

2

## MPDSW

- ▶ El proceso software
- ▶ Estándares en Ingeniería del Software
  - ▶ Utilidad de los estándares
  - ▶ Tipos de estándares
- ▶ Estándares relacionados con el proceso software
  - ▶ SEI's CMM ([Software Engineering Institute](#) -*Capability Maturity Model*)
  - ▶ De procesos estándar del ciclo de vida
    - ▶ ISO 9000
    - ▶ Estándar IEEE 1074-1998
    - ▶ Estándar IEEE/EIA (ISO/IEC) 12207

### QUE ES UN PROCESO DE SOFTWARE?

Es un conjunto de actividades y resultados asociados que producen un producto de software.

SOFTWARE: Conjunto de programas que están relacionados con un sistema informática de forma automatizada. (PC, celulares, etc).

EXISTEN 4 ACTIVIDADES FUNDAMENTALES DE PROCESO COMUNES PARA TODOS LOS PROCESOS DEL SOFTWARE:

1. ESPECIFICACION DE REQUISITOS DEL SOFTWARE
2. DESARROLLO DEL SOFTWARE: ANÁLISIS – DISEÑO: INTERFACES, DISEÑO DE BASE DE DATOS, DISEÑO DE ARQUITECTURA DEL SOFTWARE – CONSTRUCCIÓN
3. VALIDACIÓN DEL SOFTWARE
4. EVALUACIÓN DEL SOFTWARE: PORQUE LOS PROCESOS SON DINÁMICOS. SE DEBE IR HACIENDO UNA MANTENIBILIDAD.

5

### 1.- ¿Qué es un proceso software?

- ▶ Distintos procesos de software organizan las actividades de diferentes formas, y las describen con diferente nivel de detalle.
  - El tiempo de cada actividad varía, así como los resultados.
  - Organizaciones diferentes usan procesos diferentes para producir el mismo producto.

CLASE 2

CICLO DE VIDA DE UN PRODUCTO SOFTWARE.

SUCESION DE ETAPAS POR LAS QUE ATRAVIESA UN PRODUCTO DE SOFTWARE A LO LARGO DE SU EXISTENCIA (DESARROLLO Y EXPLOTACION)

The diagram features a large orange arrow pointing right, labeled '1.- ¿Qué es un proceso software?'. Below it, the text 'Ciclo de vida ≠ Ciclo de desarrollo' is displayed. To the right of this text is a green arrow pointing right, with the text 'Desde el análisis hasta la entrega al usuario'. Below the green arrow is a larger orange arrow pointing right, with the text 'Toda la vida del sistema: desde la concepción hasta el fin de uso.'

**ESTANDARES EN INGENIERIA DE SOFTWARE:**

**ESTÁNDAR:** CONJUNTO DE CRITERIOS APROBADOS, DOCUMENTADOS Y DISPONIBLES PARA DETERMINAR LA ADECUACIÓN DE UNA ACCIÓN (ESTÁNDAR DE PROCESO) O DE UN OBJETO (ESTÁNDAR DE PRODUCTO).

**GUÍA:** DOCUMENTO DE CRITERIOS BIEN DEFINIDOS Y DOCUMENTADOS QUE ENCAMINAN UNA ACTIVIDAD O TAREA.  
- ES MÁS FLEXIBLE QUE UN ESTÁNDAR.

**¿POR QUÉ USAR ESTANDARES EN LA INGENIERIA DE SOFTWARE?**

SEGÚN SOMERVILLE LOS ESTANDARES SON ÚTILES PORQUE:

- AGRUPAN LO MEJOR Y MAS APROPIADO DE LAS BUENAS PRACTICAS Y USOS DEL DESARROLLO DEL SOFTWARE
- ENGLOBAN LOS CONOCIMIENTOS QUE SON PATRIMONIO DE UNA ORGANIZACIÓN
- PROPORCIONA UN MARCO PARA IMPLEMENTAR PROCEDIMIENTOS DE ASEGURAMIENTO DE CALIDAD.
- PROPORCIONAN CONTINUIDAD ENTRE EL TRABAJO DE DISTINTAS PERSONAS

**TIPOS DE ESTANDARES EN INGENIERIA DE SOFTWARE**

**ESTANDARES PARA DATOS:**

DEBE ASIGNAR NOMBRES A LOS DATOS Y ESPECIFICAR LONGITUD Y TIPO HASTA LOS RELACIONADOS CON BBDD

EJ: SQL2, SQL1999, ODMG 2.0

**ESTANDARES DE CODIFICACION:**

## ABREVIATURAS Y DESIGNACIONES FORMALES PARA DESCRIBIR ACTIVIDADES DENTRO DE LA ORGANIZACIÓN

### ESTÁNDARES ESTRUCTURALES:

POLÍTICAS DE DIVISIÓN DEL SOFTWARE EN MODULOS.

### ESTÁNDARES DE DOCUMENTACIÓN:

### ESTÁNDARES DE PROCESO SOFTWARE:

### EJEMPLOS DE ESTÁNDARES DE INGENIERÍA DE SOFTWARE:

IEEE STANDARDS COLLECTION SOFTWARE ENGINEERING – 1998 EDITION

IEEE Std 610.1990 GLOSSARY SOFTWARE ENGINEERING

IEEE Std 829-1983, STANDARD FOR SOFTWARE TEST DOCUMENTATION

IEEE Std 830-1993 RECOMMENDED PRACTICE FOR SOFTWARE REQUIREMENTS SPECIFICATIONS

11

2.- Estándares en ingeniería del software.  
Ejemplos de estándares en ingeniería del software

- IEEE Standards Collection Software Engineering – 1998 Edition
- IEEE Std. 610.12-1990, Glossary of Software Engineering Terminology
- IEEE Std. 829-1983, Standard for Software Test Documentation
- **IEEE Std. 830-1993, Recommended Practice for Software Requirements Specifications.**
- IEEE Std. 1045-1992, Standard for Software Productivity Metrics
- IEEE Std. 1062-1987, Recommended Practice for Software Acquisition
- IEEE Std. 1063- 1987, Standard for Software User Documentation
- IEEE Std. 1219-1992, Standard for Software Maintenance
- **IEEE Std. 1074 -1997, Standard for Developing Software Life Cycle Processes**
- **IEEE Std. 12207 -1996, Industry Implementation of International Standard**

## Estándares IEEE (cont.)

- 1175.1-2002, Guide for CASE Tool Interconnections - Classification and Description
- 1219-1998, Standard for Software Maintenance
- 1220-1998, Standard for the Application and Management of the Systems Engineering
- 1228-1994, Standard for Software Safety Plans
- **1233-1998, Guide for Developing System Requirements Specifications**
- 1320.1-1998, Standard for Functional Modeling Language-Syntax and Semantics for IDEF0
- 1320.2-1998, Standard for Conceptual Modeling Language Syntax and Semantics...
- 1362-1998, Guide for Information Technology-System Definition-Concept of Operations
- 1420.1-1995, Standard for Information Technology-Software Reuse-Data Model for Reuse
- 1420.1a-1996, Supplement to Standard for Information Technology-Software Reuse-Data
- 1420.1b-1999, IEEE Trial-Use Supplement to Standard for Information
- 1462-1998, Standard - Adoption of International Standard ISO/IEC 14102: 1995; 1465-1998, Standard - Adoption of International Standard ISO/IEC 12119: 1994(E)
- 1471-2000, Recommended Practice for Architectural Description of Software Intensive
- **1490-1998, Guide - Adoption of PMI Standard - A Guide to the Project Management Body of Knowledge**
- 1517-1999, IEEE Standard for Information Technology-Software Life Cycle Processes-Reuse
- 1540-2001, Standard for Software Life Cycle Processes- Risk Management
- **2001-2002, Recommended Practice for Internet Practices - Web Page Engineering**
- 14143.1-2000, Adoption of ISO/IEC 14143-1:1998 Information Technology-Software

### PROCESO SOFTWARE (PS) EFECTIVO:

HABILITA LA ORGANIZACION A INCREMENTAR SU PRODUCTIVIDAD AL DESARROLLAR SOFTWARE:

- PERMITE ESTANDARIZAR ESFUERZOS, PROMOVER REUSO, REPETICION Y CONSISTENCIA ENRE PROYECTOS.
- PROVEE LA OPORTUNIDAD DE INTRODUCIR MEJORES PRACTICAS DE LA INDUSTRIA.
- PERMITE ENTENDER QUE LAS HERRAMIENTAS DEBEN SER UTILIZADAS PARA SOPORTAR UN PROCESO
- ESTABLECE LA BASE PARA UNA MAYOR CONSISTENCIA Y MEJORAS FUTURAS

### PROCESO DE SOFTWARE MEJORA:

- DEFINE COMO MANEJAR LOS CAMBIOS Y LIBERACIONES A SISTEMAS DE SOFTWARE EXISTENTES.
- DEFINE COMO LOGRAR LA TRANSICIÓN DEL SOFTWARE A LA OPERACIÓN, Y COMO EJECUTAR LOS ESFUERZOS DE OPERACIÓN Y SOPORTE.

NECESITAMOS UN PROCESO DE SOFTWARE CUYA FUNCIONALIDAD ESTE PROBADA EN LA PRACTICA, Y PERSONALIZADA PARA QUE CUMPLA CON NUESTRAS NECESIDADES ESPECIFICAS.

