Tema 1: Introducción a la Ingeniería del Software

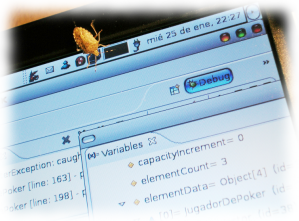
BLOQUE I:
INTRODUCCIÓN Y PARADIGMAS DE DESARROLLO EN INGENIERÍA DEL SOFTWARE
Segundo curso de Grado en Ingeniería Informática

Curso 2014-2015

Javier Sánchez Monedero
http://www.uco.es/users/i02samoj

5 de enero de 2015

Índice



Índice

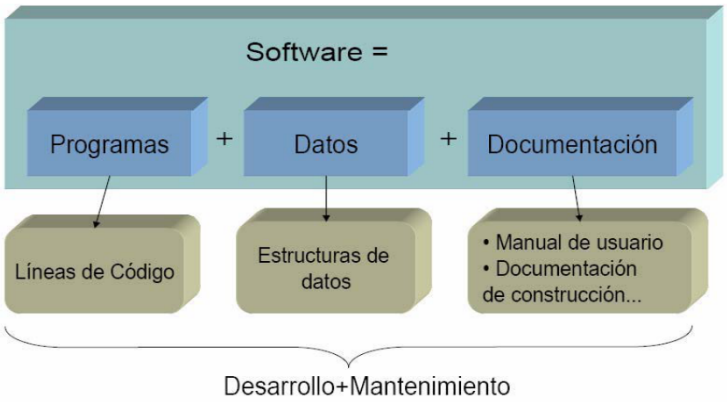


Figura 1: Composición del software Ruiz [2013]

1.

"*Meses de programación pueden ahorrarte horas de diseño*"

2. ¿Qué es la Ingeniería del Software?

¿Qué es la Ingeniería del Software?

¿Qué es el software?

Antes de empezar con el estudio de la ingeniería del software, debemos ponernos en contexto y definir qué es el software, qué actividades relacionadas con el software cubre la ingeniería del software y cómo hemos llegado, como industria, a la conclusión de que la ingeniería es el enfoque adecuado para estas actividades .

Software

Denominamos *software* a todo aquello intangible (no físico) que hay en un ordenador, incluyendo el conjunto de programas informáticos que indican la secuencia de instrucciones que un ordenador debe ejecutar durante su funcionamiento (también denominado código) y el resto de los datos que este ordenador manipula y almacena Pradel i Miquel and Martos [2010].

Hardware

Por oposición, denominamos *hardware* al conjunto de componentes físicos de un ordenador. Este hardware ofrece una serie de instrucciones que el ordenador es capaz de ejecutar cuando ejecuta un programa Pradel i Miquel and Martos [2010].

Nota: los programas incluyen bibliotecas externas, como mínimo las que lo relacionan con el sistema operativo

El desarrollo software

El desarrollo de software es el acto de producir o crear software.

A pesar de que el desarrollo de software incluye la programación (la creación del código fuente), usamos el término ***desarrollo de software*** de una manera más amplia para referirnos al conjunto de actividades que nos llevan desde una determinada idea sobre lo que queremos hasta el resultado final del software.

Actividades del desarrollo:

- Programación
- Compilación
- El estudio y la documentación de las necesidades de los usuarios
- El mantenimiento del software una vez se empieza a usar
- La coordinación del trabajo en equipo de las diferentes personas que intervienen en el desarrollo
- La redacción de manuales y ayudas de uso para los usuarios
- Verificación
- etc.

Como en otras áreas, al crecer la complejidad, y el coste e impacto del software, se empiezan a aplicar **técnicas de ingeniería** al desarrollo software

El software presenta una serie de **peculiaridades** (en comparación con otros *productos*):

- El producto software es enteramente **conceptual**.
- No tiene propiedades físicas como peso, color o voltaje, y, en consecuencia no está sujeto a leyes físicas o eléctricas.
- Su naturaleza conceptual crea una **distancia intelectual** entre el software y el problema que el software resuelve.
- Difícil para una persona que entiende el problema entender el sistema software que lo resuelve.
- Para probar es necesario disponer de un sistema físico.
- El mantenimiento no es sólo una substitución de componentes.

Hoy en día, aún existe debate sobre si el desarrollo de software es un arte/artesanía o una disciplina de ingeniería. Obviamente asumiremos lo segundo. En todo caso, **desarrollar software de calidad** con el mínimo coste posible ha resultado ser una **actividad**, en general, muy **compleja**.

