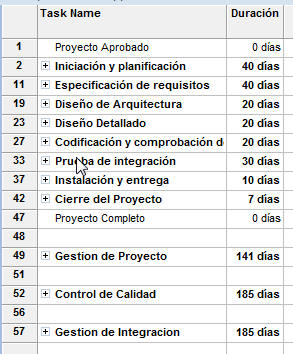
## Principales características de los modelos más utilizados para el desarrollo de SW y su relación con una planificación de proyectos eficiente

## Modelo Cascada

El formato “puro” del modelo apenas se utiliza en la actualidad, pero aún puede utilizarse una variante (descrita a continuación en la plantilla WBS de cascada asociada) de forma eficaz para muchos proyectos.

El modelo de cascada está compuesto por un conjunto de fases ordenadas que es fundamentalmente un desarrollo descendente. Las “salidas” de una fase se convierten en las “entradas” de la siguiente fase. Este modelo se centra en la documentación, cada fase espera la finalización de ciertos documentos antes de que se considere completa. Se espera un “paso de verificación” o revisión al final de cada fase antes de que se considere completa dicha fase, esta revisión asegura que todas las salidas necesarias se hayan realizado y aprobado. Se identifica cualquier cambio necesario para actualizar salidas de la fase anterior y se crean elementos de acción para realizar los cambios necesarios. A continuación en los siguientes gráficos se muestra la relación entre la WBS (trabajo) y el cronograma (actividades, duraciones, responsables, etc….)

## 

## Descripciones de las fases

**Iniciación y planificación**: fase de gestión del proyecto en la que se inicia el proyecto (creado el documento de Resumen con los compromisos del proyecto y definidos los criterios de aceptación) y se planifica (plan del proyecto y documentos asociados creados).

**Especificación de requisitos/requerimientos**: primera fase de desarrollo del producto. Se obtienen, analizan y documentan los requisitos del sistema o producto. Se realiza cualquier prototipo necesario para dar soporte a las especificaciones de requisito. El líder de proyecto incorpora tareas de “Seguimiento y Control”. Esto continúa hasta la fase de cierre.

**Diseño de arquitectura (diseño de alto nivel o preliminar):** define la arquitectura del sistema o producto (subsistema, clase o módulo, base de datos, hardware, etc.) y vuelve a definir los entornos de producción y prueba.

**Diseño detallado**: define el diseño de clase o módulo a gran nivel de detalle.

**Codificación y comprobación de unidad**: realiza la codificación y comprobación de la unidad. La integración de “unidades” completas debe realizarse en esta fase, evitando la realización de otra integración posterior.

**Prueba de integración**: se prueba el sistema o producto integrado. Cualquier defecto que se encuentre se registra, se informa a los responsables de solucionarlo y se soluciona (lleva varios ciclos de prueba).

**Instalación y entrega**: se efectúa la comprobación de aceptación, los representantes del cliente/usuario aceptan formalmente el sistema o producto y se hace la entrega formal.

**Cierre**: El cierre es una fase de gestión del proyecto en la que se celebra una reunión de revisión del proyecto, se crea, revisa y aprueba un documento de revisión y se finaliza y archiva el proyecto. Y se registran las Lecciones aprendidas que contiene, ente otros temas, las características más significativas del ciclo de vida del proyecto.

**Gestión de Proyecto:** Esta actividad se realiza durante la ejecución de todo el proyecto, envuelve las actividades de gestión de riesgos, recursos y actividades propias de la ingeniería de Software como ser análisis, diseño, codificación, pruebas, etc. Su duración es equivalente a todo el ciclo de desarrollo de software.

**Control de Calidad:** Esta actividad se realiza a lo largo de todo el proyecto, se utilizan los informes de avance y distintas minutas de reunión que plasmen aspectos de monitoreo, seguimiento de desvíos, etc., que se presentan durante todas las clases/reuniones con la cátedra.

**Gestión de Integración:** Representa todas las actividades que se llevan adelante para que la documentación presentada sea coherente con entre sí y principalmente esté en línea con el producto que se está construyendo. Permite mantener la trazabilidad ente los distintos procesos de gestión llevados a cabo durante todo el proyecto, manteniéndolos integrados (gestión del riesgo, gestión de los recursos, gestión del tiempo, gestión de las comunicaciones, gestión de la calidad, etc...). La gestión de integración se realiza a lo largo de todo el proyecto.

