Universidad Nacional de Río Cuarto
Facultad de Ciencias Exactas, Físico-Químicas y Naturales
www.exa.unrc.edu.ar

Departamento de Computación
http://dc.exa.unrc.edu.ar

INTRODUCCION

ANALISIS Y DISEÑO DE SISTEMAS (CÓD 3303)

Mg. Marcela Daniele

marcela@dc.exa.unrc.edu.ar

Tercer año

Analista en Computación,
Profesorado y Licenciatura en Ciencias de la Computación

AÑO 2021

Análisis y Diseño de Sistemas (3303)

- Ingeniería de Software: Introducción y Conceptos
- Características del Software
- Modelos de Desarrollo de Software
- Bibliografía Consultada

31 de marzo de 2021

Ingeniería……

- Disciplina basada en un *conjunto de conocimientos técnicos y científicos*, dedicados a la *innovación, invención, diseño, desarrollo, y optimización* de técnicas, sistemas, procesos y herramientas.
  - *Transforma el conocimiento en soluciones prácticas para beneficio de la humanidad*.
  - Ofrece soluciones a problemas sociales, económicos e industriales.
  - Dispone de técnicas formales y estándares para ser aplicadas a la resolución de los problemas.

El desarrollo de software es una actividad de ingeniería

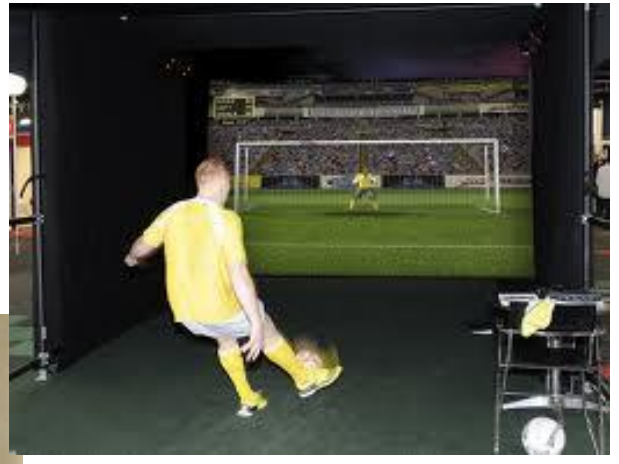
El Software es:

- Es inmaterial
- Un pequeño **error** puede causar un gran efecto
- Se modifica fácilmente
- Sufre permanentes **cambios**
- No resulta fácil **medir** ni **estimar** tiempo, costo, esfuerzo

* El propósito del desarrollo de software es **la producción eficaz y eficiente de un producto software que resuelva el problema y satisfaga los requisitos del cliente**.

* Es un proceso intelectual, afectado por la creatividad, criterio y formación de los involucrados.

* Construir software es una actividad de ingeniería que presenta mayor dificultad para encontrar estándares.



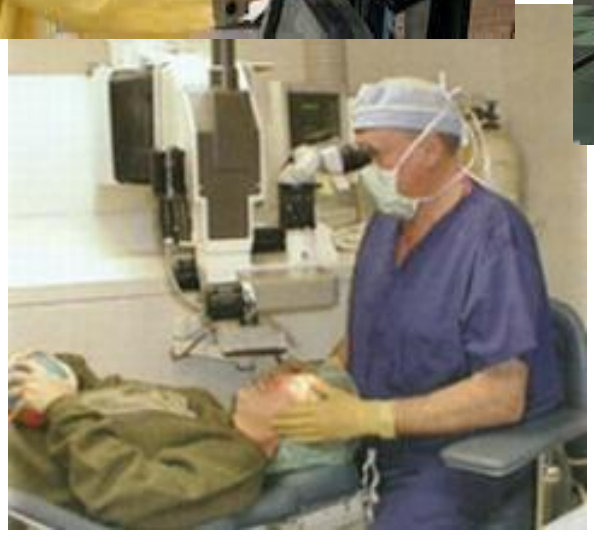


Grafico 2 Distribución por Grupos etarios

Donde se aplica el Software ?

El software puede aplicarse a numerosas y diversas situaciones del mundo real

- ✓ problemas con un conjunto específico de acciones que lleven a su resolución,
- ✓ problemas que se describen formalmente indicando la solución deseada,
- ✓ problemas que el humano resuelve utilizando una multitud de reglas heurísticas
- ✓ problemas de los que no se tiene una idea clara de cómo resolver, pero se conoce alguna solución apropiada para algunos ejemplos de los datos de entrada.

Software de base, Software de tiempo real, Software de gestión, Software científico y de ingeniería, Software embebido, Software de inteligencia artificial, Software Educativo, webapps, Sistemas Colaborativos, ……

Por qué es necesario el Análisis y Diseño de Sistemas

- El análisis y diseño de sistemas tiene el propósito de *analizar sistemáticamente el flujo de datos de entrada, procesarlos y producir la salida* de información esperada.

- El análisis de sistemas estudia, diseña e implementa mejoras para dar soluciones a un problema planteado.

- La implementación de un sistema sin una planificación adecuada lleva a grandes fracasos.

La necesidad del Análisis y Diseño de Sistemas

Como cualquier disciplina,

la **ingeniería de software** requiere de compromiso,

y el ingeniero de software debe tomar decisiones

acertadas respecto a los recursos que utilizará, en

cuanto a software, hardware, personas y herramientas,

dependiendo de cada problema particular.

