que sean personalizados según las necesidades.

Por tal motivo se recurre a una estrategia de *Business Intelligence (BI)* ya que las aplicaciones de inteligencia empresarial pueden desarrollar una comprensión de la construcción de un caso de negocios dándole valor a la información almacenada de sus clientes.

Son muchos los conceptos, herramientas y tecnologías que se engloban, dentro una solución completa de Inteligencia de Negocios. El objetivo de la inteligencia de negocio es proporcionar la información adecuada, en el momento adecuado, para la persona adecuada y en el formato adecuado. Los pasos para lograr este objetivo son:

1. Tener información registrada, el objetivo de análisis (bases de datos fuente).
2. Identificar las distintas fuentes de datos y recolectar toda la información.
3. Transformar, combinar y almacenar la información en un almacén de datos (creación de *data warehouse*).
4. Crear informes a partir de la información almacenada y modelada en el *data warehouse* (Contel Rico, 2011).

Por tal motivo se decide construir un repositorio *data warehouse* como primera fase de la iniciativa de Inteligencia de Negocios.

3.2 Contexto del problema

La consolidación de los datos de la cartera de clientes en un *data warehouse* permite tener una visión de 360 grados de este, el *data warehouse* es la base para impulsar esta iniciativa de Inteligencia de negocio enfocada al cliente.

La iniciativa consiste en crear un *data warehouse* el cual es alimentado de 4 bases de datos de la institución bancaria donde se encuentra información básica del cliente, saldos, créditos entre otras clasificaciones.

Primero se extraen los datos de las cuatro bases de datos fuente de la institución bancaria, depositándolos en archivos planos (*image copies*) y usando el formato *UTF8*, estos archivos se colocan en una ruta especifica de un servidor, la institución bancaria está a cargo de esta tarea. Posteriormente la herramienta *ETL* (*AB Initio*) toma los *image copies* y se procesan aplicando las transformaciones a los datos, dichas transformaciones se indican en el documento de mapeo, del cual se habla en la Sección 3.5.1. Una vez listos los datos la herramienta *ETL* genera archivos que contienen los datos procesados, también se hace uso del formato UTF8. Finalmente se cargan en al *data warehouse*, lo descrito anteriormente son los pasos principales de la implementación (Figura 1). Así se crea el *data warehouse*, creando una ventana única a los datos analíticos.

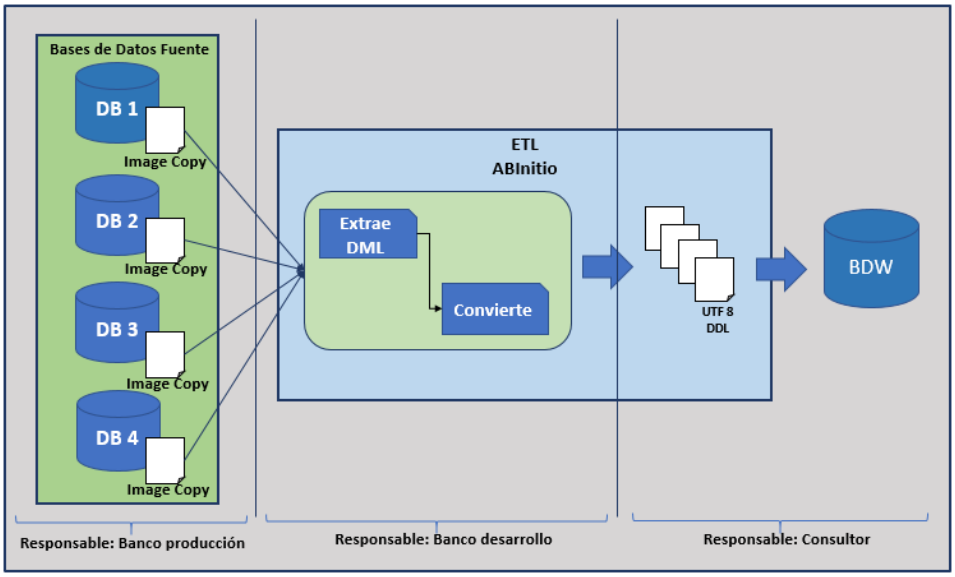

Figura 1 - Diagrama de implementación de data warehouse.

3.3 Modelado de Negocio

Para la Fase I de este proyecto se contemplan 4 bases de datos como fuente de información las cuales se listan aquí:

- Base de datos 1: Para información bancaria fundamental del cliente, entendemos por fundamental dirección, nombre, apellidos, número de cuenta por mencionar.
- Base de datos 2: Para información crediticia detallada del cliente.
- Base de datos 3: Para información detallada de tarjetas de crédito.
- Base de datos 4: Para información del buró de crédito.

El desarrollo contempla para la primera carga la base de datos 1 de la cual se tienen 15 tablas fuente y 990 atributos fuente en total, basados en la información proporcionada además se consideran 1,992 atributos con los cuales se alimenta el *data warehouse*.

3.3.1 Definición de conceptos clave

Como parte del perfilado de datos de servicios financieros teniendo en cuenta que el perfilado de datos se refiere al análisis de datos de los sistemas para entender su contenido, estructura, calidad y dependencia (Loshin, Data prix knowledge is the goal, 2012), se definen 9 entidades principales dentro de la primera fase, se le llama entidad a la clasificación que se le da a los datos, entre las entidades que se encuentran contempladas para la primera fase están *involved party*, a*rragement*, *condition*, *product*, *location*, *classification*, *businees direction item*, *event* y *resource item* (Figura 4).

Figura 2 - Entidades del perfilamiento de datos de servicios financieros.

Algunas de las descripciones de las entidades del perfilamiento de datos de servicios financieros se presentan a continuación:

Involved Party – Información básica de todos los participantes sobre los cuales se desea mantener información (clientes, empleados, grupos, etc.) y el rol que desempeñan.

Location – Información de contacto y domicilio (correo electrónico, teléfono, dirección, etc.) relacionado a clientes o empleados.

3.4 Necesidades del negocio

Con la finalidad de proponer una solución adecuada a la necesidad de la institución bancaria que es ofrecer productos nuevos y personalizados a sus clientes, el objetivo del *data warehouse* es cargar cuatro sistemas de información de clientes de la institución bancaria, los cuales tienen detalle de la información de estos. Para ello se llevan a cabo las siguientes etapas en el ciclo de vida del software:

- Realizar perfilado de datos.
- Elaborar el mapeo de los sistemas fuente.
- Hacer el desarrollo de los procesos de carga.

3.5 Data warehouse

Un *data warehouse* es una base de datos corporativa que se caracteriza por integrar y depurar información de una o más fuentes distintas, para luego procesarla permitiendo su análisis. El término *data warehouse* fue acuñado por primera vez por Bill Inmon, y se traduce literalmente como almacén de datos (S.L., 2007).

Para comprender el concepto de *data warehouse*, es importante entender cuál es su proceso de construcción, *ETL* (Extracción, Transformación y Carga) es una de las principales herramientas que apoyan el desarrollo de este.

Extracción: obtención de información de las distintas fuentes tanto internas como externas.
Transformación: filtrado, limpieza, depuración, homogeneización y agrupación de la información.
Carga: organización y actualización de los datos y los metadatos en la base de datos.

En un sistema de producción, el dato se actualiza con cada nueva transacción, el valor anterior se pierde y el dato se actualiza constantemente. Los sistemas de producción raramente conservan el historial de los valores de este dato.

En un *data warehouse*, el dato no debe actualizarse nunca, este sistema almacenará así el historial. Entonces se debe asociar una referencia de tiempo al dato a fin de poder identificar un valor particular en el tiempo (López Bernal, 2002). Por tal motivo se aplica el proceso *CDC* por sus siglas en inglés *Change Data Capture*.

3.5.1 Mapeo de datos

Mapeo de datos es el proceso de relacionar campos de un archivo fuente a sus campos destino, es considerado el paso más importante en el diseño de un *data warehouse* e impacta en el éxito o el fracaso del proyecto (UI Haq, 2006).

El resultado del proceso es el documento de mapeo y contiene información de los campos fuentes, campos destino, transformaciones de los campos e información de las reglas de negocio.

El documento de mapeo funge como documento de requerimiento y sirve para la creación de escenarios de prueba, mapeo es el proceso de análisis, alcance y calibración entre los datos del sistema fuente para adecuarlos apropiadamente a la estructura del sistema destino.

El documento de mapeo se encuentra dividido en columnas con información del sistema fuente y transformación/sistema destino, las columnas más significativas para efecto de pruebas se mencionan enseguida:

A. Sección sistema fuente del documento de mapeo:
   - *Source File/Table.*
   - *Source Field.*
   - *Source Definition.*
   - *Source Datatype.*
   - *Source Comments.*

La Figura 3 es un ejemplo de la visualización de las columnas fuente.

D	I	J	M	
KEY_GROUP	SOURCE_FILE (TABLE)	SOURCE_FIELD	SOURCE_DATATYPE	
830	T001	IDEDIREC	CHARACTER 2	E
830	T001	NOMCALLE	CHARACTER 24	
830	T001	NUMCALLE	CHARACTER 7	
830	T001	TIPOVIVI	CHARACTER 2	E
830	T001	ENTRADA	CHARACTER 2	E
830	T001	PISO	CHARACTER 2	E
830	T001	DEPARTAM	CHARACTER 4	
830	T001	COLONIA	CHARACTER 40	
830	T001	DELMUNI	CHAR 3	

Figura 3 - Documento de mapeo fuente.

B. Sección sistema destino del documento de mapeo:
   - *Target Transformation.*
   - *Target Table.*
   - *Target Attribute.*
   - *Target Datatype.*
   - *Target Occurs Name.*
   - *Target Occurs Rule.*
   - *Target Comments.*

La Figura 4 es un ejemplo de la visualización de las columnas destino.

DATA_QUALITY_OBSERVATION	TARGET_TRANSFORMATION	TARGET_TABLE	TARGET_ATTRIBUTE
	TRIM SPACES THEN STRAIGHT MOVE	LO.DAT	LO_SOURCE_SYSTEM_UNIQU
	TRIM SPACES FOR ALL THEN:NOMCALLE	LO.DAT	LO_SOURCE_SYSTEM_UNIQU
	TRIM SPACES FOR ALL THEN:NOMCALLE	LO.DAT	LO_SOURCE_SYSTEM_UNIQU
	TRIM SPACES THEN STRAIGHT MOVE	LO.DAT	LO_SOURCE_SYSTEM_UNIQU
	TRIM SPACES FOR ALL THEN:NOMCALLE	LO.DAT	LO_SOURCE_SYSTEM_UNIQU
	TRIM SPACES FOR ALL THEN:NOMCALLE	LO.DAT	LO_SOURCE_SYSTEM_UNIQU
	TRIM SPACES FOR ALL THEN:NOMCALLE	LO.DAT	LO_SOURCE_SYSTEM_UNIQU
	TRIM SPACES FOR ALL THEN:NOMCALLE	LO.DAT	LO_SOURCE_SYSTEM_UNIQU

Figura 4 - Documento de mapeo destino.

