Ingeniería de Software

Unidad Nº 1
Introducción a la Ingeniería de Software

Juan José Vanzetti

Software – Definiciones

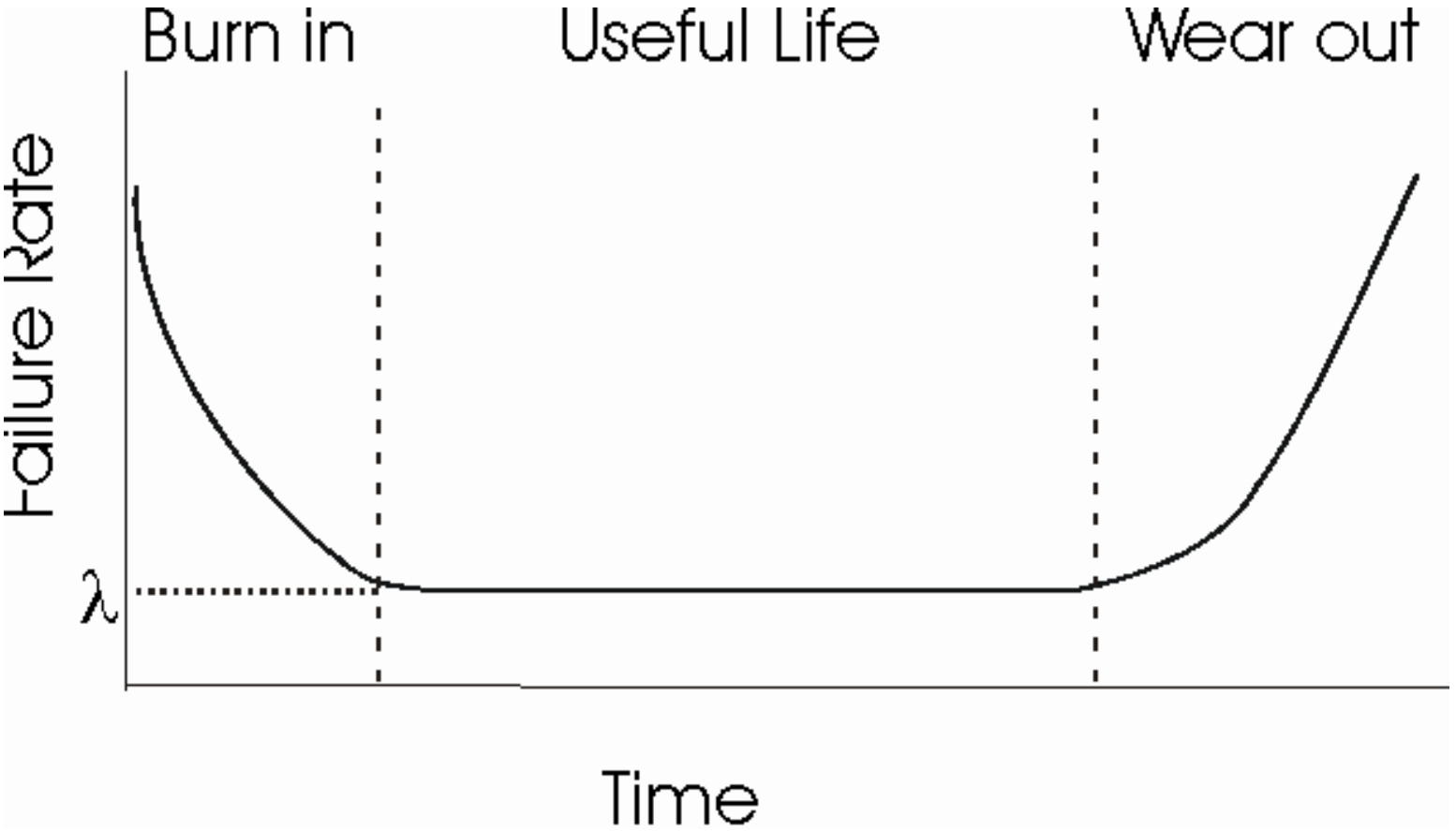
- ¿Qué es el Software?
  - "*Conjunto de Programas que controlan la operación de una computadora*" Longman's Dictionary
  - "*un conjunto de instrucciones (programas de computadoras) que cuando se ejecutan proporcionan la función y el rendimiento deseado*," … *"estructuras de datos que permiten a los programas manipular adecuadamente la información y documentos que describen la operación y el uso de los programas"* Pressmam
  - *"...., la corporización de las funciones de un sistema, el conocimiento capturado acerca de un área de aplicación, la colección de los programas, ..., y la información producida durante el desarrollo de software"* Freeman

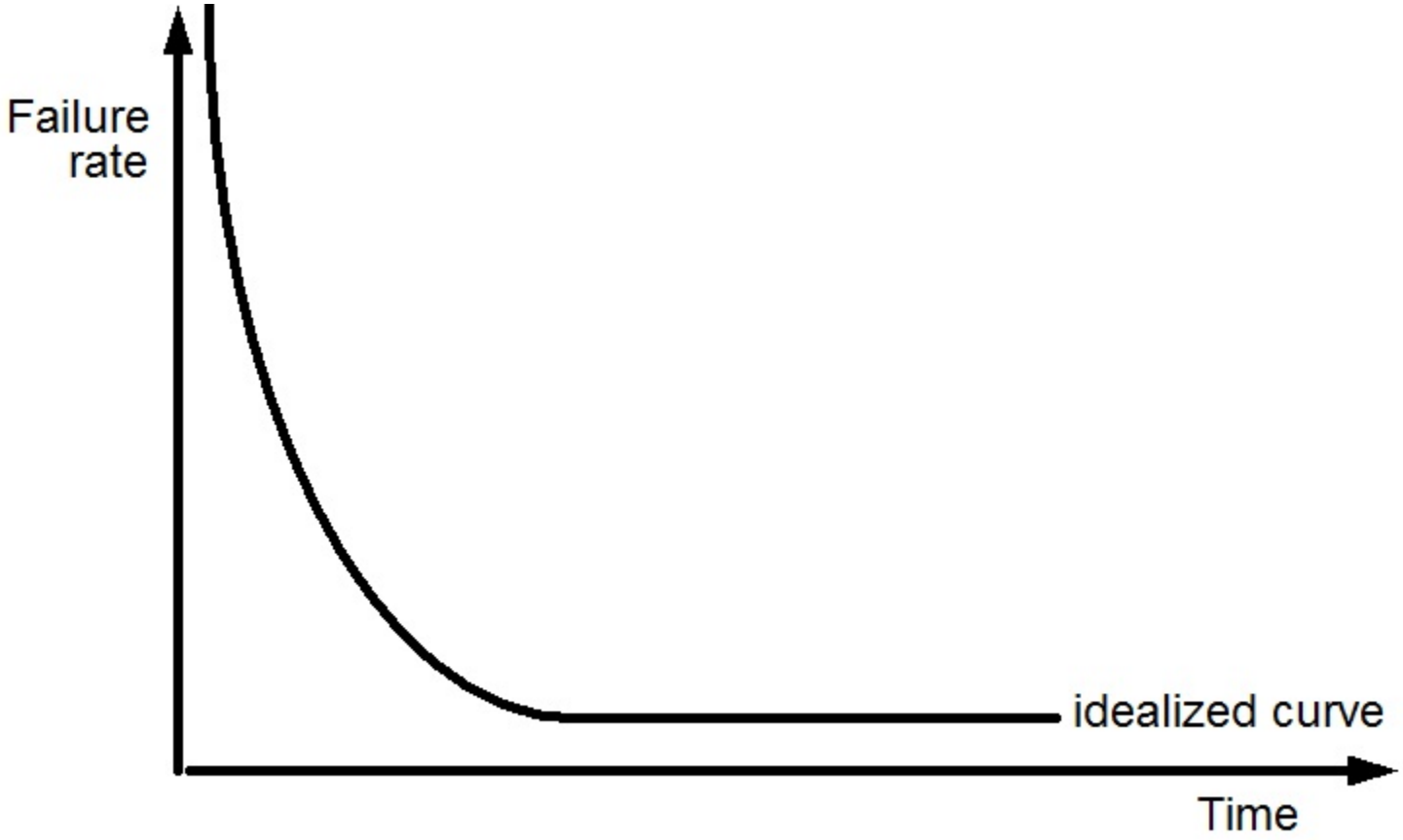
Software - ¿Qué es ?

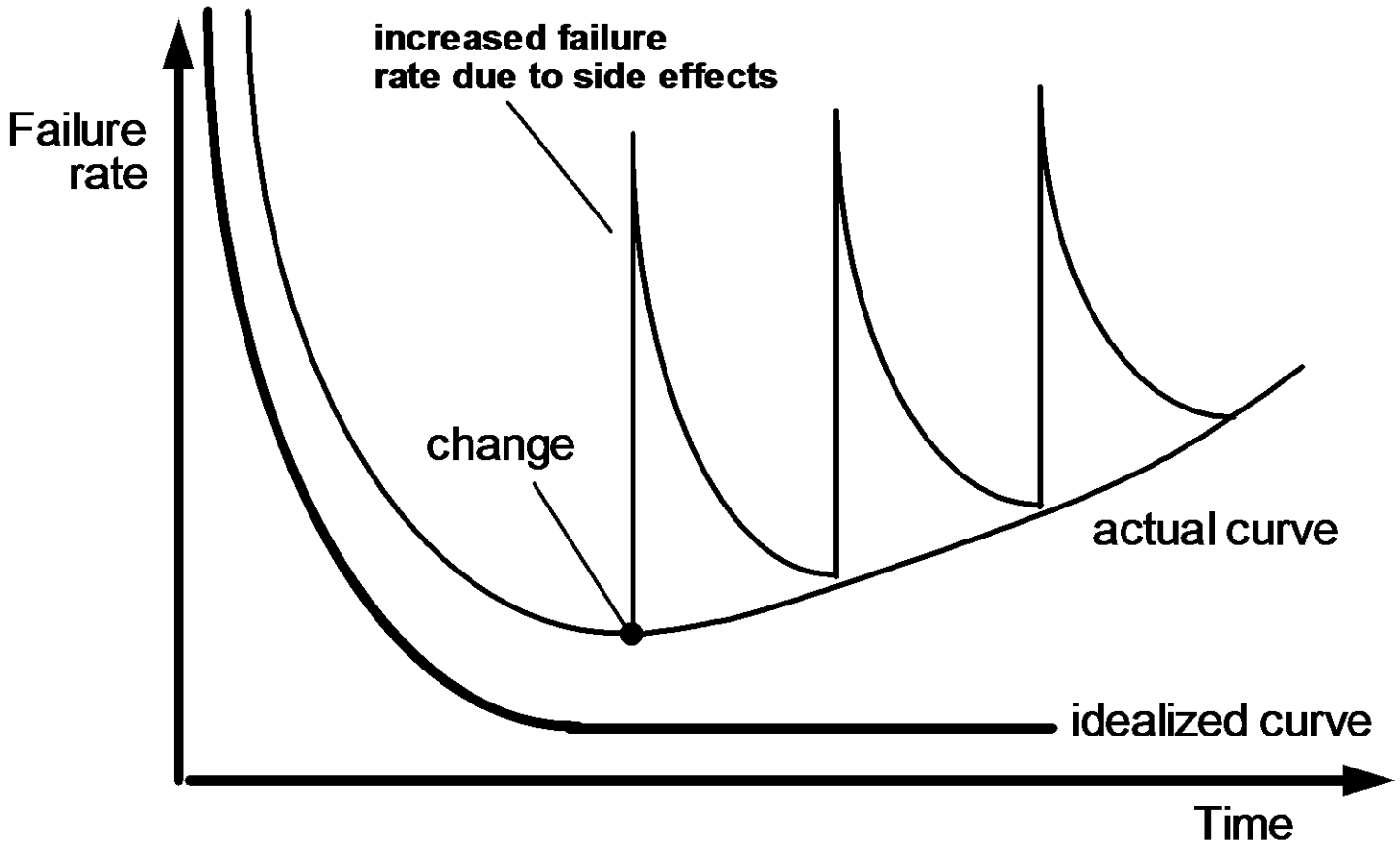
- El software es *más* que programas.
- Hay una característica de él que *debe* atenderse:el hecho que **es un sistema**. *Freeman*
- El software es muchas cosas, pero todos son aspectos de la *información*. En definitiva:
  - si software = programas ejecutables, excluimos una cantidad de información que debemos llamar de alguna manera
  - si incluimos toda la información relevante a una pieza de software ejecutable, entonces nos debemos relacionar con esa información en la misma forma rigurosa y sistemática que lo hacemos con el software ejecutable.