Ingeniería de Software

La Ingeniería de Software es el campo de las Ciencias de la Computación que se dedica al estudio de la construcción de sistemas de software, de mediano y gran tamaño, y que requieren de uno o más equipos de ingenieros de software para su desarrollo.

El ***Ingeniero de Software*** construye, modifica o reutiliza componentes de software (CS) para ser combinados con otros CS, realizados por otros ingenieros o equipos de ingenieros.

El ***Programador*** escribe programas correctos, eficientes y completos.

Cómo surge la Ingeniería de Software?

✓ Los primeros programas se escribían en un lenguaje entendible por la computadora. Usuario + Máquina.

✓ Se evolucionó, el usuario daba su especificación al programador para que escribiera sus programas. Se obtenían buenos resultados.

El software se en base a experiencias previas, ensayo-error..

✓ Con el surgimiento de proyectos de software más grandes y complejos, el avance del HW y SW, y su aplicación en diversos contextos, llevó a:

- software de baja calidad
- incumplimiento de tiempos y de presupuestos
- incrementos dramáticos en los costos de mantenimiento.

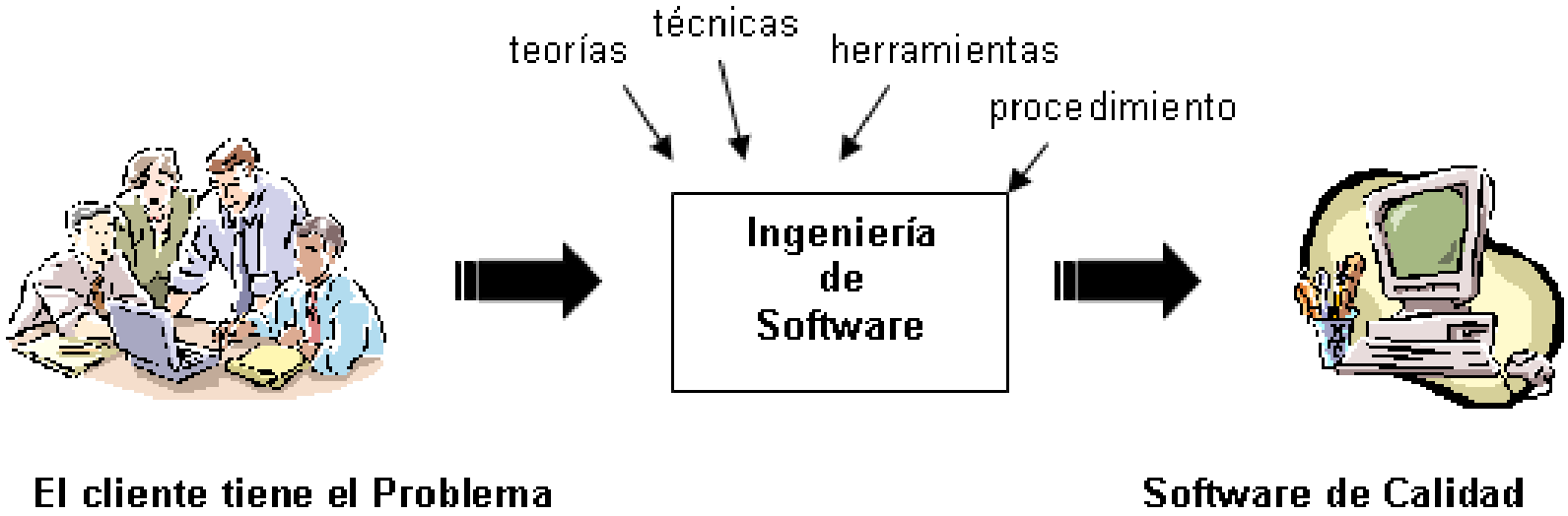
✓ A fines de los 60's se produce la denominada Crisis del Software.

✓ En los 70's surgen métodos estructurados (Jourdon, DeMarco).

✓ En los 80's y 90's el paradigma estructurado evolucionó hacia otros paradigmas, como los orientados a objetos (RUP, Diseño por contratos).

✓ En los 90 se da la masificación de Internet, y con ello mayor demanda en la celeridad del desarrollo de software. Surgen los Métodos ágiles con respuestas más rápidas.

Ingeniería de Software



*La **Ingeniería de Software** es una disciplina que se basa en un **Proceso** y aplica **Métodos** y **Herramientas** Y obtiene como resultado un **Producto de Software** confiable y que funcione eficientemente en máquinas reales con costos razonables.*

Conceptos- Desarrollo de Software

- **PROCESO o MODELO DE DESARROLLO DE SW:** conjunto de actividades y resultados que producen un **Producto de Software**. Define un marco de trabajo. Define la secuencia de etapas o pasos a seguir.

- **METODOS:** indican "cómo" construir técnicamente el SW.

- **HERRAMIENTAS:** brindan un soporte automático al Proceso y a los Métodos.

- **PRODUCTO DE SW:** sistema que se realiza para un usuario con la documentación correspondiente. Genérico o a medida.

- **ATRIBUTOS DEL PRODUCTO DE SW:** características que distinguen y describen al producto.

Capas de la Ingeniería de Software

El rol del Ingeniero de Software

Para pequeños o medianos proyectos de software **("programming-in-the-small")**

- Requiere educación y experiencia.
- Deber ser buen programador, independiente del lenguaje de programación, con sólida formación en estructura de datos y algoritmos.