3.5.2 Herramienta ETL

La tecnología elegida para el desarrollo *ETL* es *Ab Initio*, *Ab Initio* es una plataforma de *Business Intelligence* compuesta por seis productos de procesamiento de datos. Es una poderosa herramienta de procesamiento paralelo basada en *GUI* para la gestión y análisis de datos *ETL* (Team, 2018).

3.5.3 CDC

CDC (Change Data Capture) se refiere al proceso de capturar cambios realizados en la fuente de datos, su objetivo garantizar la sincronía de los datos. Un *data warehouse* debe mantener el historial de cambios, inserción de nuevos registros, actualización y eliminación de los registros son los tipos de cambios que los procesos *CDC* deben detectar en el sistema origen (Kondamuri, 2018).

3.6 ID y Objetivos de las pruebas

El *ID* correspondiente al plan de pruebas es: RAC-PP1.0 (Repositorio Analítico de Clientes Test, Plan de Pruebas, Versión 1.0)

Los objetivos definidos para las pruebas de la creación del *DWH* se mencionan enseguida:

- Levantar o recibir problemas y decidir los caminos viables para su solución durante la ejecución de las pruebas.
- Lograr el 100% de cobertura de pruebas para reducir problemas en las fases de aceptación de usuario y en la puesta a producción.
- Proporcionar soporte a las pruebas de aceptación de usuario para garantizar la fluidez de estas.
- Asegurar que el producto entregado este de acuerdo a los requerimientos de negocio, los documentos de mapeo fungen con este papel. Dentro del proceso *ETL* se consideran los siguientes puntos para validación:

A) Pruebas de ETL

Para este tipo de validaciones se decide contemplar dos escenarios, la validación de datos, validación de integridad referencial y cifras control.

La primera se refiere a la integridad de datos, se prueba que las transformaciones se aplicaron correctamente y no hubo una alteración.

Para el segundo escenario se valida los requisitos de las llaves primarias de ser única y contener valores nulos, por último, las llaves foráneas deben tener el mismo tipo de dato que la clave primaria a la que referencia.

El tercer escenario valida que el número de datos de la fuente sea el mismo que en el destino.

B) Manejo de Errores

Cuando algunos registros no se puedan transformar o cargar en la tabla destino a causa de un formato no válido, valor no valido o no pasan las reglas de validación.

Además, se valida notificaciones cuando se registran errores en el proceso de transformación y carga.

C) CDC Change Data Capture

El *CDC* es el proceso de capturar los cambios de datos incrementales en el *DWH*.

Se consideran los siguientes escenarios para este tipo de validaciones:

- Insertar registros.
- Insertar y actualizar registros.
- Insertar, actualizar y borrar registros.
- Actualizar registros.
- Insertar y borrar registros.
- Actualizar y borrar registros.
- Borrar registros.

3.7 Alcance de Pruebas

El tipo y la secuencia de las pruebas se deciden o definen según las necesidades específicas de cada proyecto. Para las validaciones de esta base de datos *data warehouse* en especifico se establecen los siguientes niveles de pruebas:

A) Pruebas de Integración de Sistema (SIT):

Para el proceso de integración de datos *ETL* (Extraer, Transformar y Cargar) se establece una validación de los pasos para traer los datos de las fuentes origen. Los datos vienen de 4 diferentes bases de datos (base de datos 1, 2, 3 y 4).

El objetivo es confirmar que los datos se extraen, transforman y cargan de acuerdo con las especificaciones aprobadas por la institución bancaria, esto incluye:

- Confirmar que los datos se reciben en el formato correcto.
- Confirmar que los datos caen dentro del rango de valores aceptado, donde se especificó una gama reconocida de valores.
- Confirmar que los datos se están transformando.

Para validar el proceso *CDC*, la carga de datos considera dos ciclos por fuente.

B) Pruebas de Aceptación de Usuario (UAT):

En inglés *UAT (User Acceptance Testing*), son las pruebas formales respecto a las necesidades del usuario, requerimientos y los procesos de negocio (Mera Paz J. , 2016).

Solo se da soporte al equipo de pruebas de la institución bancaria, guiando en la creación de la documentación de pruebas, apoyo en la creación de la estrategia de pruebas.

C) Pruebas de Regresión:

Estas pruebas solo se limitan a la validación de defectos una vez que se hace una nueva carga de datos con la solución.

Las siguientes fases de prueba están fuera de alcance en este trabajo:

D) Pruebas de performance y seguridad:

El plan no contempla en esta etapa este tipo de pruebas, ya que un equipo de pruebas especializado en ello se encarga de la planeación y ejecución.

E) Pruebas Unitarias:

El equipo de desarrollo es responsable de dichas pruebas, este tipo pruebas no están a cargo del equipo de pruebas.

F) Pruebas post producción:

Son realizadas una vez puesto en producción el sistema se realizan validaciones.

G) Pruebas de recuperación y respaldo:

Incluyen recuperación de desastres; gestión de usuarios; tareas de mantenimiento; carga de datos y tareas de migración; comprobaciones periódicas de vulnerabilidades de seguridad; pruebas de aceptación contractual y normativa; pruebas alfa y beta (Mera Paz J. , 2016).

3.8 Estrategia de pruebas

Es el enfoque general de alto nivel, si la política y la estrategia de pruebas están definidas, guiaran la planificación, pero si no, se debe pedir que se establezcan y definan.

La institución bancaria cuenta con su propia estrategia de pruebas, todos los proyectos deben desarrollarse con base en ella por temas de políticas.

El caso de estudio se basa en la estrategia de pruebas definida por el equipo de pruebas de la institución bancaria, esta estrategia indica que las actividades dentro del proceso fundamental de pruebas están divididas en: *planning*, *preparation*, *execution* y *analysis & reporting* además indica el rol del o los responsables de estas actividades.

Adicional determina las tareas que deben desarrollarse en cada una de las divisiones tal es el caso de *test cases*, *test scenarios*, *requirements*, *traceability matrix* y de acuerdo a la estrategia se tienen los niveles de pruebas *user acceptance testing*, *system integration testing* y *system testing*, esta descripción se puede observar en la (Figura 5).

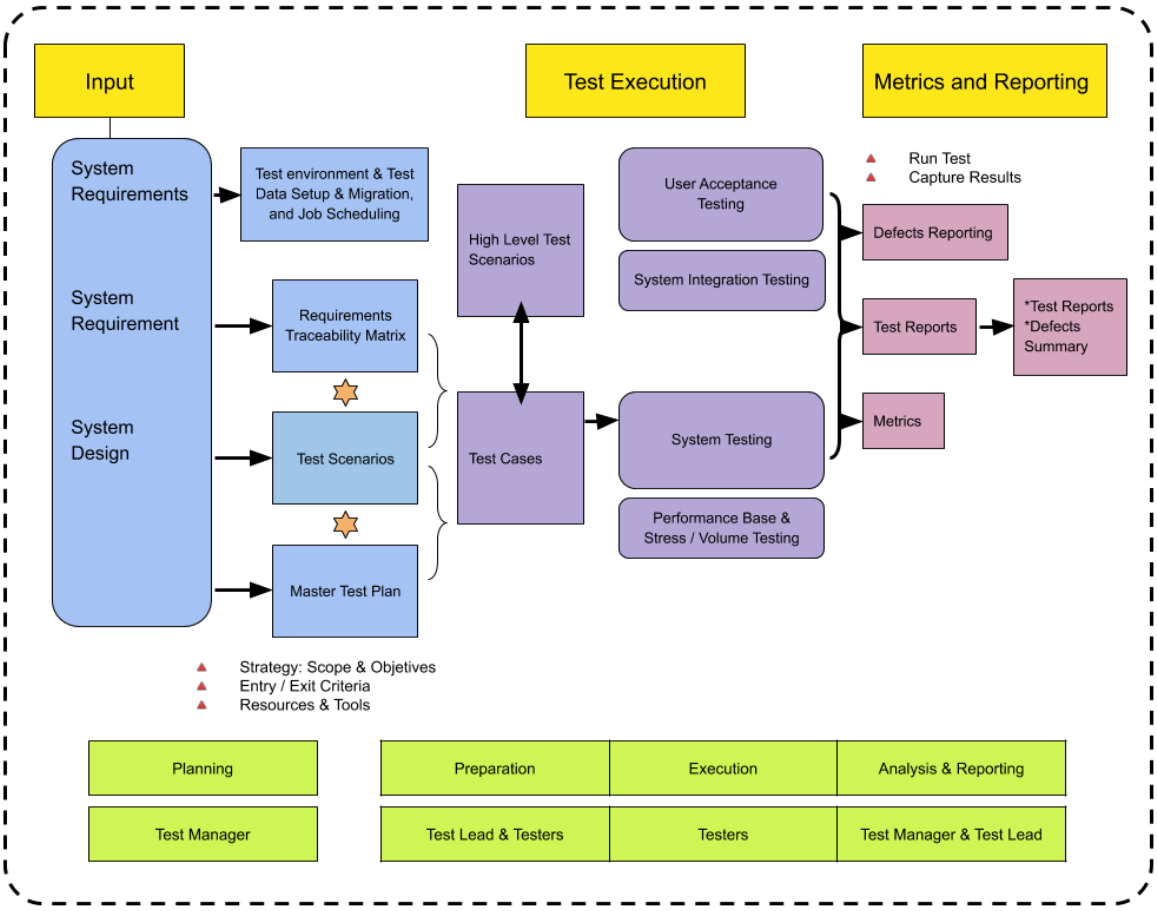

Figura 5 - Diagrama de actividades dentro de la estrategia de pruebas.

3.9 Niveles de pruebas y fases

Existen cuatro tipos de nivel de pruebas: pruebas de componentes, pruebas de integración, pruebas de sistema y pruebas de aceptación, cada uno de ellos tiene sus propios objetivos, los niveles de prueba se definen dependiendo del proyecto. Para probar el repositorio analítico de clientes se establecen dos niveles de prueba, pruebas de integración y pruebas de aceptación de usuario.

3.9.1 Proceso / logística de las pruebas de integración

Las actividades son asignadas de acuerdo con el rol que se desempeña dentro del proyecto, la institución bancaria cuenta con una contraparte para cada uno de los roles con la finalidad de dar un adecuado seguimiento a las actividades y ejecuciones.

En este trabajo hay dos líderes, el líder de pruebas consultor y el líder de pruebas de la institución bancaria, cada uno de ellos tienen asignadas diferentes tareas o actividades lo cual es importante definir y documentar.

El líder de pruebas consultor es responsable de definir el alcance de pruebas, crear un plan de pruebas, revisar casos de prueba, revisar resultados de los casos de prueba, realizar reporte de ejecución, dar seguimiento a los defectos y realizar un reporte final.

El líder de pruebas de la institución bancaria es responsable de coordinar la creación de los datos de prueba con el equipo de *TI* de la Institución Bancaria, revisar casos de prueba y dar aprobación, revisar y dar aprobación del plan de pruebas.

