Universidad Nacional Arturo Jauretche
Ingeniería de Software I
Ciclo de Vida
Dr. Sergio D. Conde
Ciclo de Vida

Para desarrollar los diferentes tipos de ciclo de vida que tenemos en Ingeniería de Software nos vamos a referir primero a diferentes tipos de Procesos que utilizamos

- Modelo cascada
- Desarrollo incremental
- Ingeniería de software orientada a la reutilización(Reuse-Oriente Software)

Es necesario tener en claro que los procesos de modelos no son ***excluyentes*** entre sí. A menudo en el desarrollo de sistemas grandes se usan de manera conjunta.

Se selecciona un modelo de proceso para la ingeniería del software según la naturaleza del proyecto y de la aplicación.

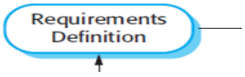
Empezaremos a descubrir el primer ciclo de vida:

1. **Modelo en Cascada:**

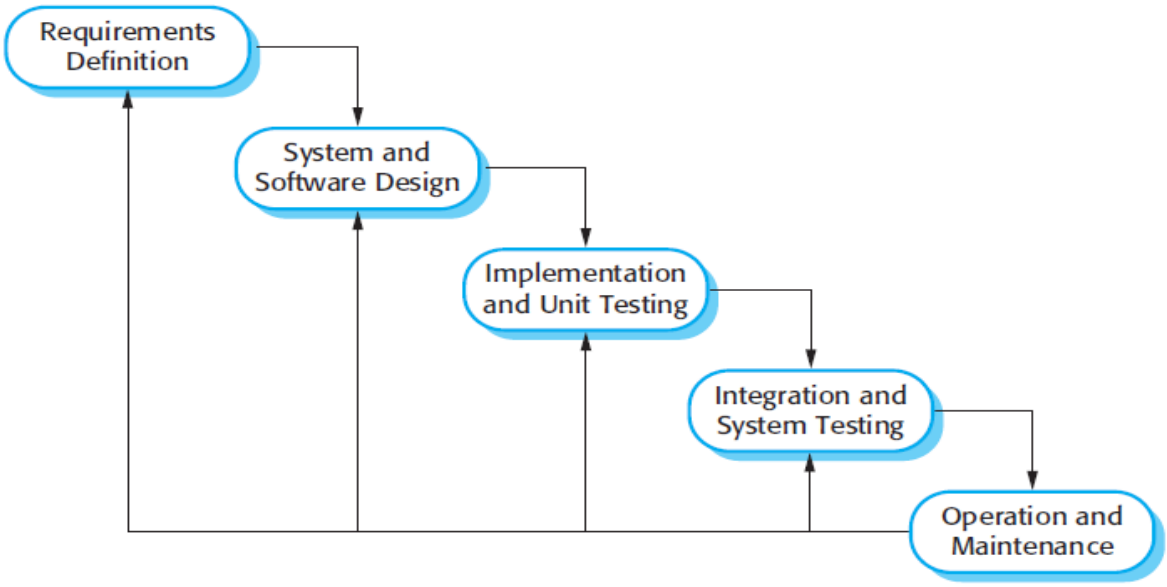
El origen de este modelo de proceso de desarrollo de software que se publicó y derivó de procesos de ingeniería de sistemas más generales.

Las etapas que lo componen son:

Primera Etapa: Análisis y definición de requerimientos:



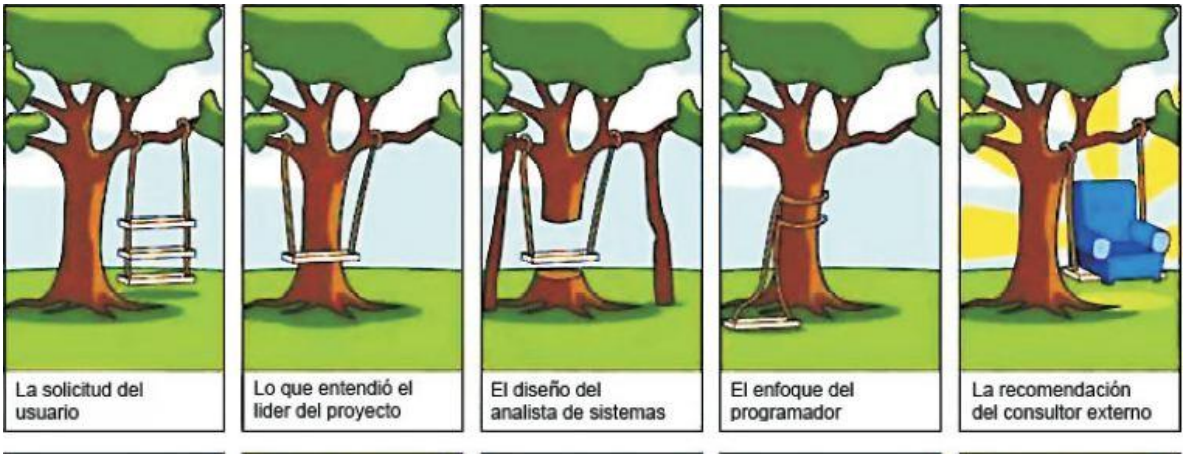
Los servicios restricciones y metas del sistema se definen a partir de las consultas que son realizadas a los usuarios.