### ELEMENTOS TIPICOS PRODUCTO SOFTWARE:

ELEMENTOS	DESCRIPCION
ACTIVIDAD	DEFINE LAS ACTIVIDADES QUE SE LLEVAN A CABO EN UN MOMENTO DADO DEL DESARROLLO DE SOFTWARE
FLUJO DE TRABAJO	COLECCIÓN ESTRUCTURADA DE ACTIVIDADES Y ELEMENTOS ASOCIADOS QUE PRODUCEN UN RESULTADO DE VALOR
ROL	SON RESPONSABLES POR LLEVAR A CABO LAS ACTIVIDADES DEL PROCESO, PUEDEN SER PERSONAS O HERRAMIENTAS.
PRODUCTO O ARTEFACTO	SON LAS ENTRADAS Y SALIDAS DE LAS ACTIVIDADES (DOCUMENTACION, MODELOS, REPORTES, PLANES, ETC)

DISCIPLINA	COJUNTO INTEGRADO POR ACTIVIDADES RELATIVAS A UNA RAMA PARTICULAR DE CONOCIMIENTO, EJ: ANÁLISIS Y DISEÑO.
------------	---



## MODELOS DE PROCESOS

La ingeniería del software establece y se vale de una serie de modelos que instauran y muestran las distintas etapas y estados por los que pasa un producto software, desde su concepción inicial, pasando por su desarrollo, puesta en marcha y posterior mantenimiento, hasta la retirada del producto. A estos modelos se les denomina "Modelos de ciclo de vida del software".



## MODELOS DE PROCESOS

### Modelos tradicionales

Formados por un conjunto de fases o actividades en las que no tienen en cuenta la naturaleza evolutiva del software

- Clasico. lineal o en cascada
- Estructurado
- Basado en prototipos
- Desarrollo rapido de aplicaciones (RAD)

### Modelos evolutivos

Son modelos que se adaptan a la evolución que sufren los requisitos del sistema en función del tiempo

- En espiral
- Evolutivo
- Incremental
- Modelo de desarrollo concurrente

### Modelos orientados a la reutilización

- Basado en componentes
- Proceso Unificado

### Modelos para sistemas orientados a objetos

#### Modelos con un alto grado de iteratividad y solapamiento entre fases

- De agrupamiento
- Fuente
- Basado en componentes
- Proceso Unificado

### Procesos ágiles

- Programacion extrema (XP) Desarrollo de software adaptativo
- Scrum. Crystal

### Modelos para sistemas web

- UML-based Web Engineering

# MODELOS DE PROCESOS

## ■ Modelos tradicionales

Formados por un conjunto de fases o actividades en las que no tienen en cuenta la naturaleza evolutiva del software

- Clásico, lineal o en cascada
- Estructurado
- Basado en prototipos
- Desarrollo rápido de aplicaciones (RAD)

## ■ Modelos evolutivos

Son modelos que se adaptan a la evolución que sufren los requisitos del sistema en función del tiempo

- En espiral
- Evolutivo
- Incremental
- Modelo de desarrollo concurrente

## ■ Modelos orientados a la reutilización

- Basado en componentes
- Proceso Unificado

## ■ Modelos para sistemas orientados a objetos

Modelos con un alto grado de iteratividad y solapamiento entre fases

- De agrupamiento
- Fuente
- Basado en componentes
- Proceso Unificado

## ■ Procesos ágiles

- Programación extrema (XP)
- Desarrollo de software adaptativo
- Scrum, Crystal ...

## ■ Modelos para sistemas web

- UML-based Web Engineering

## MODELO CLASICO, LINEAL O EN CASACADA

ES EL MODELO MAS ANTIGUO PROPUESTO POR WINSTON ROYCE EN 1970

BASADO EN LA MENTALIDAD DE LINEA DE ENSAMBLAJE (CARTESIANO)

ES SENCILLO Y FACIL DE ENTENDER

EL PROYECTO PASA ATRAVES DE UNA SERIE DE FASES

Modelo Clásico, Lineal o en Cascada

Ventajas

Puede ser apropiado, en general, para proyectos estables (especialmente los proyectos con requisitos no cambiantes) y donde es posible y probable que los diseñadores predigan totalmente áreas de problema del sistema y produzcan un diseño correcto antes de que empiece la implementación.

- Funciona bien para proyectos pequeños donde los requisitos están bien entendidos.

Es un modelo en el que todo está bien organizado y no se mezclan las fases. Es simple y fácil de usar.

Debido a la rigidez del modelo es fácil de gestionar ya que cada fase tiene entregables específicos y un proceso de revisión. Las fases son procesadas y completadas de una vez.

## Modelo Clásico, Lineal o en Cascada

### ► Ventajas

- Puede ser apropiado, en general, para proyectos estables (especialmente los proyectos con requisitos no cambiantes) y donde es posible y probable que los diseñadores predigan totalmente áreas de problema del sistema y produzcan un diseño correcto antes de que empiece la implementación.
- Funciona bien para proyectos pequeños donde los requisitos están bien entendidos.
- Es un modelo en el que todo está bien organizado y no se mezclan las fases. Es simple y fácil de usar.
- Debido a la rigidez del modelo es fácil de gestionar ya que cada fase tiene entregables específicos y un proceso de revisión. Las fases son procesadas y completadas de una vez.

## Modelo Clásico, Lineal o en Cascada

### - Inconvenientes

- En la vida real, un proyecto rara vez sigue una secuencia lineal, esto crea una mala implementación del modelo, lo cual hace que lo lleve al fracaso.
- Difícilmente un cliente va a establecer al principio todos los requisitos necesarios, por lo que provoca un gran atraso trabajando en este modelo, ya que este es muy restrictivo y no permite movilizarse entre fases

Los resultados y/o mejoras no son visibles progresivamente, el producto se ve cuando ya está finalizado, lo cual provoca una gran inseguridad por parte del cliente que quiere ir viendo los avances en el producto

## Modelo Clásico, Lineal o en Cascada

### ► Inconvenientes

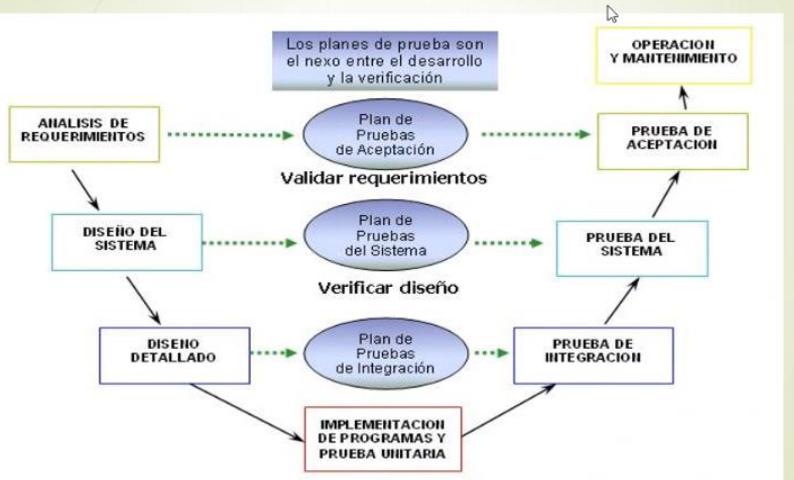
- En la vida real, un proyecto rara vez sigue una secuencia lineal, esto crea una mala implementación del modelo, lo cual hace que lo lleve al fracaso.
- Difícilmente un cliente va a establecer al principio todos los requisitos necesarios, por lo que provoca un gran atraso trabajando en este modelo, ya que este es muy restrictivo y no permite movilizarse entre fases.
- Los resultados y/o mejoras no son visibles progresivamente, el producto se ve cuando ya está finalizado, lo cual provoca una gran inseguridad por parte del cliente que quiere ir viendo los avances en el producto.

## MODELO EN V

## Modelo V

- El modelo en V se desarrolló para terminar con algunos de los problemas que se vieron utilizando el enfoque de cascada tradicional
- Dice que las pruebas necesitan empezarse lo más pronto posible en el ciclo de vida.
- Es un modelo que ilustra cómo las actividades de **prueba** (verificación y validación) se pueden integrar en cada fase del ciclo de vida. Dentro del modelo en V, las pruebas de validación tienen lugar especialmente durante las etapas tempranas, por ejemplo, revisando los requisitos de usuario y después por ejemplo, durante las pruebas de aceptación de usuario.
- Describe las actividades y resultados que han de ser producidos durante el desarrollo del producto. La **parte izquierda de la V** representa la descomposición de los requisitos y la creación de las especificaciones del sistema. El **lado derecho de la V** representa la integración de partes y su verificación.
- V significa "Validación y Verificación".

## Ciclo de Vida en V



## MODELO V VENTAJAS

## Modelo V

### ► Ventajas

- Es un modelo simple y fácil de utilizar.
- En cada una de las fases hay entregables específicos.
- Tiene una alta oportunidad de éxito sobre el modelo en cascada debido al desarrollo de planes de prueba en etapas tempranas del ciclo de vida.
- Es un modelo que suele funcionar bien para proyectos pequeños donde los requisitos son entendidos fácilmente.