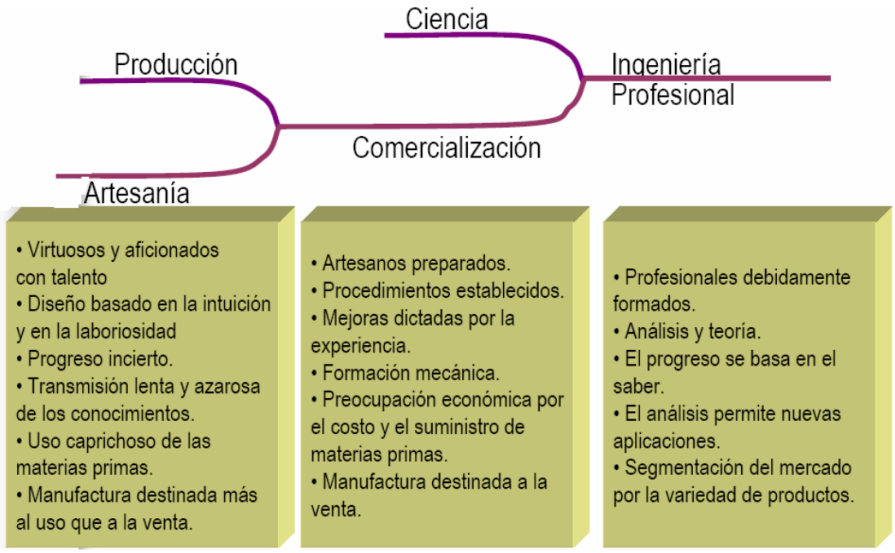


Figura 2: Evolución del Software hacia la Ingeniería Ruiz [2013]

Ámbito de la ingeniería del software

Inicio de la informática
Grandes ordenadores como el ENIAC dedicados a cálculos matemáticos (*computadoras*)

Actualidad
Ordenadores en todas partes cada vez más pequeños y con más potencia (automóviles, relojes...)

Según el ámbito las **diferencias** son **abismales**:

- Red social: actualizar el software consiste en desplegar una nueva versión en una red de servidores
- Sistema de navegación de un coche: actualizar el software requiere el coste de llevar todos los coches a talleres

A grandes rasgos, estos son ámbitos del software, que puede pertenecer a más de una categoría:

- **Software de sistemas**. Dan servicio a otros programas. **No** tienen un **propósito específico**. Por ejemplo: un servidor de bases de datos no se diseña pensando si almacenará los resultados de la Quiniela o los datos de un hospital. Son programas escritos para dar servicio a otros programas,

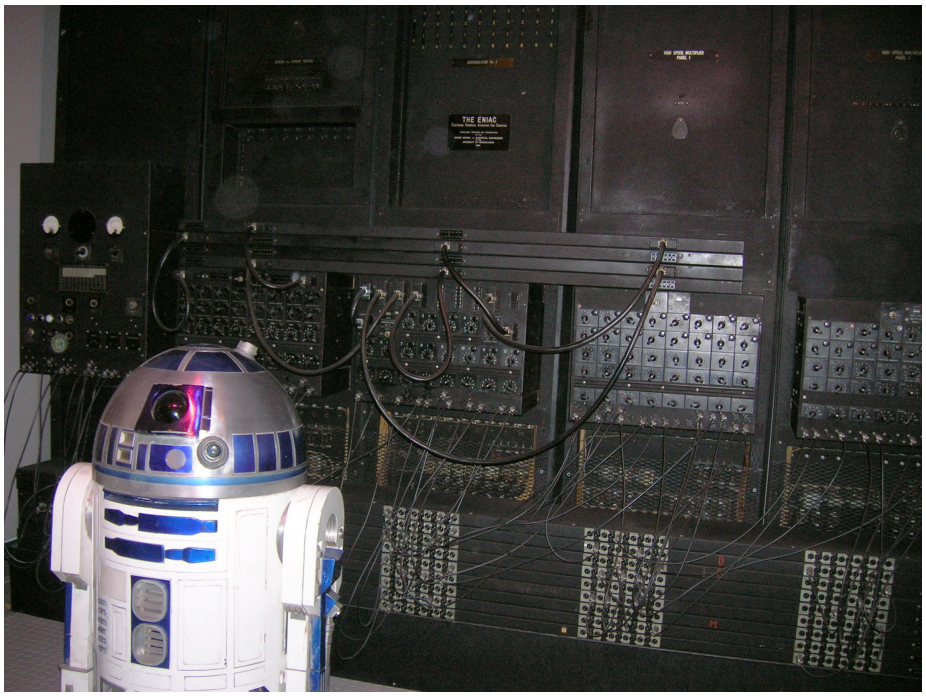


Figura 3: ENIAC y R2D2 en el Museo del Aire y el Espacio de Washington

Figura 4: Reloj “inteligente”

como los sistemas operativos o los compiladores. Este tipo de programas suelen interactuar directamente con el hardware, de manera que sus usuarios no son los usuarios finales que usan el ordenador, sino otros programadores.

- **Software de aplicación**. Son programas independientes que resuelven una necesidad **específica**, normalmente de una organización. Por ejemplo, sistema de citas de un centro de salud. Pueden ser desarrollados a medida (para un único cliente) o como software de propósito general (se intentan cubrir las necesidades de varios clientes y es habitual que éstos utilicen sólo un subconjunto de la funcionalidad total).
- **Software científico y de ingeniería**. Muy enfocados al cálculo y a la simulación, se caracterizan por la utilización de algoritmos y modelos matemáticos complejos.
- **Software empotrado**. Forma parte de algún dispositivo. Ejemplos: sistemas de frenado, lavadoras...
- **Aplicaciones web**. Se caracterizan por unificar fuentes de datos y diferentes servicios en entornos altamente distribuidos. Es en este ámbito donde las metodologías ágiles son muy populares.
- **Software de inteligencia artificial**. Utilizan técnicas y algoritmos muy diferentes, e incluso lenguajes, por ejemplo los lenguajes de programación funcional como Haskell, CLIPS...