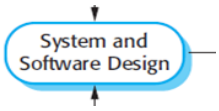
El resultado de estas consultas se traducen y sirven en una especificación del sistema.

Tener en cuenta que vamos a definir Requerimientos Funcionales y No Funcionales.

Veamos una imagen muy conocida:



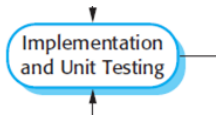
Tener cuidado al interpretar los requerimientos del usuario.

Segunda Etapa: Diseño del sistema y del software: 

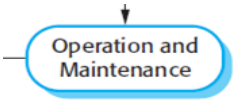
El proceso de diseño del sistema divide los requerimientos del sistema en hardware , software, firmware según corresponda.

Se establece una arquitectura completa del sistema.

En la etapa de diseño del software se identifican y describen las abstracciones del sistema de software y sus relaciones.

Tercera Etapa: Implementación y prueba de unidades. 

Durante esta etapa el diseño del software se lleva a cabo como un conjunto de unidades de programas. La prueba de unidad implica verificar que cada una cumpla con su función.

Cuarta Etapa: Funcionamiento y mantenimiento: 

Por lo general es la etapa más larga del ciclo de vida, El sistema se instala y se pone en funcionamiento práctico.

El mantenimiento no implica corregir e**rrores no descubiertos** en las etapas anteriores del ciclo de vida.

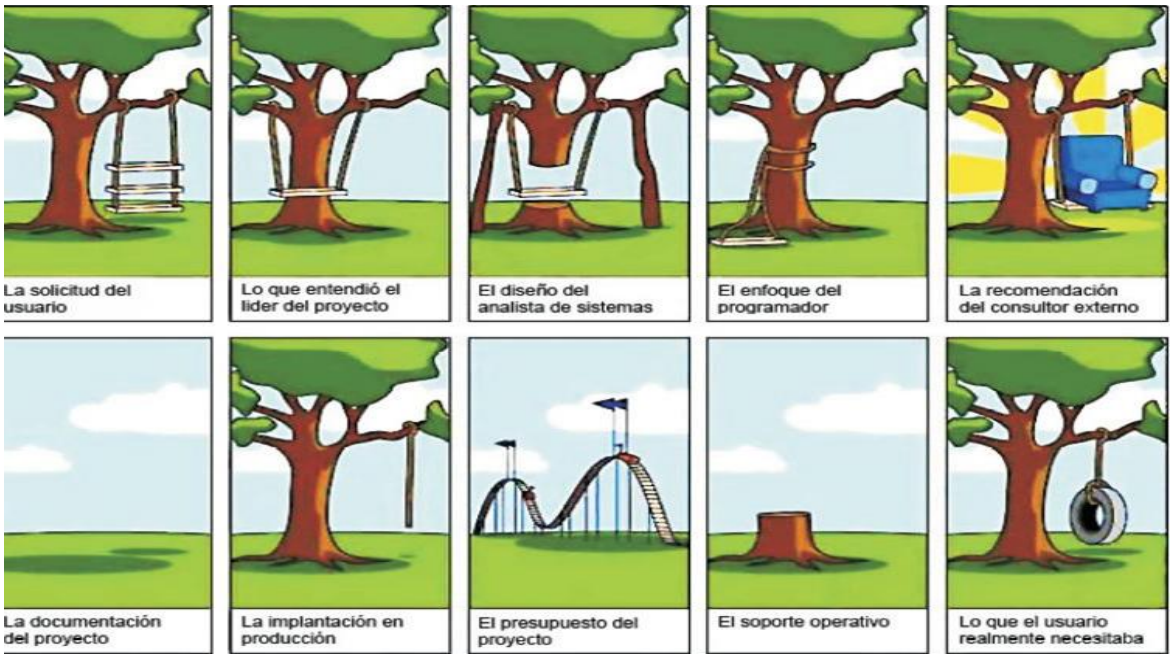
Implica desarrollar un mantenimiento adecuado para el software funcione adecuadamente.

