Universidad del Caribe

MODELOS TRADICIONALES

- PESADOS

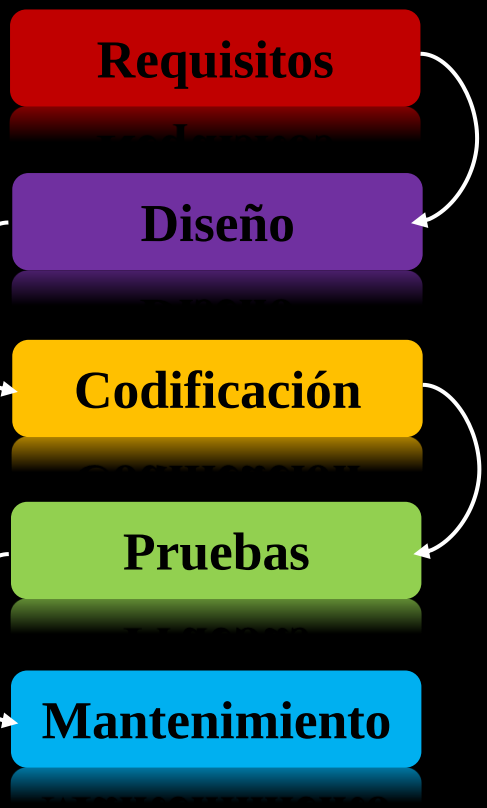
EQUIPO
ARCEO PADILLA SANTIAGO ALEJANDRO
CANUL POOT ENGER ISAAC
CARVAJAL CRUZ AXEL JOAN
CAUICH MATOS FERNANDO ISAI
GUZMAN MARTINEZ JONATHAN ALBERTO
OROZCO CELIS JOSE GUILLERMO

ÍNDICE

MODELO CASCADA **3**
MODIFICACIONES AL MODELO CASCADA 5
CASCADA V 5
FASES SOLAPADAS 6
SUBPROYECTOS 6
REDUCCIÓN DE RIESGO 8
MODELO EVOLUTIVO 9
DESARROLLO EN ESPIRAL 9
DESARROLLO INCREMENTAL 11
DESARROLLO INTERATIVO 11
PROTOTIPOS DESECHABLES 12
MODELO ORIENTADO AL REUSO **12**
COTS - Based Software Development 13
Product – Line Development 13
Architecture – Driven component Development 13
MODELO HÍBRIDO **14**

MODELO CASCADA

El modelo de cascada (también conocido como cascada pura) es el predecesor de todos los modelos de ciclo de vida y ha servido de base para otros. En un modelo en cascada, un proyecto progresa a través de una secuencia ordenada de etapas, partiendo desde el concepto inicial del software hasta la prueba del sistema. El proyecto realiza una revisión al final de cada etapa para determinar si está preparado para pasar a la siguiente etapa.
El modelo en cascada consta de 5 etapas, que son las actividades fundamentales en cualquier desarrollo de software.



1. ANÁLISIS Y ESPECIFICACIÓN DE REQUERIMIENTOS

Se identifican cuáles son los requerimientos del cliente en base a sus necesidades. Se definen los servicios, metas y restricciones del sistema a partir de consultas con los clientes y usuarios. Los requerimientos se documentan en la Especificación de Requerimientos.

2. DISEÑO

Durante el diseño de software se determina la estructura del sistema de software y de sus datos antes de iniciar su codificación. El diseño del sistema se divide en dos etapas:

- Diseño de alto nivel (**Top Level Design: TLD**). - El sistema se divide en partes y se establecen las relaciones entre ellas. Se define qué hará cada una de las partes.
- Diseño detallado (**Detailed Design: DD**). - Se establece cómo debe funcionar cada una de las partes del sistema.

3. CODIFICACIÓN

Durante esta etapa se implementa en código lo que esta descrito en el documento de diseño. Se producen programas ejecutables libres de errores de compilación.

4. PRUEBAS

El sistema de software es un conjunto de módulos. Primero se prueba cada módulo de forma independiente, es decir, se realizan las pruebas unitarias. Después se llevan a cabo las pruebas de integración y de sistema, en las que se integra y prueba todos los módulos como un sistema completo para asegurar que se cumplan los requerimientos del software. Después de pasar las pruebas, el sistema de software se entrega al cliente.

5. MANTENIMIENTO

Una vez que se entrega al cliente, el sistema se instala y se pone en operación. El mantenimiento implica corregir defectos no descubiertos en las etapas anteriores del ciclo de vida y mejorar la implantación de las unidades del sistema. Por lo general (aunque no necesariamente), ésta es la fase más larga del ciclo de vida.