El rol del Ingeniero de Software

Para proyectos de software más grandes y más complejos (**"programming-in-the-large"**), requerirá además:

- Conocer distintas métodos de modelado y diseño.
- Comprender y traducir los requerimientos del usuario en especificaciones precisas.
- Habilidad para definir e interpretar los distintos niveles de abstracción en los diferentes estados del proyecto.
- Construir modelos correctos, completos y precisos.
- Buena comunicación con los miembros del equipo.
- Capacitación p/planificar tareas y manejar equipos de trabajo.
- Actualmente, el IS se especializa en áreas específicas.

Características de Calidad del SW (I)

- **Exactitud (Correctness):** habilidad del SW para ejecutar sus tareas correctamente de acuerdo a la especificación.
- **Robustez (Robustness):** reacción apropiada a condiciones anormales. Complementa la exactitud. Normal y anormal relativos a la especificación.
- **Extensibilidad (Extensibility)**: facilidad del software de adaptar los cambios a la especificación.
- **Reusabilidad (Reusability):** habilidad de los elementos de software para ser usados en aplicaciones diferentes.
- **Compatibilidad (Compatibility):** facilidad de combinación entre elementos de software.

Características de Calidad del SW (II)

- **Eficiencia (Efficiency):** habilidad de un sistema para hacer la menor cantidad de demandas posibles a los recursos de hardware, como el tiempo de procesador, la cantidad de memoria interna y externa.

- **Portabilidad (Portability):** facilidad de transferencia del software.

- **Facilidad de uso (Ease to use):** facilidad con la que las personas pueden usar un producto de software y aplicarlo para resolver problemas. Facilidad de instalación.

- **Funcionalidad (Functionality):** es la extensión de las posibilidades que provee un sistema.

- **Oportuno (Timeliness):** habilidad del software de estar resuelto cuando (o antes) se acordó con los usuarios. Es una de las grandes frustraciones de la industria del software.

Características de Calidad del SW (III)

- **Verificable (Verifiability):** procedimientos de aceptación, prueba de datos y procedimientos para detectar fallas.
- **Integridad (Integrity):** proteger a sus programas y datos de accesos y modificaciones no autorizadas.
- **Reparable (Repairability):** reparar defectos con facilidad.
- **Economía (Economy):** habilidad de ser completado de acuerdo al presupuesto pactado. Relacionado con el tiempo.
- **Documentación**: para que los usuarios comprendan las funcionalidades del sistema, para que los desarrolladores comprendan la estructura e implementación de un sistema.
- **Acerca del Mantenimiento del Software:** modificaciones por cambios en requerimientos del usuario, formato de los datos, documentación, Hardware, cuestiones externas, corrección de errores.

Actividades de un Proceso de desarrollo de software

Actividades Estructurales del Ciclo de Vida

Comunicación y Análisis del problema a resolver

Planeación

Modelado y diseño de la solución propuesta

Construcción o Implementación

Instalación o Despliegue

Actividades de Gestión

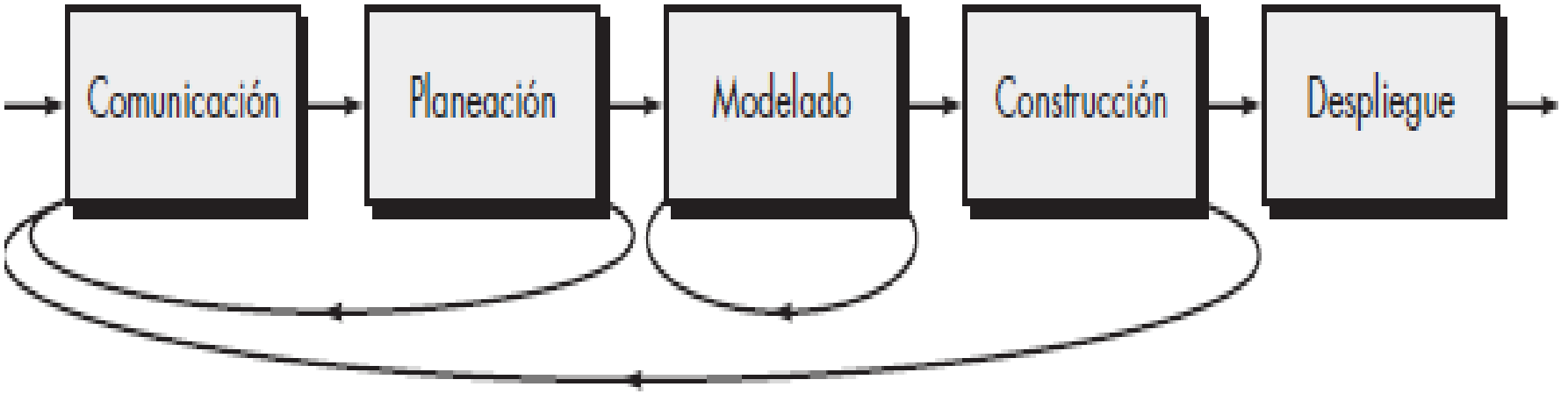
Seguimiento y control del proyecto, Administración de Riesgos, Aseguramiento de la Calidad, Medición, Gestión de Cambios, Administrar la reutilización, Revisiones Técnicas