Software - Características

- Cuáles son sus características?
  - El software es único (Cochran)
  - Maleabilidad
  - Es intangible
  - Alto contenido intelectual
  - Su proceso de desarrollo es mano de obra intensivo,basado en equipos y por proyectos
  - No hay separación entre R&D y producción


Burn in
Useful Life
Wear out
Failure Rate
λ
Time


Failure
rate
idealized curve
Time


increased failure
rate due to side effects
Failure
rate
change
actual curve
idealized curve
Time

Trabajo en la industria del software, en la que el cambio es norma. Un programa de software popular, ya se trate de una enciclopedia electrónica, de un procesador de textos o de un sistema de banca en línea, se mejora cada uno o dos años incorporándole importantes características nuevas e incontables perfeccionamientos. Antes de efectuar las mejoras, escuchamos la opinión de los clientes y estudiamos las oportunidades que ofrece la nueva tecnología.

Software – Características - Brooks

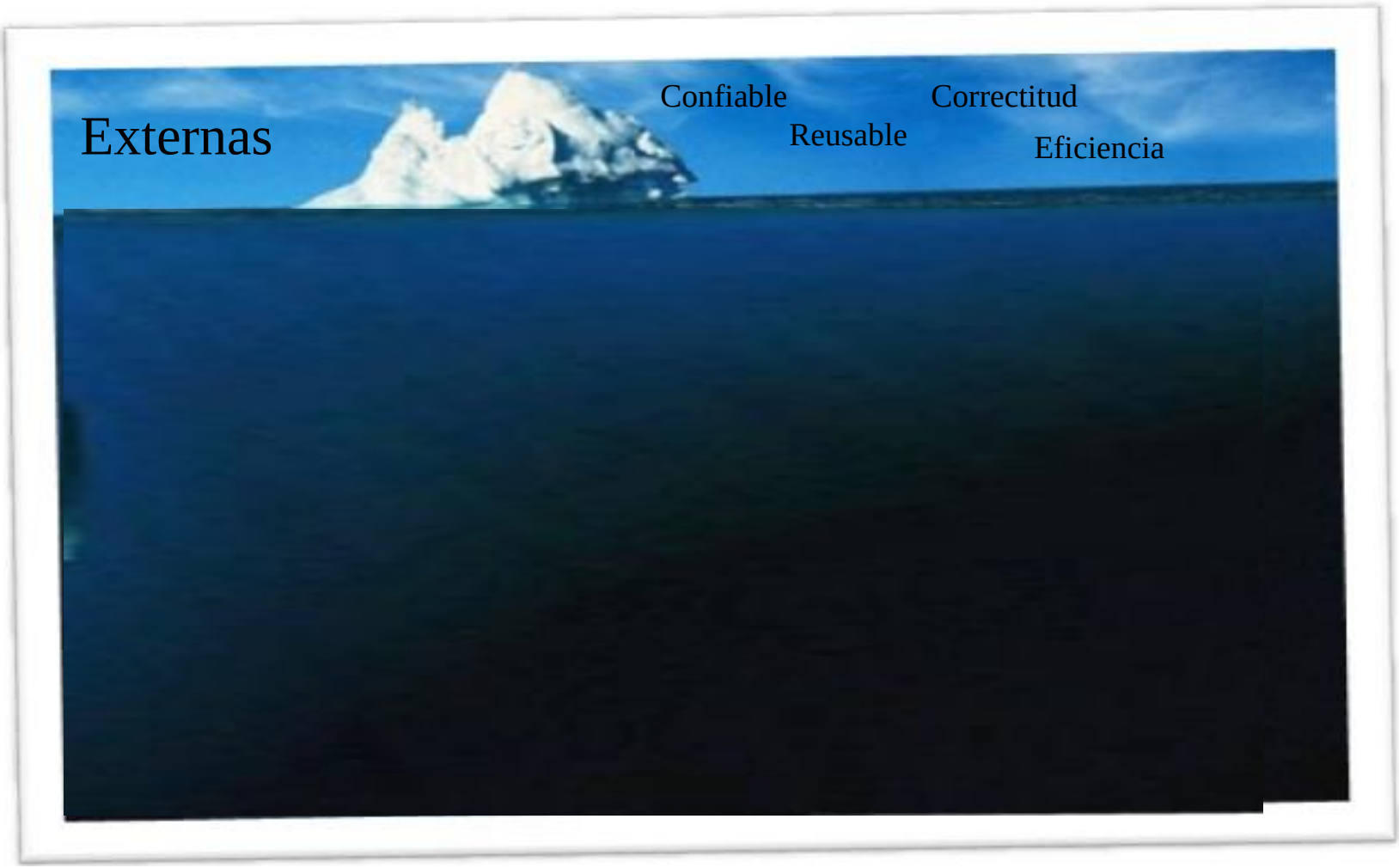
- ¿Cuáles son sus características?
  - **"No Silver Bullet" Brooks**
    - **Características Esenciales**
      - Complejidad
      - Conformidad
      - Cambio
      - Invisibilidad
    - **Características Accidentales**
      - Lenguajes de más alto nivel
      - Tiempo compartido
      - Ambientes de desarrollo

Actividad N° 1.1

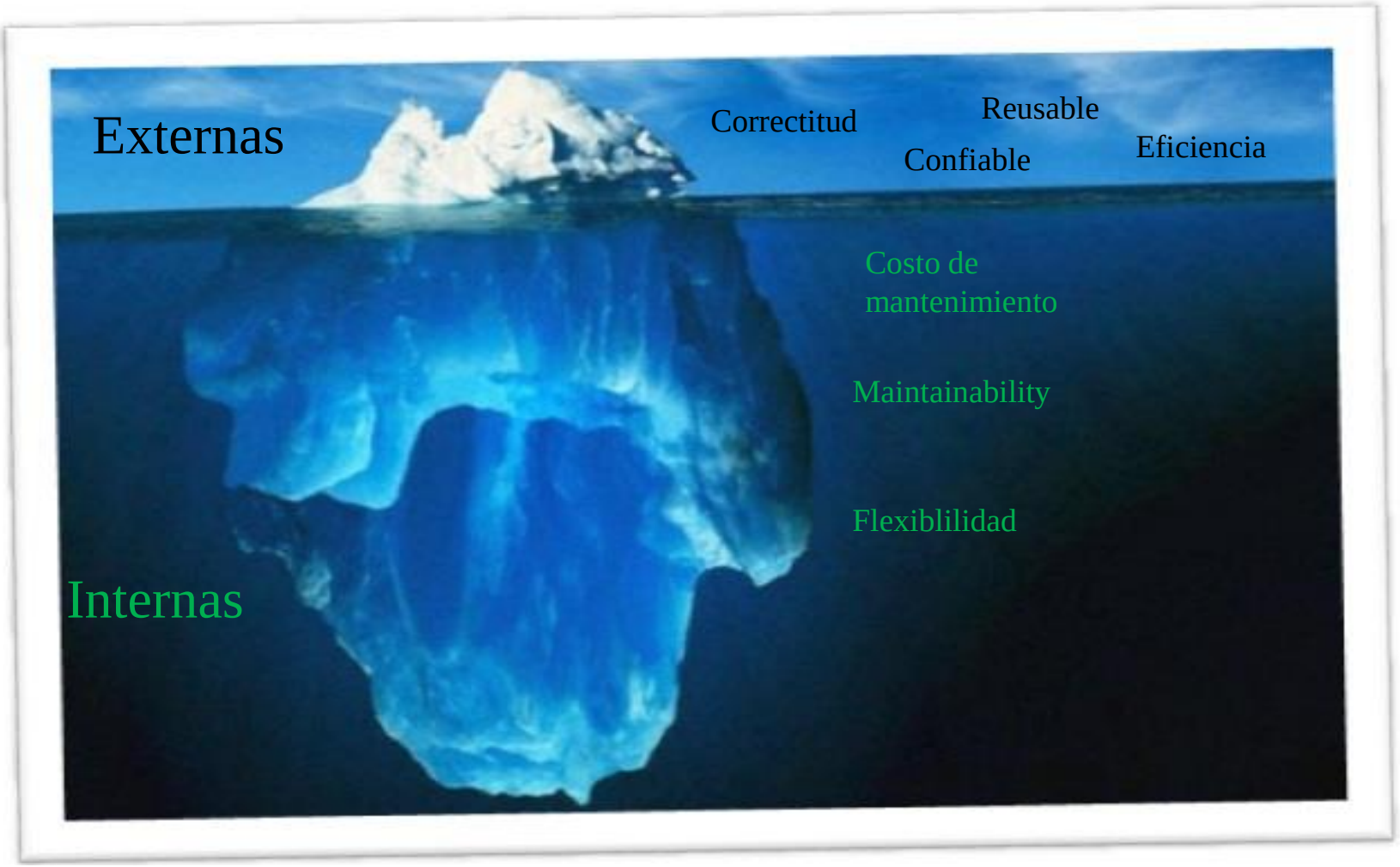
- "El software es algo complejo, difícil de visualizar, que tiene que resolver problemas caprichosos y cambiantes y debe funcionar en diversidad de medios, que al mismo tiempo son inestables"

- Realizar un resumen del Paper "No Silver Bullet"
- **Requisitos a evaluar**
  - No más de 500 palabras

Software – Cualidades



Software – Cualidades



Software – Cualidades

Usuario	Productor	Project manager
• Confiable • Eficiente • Fácil de Usar	• Verificable • Mantenible • Portable • Extensible	• Productivo • Fácil de controlar

- Cualidades del software
  - Corrección funcional
  - Confiabilidad
  - Robustez
  - Performance
  - Amistosidad
  - Verificabilidad
  - Mantenibilidad
- Procesos y productos
  - Reusabilidad
  - Portabilidad
  - Comprensibilidad
  - Interoperatividad
  - Productividad
  - Oportunidad
  - Visibilidad

- **Corrección funcional**
  - Un software es funcionalmente correcto si se comporta de acuerdo a las especificaciones de las funciones que debe proveer
    - se asume que existe una especificación
    - equivalencia entre software y su especificación
  - En general...
    - Las especificaciones de sistemas medianos-grandes no son completas
    - No se logra verificar todo
    - Cómo garantizamos la corrección de la verificación?
  - Y nunca garantiza que se implementen los requerimientos deseados

- **Confiabilidad (reliability)**
  - Informalmente, el software es confiable si el usuario puede depender de él
  - Formalmente, es la probabilidad de que el software opere según lo esperado en un cierto período de tiempo
  - garantía de confiabilidad vs. disclaimer
- **Robustez:** Un software es robusto si se comporta "razonablemente" aún en circunstancias no definidas en la especificación
- **Performance:** Eficiencia, precisión, seguridad (security), tiempo de respuesta...
- **Usabilidad:** Facilidad de uso por parte de los usuarios. Diseño de interfaces
- **Eficiente** Usa de una forma económica los recursos disponibles. El software no debería malgastar el uso de los recursos del sistema

- **Mantenibilidad**

  Reparabilidad: corrección de defectos con poco trabajo:
  - Buen diseño modular
  - Information hiding

  Evolucionabilidad:
  - Buena documentación
  - Anticipar el cambio!
- **Comprensible:** Debe ser fácilmente comprensible por los otros
- **Interoperable:** Debe poder cooperar y co-existir con los otros sistemas
- **Reusable:** Puede integrarse como componente dentro de otras aplicaciones.
- **Portable:** Puede ser usado, sin necesidad de modificarlo, en diferentes plataformas.
- **Verificable:** Puede ser verificado fácilmente.

- **Verificable** Puede ser verificado fácilmente
- **Mantenible** Puede ser modificado fácilmente y los cambios que se efectúan sobre algunos componentes de una misma aplicación no deben afectar a los otros componentes. El software debería poder evolucionar para satisfacer requerimientos de cambio.
- **Portable** Puede ser usado, sin necesidad de modificarlo, en diferentes plataformas.
- **Comprensible** Debe ser fácilmente comprensible por los otros
- **Interoperable** Debe poder cooperar y co-existir con los otros sistemas
- **Reusable** Puede integrarse como componente dentro de otras aplicaciones.

