Dpto. Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur

ELEMENTOS DE BASES DE DATOS

Segundo Cuatrimestre 2015

Clase 25:

Conceptos de Ingeniería de Software

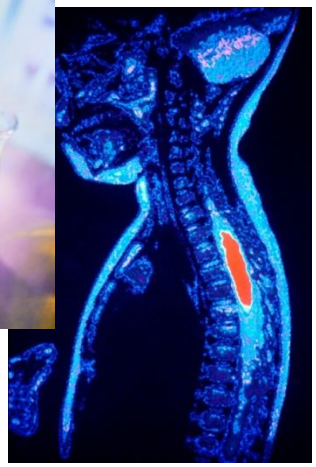
Mg. María Mercedes Vitturini
[mvitturi@uns.edu.ar]

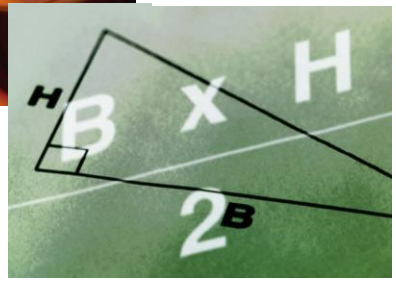
Ingeniería de Software

- Es un área de las ciencias de la computación que estudia la construcción de sistemas de software tan **grandes** y **complejos** que requieren de un grupo de ingenieros.

Construcción de
múltiples versiones de software
por múltiples personas.
(Parnas - 1968)

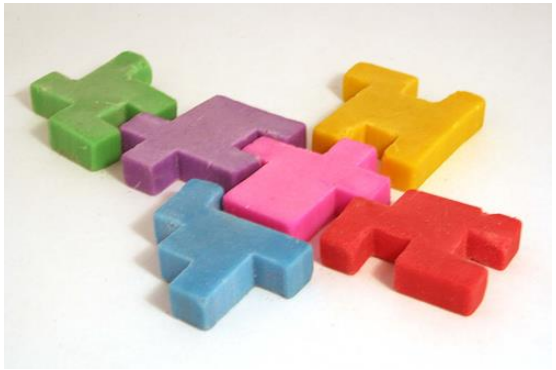
¿Sistemas?





¿Sistemas?

- Sistema de computación,
- Sistema operativo,
- Sistema de liquidación de sueldos,
- Sistema educativo,
- Sistema de gobierno,
- Sistema de ingreso a la UNS,
- Sistema de gestión de calidad,
- Sistema digestivo,
- Sistema numérico,
- Sistema …



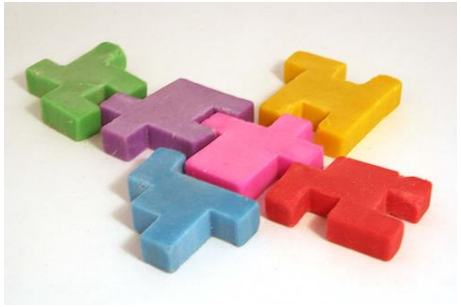
Sistema – Definiciones

Definición

1. Conjunto de **ítems interrelacionados** que interactúan de forma ordenada y **contribuyen a un todo**.

Otras definiciones:

1. Conjunto de **reglas o principios** sobre una materia racionalmente enlazados entre sí que **explican un todo.**
2. **Procedimiento organizado** y establecido.



Sistemas - Ejemplos

- Un grupo de órganos que cumplen una función (sistema digestivo).
- Un grupo de cuerpos interactuando bajo influencia de fuerzas relacionadas (sistema gravitacional).
- Un patrón o arreglo armónico (sistema de numeración)
- Un procedimiento organizado y establecido (sistema de producción)

Sistemas – Clasificación

- **Sistemas naturales**:
  - Sistemas Físicos (geológicos, moleculares, etc.)
  - Sistemas Vivientes (animales, plantas)

- **Sistemas construidos por el hombre**:
  - Manuales.
  - Automatizados: apoyados en TIC's.
  - Mixtos.

De nuestro interés

Sistemas Automatizados (SA)

Sistemas Automatizados: sistemas hechos por el hombre y controlados por una o más computadoras. En general se componen de:

- Hardware: CPU, discos, impresoras, etc.
- Software: sistema operativos, bases de datos, programas de aplicación, etc.
- Personas: proveen y/o consumen lo que produce el sistema.
- Datos: información que se mantiene por período de tiempo.
- Procedimientos: políticas e instrucciones para operar el sistema.
- Documentación: manuales, formularios y otros modelos que describen en sistema.

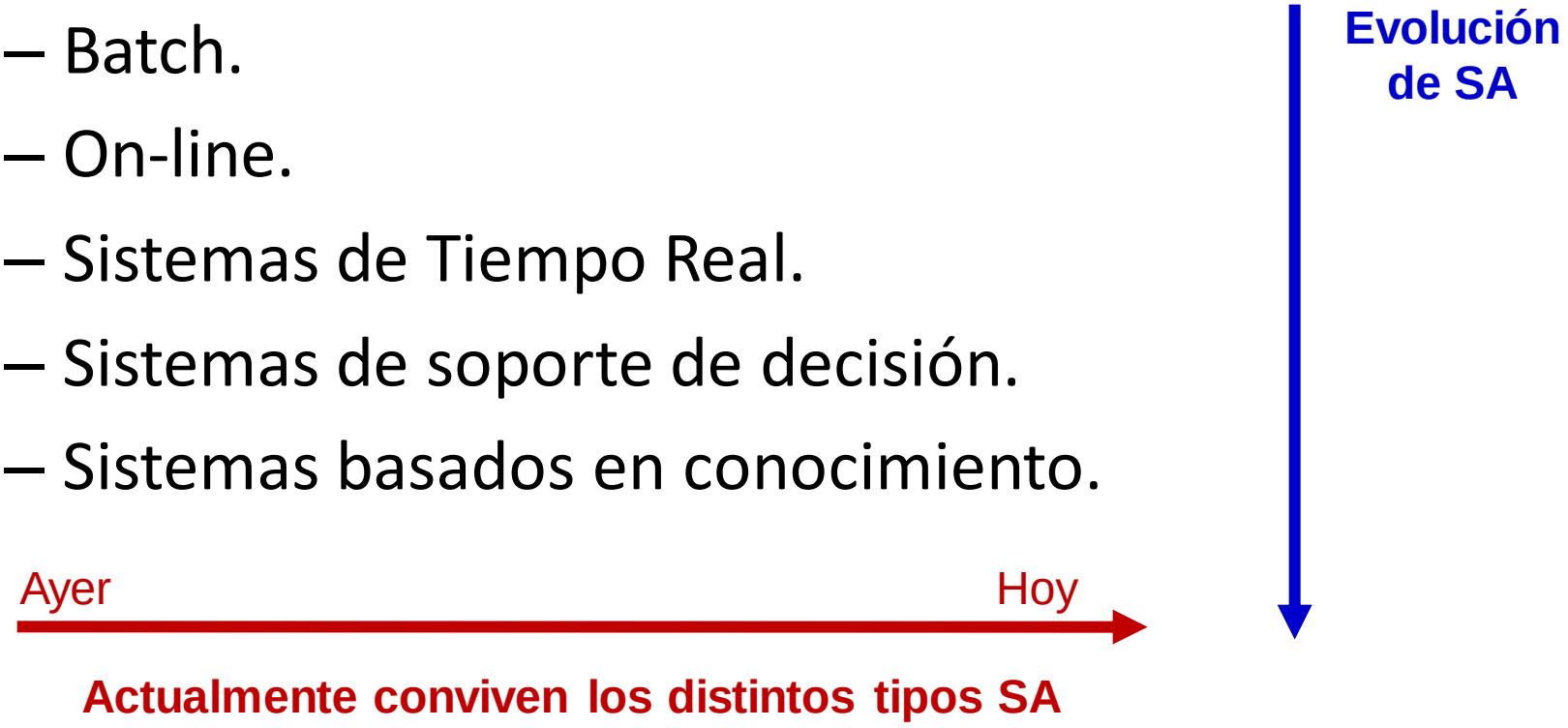
Sistemas de Software

Sistema/Aplicación de software – es una colección de *componentes de software* interrelacionados que trabajan conjuntamente para cumplir algún objetivo.

- Aún los sistemas de software simples se componen de varios componentes.
- El funcionamiento exitoso de cada componente depende del funcionamiento correcto de otros componentes.
- En general los sistemas son jerárquicos e incluyen a otros sistemas que se conocen como *susbistemas.*

Tipos de Sistemas Automatizados

- Evolución de los sistemas de software en el tiempo:
  - Batch.
  - On-line.
  - Sistemas de Tiempo Real.
  - Sistemas de soporte de decisión.
  - Sistemas basados en conocimiento.

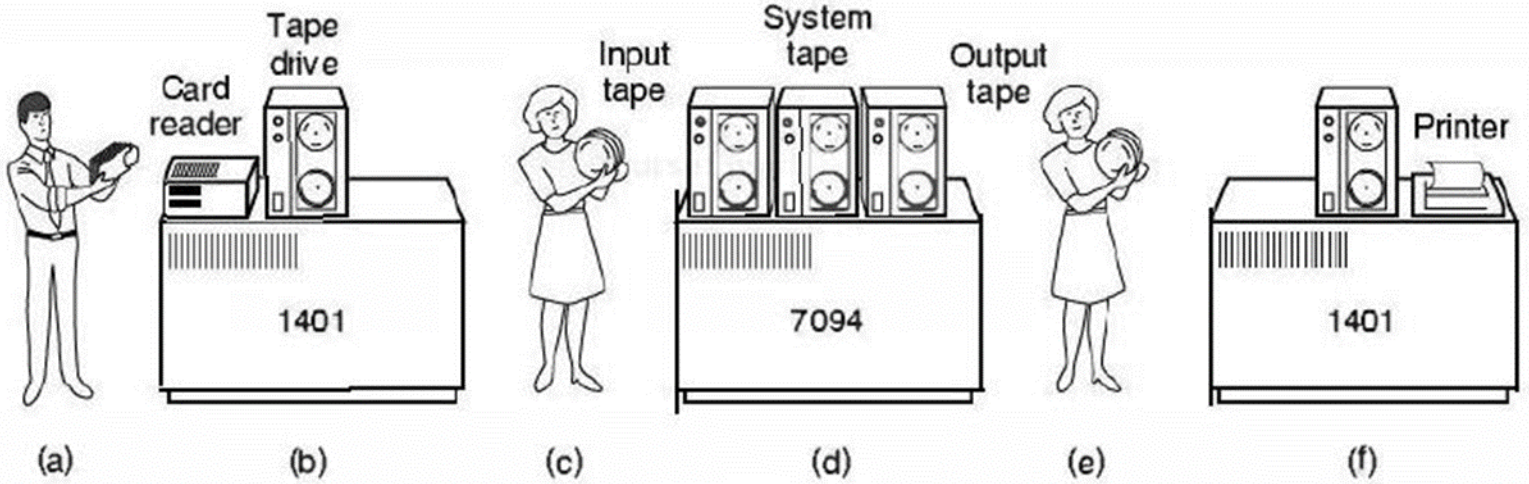


Sistemas Batch

Características

- Recolectan datos por un período de tiempo.
- No interactúan con usuarios.
- Procesan varias transacciones juntas.
- Generalmente tienen acceso secuencial a la mayoría de la información.
  - Ejemplo: políticas de backup, algunos tipos de sensores.