Siempre se establece que el software se entrega con 0(cero) error.

Desventajas del Modelo de Cascada:

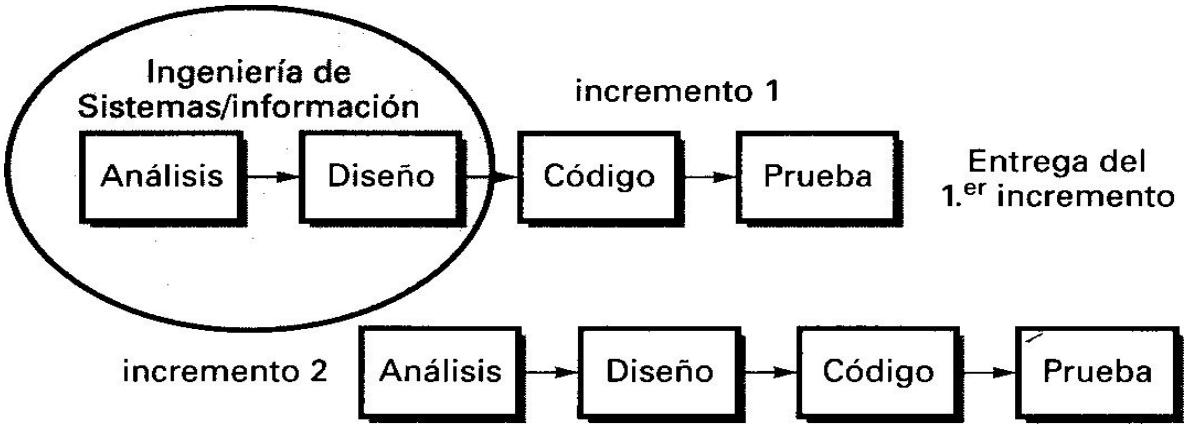
- Es rígido.
- Poco Flexible.
- Muchas Restricciones.
- Es costoso: cualquier error eleva el costo.
- Difícil de detectar errores.
- Se necesita tener en claro los requerimientos al comienzo del proyecto.

Volveremos a recordar y completar la imagen anterior para poder comprender los requerimientos que pide el usuario.



2. **Desarrollo Incremental o Evolutivo**

Se basa en la idea de desarrollar una implementación inicial , luego exponerla a los comentarios del usuario y refinando a medida que se van liberando las diferentes versiones hasta que se llega a un sistema adecuado.



Observando el esquema podemos determiner que en el **Incremento 2** se empieza al Análisis a partir del Diseño de la Entrega del Primer Incremento.

Si hubiese un **Incremento 3** se empieza al Análisis a partir del Diseño de la Entrega del Segundo Incremento y asi sucesivamente.

Ventajas del Modelo Incremental o Evolutivo:

- Se reduce el tiempo de desarrollo inicial ya que se implementa una funcionalidad parcial.
- Si se detecta un error, se desecha la última iteración.
- No es necesario tener en claro los requerimientos al comienzo del proyecto.

Dentro del **Modelo Incremental o Evolutivo tenemos:**

2.1. Desarrollo Exploratorio:

- El objetivo del proceso ***es trabajar con el cliente para explorar sus requerimientos y entregar el sistema final.***
- El desarrollo empieza con las partes del sistema que se comprende mejor.
- El sistema evoluciona agregando nuevos atributos propuestos por el cliente.