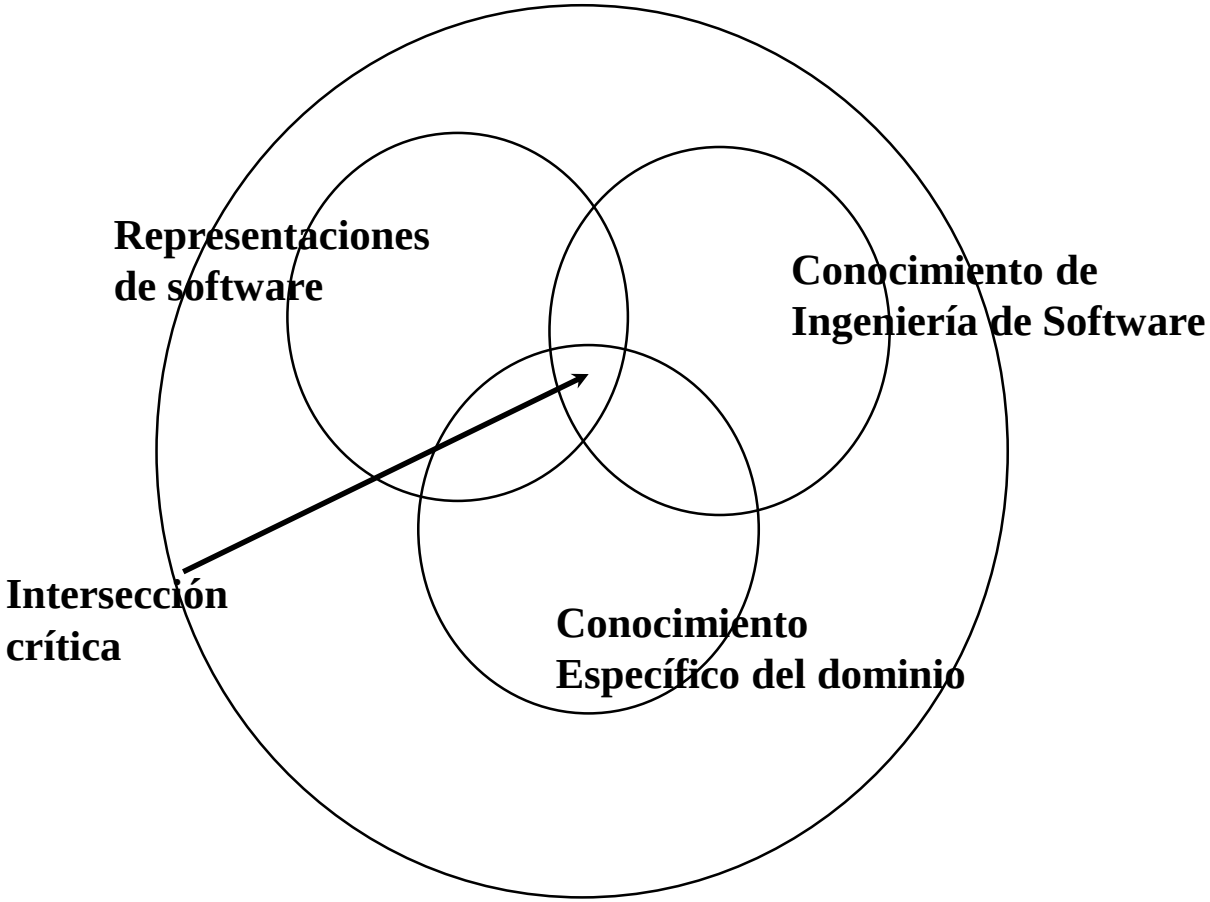
Atributos de Calidad del Software (Bell 2000)

- **Fiable**
- **Eficiente**
- **Robusto**
- **Correcto**
- **Portable**
- **Adaptable (extensibilidad)**Modificar alguna función sin que afecte a sus actividades.
- **Inteligible**: Diseño claro, bien estructurado y documentado.
- **No Erróneo**: No exista diferencia entre los valores reales y los calculados
- **Reutilizable** (reusabilidad)

Atributos de Calidad del Software (Sommerville 2004)

- Mantenibilidad
- Confiabilidad
  - fiabilidad
  - seguridad
  - protección
- Eficiencia
- Usabilidad

Software. Clases de información



Extraído de "El Tao de la Programación" por Seth Robertson

Hardware se encontró al **Software** en el camino a Changtse.

Software dijo: *"Tu eres el Yin y yo soy el Yang. Si viajamos juntos nos volveremos famosos y ganaremos enormes sumas de dinero."*

Y así, la pareja marchó junta, pensando en conquistar al mundo.

Luego ellos conocieron al **Firmware**, quien estaba vestido con harapos raídos, y cojeaba apoyado en un bastón espinoso.

El Firmware les dijo: *"El Tao reside más allá del Yin y Yang. Es silencioso y erguido como un pozo de agua. No busca fama, por lo tanto nadie sabe de su presencia.*

No busca fortuna, porque es completo dentro de sí mismo. Existe más allá del espacio y el tiempo."

El Software y Hardware, avergonzados, regresaron a sus casas.

- **Software como producto**
  - Desde los 60 (separación Sw y Hw) comenzó a constituirse como producto
  - El software es tanto un producto como un objeto técnico

- **Software como conocimiento**
  - Si los programas, contienen conocimiento, entonces las versiones iniciales también contienen conocimiento, y lo perdemos si nos reducimos a los conceptos del software ejecutable.
  - No perder este conocimiento es una de las principales razones de la reusabilidad del software.

Software – Características conceptuales

- Su producción es **humano-intensiva**.
- Tradicionalmente en la ingeniería el ingeniero dispone de herramientas para describir el producto que son distintas del producto, no es así en la ingeniería de software.
- Las cualidades del producto de software están a menudo entremezcladas en especificaciones con las cualidades del diseño

Software como **producto y vehículo** por el cual se entrega el producto más importante de nuestro tiempo: **La Información**

Software - Cronología

- **1954** Primer assembler (IBM701)
- **1955** UNIVAC pone en marcha el primer sistema comercial:Sueldos y Jornales
- **1956** ERM (Bank of America) revoluciona la banca comercial
- **1962** Nace Informatics, compañía de servicios de software que tendría el primer producto con un millón US$ de ventas (Mark IV). Anuncia el Mark I: atacaba recuperación de información y generación de reportes.
- **1963** Sistema SABRE de American Airlines SKETCHPAD: precursor del CAD WYSWYG (What You See is What You Get
- **1964** IBM 360 – primer OS de gran envergadura PL/1 BASIC. Nace ADR, la empresa que produjo el primer producto de software (Autoflow)
- **1965** Se estima que había de 40 a 50 empresas de servicios de software
- **1968** Conferencia de la NATO, se habla por primera vez de Software Engineering. ADR patenta autoflow: primera patente de software de la historia. Reconoce al software como un producto no como un servicio
- **1969** Codd propone el modelo relacional a IBM (System R) UNIX (Thomson y Ritchie en ATT)

Software - Cronología

- **1971** 29 productos exceden el millón de dólares de ventas
- **1972** Smalltalk en Xerox PARC
- Debut de la tomografía computada
- **1975** Primer BASIC de microcomputadora (B. Gates y P.Allen)
- **1977** ARPA define los protocolos TCP/IP.
  Primera Conferencia International de Ingeniería de Software
  Comienza a publicarse IEEE Transactions on Software Engineering
- **1979** dBase II (George Tate en Ashton tate)
- **1980** Primer procesador de texto (Brian Reid)
- **1983** Microsoft anuncia Windows - Unix System V
- **1984** DB2 base relacional de IBM Comienza a publicarse IEEE Software
- **1985** Lanzamiento de Windows- Cray 2
- **1990** Lanzamiento Windows 3.0
- **1992** Sun Solaris Sistema operativo multihilo- multiprocesamiento
- **1993** Windows NT
- **1995** Java – Windows 95

Actividad N° 1.2

- Realizar un su propia cronología del software que considere.

- **Requisitos a evaluar**
  - Actividad Grupal
  - Próxima clase

Perspectiva del desarrollo de software

Década 50-60:

"Software como un añadido".
Desarrollo artesanal, a medida.
Lenguajes de bajo nivel.

Década 60-70:

Software como producto.
Década lenguajes y compilación.
"Crisis del software".

Década 70-80:

Programación estructurada.
Ingeniería del Software.
Primeros métodos estructurados.

Década 80-90:

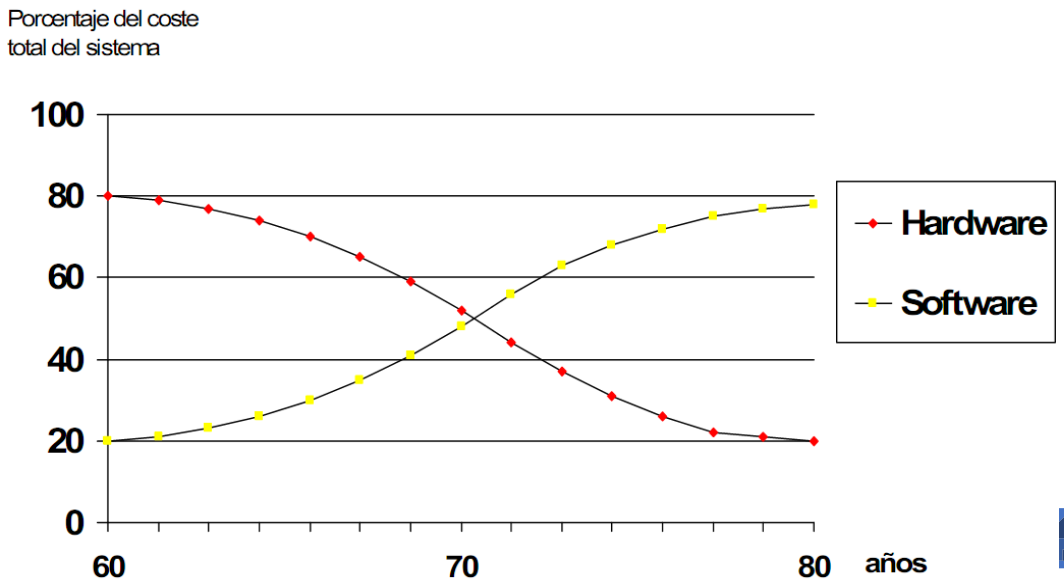
Tecnología de SGBDs, SOs...
Nuevos paradigmas de programación y de producción de programas:OO C/S

90's - actualidad:

POO, programación visual
Análisis/Diseño OO.
UML (1997)
Tecnología CASE (2ª generación)
Métodos ágiles. Componentes y reutilización
Interoperabilidad (CORBA, .NET...)
Internet Comercio electrónico

La problemática actual del software

- Incapacidad para estimar tiempo, costo y esfuerzo para el desarrollo de un producto software.
- Falta de calidad del producto software.
- Avance del hardware y necesidad de aplicaciones más complejas.
- Cambio en la relación entre el coste hardware/software.