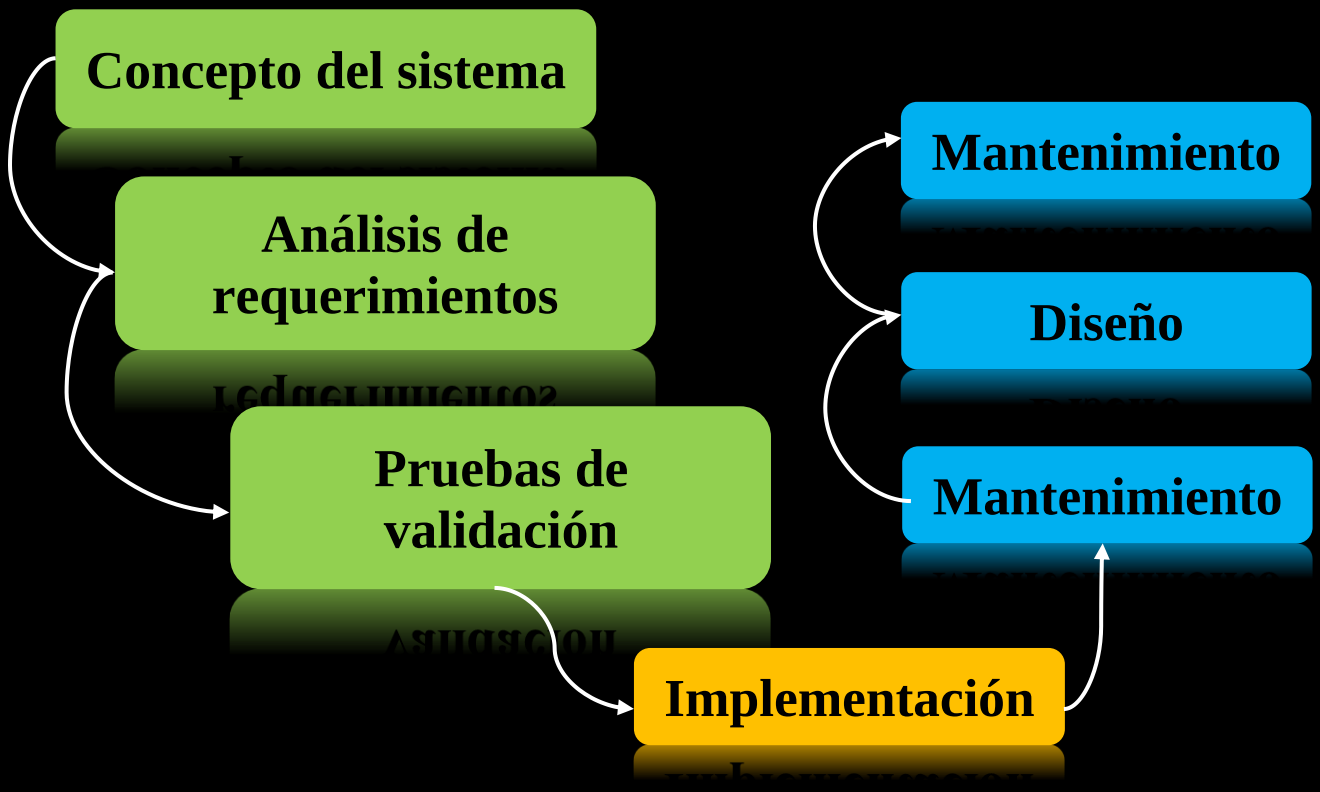
FASE	ARTEFACTO DE ENTRADA	ARTEFACTO DE SALIDA
Requerimientos	Necesidades del cliente	Especificación de requerimientos
Diseño	Especificación de requerimientos	Documento de diseño
Codificación	Documento de diseño	Programa ejecutable
Pruebas	Programas ejecutables	Producto software a entregar

Modificaciones al modelo cascada

Con el fin de solucionar las limitaciones del modelo en cascada surgieron varios modelos que lo modifican de alguna manera. En esta sección presentamos los principales modelos que surgieron a partir de hacer alguna modificación al desarrollo en cascada.

1. CASCADA V

En la cascada en V cada fase del desarrollo tiene una fase correspondiente de verificación y validación.



VENTAJAS

- Produce un sistema robusto y fiable
- Evita el flujo descendente de los defectos
- Funciona bien para pequeños proyectos dende los requisitos son fáciles de entender

DESVENTAJAS

- No funcionan si los requerimientos no son claros desde el principio
- Escribir pruebas por hace rígido el proceso e impide a los probadores seleccionar la forma más eficaz y eficiente de planificar y ejecutar las pruebas

3. FASES SOLAPADAS

El modelo de cascada con fases solapadas puede evitar algunos inconvenientes del modelo de cascada pura, al solapar sus etapas. Es decir, este modelo sugiere un mayor grado de solapamiento. Por ejemplo, sugiere que se debería tener bien hecho el diseño global y quizás a medio hacer el diseño detallado antes de considerar completo el análisis de requerimientos. Otra característica de interés del modelo de Cascada con Fases Solapadas es que se puede reducir sustancialmente la documentación necesaria entre etapas.