Sistemas Batch



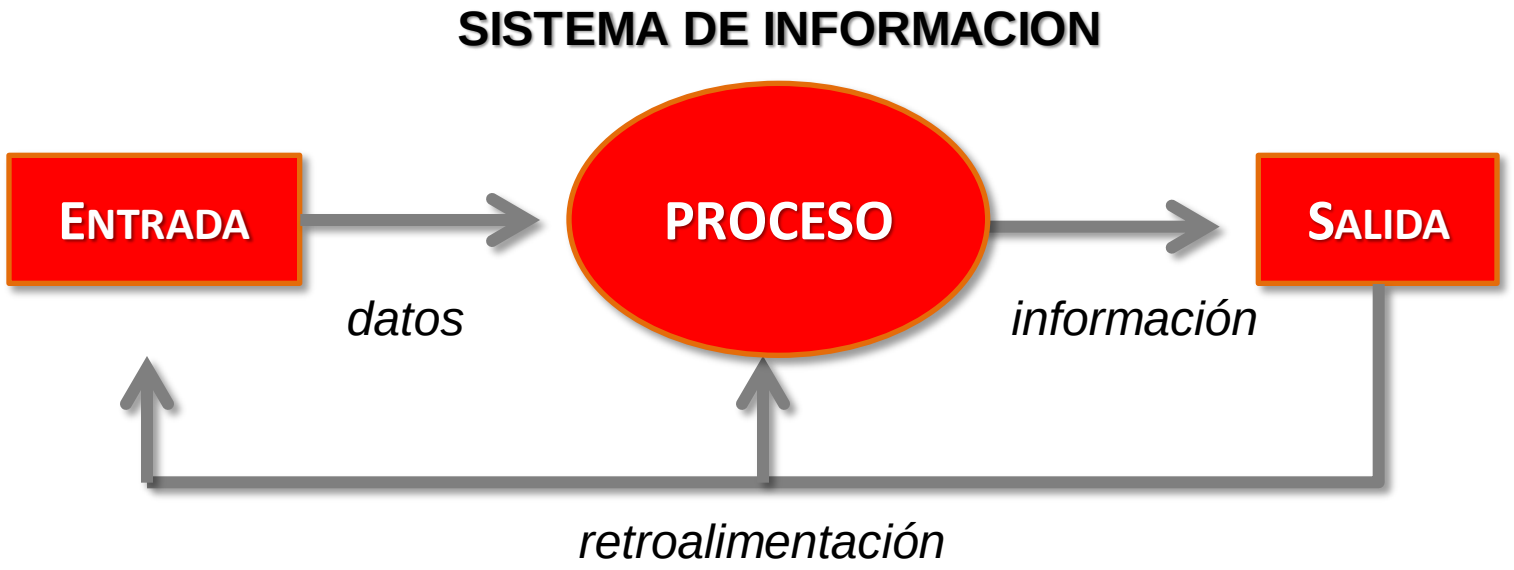
An early batch system. (a) Programmers bring cards to 1401. (b) 1401 reads batch of jobs onto tape. (c) Operator carries input tape to 7094. (d) 7094 does computing. (e) Operator carries output tape to 1401. (f) 1401 prints output.

Sistemas on-line

Características

- La transacción se registra cuando sucede.
- Procesa de a una transacción por vez.
- Interactúa con el usuario.
- Requiere acceso rápido a los datos.
- Se accede en forma aleatoria a una porción de los datos.
- Las transacciones son sencillas.
  - Ejemplo: sistemas de facturación, sistemas de compras vía Web.

Sistemas de Información (SI)



Proceso – tareas relacionadas de manera lógica para producir un resultado.

- Para definir el proceso se requiere de conocimiento.

Dato vs. Información

DATO – representa un hecho aislado.

- Ejemplos:
  - el número de registro de un alumno,
  - el saldo de una cuenta,
  - los artículos de una factura…

INFORMACIÓN – conjunto de datos organizados con un valor adicional más allá de los hechos individuales.

- Ejemplos:
  - ventas del último mes comparadas con el año anterior,
  - artículos más adquiridos por mujeres entre 20 y 30 años…

Características de la información útil

- *Exacta:* libre de errores.
- *Completa:* considera todos los datos relevantes.
- *Flexible:* sirve para una variedad de propósitos.
- *Relevante:* importante para las personas que toman decisiones.
- *Económica:* costo de producción es conveniente.
- *Confiable:* se puede depender de ella.
- *Segura:* no accesible a usuarios no autorizados.
- …

Sistemas de Tiempo Real

Sistemas que controlan un ambiente recibiendo datos, procesándolos y devolviéndolos con suficiente rapidez como para influir en dicho ambiente en ese momento.

Características:

- Interactúan con personas y ambiente.
- Una respuesta fuera de tiempo puede ser catastrófica.
- Requieren de: manejo de interrupciones, asignación de prioridades, control sobre el entorno.
- Ejemplos:
  - Control de procesos.
  - Adquisición de datos de alta velocidad (satélites).
  - Sistemas de monitoreo de pacientes.

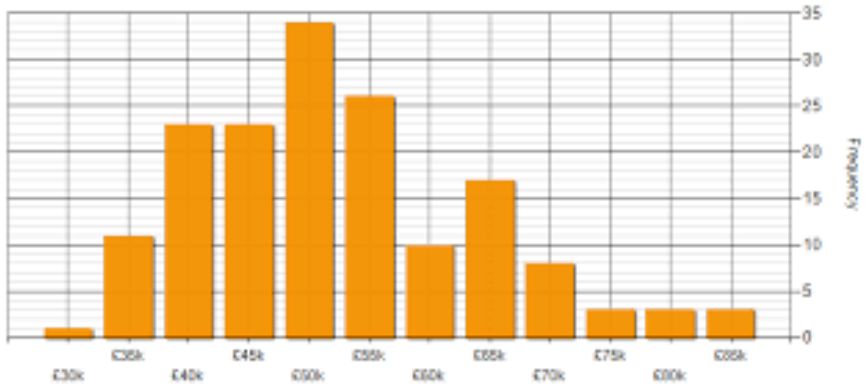
Sistemas de Tiempo Real

Sistemas de soporte de decisión

Características

- No toman decisiones por si solos, sino que colaboran con la toma de decisión.
- No poseen salidas programadas.
- Pueden presentar la información de varias maneras.

- Ejemplos:
  - Datawarehouse.
  - Planillas de cálculo.



Sistemas basados en conocimiento

Características

- Sistemas expertos.
- Imitan el comportamiento de una persona en tareas inteligentes.
- Utilizan técnicas de Inteligencia Artificial.

- Ejemplos:
  - Sistemas de ayuda.
  - Algún tipo de software educativo.

Ingeniería de Software (IS)

Área de las ciencias de la computación que estudia la construcción de sistemas de software de calidad

Ingeniería de Software

- Es un área de las ciencias de la computación que estudia la construcción de sistemas de software tan **grandes** y **complejos** que requieren de un grupo de ingenieros.

Construcción de
múltiples versiones de software
por múltiples personas.
(Parnas - 1968)

Producción de software

La producción de software evolucionó con el tiempo:

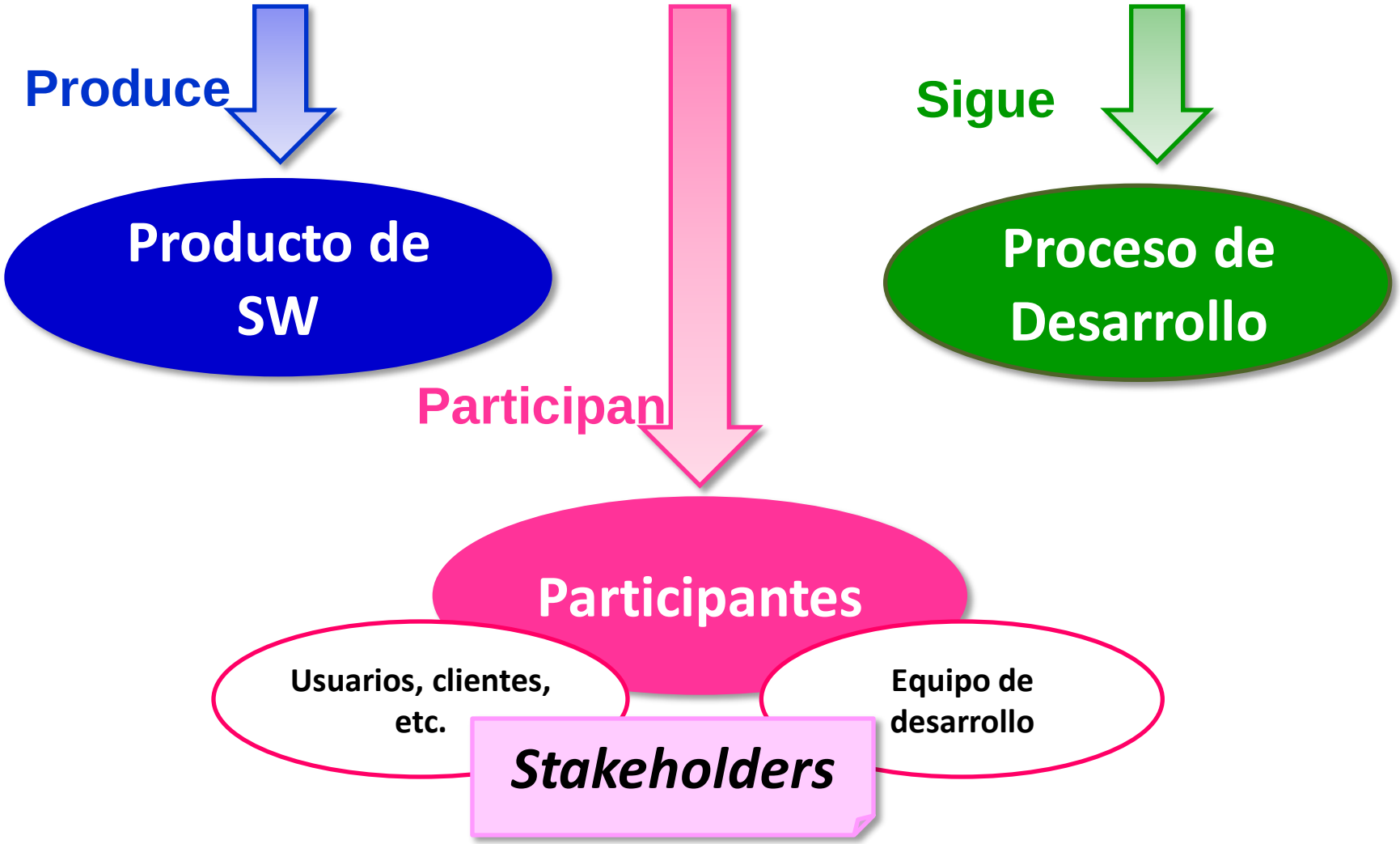
1. Ubicar un conjunto de instrucciones juntas para que la computadora haga algo útil:
   - Problema bien definido.
   - Programación escrita por el propio interesado.
2. Lenguajes de programación de más alto nivel y computadoras más accesibles,
   - Se distinguen los roles programador y usuario.
   - Proyectos de software de mayor escala (sistemas operativos)
3. Ingeniería de Software.
   - Software como parte de un sistema más complejo.
   - Productos de ingeniería.

Ingeniería de Software (IS)

Generalidades

- Es esencialmente una actividad en equipo: un ingeniero de software desarrollará un componente de software que se combinará con otros componentes desarrollados por otros ingenieros.
- Existen versiones del producto.
- El producto perdurará en el tiempo y está *sujeto a cambios.*
- Requiere de un trabajo disciplinado.