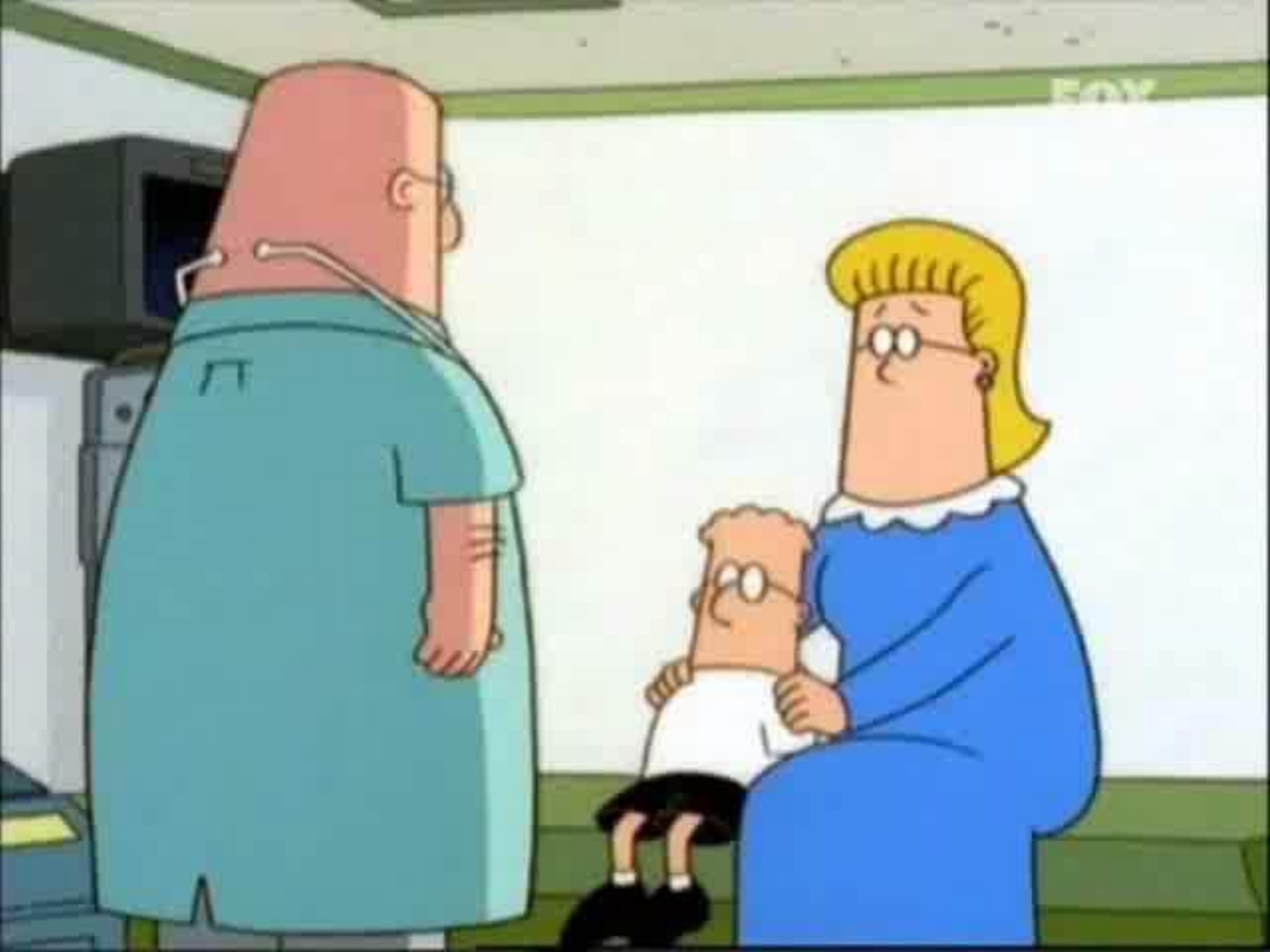


La problemática actual del software

- ¿Porqué lleva tanto tiempo terminar los programas?
- ¿Porqué es tan elevado su costo?
- ¿Porqué no podemos encontrar todos los errores antes de entregar el software a nuestros clientes?
- ¿Porqué nos resulta difícil constatar el progreso conforme se desarrolla el sw.?

Algunas causas

- Naturaleza “no física” de la programación.
- Problemas derivados de la intervención de grupos.
- Problemas de comunicación con los clientes.
- Poco esfuerzo en el análisis y el diseño.
- Problemas de gestión.
- Planificaciones optimistas, plantillas poco cualificadas
- A veces, el sw debe “solucionar” los problemas del sistema global.
- Difusión limitada de las nuevas técnicas, métodos y herramientas.
- ..industria pendiente de su ‘revolución industrial’.


FOX

¿Qué es la Ingeniería de Software?

"La profesión condenada"

E. Dijkstra

- Definamos primero Ingeniería:

> La ciencia por la cual las propiedades de la materia y las fuentes de energía de la naturaleza son convertidos en elementos útiles a los hombres por medio de estructuras, máquinas y productos.
>
> Webster's Dictionary

Ingeniería Software

- *"(1) El uso de métodos sistemáticos, disciplinados y cuantificables para el desarrollo, operación y mantenimiento de software, es decir, la aplicación de prácticas de Ingeniería Software. (2) el estudio de técnicas relacionadas con (1)"* - IEEE –

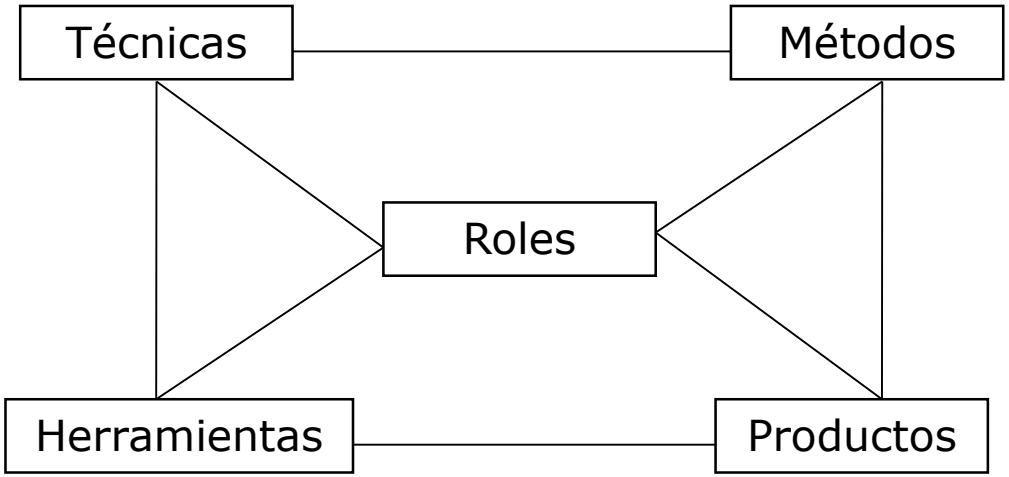
- *"es la disciplina tecnológica y de administración que se ocupa de la producción y evolución sistemática de productos de software que son desarrollados y modificados dentro de los tiempos y costos estimados"* - Fairley –

- *es el campo de la ciencia de la computación que trata con la construcción de sistemas de software que son tan grandes o complejos que son construídos por un equipo o equipos de ingenieros* - Ghezzi -

- *"La aplicación práctica del conocimiento científico al diseño y construcción de programas y la documentación asociada requerida para desarrollar, operar y mantenerlos." Boehm*

- *La Ingeniería del Software es la disciplina que establece el uso de principios de ingeniería robustos, orientados a obtener software económico que sea confiable y funcione de manera eficiente" Pressman*

- Resumiendo, la **Ingeniería de Software** es:
  - conjunto de Procesos cuya finalidad es construir un producto de software de calidad por medio de técnicas y herramientas establecidas
  - ¿ Cuales teorías, métodos e instrumentos ?
  - ¿ Qué es un producto software?
  - ¿ Qué es un software de calidad?



Principios Ingeniería de Software

Cada estrato se basa en los inferiores y es más susceptible a cambios con el paso del tiempo

Principios de la Ingeniería del Software

- **Rigor y formalidad.**
  - Sólo una aproximación rigurosa puede producir productos más confiables, controlar sus costos e incrementar su confiabilidad.
- **Separación de intereses.** Se puede hacer según varios criterios:
  - Tiempo (ciclo de vida del software)
  - Cualidades
  - Vistas (flujo de datos, de control, estático, dinámico, etc.)
  - Partes (estructura, diseño)
- **Modularidad.** No sólo aplica a los aspectos estructurales, sino a todo el proceso de desarrollo.
- **Abstracción.** Los modelos que construimos para entender los fenómenos son abstracciones de la realidad.
- **Anticipación del cambio**
- **Generalidad.** Focalizar su atención en el descubrimiento de un problema más general que puede estar oculto detrás del problema en cuestión
- **Incrementalidad .** Este principio puede aplicarse al identificar tempranamente subconjuntos útiles de una aplicación para así obtener rápido feedback.

- ¿Cuándo un proyecto es exitoso?
  - Entregó la funcionalidad requerida
  - Tardó lo presupuestado
  - Costó lo presupuestado
  - El producto obtenido ES utilizado
  - Quien usa el producto está satisfecho
  - Está asegurada la continuidad en la utilización del producto por un lapso adecuado

Tener una formulación clara y no ambigua de la visión o misión del proyecto	68
Tener un líder técnico capaz de liderar el proyecto exitosamente	63
Estar cada persona comprometida con el proyecto	63
Trabajar bien todos juntos	60
Tener una especificación escrita y detallada de lo que se supone que hará el software	58
Tener documentos detallados de arquitectura y diseño	55
Realizar tempranamente entrevistas del equipo del proyecto con gente que usará el software	53
Tener un prototipo de la interfase del usuario que sea realista y demuestre de manera fiel la funcionalidad que el sistema tendrá	52
Tener el equipo de proyecto todo la experiencia necesaria para completar el proyecto	51
Tener un Plan de Desarrollo de Software escrito y detallado, el cual incluye tiempo para vacaciones, días feriados, días perdidos y entrenamiento y asignar los recursos a menos del 100%.	48
Considerar todos los miembros del proyecto que la visión es realista	45
Tener el equipo del proyecto la experiencia en el ambiente de negocios en cual el software operará	42
Incluir en el plan de proyecto una lista de los riesgos actuales y que se actualice periódicamente	33

Cómo obtener un producto de calidad? Hace unos años...

- Lo importante era "que ande"
- No existían prácticas consensuadas
- La búsqueda de la bala de plata que solucione todos los problemas del software

Cómo obtener hoy un producto de calidad

- No existe la bala de plata
- No sólo "que ande"
- Responsabilidad profesional
- Prácticas consensuadas (*best practices*), como en otras disciplinas

De dónde salen estas prácticas?

- Asociaciones profesionales / Centros especializados
- Profesionales prestigiosos / Trabajos especializados
- Como en otras disciplinas, comienza a existir un “conocimiento profesional establecido” (que no quiere decir que sea perfecto)

Algunas ´´best practices´´

- Análisis de factibilidad
- Análisis de riesgos
- Planificación/seguimiento
- Control de configuraciones
- Automatizar - Uso de herramientas
- Análisis de requerimientos
- Diseño
- Inspecciones y revisiones
- Testing

- El problema: Aún no sabemos cómo desarrollar software exitosamente
  - De la crisis del Hardware (60's 70's)...
  - ... pasamos a la ***Crisis del Software*** ...
  - ... pero el verdadero problema *¿está en la gente ?*

Peopleware is the real issue. We are the problem and we are the solution. How convenient"."Constantine On Peopleware", L. Constantine

- **Síntomas de la crisis del software (finales 70). El Software:**
  - No es fiable y necesita de un mantenimiento permanente.
  - A menudo es imposible de mantener, carece de transparencia y no se puede modificar ni mejorar.
  - Se entrega muy a menudo con retrasos y con costos superiores a los presupuestados,
- **más que de una crisis puntual se trata de una enfermedad crónica**
- **la fábrica de software envejece:**
  - Hay aplicaciones de gestión con veinte años de antigüedad, muchas modificaciones acumuladas, muy difíciles de mantener:la más pequeña modificación puede hacer que falle todo el sistema.
  - Existen aplicaciones de ingeniería antiguas, pero vigentes, cuya estructura interna nadie conoce con detalle.
  - Existen sistemas empotrados que presentan un comportamiento inexplicable, bajo ciertas circunstancias, pero que no pueden darse de baja porque no hay otra aplicación que las reemplace.

- ¿Qué características posee el desarrollo de software?
  - Es una ***actividad mental***
  - Realizada de ***manera grupal***
  - Cuyo producto final tiene características esenciales muy particulares
  - ... y cada día hay que hacer y trabajar en algo distinto!!

- ¿Qué características deben poseer los procesos de desarrollo de software?
  - Flexibles
  - Tener aceptación
  - Deben atacar los problemas esenciales del software

- “Una cultura incluye un conjunto de **valores**, **objetivos** y **principios** compartidos que guían el **comportamiento**, las **actividades** y **prioridades** de un grupo de personas trabajando hacia un objetivo común” Creating a Software Engineering Culture, K. Wiegers

- **¿Qué sucede cuando no se entiende que es desarrollar software?**
  - Se crean mitos
  - Se cree en “balas de plata”
  - No hay gerenciamiento

Actividad N° 1.3

- Tomando como referencia el Paper "**Creating a Software Engineering Culture" Karl Wiegers** , realizar un análisis del impacto de la cultura de la organización en el desarrollo de software.

- **Requisitos a evaluar**
  - Actividad Grupal
  - Vencimiento 15 días

- Mitos de la administración
  - *" Ya se tiene un libro lleno de estándares y procedimientos para la construcción de software. ¿ Esto proporcionará a mi gente todo el conocimiento necesario?"*
  - *"Si se está atrasado en el itinerario es posible contratar más programadores para así terminar a tiempo"*
  - *"Si decido subcontratar el proyecto de software a un tercero, puedo relajarme y dejar que esa compañía lo construya"*

- Mitos del cliente
  - *"Un enunciado general de los objetivos es suficiente para comenzar a escribir programas; los detalles se pueden afinar después"*
  - *"Los requerimientos del proyecto cambian de manera continua, pero el cambio puede ajustarse con facilidad porque el software es flexible?"*

- Mitos del desarrollador
  - *"Una vez que el programa ha sido escrito y puesto a funcionar, el trabajo está terminado"*
  - *"Mientras el programa no se esté ejecutando, no existe forma de evaluar su calidad"*
  - *"El único producto del trabajo que puede entregarse para tener un proyecto exitoso es el programa en funcionamiento"*
  - *"La ingeniería del software obligará a emprender la creación de una documentación voluminosa e innecesaria y de manera invariable tornara mas lento el proceso"*

- El cliente transmite siempre de forma clara y completa lo que necesita en su sistema de información.