VENTAJAS

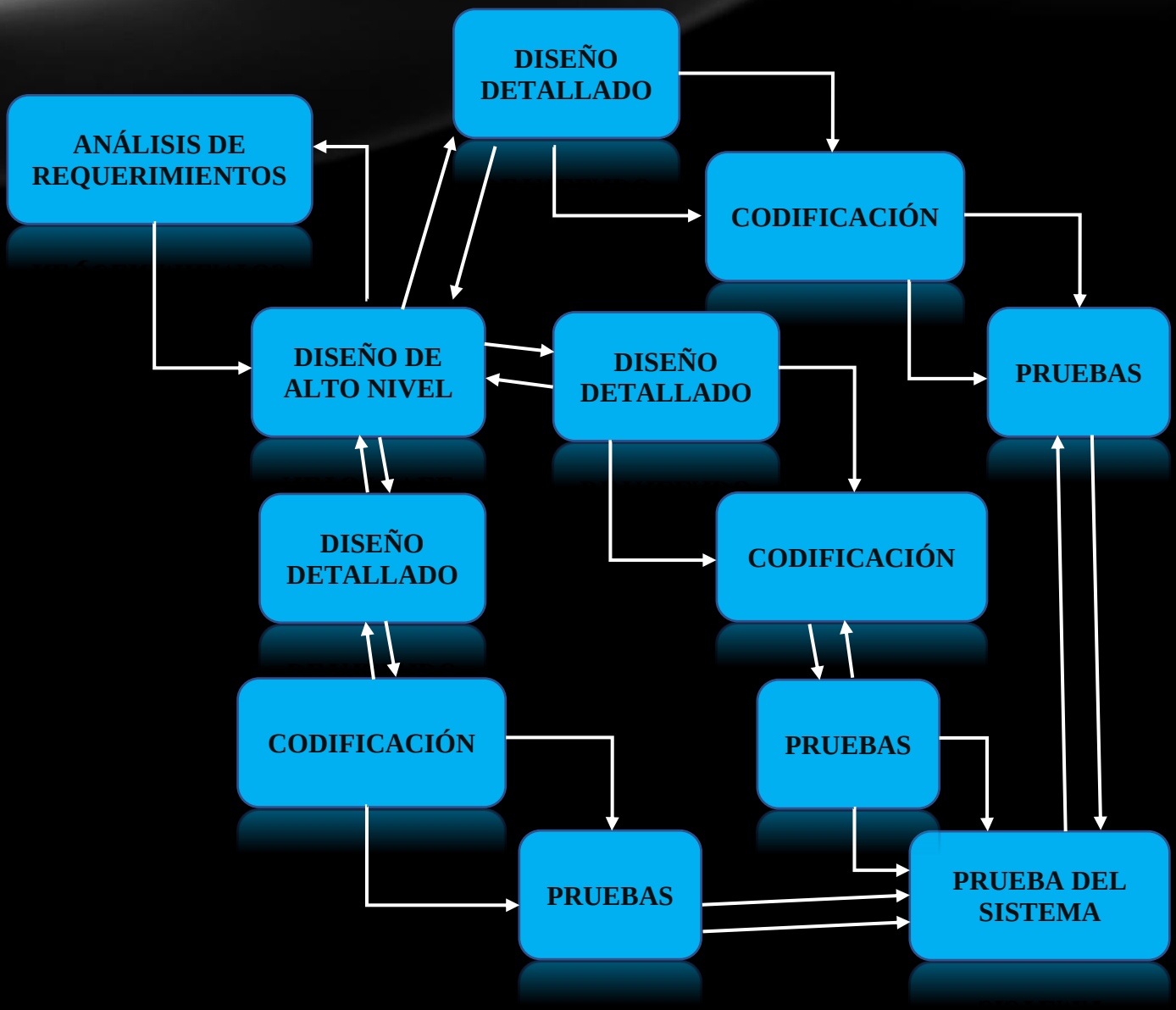
- La siguiente etapa retroalimenta a la anterior y permite mejorarla
- Se pueden descubrir ideas importantes al avanzar en los ciclos de desarrollo
- Funciona en proyectos grandes

DESVENTAJAS

- No funcionan si los requerimientos no son claros desde el principio
- La realización de actividades en paralelo puede provocar una mala comunicación e ineficacia
- Los pasos son más ambiguos, y esto hace más difícil el seguimiento del proceso

2. SUBPROYECTOS

Otro problema presente en el modelo de cascada pura, desde el punto de vista del desarrollo ágil, es que se supone que se ha terminado completamente una etapa antes de iniciar la siguiente (por ejemplo, no es posible iniciar el diseño detallado si no se ha terminado completamente el diseño global). En la cascada con subproyectos se permite la ejecución de algunas de las tareas de la cascada en paralelo (subproyectos), siempre que se haya realizado una cuidadosa planificación.



VENTAJAS

- Cada subsistema lógicamente independiente es un subproyecto que avanza a su propio ritmo
- Se produce una buena documentación completa del sistema

DESVENTAJAS

- Pueden existir interdependencias no previstas entre los subproyectos
- No tiene buena aceptación de cambios en los requerimientos

4. REDUCCIÓN DE RIESGOS

Otra de las limitaciones del modelo de cascada pura es que requiere la definición completa de los requerimientos antes de comenzar la etapa de diseño global. El modelo de cascada con reducción de riesgos incorpora una espiral en lo alto de la cascada para controlar el riesgo de los requerimientos. A este nivel es posible desarrollar un prototipo de interfaz de usuario, utilizar apuntes, tener entrevistas con los usuarios, observar cómo los usuarios interactúan con algún sistema más antiguo, o utilizar otros métodos que se consideren apropiados para la identificación de los requerimientos.

