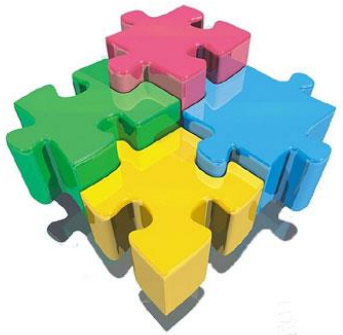
Ingeniería del Software

Software

Software



SOFTWARE es el conjunto de programas o aplicaciones, instrucciones y reglas informáticas que hacen posible el funcionamiento del equipo.

SOFTWARE es la suma total de los programas de cómputo, procedimientos, reglas de documentación y datos asociados que forman parte de las operaciones de un sistema de cómputo [IEEE Computer Society Press].

Es un producto que diseñan y construyen los ingenieros de software. Esto abarca ***programas*** que se ejecutan dentro de una computadora de cualquier tamaño y arquitectura, ***documentos*** que comprenden formularios virtuales e impresos y ***datos*** que combinan números y texto y también incluyen representaciones de la información de audio, vídeo e imágenes [Pressman].

Software Cont.

- “El software es un lugar donde se siembran sueños y se cosechan pesadillas, una ciénega abstracta y mística en la que terribles demonios luchan contra panaceas mágicas, un mundo de hombres lobo y balas de plata.”
- El software distribuye el producto más importante de nuestro tiempo: información. Transforma los datos personales (por ejemplo, las transacciones financieras de un individuo) de modo que puedan ser más útiles en un contexto local, administra la información de negocios para mejorar la competitividad, provee una vía para las redes mundiales de información (la internet) y brinda los medios para obtener información en todas sus formas.

Software Cont. 2

Programas de cómputo y documentación asociada, como ser documentos de requerimientos, arquitectura y modelos de diseño y manuales de usuario.

Los productos software pueden ser desarrollados para un cliente particular o bien para el mercado en general.

Los productos software pueden ser:

- Genéricos – desarrollados para ser vendidos a una gama de diversos clientes, ejemplo: software como Excel o Word.
- Hecho a medida – desarrollado para un cliente particular acorde a sus requerimientos y especificaciones.

Nuevo software puede ser creado desarrollando nuevos programas, configurando sistemas de software genérico o reutilizando software existente.

Aunque la industria se mueve hacia la construcción basada en componentes, la mayor parte del software se construye para un uso individualizado.

Evolución del Software

1950

Aplicaciones numéricas específicas

1960

Aplicaciones numéricas y administrativas ad hoc

1970

Se habla de una sociedad de información, grandes BD

1980

Sistemas cliente/servidor

1990

Internet

2000

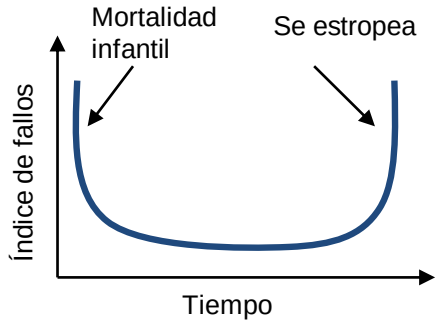
Y2K

Aplicaciones basadas en componentes

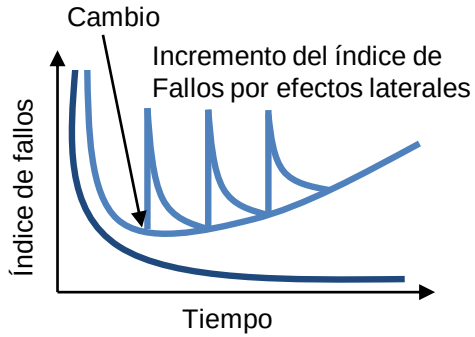
Características del software

El software al ser un elemento lógico tiene ciertas características que lo diferencian claramente respecto al hardware [Pressman].

- El software se desarrolla, no se fabrica en un sentido clásico.
  - El desarrollo y fabricación generan un producto pero desde enfoques diferentes.
- El software no se estropea; pero se deteriora.
  - Los fallos del hardware se dan al principio y al final de su vida, mientras que en el software el mantenimiento dado a lo largo de su vida introduce nuevos fallos.
- Aunque la industria tiende a ensamblar componentes, la mayoría del software se construye a la medida.
  - Esta situación esta cambiando con el uso más extendido de la programación orientada a objetos.



Curva de fallos del hardware



Curvas de fallos real e Idealizado del software

Características del Software Cont.

- Hardware
  - Sistema físico
  - Proyectos de fabricación
  - Se desgasta, aumento de errores al paso del tiempo
  - Uso pesado de componentes

- Software
  - Sistema lógico
  - Proyectos de Ingeniería
  - No se desgasta, no sufre nuevos errores (sin mantenimiento)
  - Software a la medida, aún con componentes

Categorias de Aplicación del software

Actualmente hay siete categorías de software [Pressman]

1. **Software de sistemas**
   - Conjunto de programas para servir a otros programas.
   - En general tienen una fuerte interacción con el hardware, múltiples usuarios, operación concurrente, compartición de recursos, estructuras de datos complejas, entre otras.
   - Ejemplos: compiladores, editores, utilidades de gestión de archivos, controladores, software de redes, etc.
2. **Software de aplicación**
   - Programas aislados que resuelven una necesidad específica de negocios.
   - Procesan datos comerciales o técnicos para facilitar las operaciones o toma de decisiones de negocios o técnicas.
   - Ejemplos: procesamiento de transacciones en puntos de venta, control de procesos de manufactura en tiempo real.
3. **Software de ingeniería y ciencias**
   - Se ha caracterizado por “algoritmos devoradores de números”.
   - Las aplicaciones van desde la astronomía a la vulcanología, del análisis de tensiones en automóviles a la dinámica orbital de un transbordador espacial, de la biología molecular a la manufactura automatizada.