- Cualquier persona con habilidad o conocimientos básicos de computación o de programación puede desarrollar un software que funcione adecuadamente, ya que es una actividad que no requiere conocimientos especializados

- Un software de calidad es aquel en donde no se presentan errores, y que no hay que modificar nunca porque se diseño perfectamente en las primeras etapas del desarrollo.

- Conclusión:
  - Es una “actividad mental”
  - Es una “actividad grupal”
  - No es fácil desarrollar software
  - No es igual a otras ingenierías de desarrollo y producción, hay que tener cuidado con las comparaciones
  - Hay que atacar las características esenciales –Brooks-

Proceso de Desarrollo de Software

Definición de Proceso

- *"un proceso define quién está haciendo qué, cuándo y cómo alcanzar un determinado objetivo"* *Jacobson. Booch. Rumbaugh*

- *"los procesos son unas series de pasos que involucran actividades, restricciones y recursos que producen una determinada salida esperada"* Pfleeger

- *"un proceso es una serie de eventos o fases que tienen lugar con el tiempo y tienen un propósito o resultado identificado"* Olson

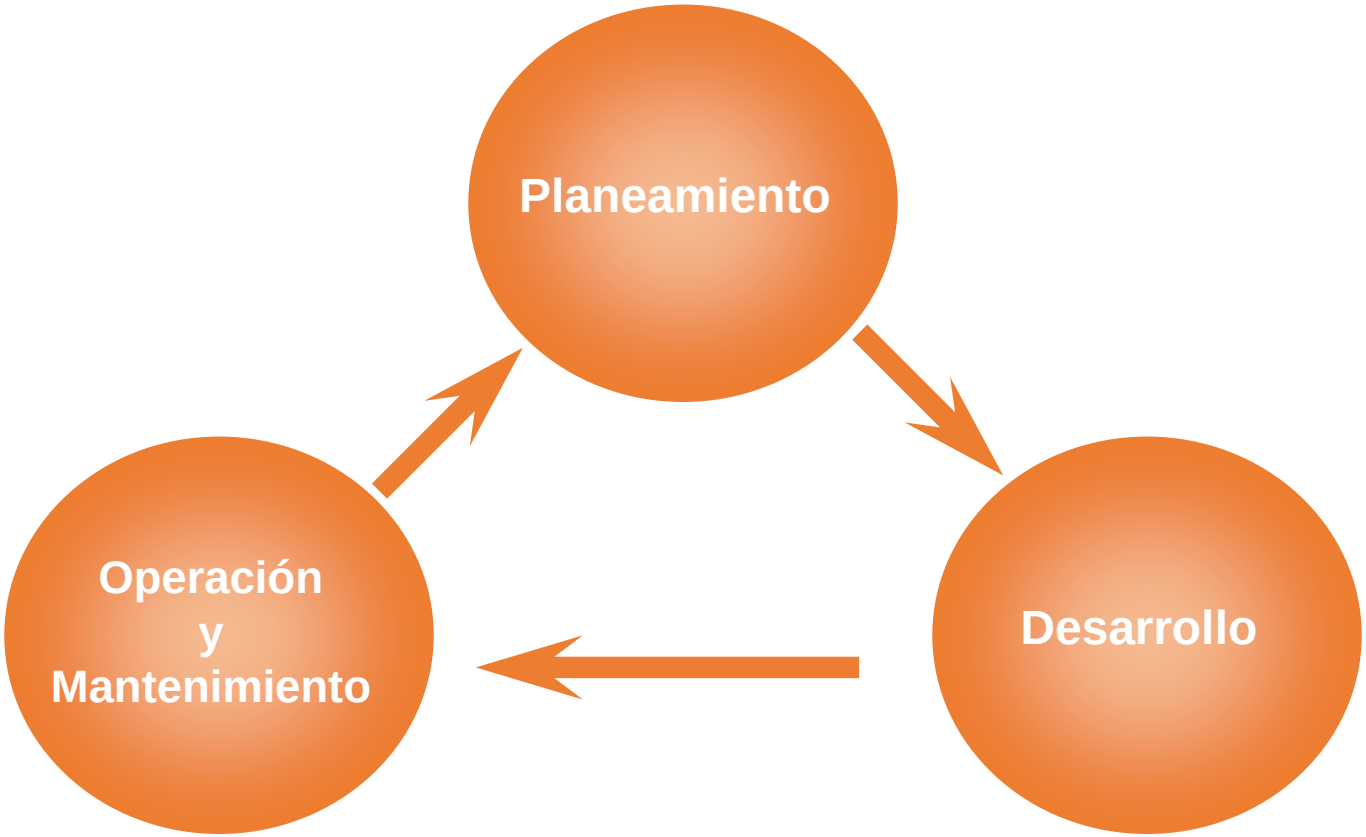
Definición de Proceso

- El proceso por dentro
  - El proceso prescribe todas las actividades principales
  - El proceso usa recursos y produce productos intermedios y finales
  - El proceso puede estar formado de subprocesos
  - Cada actividad de proceso tiene un criterio de entrada y salida
  - Las actividades son organizadas en una secuencia.
  - Cada proceso tiene un conjunto de principios guías
  - Restricciones o controles pueden aplicarse a una actividad, recurso, o producto

Importancia de la implementación de los proceso

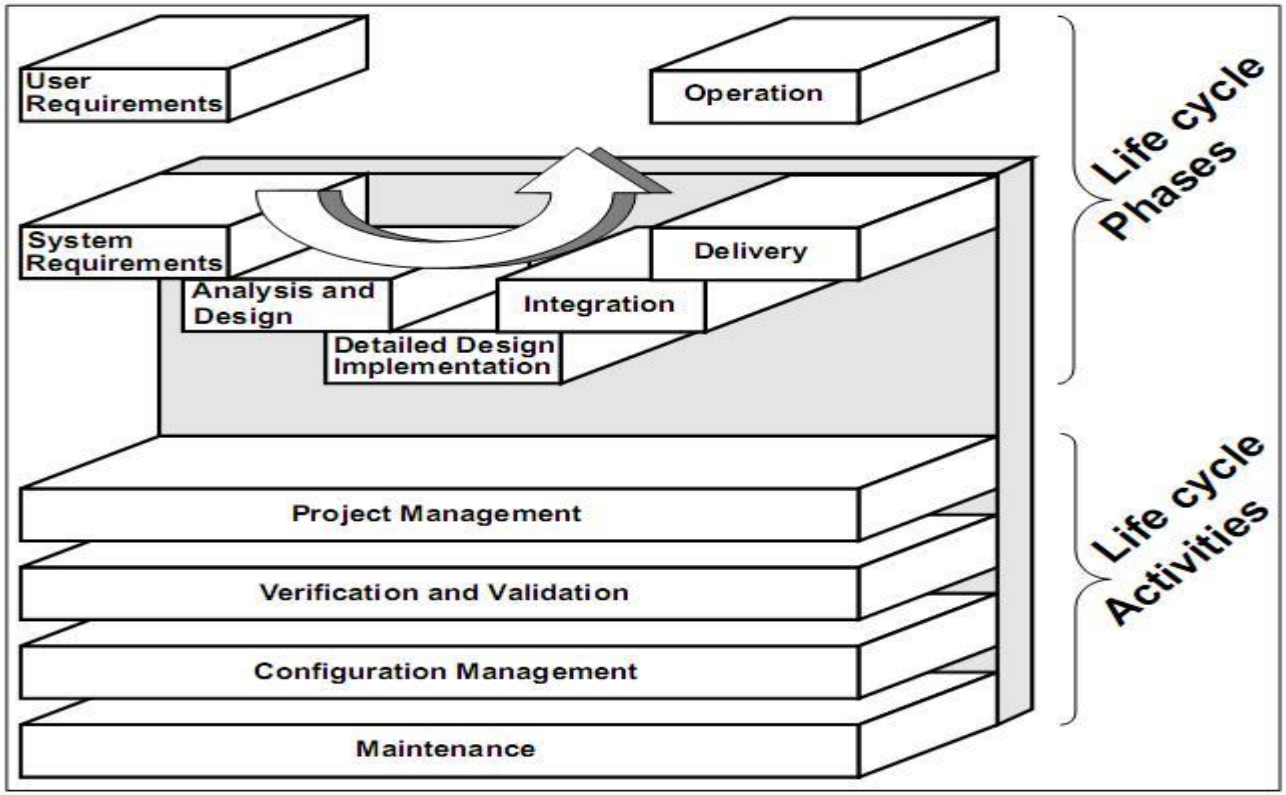
- Imponer consistencia
- Capturar experiencias
- Examinar, comprender, controlar y mejorar las actividades

Proceso de desarrollo – Ciclo de vida



Proceso de desarrollo de software

- Un conjunto de actividades, métodos, prácticas y transformaciones que la gente usa para desarrollar y mantener software y los productos asociados



- Especificación del Software
- Desarrollo del Software
- Validación del Software
- Evolución del Software

Proceso de desarrollo de software

- Características principales
  - Visibilidad
  - Predecible
  - Aceptación
  - Fiabilidad
  - Robustez
  - Mantenibilidad
  - Rapidez

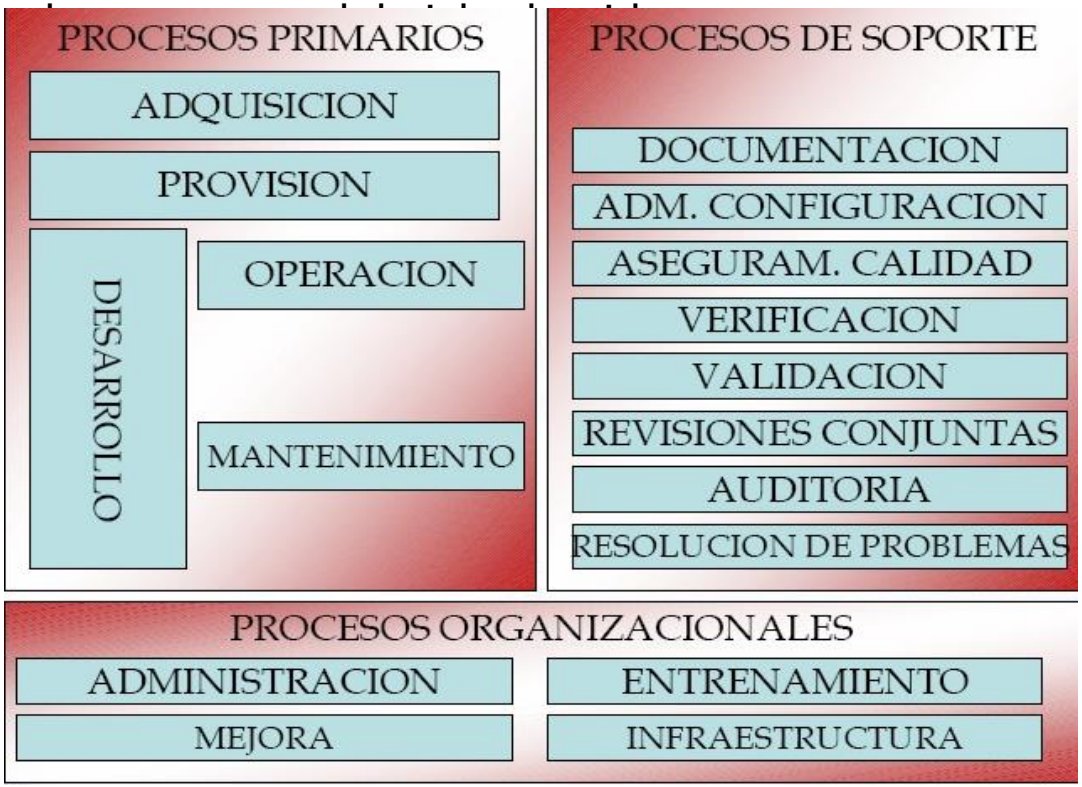
¿ Qué esperan del proceso software ?