VENTAJAS

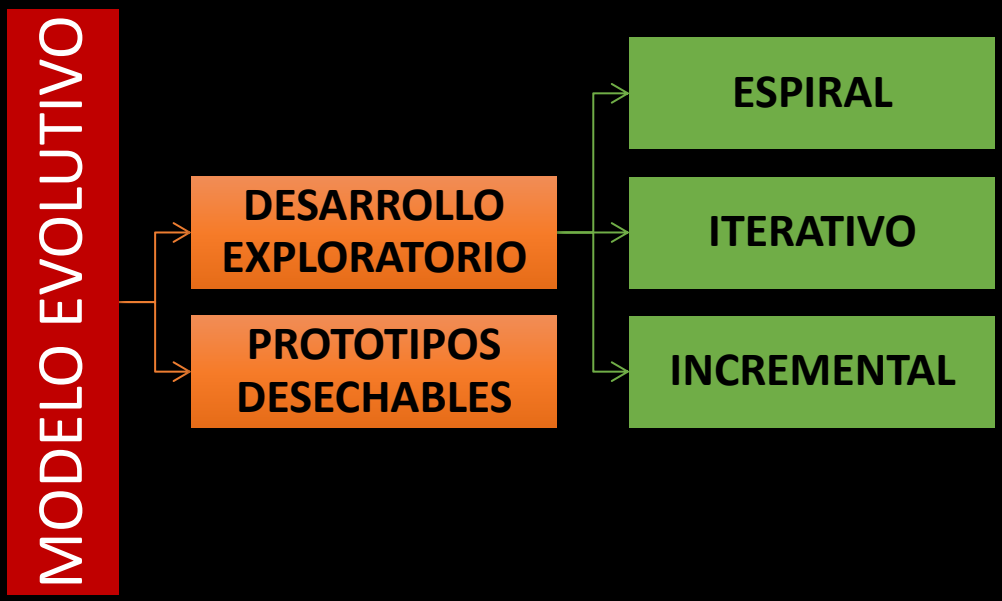
- Se reduce el riesgo de producir un sistema que no cumple con los requerimientos
- Se produce una buena documentación completa del sistema

DESVENTAJAS

- Las modificaciones en los requerimientos afectan bastante una vez que ya se esta trabajando en etapas avanzadas

MODELO EVOLUTIVO

El desarrollo evolutivo consiste en construir el sistema poco a poco en lugar de hacerlo todo "de un jalón", como en el modelo en cascada. El sistema evoluciona en cada etapa porque se le van agregando nuevas características. Los modelos evolutivos tienen como objetivo principal reducir el riesgo y la incertidumbre en el desarrollo. Se utilizan cuando se pueden entregar versiones preliminares con los requerimientos más urgentes o con las partes del sistema que se comprenden mejor. También se utilizan cuando se requiere explorar los requerimientos, porque no se tienen bien definidos. El desarrollo empieza con las partes del sistema que se comprenden mejor y se le van agregando al sistema nuevas características.