3.9.2 Proceso / logística de las pruebas de aceptación de usuario

Como parte del proceso de pruebas de aceptación de usuario la institución bancaria identifica los siguientes puntos de los cuales están a cargo:

- Identificar quien o quienes son los responsables de la ejecución de las pruebas, considerando áreas técnicas y de negocio.
- Soporte por parte del *Project Manager* de la institución bancaria para asegurar la participación en las pruebas de acuerdo con el calendario acordado de los responsables.
- Manejo de datos de prueba. Para las pruebas de aceptación de usuario se recomienda utilizar datos con las mismas características de producción, por lo cual la Institución Bancaria está a cargo del proceso para obtener dichos datos y cumplir con los requerimientos de seguridad establecidos.

El líder de pruebas consultor es responsable de brindar soporte en todo el proceso de pruebas. En caso de ser necesario brindar apoyo en la creación de los entregables, actualizar plan de pruebas, revisar casos de prueba, revisar resultados de los casos de prueba y realizar reporte final.

El líder de pruebas de la institución bancaria es responsable de identificar y definir el usuario de negocio que va a dar el visto bueno en las pruebas de aceptación. Definir en conjunto con el líder de pruebas consultor el alcance

de esta fase, coordinar la creación de los datos de prueba con el equipo de tecnologías de la Información de la institución bancaria, coordinar la ejecución, subir y actualizar casos de prueba en *ALM*, revisar casos de prueba, revisar resultados de los casos de prueba, obtener el visto bueno por parte del usuario de negocio, finalmente dar aprobación y firma de aceptación de esta fase de pruebas.

El equipo de pruebas de la institución bancaria es involucrado desde la fase previa a *UAT* (*SIT*) con la finalidad de tener todo lo que la institución bancaria necesita para obtener el visto bueno del usuario de negocio (Matrices, Evidencias, documentos, etc.).

El proceso de pruebas se encuentra dividido en 4 fases: planeación preparación, ejecución y reporteo, durante este proceso existen 3 actores principales, el líder de pruebas consultor, líder de pruebas de la institución bancaria y usuario de negocio de la institución bancaria. A cada uno de ellos se le asignan las actividades definidas para este tipo de pruebas, algunas de las tareas serán compartidas.

El flujo del proceso de pruebas de aceptación de usuario va desde preparar los escenarios de negocio hasta la revisión y firma de visto bueno, este diagrama representa el proceso que se sigue en las pruebas de aceptación (Figura 6):

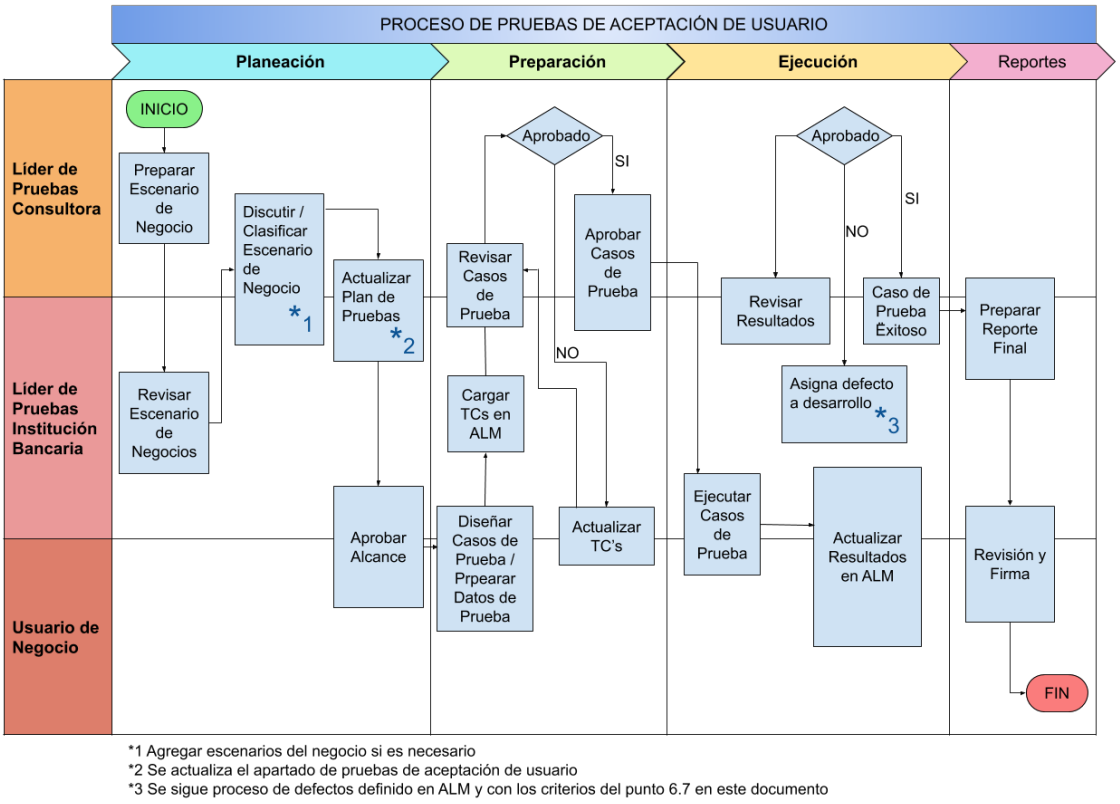

Figura 6 - Flujo de actividades de las pruebas de aceptación de usuario.

3.10 Áreas de enfoque de prueba

Las pruebas se enfocan en el subsistema *ETL*, que se refiere al proceso de Extracción Transformación y Carga. Las áreas que se contemplan son: continuidad de procesamiento, exactitud y confiabilidad.

Esto se logra con una validación punta a punta, es decir se ejecuta el proceso *ETL* y se verifica que no haya interrupciones y una vez concluido el flujo se validan únicamente los datos.

3.10.1 Pruebas Funcionales

La función de un sistema o componente se puede definir respondiendo la pregunta '¿Qué hace?'. Las funciones se describen normalmente en la especificación de los requerimientos o en los casos de uso. Las pruebas funcionales se basan en estas funciones descritas en los documentos de requerimiento, consideran el comportamiento en específico y suele aplicar la técnica *black-box* (Graham, Veenendaal, Evans, & Black, 2008).

Black box testing o pruebas de caja negra, es una técnica o método de pruebas en el cual la estructura interna, diseño o implementación del objeto probado no es conocida por la persona que ejecuta las pruebas (Ostrand, 2002).

Los tipos de pruebas funcionales para este proyecto son: conversión y manejo de errores, todos ellos se llevan a cabo en las fases de pruebas *SIT*.

3.10.2 Pruebas estructurales

Al hablar de la estructura del sistema se considera este tipo de pruebas, regularmente se considera la técnica de caja blanca. La técnica de caja blanca *white-box* en inglés se centra en lo que pasa dentro del sistema (Graham, Veenendaal, Evans, & Black, 2008).

En este plan de pruebas se consideran las validaciones de *Job stream* como prueba estructural, la cual se lleva acabo para la fase de *SIT*.

Un *job stream* consiste en una secuencia de *jobs* que van a ser ejecutados juntos, con tiempos, y otras prioridades que determinan los procesos (IBM, 2019).

3.11 Criterios de entrada/salida

En esta sección se detalla la información general de la fase de prueba *SIT*, los criterios de entrada/salida, y la lista de los productos requeridos a producir.

A) Criterios de Entrada

Dentro de los criterios de entrada para la fase de *SIT* se encuentra la aprobación del plan de pruebas y documentos de mapeo, así como la firma de estos documentos.

Las matrices de trazabilidad y casos de prueba deben de ser aceptados y aprobados, también se debe de contar con un calendario de pruebas publicado. Las pruebas unitarias deben haberse completado sin defectos severidad 1 o 2 abiertos y en caso de haber defectos severidad 3 y 4 que debe existir un plan de acción.

Respecto a los ambientes de pruebas deben cumplir con las validaciones correspondientes y asegurar el funcionamiento correcto, esto incluye los archivos de datos ubicados en el servidor unix, por último, asegurar que las conexiones estén operando correctamente.

Todos los requerimientos de datos listos en el ambiente de pruebas (Integración), el proceso de *configuration* y *change management* está definido para el manejo de versiones para *fixes* o cambios y proceso de defectos definido.

Los recursos necesarios para ejecutar los casos de prueba están definidos e identificados. Accesos a herramientas de prueba listos.

B) Criterio de Salida

Todos los casos de prueba han sido ejecutados y los resultados documentados, casos de prueba y resultados comentados en la matriz de trazabilidad.

Todos los defectos han sido documentados, no hay defectos severidad 1 o 2 abiertos. En caso de tener defectos severidad 3 y 4 abiertos, ya se tiene plan de mitigación y dicho plan ha sido aceptado por todos los involucrados en el proyecto.

Reportes de pruebas finales aceptados y firmados por los responsables de las aplicaciones.

C) Entregables

Calendario de pruebas / Ciclos de prueba, matriz de trazabilidad, casos de prueba detallados en *ALM*, resultados de pruebas, lista de defectos abiertos (en caso de que aplique) y reporte final de pruebas son los documentos que deberán de entregarse durante el proceso de prueba.

3.12 Definición de defectos y tiempos de respuesta

La severidad del defecto es definida por el especialista de pruebas. La severidad del defecto va en relación con la funcionalidad que se está probando, el impacto de la aplicación en general y en el programa. Se representa el impacto que este defecto tendría si hubiera sido visto por un

cliente. En lo que respecta a la severidad no debería cambiar durante la vida de un defecto. La severidad se asigna de acuerdo con los siguientes criterios:

- Severidad 1: El sistema está inutilizable. Función crítica no es funcional y no hay pruebas que se puedan ejecutar. Ninguna solución puede ser identificada y se necesita una respuesta inmediata.
- Severidad 2: La función principal es impactada de manera significativa. El sistema se puede utilizar, pero con altas restricciones. Ninguna solución puede ser identificada para continuar con las pruebas hasta que una corrección esté disponible. Podría causar un impacto significativo en el ciclo de pruebas.
- Severidad 3: Una característica o función no está trabajando correctamente. Las pruebas se pueden seguir y los productos están disponibles. Una solución puede ser identificada.
- Severidad 4: Un problema menor se identifica y no afecta de manera significativa la funcionalidad del sistema. Estos incidentes son generalmente cosmético o accesorios para mejorar la experiencia del usuario. Las pruebas pueden continuar sin impacto.

3.12.1 Definición de prioridades en los efectos

Prioridad de los defectos es a juicio del analista de negocios. Se relaciona con el negocio y sus necesidades:

- Prioridad Alta: El defecto tiene un impacto empresarial crítico (violación de seguridad, requisito legal o se ve afectada la relación con el cliente) o hay un impacto económico. No hay ninguna solución; necesita ser arreglado a la brevedad posible.
- Prioridad Media: El defecto no afecta a los negocios en un asunto importante, o al usuario en su día a día.
- Prioridad Baja: Impacto menor a los negocios. Es necesario resolver si hay tiempo, pero puede ser objeto de una fecha futura determinada por las necesidades de negocio o del cliente.