Ingeniería de Software



El Producto

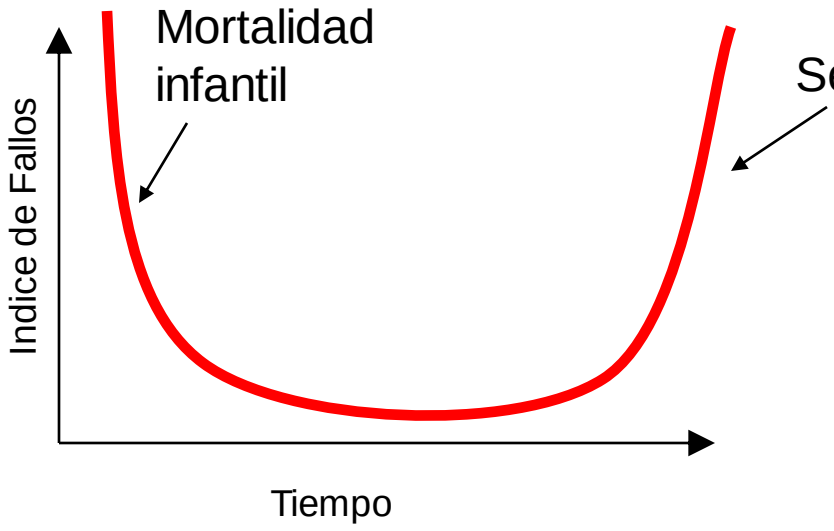
Producto de IS – es el **sistema de software** que se distribuye al cliente junto **con su documentación**.

- La IS apunta a la construcción de software como una actividad de ingeniería: *producir productos de calidad*
- Los productos de software se clasifican:
  - *Software a medida*: software desarrollado para un cliente particular bajo un contrato.
  - *Software genéricos*: desarrollados para ser vendidos a un mercado abierto.

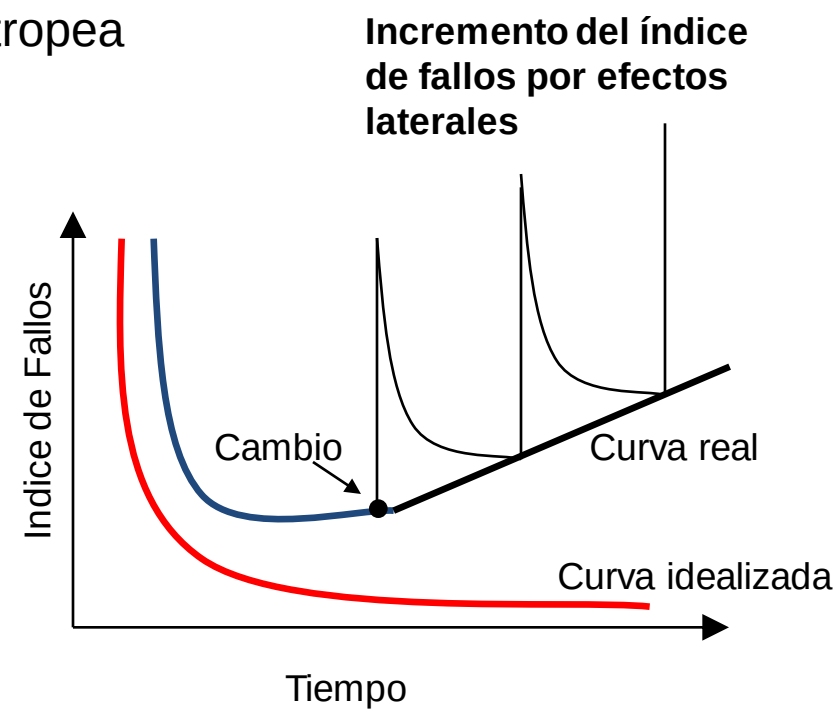
El Producto Software

- A diferencia de otros productos de ingeniería *el software es un producto particular*.
  - Es software *es lógico* y no físico (no es tangible).
  - El software *se desarrolla* no se fabrica.
    - Aunque la industria tiende a ensamblar componentes, aún la mayor parte del software se construye es a medida.
  - El software *no se estropea* (pero se deteriora).

Ejemplo: Curvas de fallos de HW y SW



Curva de fallos de HW



Curva de fallos de SW

El Proceso

Proceso de Desarrollo – define el marco de trabajo para un conjunto de tareas claves en la producción de software.

- Generalmente, en cualquier *proceso de ingeniería de software*, no importa el área de aplicación, tamaño o complejidad del producto, se puede dividir en tres fases genéricas:
  - Fase de **definición** (**qué** se espera del producto).
  - Fase de **desarrollo** (**cómo** se va hacer).
  - Fase de **mantenimiento**.

El rol Ingeniero de Software

- El rol del Ingeniero de Software evolucionó junto con la disciplina de IS
- Actualmente, un ingeniero de software debe tener dominio sobre un amplio espectro de actividades.
  - Tecnología, dirección, planificación, modelado …

habilidades interpersonales

- Las diferentes personas que interactúan con el sistema se pueden clasificar:
  - usuarios,
  - gerentes, auditores,
  - Analistas, diseñadores, programadores
  - …

Desarrollo de software - Stakeholders

PROPIETARIOS DEL SISTEMA

Patrocinan el desarrollo del sistema.

Obligación contractual

$$$, necesidades

Esquipo de Desarrollo

Construyen el sistema.

USUARIOS

Usan el sistema.

Tienen necesidades

Coordinar el desarrollo del proyecto

EL INGENIERO DE SW

Distintas realidades

Relación Equipo de desarrollo/Empresa

- En un proyecto, el desarrollo de sistemas puede estar a cargo de:
  - Un equipo de desarrollo dentro de la empresa.
  - Un equipo de desarrollo de una empresa externa (*desarrollo de terceros*).
  - Pueden hacerse desarrollos para usuarios no conocidos (*software de propósito general*).
- Según la alternativa los modelos y las necesidades de comunicación son diferentes.

Principios de Ingeniería de Software

Principios – enunciados generales y abstractos que describen las propiedades deseables de los *procesos* y *productos* de software.

- Para aplicar los principios se requieren:
  - **Métodos**: guías generales que gobiernan la ejecución de alguna actividad. Son aproximaciones rigurosas, semánticas y disciplinadas.
  - **Técnicas**: guías más técnicas y mecánicas que los métodos.

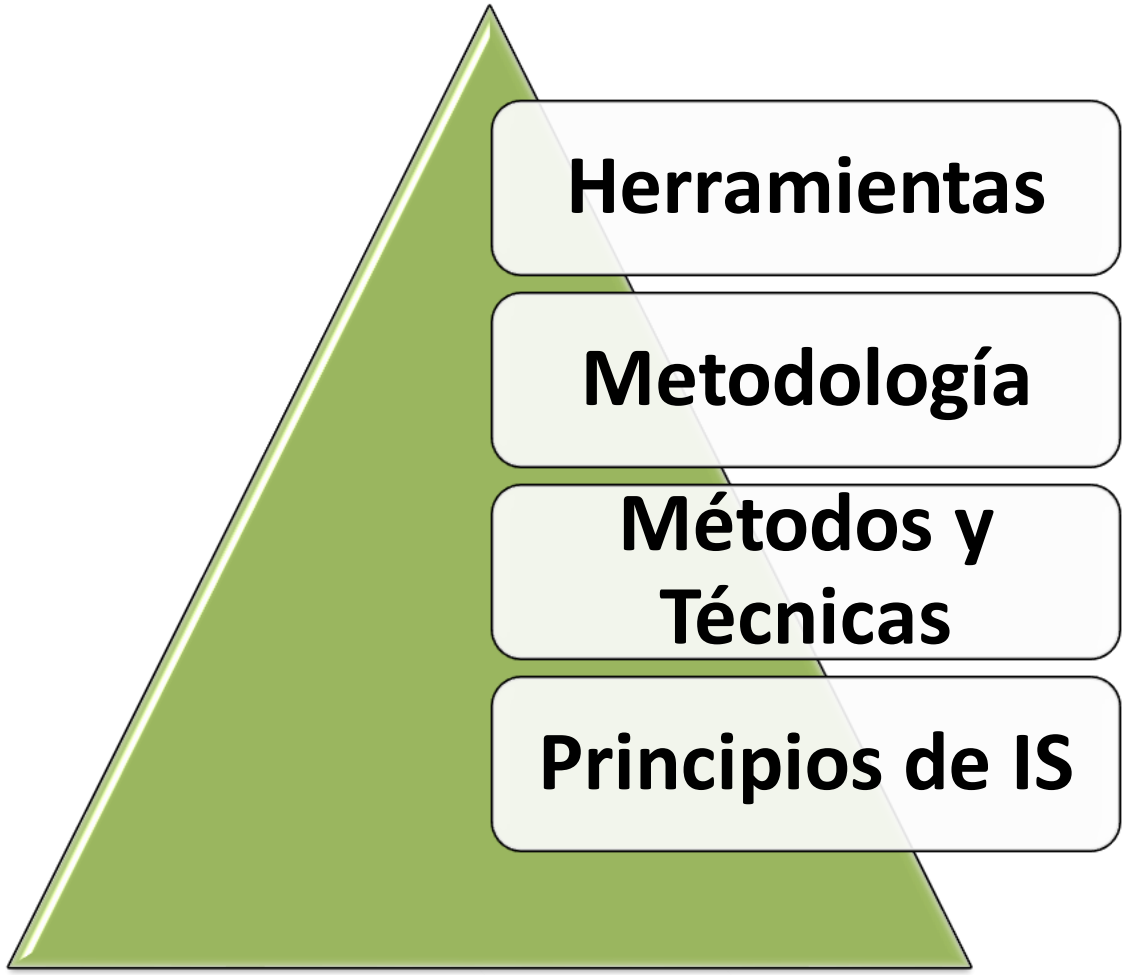
Principios de Ingeniería de Software ...

- Las **metodologías** proveen una aproximación segura para resolver un problema, preseleccionando los métodos y técnicas a ser usadas.

Metodología = métodos + técnicas

- **Herramientas**: son desarrolladas para soportar la aplicación de técnicas, métodos y metodologías.

Ingeniería de Software - Principios



Principios de Ingeniería de SW

1. **Rigurosidad y Formalismo**
   - La rigurosidad NO se define rigurosamente.
   - Formalismo: si el software es evaluado y derivado mediante reglas matemáticas y lógicas.
   - El nivel más alto de rigurosidad es el formalismo.
2. **Separación de Intereses**
   - Distribuir y repartir aspectos diferentes de un problema, para concentrarse en ellos separadamente.
3. **Modularización**
   - Dividir un SW complejo en piezas más simples llamadas módulos.
4. **Abstracción**
   - Proceso que identifica los aspectos importantes de un fenómeno y dejar de lado los detalles.

Principios de Ingeniería de Software

5. **Anticipo al cambio**
   - Distingue al SW de otros productos de ingeniería.
   - Está basado en la propiedad de maleabilidad del producto.
6. **Generalidad**
   - Resolver la “familia de problemas” antes que “el problema particular”.
7. **Incrementabilidad**
   - Caracteriza al proceso de construcción de SW en pasos basados en el anterior.

Calidades de IS para el Producto y Proceso

Las calidades miden objetivamente al producto y al proceso

Calidad

Es una propiedad que debe satisfacer un producto o un proceso o ambos.