En general, las técnicas de ingeniería del software están enfocadas al desarrollo de **software de aplicación**, y en concreto a **sistemas de información**.

Sistemas de información

Un sistema de información es un conjunto de elementos (personas, datos, técnicas y recursos materiales, fundamentalmente informáticos)

orientados al tratamiento y administración de datos e información, organizados y listos para su uso posterior, generados para cubrir una necesidad o un objetivo.

El software para sistemas de información es un tipo de software que gestiona una cierta información mediante un sistema gestor de bases de datos y soporta una serie de actividades humanas dentro del contexto de un sistema de información.

3. Evolución histórica: la crisis del software

Evolución histórica: la crisis del software

Contexto histórico años 60-70

- Pocas computadoras:

    - Grandes computadoras o *mainframes*
    - Muy pocos y muy caros
    - Poca variedad de software $\Rightarrow$ muy específico y desarrollado por el fabricante. Obviamente era difícil programar por libre.
- Se desarrolla software **artesanal**:
    - El negocio estaba en el hardware, no se le prestaba demasiada atención al software
    - Se dispone del código fuente y los desarrolladores de software compartían libremente sus programas unos con otros con ánimo constructivo (más adelante en la década de los 80 esto sería uno de los detonantes para el movimiento del **software libre**)
    - El desarrollo de software no se gestionaba según una planificación y era prácticamente imposible predecir los costes y el tiempo de desarrollo. Pero ya se aplicaban algunas técnicas de reutilización, como la programación modular.

Originalmente, el software era un producto gratuito que se incluía al comprar hardware y era desarrollado, principalmente, por las compañías fabricantes del hardware. Unas pocas compañías desarrollaban software a medida, pero no existía el concepto de software empaquetado, como producto.

Más contexto histórico

Algunas **referencias útiles** para comprender cuáles eran los conocimientos estables para el desarrollo de software en **1968** son Palacio and Ruata [2011]:

- 1962 primer algoritmo para búsquedas binarias. En 1962 se publicó el primer algoritmo para búsquedas binarias.
- 1966 fundación para la eliminación de "GoTo" y la creación de la programación estructurada. C. Böhm y G. Jacopini publicaron en 1966 el documento que creaba una fundación para la eliminación de "GoTo" y la creación de la programación estructurada.
- En 1968 los programadores se debatían entre el uso de la sentencia GoTo, y la nueva idea de programación estructurada; ese era el caldo de cultivo en el que Edsger Dijkstra escribió su famosa carta "GoTo Statement Considered Harmful" en 1968.
- 1974 Primera publicación sobre programación estructurada, por Larry Constantine, Glenford Myers y Wayne Stevens.
- 1976 primer libro sobre métrica de software por Tom Gilb.
- 1976 primer libro sobre análisis de requisitos.

La “crisis del software” y congreso OTAN

La crisis del software
La crisis del software se refiere a la dificultad en escribir programas libres de defectos, fácilmente comprensibles, y que sean verificables.

Surgió en la primera conferencia de la OTAN (Organización del Tratado del Atlántico Norte) sobre ingeniería del software (1968), se refiere a:

- Los proyectos no se ajustaban al presupuesto inicial.
- Los proyectos no terminaban en plazo.
- El software era ineficiente.
- El software era de mala calidad.
- El software no cumplía los requisitos.
- Los proyectos se volvían inmanejables y difíciles de gestionar y evolucionar.
- O peor aún... el software no llegaba a terminarse/entregarse nunca.

4. Naturaleza y Problemas del Desarrollo de Software

Naturaleza y Problemas del Desarrollo de Software

Causas de la crisis del software

- Naturaleza lógica del software
- Mala gestión de los proyectos (ausencia de datos, deficiente comunicación, etc.)
- Ausencia de entrenamiento formal en nuevas técnicas (programadores vs. ingenieros de software)
- Resistencia al cambio

Mitos del software

Mitos de los gestores

- Uso de estándares de otras ingenierías
- Mala planificación: un aumento de programadores acelera el proyecto

Mitos de los desarrolladores

- Programa funcionando = fin del trabajo
- Calidad = el programa se ejecuta sin errores
- Entrega al cliente = programa funcionando

Mitos del cliente

- Requisitos establecidos como una declaración general de objetivos
- Flexibilidad del software ante los cambios

5. La ingeniería del software

La ingeniería del software

Definición de ingeniería del software

Definición inicial
"Establecimiento y uso de principios de ingeniería para obtener software económico que trabaje de forma eficiente en máquinas reales". Fritz Baver, 1968 (conferencia OTAN (NATO))

Definición IEEE
"Ingeniería de software es la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento de software, y el estudio de estos enfoques, es decir, la aplicación de la ingeniería al software. Es la aplicación de la ingeniería al software, ya que integra matemáticas, ciencias de la computación y prácticas cuyos orígenes se encuentran en la ingeniería." Abran et al. [2004]

Otra definición
Disciplina que comprende **todos los aspectos de la producción de software**, desde las etapas iniciales hasta el mantenimiento:

- "disciplina de ingeniería": aplicación de teorías, métodos y herramientas para solucionar problemas, y teniendo en cuenta restricciones financieras y organizativas.
- "todos los aspectos de producción": comprende procesos técnicos del desarrollo y actividades como la administración de proyectos, desarrollo de herramientas, métodos y teorías.