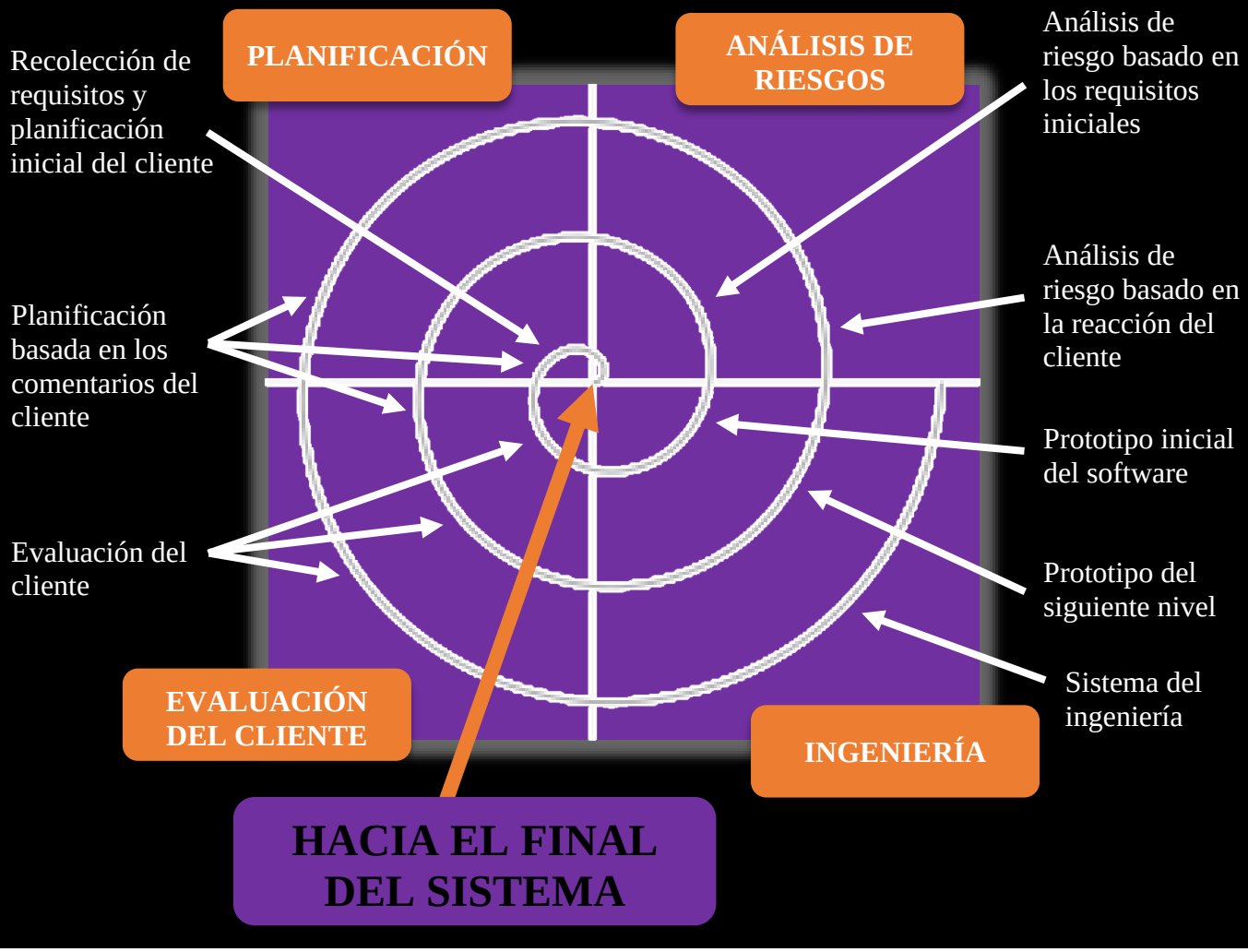


1. DESARROLLO EN ESPIRAL

Es un modelo de ciclo de vida orientado a los riesgos, dividiendo un proyecto software en mini proyectos. Cada mini proyecto se centra en uno o más riesgos importantes hasta que todos éstos estén controlados. El concepto “riesgo” puede referirse a requerimientos poco comprensibles, arquitecturas poco comprensibles, problemas de ejecución importantes, problemas con la tecnología subyacente, y demás.

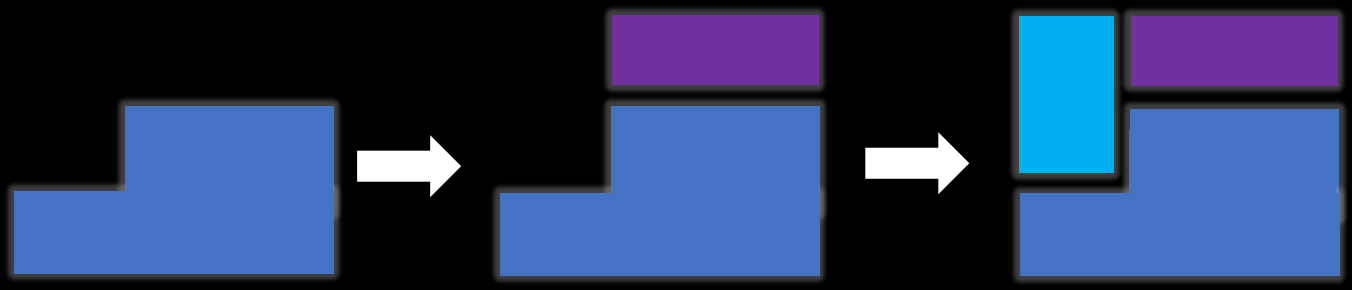
El modelo en espiral combina una parte iterativa, que es la construcción de prototipos, con la parte secuencial del modelo en cascada para ir obteniendo resultados parciales en los que se tiene cierto control. Durante las primeras iteraciones, la versión incremental puede ser un modelo en papel o un prototipo. Durante las últimas iteraciones, se producen versiones cada vez más completas del sistema.

En este modelo hay ciclos de trabajo, cada uno de los cuales estudia el riesgo antes de proceder al siguiente ciclo. Cada ciclo comienza con la identificación de los objetivos, soluciones alternativas, restricciones asociadas con cada alternativa y se evalúan para encontrar la mejor.