## 

## Cuándo elegir el ciclo de software de “cascada” para el proyecto?

Al decidir utilizar el ciclo de vida de cascada deben tenerse en cuenta las tres siguientes suposiciones clave inherentes al modelo:

1. El modelo “puro” no permite la superposición de fases (es decir, iniciar la fase siguientes antes de que se hayan completado, revisado y aprobado todas las salidas de la fase anterior). No obstante, en la práctica, se produce alguna superposición, pero debe planificarse y controlarse. No se debiera adelantar mucho porque implicaría re-trabajo y mayor esfuerzo/costo.
2. Se pueden corregir los resultados de fases previas, a medida que se evolucione más en las fases siguientes, pero puede que no existan recursos suficientes para efectuar revisiones. Se espera que en cada fase se produzcan los resultados adecuados antes de iniciar la fase siguiente.
3. No hay entrega del producto hasta la fase de cierre con la entrega final del producto completo. En proyectos de mayor duración, esto puede considerarse mucho tiempo de espera.

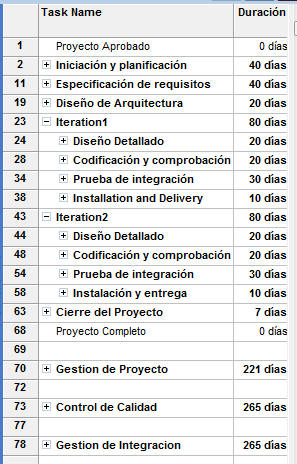
## 

## Modelos “Evolutivos”

Los evolutivos son modelos iterativos, permiten desarrollar versiones cada vez más completas y complejas, hasta llegar al objetivo final deseado, incluso evolucionar agregando nuevas funcionalidades o mejorando las ya existentes durante la fase de operación. Los modelos **Iterativo Incremental** y **Espiral** son dos de los más conocidos y utilizados del tipo evolutivo.

## Modelo “Iterativo Incremental”

Permite especificar y elaborar los distintos incrementos hasta llegar al producto global y final. Las actividades concurrentes (especificación, desarrollo y validación) sintetizan el desarrollo detallado de los incrementos, que se hará posteriormente. En cada ciclo iterativo incremental se permite la entrega de versiones parciales a medida que se va construyendo el producto final, es decir, a medida que cada incremento definido llega a su etapa de operación y mantenimiento. Cada nueva versión emitida incorpora a los incrementos anteriores nuevas funcionalidades y requerimientos que fueron analizados y validados como necesarios, hasta llegar a una integración total para obtener el producto final completo. Al igual que el modelo “Cascada”, supone que la especificación de requisitos/requerimientos de SW y el diseño de arquitectura deben realizarse completamente antes de continuar con las siguientes fases. A diferencia del modelo “Cascada”, este modelo incorpora una serie de iteraciones para desarrollar el sistema o producto en partes. El número de iteraciones puede variar de un proyecto a otro.

A continuación en los siguientes gráficos se muestra la relación entre la WBS (trabajo) y el cronograma (actividades, duraciones, responsables, etc….)

## 

## Modelo “Espiral”

En este modelo, a diferencia del “Iterativo Incremental”, no está definido el alcance funcional del desarrollo, sólo se definen partes de la especificación de requisitos/requerimientos y del diseño de arquitectura en cada iteración. Una de las características principales, por el dinamismo que este modelo conlleva, es que en cada iteración se deben realizar tareas de **Gestión del proyecto, Gestión Riesgos** y **Gestión de los Recursos**.

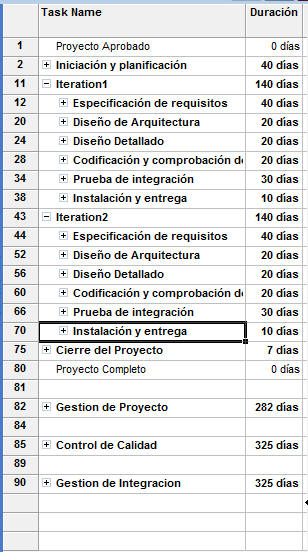
El modelo espiral es un modelo evolutivo que conjuga la naturaleza iterativa del modelo de construcción de prototipos con los aspectos controlados y sistemáticos del Modelo Cascada. En este modelo el software se construye en una serie de versiones incrementales. En las primeras iteraciones la versión incremental podría ser un modelo en papel o bien un prototipo muy básico (desechable o no). En las últimas iteraciones se construyen versiones cada vez más completas del sistema ya diseñado. El modelo se divide en un número de actividades, llamadas regiones de tareas, dentro de un **marco de trabajo**. Las regiones definidas en el modelo son las siguientes:

* Región 1: Tareas requeridas para establecer la comunicación entre el cliente/usuario y el analista o equipo de desarrollo.
* Región 2: Tareas inherentes a la definición de los recursos, tiempo y otra información relacionada con la planificación del proyecto.
* Región 3: Tareas necesarias para evaluar los riesgos técnicos y de gestión del proyecto.
* Región 4: Tareas para construir una o más representaciones del producto software.
* Región 5: Tareas para construir la aplicación, instalarla, probarla y proporcionar soporte al usuario/cliente.
* Región 6: Tareas para obtener la reacción del usuario/cliente, según la evaluación de lo creado e instalado en los ciclos anteriores.