2.2. Prototipos Desechables:

El objetivo del proceso de desarrollo ***evolutivo es comprender los requerimientos del cliente y entonces desarrollar una definición mejorada de los requerimientos para el sistema.***

El prototipo se **centra en experimentar con los requerimientos del cliente que no se comprenden del todo.**

3. Modelo en Espiral:

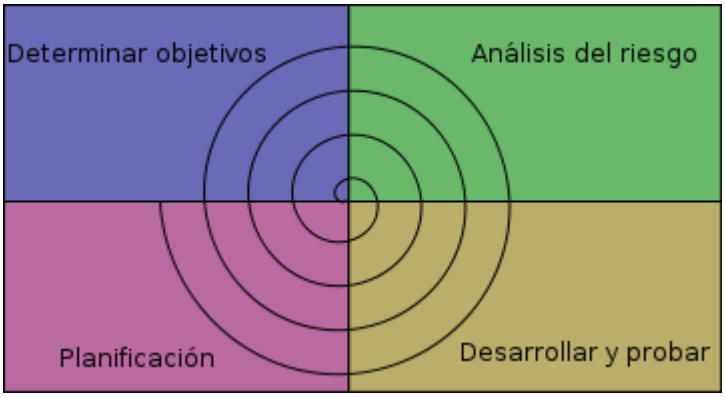
El **Modelo en Espiral**, propuesto originalmente por BOEHM en 1976, es un modelo de proceso de **software evolutivo** donde se conjuga la naturaleza de construcción de prototipos con los aspectos controlados y sistemáticos del ***Modelo Lineal y Secuencial***.

Proporciona el potencial para el desarrollo rápido de versiones incrementales del software que no se basa en fases claramente definidas y separadas para crear un sistema.

En el **modelo espiral,** el software se desarrolla en una serie de versiones incrementales. Durante las primeras iteraciones la versión incremental podría ser un modelo en papel o un prototipo, durante las últimas iteraciones se producen versiones cada vez más completas del sistema diseñado.

El **modelo en espiral** se divide en un número de actividades de marco de trabajo, también llamadas **REGIONES DE TAREAS.**

Cada una de las regiones están compuestas por un conjunto de tareas del trabajo llamado **CONJUNTO DE TAREAS** que se adaptan a las características del proyecto que va a emprenderse en todos los casos se aplican actividades de protección.



Se puede observer que son 4 Areas bien definidas:

Definición de objetivos: se establece la especificación del alcance del producto que se obtendrá al final de la iteración (catálogo de objetivos y requisitos), se identificarán riesgos y se determinarán posibles alternativas de solución.

Análisis del riesgo. se analizan los riesgos involucrados en el desarrollo del proyecto.

Desarrollar y probar: comprende las actividades de análisis, diseño, construcción e implantación de la versión del producto.

Planificación: se procede a realizar un estudio de la situación actual del sistema y del proyecto donde se determina la necesidad o no de una nueva evolución y en el caso de que sea necesaria se realiza la planificación de la misma.

El movimiento de la espiral, ampliando con cada iteración su amplitud radial, indica que cada vez se van construyendo versiones sucesivas del software, cada vez más completas.
Uno de los puntos más interesantes del modelo, es la introducción al proceso de desarrollo a las actividades de análisis de los riesgos asociados al desarrollo y a la evaluación por parte del cliente de los resultados del software.
A diferencia del modelo de proceso clásico que termina cuando se entrega el software, el modelo en espiral puede adaptarse y aplicarse a lo largo de la vida del software de computadora. Una visión alternativa del modelo en espiral puede ser considerada examinando el eje de punto de entrada en el proyecto.

Las regiones de tareas que componen este modelo son 6(seis)

- **Comunicación con el cliente:** las tareas requeridas para establecer comunicación entre el desarrollador y el cliente.

- **Planificación:** las tareas requeridas para definir recursos, el tiempo y otras informaciones relacionadas con el proyecto. Son todos los requerimientos.

- **Análisis de riesgos:** las tareas requeridas para evaluar riesgos técnicos y otras informaciones relacionadas con el proyecto.

- **Ingeniería:** las tareas requeridas para construir una o más representaciones de la aplicación.

- **Construcción y adaptación:** las tareas requeridas para construir, probar, instalar y proporcionar soporte al usuario.

- **Evaluación del cliente:** las tareas requeridas para obtener la reacción del cliente según la evaluación de las representaciones del software creadas durante la etapa de ingeniería e implementación durante la etapa de instalación.