## CLASE 3

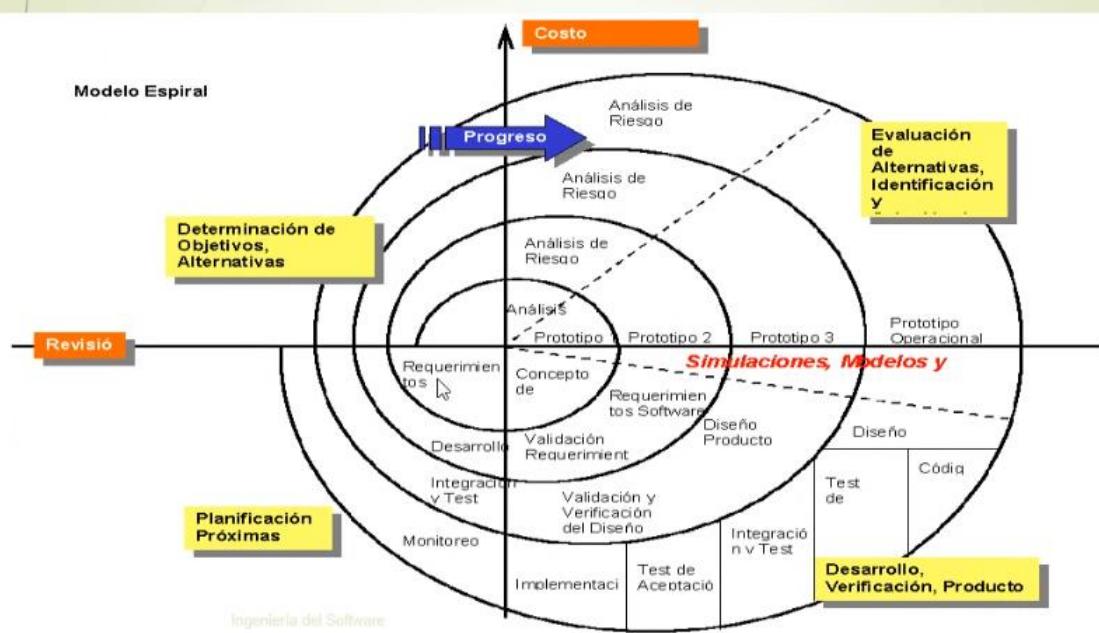
## Modelo del Ciclo de Vida en Espiral

- Es un modelo de ciclo de vida desarrollado por Barry Boehm en 1985
- Consiste en una serie de ciclos que se repiten. Cada uno tiene las mismas fases y cuando termina da un producto ampliado con respecto al ciclo anterior.
- En este sentido es parecido al modelo incremental, la diferencia importante es que tiene en cuenta el **concepto de riesgo**. Un **riesgo** puede ser muchas cosas: requisitos no comprendidos, mal diseño, errores en la implementación, etc.
- Al terminar una iteración se comprueba que lo que se ha hecho efectivamente cumple con los requisitos establecidos, también se verifica que funciona correctamente. El propio cliente evalúa el producto. No existe una diferencia muy clara entre cuando termina el proyecto y cuando empieza la fase de mantenimiento. Cuando hay que hacer un cambio, este puede consistir en un nuevo ciclo.

## Modelo del Ciclo de Vida en Espiral

- Este modelo de desarrollo combina las características **del modelo de prototipos** y el **modelo en cascada**. El modelo en espiral está pensado para proyectos largos, caros y complicados.
- Esto modelo no fue el primero en tratar el desarrollo iterativo, pero fue el primer modelo en explicar las iteraciones.
- Este modelo fue propuesto por Boehm en 1988 en su artículo "A Spiral Model of Software Development and Enhancement". Básicamente consiste en una serie de ciclos que se repiten en forma de espiral, comenzando desde el centro. Se suele interpretar como que dentro de cada ciclo de la espiral se sigue un modelo en cascada, pero no necesariamente ha de ser así.
- Este sistema es muy utilizado en proyectos grandes y complejos como puede ser, por ejemplo, la creación de un sistema operativo.

# Modelo del Ciclo de Vida en Espiral



## MODELO ESPIRAL VENTAJAS

### Modelo del Ciclo de Vida en Espiral

#### ► Ventajas

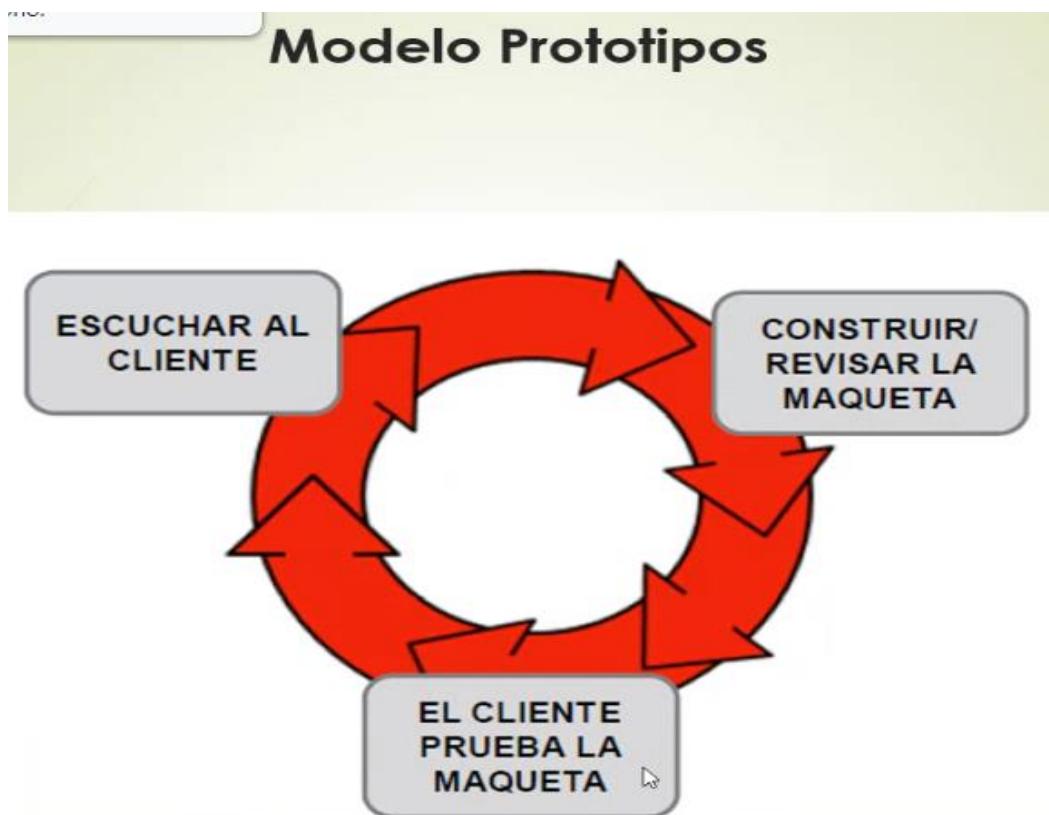
- El análisis de riesgos se hace de forma explícita y clara. Une los mejores elementos de los restantes modelos. Entre ellos:
  - Reduce riesgos del proyecto
  - Incorpora objetivos de calidad
  - Integra el desarrollo con el mantenimiento
- Además es posible tener en cuenta mejoras y nuevos requerimientos sin romper con el modelo, ya que el ciclo de vida no es rígido ni estático.
- Mediante este modelo se produce software en etapas tempranas del ciclo de vida y suele ser adecuado para proyectos largos de misión crítica.

# Modelo del Ciclo de Vida en Espiral

## ► Comparación con otros modelos

- La distinción más destacada entre el modelo en espiral y otros modelos de software es la tarea explícita de evaluación de riesgos. Para otros modelos la evaluación de riesgos es una subtarea, por ejemplo, de la planificación y gestión global. Además no hay fases fijadas para la especificación de requisitos, diseño y pruebas en el modelo en espiral. Se puede usar prototipado para encontrar y definir los requisitos.
- La diferencia es que en el incremental se parte de que no hay incertidumbre en los requisitos iniciales; en este, en cambio, se es consciente de que se comienza con un alto grado de incertidumbre. En el incremental suponemos que conocemos el problema y lo dividimos. Este modelo gestiona la incertidumbre.

## MODELOS PROTOTIPOS



## Modelo Prototipos

- Un cliente, a menudo, define un conjunto de objetivos generales para el software, pero no identifica los requisitos detallados de entrada, proceso o salida. En otros casos, el **responsable del desarrollo del software puede no estar seguro de la eficiencia de un algoritmo, de la calidad de adaptación de un sistema operativo, o de la forma en que debería tomarse la interacción hombre-máquina**. En estas y en otras muchas situaciones, un paradigma de construcción de prototipos puede ofrecer el mejor enfoque.
- El paradigma de construcción de prototipos comienza con la recolección de requisitos. El desarrollador y el cliente encuentran y definen los objetivos globales para el software, identifican los requisitos conocidos y las áreas del esquema en donde es obligatoria más definición. Entonces aparece un diseño rápido. El diseño rápido se centra en una representación de esos aspectos del software que serán visibles para el usuario/cliente. El diseño rápido lleva a la construcción de un prototipo. El prototipo lo evalúa el cliente/usuario y se utiliza para refinar los requisitos del software a desarrollar. La iteración ocurre cuando el prototipo se pone a punto para satisfacer las necesidades del cliente, permitiendo al mismo tiempo que el desarrollador comprenda mejor lo que se necesita hacer.

## Modelo Prototipos

- Se usa un prototipo para dar al usuario una idea concreta de lo que va a hacer el SI.
- Se aplica cada vez más cuando la rapidez de desarrollo es esencial.
- **Prototipado evolutivo:** el prototipo inicial se refina progresivamente hasta convertirse en versión final.
- **Prototipado desecharable:** de cada prototipo se extraen ideas buenas que se usan para hacer el siguiente

## Modelos Evolutivos

### Características:

- ▶ Gestionan bien la naturaleza evolutiva del software.
- ▶ Son iterativos: construyen versiones de software cada vez más completas.

### Se adaptan bien:

- ▶ Los cambios de requisitos del producto.
- ▶ Fechas de entrega estrictas poco realistas.
- ▶ Especificaciones parciales del producto.

meet.google.com está compartiendo tu pantalla. Dejar de compartir Ocultar

## Modelos Evolutivos



## Modelo Evolutivo - Incremental

Fusiona el modelo lineal secuencial con el de construcción de prototipos.

Cada secuencia lineal secuencial produce un incremento del software

- Incremento: producto operacional de una parte del sistema

El primer incremento suele ser el núcleo

- Requisitos básicos
- Muchas funciones suplementarias se dejan para después

Se evalúa (por el cliente) el producto entregado

- Como resultado se desarrolla un plan para el siguiente

Se itera

- Hasta elaborar el producto completo

## Modelo Evolutivo - Incremental

Fusiona el modelo lineal secuencial con el de construcción de prototipos.