3.12.2 Tiempos de respuesta

Los tiempos de respuesta esperados se enumeran en la (Tabla 1), es importante señalar que el reloj comienza cuando se asigna el defecto a un

equipo de desarrollo y que sólo se consideran días hábiles, a excepción de los defectos de severidad 1, ya que tienen que ser solucionados tan pronto como sea posible y que dentro de los tiempos de respuesta que se muestran:

Tabla 1 - Tiempos de respuesta para la solución de defectos.

Severidad	Tiempo de Respuesta
1	4 hrs.
2	6 hrs.
3	1 día.
4	Se puede acordar fecha.

3.13 Análisis de riesgos

Esta sección corresponde al levantamiento de riesgos o problemas asociados a las pruebas, también el plan de acción para prevenir y solucionar riesgos, para lo cual se requiere el apoyo de las instancias correspondientes (proyectos, *PMO*, cuerpo directivo). Este tipo de información debe de ser plasmada en un documento independiente para dar el seguimiento correspondiente, sin embargo, deben ser mencionados en todo plan de prueba.

La documentación de los riesgos debe contener información como: un identificador o número de riesgo, el nombre del riesgo, fecha en que se levantó, clasificación del riesgo, probabilidad de que ocurra un error derivado del riesgo, impacto, descripción del riesgo, estrategia de respuesta al riesgo (estatus), plan de mitigación, dueño, fecha de solución, fecha de revisión y plan de contingencia (Tabla 2).

Entre los hallazgos detectados en la etapa previa a la ejecución de pruebas se presenta el siguiente detalle:

Tabla 2 - Descripción de riesgos y plan de contingencia.

Riesgo				
No.	Fecha	Fuente/Clasificación	Probabilidad	Impacto
1	dd/mm/aaaa	*Technical*	2%	3
Descripción				
Actualmente la entrega del ambiente de pruebas para Integración ya tiene un atraso de un mes. Se sigue considerando como riesgo debido al cambio de fechas que hubo en el inicio de las pruebas.				
Respuesta a riesgo				
Status plan	Dueño	Status	Fecha planeada	Fecha actual
Accept	Nombre	*Open*	dd/mm/aaaa	dd/mm/aaaa
Plan de mitigación				
No existe uno identificado. Se debe de entregar el ambiente de acuerdo a lo especificado. Las licencias de *AB Initio* siguen pendientes para el ambiente de Integración.				
Plan de contigencia				
Fecha planeada		Fecha revisada	Fecha actual	
dd/mm/aaaa		dd/mm/aaaa	dd/mm/aaaa	
Descripción				
No aplica.				

3.14 Supuestos

Los supuestos son todas aquellas condiciones o factores suficientes para garantizar el éxito del proyecto en cada una de sus fases, sin embargo, no son controlables por el equipo a cargo. Dicho de otra forma, un supuesto es un dato que asumimos como cierto.

Cuando un supuesto no se cumple puede tener consecuencias que impacten al proyecto, estos son riesgos y en un supuesto puede contener uno o más riesgos (Betancourt, 2017).

En esta sección se registran todas las suposiciones críticas, problemas y limitaciones en el proyecto de Repositorio Analítico de Clientes, se incluye un identificador por cada supuesto (Tabla 3).

Tabla 3 - Supuestos.

Supuesto ID	Descripción
001	Los entornos de prueba estarán preparados según el plan de Ambientes. *BVT* sera ejecutado antes del comienzo de las pruebas para todas las fases.
002	Todos los defectos serán solucionados de acuerdo con los tiempos de respuesta acordados.
003	Todos los documentos a nivel de interfaz se alinearon junto con los requerimientos.
004	Los datos de prueba estarán disponibles para las pruebas de *ETL*.
005	Se proporcionarán los datos de pruebas conforme a lo solicitado para la prueba de *ETL*.
006	Todos los involucrados de desarrollo, deben de cumplir los criterios de entrada para las pruebas de Integración
007	La institución bancaria extenderá el apoyo para la preparación de datos de prueba.
008	El equipo de pruebas será capacitado en la herramienta *AB initio*
009	La institución bancaria será responsable del contenido de los archivos planos. (Calidad de Datos)

3.15 Tareas del equipo de pruebas

Para la definición de las actividades se divide en dos fases: una es previo a la fase de pruebas de integración a la cual se le denomina pre-SIT y la segunda se refiere a la fase *SIT*. Siguiendo la estrategia de pruebas a continuación se detallan las tareas y los propietarios de cada una de ellas: (Tabla 4 y Tabla 5).

Tabla 4 - Tareas detalladas en la fase Pre-SIT.

Fase	Nombre de la tarea	Dueño
Pre-SIT	Preparar calendario de SIT del plan de proyecto.	Líder de pruebas.
	Crear plan de pruebas.	Líder de pruebas consultor.
	Revisión del plan de pruebas.	Responsable de la aplicación, PM, Equipo de Pruebas, Analistas de Negocio e Institución Bancaria.
	Crear escenarios y casos de prueba.	Líder de pruebas y equipo de pruebas consultor.
	Revisión y cierre de escenarios y casos de prueba.	Equipo de pruebas, Dueños de las aplicaciones, PM, Analistas de Negocio e Institución Bancaria.
	Crear ambientes de prueba para integración para todas las aplicaciones incluyendo detalles de HW, SW.	Institución bancaria o Design Delivery.
	Disponibilidad de los datos de prueba para integración.	Institución Bancaria.
	Accesos a los ambientes, DBs y aplicaciones.	PM e Institución Bancaria.
	Resultados pruebas unitarias de acuerdo con los criterios de entrada y salida definidos.	Dueños de las aplicaciones.
	ALM & RTC acceso para todos los involucrados en el proyecto.	Líder de pruebas e institución bancaria.
	Ejecutar el *sanity test* del ambiente de pruebas.	Dueños de las aplicaciones y equipo de pruebas.

Tabla 5 - Tareas detalladas en la fase SIT.

Fase	Nombre de la tarea	Dueño
SIT	Realizar la ejecución de pruebas de integración de acuerdo con el plan maestro de pruebas y crear reporte de ejecución y defectos.	Responsable de pruebas (consultor).
	Solución de defectos de acuerdo con los SLAs.	Responsables de las aplicaciones, Program Management, Test Management, Institución bancaria.
	Refrescar datos, refrescar el ambiente (En caso de ser necesaria).	Design Delivery e institución bancaria.
	Realizar el RCA para los defectos (En caso de ser necesaria).	Dueños de las aplicaciones, Program Managment, Equipo de pruebas.
	Identificar limitaciones conocidas, lecciones aprendidas y publicarlas.	Dueños de las aplicaciones, Program Managment, Equipo de pruebas.

3.16 Roles y responsabilidades

Existen diferentes roles que participan en un proyecto de pruebas, para efectos de este documento se contemplan tres roles: desarrollo, líder de pruebas consultor y líder de pruebas de la institución bancaria cada uno de ellos tienen responsabilidades diferentes.

Se contemplan 4 fases en la etapa de pruebas: planeación, preparación, ejecución y reportes, a cada una de ellas corresponde una seria de responsabilidades las cuales son asignadas a un rol en particular ya sea como actividad primaria o secundaria.

La Tabla 6 muestra las funciones y responsabilidades durante la ejecución del proyecto repositorio analítico de clientes.

Tabla 6 - Responsabilidades de pruebas y roles.

Actividades	Desarrollo	Líder de Pruebas (Consultora)	Líder de Pruebas (Institución Bancaria)
Planeación			
Plan Maestro de Pruebas		P	S
Pruebas Unitarias	P		
Pruebas de Integración		P	S
Pruebas de Aceptación		S	P
Test Env Support		NA	NA
Preparación			
Pruebas Unitarias	P		
Pruebas de Integración		P	S
Pruebas de Aceptación		S	P
Test Env Support		NA	NA
Ejecución			
Pruebas Unitarias	P		
Pruebas de Integración	S	P	S
Pruebas de Aceptación	S	S	P
Test Env Support		NA	NA
Reportes			
Pruebas Unitarias	P		
Systems Integration Test		P	P
User Acceptance Test		S	S

P = Primario; **S** = Secundario

3.17 Calendario de Pruebas

El manejo del tiempo es fundamental para que marche correctamente un proyecto. No existe la eficacia sin el cumplimiento de metas y plazos. El calendario es una herramienta útil que se ocupa de medir las distintas tareas de un proceso en relación con los plazos previstos (Business, 2019).

Para la elaboración del calendario de pruebas del *data warehouse* se identifican 16 actividades tangibles las cuales se llevan a cabo para la validación de las 4 bases de datos fuente, tales como: entregables, planeación, *sign off* del plan de pruebas, entrenamiento, *sign off* de la matriz de prueba, carga de casos de prueba en *ALM*, datos de prueba o archivos, pre-*TRR*, *TRR* formal, configuración de ambiente y *BVT*, ejecución de pruebas integrales, ejecución de pruebas de regresión, ejecución de pruebas de

aceptación de usuario, reporte de pruebas, métricas y *go live*. Cada una de ellas es representada con un símbolo o un color específico dentro del calendario.

La Figura 7 muestra un ejemplo de la elaboración de esta herramienta:

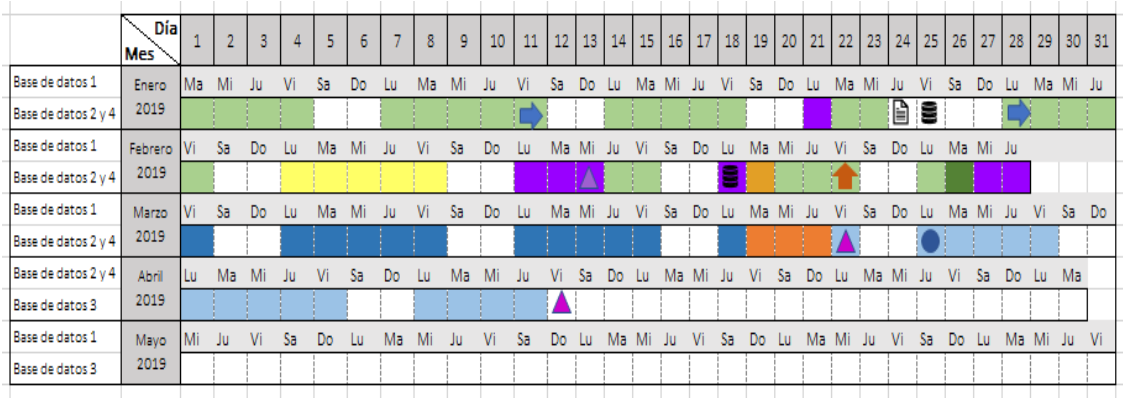

Figura 7 - Calendario de actividades de pruebas.