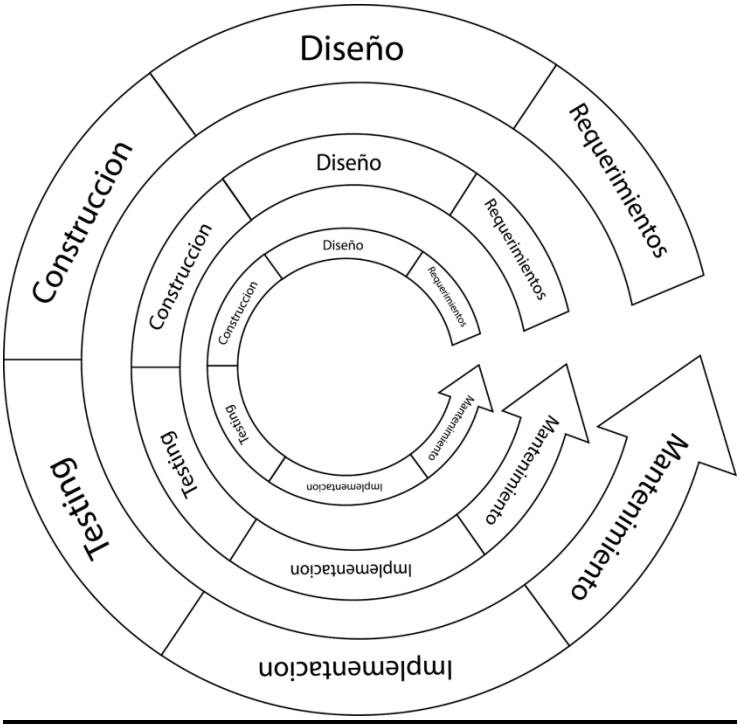
Ventajas del Modelo en Espiral:

- Es el mejor modelo para el desarrollo de grandes proyectos.
- Se puede combinar con otros modelos de desarrollo.
- Puede comenzar con un Proyecto con alto grado de incertidumbre.
- Bajo riesgo de retraso en caso de detección de errores
- El error se soluciona en la rama del espiral correspondiente.

Desventajas del Modelo en Espiral:

- El costo temporal que se suma en cada vuelta.
- Requiere comunicación con el usuario
- Por otro lado, en muchos casos se considera complicado la realización del **análisis de riesgos,** ya que en función del sistema de información resulta muy complejo definir alternativas de solución y determinar con claridad cuál de ellas es la más apropiada.
- Es complicado realizar una planificación global del proyecto, ya que eso dependerá de cómo evolucione el proyecto, producto y usuarios en las distintas iteraciones. Se puede establecer una planificación inicial, con una serie de evoluciones, pero existe una cierta incertidumbre en cuanto hasta dónde se podrá llegar con el presupuesto inicial.

4. Ciclo de Vida SDLC:



SDLC es una metodologia que basa el desarrollo de un software en varias etapas. Estas etapas varias dependiendo el autor o el modelo que se este usando, aunque la mayoria de los autores concuerdan que el modelo cascada o *waterfall*(por su nombre en ingles) tiene las etapas basicas del SDLC, considerando asi al modelo en cascada como el padre de todos los modelos de SDLC. Estas etapas son:

- Analisis de Requerimientos
- Diseño
- Codificacion o Programacion
- Testing
- Implementacion
- Mantenimiento

Etapas

1. Primera Etapa Análisis de Requerimientos

Esta etapa se enfocada completamente en la recopilacion de informacion del sistema a desarrollar. Esta etapa se encuentra conformada por 4 sub etapas:

1.1. Obtencion de requerimientos

En esta fase es cuando la informacion sobre el proyecto se obtiene, es decir el cliente y el desarrollador se reunen y se tocan temas con respecto al proyecto a desarrollar para conseguir toda la informacion necesaria para su desarrollo

1.2. Analisis de la Informacion

Una vez reunida toda la informacion refrente al proyecto, esa informacions e clasifica y se ubica la informacionr ealmente util y la informacion que simplemente no impacta en nada al proyecto

1.3.Especificacion de la informacion

La informacion ya analizada es plasmada en un documento llamado SRS (Software Requirement Specification por sus siglas en ingles) o Especificacion de los Requerimientos de Software. En este documento se plasma todo lo que se va a desarrollar y se mapea a los requerimientos del cliente creando otro documento llamado RTM (Requirements Traceability Matrix) o Matrix de Trazabilidad de Requerimientos.