Cada secuencia lineal secuencial produce un incremento del software

- Incremento: producto operacional de una parte del sistema

El primer incremento suele ser el núcleo

- Requisitos básicos
- Muchas funciones suplementarias se dejan para después

Se evalúa (por el cliente) el producto entregado

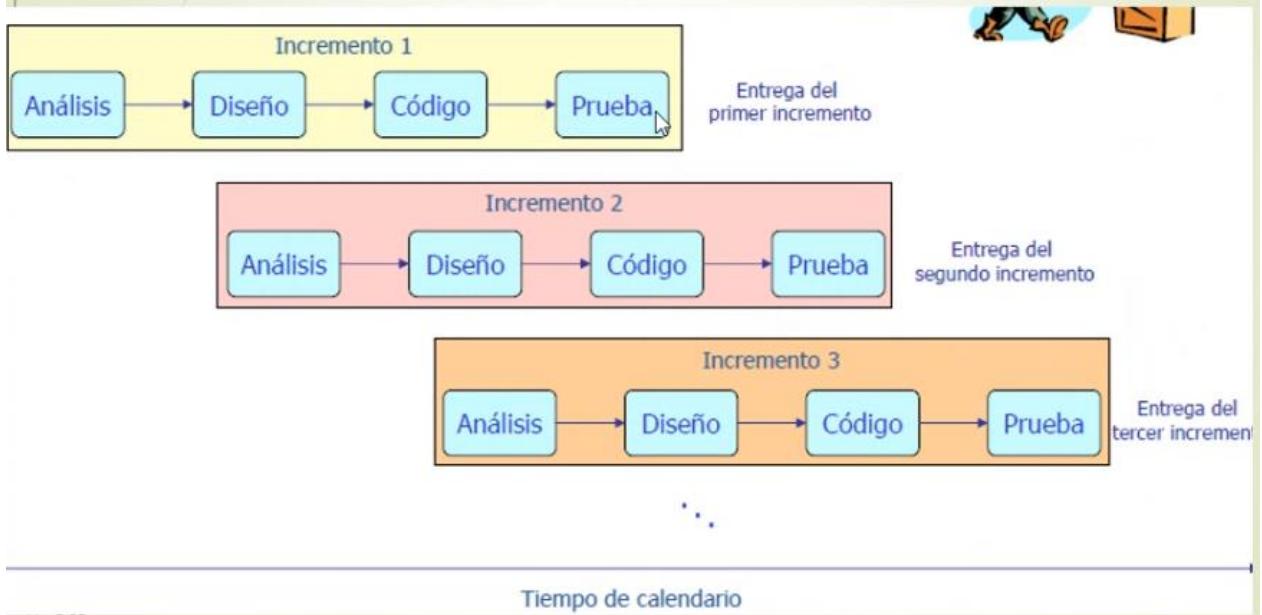
- Como resultado se desarrolla un plan para el siguiente

Se itera

- Hasta elaborar el producto completo

Presiona Esc para salir de la pantalla completa

## Modelo Evolutivo – Incremental



### Ventajas

- **Es interactivo**

Con cada incremento se entrega al cliente un producto operacional al cliente, que puede evaluarlo

- **Personal**

Permite variar el personal asignado a cada iteración

- **Gestión riegos técnicos**

Por ejemplo, disponibilidad de hardware específico

### Inconvenientes

La primera iteración puede plantear los mismos problemas que en un modelo lineal secuencial.

## Modelos Iterativo

- Entrega un sistema completo desde el principio, para posteriormente cambiar la funcionalidad de cada subsistema con cada versión.



### Características del ciclo iterativo [Muller, 1997]

- Se basa en la evolución de prototipos ejecutables, mensurables y evaluables
- Se van incorporando cambios en cada iteración
- Exige más atención e implicación de todos los actores del proyecto

### Minicascada

- Cada iteración reproduce el ciclo de vida en cascada, pero a una escala menor.
- Los objetivos de cada iteración se establecen en función de la evaluación de las iteraciones precedentes.
- Las fases tradicionales se cubren gradualmente en las diversas iteraciones.
- Las actividades internas se solapan porque dentro de una iteración no necesitan terminarse de golpe, siendo la transición entre dos actividades progresiva

## Modelos Iterativo



# Modelos Iterativo

## Evaluación de las iteraciones

- Deben definirse criterios de evaluación de las iteraciones
- Una iteración se marca por etapas intermedias que permitan medir los progresos. Debe haber al menos dos etapas
  - Revisión inicial: fija los objetivos y criterios de la iteración
  - Revisión de evaluación: valida los resultados

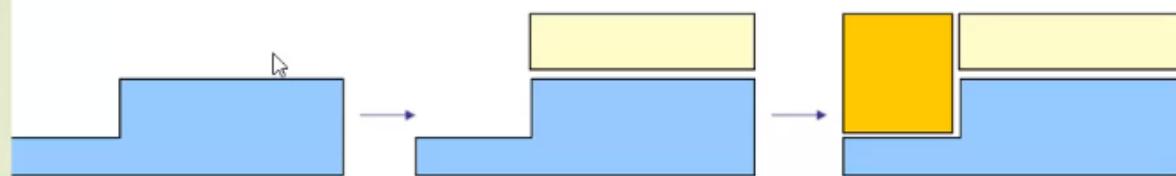
## Mitos sobre el ciclo de vida iterativo [Muller, 1997]

- El ciclo de vida iterativo engendra problemas.
- El ciclo de vida iterativo e incremental exige recomenzar n veces hasta que el resultado sea el adecuado.
- El ciclo de vida iterativo es una excusa para no planificar y gestionar un proyecto.
- El ciclo de vida iterativo sólo concierne a los desarrolladores.
- El ciclo de vida iterativo favorece siempre añadir nuevas necesidades, sin fin

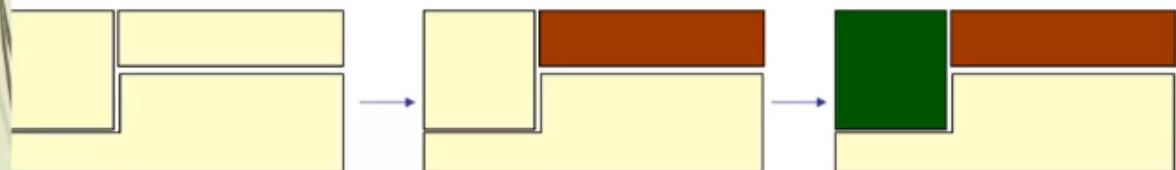
## MODELOS INCREMENTAL VS ITERATIVO

### Modelos Incremental Vs. Iterativo

**Desarrollo incremental:** sistema parcial, funcionalidad completa



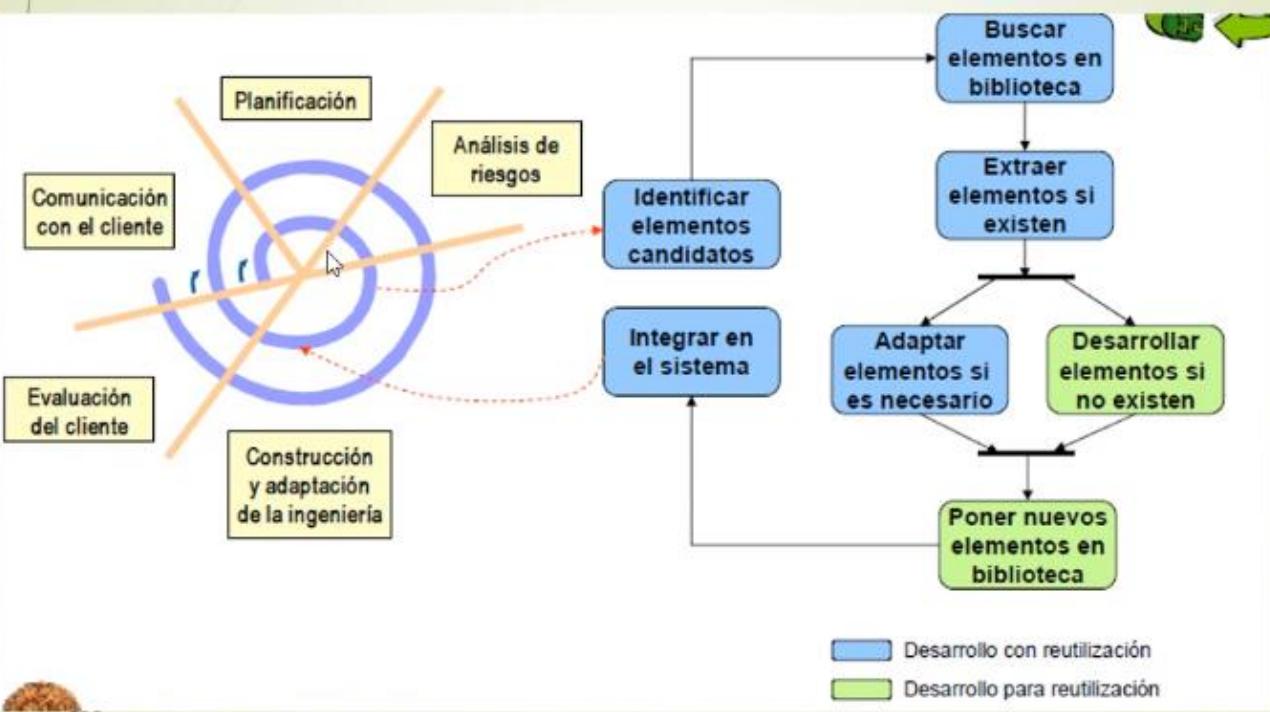
**Desarrollo iterativo:** sistema completo; funcionalidad parcial



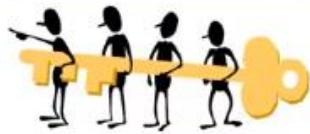
# Modelos Basados en Reutilización

- Esta aproximación se basa en la existencia de un número significativo de elementos reutilizables
- El proceso de desarrollo del sistema se centra en la integración de estos elementos en un sistema, en lugar de desarrollarlo desde cero
- Incorpora muchas características del modelo en espiral
  - Es evolutivo por naturaleza [Nierstrasz, 1992] y existe un enfoque iterativo para la creación de software
- El proceso software tiende a estructurarse en dos subprocessos distintos y separados [Karlsson, 1995]
  - El desarrollo para reutilización
    - Construcción de elementos reutilizables dentro de un dominio concreto
  - El desarrollo con reutilización
    - Construcción de aplicaciones utilizando elementos reutilizables