- Project Manager
  - Desarrollar un conjunto consistente y gerenciable de documentos del sistema
- Usuario
  - Disponer de un sistema que satisfaga sus necesidades con respecto a calidad, funcionalidad, costos, etc.
- Ingeniero software
  - Contar con un esquema en el que ejecutar sus actividades técnicas
- Empresa
  - Mejorar la habilidad organizacional para desarrollar sistemas de
  - calidad productivamente y rentablememnte


Marco de trabajo del proceso común
Actividades del marco de trabajo
Conjunto de tareas
Tareas
Hitos, entregas
Puntos SQA
Actividades de protección

Procesos del ciclo de vida ISO/IEC 12207

- *Procesos del ciclo de vida primario*: Conducen las principales funciones
- *Procesos del ciclo de vida de soporte:* Dan soporte a otros procesos para realizar una función especial
- *Procesos del ciclo de vida organizacional:* Establecen, controlan y mejora



Proceso de adquisición

- Tareas del que contractualmente adquiere un producto de software o un servicio
- Actividades: Definición de la necesidad, pedido de propuesta (RFP), selección de un proveedor y gerenciamiento del proceso hasta la aceptación del sistema

Proceso de provisión

- El servicio puede ser el desarrollo de un producto de software, un sistema conteniendo software, la operación o mantenimiento de un sistema de software
- Actividades: preparar una propuesta a un comprador, acordar un contrato, identificar los procedimientos y recursos necesarios, desarrollar, ofrecer el servicio

Proceso de desarrollo

- Actividades y tareas del desarrollador del sistema y el software
- Desarrollar software: software nuevo o modificar el existente
- Actividades: Tareas específicas del desarrollo de software

ISO/IEC 12207 - Procesos Primarios -

Proceso de operación

- Abarca la operación del software y el soporte operacional a los usuarios
- Actividades: implementación, testeo operativo, operación del sistema y soporte al usuario

Proceso de mantenimiento

- Causa: error, necesidad de mejora, adaptación.
- Alcance: modificar el código y la documentación asociada, finaliza cuando el sistema se retira del uso
- Actividades: análisis del problema y modificación, aceptación, migración, retiro del software

ISO/IEC 12207 - Procesos Secundarios -

Proceso de documentación: Objetivo: registrar la información producida por los procesos del ciclo de vida.

Proceso de gestión de configuración: Identifica y define el baseline de los ítems de software en el sistema, para controlar modificaciones y versiones de los ítems.

Proceso de aseguramiento de la calidad: Provee un framework para asegurar (el adquirente o el cliente) el cumplimiento de los productos o servicios con sus requerimientos contractuales y el acatamiento de los planes establecidos.

Proceso de verificación: Determina si los requerimientos para un sistema son completos y correctos y que el output de una actividad satisface los requerimientos y condiciones impuestos en actividades previas.

ISO/IEC 12207 - Procesos Secundarios -

Proceso de validación: Determina si el sistema final cumple con el objetivo inicial. No reemplaza otras evaluaciones, solo las complementa.

Proceso de revisión conjunta: Provee un framework para la interacción entre el "revisado" el revisor.

Proceso de auditoría: El auditor evalúa los productos y actividades del auditado con énfasis en el cumplimiento de los requerimientos y planes.

Proceso de resolución de problemas: Se resuelven los problemas tomando acciones correctivas en la medida que se detectan.

ISO/IEC 12207 – Procesos Organizacionales -

Procesos gerencial: Define las actividades y tareas del manager

Proceso de infraestructura

- La infraestructura puede incluir hardware, software, standards, herramientas, técnicas y medios
- Objetivo: definir las actividades para establecer y mantener la infraestructura necesaria

Proceso de mejora

- Objetivos: mejorar el proceso organizacional
- Actividades: evaluar, medir, controlar y mejorar el proceso de ciclo de vida

Proceso de entrenamiento: Actividades: identificar y hacer un plan para incorporar o desarrollar recursos de personal.

Proceso de desarrollo orientado a proyectos pequeños

- Factores que inciden en la categorización
  - Tamaño de la organización
  - Complejidad del proyecto
  - Interacciones

- Importancia
  - Atención en la actividad favorita
  - Atrapados en la rutina
  - No hay pautas claras

¿ Qué es un modelo de proceso de SW ?

- ¿Para qué modelos de proceso?
  - Ayuda una comprensión común
  - Ayuda encontrar inconsistencias, redundancias y omisiones
  - El modelo debería reflejar los objetivos de desarrollo
  - Permite evaluar actividades candidatas para atacar estos objetivos.
  - Permite la adecuación a cada situación especial
  - Facilita las mediciones

- Modelo de proceso y métodos (Basili)
  - Un modelo de proceso define la secuencia de métodos y el desarrollo de los documentos anexos
  - Características de un método:
    - Input
    - Output
    - Técnica
    - Formalidad
    - Énfasis
    - Definición del método
    - Perspectiva
    - Calidad del producto
    - Calidad del proceso

¿Qué es un Framework de Proceso?

- **Un mecanismo para definir un proceso de desarrollo.**

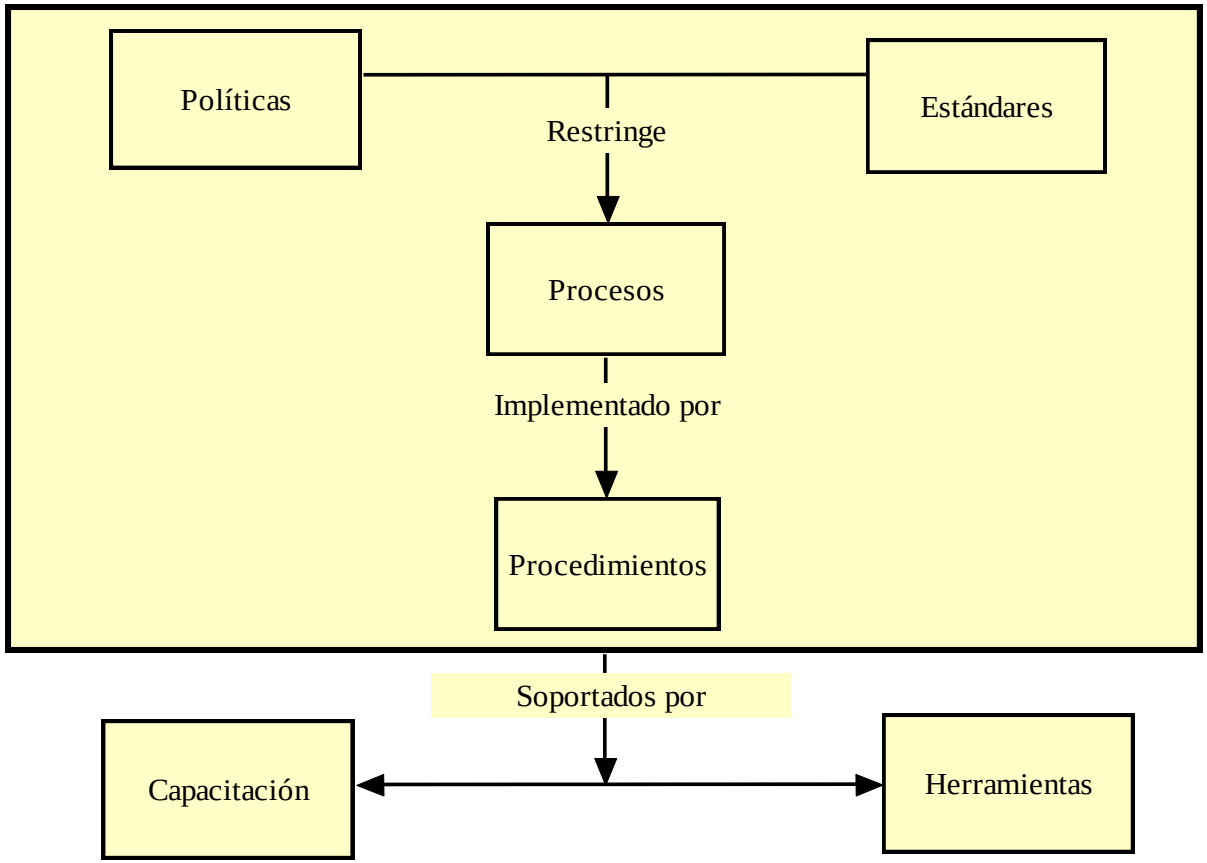
  Es como una fábrica de procesos que construye el proceso.

- **Un repositorio de fragmento de método (tareas, roles, artefactos y guías) y elementos del proceso que pueden adaptarse a necesidades individuales para definir un proceso de desarrollo.**

  Un repositorio con información de buenas prácticas, libros, publicaciones, estandarizaciones, etc.

Software Process Framework

- Herramienta de definición de procesos
- Operational Framework



Software Process Framework

- Template
  - Rol
  - Capacitación
  - Actividades
  - Criterio de entrada
  - Inputs
  - Criterio de salida
  - Outputs
  - Revisiones y auditoria
  - Administración y control de los productos de trabajo
  - Mediciones
  - Procedimientos documentados
  - Herramientas

Template

Código:	Nombre:	

Objetivos:

Justificación:

Actividades:

Paso	Nombre Actividad	Objetivo	Rol

Recursos:

Humanos:		Tecnológicos:

Técnicas:

Input:

Nombre	Origen	Criterio de Entrada: Estado o Condición	Tipo

Output

Nombre	Destino	Criterio de Salida: Estado o Condición	Tipo

Definiciones:

Restricciones:

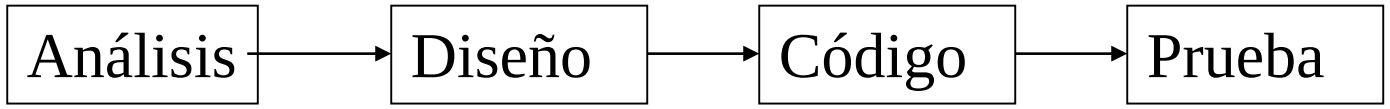
Variables a medir:

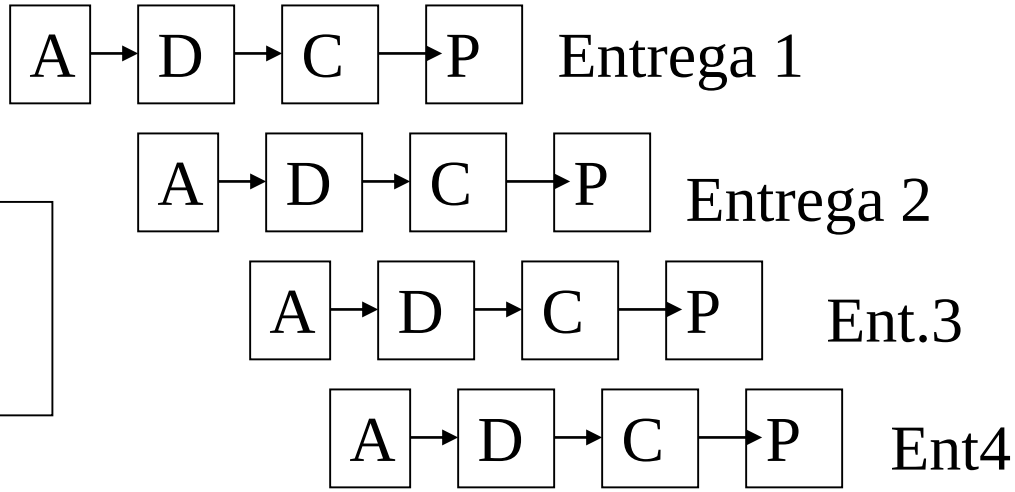
Nombre de la variable	

Observaciones:

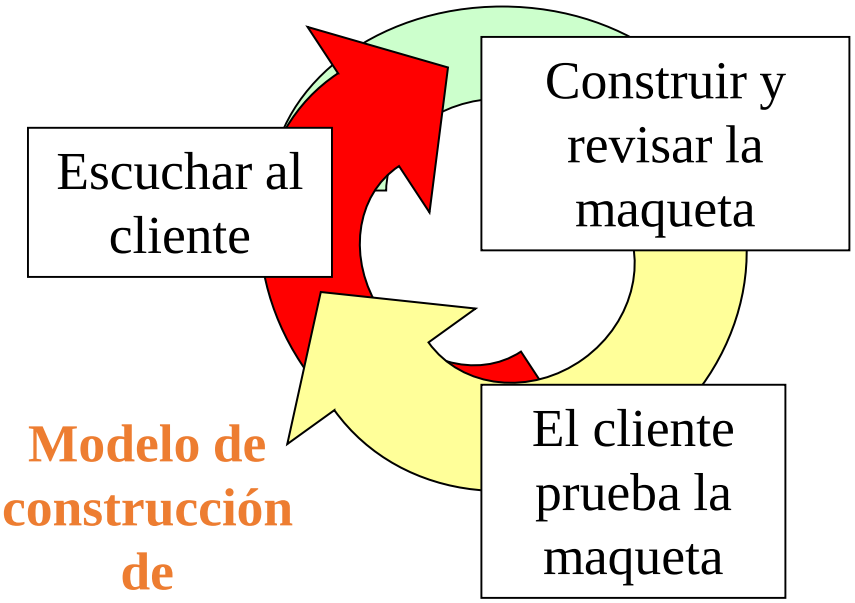
Modelos Genéricos

Modelo Cascada



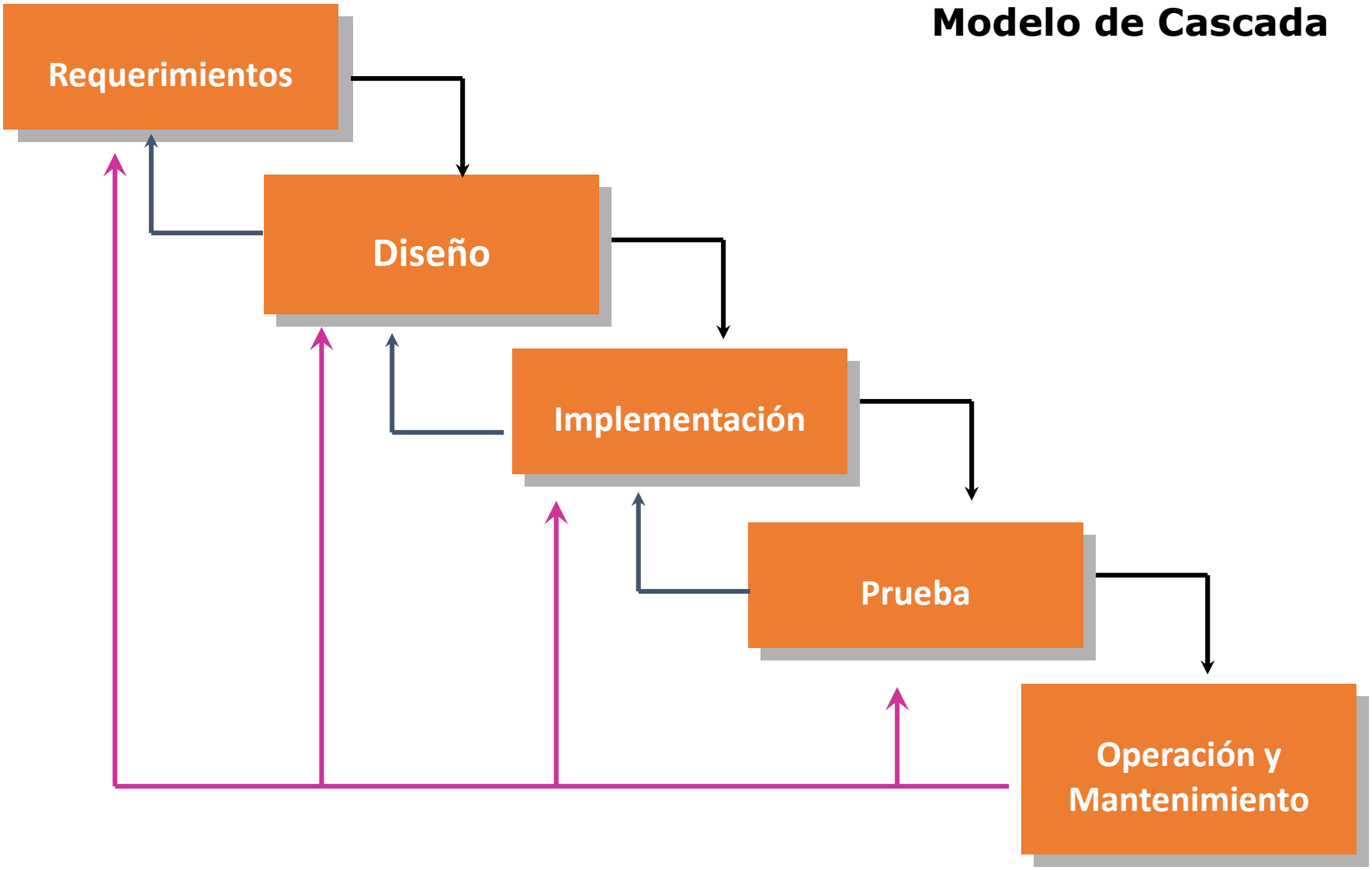


Modelo incremental

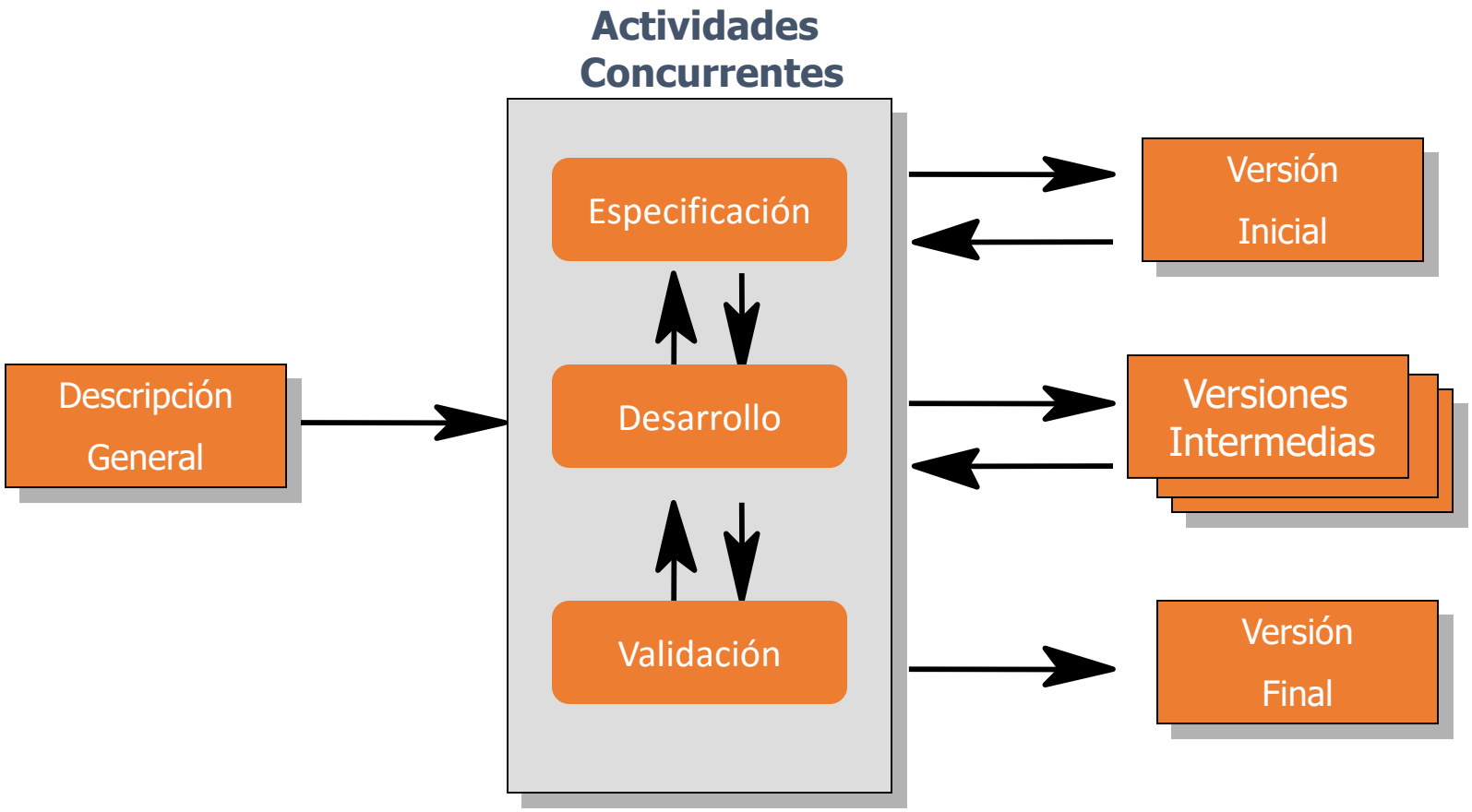


Modelo de construcción de prototipos

Modelo de Cascada

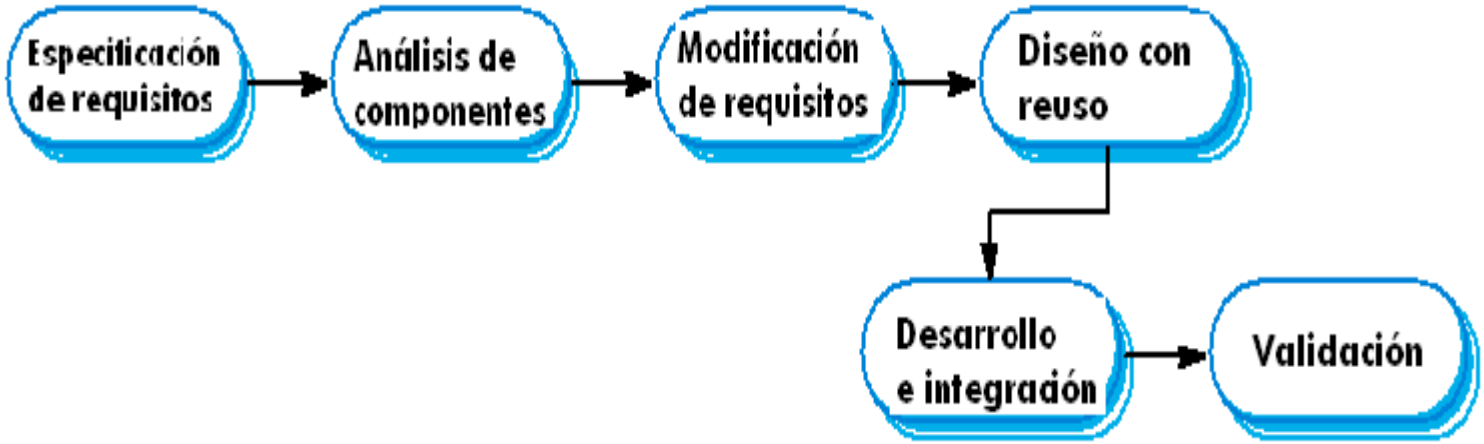


Desarrollo Evolutivo

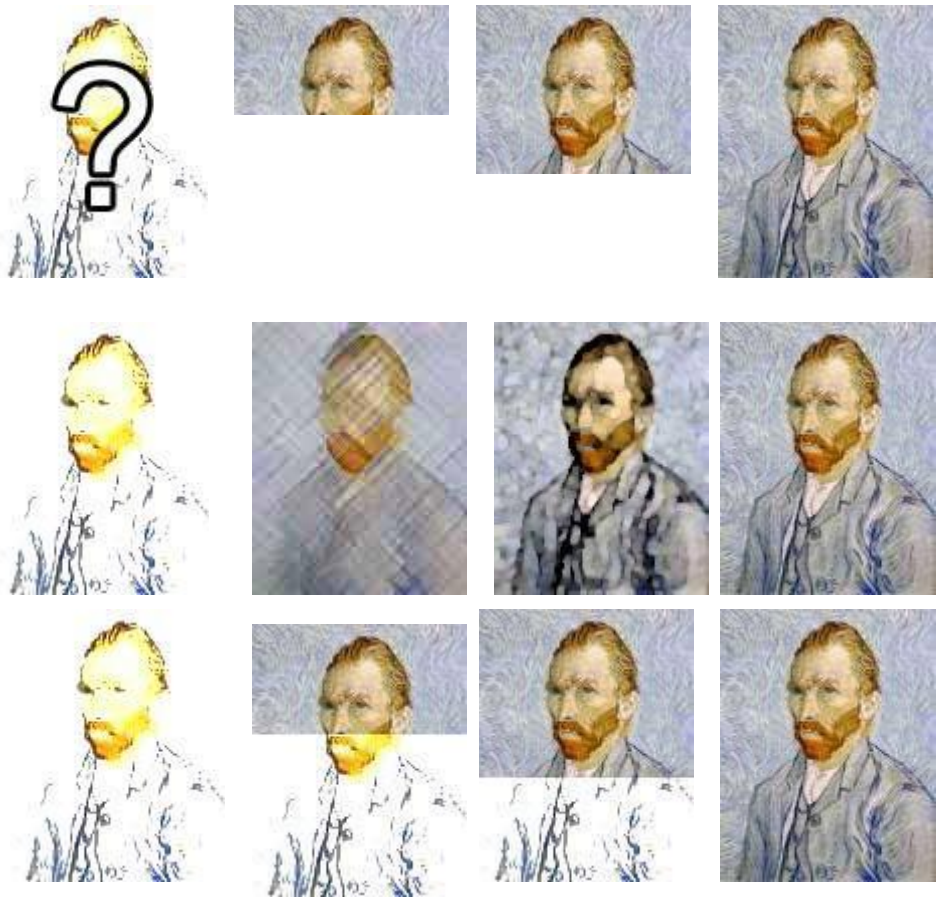


Modelo de Desarrollo de software basado en componentes

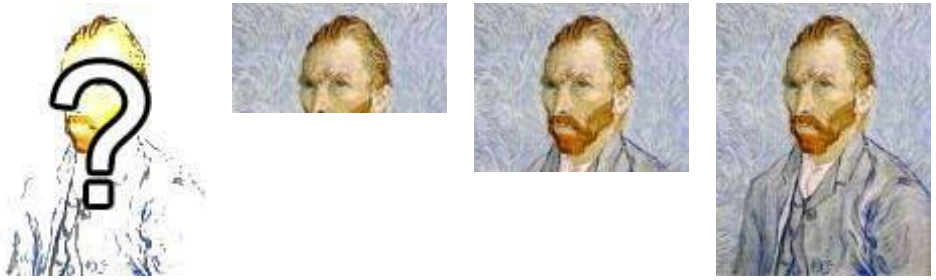
- Basado en reuso. Los sistemas se integran partiendo de componentes existentes o adquiridos.
- Etapas del proceso