- Se busca desarrollar productos de ingeniería de software de alta calidad.
- Existen diferentes enfoque de calidad para un producto de ingeniería.
- La calidad se refiere a características objetivas y mensurables.
- La calidad se relaciona con los requerimientos no funcionales para un sistema.

Calidad del Software

- Las calidades que vamos a estudiar se clasifican en:
  - **Calidad interna:** de interés para los desarrolladores.
    - Sirven para alcanzar las externas.
  - **Calidad externa**: visibles al usuario.

  - **Calidad del producto:** valora el producto que se entrega al cliente.
  - **Calidad del proceso de desarrollo:** valora el procedimiento para producir software.
  - Algunas calidades se aplican al producto, otras al proceso o *a ambos.*

Calidad del proceso y del producto

"La calidad del proceso de afecta a la calidad del producto".

- Esta afirmación es compleja de demostrar en IS. Sin embargo, la experiencia muestra que es cierta.
- Es importante ajustarse algún estándar:
  - **Producto**: de documentación, codificación, etc.
  - **Proceso**: incluyen la definición del proceso.
- Importancia de un estándar:
  - Se basan en la experiencia.
  - Proveen un marco de trabajo.
  - Colaboran con el trabajo en equipo.

Resumen Calidades del Software

Calidad	Definición	Se aplica al Producto/Proceso	Interna/Externa
Correctitud	Se dice que un software es *funcionalmente correcto* si se comporta de acuerdo a la especificación.	Producto	Externa
Confiabilidad	No existe una definición formal para confiabilidad. Se dice que el software es confiable si el usuario puede depender de él.	Producto y al proceso	Externa
Robustez	Un software se dice robusto si su comportamiento ante circunstancias no especificadas es razonable.	Producto y proceso	Externa
Eficiencia (performance)	un sistema de software es eficiente si usa los recursos en forma económica.	Producto	Externa

Resumen Calidades del Software

Calidad	Definición	Se aplica al Producto/Proceso	Interna/Externa
Amigabilidad	Se dice que un software es amigable si los usuarios lo encuentran adecuado para trabajar.	Producto y proceso	Externa
Verificabilidad	Un sistema de software es verificable si se pueden comprobar sus propiedades (por ejemplo tiempo de respuesta).	Producto y al proceso.	Interna
Mantenibilidad	Es la propiedad del software que mide el la capacidad de introducir modificaciones. Se relaciona con reparabilidad y evolutividad.	Producto y proceso	Interna
Reparabilidad	Mide la capacidad de corregir errores del software en tiempo de trabajo limitado.	Producto	Interna

Resumen Calidades del Software

Calidad	Definición	Se aplica al Producto/Proceso	Interna/Externa
Evolutividad	Mide la capacidad de modificar las el software para adaptarlo a nuevos requerimientos	Producto y proceso	Interna
Reusabilidad	Capacidad de reutilizar un software haciendo cambios menores	Producto y al proceso.	Interna
Comprensibilidad	Mide el diseño desde el punto de vista de su comprensión.	Producto	Interna (y externa)
Interoperabilidad	Es la habilidad de un sistema de coexistir y cooperar con otros.	Producto	Externa
Productividad	Mide la eficiencia del proceso.	Proceso	Interna
Puntualidad	Es la habilidad de entregar el producto a tiempo.	Proceso	Interna
Visibilidad	Un proceso es visible si sus etapas están claramente documentados.	Proceso	Interna

Conceptos vistos hasta este punto

Sistemas

- Definición. Ejemplos
- Clasificación
- Principios de Sistemas

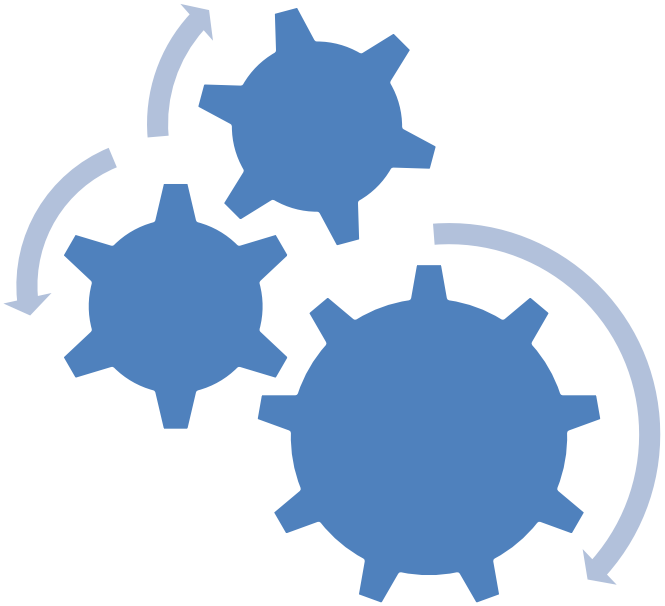
Sistemas Automatizados

- Características
- Aplicaciones.
  - Batch,
  - On-line,
  - De tiempo real,
  - De soporte de decisión,
  - Basadas en conocimiento.
  - Características

Ingeniería de Software

- Definición
- Componentes: producto, proceso, participantes, principios
- Principios de IS
  - Rigurosidad y Formalismo
  - Separación de Intereses
  - Modularización
  - Abstracción
  - Generalidad
  - Anticipo al cambio
  - Incrementabilidad
- Calidades
  - Calidades internas y externas
  - Calidades del producto y del proceso

El proceso de desarrollo de software



El objetivo de un proceso de producción es garantizar productos confiables, predecibles y eficientes.

El Proceso de Desarrollo

- El **proceso** define un *marco de trabajo* para un conjunto de tareas claves en la producción de software.

Ingeniería de Software [IEEE93]

1. La aplicación de un *enfoque sistemático, disciplinado y cuantificable* al desarrollo, operación y mantenimiento del software, esto es, **aplicación de ingeniería al software**.
2. Estudio de enfoques que permitan alcanzar 1.

El proceso

- El **proceso** forma la base para *el control de la gestión de los proyectos de software*.

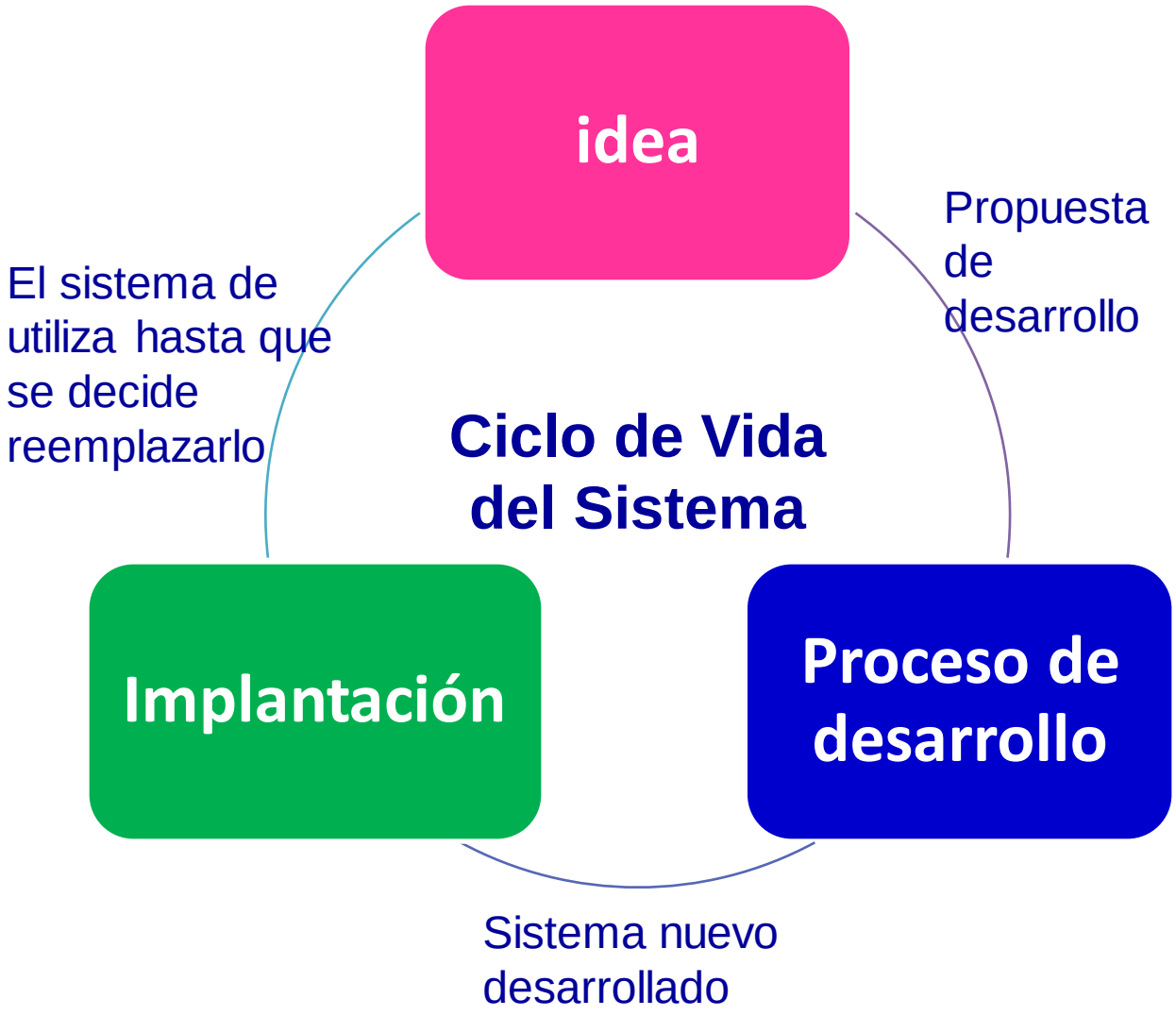
- Establece el contexto en el cual se aplican:
  - los métodos, técnicas y metodologías,
  - se generan los productos de trabajo (modelos, documentos, datos, código, etc.)
  - se asegura calidad,
  - el cambio se puede manejar.

Ciclo de Vida

Ciclo de vida de un sistema – comprende desde la concepción de la idea de un desarrollo del software, hasta que es implementado, entregado, y aún después hasta que deje de usarse.

- Cada sistema tiene un *ciclo de vida* compuesto por varias etapas.
- Existen varios patrones o modelos de procesos, el ciclo de vida de un sistema puede estar conducido por alguno de ellos.

Ciclo de Vida del Sistema



ISO/IEC 12207
Information Technology / Software Life Cycle Processes

”Un marco de referencia que contiene los procesos, las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto de software, abarcando la vida del sistema desde la definición de los requisitos hasta la finalización de su uso”

El primer modelo de proceso de desarrollo

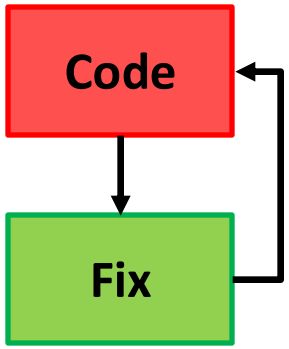
Code-Fix

- Fue la primera aproximación de desarrollo.
- No es un modelo de proceso.