Actividad de:

- Modelado
- Solución de problemas

- Adquisición de conocimiento
- Dirigida por una fundamentación

Otras definiciones según Wikipedia Wikipedia [2014b]

- Ingeniería de software es el estudio de los principios y metodologías para el desarrollo y mantenimiento de sistemas software (Zelkovitz, 1978).
- Ingeniería de software es la aplicación práctica del conocimiento científico al diseño y construcción de programas de computadora y a la documentación asociada requerida para desarrollar, operar y mantenerlos. Se conoce también como desarrollo de software o producción de software (Bohem, 1976).
- La ingeniería de software trata del establecimiento de los principios y métodos de la ingeniería a fin de obtener software de modo rentable, que sea fiable y trabaje en máquinas reales (Bauer, 1972).

La Ingeniería del Software

Desde 1968 hasta la fecha han sido muchos los esfuerzos realizados por los departamentos de informática de las universidades, y por organismos de estandarización (SEI, IEEE, ISO, OMG) para identificar las **causas del problema** y **definir pautas estándar** para la **producción** y **mantenimiento** del software.

Los esfuerzos se han encaminado en tres direcciones principales:

- Identificación de los factores clave que determinan la calidad del software.
- Identificación de los procesos necesarios para producir y mantener software.
- Acotación, estructuración y desarrollo de la base de conocimiento necesaria para la producción y mantenimiento de software.

El resultado ha sido la **necesidad de profesionalizar el desarrollo, mantenimiento y operación de los sistemas de software**, introduciendo métodos y formas de trabajo sistemáticos, disciplinados y cuantificables.

Observación con precaución

La forma de trabajo de programadores individuales surgida por la necesidad de los primeros programas, ha creado una cultura de la programación heroica, para el desarrollo de software que es la principal causa de los problemas apuntados, y en la actualidad una de las principales resistencias a la implantación de técnicas de ingeniería para el desarrollo de sistemas.

Valorar estas afirmaciones en contraste con el manifiesto ágil y los principios de gestión SCRUM.

¿Qué es la Ingeniería del Software?
La IS es aplicar el sentido común al desarrollo de sistemas software

¿Qué es el sentido común (en IS)?

- Planificar antes de desarrollar
- Diseñar antes de programar ("Meses de programación...")
- Reutilizar diseños que funcionan y son mantenibles (por ejemplo utilizando "Patrones de Diseño")
- Utilizando las herramientas apropiadas ⇒ Herramientas CASE (*Computer Aided Software Engineering*, ingeniería de software asistida por computadora) Wikipedia [2014a].

Algunas herramientas CASE

- Edición de diagramas (DIA, Enterprise Architect...)
- Comprobar la consistencia de los diagramas
- Generación de documentación (Doxygen, JavaDoc...)
- Seguimiento de actividades del proyecto (Gnome Planner...)
- Diferente soporte según el tipo (consultar más tipologías en Wikipedia [2014a]):
  - *Upper-CASE*. Herramientas que ayudan en las actividades de captura de requisitos, análisis y diseño
  - *Lower-CASE*. Herramientas para la programación, depuración y pruebas (Eclipse, GIT, gdb)

La Ingeniería del Software es una ingeniería muy joven que necesitaba procesos de **estandarización** Palacio [2014].

Los estándares son útiles porque:

- Agrupan lo mejor y más apropiado de las buenas prácticas y usos del desarrollo de software.
- Engloban los "conocimientos".
- Proporcionan un marco para implementar procedimientos de aseguramiento de la calidad.
- Proporcionan continuidad y entendimiento entre el trabajo de personas y organizaciones distintas.

Esta es la teoría, en la práctica los monopolios suelen atentar contra la estandarización. "España, el país que apuesta por los estándares abiertos pero sólo usa PDF", eldiario.es

Estandarización de la IS

Entidades de estandarización:

- **Instituto de Ingeniería del software (SEI)**. CMM/CMMI. www.sei.cmu.edu/
- **Organización Internacional para la Estandarización (ISO)**. Fundada en 1947. ISO/IEC 12207 e ISO/IEC TR 15504. www.iso.org
- **IEEE Computer Society**. IEEE es el Instituto de Ingenieros en electricidad y electrónica (Institute of Electrical and Electronics Engineers). La IEEE Computer Society es una sociedad integrada en IEEE, ha desarrollado estándares para todas las áreas de Ingeniería del Software. www.computer.org
  - IEEE Std. 830 Prácticas recomendadas para las especificaciones de software.
  - IEEE Std. 1362 Guía para la especificación del documento de requisitos "ConOps"
  - IEEE Std. 1063 Estándar para la documentación de usuario de software.
  - IEEE Std. 1012 Estándar para la verificación y validación de software.
  - IEEE Std. 1219 Estándar para el mantenimiento del software
- **Object Management Group (OMG)**. www.omg.org
  - Unified Modeling Language (UML) (*presente en el plan de estudios*)
  - Model Driven Architecture (MDA)
  - ...