# Modelos Basados en Reutilización



## Metodologías



### Metodología de Desarrollo de Software:

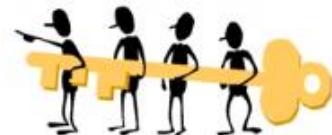
1. Conjunto de pasos y procedimientos que deben seguirse para el desarrollo de software.
2. Conjunto de filosofías, fases, procedimientos, reglas, técnicas, herramientas, documentación y aspectos de formación para los desarrolladores de SI.
3. **Conjunto de procedimientos, técnicas, herramientas y soporte documental que ayuda a los desarrolladores a realizar nuevo software.**

### Metodologías – Definiciones de Autores



- Conjunto de filosofías, fases, procedimientos, reglas, técnicas, herramientas, documentación y aspectos de formación para los desarrolladores de sistemas de información. Según esto, una metodología es un conjunto de componentes que especifican: *Cómo dividir un proyecto en etapas; Qué tareas se llevarán a cabo en cada etapa; Qué salidas se producen y cuándo deben producirse; Qué restricciones se aplican; Qué herramientas van a ser utilizadas; Cómo se gestiona y se controla el proyecto* [Maddison, 1983]
- Una metodología es una aproximación organizada y sistemática para el ciclo de vida del sistema o sus partes. Especifica las tareas individuales y sus secuencias [Palvia y Nosek, 1993]
- Conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar nuevo software [Piattini et al., 2004]

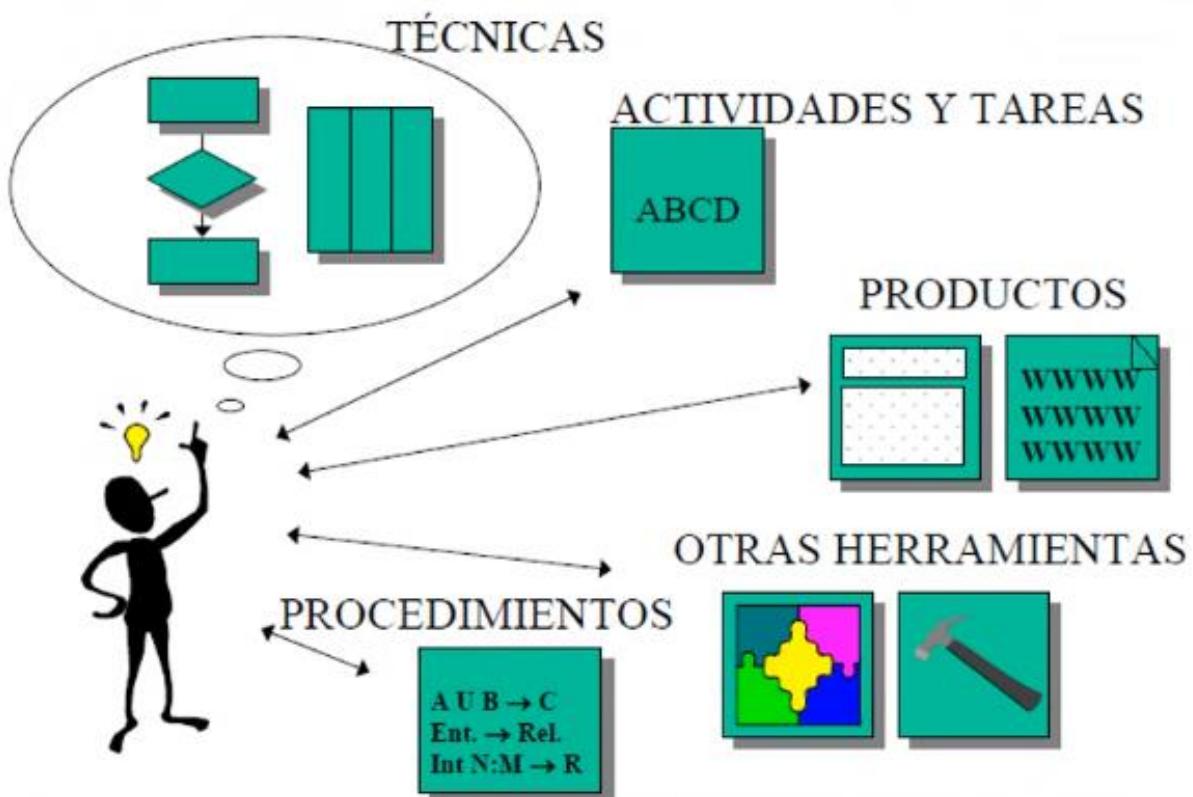
# Metodologías



- Desde una perspectiva de Ingeniería de Software, una metodología
  - Describe cómo se organiza un proyecto
  - Establece el orden en el que la mayoría de las actividades tienen que realizarse y los enlaces entre ellas
  - Indica cómo tienen que realizarse algunas tareas proporcionando las herramientas concretas e intelectuales
- Con una metodología se intentan cubrir las siguientes necesidades [Piattini et al., 2004]
  - Mejores aplicaciones
  - Mejor proceso de desarrollo
  - Establecer un proceso estándar en una organización

## METODOLOGIAS

### Metodologías – Elementos





## Metodologías – Elementos

### • Actividades y Tareas

- El **Proceso** se descompone hasta el nivel de Actividades y Tareas (actividades elementales)

### • Procedimientos

- Definen la forma de llevar a cabo las **Tareas**
- Vínculo de Comunicación entre Usuarios y Desarrolladores

### • Productos

- Obtenidos como resultado de seguir un **Procedimiento**
- Pueden ser Intermedios o Finales



## Metodologías – Elementos

### • Actividades y Tareas

- El **Proceso** se descompone hasta el nivel de Actividades y Tareas (actividades elementales)

### • Procedimientos

- Definen la forma de llevar a cabo las **Tareas**
- Vínculo de Comunicación entre Usuarios y Desarrolladores

### • Productos

- Obtenidos como resultado de seguir un **Procedimiento**
- Pueden ser Intermedios o Finales

## Metodologías – Elementos

- Ciclo de Vida vs Metodología vs Método
  - Una **Metodología** puede seguir uno o varios modelos de Ciclo de Vida.
  - Un **Ciclo de Vida** indica qué obtener, pero no cómo.
  - Metodología  $\equiv$  Ciclo de Vida + ¿cómo?
    - Detalle de técnicas, procedimientos y artefactos.
  - Metodología >>> **Método**
    - Podemos considerar una metodología como un conjunto de métodos + ....

**METODOLOGÍA = CICLO DE VIDA + colección de MÉTODOS**  
para llevarlo a cabo

## OBJETIVOS DE LAS METODOLOGIAS.

## Objetivos de las Metodologías

- Establecer los requisitos de un sistema software de una forma acertada
- Proporcionar un método sistemático de desarrollo de forma que se pueda controlar su proceso
- Construir un sistema software dentro de un tiempo apropiado y unos costes aceptables
- Construir un sistema que esté bien documentado y que sea fácil de mantener
- Ayudar a identificar, lo antes posible, cualquier cambio que sea necesario realizar dentro del proceso de desarrollo
- Proporcionar un sistema que satisfaga a todas las personas afectadas por el mismo

[Piattini et al., 2004]

## Características deseables de una Metodología



- Una metodología debe cubrir [Henderson-Sellers y Firesmith, 1999]
  - Un proceso de ciclo de vida completo, que comprenda aspectos tanto del negocio como técnicos
  - Un conjunto completo de conceptos y modelos que sean internamente consistentes
  - Una colección de reglas y guías
  - Una descripción completa de artefactos a desarrollar
  - Una notación con la que trabajar, idealmente soportada por diversas herramientas CASE y diseñada para una usabilidad óptima
  - Un conjunto de técnicas probadas
  - Un conjunto de métricas, junto con asesoramiento sobre calidad, estándares y estrategias de prueba
  - Identificación de los roles organizacionales
  - Guías para la gestión de proyectos y aseguramiento de la calidad
  - Asesoramiento para la gestión de bibliotecas y reutilización

## METODOLOGIAS – EVOLUCION

En los ultimos 30 años se ha desarrollado software siguiendo tres filosofías principales:

- Convencional:

### Metodologías - Evolución



#### Desarrollo Convencional:



- ❖ Años 50
- ❖ Desarrollo artesanal y ausencia de Metodología
- ❖ Enfocado en la Tarea de Programación
- ❖ Inconvenientes:
  - ⌚ Los resultados finales son impredecibles
  - ⌚ No hay forma de controlar lo que está sucediendo en el Proyecto
  - ⌚ Los cambios organizativos afectan negativamente al proceso de desarrollo
  - ⌚ El éxito de los proyectos se basa mucho en la figura del “héroe”

y los héroes siempre acaban cansándose

- Estructurada

## Metodologías - Evolución



### Evolución del Desarrollo Estructurado:

- ❖ Años 60 (entorno académico), mediados 70 (industria)

#### Programación Estructurada:



- ✓ Normas para escribir código
- ✓ Facilitar comprensión de Programas
- ✓ Normas para la aplicación de estructuras de datos y de control

#### Convencional

```

10      CLS
20      A=10
30      INPUT B
40      IF B=A THEN GOTO 50 ELSE GOTO 70
50      PRINT "A Y B SON IGUALES"
60      GOTO 100
70      IF A>B THEN GOTO 80 ELSE GOTO 90
80      B= B + 1; GOTO 40
90      B= B - 1; GOTO 40
100     END
    
```