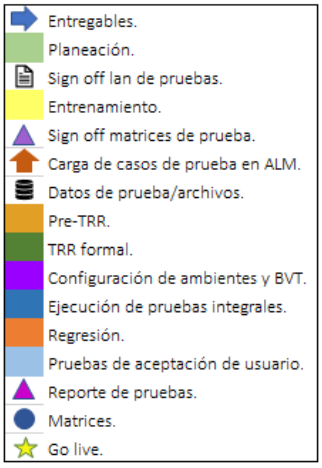

Figura 8 - Simbología y colores del calendario.

3.18 Principales hitos

Los hitos son tareas que simbolizan un logro importante, una forma de conocer el avance del proyecto y se usan para monitorear la ejecución de este. Por tanto, es recomendable colocar algunos hitos dentro del plan solamente como señal de que se llega a un punto importante en el proyecto. Estos hitos sirven como herramientas de comunicación para los responsables de la empresa y las personas encargadas de realizar el proyecto.

La Tabla 7 muestra los hitos más importantes y la Tabla 8 los responsables de cada uno de ellos:

Tabla 7 - Hitos más importantes

Hitos	enero'19	febrero'19	marzo'19	abril'19	mayo'19
Entrega plan de pruebas a la institución bancaria para revisión.	11-ene				
Entrega de matrices al equipo de pruebas de la institución bancaria para revisión.	28-ene				
Sign off plan de pruebas.	30-ene				
Sign off matrices de pruebas - base de datos 1.		13-feb			
Configuración del ambiente y BVT.	21-ene	18-feb			
Archivos de datos de prueba - base de datos 1.	25-ene				
Carga de casos de prueba en ALM - base de datos 1.		22-feb			
Ejecución formal de TRR - base de datos 1.		26-feb			
Ejecución de pruebas de integración base de datos 1.		29-feb			
Ejecución de pruebas de aceptación base de datos 1.			23-mar		

Actividad terminada	
Actividad en progreso	
Actividad con atraso	

Tabla 8 - Responsables de los hitos.

Responsables de los hitos	Hitos
Consultora	Entrega plan de pruebas a la institución bancaria para revisión.
Consultora	Entrega de matrices al equipo de pruebas de la institución bancaria para revisión.
Institución bancaria	*Sign off* plan de pruebas.
Institución bancaria	*Sign off* matrices de pruebas - base de datos 1,2,3,4.
Ambos	Configuración del ambiente y *BVT*.
Institución bancaria	Archivos de datos de prueba - base de datos 1,2,3,4.
Consultora	Carga de casos de prueba en *ALM* - base de datos 1,2,3,4.
Consultora	Ejecución formal de *TRR* - base de datos 1,2,3,4.
Consultora	Ejecución de pruebas de integración base de datos 1,2,3,4.
Institución bancaria	Ejecución de pruebas de aceptación base de datos 1,2,3,4.

3.19 Recursos necesarios

Al hablar de recursos para la fase de pruebas se consideran dos aspectos importantes, los ambientes de pruebas y los recursos humanos, la definición de los ambientes de pruebas va enfocada a las características que deben de cumplir para lograr que sea lo más parecido al ambiente productivo. En cuanto a los recursos humanos se refiere al equipo de personas que estará llevando a cabo las pruebas, cada una de ellas juega un rol específico para cumplir los objetivos del plan.

3.19.1 Ambientes de pruebas

Los ambientes de prueba son una parte de la fase de ensayos en el ciclo de producción. La fase de integración combina estos cambios en el entorno de integración en preparación para el ambiente de prueba, donde se prueban estos cambios.

Un ambiente de prueba a veces es el mismo que el de desarrollo e integración y se hace para comparar tantos entornos de producción como sean posibles para la exactitud en las pruebas.

Los ambientes de prueba pueden ser utilizados para actividades de pruebas de aceptación del usuario. Se suelen utilizar configuraciones de

hardware similares a las del entorno de producción, con sujetos de prueba que proporcionan una previsión del rendimiento (Zeske, 2018).

Los ambientes de prueba idóneos y óptimos mejoran la eficiencia de la etapa de pruebas, el equipo de pruebas define de 3 ambientes de prueba para este proyecto:

- Ambiente de Integración para *SIT*.
- Ambiente de *UAT* para las pruebas de aceptación de usuario.

El recuadro que se muestra a continuación muestra la capacidad base que se necesita para cada ambiente de acuerdo con el equipo de Arquitectura y Desarrollo. Se contempla características de servidor (Tabla 9) y base de datos (Tabla 10).

Tabla 9 - Características Servidor.

Ambiente	Caracteristicas
Desarrollo	8 CPU; 64 GB RAM, 1.2 TB Disco - 200 GB para almacenamiento de datos fuente. 600 GB para espacio de trabajo, 200 GB para datos destino de 2 ciclos.
SIT	4 CPU; 32 GB RAM, 2 TB Disco.
UAT	8 CPU; 64 GB RAM, 10 TB Disco - 4 TB para almacenamiento de datos fuente. 4 TB para espacio de trabajo. 2 TB para carga inicial.
Producción	16 CPU; 128GB RAM, 10 TB Disco - 4 TB para almacenamiento de datos fuente. 4 TB para espacio de trabajo. 2 TB para carga inicial.

Tabla 10 - Caracteristicas Base de Datos.

Ambiente	Caracterisitcas
Desarrollo	1.2 TB Disco - 400 GB de almacenamiento. 400 GB para inidices y rollback, 400 GB para espacio de desarrollo
SIT	2 TB Disco
UAT	6 TB Disco - 2 TB de almacenamiento, 2TB para indices y rollback, 2 TB para ejecutar multiples ciclos
Producción	8 TB Disco - 2 TB de almacenamiento, 2 TB para indices y rollback, 2 TB para 1 año de carga histórica

Para cada ambiente se tiene hardware, software y herramientas definidas, además de una serie de pruebas específicas a ejecutar. Así mismo también existe un responsable que se encarga de administrar la instalación de los ambientes y certificar que están listos para su uso (Tabla 11).

Tabla 11 - Características de los ambientes de prueba.

Nivel de Pruebas	Ambiente - Hardware, Software y herramientas	Pruebas a ejecutar	Responsable
Pruebas Unitarias	Ambiente de desarrollo	Pruebas unitarias de los módulos impactados.	*Design Delivery* e Institución Bancaria
Pruebas de Integración	SIT ambiente de pruebas: *AB Initio–ETL Analyser*, *ETL Domain Server*, *ETL Engine* Base de datos Oracle Fuente de datos de todos las bases de datos fuente que están en el alcance. Unix Server	Pruebas de Integración. *ETL* y el proceso de obtención de datos de las fuentes que están en el alcance	*Design Delivery* e Institución Bancaria
UAT / Pre production	UAT ambiente de pruebas: *AB Initio–ETL Analyser*, ETL *Domain Server*, *ETL Engine* Base de datos Oracle Fuente de datos de todos bases de datos fuente que están en el alcance. Copia de datos productivos. Unix Server	Pruebas de aceptación de usuario	*Design Delivery* e Institución Bancaria

3.19.2 Planeación de Recursos

La estimación de los recursos del equipo de pruebas se presenta al *PM* de la Consultora y la institución bancaria, la cual se describe en esta sección.

A continuación, se muestra la estructura propuesta para el equipo de pruebas de Repositorio Analítico de Clientes:

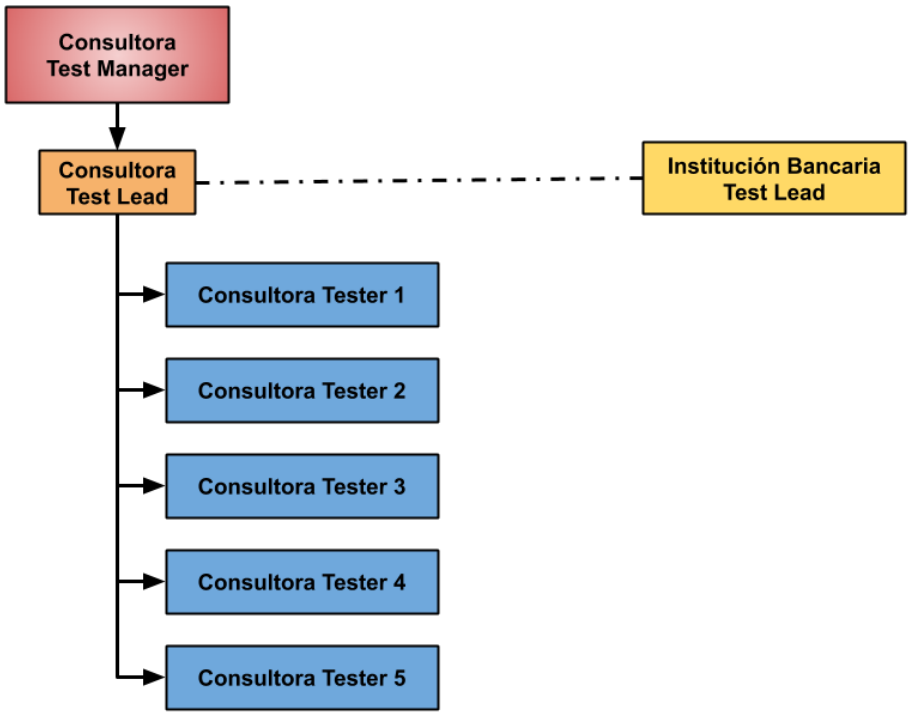

Figura 9 - Organigrama del equipo de pruebas

Adicional la institución bancaria asigna un líder de prueba como contraparte del líder de pruebas de la consultora. Dicho líder es el punto de contacto de la consultora para discutir todos los puntos relacionados con las pruebas del proyecto Repositorio Analítico de Clientes, como es el caso de la obtención de datos de prueba. También se considera un *Test Manager* por partes de dicha institución con el objetivo de monitorear las actividades de pruebas y asegurar que los entregables sean de calidad. El *Test Manager* y el líder de pruebas de la institución bancaria son los responsables de la ejecución y planeación de las pruebas *UAT*, además de introducir al cliente en las nuevas implementaciones.

3.20 Estrategia de datos de prueba y herramientas

La estrategia de datos contempla el análisis de la información que se necesita para poder llegar a los escenarios propuestos por el equipo de pruebas y así poder ejecutar los casos de pruebas.