Estándares y necesidades de la IS:

- **Definirse a sí misma**: ¿Cuáles son las áreas de conocimiento que la comprenden? → SWEBOK: Software Engineering Body of knowledge
- **Definir los procesos que intervienen en el desarrollo, mantenimiento y operación del software** → ISO/IEC 12207: Procesos del ciclo de vida del software.
- **De las mejores prácticas, extraer modelos de cómo ejecutar esos procesos para evitar los problemas** de la "crisis del software" → CMM/CMMI, ISO/IEC TR 15504
- **Definir estándares menores para dibujar criterios unificadores** en requisitos, pruebas, gestión de la configuración, etc. → IEEE 830-IEEE 1362, ISO/IEC 14764, UML...

El **proyecto SWEBOK** (*Software Engineering Body of Knowledge*) comenzó sus actividades de manera efectiva dentro del SWECC 1 en 1997 (aunque el comité SWECC se creó en 1993).

El proyecto parte de la suposición de que es necesario establecer cuál es el cuerpo de conocimiento que deben conocer los ingenieros del software, y en su desarrollo ha agrupado este conocimiento en varias áreas (ver Bourque and Fairley [2014] o `www.swebok.org`).

Solución de Problemas

Los ingenieros de software **buscan una solución** adecuada, en varios **pasos**:

- Formular el problema
- Analizar el problema
- Buscar soluciones
- Decidir la solución más adecuada
- Especificar la solución

Actividades (veremos más detalle en el siguiente tema):

- Obtención de requerimientos
- Análisis
- Diseño del sistema
- Implementación

Evaluación de la adecuación de los modelos:

- Revisión del análisis
- Revisión del diseño
- Pruebas
- Administración: calendario y presupuesto

6. Conceptos de IS

Conceptos de IS

Roles en IS

Papel o Rol

- Conjunto de responsabilidades en el proyecto o en el sistema

- Asociado con un conjunto de tareas y se asigna a un participante
- Un mismo participante puede cumplir varios papeles

Roles en IS (*¡En un sistema de información encargado desde fuera!*):

- **Cliente**: encarga y paga el sistema
- **Desarrolladores**: construyen el sistema (analistas, diseñadores, programadores, calidad (pruebas)...)
- **Gerente o director del proyecto**: planifica y calcula el presupuesto, coordina a los desarrolladores y cliente
- **Usuarios finales**: los que van a utilizar el sistema

Otros conceptos

- **Sistemas y modelos**
  - **Sistema**: realidad subyacente
  - **Modelo**: cualquier abstracción de la realidad
- **Productos** de trabajo o **Entregable**:
  - Artefacto o elemento que se produce durante el desarrollo (documento, fragmento de software,...)
- Actividades, tareas y recursos:
  - **Actividad** (o fase): conjunto de tareas que se realiza con un propósito específico (obtención de requisitos, entrega, administración,...) que pueden componerse de otras actividades
  - **Tarea**: unidad elemental de trabajo que puede ser administrada; consumen recursos, dan como resultado productos de trabajo y dependen de productos de trabajo producidos por otras tareas
  - **Recursos**: bienes que se utilizan para realizar el trabajo: tiempo, equipamiento, **personas**...
- Objetivos, Requerimientos y Restricciones:
  - **Objetivos**:
    - Principios de alto nivel que se utilizan para guiar el proyecto
    - Definen los atributos realmente importantes del sistema (seguridad, fiabilidad,...)
    - A veces hay conflicto entre objetivos (por ejemplo, seguridad y bajo coste) que aumentan la complejidad del proyecto
  - **Requerimientos**:

        - Características que debe tener el sistema
        - **Requerimiento funcional**: área de funcionalidad que debe soportar el sistema (por ejemplo, proporcionar billetes de tren)
        - **Requerimiento no funcional**: restricción que se establece sobre el funcionamiento del sistema (por ejemplo, proporcionar billetes de tren en menos de un segundo). *¿Qué disciplina nos ayudaría a cumplir este requerimiento?*
    - **Otras restricciones**: por ejemplo, utilización de un determinado lenguaje, de una determinada plataforma o de un sistema antiguo que el cliente no quiere retirar.

- Notaciones, métodos y metodologías:

    - **Notación**: conjunto de reglas gráficas o de texto para representar un modelo (UML, *Unified Modelling Language*, es una notación gráfica orientada a objetos para representar modelos)
    - **Método**: técnica repetible para resolver un problema específico. Por ejemplo:
        - un algoritmo de ordenación es un método para ordenar elementos en una lista
        - la administración de la configuración es un método para el seguimiento de los cambios
    - **Metodología**: colección de métodos para la resolución de una clase de problemas (OMT, metodología de Booch, Catalysis, Proceso Unificado de Desarrollo,...)