Francisco Ruiz, Michael González Harbour - IS1

Convencional	Estructurado
<pre> 10      CLS 20      A=10 30      INPUT B 40      IF B=A THEN GOTO 50 ELSE GOTO 70 50      PRINT "A Y B SON IGUALES" 60      GOTO 100 70      IF A&gt;B THEN GOTO 80 ELSE GOTO 90 80      B= B + 1; GOTO 40 90      B= B - 1; GOTO 40 100     END     </pre>	<pre> PROGRAM NUMEROSIGUALES BEGIN   CLEARSCREEN;   A :=10;   INPUT B;   REPEAT     IF B=A THEN PRINT "A Y B SON IGUALES"     ELSE REDUCEDIFERENCIA(A,B);   UNTIL B=A; END; PROCEDURE REDUCEDIFERENCIA(A,B); BEGIN   IF A&gt;B THEN B:= B+1   ELSE B:= B - 1 END     </pre>



DISEÑO ESTRUCTURADO MITAD DE LOS 70

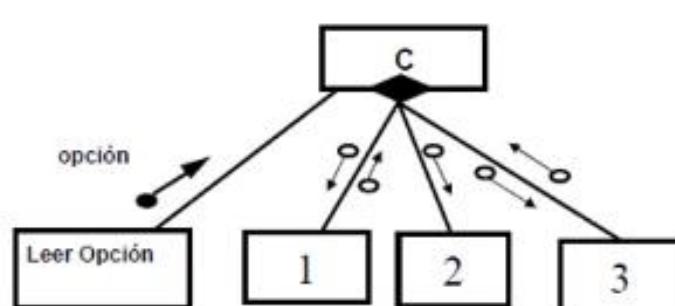
## Metodologías - Evolución



### Evolución del Desarrollo Estructurado:

- ❖ Diseño Estructurado (mitad años 70)

- ✓ Mayor nivel abstracción (independencia del lenguaje programación)
- ✓ Elemento básico de diseño: Módulo
- ✓ Modularidad. Medidas de Calidad de Programas





## Metodologías - Evolución

### Evolución del Desarrollo Estructurado:

#### III. Análisis Estructurado (finales años 70)

- ✓ Previamente: Descripción Narrativa Requisitos → Especificaciones:
  - Monolíticas
  - Redundantes
  - Ambiguas
  - Imposibles de Mantener
- ✓ Se obtienen **Especificaciones Funcionales**:
  - Gráficas
  - Particionadas
  - Mínimamente redundantes
- Orientada a Objetos

## Metodologías - Evolución



### Desarrollo Orientado a Objetos:

- ❖ Esencia: Identificación y organización de conceptos del dominio de la aplicación y no tanto de su representación final en un lenguaje de programación
- ❖ Años 80
- ❖ Trata **Funcionalidad y Datos de forma conjunta**.
- ❖ Principios:
  - ✓ Abstracción
  - ✓ Ocultación de Información (Encapsulamiento)
  - ✓ Modularidad
- ❖ Las técnicas estructuradas han influido en estas metodologías.

## Metodologías - Evolución

- Se fundamentan en la integración de los dos aspectos de los sistemas de información: **datos y procesos**
- En este paradigma un sistema se concibe como un conjunto de objetos que se comunican entre sí mediante mensajes
- El objeto encapsula datos y operaciones
  - Este enfoque permite un modelado más natural del mundo real y facilita enormemente la reutilización del software
- Las metodologías orientadas a objetos acortan la distancia existente entre el espacio de conceptos (lo que los expertos o usuarios conocen) y el espacio de diseño e implementación

### PROCESO UNIFICADO

## Proceso Unificado

- Los autores de UML
  - Booch: método Booch
  - Rumbaugh: OMT
  - Jacobson: proceso Objectory
- También conocido como RUP: Rational Unified Process

# Proceso Unificado



- Es un proceso de desarrollo de software
  - Dirigido por casos de uso
  - Centrado en la arquitectura
  - Iterativo e incremental
- Utiliza UML para definir los modelos del sistema software
- El sistema software en construcción está formado por
  - *componentes* software
  - interconectados a través de *interfaces*



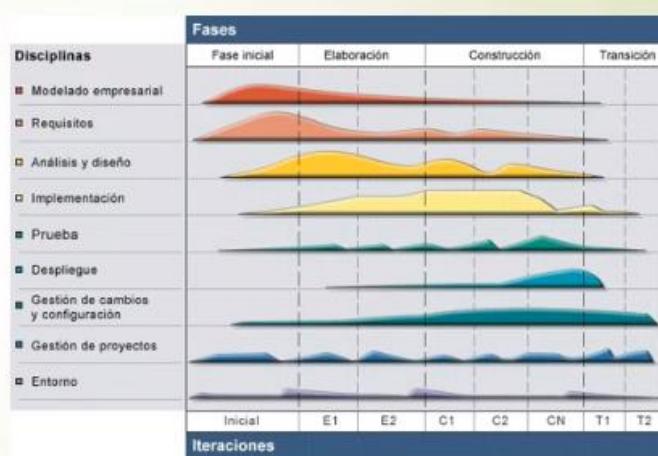
Ac  
Ve

## Proceso Unificado



RUP determina que el ciclo de vida del proyecto consiste en **cuatro fases**.

Estas **fases** permiten que el proceso sea presentado a alto nivel de una forma similar a como sería presentado un proyecto basado en un estilo en cascada, aunque en esencia la clave del proceso recae en las **iteraciones de desarrollo** dentro de todas las fases. También, cada **fase** tiene un objetivo clave y un **hito al final** que denota que el objetivo se ha logrado.



Activar  
Ve a Com

- MODELADO DE NEGOCIO
- REQUISITOS
- ANALISIS Y DISEÑO
- IMPLEMENTACION
- PRUEBA
- DESPLIEGUE
- GESTION DE CAMBIOS Y CONFIGURACION
- GESTION DE PROYECTOS
- ENTORNO

#### FASES DE MODELO RUP:

- FASE INICIAL
- ELABORACION
- CONSTRUCCION
- TRANSICION

## Proceso Unificado



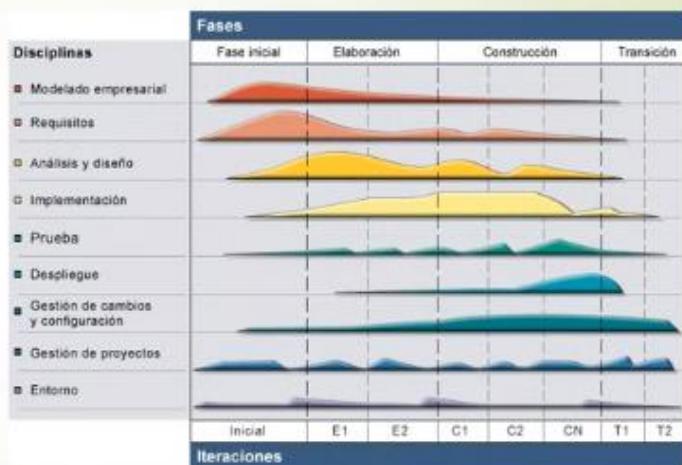
Dentro de cada *iteración*, las tareas se categorizan en nueve disciplinas:

### - Seis disciplinas de ingeniería

- ❖ Modelaje de negocio
- ❖ Requisitos
- ❖ Análisis y diseño
- ❖ Implementación
- ❖ Pruebas
- ❖ Despliegue

### - Tres disciplinas de soporte

- ❖ Gestión de la configuración y del cambio
- ❖ Gestión de proyectos
- ❖ Entorno



## Proceso Unificado



RUP se basa en un conjunto de **módulos o elementos** de contenido, que describen qué se va a producir, las habilidades necesarias requeridas y la explicación paso a paso describiendo cómo se consiguen los objetivos de desarrollo.

Los **módulos principales**, o **elementos de contenido**, son:

**Roles** (quién): un rol define un conjunto de habilidades, competencias y responsabilidades relacionadas.

**Productos de trabajo** (qué): un producto de trabajo representa algo que resulta de una tarea, incluyendo todos los documentos y modelos producidos mientras que se trabaja en el proceso.

**Tareas** (cómo): una tarea describe una unidad de trabajo asignada a un rol que proporciona un resultado significante.

## DESARROLLO AGIL

## Desarrollo Agil

El desarrollo de software ágil hace referencia a un grupo de metodologías de desarrollo de software que se basan en principios similares. Generalmente promueven:

- ❖ Un proceso de gestión de proyectos que fomenta la inspección y adaptación frecuente.
- ❖ Una filosofía líder que fomenta trabajo en equipo, organización propia y responsabilidad.
- ❖ Un conjunto de mejores prácticas de ingeniería que permite la entrega rápida de software de alta calidad.
- ❖ Un enfoque de negocio que alinea el desarrollo con las necesidades de los clientes y los objetivos de la compañía

## Extreme Programming (XP)

### ■ Modelo de proceso de Kent Beck

*Un modo ligero, eficiente, de bajo riesgo, flexible, predecible, científico y divertido de producir software*

### ■ Características

- Alta visibilidad debido a ciclos muy cortos
- Planificación incremental
- Se adapta a cambios de negocio

## Extreme Programming (XP)

Es una metodología ágil centrada en **potenciar las relaciones interpersonales** como clave para el éxito en el desarrollo de software, **promoviendo el trabajo en equipo**, **preocupándose por el aprendizaje de los desarrolladores**, y **propiciando un buen clima de trabajo**.

Se **basa** en :

- La realimentación continua entre el cliente y el equipo de desarrollo
- Comunicación fluida entre todos los participantes
- Simplicidad en las soluciones implementadas
- Coraje para enfrentar los cambios.

## Extreme Programming (XP)

XP se **define** como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

Los principios y prácticas son de sentido común pero llevadas al extremo, de ahí proviene su nombre.



### Elementos de la metodología

- Las historias de usuario
- Roles
- Proceso

### ELEMENTOS DE METODOLOGIA XP

## Extreme Programming (XP)

### Elementos de la metodología - Las historias de usuario

Son la técnica utilizada para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales.

El tratamiento de las historias de usuario es muy dinámico y flexible.

Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarlas en unas semanas. Las historias de usuario se descomponen en tareas de programación y se asignan a los programadores para ser implementadas durante una iteración .

## Extreme Programming (XP)

### Elementos de la metodología - Roles

**Programador:** el programador escribe las pruebas unitarias y produce el código del sistema

**Cliente:** escribe las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en apoyar mayor valor al negocio.

**Encargado de pruebas (tester):** ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para las pruebas.

**Encargado de seguimiento (tracker):** proporciona realimentación al equipo. Verifica el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, para mejorar futuras estimaciones. Realiza el seguimiento del progreso de cada iteración.

### EXTREME PROGRAMMING (XP):



## Extreme Programming (XP)

### Elementos de la metodología - Roles

**Entrenador (coach):** es el responsable del proceso global. Debe proveer guías al equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente.

**Consultor:** es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto, en el que puedan surgir problemas.

**Gestor (big boss):** es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es de coordinación

## Extreme Programming (XP)

### Elementos de la metodología - Proceso

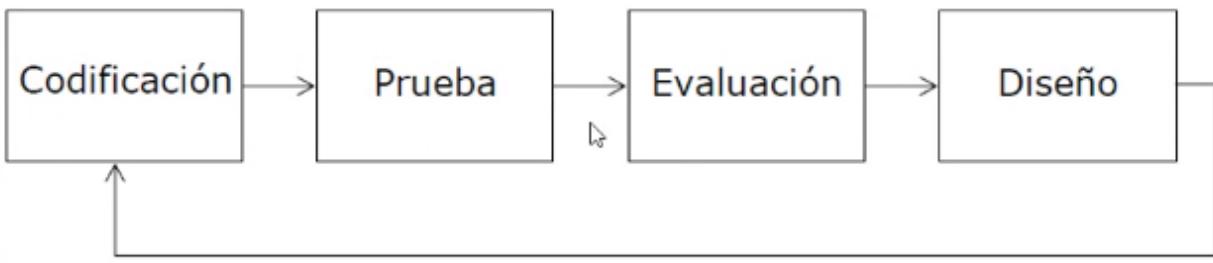
El ciclo de desarrollo consiste (a grandes rasgos) en los siguientes pasos:

1. El cliente define el valor de negocio a implementar
2. El programador estima el esfuerzo necesario para su implementación
3. El programador construye ese valor
4. Vuelve al paso 1

En todas las iteraciones de este ciclo tanto el cliente como el programador aprenden. No se debe presionar al programador a realizar más trabajo que el estimado, ya que se perderá calidad en el software o no se cumplirán los plazos. El ciclo de vida ideal de XP consisten en 6 fases: exploración, planificación de la entrega, iteraciones, producción, mantenimiento y muerte del proyecto.

## Extreme Programming (XP)

- La estructura del proceso, si la hay, es un poco atípica



## Extreme Programming (XP)

### ■ Ventajas:

- Bueno para especificaciones cambiantes
- Fundamentación práctica

### ■ Inconvenientes:

- Poco probado
- Poco compatible con especificaciones/diseños totales
- Sólo funciona con equipos pequeños (hasta diez personas)

## Extreme Programming (XP)

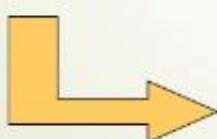
### ■ Diferencias fundamentales (hay más que ya se verán)

- No hay requisitos explícitos sino que el cliente participa en el desarrollo
- Se empieza por automatizar las pruebas
- Se desarrolla siempre la versión más simple posible que resuelva el problema
- Se ejecutan todas las pruebas todos los días
- Se cambia el diseño (aunque sea radicalmente) siempre que haga falta

SEGÚN LAS CARACTERÍSTICAS DADAS DESCRIBA A QUE METODOLOGÍA CORRESPONDE: PROCESOS ITERATIVO; ADOPTA TÁCTICAS DE OTRO ENFOQUE ÁGIL, COMO XP

### 3.- Estándares relacionados con el proceso software. IEEE 1074-1998. Developing Software Life Cycle Processes

- Ciclo de vida: "una aproximación lógica a la adquisición, el suministro, el desarrollo, la explotación y el mantenimiento del software"
- El estándar requiere la definición de un ciclo de vida



pero no implica  
ninguno determinado

#### ESTANDARES IEE 1074

### 3.- Estándares relacionados con el proceso software. IEEE 1074-1998. Developing Software Life

#### Cycle Processes

- Cada organización debe asociar las actividades definidas en el estándar a su propio ciclo de vida del software.



*Si no lo ha definido,  
debe hacerlo*

- El seguimiento del estándar no implica el uso de ningún método específico, ni la creación de determinados documentos



*prescribe los **procesos** del ciclo de  
vida, no los **productos** del mismo.*

### 3.- Estándares relacionados con el proceso software. IEEE 1074-1998. Developing Software Life Cycle Processes

Sección	Título	Procesos
2	Procesos de modelo de ciclo de vida del software	Modelo del Ciclo de vida del software
3	Procesos de gestión del proyecto	Inicio del proyecto Monitorización y control del proyecto Gestión de la calidad del software
4	Procesos pre-desarrollo	Exploración de conceptos Asignación del sistema
5	Procesos de desarrollo	Requisitos Diseño Implementación
6	Procesos post-desarrollo	Instalación Operación y soporte Mantenimiento Fin de uso
7	Procesos integrales	Verificación y validación Gestión de la configuración del software Desarrollo de la documentación Entrenamiento

### 3.- Estándares relacionados con el proceso software. IEEE 1074-1998. Developing Software Life Cycle Processes

35

- Procesos divididos en actividades (obligatorias y opcionales):
  - Información de entrada
  - Descripción
  - Información de salida
- Antes de empezar un proyecto, revisar las actividades para ver si son aplicables, y establecer un orden.
- **Conformidad con el estándar:** realización de todas las actividades obligatorias.

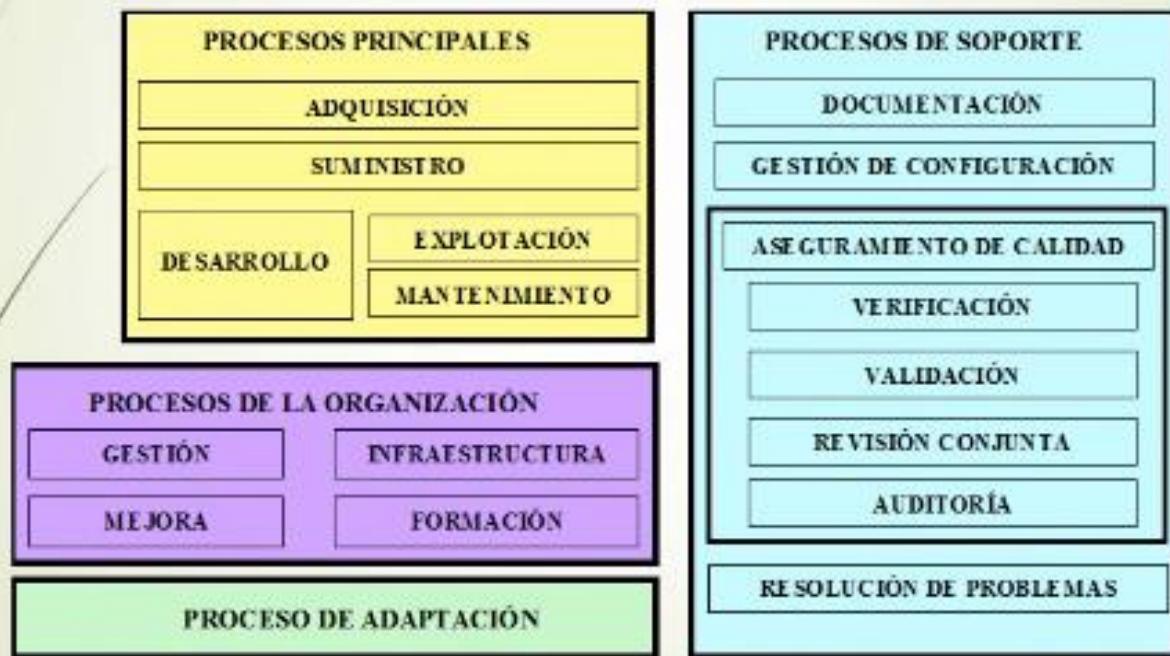
3.- Estándares relacionados con el proceso software.  
Procesos estándar - IEEE/EIA (ISO/IEC) 12207. Information technology – Software life cycle processes (Esteban Piattini 95)

- ▶ Establece un marco común para los procesos de ciclo de vida.
- ▶ Emplea términos bien definidos.
- ▶ Describe el ciclo de vida.
  - Desde la definición de requisitos hasta el fin de uso, y contiene procesos para adquirir y suministrar productos y servicios software.

3.- Estándares relacionados con el proceso software.  
Procesos estándar - IEEE/EIA (ISO/IEC) 12207. Information technology – Software life cycle processes: Ciclo de Vida.

- ▶ También proporciona procesos para **definir, controlar y mejorar** los procesos de ciclo de vida software.
- ▶ El marco descrito por el estándar está diseñado para ser **adaptado** a cada organización y proyecto.
- ▶ El **proceso de adaptación** consiste en la eliminación de procesos, actividades y tareas no aplicables (tb. se pueden añadir).

3.- Estándares relacionados con el proceso software. Procesos estándar - IEEE/EIA (ISO/IEC) 12207. Information technology – Software life cycle processes: *Procesos del Ciclo de Vida.*



3.- Estándares relacionados con el proceso software. Procesos estándar - IEEE/EIA (ISO/IEC) 12207. Information technology – Software life cycle processes: *Procesos del Ciclo de Vida.*

### Procesos principales:

- Útiles a las personas que inician o realizan el desarrollo, la explotación o el mantenimiento del software durante su ciclo de vida
  - compradores, suministradores, personal de desarrollo, operadores y personal de mantenimiento del software

3.- Estándares relacionados con el proceso software.  
Procesos estándar - IEEE/EIA (ISO/IEC) 12207. Information technology – Software life cycle processes: Procesos del Ciclo de Vida.