1.4.Validacion de la Informacion

En esta etapa los documentos creados son mostrados al cliente y el cliente da su aprobacion o desaprobacion a los documentos. Si es necesario los documentos se modifican o se crean de nuevo, para su modificacion o recreacion es necesario pasar nuevamente por las 4 fases.

2. Segunda Etapa Diseño

En esta fase, los documentos creados en la primer etapa: SRS y RTM son traducidos a documentos de diseño, dentro de todos los documentos de diseño que se pueden realizar, es necesario realizar documentos de alto y bajo nivel.

2.1.Diseño de Bajo Nivel

En estos documentos se plasma el algoritmo o la logica a usar en el proyecto, es decir, en estos documentos describen en fondo la funcionalidad del sistema, los documentos mas utilizados son:

- · Pseudocodigo
- · Diagrama de Clases

2.2. Diseño de Alto Nivel

En estos documentos se plasma mediante diagramas el flujo de la informacion, se plasma como la informacion va cambiando de proceso a proceso. Los diagramas utilizados varian de empresa a empresa, pero los mas usados son:

- Diagramas de Contexto
- Diagramas Entidad-Relacion
- Diagramas de Flujo de Datos
- Diagrama de Transicion de Estados

3. Tercera Etapa: Construcción: Codificación o Programación

La codificacion o Programacion es la etapa en la que los documentos de diseño son traducidos a lenguaje de programacion.

Esta fase es quisas la mas simple debido a que solo hay que crear o construir lo que en el diseño se penso aunque no solo se codifica, se tiene que validar y verificar el código.

La validacion es tambien llamada “Unit Testing” o Testing de unidad. El programador verifica que el programa realice exactamente lo que se pide en el SRS ya que este documento son las reglas del negocio; en pocas palabras el programador valida que el programa realice exactamente lo que tiene que hacer. Esta etapa la puede hacer el mismo programador(Desk Check o Prueba de Escritorio) o un colega(Peer Review o Revizion de Compañero).

La verificacion es verificar que el codigo cumpla con las especificaciones o estandares, ya sean propios de la empresa que lo desarrolla, Los estandares del cliente o en todo caso ambos estandares.

Por obvias razones de valida primero el programa, para certificar su funcionamiento, para despues dar paso a aplicar los estandares ya sean de codigo o de Interfaces o cualquier estandar que aplique al proyecto en realizacion.

4. Cuarta Etapa: Testing

Esta etapa, es crucial en cualquier proyecto de desarrollo debido a que la etapa de Testing o Pruebas certifica la calidad del producto y entre mas calidad el producto tenga, mas costoso el producto es.

El QA Team o Equipo de Calidad es el encargado de llevar acabo esta fase ya que un programador no lo puede realizar, no debido a su falta de habilidad o conocimiento, se refiere

mas a un aspecto de objetividad, para ejemplificar esto voy a citar la frase del primer entrenador que tuve:

Para poder llevar acabo el Testing se necesitan 2 cosas:

- · La versión beta del programa
- · Documento donde se certifique que se alcanzo un 100% de cobertura de código

Una vez que los Testers tengan estos documentos en sus manos, el Lider de Proyecto Realiza los siguientes documentos:

- · Strategy Plan
- · Test Plan
- · Traceability Matrix

Y en base a estos 2 doucmentos los Testers realizan los siguientes documentos:

- · Test Data
- · Test Suites
- · Test Cases
- · Test Scripts

Concluida la Fase de Testing, el leader en base a la información generada, genera un Test Execution Report o Reporte de Ejecucion de Pruebas, el cual sera utilizado por el Lider de Proyecto para decidir si el programa esta listo para su implementacion o necesita algunas mejoras o correcciones.

Strategy Plan

Este Documento es usado para determinar que es lo que se va a medir, El ambiente de prueas necesario y los criterios para detener la etapa del Testing y para reanudar esta etapa. Este documente requiere la firma tanto del Lider como del cliente para que ambos esten enterados de la Estrategia a seguir durante esta fase.

Test Plan

En este documento se determinan el acance de lo que se va a probar, que entre y que no en este apartado es importante para asi dedicar el tiempo que se necesita a las partes que realmento lo necesitan. Tambien se definen los tipos de pruebas que se realizaran y que se necesita para realizar cada Prueba; Un cronograma es definido para determinar los tiempos

que se dedicaran a cada prueba. Al igual que el documento Anterior, este tambien necesita la firma de ambos, el lider de proyecto y el cliente para que ambos esten enterados y en comun acuerdo se decida los alcances de esta etapa.