3.20.1 Estrategia de datos

El uso de datos de prueba varía dependiendo de la fase en que se realiza la prueba. La siguiente tabla define el uso de datos de prueba:

Tabla 12 - Características de los datos de prueba

Fase de Prueba	Uso de Datos y Fuente
Pruebas de Integración	Las necesidades de datos de prueba especificados, junto con los casos de prueba de *SIT*. Datos de referencia suministrado y poblado por la Institución Bancaria. Los datos deben de venir de cada fuente de acuerdo a las pruebas definidas. En este caso la Institución Bancaria será responsable de proporcionar los archivos planos. Los datos deben de ser entregados con cifras control.
Pruebas de Aceptación de usuario	Las necesidades de datos de prueba especificados, junto con casos de prueba *UAT*. Datos de referencia suministrados y poblados por la Institución Bancaria . Todas las bases de datos origen para cargar los datos de prueba son productivos. En este caso la Institución Bancaria será responsable de proporcionar los archivos planos. Los datos deben de ser entregados con cifras control

Test Manager/Líderes tienen las siguientes responsabilidades con los datos:

- Identificar las necesidades de datos para las pruebas de Integración y captura de los mismos en las matrices de datos (si es necesario).
- Asegurar la asignación de datos de acuerdo con los requerimientos definidos.
- Asegurarse que los datos migrados están disponibles para las pruebas de integración y aceptación.
- Administrar las cuestiones relacionadas con los datos.
- Será obligatorio que todas las organizaciones participantes en la prueba hayan firmado acuerdos de confidencialidad, y que han hecho a todos los miembros del equipo de prueba conscientes de esta responsabilidad.

- El líder de pruebas de Banorte es responsable de coordinar la creación de datos y de que estos cumplan las medidas de seguridad necesarias para datos sensitivos (enmascaramiento, datos *dummy*, etc).
- Banorte será el responsable de proporcionar el proceso para obtener los datos de prueba.

3.20.2 Herramientas de prueba

Existen herramientas que proveen soporte para la gestión del proceso, la gestión de las pruebas aplica durante todo el ciclo de desarrollo de software por tal motivo las herramientas de gestión de pruebas deben ser introducidas desde el principio del desarrollo. Una herramienta de gestión de pruebas también puede administrar los casos de prueba, de inicio a fin del desarrollo incluso después de que el software inicie el ciclo de producción (Graham, Veenendaal, Evans, & Black, 2008).

La herramienta utilizada en el proceso de pruebas para este proyecto es *HP Quality Center ALM,* el proyecto Repositorio Analítico de Clientes hace uso de esta herramienta para desplegar la planificación y gestión del proceso de pruebas además de la gestión y elaboración de Informes de defectos.

ALM es una aplicación basada en web usada para todos los aspectos esenciales del plan de prueba de: gestión de pruebas, laboratorio de pruebas y administración de Defectos. Todos los casos de prueba se asignan a un requerimiento o documento técnico.

El uso de *ALM* se utiliza de acuerdo con los lineamientos especificados por la institución bancaria ya que es la herramienta con la que cuenta la institución bancaria y por tal motivo se decide implementarla.

3.20.2.1 Versionamiento de Código

El control de versiones es un sistema que registra los cambios realizados sobre un archivo o conjunto de archivos a lo largo del tiempo de tal manera que sea posible recuperar versiones especificas más delante. La herramienta utilizada en este proyecto es *Rational Team Concert* (*RTC*) de *IBM*.

Es importante saber sobre qué versión de código se están ejecutando las pruebas, ya que los defectos se pueden generar por no probar en la versión adecuada.

El equipo de pruebas no hace uso de esta herramienta, los responsables de su administración son los equipos de *Configuration Management* y *Delivery*.

3.20.3 Herramientas de acceso a datos

La base de datos que se usa en el proyecto es Oracle por este motivo el equipo de pruebas requiere Oracle / *Data architect* / *SQL* para realizar los análisis de las cargas en las bases de datos. Con dichos accesos se busca validar o realizar las siguientes acciones:

- Cuando un proceso está ocasionando bloqueos a la base de datos.
- Poder matar procesos que nos impidan continuar con el proceso que se está ejecutando en la base de datos.
- Saber cuáles son los procesos que están consumiendo más recursos en la base de datos, revisando la actividad que se está realizando en ese momento.
- Cuantos cursores se están abriendo por cada proceso que se está ejecutando.
- Conocer los costos de los procesos que están corriendo para de esta forma poder optimizarlos.
- Revisar los *HOT OBJECTS* de la base de datos.
- Poder revisar el estado de la base de datos.
- Saber cuántos usuarios están conectados en oracle y el número de sesiones por usuario.
- Consulta de Memoria Share_Pool libre y usada.
- Se van a estar ejecutando procesos de carga masiva de Información por lo que necesitamos revisar que está pasando en la base d datos cuando está corriendo el proceso.

Adicional a esto se requiere acceso a la herramienta *ACE* de *AB Initio* y al servidor de Unix.

3.21 Entregables de pruebas

Aquí se listan los documentos que se entregan como parte de la ejecución del plan de pruebas, en ellos se especifica el nombre del documento, la persona que los entrega, la persona que lo recibe, así como la fecha planeada y la fecha en que se entrega el documento. Los documentos establecidos como entregables son: la matriz de pruebas que contienen los casos de pruebas, reporte de errores (defectos), evidencias de las pruebas, reportes de ejecución, y otros documentos de carácter importante para el proyecto.

Los documentos que el equipo de pruebas consultor provee como entregables a la institución bancaria son:

- Plan de pruebas.
- Escenarios de prueba.
- Casos de prueba.
- Matrices de trazabilidad.
- Reporte semanal.
- Resultados de prueba.
- Reporte final de pruebas por fase.
- *Test Readiness Review* (*TRR*).

Una vez entregados se hace una inspección formal del plan de pruebas y de los casos de prueba y los registros se comparten con todas las partes interesadas después de cada revisión.

3.21.1 RTVM

RTVM por sus siglas en ingles *Requirements Traceability Verification Matrix* es un documento que vincula las necesidades del usuario con los requisitos detallados y el proceso de verificación del sistema. El propósito de la *RTVM* es garantizar que todos los requisitos estén asociados con las necesidades del usuario y que todos los requisitos detallados definidos para un sistema se verifiquen mediante pruebas u observación. El *RTVM* es una herramienta para que el equipo de pruebas se asegure de que se incluyan todos los requisitos detallados y que ninguno quede fuera durante el desarrollo y las pruebas del sistema. Los requisitos detallados deben rastrearse hacia las pruebas y hacia atrás a las necesidades del usuario. Los requisitos detallados

están vinculados a las necesidades del usuario sin necesidad de que el usuario quede sin atender.

El contenido del documento *RTVM* de acuerdo con el proyecto debe contener información específica, la cual se clasifica en elementos del artefacto, trazabilidad, especificaciones técnicas y fases de pruebas. Los elementos requeridos son:

- Nombre del documento: Nombre del documento utilizado para la creación de la *RTVM* con su extensión.
- Versión del documento: Identifica la versión del documento utilizado.
- Estatus de requerimiento: Indica el estatus más actual del requerimiento (Diseño, Construcción, Revisión, Finalizado).
- ID referencia/descripción: Referencia que contiene dentro del documento por funcionalidad.
- Descripción funcional: Descripción de cada una de las funcionalidades que contenga el requerimiento.
- Criterios de aceptación del usuario: Criterios por los que el cliente evalué el requerimiento como aceptado.
- Reglas de negocio: Las reglas de negocio por cada criterio de aceptación.
- Sistema: Indica el nombre del sistema para el cual se valida cada funcionalidad.
- Componente: Identificar el módulo (s), programa (s) u otro componente (s) (por ejemplo, los componentes de software, documentación, materiales de capacitación) donde se interactúa el requerimiento.
- *Change Request (CR)*: Identificador del *CR* (sí aplica).
- Tipo: Ingresa "RF" si el requerimiento funcional, ingresa "RNF" si el requerimiento no es funcional.
- Métodos de prueba: A (Análisis), D (demostración), I (Inspección), S / M (Simulación / Modelo), T (prueba).
- ID Casos de prueba: Número de Identificación de Casos de Prueba.
- Descripción casos de prueba: Nombre del caso de pruebas.

De acuerdo con el requerimiento del caso de estudio el documento de requerimiento es el documento de mapeo, cada regla de transformación funge como una regla de negocio y a su vez como un requerimiento. Como se muestra en la Tabla 13 se tiene los datos del artefacto (documento de mapeo).

Tabla 13 - Datos de la sección elementos del artefacto.

Elementos del Artefacto		
Nombre Documento	**Versión del Documento:**	**Estatus del Requerimiento:**
Base Datos Mapping FINAL 20140213_0402	0010	Finalizado

Se agrega la descripción funcional que se va a validar con su respectiva regla de negocio, que para efectos del proyecto es la regla de transformación a verificar durante la ejecución de pruebas como se puede ver en la Tabla 14.

Tabla 14 - Datos de la sección trazabilidad.

Trazabilidad			
ID Referencia /Descripción	**Descripción Funcional**	**Criterios de Aceptación del usuario**	**Reglas de Negocio**
1	Cargar los archivos fuente de base de datos 1 en el DWH.	TBD	Create if is_defined(string_lrtrim(AM_ACCT)) and is_defined(string_lrtrim(AM_USER_DEMO_1)) and (string_lpad(string_lrtrim(AM_USER_DEMO_1),8,"0")) != "00000001"

Lo que procede es agregar las especificaciones técnicas como se ve en la Tabla 15, de acuerdo con la definición de cada campo tipo el valor *RNF* significa requerimiento no funcional.

Tabla 15 - Datos de especificaciones técnicas.

Especificaciones Técnicas				
BD	**Archivo Fuente**	**Tabla**	**CR**	**TIPO**
Base de datos 1	AM left outer join AMBSRL, target IP	IP	N/A	RNF

Por último, se incluyen los valores de los campos de la sección pruebas integrales ya que es el nivel de prueba que se define en el plan de pruebas como se puede notar en la Tabla 16.

Tabla 16 - Datos de pruebas integrales.

Pruebas Integrales			
Método de pruebas	**ID Casos de Prueba**	**Descripción Casos de Prueba**	**Resultado**
T	1	Record count validation between source and target table IP	Paso

3.21.2 Matriz de pruebas

La matriz de prueba es el documento donde se concentran todos los escenarios y casos de prueba que se ejecutan para la validación del sistema. Un caso de prueba es un conjunto de condiciones bajo las cuales se determina que una parte especifica de una aplicación funciona de acuerdo con los requerimientos, y para escribir un caso de prueba se debe tener precondiciones, paso a reproducir, resultado esperado, datos de prueba, etc., mientras que un escenario solo se necesita saber que debe realizar la aplicación. Un caso de prueba responde a la pregunta cómo debe probarse y un escenario responde a la pregunta que se debe probar (Vargas, 2017).

Los casos de prueba se diseñan para:

- Condiciones válidas de entrada/salida.
- Condiciones inválidas e inesperadas de entrada/salida.
- Condiciones límite.
- Condiciones de error.

Los datos que debe contener el caso de pruebas dentro de la matriz de pruebas y de acuerdo con el proyecto son: ID del caso de pruebas, nombre de tabla destino, tipo de validación, escenario de prueba, nombre del caso de prueba, descripción, puntos de validación, resultados esperados, regla de transformación, estatus, *query* de la consulta a la base de datos fuente y *query* de consulta a la base de datos destino.