Iterativo / Incremental

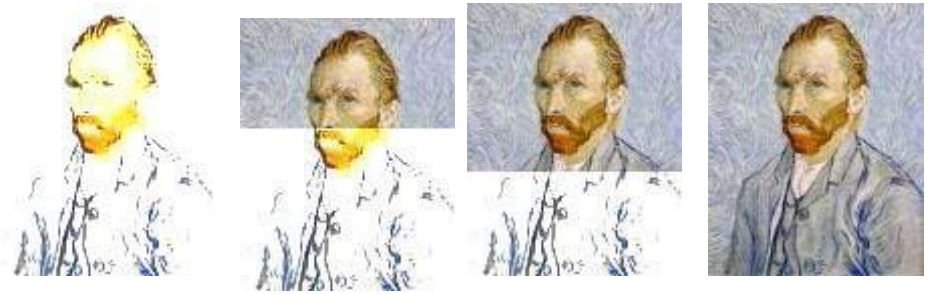


Iterativo / Incremental



Incremental

Iterativo



Incremental & **Iterativo**

Workflows centrales

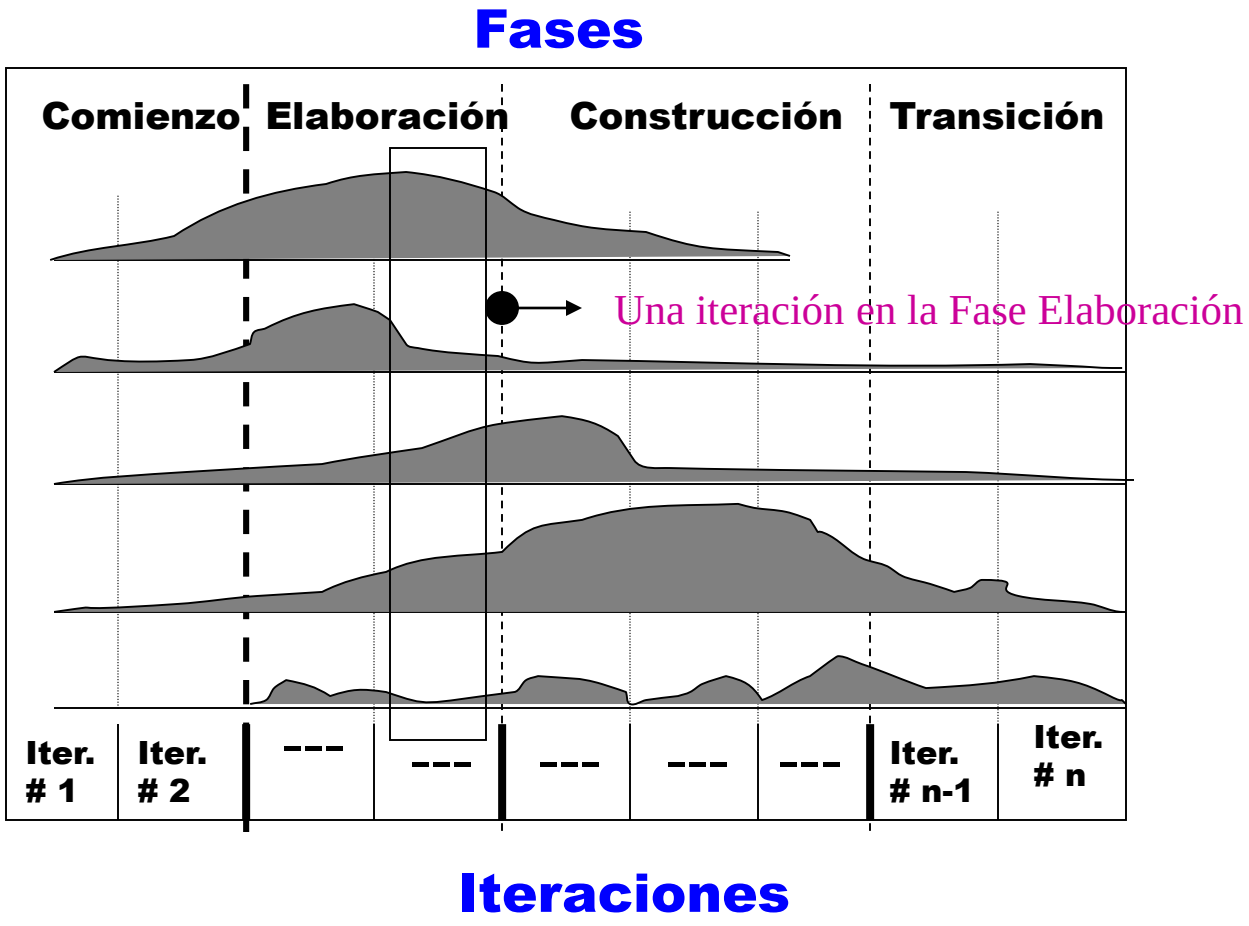
Requerimientos

Análisis

Diseño

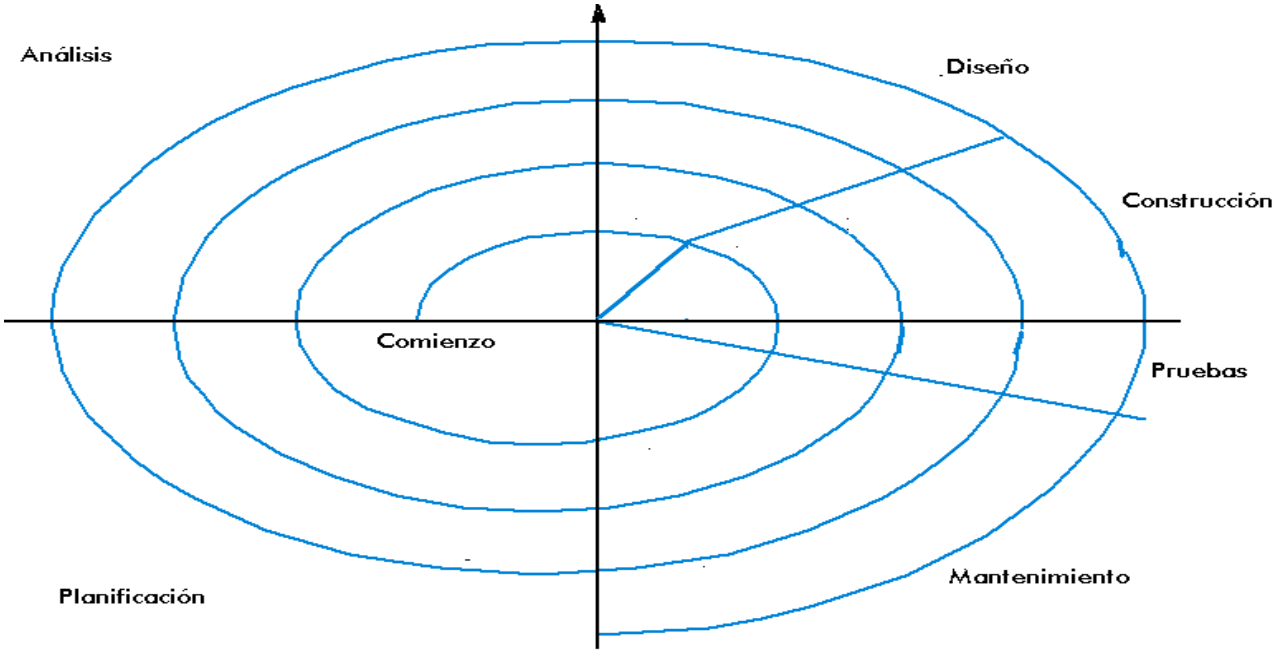
Implementación

Prueba



Modelo de Desarrollo en Espiral

- El Proceso de representa como una espiral, en lugar de una secuencia de actividades.
- Cada vuelta de la espiral representa una fase del proceso.
- No hay fases fijas – se seleccionan vueltas en la espiral segun sea requerido.
- Los riesgos se evalúan y resuelven a través de todo el proceso.



Manufactura Predecible	Desarrollo de Nuevos Productos
Es posible terminar primero las especificaciones de forma completa, y luego construir.	Rara vez es posible crear detalladas especificaciones, anticipadamente y sin cambios.
En etapas tempranas se puede estimar de manera confiable, esfuerzo y costo.	No es posible estimar. Conforme emergen los datos empíricos, es posible ir planeando y estimando incrementalmente.
Es posible identificar, definir, programar y ordenar todas las actividades detalladas.	Cerca del inicio no es posible. Se requieren pasos adaptativos conducidos por la retroalimentación de la construcción.
La adaptación a cambios impredecibles no es la norma, y los ratios de cambio son relativamente bajos.	La adaptación a cambios impredecibles es la norma. Los ratios de cambio son altos.

- Las Metodologías Ágiles valoran:
  - Al individuo y las interacciones en el equipo de desarrollo más que a las actividades y las herramientas
  - La colaboración con el cliente más que la negociación de un contrato
  - Responder a los cambios más que seguir estrictamente una planificación

Clásico	Ágil
Requisitos detallados	Visión general del producto
Planificación estricta	Adaptación a la situación
Requerimientos no cambiantes	Evolución constante
Seguimiento y Control	Autogestión
División y especialización	Equipo multi- disciplinar

Principios de los Procesos Ágiles

1. La mayor prioridad es satisfacer al cliente con entregas continuas y tempranas de software valioso.
2. Los cambios en los requerimientos son bienvenidos, aun en etapas avanzadas del desarrollo.
3. Entregas de software funcionando, frecuentemente, desde un par de semanas a un par de meses, con preferencia a escalas de tiempo menores.
4. La gente de negocio y los desarrolladores trabajan juntos diariamente a lo largo del proyecto.
5. La construcción de proyectos con individuos motivados. Hay que ofrecerles el ambiente y soporte que ellos necesitan y confiar en que harán el trabajo.
6. El método más eficiente y efectivo de comunicación de información a y con el equipo de desarrollo es la conversación cara a cara.
7. El software funcionando es la principal medida de progreso.
8. Los procesos ágiles promueven desarrollo sostenible.
9. Sponsors, desarrolladores y usuarios deberían poder mantener pasos constantes, indefinidamente.
10. La atención continua a la excelencia tecnológica y al buen diseño mejoran la agilidad.
11. La simplicidad, el arte de maximizar la cantidad de trabajo no hecho, es esencial.
12. Las mejores arquitecturas, requerimientos y diseños emergen de equipos que se organizan ellos mismos.
13. En intervalos regulares, el equipo refleja como puede volverse más eficiente, entonces modifican y ajustan su comportamiento de manera acorde a esto.

Desarrollos Ágiles o Livianos

- Los métodos ágiles son un subconjunto de los métodos iterativos.
- El modelado ágil no es un proceso completo o un método ágil, es un conjunto de principios y prácticas para modelado y análisis de requerimientos, que complementa a la mayoría de los métodos de desarrollo iterativos e incrementales.
- Algunos métodos ágiles:
  - **SCRUM**
  - **XP**
  - **Métodos Cystal**
  - **Desarrollo de Software Adaptativos (ASD)**
  - **Modelo Dinámico de Entrega de Soluciones (DSDM)**
  - **Desarrollo Conducido por Características (FDD)**

Actividad N 2.1

- Conteste a la siguiente pregunta (en no más de cinco líneas): Qué metodologías (modelos) de ingeniería de software de los presentados en el curso piensa que sería el más eficaz? Por qué?
- Elija dos metodologías (o modelos) de desarrollo de software y, usando una tabla, compare las mismas eligiendo criterios diferentes. Cada comparación no debe exceder más de 2 líneas.

- **Requisitos a evaluar**
  - Actividad Grupal
  - Vencimiento 7días