Características

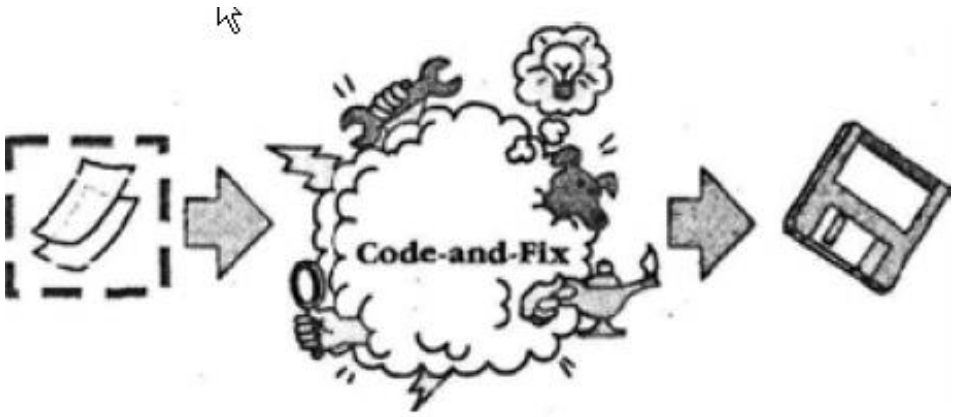
- El desarrollo de software era una tarea unipersonal.
- El problema era sencillo y bien entendido.
- Los lenguajes de programación bajo nivel.

Code & Fix



- *Actividades*
  - Escribir código
  - Corregirlo (mejorar o agregar funcionalidades).

- *Observaciones*
  - Después de algunas evoluciones se torna no confiable.
  - Posible en aplicaciones cortas y bien entendidas.



Hitos

Cambios importantes:

- Se separan los roles *programador* y *usuario*.
  - El producto no satisface las necesidades del usuario.

- Necesidad de *aplicaciones más complejas*.
  - Se requiere mayor calidad.

- El *desarrollo* se convierte en una *actividad grupal.*
  - Se necesitan modelos de comunicación.

Actividades

Una organización tradicional para las actividades de un proceso de desarrollo (no excluyentes):

1. Estudio de factibilidad
2. Elicitación, análisis y especificación de requerimientos - *¿qué?*
3. Diseño - *¿cómo?*
   - de arquitectura
   - de programas.
4. Codificación y testeo de módulos.
5. Integración y testeo de sistemas.
6. Distribución, entrega y mantenimiento.
7. Otras actividades.

Ingeniería de Software – C. Guezzy. Ed. 2002 Distintos autores usan diferentes actividades

1- Estudio de Factibilidad

- En general para cualquier desarrollo la primer actividad es el *estudio de factibilidad*.
  - Se *evalúan costos y beneficios* del desarrollo.
  - Cuanto más conocimiento se tenga del problema mejor. Sin embargo, *no se le destina demasiado tiempo*.
  - Incluye:
    - Una definición general del problema.
    - Soluciones alternativas y beneficios esperados.
    - Recursos requeridos, costos, tiempos, fechas de entrega para cada solución alternativa.
- Según el marco de trabajo este estudio debe ser más o menos formal.

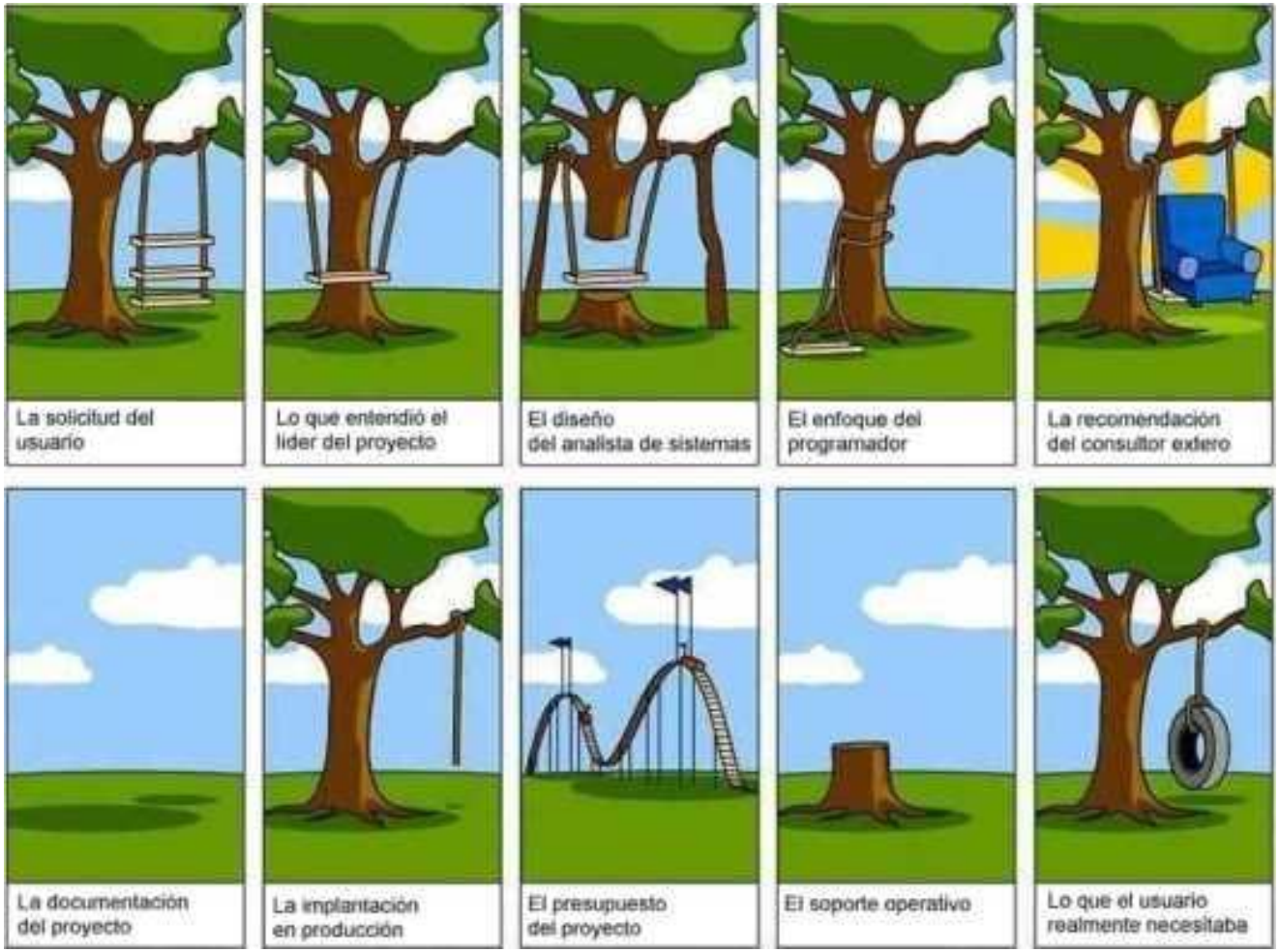
2 – Elicitación, análisis y especificación de requerimientos

- Identificar las calidades (necesidades) requeridas por la aplicación: funcionales y no funcionales.
- Identificar ***qué?*** y no *cómo?*. La clave:

Separar, abstraer y particionar

- Descripción en distintos niveles de abstracción.
  - Separar en partes analizables.
  - Separar en distintas visiones del software:
    - Datos - funciones - restricciones - control (comportamiento)

Requerimientos del Sistema



3 – Definición de arquitectura y diseño detallado del sistema

- Diseño de arquitectura
  - Es de alto nivel.
  - Partición de funciones entre clientes y servidor.
- Diseño de módulos
  - Descomposición modular.
  - Definición de componentes, interfaces e interconexiones.
- Documentar!

4 – Codificación y Testeo de Módulos

- Se escriben programas en un lenguaje de programación.
- Una buena práctica es *definir y adoptar convenciones* de programación y testeo.
- Puede incluir revisiones de código (debbuging y testeo y chequeos de performance).
- Resultado: módulos implementados y testeados.

5- Integración y Testeo de Sistema

- Se *ensamblan los componentes* desarrollados y testeados separadamente.
- En la etapa final se puede incluir testeos con:
  - Datos reales.
  - Clientes reales (alpha test).
- Aplicar estándares (*top down* o *buttom-up*).
- *Documentar* los casos de prueba.

6 – Distribución, entrega y mantenimiento de sistema

- Cuando se completa el desarrollo de la aplicación.
- Primero se entrega a un grupo reducido de clientes antes de la entrega oficial (beta testing).
  - Objetivos: experimentación y retroalimentación.
- Pueden hacerse cambios antes de la entrega oficial.
- El mantenimiento incluye las actividades realizadas una vez que el producto es entregado al cliente.

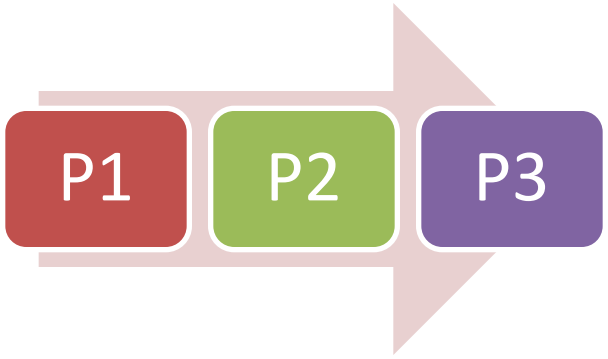
Mantenimiento

- El costo de mantenimiento en muchos casos representa más del 60% del costo del producto.
  - Corregir errores encontrados después de la entrega (*mantenimiento correctivo*)
  - Adaptar la aplicación a cambios de entorno (*mantenimiento adaptativo)*
  - Cambiar o agregar funciones o calidades del sistema *(mantenimiento perfectivo)*
- El 50% de los costos de mantenimiento se atribuyen a cambios en los requerimientos.

7 – Otras actividades

- Dependiendo del tipo de desarrollo puede ser necesario incorporar nuevas actividades:
  - Documentación.
  - Control de calidad
    - Walk-through.
    - Validación.
    - Verificación.
  - Administración
    - Definición de estándares.
    - Tratamiento con recursos (personas).

Modelos de Procesos Genéricos



Un modelo de proceso provee una representación abstracta del proceso de desarrollo de software

Modelos de Procesos

- Vamos a estudiar algunos *modelos de procesos* clásicos para la construcción de software.
- Los modelos de procesos *evolucionan* en la medida que evoluciona la *tecnología y los requerimientos a satisfacer*.
- Existen cientos de modelos, la mayoría son variaciones de otros
  - Cascada.
  - Evolutivos.
  - Transformacionales.

TIP! Cada proyecto particular define su propio ciclo de vida

Modelo de CASCADA (MC)

- Las tareas o actividades de desarrollo se organizan en **cascada**, una después de la otra.
- La salida (*output*) de una etapa es la entrada (*input*) de la siguiente etapa.
  - Las etapas se pueden dividir en actividades ejecutadas concurrentemente.