Principios de la Ingeniería del Software (clásica)

Principios de la Ingeniería del Software (**clásica**) Ruiz [2013]:

- Haz de la calidad la razón de trabajar
- Una buena gestión es más importante que una buena tecnología
- Las personas y el tiempo no son intercambiables
- Seleccionar el modelo de ciclo de vida adecuado
- Entregar productos al usuario lo más pronto posible
- Determinar y acotar el problema antes de escribir los requisitos
- Realizar un diseño
- Minimizar la distancia intelectual
- Documentar
- Las técnicas son anteriores a las herramientas

- Primero hazlo correcto, luego hazlo rápido
- Probar, probar y probar (** incluye inspecciones **)
- Introducir las mejoras y modificaciones con cuidado
- Asume responsabilidades
- La entropía del Software es creciente
- La gente es la clave del éxito
- Nunca dejes que tu jefe o cliente te convenza para hacer mal un trabajo
- La gente necesita sentir que su trabajo es apreciado
- La educación continua es responsabilidad de cada miembro del equipo
- El compromiso del cliente es el factor más crítico en la calidad del software
- Tu mejor desafío es compartir la visión del producto con el cliente
- La mejora continua de tu proceso de desarrollo de software es posible y esencial
- Tener procedimientos escritos de desarrollo de software puede ayudar a crear una cultura compartida de buenas prácticas
- La calidad es el principal objetivo; la productividad a largo plazo es una consecuencia de alta calidad
- Haz que los errores los encuentre un colaborador y no un cliente
- Una clave en la calidad en el desarrollo de software es realizar iteraciones en todas las fases de desarrollo
- La gestión de errores y solicitud de cambios es esencial para controlar calidad y el mantenimiento
- Si mides los que haces, puedes aprender a hacerlo mejor
- Haz lo que tenga sentido, no recurras a los dogmas
- No pueden cambiar todo de una vez. Identifica los cambios que se traduzcan en los mayores beneficios, y comienza a implementarlos

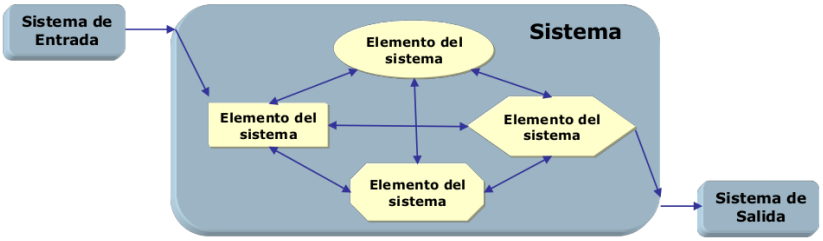


Figura 5: Visión general de un sistema.

7. Ingeniería de Sistemas e Ingeniería del Software

Ingeniería de Sistemas e Ingeniería del Software

Ingeniería de Sistemas

- ISO 12207 ISO [2008] establece un nexo con la Ingeniería de sistemas al considerar al software como parte de un sistema.
- Desde esta perspectiva se establece a la Ingeniería de sistemas como fundamento de la Ingeniería del Software. Es la **primera solución a la crisis del software**

¿Qué es un sistema?
"Colección de componentes organizados para cumplir una función o conjunto de funciones específicas". IEEE Standard 610.12-1990

"Colección de elementos relacionados de forma que puedan realizar un objetivo tangible". Pressman 1982

constructo: Construcción teórica para resolver un problema científico determinado.

- **Sistema**: conjunto de elementos de hardware, software, personas, procedimientos, herramientas y otros factores organizativos, organizados para llevar a cabo un objetivo común
- **Sistema Software**: Sistema o sub-sistema formado por una colección de programas y documentación que de forma conjunta satisfacen unos determinados requisitos. Se denomina "**sistema intensivo de software**" cuando prácticamente todo el sistema es software, y "**sub-sistema de software**" cuando es parte de uno mayor.
- El término "**Ingeniería de sistemas**" surgió por primera vez en 1956, y fue propuesto por H. Hitch, presidente del departamento de Ingeniería Aeronaútica de la Universidad de Pensilvania, para intentar desarrollar una disciplina de ingeniería que pudiera abarcar el desarrollo de grandes sistemas que empleaban diversas disciplinas de ingenierías específicas:

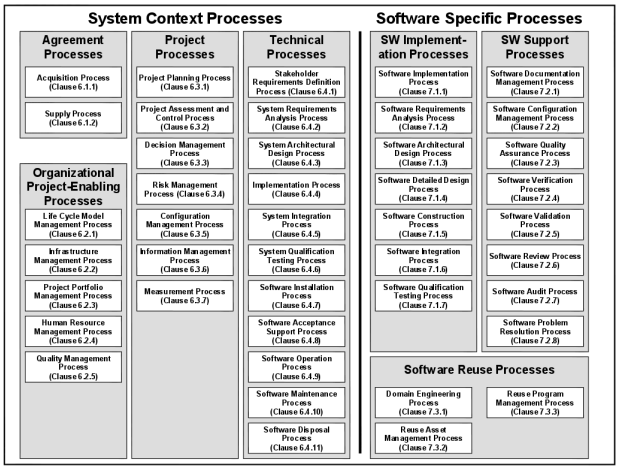


Figura 6: Procesos del estándar ISO 1227. ISO 1227 define los procesos que componen el ciclo de vida del software