Las actividades enunciadas para el marco de trabajo son generales y se aplican a cualquier proyecto, grande, mediano o pequeño, complejo o no. Las regiones que definen esas actividades comprenden un conjunto de tareas del trabajo, ese conjunto sí se debe adaptar a las características de cada proyecto en particular. Los proyectos pequeños requieren baja cantidad de tareas y también de formalidad, en proyectos mayores y/o críticos cada región de tareas contiene actividades de más alto nivel de formalidad, estas características hacen que el número de iteraciones varíe de un proyecto a otro. Sin embargo en cualquier caso se aplican actividades de soporte, seguridad y calidad (por ejemplo, gestión de configuración del software, garantía de calidad, etc.).

## Cómo se trabaja “en espiral”?

Al inicio del proceso evolutivo, se trabaja alrededor del espiral comenzando por el centro, el primer ciclo de la espiral puede producir el desarrollo de una [especificación](http://es.wikipedia.org/wiki/Especificaci%C3%B3n) del producto; los pasos siguientes podrían generar un [prototipo](http://es.wikipedia.org/wiki/Prototipo) (desechable o no), y progresivamente versiones más sofisticadas del producto software. De esta forma cada paso por la región de planificación provoca ajustes en el plan del proyecto, el costo y planificación se realimentan en función de la evaluación del usuario/cliente. El director del proyecto debe ajustar el número de iteraciones requeridas para completar el desarrollo. El modelo espiral puede ir adaptándose y aplicarse a lo largo de todo el ciclo de vida del software (en el modelo cascada el proceso termina con la entrega del software).. A continuación en los siguientes gráficos se muestra la relación entre la WBS (trabajo) y el cronograma (actividades, duraciones, responsables, etc.



## 

## Ventajas del modelo de entrega por fases (Evolutivo Incremental y Espiral)

Las ventajas de los ciclos de vida de entrega por fases incluyen:

1. Permitir al cliente/usuario recibir y evaluar las partes del sistema o producto antes de que finalice el proyecto. Esto aumenta el nivel de confianza del cliente/usuario y del equipo del proyecto y permite descubrir problemas con anticipación.
2. Permitir a los usuarios aprender el nuevo sistema o producto más lentamente, interactuando adecuadamente y con más atención, puesto que podrán hacerlo poco a poco en lugar de aprender todo el sistema o producto de una vez.
3. En algunos casos (en función de la selección de funcionalidad que se va a implementar en cada iteración), los clientes/usuarios pueden pedir que se detenga el proyecto antes de su finalización y aún tendrán un sistema o producto útil.
4. Si el alcance del proyecto no puede definirse en su totalidad al principio, porque el dominio/negocio es nuevo o el producto es tan grande y/o complejo que podría llevar años su definición, el modelo “Evolutivo” permite al sistema o producto “crecer” a medida que el cliente y el equipo del proyecto descubren lo que necesitan.

## 

## Costos del modelo de entrega por fases (Evolutivo Incremental y Espiral)

Estos modelos tienen costos asociados que deben tenerse en consideración, a saber:

1. Los clientes deben estar preparados para recibir y evaluar el número de entregas planificado. Cada entrega interrumpirá el funcionamiento normal y los clientes/usuarios deben poder indicar los recursos necesarios.
2. Una vez realizada la primera entrega a un cliente/usuario, debe proporcionarse soporte para el software instalado. A veces, esto recae en el personal del proyecto. Las funciones y responsabilidades en esta actividad de soporte a los productos entregados e instalados deben asignarse durante la fase de planificación del proyecto, y asignarse recursos adecuados de forma que el equipo del proyecto continúe con el desarrollo de la siguiente iteración mientras da soporte y/o modifica a las entregas anteriores.
3. Existe el riesgo de que los entregables de la iteración “n” no cubran las necesidades o expectativas de un cambio significativo de los requisitos o que el diseño implementado anteriormente durante las repeticiones “1”….”.n-1” provoque problemas de integración y coexistencia para el cliente/usuario. Deben estimarse reservas de contingencia para imprevistos en caso de que sea necesario mayor esfuerzo, re-trabajo del equipo del proyecto, etc.
4. El modelo “en Espiral” conlleva un alto riesgo, puesto que se “descubrirán” nuevos requisitos/requerimientos en cada iteración y el diseño tiene que adaptarse a los mismos. A veces, será necesario realizar todo el trabajo de nuevo. Los proyectos que utilicen este modelo son más difíciles de estimar y pueden costar más de lo esperado (a veces mucho más). Es por ello que debe hacerse una eficiente gestión de riesgos.