Traceability Matrix

Este documento se crea para tener un “mapping” o mapeo de los Test con los requerimientos, para determinar su prioridad, severidad y los tiempos necesarios para cada test y en caso de que surgan problemas o dudas, verificar con mayor velocidad el requerimiento que es cubierto por el tester con dudas.

Test Data

En este documento se define la informacion que sera usada para esta etapa, es decir, la informacion y/o datos necesarios para ejecutar un Testing

Test Suite

Este documento es opcional, ya que este solo contendra un recopilado de ciertos Testing que cumplan con el escenario de pruebas.

Con escenario de pruebas nos referimos a un escenario en especifico, una accion que se debe de realizar en el sistema, por ejemplo: Podemos crear un Escenario de prueba para la accion de Agregar, otro mas para Buscar y asi sucesivamente.

Aunque en teoria los Test Suites son creados para especificas acciones, en ocasiones se crean para ciertas regiones, es decir si el sistema va a ser usado tanto en Jalisco como en Guanajuato, se puede crear un Test Suite para Jalisco y otro mas para Guanajuato.

Test Case

Este es el documento en el cual se especifica que accion o movimiento se va a probar, en esencia se puede decir que es el documento mas “pequeño” que se crea en esta fase, ya que se describe una especifica situacion a probar.

Test Script

Este documento se puede crear por separado o dentro del Test Case, ya que este documento define la serie de pasos a seguir para poder ejecutar un determinado Testing

Este documento siempre va ligado a uno o varios Testing.

5. Quinta Etapa: Implementacion

En esta etapa el programa ya se encuentra listo, donde el 95% o más de los defectos se han encontrado y se han corregido y el programa se encuentre preparado para su implementación en el ambiente de producción.

Durante esta etapa se generan varios documentos, entre los que destacan:

- Roll-Back plan o Plan de Revertir
- Trainning o Capacitacion
- Release Notes o Notas de Liberacion

Roll-back Plan

Documento en el cual se indica que pasos a seguir en caso de que la instalacion o la implementacion halla fracasado para “desinstalar” y regresar el ambiente de produccion al momento anterior a su instalacion

Training

El training como tal no es un documento, pero se genera documentación necesaria para que el usuario final entienda el funcionamiento del programa.

En algunos casos la capacitación es dada por un tester por cierto tiempo, donde este demuestra como es el manejo de la aplicación y enseño a los usuarios el correcto uso de esta aplicación.

Release Notes

Documento en el cual se indican todas las notas referentes al release actual, pueden ser notas, comentarios o incluso un manual en el que se indique como resolver dudas o problemas

6.Sexta Etapa: Mantenimiento

En esta etapa el programa es tomado y ya dependiendo del caso que sea, se actualiza, adapta o corrige o mantiene, es decir, se toma el programa se vuelve a codificar, se vuelve a testear y se vuelve a dar la implementacion dependiendo de los requerimientos o necesidades actuales del sistema.

Reutilización del software.

Las etapas de requerimientos y la de validación son comparables a las de otros procesos, las etapas del proceso:

- ✓ Análisis de componentes.
- ✓ Modificación de Requerimientos.
- ✓ Diseño de sistemas con reutilización.
- ✓ Desarrollo e integración.

Análisis de componentes:

Teniendo una especificación de requerimientos se buscan los componentes para esta especificación, en la mayoría de los casos no existe la concordancia exacta y los componentes que se utilizan sólo proporcionan parte de la funcionalidad requerida.

Modificación de Requerimientos

En esta etapa los requerimientos se analizan utilizando información acerca de los componentes que han descubierto.

Si las modificaciones no son posibles la actividad de análisis de componente se pueden llevar a cabo nuevamente para buscar soluciones alternativas.

Diseño de sistemas con reutilización

En esta fase se diseña o se reutiliza un marco de trabajo para el sistema.

Los diseñadores tienen en cuenta los componentes que se reutilizan y organiza el marco de trabajo para que los satisfaga si los componentes reutilizables no están disponibles es posible que se tenga que diseñar un nuevo software.

Desarrollo e integración

Para crear el sistema el software que no se puede adquirir se desarrolla y los componentes y los sistemas se integran.

En este modelo, la integración de sistemas es parte del proceso de desarrollo más que una actividad separada**.**