Origen

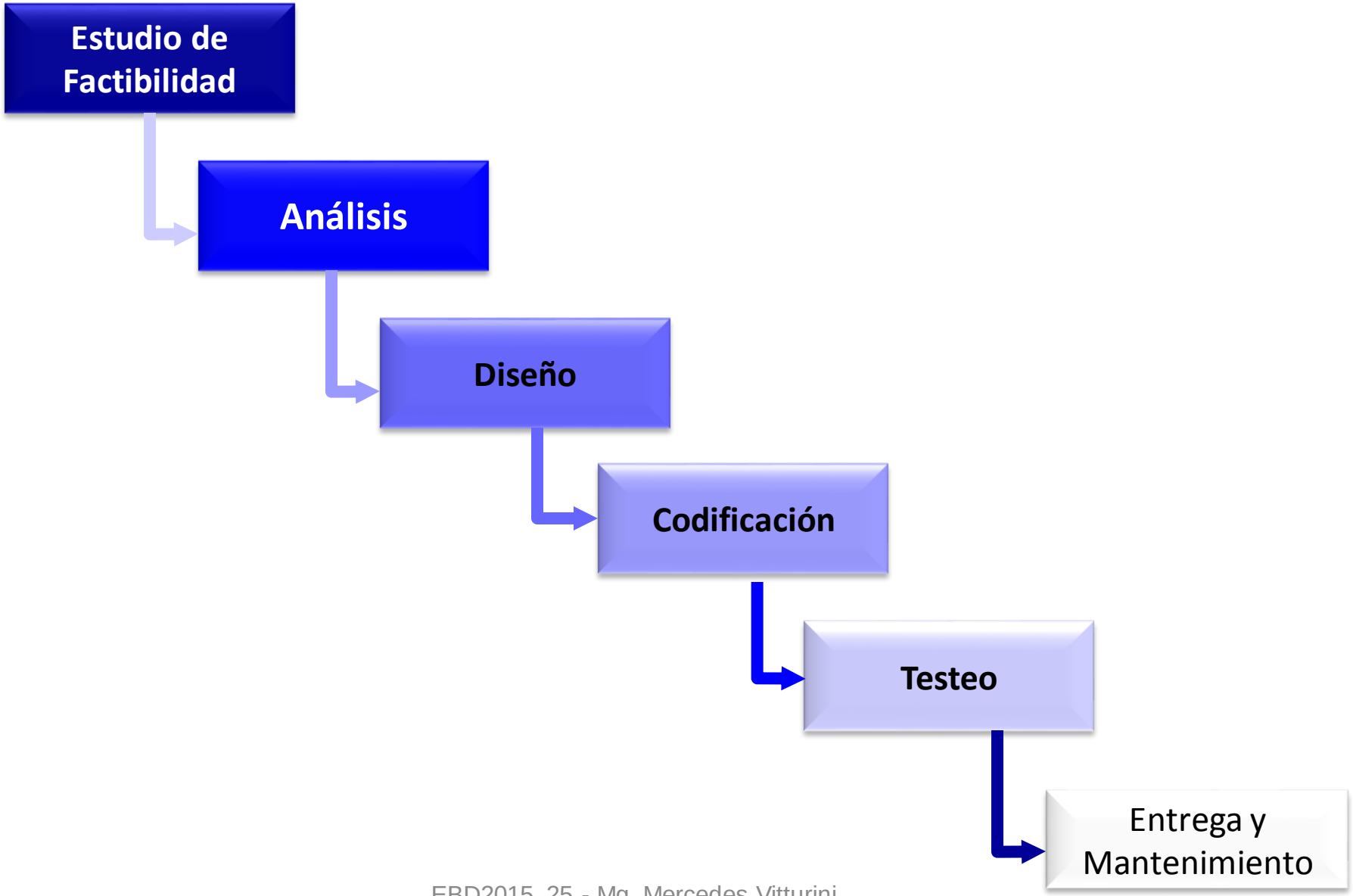
- Creado en los años ’60 en un proyecto de defensa de EEUU.
- Se usó para las primeras aplicaciones es crítica y compleja.

Modelo Cascada

"The simplest process model is the *waterfall model*, which states that the phases are organized in a linear order. The model was originally proposed by Royce, though variations of the model have evolved depending on the nature of activities and the flow of control between them. In this model, a project begins with feasibility analysis. Upon successfully demonstrating the feasibility of a project, the requirements analysis and project planning begins. The design starts after the requirements analysis is complete, and coding begins after the design is complete. Once the programming is completed, the code is integrated and testing is done. Upon successful completion of testing, the system is installed. After this, the regular operation and maintenance of the system takes place"

A Concise Introduction to Software Engineering - Pankaj Jalote

Modelo CASCADA



Modelo Cascada

Características:

- Es un *modelo secuencial*. Cada fase debe estar terminada antes de pasar a la siguiente.
- La definición de las actividades depende del tipo de proyecto y la relación equipo de desarrollo cliente.
- Cada actividad *produce un documento de salida* que es el ingreso para la siguiente actividad.
- Las empresas que usan este modelo de proceso deben definir

MC – Conclusiones

- Fue el primer modelo.
- Es un modelo *conducido por la documentación.*

Contribuciones:

- Introduce *disciplina* al proceso.
- *Pospone la implementación* hasta que los objetivos estén claros.
- Es el paradigma más antiguo y más extensamente usado en ingeniería de software.
- *Impone puntos de control* claros.

MC – Conclusiones

Desventajas*:*

- Demasiado rígido.
- Retrasa la detección de problemas críticos.
- Difícil la estimación de recursos.
- Es lineal y los proyectos reales raras veces siguen un modelo lineal.
- El usuario interviene al principio y al final del proyecto.
- No provee anticipación al cambio.
- Basado en documentación. Parece burocrático.
- Los costos se trasladan al mantenimiento fácilmente.
- No permite implementaciones parciales.

El paradigma evolutivo

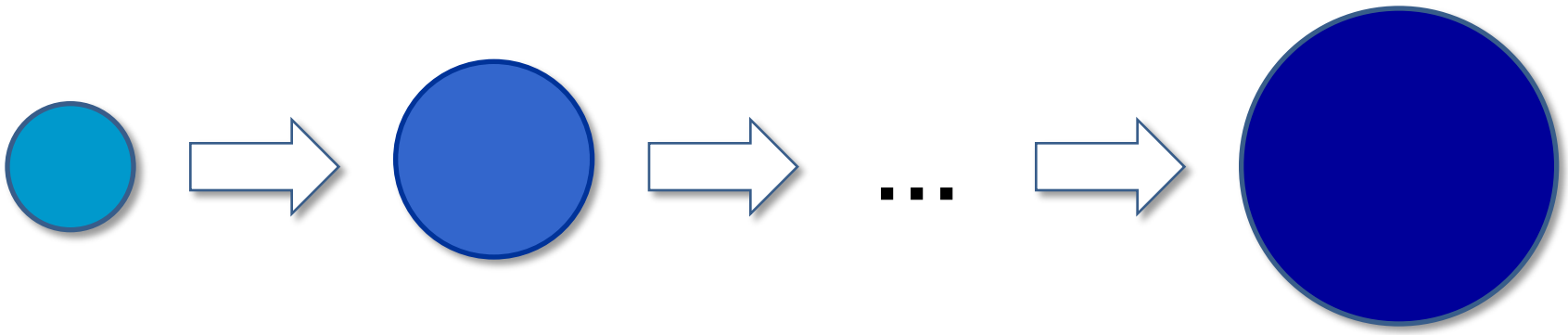
Motivación – las fallas de la primer versión de una aplicación conducen a la necesidad de rehacerla.

HACERLO DOS VECES.

- *Problema – gap* entre la definición de requerimientos y la distribución de la aplicación.
- *Estrategia –*
  - Entregar algo y medir el valor agregado.
  - Ajustar diseño y objetivos.
  - Iterar: desarrollo de versiones cada vez más completas del software.

Modelos Evolutivos

- Proceso evolutivos:
  - Prototipo
  - Iterativo incremental.
  - Espiral



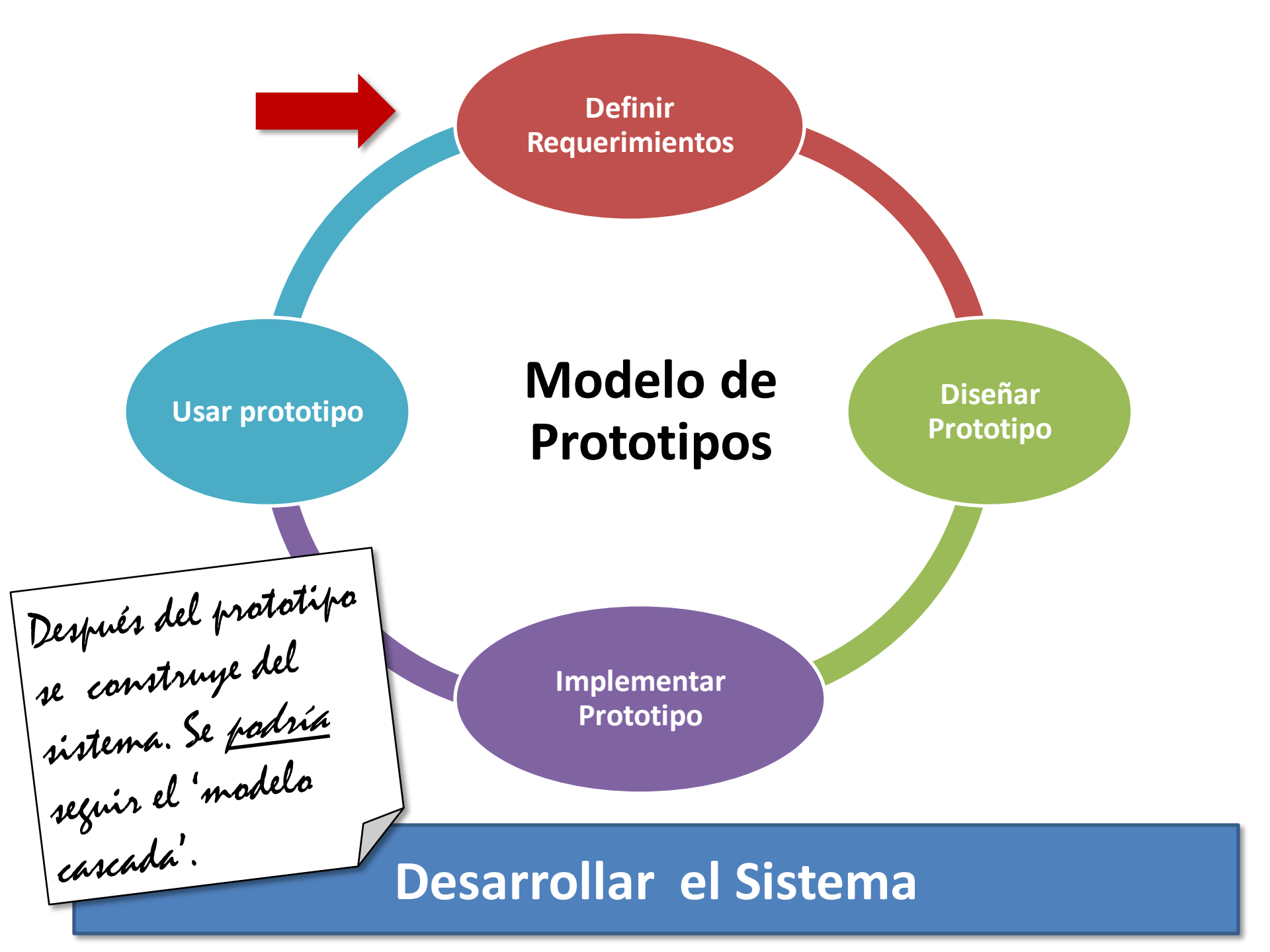
Modelo de PROTOTIPO (MP)

- Primer desarrollo: elaborar *un prototipo para descartar*. El prototipo sirve para validar requerimientos y resolver aspectos críticos del diseño.

Objetivos

  - Investigación repetida de requerimientos y diseño.
  - Reducir el riesgo que la aplicación no se ajuste a las necesidades del cliente.

- La revisión del prototipo puede resultar en revisar los requerimientos anteriores.