La Práctica de la Ingeniería de Software

1. **Entender el problema (comunicación y análisis)**
   - *¿Quiénes son los participantes?*
   - *¿Cuáles datos, funciones y características se requieren para resolverlo?*
   - *¿Puede fraccionarse? ¿Es posible representarlo gráficamente?*
2. **Planear la solución (modelado y diseño del software)**
   - *¿Ha resuelto problemas similares? ¿son reutilizables sus elementos?*
   - *¿Algún software existente implementa características, datos y funciones que se requieren? ¿Pueden definirse problemas más pequeños?*
3. **Ejecutar el plan (generación del código)**
   - *¿El código fuente se corresponde con el modelo del diseño?*
   - *¿Cada parte componente de la solución es correcta? ¿El diseño y código se han revisado?*
4. **Examinar la exactitud del resultado (garantizar calidad)**
   - *¿Puede probarse cada parte componente de la solución? ¿Se ha implementado una estrategia* razonable para hacer pruebas?
   - *¿La solución produce resultados respecto de los datos, funciones y características que se requieren? ¿El software se ha validado contra todos los requerimientos de los participantes?*

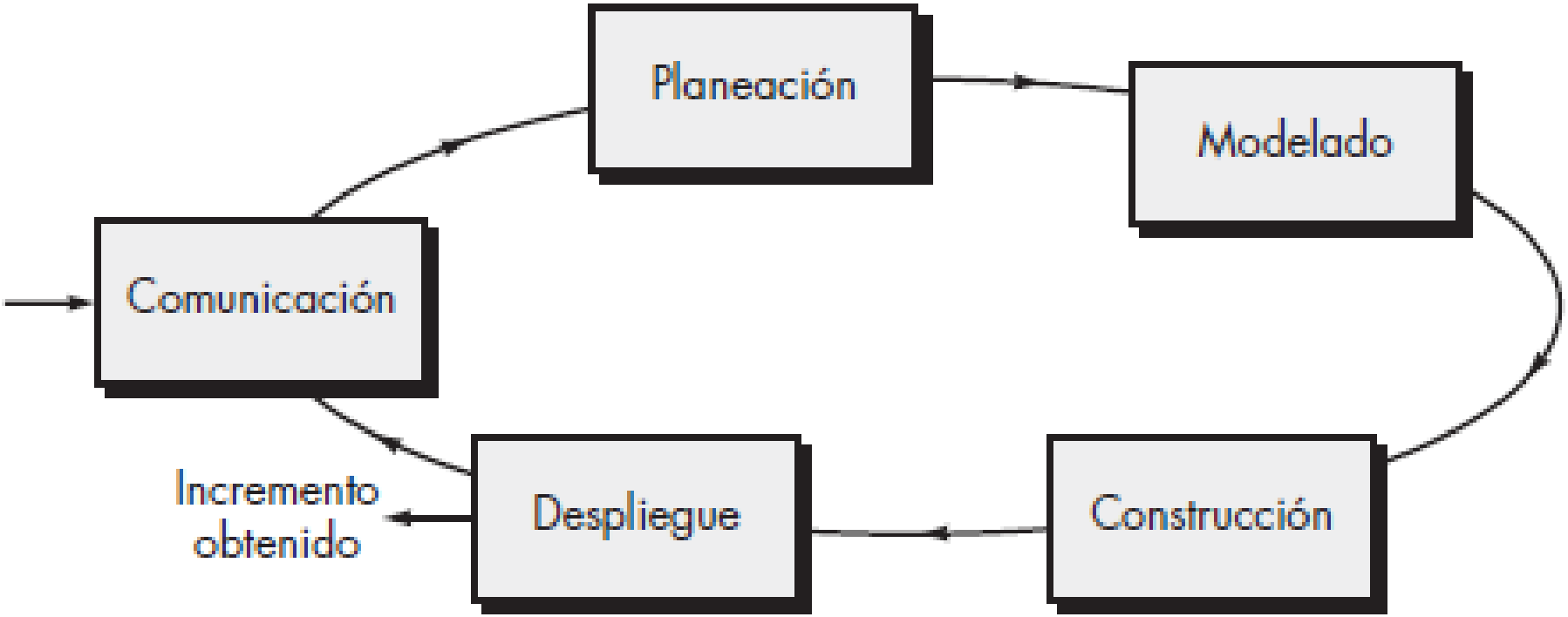
Flujos de Proceso

a) Flujo de proceso lineal



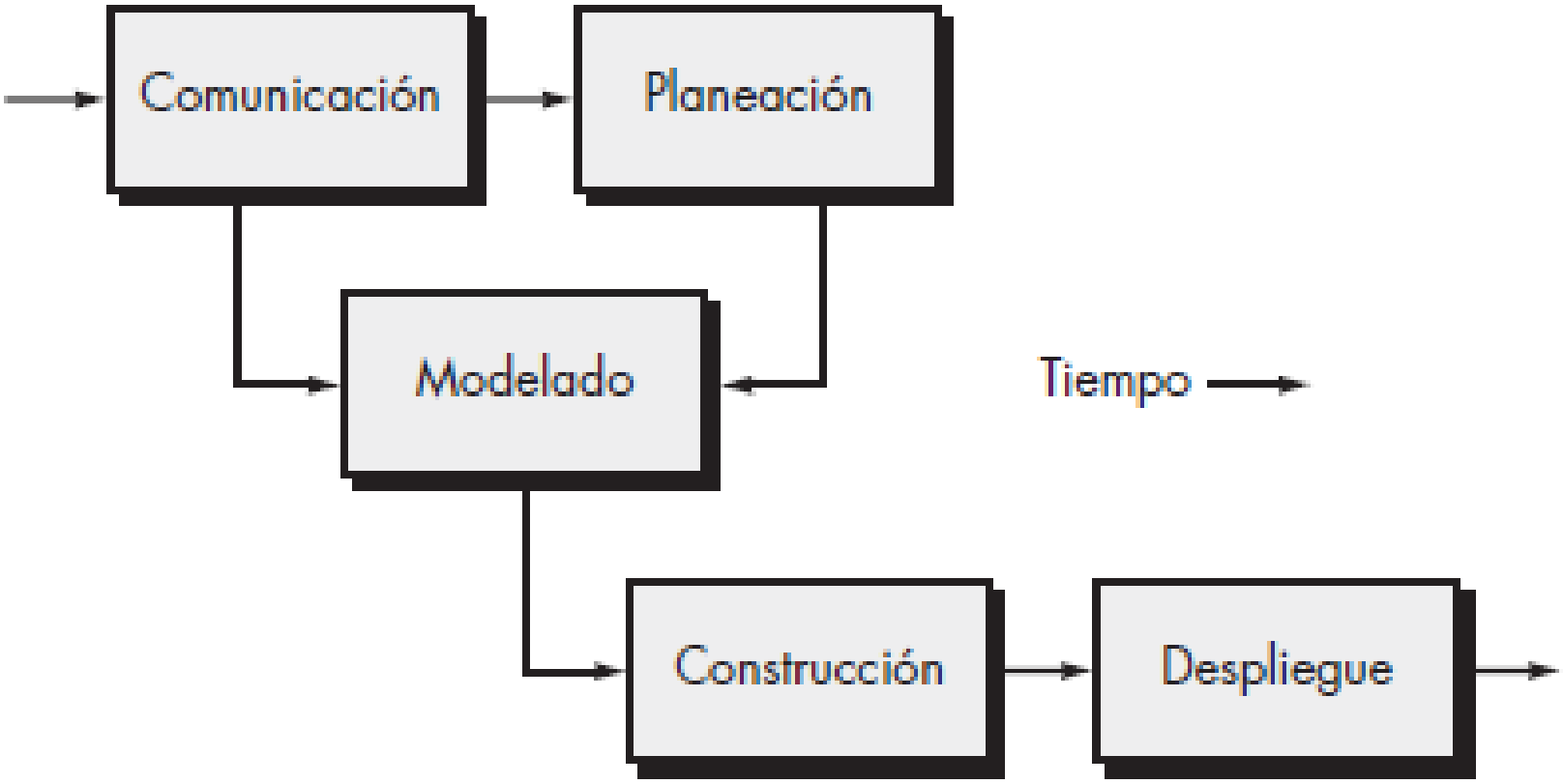
b) Flujo de proceso iterativo

Flujos de Proceso



c) Flujo de proceso evolutivo