Para el nivel de prueba *SIT* y control de ejecución de cada caso de prueba se elabora la matriz de prueba que contiene los escenarios y casos que se

ejecutan para validar la integridad referencial, consistencia de los datos y *CDC*, por ejemplo, la Tabla 17 y la Tabla 18 muestran uno de los escenarios representativos de la validación de integridad referencial de datos y su respectivo caso de prueba.

Tabla 17 - Ejemplo de la matriz de prueba SIT Integridad referencial parte 1.

#	Tabla	Archivos fuente	Tipo de validación	Escenario de prueba	Caso de prueba	Descripción
Validación Integridad Referencial						
01AL	**CUST**	**PE**	Integridad y exactitud de los datos	Validar que el registro se creó correctamente en la base de datos destino de acuerdo con la regla de transformación.	Validar regla de transformación Customer / Involved Party Positivo	Validar que se cree el registro en la tabla **CUST** si el campo NUMCLIEN está definido.

Tabla 18 - Ejemplo de la matriz de prueba SIT Integridad referencial parte 2.

Puntos de validación	Resultados Esperados	Regla de Transformación	Paso / Fallo	Consulta en la fuente (query)	Consulta en el objetivo (query)
-Validar el número de Registros entre la Tabla (s) fuente y Destino. -Validar que los datos entre la Tabla (s) fuente y la tabla CUST (Destino) sean los mismos. -Validar la nomenclatura del archivo de carga (archivo que será cargado a la base de datos). -Validar la ruta del archivo de carga.	Mismo número de registros de todos los clientes válidos en el archivo de origen vs el número de registros de la tabla destino CUST. Mismos datos en tabla (s) fuente y tabla destino. La nomenclatura y ruta del archivo de carga son los correctos.	Create if (is_defined(NUMCLIEN) && !is_blank(NUMCLIEN))		SELECT COUNT(DISTINCT(NUMCLIEN)) FROM PE WHERE NUMCLIEN IS NOT NULL	SELECT COUNT(*)FROM CUST WHERE SRC_SYS = 'Base1';

3.21.3 TRR

Por sus siglas en inglés *Test Readiness Review* es un documento sobre el cual se determina si el sistema bajo revisión está listo para proceder a las pruebas formales, al decidir si los procedimientos de prueba están completos y verificar su cumplimiento con los planes de prueba y las descripciones.

Normalmente, se realiza una *TRR* antes de cada elemento de configuración de prueba principal, incluido el hardware y el software, y proporciona a la gerencia la seguridad de que un sistema se ha sometido a un proceso de prueba exhaustivo y está listo para la renovación de la siguiente fase de prueba.

El TRR evalúa los objetivos de la prueba, los métodos y procedimientos de prueba, el alcance de las pruebas y la seguridad y confirma que los recursos de prueba requeridos se han identificado y coordinado adecuadamente para respaldar las pruebas planificadas.

Test Readiness Review (*TRR*) se define ejecutar en cada nivel de prueba (Integración y Aceptación), los parámetros a evaluar con este documento (*TRR*) se muestran en la Figura 10 y Figura 11.

TRR Scorecard

This tab will show the calculated scores for each of the TRR Standard Criteria, and the resulting

Data warehouse dd/mm/aaaa TRR Standard Criteria	Weighting Factor	SCORES FOR TRR STANDARD CRITERIA	TRR SCORE
TRR1-Test Planning	**3.00**	**4.00**	**12.00**
TRR1.1-Test scope, objectives and assumptions have been documented, reviewed by stakeholders and agreed upon	0.43	4.00	1.71
TRR1.2-Test entrance and exit criteria has been documented, reviewed by stakeholders and agreed upon	0.43	4.00	1.71
TRR1.3-All testing tasks are allocated to identified resources	0.43	4.00	1.71
TRR1.4-Problem/defect management procedures, including turnaround time by severity, have	0.43	4.00	1.71
TRR1.5-Technical risks, dependencies and issues have been identified and mitigation plans are in place	0.43	4.00	1.71
TRR1.6-Change management procedures to be followed during test execution are in place	0.43	4.00	1.71
TRR1.7-Test plans have been reviewed by stakeholders, approved, and results are documented.	0.43	4.00	1.71
TRR2-Requirements Traceability Matrix [RTVM]	**1.00**	**4.00**	**4.00**
TRR2.1-All requirements that will be tested are included in the RTVM.	0.33	4.00	1.33
TRR2.2-Test types and methods to be performed for each requirement are documented	0.33	4.00	1.33
TRR2.3-All requirements that will be tested are traced to a test case ID	0.33	4.00	1.33
TRR3-Development is completed and deployed in QA1	**7.00**	**4.00**	**28.00**
TRR3.1-All applications code was deployed in QA	1.40	4.00	5.60
TRR3.2-Design was approval by Architecture Team if apply	1.40	4.00	5.60
TRR3.3-A work product inspection review was performed for code	0.00	4.00	0.00
TRR3.4-Results for Unit Test and System Test as well as defects are documented and evidence is provided to Test Team.	1.40	4.00	5.60
TRR3.5 The BRR was approved	0.00	4.00	0.00
TRR3.6 The SRR was approved	0.00	4.00	0.00
TRR3.7 The PDR was approved	1.40	4.00	5.60
TRR3.8 The CDR was approved	1.40	4.00	5.60

Figura 10 - TRR scorecard sección 1, 2 y 3

TRR4-Test Environment Readiness	**7.00**	**4.00**	**28.00**
TRR4.1-Test environment is available (up and running)?	2.33	4.00	9.33
TRR4.2-A preliminary and final APPScan was ran	0.00	4.00	0.00
TRR4.3-No High or medium vulnerabilities was inyected	0.00	4.00	0.00
TRR4.4-All applications involved have been configured and recompiled correctly with the last code version? Smoke Testing has completed successfully? (Latest code version has been deployed to test environment (SIT))	2.33	4.00	9.33
TRR4.5-BVT was executed correctly	2.33	4.00	9.33
TRR5-Test Data Readiness	**3.75**	**4.00**	**15.00**
TRR5.1-Test data has been documented and created/obtained	0.75	4.00	3.00
TRR5.2-Test data creation/acquisition procedures and schedule are in place and agreed upon	0.75	4.00	3.00
TRR5.3-Data feeds have been identified	0.75	4.00	3.00
TRR5.4-Test data refresh procedures and schedule are in place and agreed upon	0.75	4.00	3.00
TRR5.5-Test data has been loaded and is available in the test environment	0.75	4.00	3.00
TRR6-Test Team Readiness	**3.25**	**4.00**	**13.00**
TRR6.1-All test cases are complete, inspected and results are documented	0.65	4.00	2.60
TRR6.2-Test team members have completed training as documented in project training plan	0.65	4.00	2.60
TRR6.3-Test Execution Plan (S-Curve) has been created	0.65	4.00	2.60
TRR6.4-Test team dependencies on support groups have been communicated and responsibilities agreed to	0.65	4.00	2.60
TRR6.5-Regression test cases have been selected and reviewed	0.65	4.00	2.60
		SCORE >>	51.25

51.25

TRR Scoring Categories

- Less than 85 is considered red (very high risk to deliver the project),
- Greater than or equal to 85 are considered green (low risk to deliver the project).

Test Readiness Execution		
		NO GO

Summary of Open Items in Review Log by Severity	
1- Very High	4
2 -High	4
3 -Medium	6
4- Low	3
Total	17

Figura 11 - TRR socorecard sección 4, 5 y 6

El resultado de la evaluación debe ser igual a 85 o mayor para poder iniciar con la fase de ejecución de pruebas.

3.22 Estimaciones

Para la estimación de ejecución del proyecto se usa la herramienta Curva-S, esta curva, representa en un proyecto el avance real respecto al planificado en un periodo acumulado hasta la fecha. La curva recibe el nombre de "S" por su forma. Normalmente, al principio del proyecto hay una tendencia de costos acumulados crecientes, mientras que estos costos acumulados decrecen hacia el final.

3.23 Defectos

Se declara defecto todo aquello que no cumpla con el requerimiento de negocio y a alguna falla del sistema durante la ejecución. Todo defecto tiene un ciclo de vida, el cual se puede definir como el viaje del defecto por diferentes estados durante su vida útil. Estos estados pueden variar dependiendo de la organización, del proyecto o de las herramientas que utilicemos, aun así los estados podría definirse como lo muestra la Figura 12.

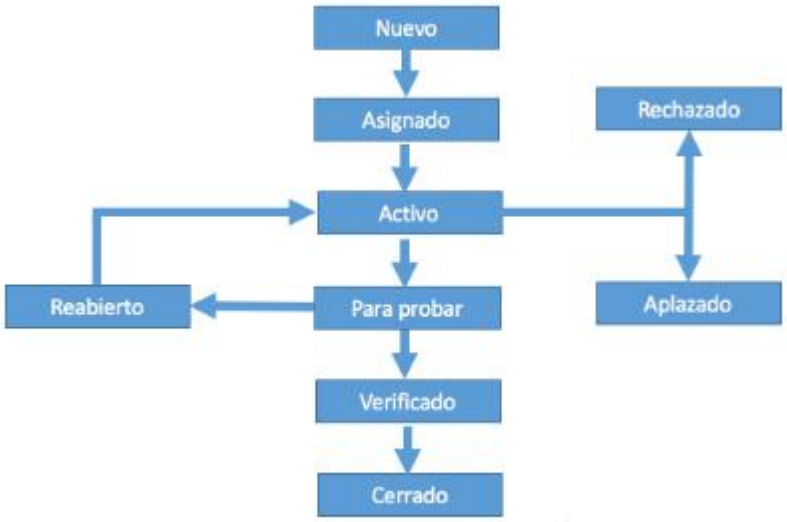

Figura 12 - Ciclo de vida de un defecto.

3.23.1 Proceso de seguimiento de defectos

ALM es la herramienta que se utiliza en el proyecto para seguimiento de defectos.

Los siguientes son los pasos para seguir durante la gestión de defectos con los dueños y su acción se muestran en la Tabla 19.

Tabla 19 - Tareas de gestión de defectos y dueños de las tareas.

Actividad	Responsable
Identificar, plantear y registrar el defecto.	Equipo de pruebas
Asignar y analizar el defecto.	Desarrollo
Actualizar el defecto con hallazgos/análisis.	Desarrollo
Desarrollar la solución.	Desarrollo
Instalar cambios.	Desarrollo/*Configuration Mgmt*
Realizar el test.	Equipo de pruebas
Actualizar el log del defecto.	Equipo de pruebas
Cerrar / Re-abrir defectos.	Equipo de pruebas

3.23.2 Revisión de defectos

Para el seguimiento de defectos se definen los siguientes puntos:

- Durante *SIT* los equipos deben tener juntas diarias para revisar el estatus de los defectos. Equipo de desarrollo, *program management*, *test management*, negocio y los especialistas técnicos de pruebas deben atender la misma.
- Juntas urgentes de defectos deben ser convocadas para los defectos de alta severidad en todas las fases de prueba. Un equipo pequeño del proyecto será convocado para realizar el análisis inicial. Dichas juntas serán convocadas por el Líder de pruebas.
- *Project Manager* son los puntos de escalación para los defectos no solucionados de acuerdo con lo esperado.