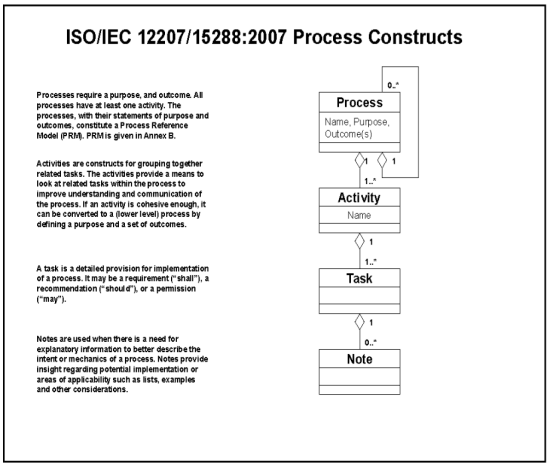


Figura 7: Constructos del proceso estándar ISO 1227

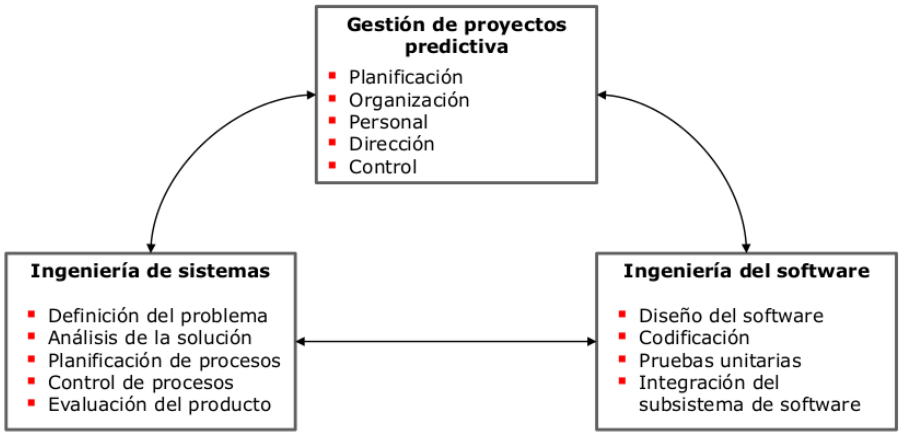


Figura 8: Ing. de sistemas - Gestión de proyectos predictiva - Ing. del Software (I). Fuente Palacio [2014]

construcción de bombarderos, submarinos, etc. Los principios de Ingeniería de sistemas desarrollados en los 60 y 70 se aplicaron en programas como el Apolo, o el programa de misiles balísticos USAF/USN.

La ingeniería de sistemas define el plan para gestionar las actividades técnicas del proyecto. Identifica el ciclo de desarrollo y los procesos que será necesario aplicar. Desde la Ingeniería de sistemas se desarrolla la línea base técnica para todo el desarrollo, tanto de hardware como de software. Palacio [2014]

8. Metodologías ágiles

Metodologías ágiles

Reflexión

Si observáis...

- la mayor parte de los principios y conceptos se refieren más al **QUÉ**, que al **CÓMO**.
- depende de los autores (incluso depende del país) se consideran unas **tareas propias de la programación** y otras de la **ingeniería del software**. La IS dice que hay que hacer pruebas y software fácilmente modificable, pero, en general, no dice cómo hacerlo: algunos temas se entrelazan con otras áreas/asignaturas.

Figura 9: Ing. de sistemas - Gestión de proyectos predictiva - Ing. del Software (II). Fuente Palacio [2014]

- Por otro lado, este tipo de Ingeniería del Software se denomina como "**ingeniería predictiva**" ya que trata de anticipar el mayor número de requisitos y delimitarlos en el tiempo.

Manifiesto Ágil

En marzo de 2001, 17 críticos de los modelos de producción basados en procesos, convocados por Kent Beck, que había publicado un par de años antes el libro en el que explicaba la nueva metodología Extreme Programming (Beck, 2000) se reunieron en Salt Lake City para discutir sobre el desarrollo de software. En la reunión se acuñó el término "**Métodos Ágiles**" para definir a aquellos que estaban surgiendo como alternativa a las metodologías formales: CMM-SW, (precursor de CMMI) PMI, SPICE (proyecto inicial de ISO 15504), a las que consideraban excesivamente "pesadas" y rígidas por su carácter normativo y fuerte dependencia de planificaciones detalladas, previas al desarrollo.

Los integrantes de la reunión resumieron en cuatro postulados lo que ha quedado denominado como "**Manifiesto Ágil**", que son los valores sobre los que se asientan estos métodos.