Flujos de Proceso



d) Flujo de proceso paralelo

Metodologías Tradicionales secuenciales y evolutivas y Metodologías Ágiles

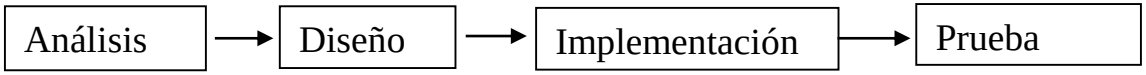
- Las **metodologías tradicionales** se centran en el control del proceso, mediante una rigurosa definición de roles, actividades, artefactos, herramientas y notaciones para el modelado. Incluyen una documentación detallada. Ej: Proceso Unificado.
- Las **metodologías ágiles**, dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones cortas y respuestas rápidas. Poca documentación. Ej: SCRUM, XP (eXtreme Programming), Kanbam, Crystal Methodologies, FDD (Feature-Driven Development), DSDM (Dynamic Systems Development Method).

Algunos Modelos para al desarrollo de SW

- MODELO LINEAL SECUENCIAL O EN CASCADA.
- MODELO DE CONSTRUCCION DE PROTOTIPOS.
- MODELO EN ESPIRAL.
- ENSAMBLAJE DE COMPONENTES.
- MODELO DE LOS METODOS FORMALES.
- DISEÑO POR CONTRATOS.
- El PROCESO UNIFICADO
- SCRUM

MODELO LINEAL SECUENCIAL O EN CASCADA

- Enfoque sistemático.
- Se Presenta con actividades por separado:
  1. Especificación de requerimientos – Análisis
  2. Diseño
  3. Implementación – Código
  4. Testeo - Prueba



- Es el paradigma más antiguo y más utilizado.
- Podría resultar útil cuando deben hacerse adaptaciones o mejoras bien definidas a un sistema ya existente.