3.24 Proceso para reporte de avance

Para el proyecto de repositorio analítico de clientes los reportes son usados para dar seguimiento al avance, este Informe de estatus le informa al equipo de trabajo el estado de la ejecución y los defectos durante ejecución *SIT/UAT.*

El informe resumen de la prueba es el resumen ejecutivo con los resultados de las pruebas. Incluye el análisis de la información y resultados para permitir decisiones de gestión basadas en riesgos sobre la conveniencia de continuar con el siguiente nivel de prueba o para la ejecución del proyecto, o si es necesario realizar más pruebas. Características del reporte:

- Reporte semanal de pruebas.

- Este reporte semanal detalle el avance en las actividades de *SIT/UAT* para el proyecto.
- Reporte diario de defectos.

Adicional el equipo de pruebas va a mandar diariamente el informe del estatus de los defectos. Este reporte se va a manejar para las fases de *SIT/UAT.*

Puntos importantes para considerar en cuanto a los reportes:

1. En la primera línea se mencionan los avances más relevantes.
2. El reporte debe contener número de casos ejecutados, casos exitosos, casos en proceso, casos sin ejecutar y casos fallados.
3. Adicional a las métricas se manejan un estatus en color. Los colores para utilizar y su significado son:
    - Verde - La desviación planeada vs. la real tiene una desviación menor al 5%.
    - Amarillo - La desviación planeado vs. real tiene una desviación mayor al 5% y menor al 25%. La capacidad para completar las pruebas en tiempo se encuentra en riesgo, se necesitan tomar acciones para el cierre exitoso de las pruebas.
    - Rojo - La desviación planeado vs. real tiene una desviación mayor al 25%. Existe un alto riesgo de no terminar las pruebas en tiempo y es necesario buscar el apoyo necesario para lograr resolver los problemas identificados.

3.25 Carta de Salida

Una vez terminada las pruebas de aceptación de usuario y habiendo obtenido un resultado exitoso, se procede a crear un reporte final de pruebas donde se detalla un resumen de las pruebas que contiene resumen de los casos de pruebas (número total de casos ejecutados, exitosos, fallados, bloqueados, en progreso y no ejecutados) y análisis de defecto.

Finalmente, las partes involucradas se reúnen para discutir los resultados y al estar de acuerdo se firma una carta de salida, la cual es un documento donde el cliente firma dando el visto bueno y/o aprobación de los resultados.

Conclusiones

En esta tesina se ha documentado el desarrollo de un plan de pruebas, describiendo cada una de las secciones que lo componen. Además, en el capítulo de caso de estudio se planteó el tipo de pruebas a realizar para validar la integridad referencial y consistencia de los datos de una empresa del sector financiero. Por lo tanto, se cumple el objetivo general planteado en esta tesina.

Adicionalmente, se presentó la información principal del negocio, con base a la cual se hace el análisis para elaborar un plan de pruebas, la planificación hace referencia al desarrollo del plan de pruebas planteado en esta tesina, mientras que la ejecución va en función al caso de estudio descrito en el presente trabajo.

Con la información documentada en esta tesina, se pudo observar que la planificación de las pruebas es uno de los principales puntos para probar software con éxito. En ausencia de un buen plan de pruebas es probable que éstas no se ejecuten en el plazo estipulado e incluso se incrementaría el costo del proyecto al consumir más tiempo y recursos.

Por lo tanto, es evidente que la planificación de las pruebas tiene que estar incluida en el plan de desarrollo general de un proyecto de software.

Finalmente, una contribución adicional que se puede mencionar sobre de esta tesina, es que puede ser de apoyo a los estudiantes del área de ingeniería del software para darles fundamentos sobre la calidad del software.

Glosario

AB INITIO	Plataforma de Bussines Intelligence, se utiliza comúnmente para extraer, transformar y cargar datos.
ALM	Application Lifecycle Management, software es un programa de gestión del ciclo de vida de las aplicaciones que le ayuda a definir, construir, probar y entregar aplicaciones con rapidez y calidad.
BA	Business Analyst.
BDWM	Banking Data Warehouse Model.
BRR	Business Requirement Review.
BVT	Build Verification Test.
DB	Data base
CAR	Customer Analytics Repository.
CBI	Customer Business Intelligence.
CDC	Change Data Capture.
CDR	Critical Design Review.
ETL	Extract Transform Load.
FRs	Functional Requirements.
IBM	International Business Machines Corporation
ID	Identificador
IEEE 829	Instituto de Ingeniería Eléctrica y Electrónica, estándar para documentación de pruebas de software.
ISO 25010	International Organization for Standardization. Modelo de calidad que establece el sistema para la evaluación de la calidad del producto.
Job Stream	Una serie de programas relacionados que se ejecutan en un orden prescrito.
NFRs	Non Functional Requirements.
PDR	Preliminary design review.
PM	Project Management/ Project Manager.
RTC	Rational Team Concert software
RTVM	Requirement Traceability Verification Matrix.

Sanity test	Es un subconjunto de pruebas de regresión.
SIT	System Integration Testing (Pruebas de Integración de Sistema).
SME	Subject Matter Expert.
SOW	Statement of Work.
SQL	Structured Query Language.
SRR	System requirement review.
STAKEHOLDER	Son todas aquellas personas, grupos y entidades que tienen intereses de cualquier tipo en una empresa y se ven afectados por sus actividades.
TESTWARE	Término para denominar todos los elementos producidos durante el proceso de prueba necesarios para la planificación, el diseño y la ejecución de la prueba.
TI	Tecnologias de la Información
TM	Test Manager.
UAT	User Acceptance Testing (Pruebas de Aceptación de usuario).
UTF 8	8-bit Unicode Transformation Format, es un formato de codificación de caracteres Unicode e ISO 10646 utilizando símbolos de longitud variable.

Referencias

Andalucia, J. d. (2019). *Marco de desarrollo de la junta de andalucía*. Obtenido de http://www.juntadeandalucia.es/servicios/madeja/contenido/subsistemas/verificacion/estrategia-pruebas

Betancourt, D. F. (17 de 02 de 2017). *Los supuestos en la metodología de marco lógico.* Obtenido de https://ingenioempresa.com/supuestos-marco-logico/

Business, O. (2019). *OBS Business* . Obtenido de https://www.obs-edu.com/int/blog-project-management/gestion-del-tiempo/que-beneficios-tiene-programar-el-cronograma-de-un-proyecto

Campos Chiu, C. (2015). *LAS PRUEBAS EN EL DESARROLLO DE SOFTWARE.* CDMX: UNAM.

Chavarria, A. (19 de 11 de 2018). *AVANTICA*. Obtenido de https://www.avantica.net/es/blog/pruebas-de-regresion-6-recomendaciones

Cillero, M. (2001). *manuel.cillero.es*. Obtenido de https://manuel.cillero.es/doc/metrica-3/procesos-principales/asi/actividad-10/

Contel Rico, B. (2011). *DESARROLLO DE UNA SOLUCIÓN BUSINESS INTELLIGENCE EN UNA EMPRESA DEL SECTOR DE ALIMENTACIÓN.* Valencia: UPV.

Garcia Mireles, G. A. (2016). *Departamento de matemáticas*. Obtenido de http://www.mat.uson.mx/mireles/conceptosCalidad/modeloCalidad.html

Graham, D., Veenendaal, E. V., Evans, I., & Black, R. (2008). *Foundations of Software Testing: ISTQB Certification.* Londres: Thomson.

IBM. (2019). *IBM Knowledge Center*. Obtenido de https://www.ibm.com/support/knowledgecenter/en/SSRULV_8.6.0/com.ibm.tivoli.itws.doc_8.6/awsrgjsdefn.htm

IEEE. (03 de 2014). *JMPOVEDAR.* Obtenido de https://jmpovedar.files.wordpress.com/2014/03/ieee-829.pdf

Institute, I. S. (2019). *Test Institute*. Obtenido de https://www.test-institute.org/Software_Testing_Levels.php

Kondamuri, M. (16 de Julio de 2018). *Exsilio solutions*. Obtenido de https://blog.exsilio.com/all/how-to-use-change-data-capture/

López Bernal, B. (2002). *Guía para la construcción de un Datawarehouse.* Nuevo León: Tesis.

Loshin, D. (2003). *Business Intelligence The savvy.* USA: Morgan Kaufmann.

Loshin, D. (5 de 05 de 2012). *Data prix knowledge is the goal*. Obtenido de https://www.dataprix.com/data-governance-notas-referencias

Mera Paz, J. (2016). Análisis del proceso de pruebas de calidad de software. *Ingeniería Solidaria*, 163-176.

Mera Paz, J. A. (2016). Análisis del proceso de pruebas de calidad de software. *Ingeniería solidaria*, 163-176.

Ostrand, T. (2002). *Encyclopedia of Software Engineering.* John Wiley & Sons, Inc.

QA, P. (20 de enero de 2018). *ProfessionalQA.com*. Obtenido de http://www.professionalqa.com/ieee-standard-829

Ramos, T. (11 de Diciembre de 2018). *ISO/IEC/IEEE 29119 Software Testing*. Obtenido de https://softwaretestingstandard.org/part3.php

Ranjan, J. (2008). Business justification with business intelligence. *VINE*, 461-475. Obtenido de https://www.emerald.com/insight/content/doi/10.1108/03055720810917714/full/html

S.L., S. e. (2007). *https://www.sinnexus.com/business_intelligence/datawarehouse.aspx*.

Sánchez Peño, J. (2015). *Pruebas de Software. Fundamentos y Técnicas.* Madrid: Universidad Politécnica de Madrid.

Soares, G. (2011). Las Pruebas del Software y su Relación con el Contexto del Producto . *Antioqueña de las Ciencias Computacionales y la Ingeniería de Software*, 15-18.

Team, A. (11 de Enero de 2018). *Alooma*. Obtenido de https://www.alooma.com/answers/what-is-ab-initio-etl

UI Haq , Q. S. (2006). *Data Mapping for Data Warehouse Design.* Morgan Kaufmann.

Valentin Pozo, G. M., & Veliz Ticse , E. I. (2014). *Implementación de estándar de calidad en el proceso de pruebas para aplicativos web del grupo orbis.* Lima: USMP.

Vargas, M. (19 de Octubre de 2017). *Los andes training*. Obtenido de https://losandestraining.com/2017/10/19/casos-de-prueba-yo-escenarios/

Villagómez, C. (8 de 03 de 2017). *CCM*. Obtenido de https://es.ccm.net/contents/223-ciclo-de-vida-del-software

Zeske, M. (01 de 02 de 2018). *Cuida tu dinero*. Obtenido de https://www.cuidatudinero.com/13098460/definicion-de-un-ambiente-de-prueba