Definir Requerimientos
Diseñar Prototipo
Implementar Prototipo
Usar prototipo
Modelo de Prototipos
Después del prototipo se construye del sistema. Se podría seguir el 'modelo cascada'.
Desarrollar el Sistema

Modelo de Prototipos

- **Actividades**
  - *Recolección requerimientos del sistema*: desarrollador y cliente definen los objetivos globales.
  - Se *elabora un diseño rápido*, centrado en los aspectos del software que son visibles a los usuarios (enfoques de entrada y formatos de salida).

Desventajas

- El cliente ve *el prototipo y lo confunde con el sistema real*.
- Se toman *decisiones rápidas* para poder construir el prototipo, que *son difíciles de revertir* (por ejemplo el lenguaje de programación).
- El prototipo para descartar *nunca se descarta.*

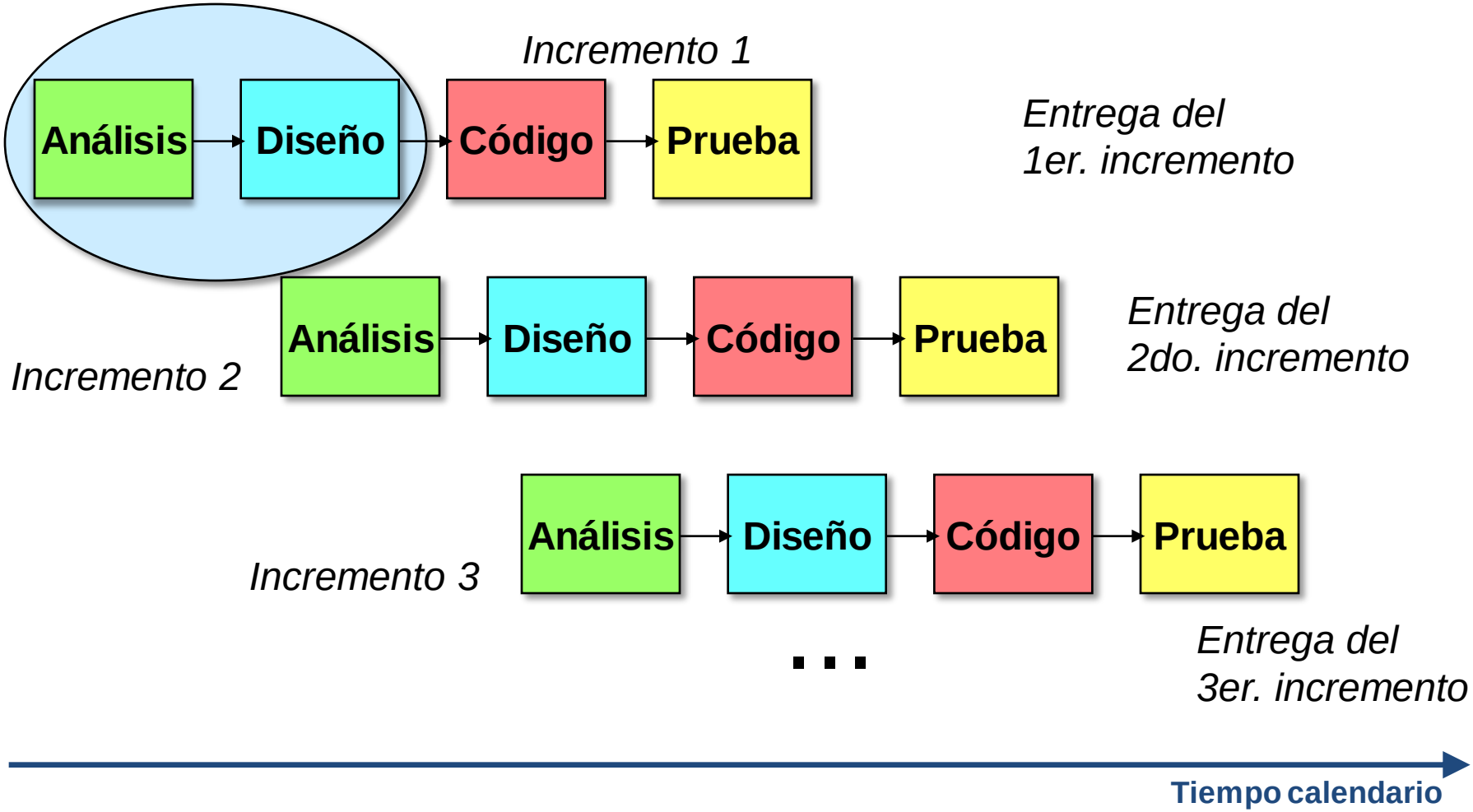
Modelo Iterativo Incremental

Iterativo – hacemos varias veces lo mismo.

Incremental – el producto se "incrementa" a medida que se avanza. También se los llamados "evolutivos".

- Consisten en de *incrementos expandidos de un producto de software operativo*, conducidos por la evolución determinada según la experiencia operativa.
- Los incrementos se distribuyen a medida que se completan.
- *Incremento integrado:* es una unidad de software auto-contenida que realiza algún propósito útil.

Modelo Iterativo Incremental



Ejemplo

- Un ejemplo de desarrollo siguiendo el modelo iterativo incremental para un procesador de textos:
  - **Incremento #1**: administración de archivos básicos y producción de documentos.
  - **Incremento #2**: funciones de edición más sofisticadas (gráficos, tablas, etc.)
  - **Incremento #3**: corrección gramatical y ortográfica.

- Al primer incremento se lo denomina *producto esencial*.

Paradigma Iterativo Incremetal

Ventajas

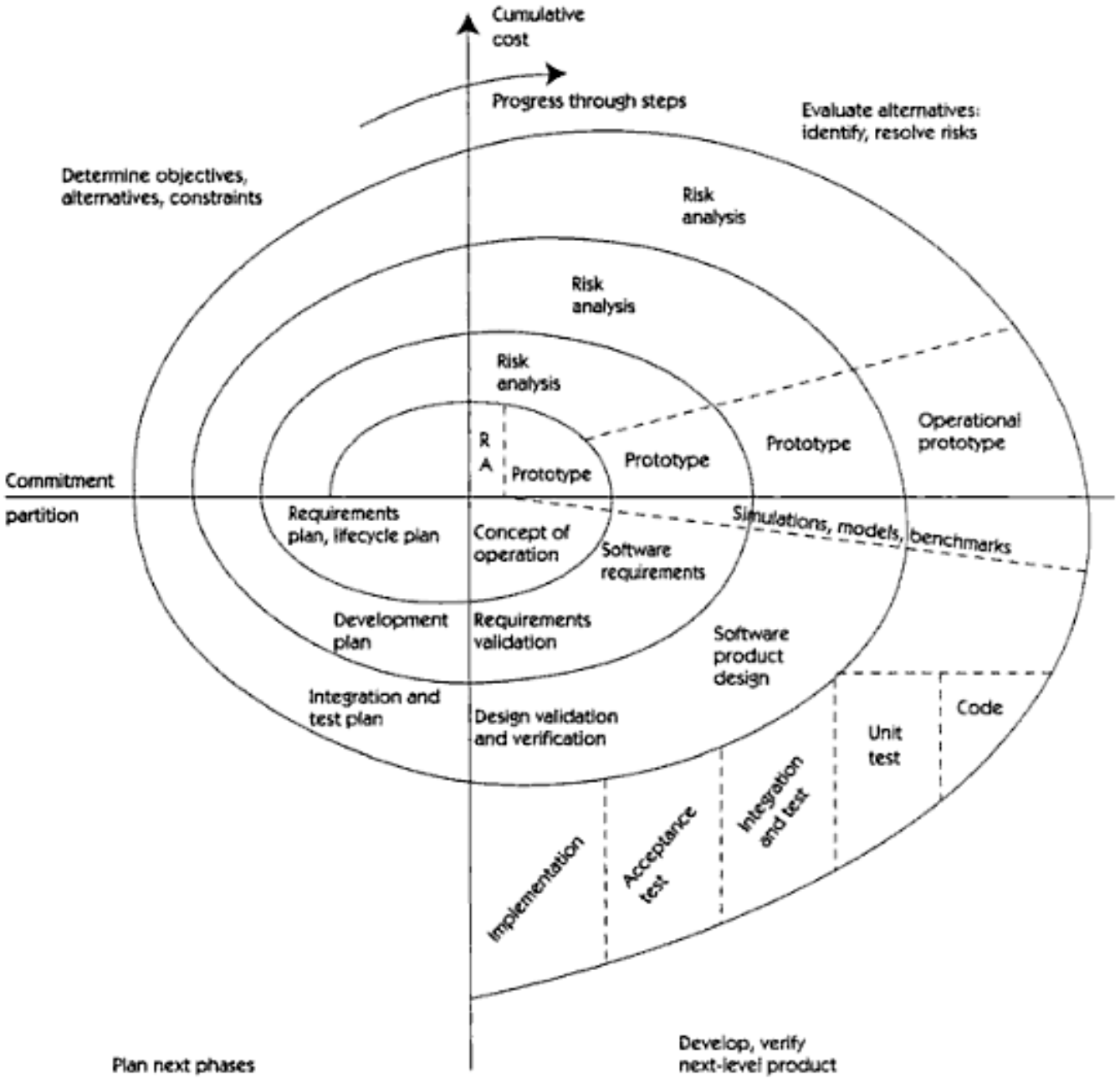
- El usuario ve algo rápidamente.
- Se admite que lo que se está construyendo es el sistema, y por lo tanto se piensa en su calidad desde el principio.
- Los ciclos van mejorando con las experiencias de los anteriores.
- Los trabajadores del equipo de desarrollo se especializan en ciertas actividades.

Desventajas

- Es difícil mantener el foco. Los incrementos entran en mantenimiento mientras se está desarrollando nuevos.
- Es difícil seguir procesos puramente iterativos incrementales.