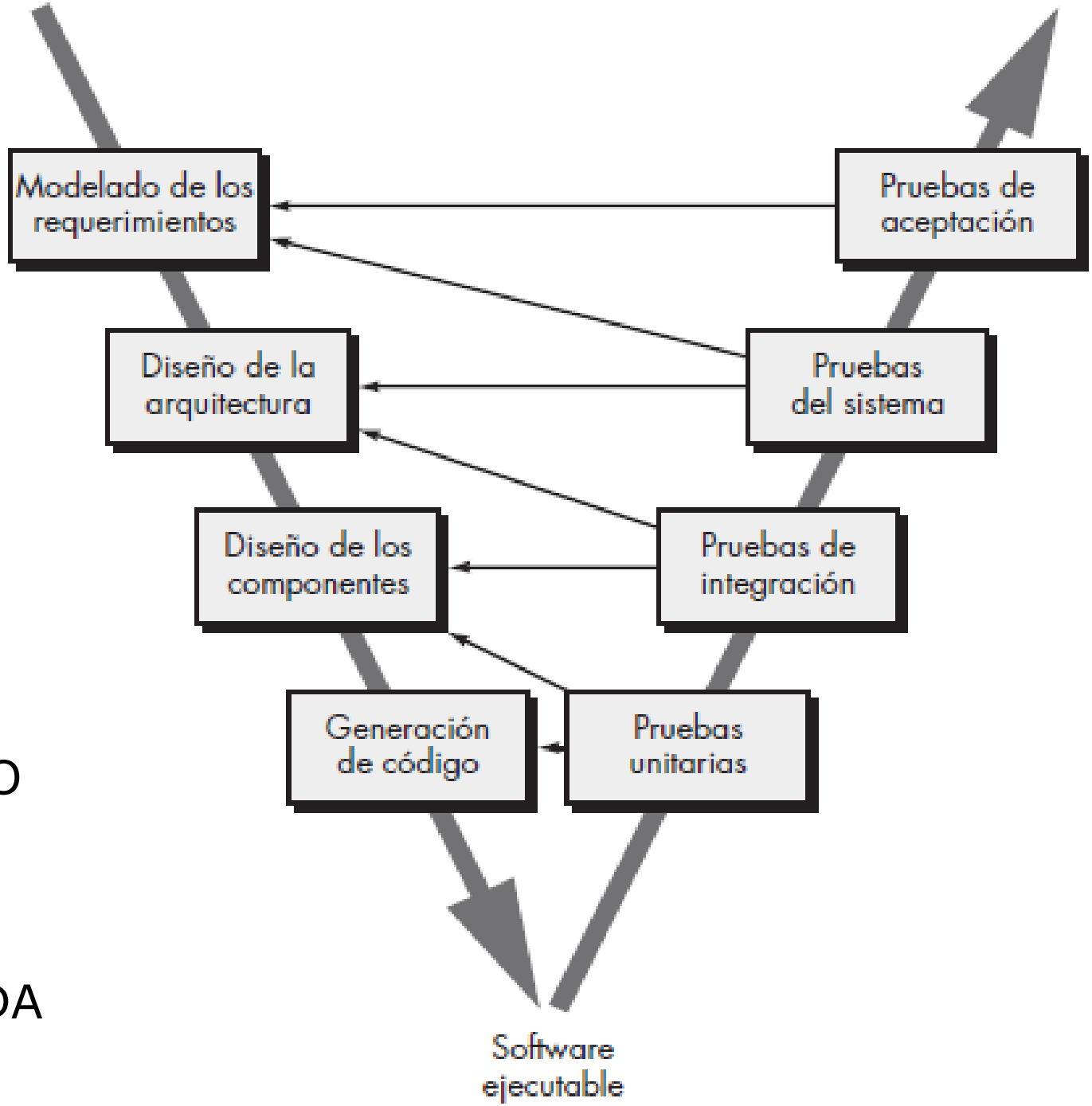
Posibles Problemas:

- Los proyectos rara vez siguen un modelo secuencial.
- Es difícil que el cliente exponga todas las necesidades al comienzo.
- Se requiere de mucha paciencia del cliente.
- Un error detectado en el programa podría ser catastrófico.
- Es difícil respetar los tiempos.

MODELO

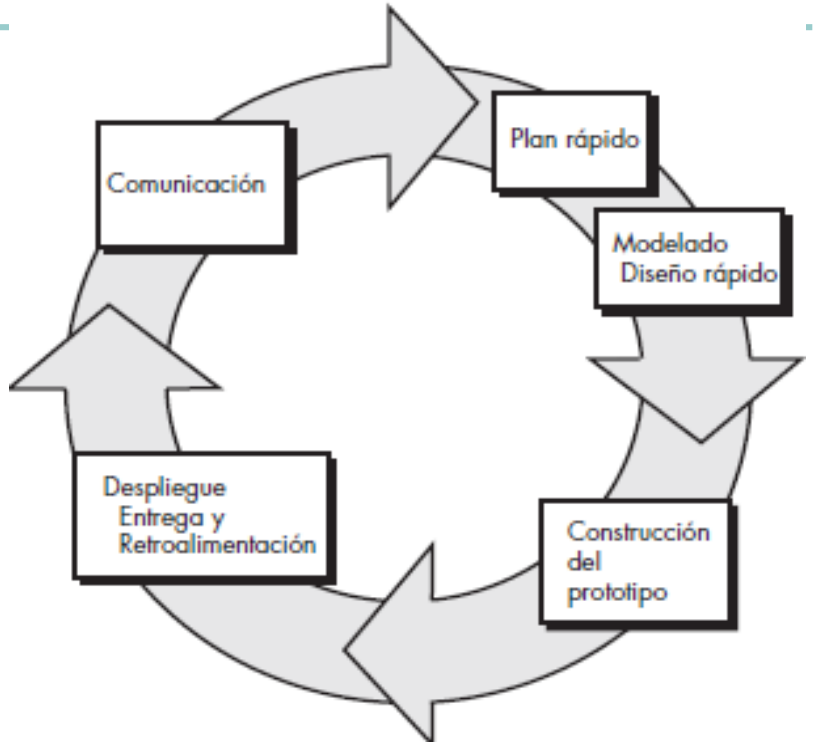
EN

CASCADA



MODELO DE CONSTRUCCION DE PROTOTIPOS

Se hace una versión rápida, se implementa, y luego se va aplicando refinamiento hasta llegar a un sistema robusto y mantenible.
Las etapas se van entrelazando.