3. DESARROLLO INCREMENTAL

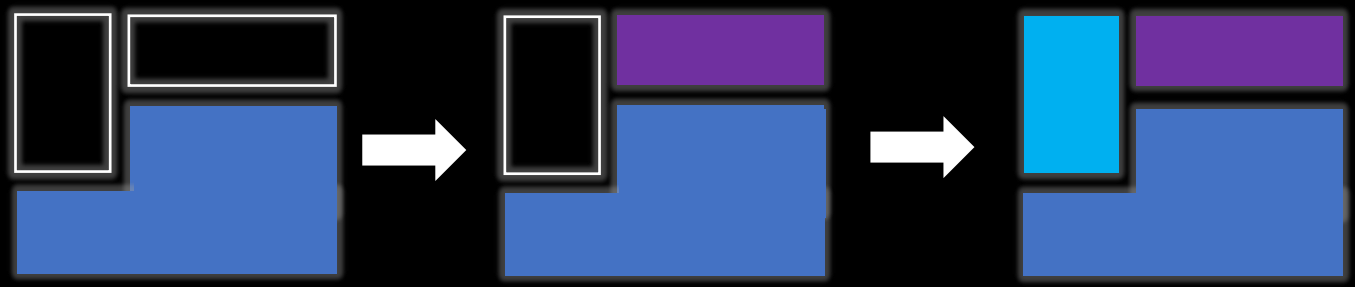
En este modelo, el software se entrega en partes pequeñas, pero utilizables, llamadas incrementos. Cada incremento se construye sobre un sistema parcial que ya ha sido entregado



Por ejemplo, supongamos que se construye un sistema de nómina, en el que primero se entrega la parte que captura los datos de los empleados. Luego se agrega la parte que calcula la quincena en base al salario y los impuestos. Posteriormente se le agrega la función de calcular el pago del aguinaldo. Y finalmente el sistema queda completo con la función de calcular el pago de la prima vacacional.

2. DESARROLLO ITERATIVO

En el desarrollo iterativo se entrega el esqueleto de un sistema completo desde el principio, y se va agregando funcionalidad con cada nueva versión.



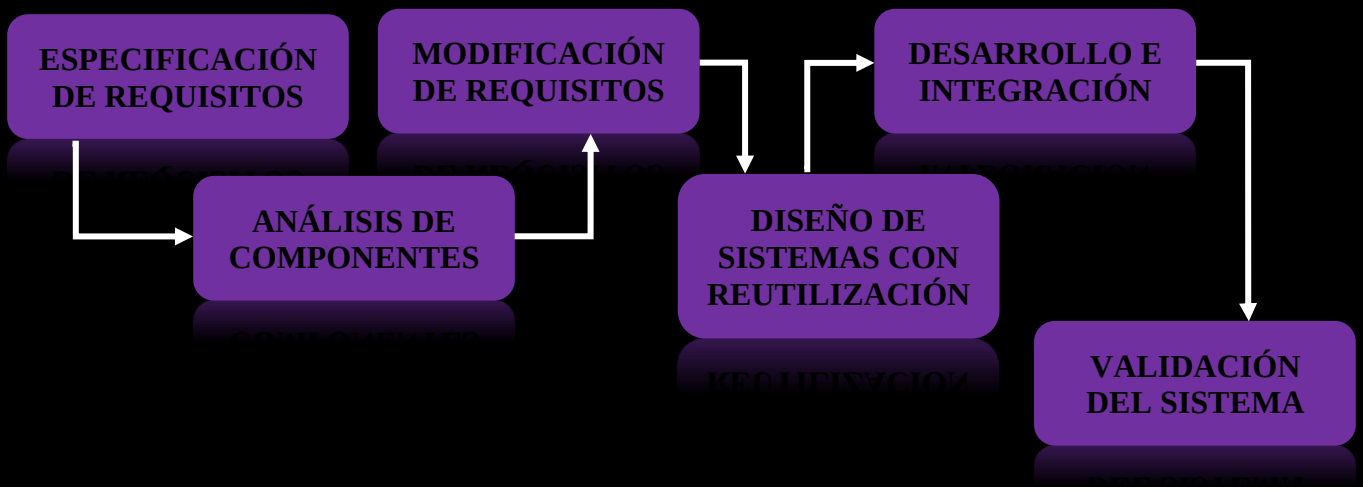
Por ejemplo, se entrega un sistema operativo funcionando completamente (Sistema Operativo para PC). Después se mejora el sistema operativo para que funcione con ventanas (Windows), posteriormente se agrega la capacidad de cambiar el tamaño de las ventanas (Windows 96). Con cada nueva entrega el sistema operativo tiene más facilidades (Windows 98, XP, Vista, 7, 8, 9, 10). Así la empresa gana dinero con la entrega de una nueva versión.

4. PROTOTIPOS DESECHABLES

El objetivo de los prototipos 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. El software se desarrolla para aprender más sobre un problema o explorar la factibilidad o la conveniencia de las posibles soluciones.
Un prototipo desechable no está pensado para ser utilizado como componente real del software que se entrega al cliente. Un prototipo no es un producto de calidad que deba mantenerse a largo plazo. Por el contrario, los prototipos son creados y probados rápidamente, para luego ser desechados. Sin embargo, es común que, por presiones de tiempo, se trate de enviar un prototipo al mercado como si éste fuera el producto final. En general, siempre existirá un conflicto entre un desarrollo rápido y un producto de calidad.