Hasta 2005, entre los defensores de los modelos de procesos y los de modelos ágiles fueron frecuentes las posturas radicales, más ocupadas en descalificar al otro, que en estudiar sus métodos y conocerlos para mejorar los propios. Palacio and Ruata [2011]

Manifiesto Ágil

Estamos poniendo al descubierto mejores métodos para desarrollar software, haciéndolo y ayudando a otros a que lo hagan. Con este trabajo hemos llegado a valorar:

- *A los individuos y su interacción, por encima de los procesos y las herramientas.*

- *El software que funciona, por encima de la documentación exhaustiva.*
- *La colaboración con el cliente, por encima de la negociación contractual.*
- *La respuesta al cambio, por encima del seguimiento de un plan.*

Aunque hay valor en los elementos de la derecha, valoramos más los de la izquierda.

Metodologías ágiles respecto a los estándares

Críticas de las metodologías ágiles

Estas observaciones, junto a otras, son la motivación del desarrollo de las conocidas como metodologías ágiles [Palacio and Ruata, 2011].

Modelos híbridos

En Palacio and Ruata [2011] podemos ver el ejercicio dialéctico entre Tesis (Ingeniería del Software *clásica*) y Antítesis (metodologías ágiles) que hoy en día se está resolviendo de manera constructiva tratando de seleccionar las partes más beneficiosas de ambas. En todo caso la mejor técnica o mixtura de técnicas dependerá siempre del ámbito de aplicación.

Reflexiones en Wikipedia

En el artículo *Proceso para el desarrollo de software* Wikipedia [2014c]:

La gran cantidad de organizaciones de desarrollo de software implementan metodologías para el proceso de desarrollo. Muchas de estas organizaciones pertenecen a la industria armamentística, que en los Estados Unidos necesita un certificado basado en su modelo de procesos para poder obtener un contrato.

El estándar internacional que regula el método de selección, implementación y monitoreo del ciclo de vida del software es ISO 12207.

...

Los modelos de desarrollo de software son una representación abstracta de una manera en particular. Realmente no representa cómo se debe desarrollar el software, sino de un enfoque común. Puede ser modificado y adaptado de acuerdo a las necesidades del software en proceso de desarrollo. Hay varios modelos para perfilar el proceso de desarrollo, cada uno de las cuales cuenta con pros y contras. El proyecto debería escoger el más apropiado para sus necesidades. En ocasiones puede que una combinación de varios modelos sea apropiado. Existen tres paradigmas de los modelos de desarrollo de software:

Advertencia

Advertencia

Estos apuntes no contienen todo el material necesario respecto al tema. Intentan dar una visión de conjunto y proporcionar los conocimientos básicos para que podáis consultar la bibliografía y manuales de referencia, así como material complementario colgado en Moodle y los ejercicios realizados en clase.

Bibliografía y enlaces

Referencias

A. Abran, P. Bourque, R. Dupuis, J. W. Moore, and L. L. Tripp. *Guide to the Software Engineering Body of Knowledge - SWEBOK*. IEEE Press, Piscataway, NJ, USA, 2004 version edition, 2004. ISBN 0769510000. URL http://www.swebok.org/ironman/pdf/SWEBOK_Guide_2004.pdf.

P. Bourque and R. Fairley, editors. *Guide to the Software Engineering Body of Knowledge, Version 3.0,*. IEEE Computer Society, 2014. URL www.swebok.org.

ISO. ISO/IEC 12207:2008 Systems and software engineering – Software life cycle processes. Technical report, International Organization for Standardization, Geneva, 2008. URL http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=43447. Disponible también en http://www.ing.unisannio.it/cimitile/ingsw/dispense/ISO_12207.pdf.

J. Palacio. Compendio de Ingeniería del Software, 2014. URL http://www.safecreative.org/work/1401289956290.

J. Palacio and C. Ruata. Scrum Manager Gestión de Proyectos, 2011. URL http://www.scrummanager.net/bok/index.php?title=Main_Page. version 1.4.

J. Pradel i Miquel and J. A. R. Martos. *Introducción a la ingeniería del software. Asignatura Ingeniería de ingeniería del Software*. Universitat Oberta de Catalunya, 2010. FUOC PID_00171152.

I. T. L. Ruiz. Apuntes de la Asignatura Ingeniería del Software. Tema 1: Introducción a la Ingeniería del software. 2013.

Wikipedia. Computer-aided software engineering — Wikipedia, The Free Encyclopedia, 2014a. URL http://en.wikipedia.org/w/index.php?title=Computer-aided_software_engineering&oldid=629476995. [Online; accessed 20-October-2014].

Wikipedia. Ingeniería de software – Wikipedia, La enciclopedia libre, 2014b. URL http://es.wikipedia.org/w/index.php?title=Ingenier%C3%ADa_de_software&oldid=77526102. [Internet; descargado 20-octubre-2014].

Wikipedia. Proceso para el desarrollo de software — Wikipedia, La enciclopedia libre, 2014c. URL http://es.wikipedia.org/w/index.php?title=Proceso_para_el_desarrollo_de_software&oldid=77406029. [Internet; descargado 21-octubre-2014].