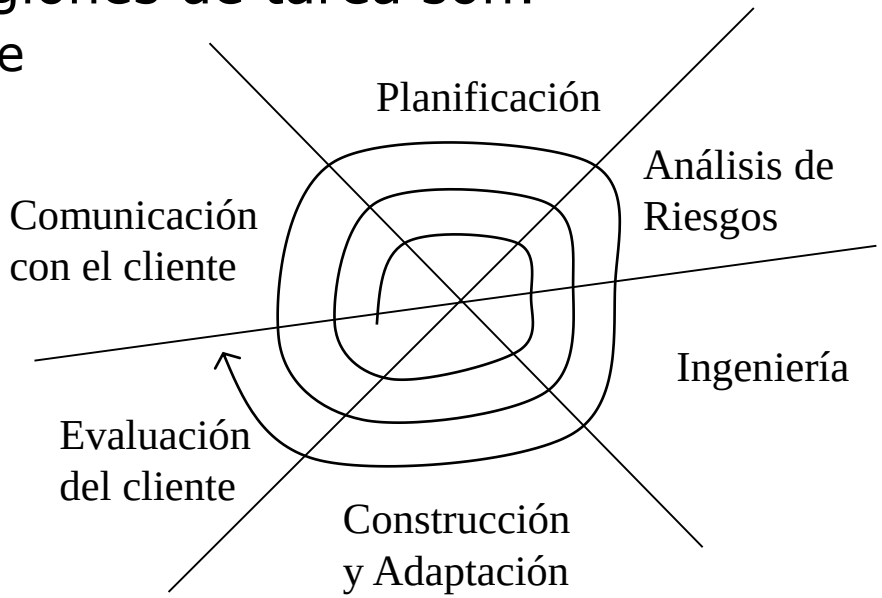


Posibles Problemas:

- El cliente pide pequeños ajustes al prototipo y quiere comenzar a operar inmediatamente con el sistema.
- Para el desarrollador, hacer que el sistema funcione rápidamente lo puede llevar a realizar malas selecciones.
- Aunque desechar el prototipo en algunas es imposible, puede servir cuando es un proceso.

MODELO EN ESPIRAL

- Los espacios de trabajo que componen el proceso de desarrollo lineal se ejecutan de manera iterativa.
- En cada iteración se produce una nueva versión más compleja y completa del sistema.
- Es un modelo de proceso de software evolutivo.
- Los espacios de trabajo o regiones de tarea son:
  1. Comunicación con el cliente
  2. Planificación
  3. Análisis de riesgos
  4. Ingeniería
  5. Construcción y adaptación
  6. Evaluación del Cliente



Posibles Problemas:

- Poco utilizado.
- Difícil convencer al cliente de un desarrollo evolutivo y controlable.
- Requiere de mucha habilidad para descubrir riesgos potenciales.

ENSAMBLAJE DE COMPONENTES

- Basado en la Tecnología de Objetos.
- Es un proceso evolutivo que también usa una espiral.
  - Se crean clases que incluyen tanto datos como los algoritmos para manejar esos datos.
  - Las clases creadas se almacenan en un repositorio o depósito o biblioteca de clases o componentes.
  - En base a los requerimientos de los usuarios, se seleccionan los componentes del repositorio y se ensamblan entre si para producir el sistema final.

Posibles Problemas:

- Complejidad para construir y mantener el repositorio.
- Los componentes deben ser catalogados y clasificados con criterios bien definidos para su organización.

MODELO DE LOS METODOS FORMALES

Se produce una especificación matemática formal del sistema y se van aplicando transformaciones a esa especificación preservando la correctitud.

Posibles Problemas:

- Es un proceso más costoso
- Consume mucho tiempo
- Requiere de profesionales expertos
- No favorece la comunicación con el cliente

- Los métodos formales se utilizan en combinación con otros métodos.

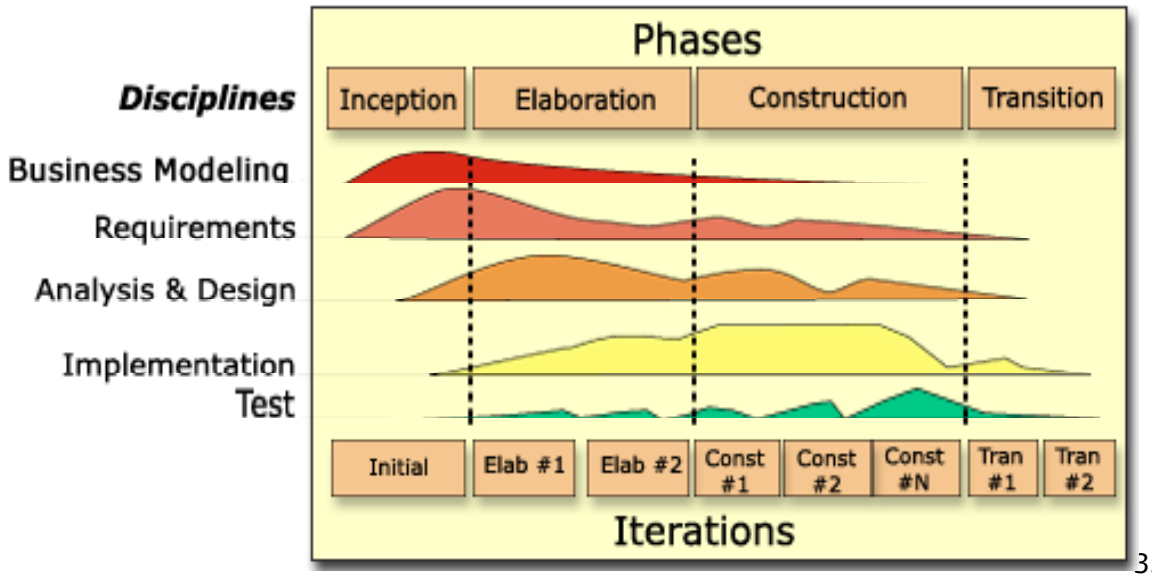
- En sistemas con funcionalidades críticas son una alternativa que asegura un producto de calidad.