MODELO ORIENTADOS AL REUSO

La reusabilidad (reuso) tiene como objetivo utilizar algunos elementos ya entregados del producto de software, tales como elementos de la especificación de requerimientos, del diseño, del código o de cualquier producto comercialmente disponible. Va desde la reutilización de clases y métodos en las bibliotecas hasta la reutilización de sistemas de aplicación completos. La reusabilidad en el desarrollo de sistemas se basa en la existencia de componentes reutilizables, y tiene como finalidad usar de nuevo ideas, arquitecturas, diseños o código de una aplicación para construir otras. De aquí que dos aspectos clave a considerar en la calidad de un sistema de software a desarrollar, sean su reusabilidad y su extensibilidad.
El proceso de desarrollo de un sistema con reuso se enfoca en integrar los componentes reutilizables en el sistema en lugar de desarrollarlos desde cero. A la versión moderna de reuso se le llama Desarrollo basado en componentes.



3. COTS – Based Software Development

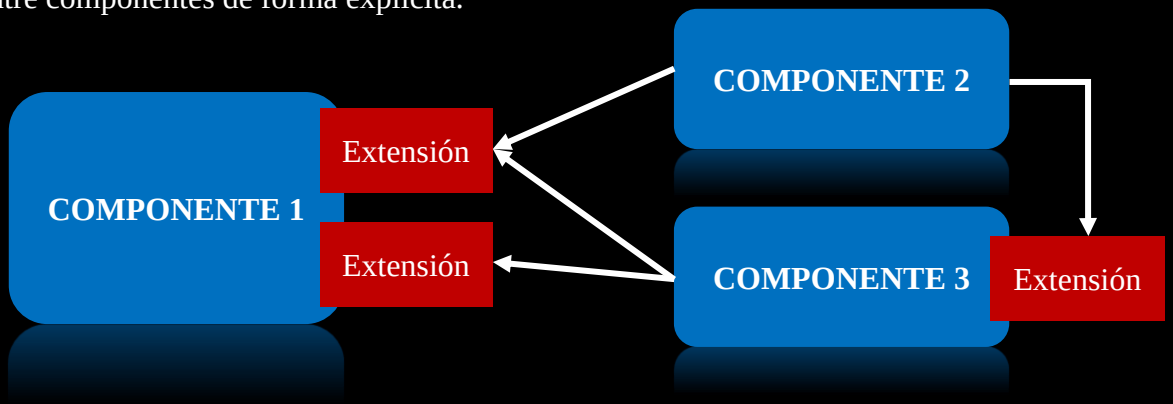
COTS: Commercial On The Shell .- Un producto COTS es un sistema de software de uso general, que incluye muchas características y funcionalidades que pueden adaptarse a las necesidades de diferentes clientes sin tener que cambiar el código fuente. Ejemplos: Paquetes ERP (Enterprice Resource Planning) para la gestión de nómina, facturas, ventas, etc., aplicaciones ofimáticas (procesador de texto, hoja de cálculo, correo electrónico...), paquetes de diseño asistido por computadora.

2. Product – Line Development

Se construyen familias de productos (llamadas líneas de productos) que comparten la misma arquitectura y se construyen a partir de componentes comunes. Cada línea de producto pertenece a un dominio de aplicación específico o a una estrategia de mercado. El objetivo es vender casi el mismo sistema a muchos clientes con costos mínimos. Ejemplos: una aplicación para diferentes Smartphone, motor de un videojuego para diferentes plataformas.

1. Architecture – Driven Component Development

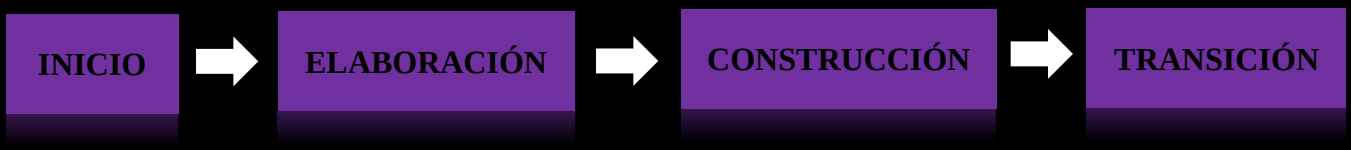
El objetivo de la reutilización de software de arriba hacia abajo es lograr la reutilización planificada de los componentes por medio de un modelo de componentes trazable. La trazabilidad es un requisito clave para el modelo de componentes. El proceso de software tradicional no es adecuado para el desarrollo de componentes de software reutilizables. Este documento propone el enfoque de desarrollo de componentes impulsado por la arquitectura que da como resultado un modelo de componentes trazable. Los patrones de diseño de la arquitectura son los elementos clave de la reutilización del software. Los patrones se utilizan como medio para expresar los resultados de las diferentes fases de desarrollo y se utilizan para describir la pericia y la experiencia del desarrollo. Los patrones admiten la reutilización de software de arriba hacia abajo mediante la identificación y el modelado de relaciones entre componentes de forma explícita.