#### **Procesos de soporte:**

- Sirven de apoyo al resto.
- Contribuyen al éxito y calidad del proyecto software.
- Se aplican en cualquier momento del ciclo de vida.

3.- Estándares relacionados con el proceso software.  
Procesos estándar - IEEE/EIA (ISO/IEC) 12207. Information technology – Software life cycle processes: Procesos del Ciclo de Vida.

#### **Procesos de la organización (procesos generales):**

- Objetivo: establecer, implementar y mejorar la organización  
(gestión, formación del personal, mejora del proceso, etc.)
- Se realizan fuera de proyectos específicos, a nivel organizativo.

3.- Estándares relacionados con el proceso software.  
Procesos estándar - IEEE/EIA (ISO/IEC) 12207. Information technology – Software life cycle processes: Procesos del Ciclo de Vida.

#### Proceso de adaptación:

- ▶ Permite adaptar el estándar a cada proyecto y organización.
- ▶ Factores que influencian la forma de adquirir, desarrollar, explotar o mantener un sistema:
  - ▶ Tamaño y complejidad del proyecto.
  - ▶ Requisitos del sistema.
  - ▶ Métodos de desarrollo.
  - ▶ Variaciones en las políticas y procedimientos de la organización...

#### PROCESO DE ADQUISICION

## Procesos principales: Proceso de adquisición.

- ▶ Actividades y tareas que el comprador, el cliente o el usuario realizan para adquirir un sistema o producto (servicio) software
  - ▶ Preparación y publicación de una solicitud de ofertas.
  - ▶ Selección del suministrador del software.
  - ▶ Gestión de los procesos desde la adquisición hasta la aceptación del producto.

46

## Procesos principales: Proceso de suministro

- Actividades y tareas que realiza el suministrador
  - Se inicia al preparar una propuesta para atender una petición de un comprador, o por la firma de un contrato con el comprador para proporcionarle un producto software
    - Identificación de procedimientos y recursos para gestionar bien el proyecto.
    - Desarrollo de los planes del proyecto.
    - Ejecución de los planes del proyecto hasta la entrega del producto software al comprador.

47

## Procesos principales: Proceso de desarrollo

- Contiene las actividades y tareas realizadas por el desarrollador.
- Integra las siguientes actividades:

Implementación del proceso.

Integración del software.

Análisis de requisitos del sistema.

Prueba del software.

Diseño de la arquitectura del sistema.

Integración del sistema.

Análisis de los requisitos del software.

Prueba del sistema.

Diseño de la arquitectura del software.

Instalación del software.

Diseño detallado del software.

Soporte del proceso de

Codificación y prueba del software.

aceptación del software.

# Ciclo de Vida “CASCADA”



## PROCESO DE DESARROLLO IMPLEMENTACION

### Procesos principales: Proceso de desarrollo. Implementación del proceso

- Si no está especificado en el contrato, el desarrollador definirá un modelo de ciclo de vida
  - apropiado al ámbito, magnitud y complejidad del proyecto.
- Las actividades y tareas del proceso de desarrollo serán seleccionadas y relacionadas con el modelo de ciclo de vida.

## Procesos principales:

Proceso de desarrollo. Implementación del proceso

- Si no están indicados en el contrato el desarrollador deberá seleccionar, adaptar y utilizar aquellos estándares, métodos, herramientas y lenguajes de programación que son apropiados (y están documentados) para realizar las actividades del proceso de desarrollo y de los procesos de soporte.

### PROCESO DE DESARROLLO ANALISIS DE REQUISITOS

## Procesos principales:

Proceso de desarrollo.  
Análisis de requisitos del sistema

- Los requisitos del sistema incluyen:
  - funciones y capacidades
  - requisitos de seguridad
  - requisitos de interacción hombre-máquina
  - interfaces del sistema ↴
  - restricciones aplicables al diseño
  - requisitos de aceptación
  - ...

### Procesos principales:

#### Proceso de desarrollo - Diseño de la arquitectura del sistema

- Se identifica la **arquitectura de alto nivel** del sistema:
  - ▶ Se determinan los principales componentes hardware, software y las operaciones manuales
  - ▶ Se asignan los requisitos del sistema a dichos componentes

### Procesos principales:

#### Proceso de desarrollo - Análisis de los requisitos del software

- Se identifican y documentan los requisitos del software, incluyendo:
  - ▶ especificaciones funcionales y de capacidad (rendimiento de la aplicación, etc.)
  - ▶ interfaces externas
  - ▶ seguridad y protección (de la información, daños personales, etc.)
  - ▶ datos que se van a manejar y requisitos de la BD
  - ▶ requisitos de instalación y de aceptación
  - ▶ requisitos de mantenimiento..

## Procesos principales: Proceso de desarrollo - Análisis de los requisitos del software

- Varios estándares definidos para esta fase:

- ▶ IEEE 830- 1998.  
*Recommended Practice for Software Requirements Specifications*
- ▶ DI-IPSC- 81433. *Software Requirements Specification* (estándar del DoD)  
(<http://diamond.spawar.navy.mil/498/mil-498.html>) (ya superado)

 meet.google.co

## Procesos principales: Proceso de desarrollo - Diseño de la arquitectura del software

- ▶ Componentes principales del software
- ▶ Versión preliminar de los manuales de usuario
- ▶ Requisitos de las pruebas
- ▶ Planificación de la integración del software

## Procesos principales: Proceso de desarrollo - Diseño detallado del software

- ▶ Diseño detallado de cada componente sw.
- ▶ Diseño detallado de las interfaces.
- ▶ Diseño detallado de la BD
- ▶ Actualizar manuales de usuario.
- ▶ Def. y documentar los req. de prueba.
- ▶ Actualizar req. de prueba para la integración del sw.
- ▶ Evaluar todo lo anterior.
- ▶ Reuniones de revisión.

### PROCESO DE DESARROLLO CODIFICACION

## Procesos principales: Proceso de desarrollo - Codificación y prueba del software

- ▶ Se desarrollan los componentes software y las bases de datos
- ▶ Se prueban los componentes (prueba de unidad)
- ▶ Se actualizan los manuales de usuario

## Procesos principales: Proceso de desarrollo - Actividades finales

### ► **Integración del software**

- Se integran los componentes del software y se prueban según sea necesario

### ► **Prueba del software**

- De acuerdo con los requisitos de cualificación (validación) especificados para el software

### ► **Integración del sistema**

- Se integra hardware, software y operaciones manuales.

## Procesos principales: Proceso de desarrollo - Actividades finales

### ► **Prueba del sistema**

- Análoga a la del software, pero de acuerdo con los requisitos de cualificación especificados para el sistema

### ► **Instalación del software**

- En el entorno donde vaya a funcionar
- Cuando reemplace a otro sistema, el estándar recomienda mantener funcionamiento paralelo un tiempo

## Procesos principales: Proceso de desarrollo - Actividades finales

### ► Soporte del proceso de aceptación del software

- Finalmente, se debe dar apoyo a la revisión de aceptación y a la prueba del software por el comprador.

## Procesos principales: Proceso de desarrollo - Actividades finales

### ► Integración del software

- Se integran los componentes del software y se prueban según sea necesario

### ► Prueba del software

- De acuerdo con los requisitos de cualificación (validación) especificados para el software

### ► Integración del sistema

- Se integra hardware, software y operaciones manuales.

## Procesos principales: Proceso de desarrollo - Actividades finales

### ► Prueba del sistema

- Análoga a la del software, pero de acuerdo con los requisitos de cualificación especificados para el sistema

### ► Instalación del software

- En el entorno donde vaya a funcionar
- Cuando reemplace a otro sistema, el estándar recomienda mantener funcionamiento paralelo un tiempo

## Procesos principales: Proceso de desarrollo - Actividades finales

### ► Soporte del proceso de aceptación del software

- Finalmente, se debe dar apoyo a la revisión de aceptación y a la prueba del software por el comprador.

## Procesos principales: Proceso de explotación

- ▶ También llamado **de operación**.
- ▶ Explotación del software y del soporte del mismo.
- ▶ La explotación del software está integrada en la del sistema, por lo que las actividades y tareas de este proceso se aplican al sistema completo.

## Procesos principales: Proceso de explotación

- ▶ El sistema debe ser operado de acuerdo con la documentación de usuario en su entorno previsto
- ▶ Entre otras actividades, el operador deberá:
  - ▶ Desarrollar un plan para llevar a cabo las actividades y tareas de este proceso.
  - ▶ Procedimientos para comprobar el producto software en su entorno de operación, enviando informes de problemas y peticiones de modificación al **proceso de mantenimiento**.
  - ▶ El operador debe proporcionar asistencia a los usuarios.

## Procesos principales: Proceso de mantenimiento

- El software o la documentación necesita ser modificado, debido a problemas o a necesidades de mejora o adaptación, p.e.:
  - nuevos errores detectados
  - cambios en la legislación
  - cambios en el entorno
  - necesidad de mejoras
  - migración a un nuevo entorno operativo
  - se va a terminar con su uso...

## Procesos principales: Proceso de mantenimiento

*“Modificar el software existente manteniendo su consistencia”*

- Comprende las siguientes actividades:
  - Implementación del proceso de mantenimiento.
  - Análisis del problema y de la modificación.
  - Implementación de la modificación.
  - Revisión y aceptación del mantenimiento.
  - Migración.
  - Fin de uso del software.

# Procesos de soporte

- ▶ Sirven de apoyo al resto de procesos.
- ▶ Se aplican en cualquier momento del ciclo de vida:
  - ▶ Documentación
  - ▶ Gestión de la configuración
  - ▶ Aseguramiento de la calidad
  - ▶ Verificación
  - ▶ Validación
  - ▶ Revisión conjunta
  - ▶ Auditoría
  - ▶ Resolución de problemas