DISEÑO POR CONTRATOS (Meyer)

- Metodología de diseño de software OO, basada en *Contratos*.
- El diseño por *Contratos*, tiene sus raíces en los métodos formales para la construcción de software.
- El sistema es visto como un conjunto de *elementos de software* que cooperan entre si.
- Los elementos juegan un rol principal: *Proveedores* o *Clientes*.
- La cooperación establece claramente *obligaciones* y *beneficios*, y su especificación son los *Contratos*.
- Los *Contratos* especifican expresiones lógicas o *aserciones*.
  - Precondición: requisitos para que una rutina sea aplicable.
  - Poscondición: propiedades garantizadas a la salida de una rutina.
  - Invariante: expresa las restricciones semánticas. Se añade implícitamente a cada pre o poscondición.
  - Pre + Poscondición = Contrato entre la clase y sus clientes.

PROCESO UNIFICADO

- Es una metodología de desarrollo de software OO.
- **Basado en Casos de Uso**
  - Caso de Uso: funcionalidad del sistema que da un resultado de valor para un usuario.
- **Centrado en la Arquitectura**
  - Arquitectura: forma del sistema. Diferentes vistas en las distintas etapas del proyecto.
- **Iterativo e Incremental**
  - Iteración: dividir el proyecto en miniproyectos. Pasos en el flujo de trabajo.
  - Incremento: crecimiento del producto.



METODOLOGIA AGIL: SCRUM

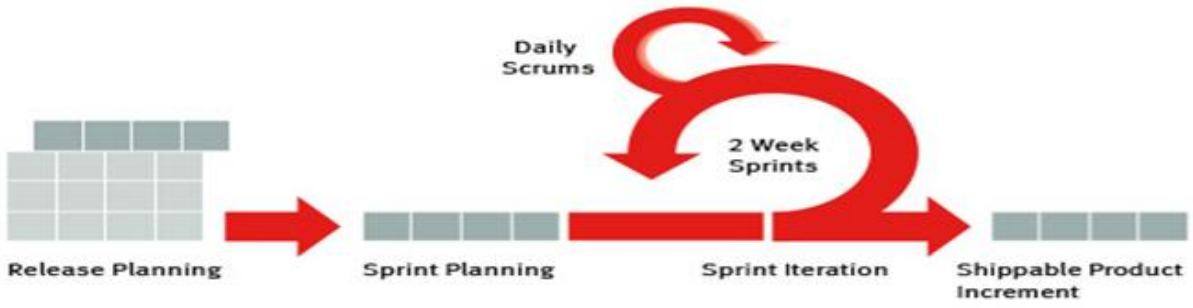
- Basado en un desarrollo iterativo e incremental.
- 2001 se escribió el Manifiesto Ágil.
- Entornos complejos y flexibles que exigen rapidez en los resultados. Con requisitos muy cambiantes.
- Trabajo en equipo. Equipos autocontrolados y pequeños (5 a 9). Comunicación verbal. Reuniones cortas y diarias.
- Sprint (iteraciones cortas). Entrega de un producto funcional al finalizar cada sprint, vistas diarias del proyecto.
- Los Equipos están altamente integrados y comprometidos

Posibles condicionantes:

- El cliente debe estar muy involucrado, de forma activa y continua.
- Escasa documentación.
- Puede resultar más difícil de aplicar en grandes proyectos .
- Si una tarea no está bien definida incrementa costos y tiempos.
- Un equipo poco comprometido podría llevar al fracaso del proyecto

METODOLOGIA AGIL: SCRUM

- **Sprint**: iteraciones de corta duración (1-4 semanas). Entrega de un producto funcional al finalizar cada sprint, vistas diarias del proyecto.
- Se elabora un **plan de proyecto** a partir de la **lista de requisitos**
  - El **cliente prioriza los requisitos**, según valor y costo, y se determinan los sprints, junto al equipo que la realizará.
  - El **equipo planifica la iteración**, elabora la lista de tareas a realizar para cumplir con el sprint que se comprometió.
- **Sprint Meeting**: reunión diaria del equipo (15 min), para supervisar el trabajo realizado y realizar adaptaciones si es necesario.
- **Scrum Master:** un miembro del equipo asume este rol, controla y está atento a problemas que pueden presentarse.
- El Sprint finaliza con la **demostración** y la **retrospectiva**.



Bibliografía Consultada

- Capítulo 1:"Fundamentals of Software Engineering". Carlo Ghezzi.
- Capítulo 1, 2: Software Engineering A Practitioner's Approach. Roger Pressman.
- Capítulo 1, 2: Ingeniería de Software. Ian Sommerville.
- Capítulo 1: "Object Oriented Software Construction". Bertrand Meyer.
- www.agilemanifesto.org - http://www.agile-spain.com/