MODELO HÍBRIDO

El Proceso Unificado (RUP: Rational Unified Process) se concibió para el diseño orientado a objetos. Sin embargo, puede usarse para el paradigma estructurado. El RUP es un modelo híbrido porque reúne elementos de todos los modelos de procesos genéricos (cascada, evolutivo y reuso). Sus características predominantes son el desarrollo iterativo, el desarrollo dirigido por riesgos y el desarrollo incremental del sistema. El Proceso Unificado visualiza al sistema desde tres perspectivas:

1. Perspectiva dinámica: son las fases del modelo sobre el tiempo.
2. Perspectiva estática: son las actividades del proceso y se les llama flujos de trabajo.
3. Perspectiva práctica: sugiere buenas prácticas a utilizar durante el desarrollo del sistema. **Perspectiva dinámica. -** Se dice que las fases son sobre el tiempo porque se aplican de forma secuencial, éstas son: Inicio, Elaboración, Construcción y Transición son iterativas. Cada una de estas fases es iterativa, es decir, pueden constar de una o más iteraciones.



Las características de la fase de inicio son las siguientes:

- Se establece la razón de ser del proyecto y se determina su alcance.
- Se adquiere una visión aproximada del proyecto en términos de los principales requerimientos.
- Se desarrolla un primer análisis del negocio.
- Se efectúan las primeras estimaciones, aún no precisas, del proyecto.
- Ésta no es precisamente una fase de recolección de requerimientos, sino una especie de fase de viabilidad.

A continuación, se mencionan las características de la fase de elaboración:

- Recolección de requerimientos más detallados.
- Se realiza análisis y diseño de alto nivel con el objetivo de definir la arquitectura base del sistema.
- Se crea el plan de construcción.
- Se refina la visión que se tiene del proyecto.
- Se identifican más requerimientos.
- Se determina de forma más refinada el alcance del proyecto.
- Análisis y mitigación de los principales riesgos.

Las características de la fase de construcción son las siguientes:

- Consta de varias iteraciones.
- En cada iteración se construye software que satisface un subconjunto de los requerimientos del proyecto.
- Los requerimientos a implementar son aquellos de menor riesgo y de fácil implementación.
- El software construido en cada iteración es software de calidad probado e integrado.

A continuación, se mencionan las características de la fase de transición:

- Pruebas beta.
- Refinamiento del desempeño del sistema.
- Entrenamiento del usuario.
- Lanzamiento del sistema para su puesta en producción.

Perspectiva estática. - Los flujos de trabajo son las actividades que se llevan a cabo durante el proceso.

Éstos son similares a las etapas del modelo en cascada, pero además se toman en cuenta otros aspectos. Las fases son dinámicas y tienen objetivos. Los flujos de trabajo son estáticos y son actividades técnicas que no están asociadas con fases únicas, sino que pueden utilizarse durante el desarrollo para alcanzar los objetivos de cada fase. Como se puede observar, la intensidad de los flujos de trabajo depende de la fase del proceso, por ejemplo, la actividad de análisis de requerimientos se lleva a cabo principalmente al inicio y en la elaboración, mientras que la gestión de configuración y del proyecto se lleva a cabo a lo largo de todo el proyecto.

FLUJO DE TRABAJO	DESCRIPCIÓN
MODELADO DEL NEGOCIO	Los procesos del negocio se modelan utilizando casos de uso de negocio.
REQUERIMIENTOS	Se definen los actores que interactúan con el sistema y se desarrollan casos de uso para modelar los requerimientos del sistema.
ANÁLISIS Y DISEÑO	Se crea y documenta y un modelo de diseño utilizando modelos arquitectónicos, modelos de componentes, modelos de objetos y modelos de secuencias.
IMPLEMENTACIÓN	Los componentes del sistema se estructuran y se implantan en subsistemas. La generación automática de código de los modelos del diseño ayuda a acelerar este proceso.
PRUEBAS	Son un proceso iterativo que se lleva a cabo conjuntamente con la implementación, se deben hacer las pruebas del sistema.
PUESTA EN MARCHA (DESPLIEGUE)	Se crea un "release" del producto, se distribuye a los usuarios y se instala en su lugar de trabajo.
GESTIÓN DE CONFIGURACIÓN Y CAMBIOS	Este flujo de trabajo de soporte gestiona los cambios del sistema.
GESTIÓN DEL PROYECTO	Este flujo de trabajo de soporte gestiona el desarrollo del sistema.
ENTORNO	Este flujo de trabajo se refiere a hacer herramientas de software apropiadas disponibles para los equipos de desarrollo de software.

En el Proceso Unificado se incluye la utilización del software en un entorno de usuario como parte del proceso, y se le nombra la puesta en marcha. Cada una de las fases se divide a su vez en varias iteraciones (la de inicio solo consta de varias iteraciones en proyectos grandes). Estas iteraciones ofrecen como resultado un incremento del producto desarrollado que añade o mejora las funcionalidades del sistema en desarrollo