Modelo Espiral (MEs)

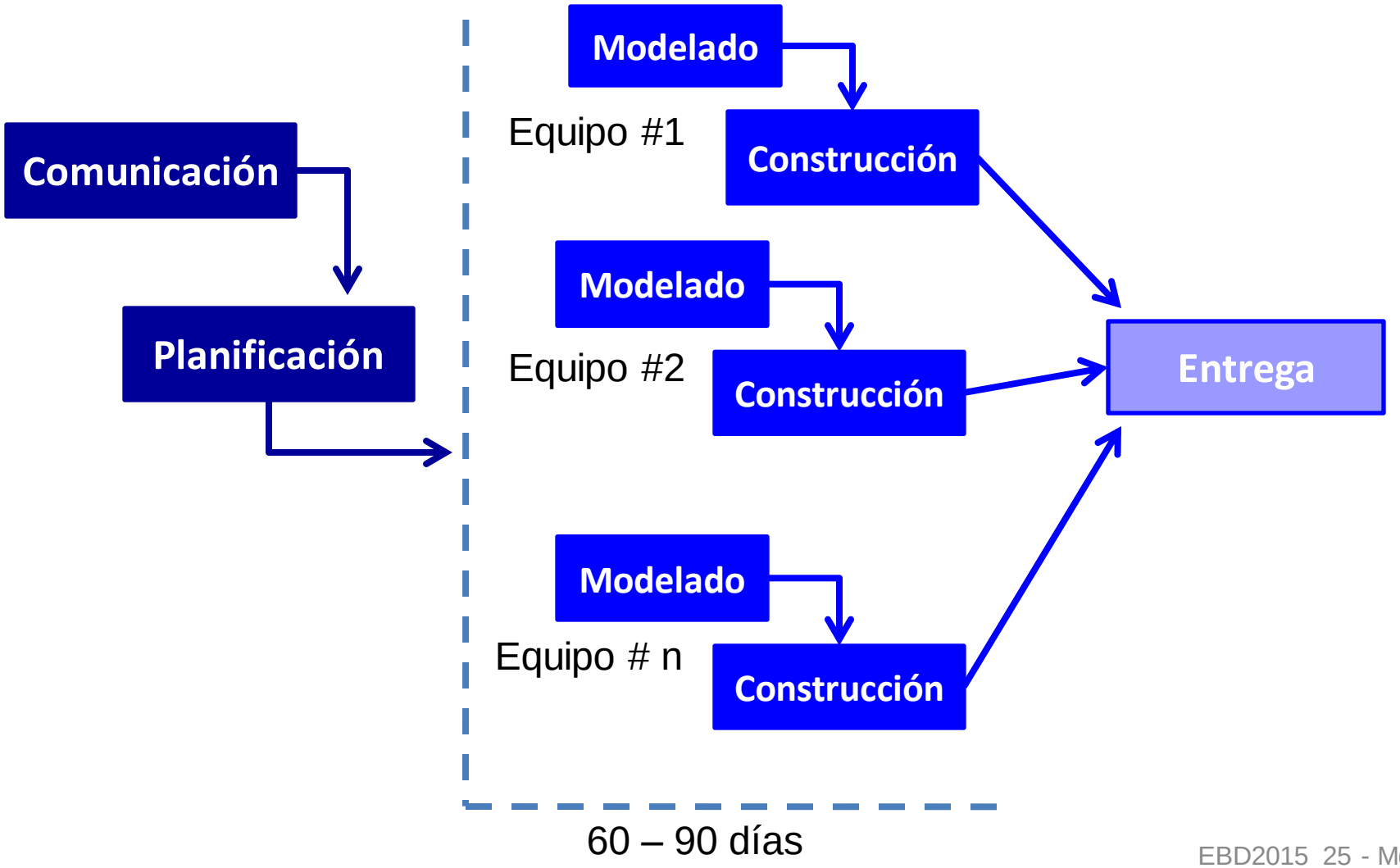
- El modelo *combina las actividades* del **desarrollo** con la **gestión del riesgo**, para minimizar y controlar el riesgo.
  - **Riesgo**: circunstancias potencialmente adversas que pueden perjudicar el proceso de desarrollo y la calidad de los productos.
  - **Administración del riesgo**: disciplina cuyos objetivos son manejar y eliminar los ítems de riesgo del software antes que se transformen en amenaza.
- Es cíclico.
- Es un meta-modelo.
- Estrategia: comenzar por los desarrollos de mayor riesgo.


Cumulative cost
Progress through steps
Evaluate alternatives: identify, resolve risks
Determine objectives, alternatives, constraints
Risk analysis
Risk analysis
Risk analysis
R A
Prototype
Prototype
Prototype
Operational prototype
Commitment partition
Requirements plan, lifecycle plan
Concept of operation
Software requirements
Simulations, models, benchmarks
Development plan
Requirements validation
Software product design
Integration and test plan
Design validation and verification
Code
Unit test
Integration and test
Acceptance test
Implementation
Plan next phases
Develop, verify next-level product

Desarrollo Rápido de Aplicaciones (DRA)

- Es un modelo de proceso que apunta a proyectos rápidos que finalizan en 60-90 días.
- Utiliza un enfoque de construcción basado en componentes.
- Requiere muchos recursos humanos como para crear varios equipos DRA.
- Desarrolladores y clientes deben estar comprometidos en las actividades necesarias para completar un sistema en un marco de tiempo abreviado.

Desarrollo Rápido de Aplicaciones



DRA

Desventajas

- Tiene requerimientos fuertes de personal de desarrollo.
- No es un modelo adecuado para sistemas que requieren de muchas actividades de mantenimiento.
- No es adecuado cuando los riesgos técnicos son altos.
- No funciona si no existe un compromiso real del cliente.

Generalidades

¿Cuál es la importancia de los modelos de proceso?

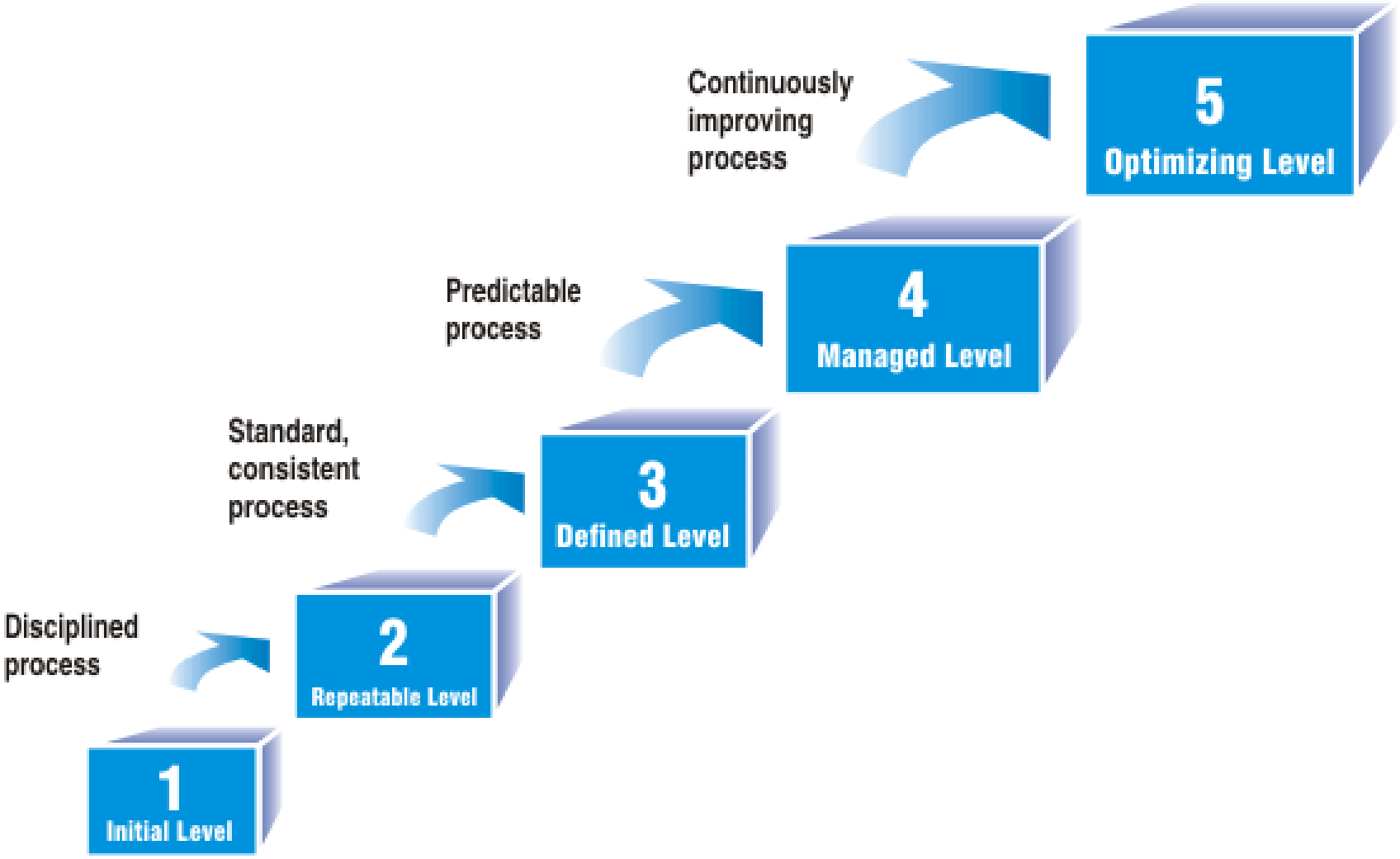
- Organizan la tarea de desarrollo de software.
- Dan el marco para la planificación y control del desarrollo.
- Permiten asegurar calidad en el producto.
- Dan *visibilidad* al proceso de desarrollo

Madurez en Desarrollo de SW

CMMI– Modelo integrado de Capacidad y Madurez

- El desarrollo de sistemas en una organización madura *disminuye costos y tiempo de producción* y aumenta la productividad y la calidad del producto.
- El Modelo de Integrado de Capacidad y Madurez (CMMI), Desarrollados por Software Engineering Institute (SEI) provee un estándar de referencia para evaluar el nivel de madurez del desarrollo de sistema de una organización.

Nivel de Madurez



Nivel de Madurez

Nivel	Características	Resultados
Inicial	- Ausencia de gestión de proyectos. - El proceso de software es cambiante e irregular: - Los planes, estimaciones y calidad son impredecibles. - El rendimiento depende de la capacidad individual de los miembros del grupo. - Se establecen programas de formación del personal de desarrollo y mantenimiento.	Productividad y calidad escasa. Riesgo máximo
Repetible	- Los procesos de software son estables y repetibles. - La organización establece políticas de gerencia de proyectos y procesos. - La planificación se basa en proyectos similares. - Existen estándares definidos y exigidos. - El proceso se enmarca en un sistema de gerencia de proyectos basado en experiencias pasadas.	Productividad y calidad baja. Riesgo alto.

Nivel de Madurez

Nivel	Características	Resultados
Definido	-Los procesos son definidos: estandarizados, documentados e institucionalizados. - Los procesos de ingeniería y gerencia son estables y se integran en uno sólo. - Existe un entendimiento común de los procesos, funciones y responsabilidades. - La organización mantiene un grupo dedicado a la definición, mejoramiento y difusión del proceso de Ingeniería de Software.	Productividad y calidad media. Riesgo medio.
Gestionado	- Los procesos son medibles o cuantificables - La productividad y la calidad se miden y registran para cada proyecto de la organización. - Se fijan metas cuantitativas de la calidad del software. -Mediante el uso de métricas de software, se crea una base cuantitativa para la evaluación y estimación en proyectos futuros.	Productividad y calidad alta. Riesgo mínimo.
Optimizado	- Los procesos se mejoran continuamente. - La organización busca lograr el nivel máximo de capacidad. - Se incorporan nuevas tecnologías y métodos para mejorar los procesos.	Productividad y calidad total. Riesgo nulo.

Temas de Esta Clase

- Sistemas. Clasificación. Propiedades
- Ingeniería de Software. Definición. El ingeniero de software.
- Principios, producto y el proceso.
- Calidades
- Ciclo de vida
- El modelos de proceso genéricos.
  - Cascada.
  - Evolutivos: Prototipo, incremental, espiral.
  - Desarrollo rápido de aplicaciones.
- El Modelo integrado de Capacidad y Madurez

Temas de esta Clase

Bibliografía

- *Fundamentals of Software Engineering* – C. Ghezzy. Capítulos 1,2, 3 y 7.
- *A Concise Introduction to Software Engineering* – Pankaj Jalote. Capítulos 1 y 2
- *Ingeniería de Software* – Ian Sommerville. Capítulo 2.

Temas de la Clase de Hoy

- Sistemas. Clasificación. Propiedades
- Ingeniería de Software. Definición. El ingeniero de software.
- Principios, producto y el proceso.
- Calidades

Bibliografía

- "Fundamentals of Software Engineering" - Carlo Ghezzi. Capítulo 1, 2 y 3.
- "Ingeniería de Software" – Ian Sommerville. Capítulo 2.
- "A Concise Introduction to Software Engineering" Pankaj Jalote – Capítulo 1