Categorias de aplicaciones del software … Cont.

Categorías del software …

4. **Software incrustado**
   - Reside dentro de un producto o sistema y se usa para implementar y controlar funciones para el usuario final y para el sistema en sí.
   - Ejecuta funciones limitadas y particulares o provee una capacidad de funcionamiento y control.
   - Ejemplos: control del tablero de un microondas, funciones digitales en un automóvil como el control de combustible.
5. **Software de línea de productos**
   - Es diseñado para proporcionar una capacidad específica para uso de muchos consumidores diferentes.
   - Se centra en un mercado particular o a mercados masivos.
   - Ejemplos: control de inventario de productos, procesadores de textos, hoja de cálculo, etc.
6. **Aplicaciones Web**
   - Llamadas Webapps. Esta categoría de software centrado en redes agrupa una amplia gama de aplicaciones.
   - Van desde sencillas páginas dinámicas hasta ambientes de cómputo sofisticados con integración a bases de datos corporativas y aplicaciones de negocios.
7. **Software de inteligencia artificial**
   - Hace uso de algoritmos no numéricos para resolver problemas complejos para los que no son adecuados los análisis directos.
   - Ejemplos: sistemas expertos o basados en conocimientos, reconocimiento de patrones, imágenes, voz, redes neuronales artificiales, etc.

La crisis del software

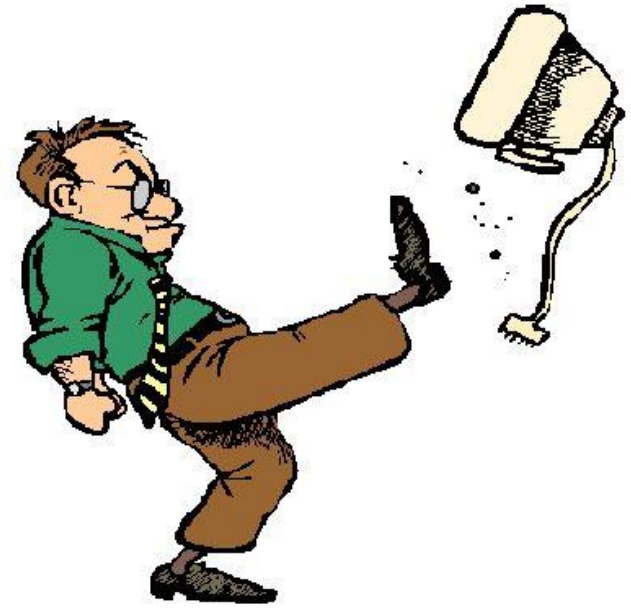
La mayoría de los expertos están de acuerdo en que la manera más probable para que el mundo se destruya es por accidente.

Ahí es donde nosotros entramos; somos profesionales de la informática, provocamos accidentes [Nathaniel Borenstein].

Se trata más de una aflicción crónica que de una crisis.

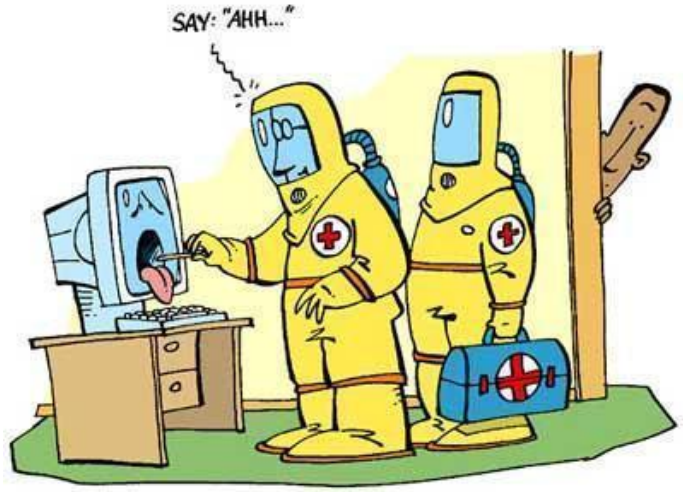
- Aflicción porque causa pena o desastre.
- Crónica porque es duradero y reaparece con frecuencia.
- No se le puede llamar del todo crisis porque no ha sido un punto decisivo en el curso de la "enfermedad".

Independientemente de que se le llame crisis o aflicción, se alude a un conjunto de problemas relacionados con el desarrollo de software.



Factores que contribuyen a la Crisis del Software

1. Problemas cada vez más grandes,
2. La falta de una formación adecuada en ingeniería de software,
3. El aumento de la falta de habilidades,
4. Baja productividad.



Software heredado

Los sistemas de software heredado [...] fueron desarrollados hace varias décadas y han sido modificados de manera continua para que satisfagan los cambios en los requerimientos de los negocios y plataformas de computación. La proliferación de tales sistemas es causa de dolores de cabeza para las organizaciones grandes, a las que resulta costoso mantenerlos y riesgoso hacerlos evolucionar.

WEBAPPS

Uso intensivo de redes.
Concurrencia.
Carga impredecible.
Rendimiento.
Disponibilidad.
Orientadas a los datos.
Contenido sensible.
Evolución continua.
Inmediatez.
Seguridad.
Estética.

Mitos del **Software**

Mitos

Los mitos son creencias que tienen sobre el software, tanto los desarrolladores como los que lo emplean ; poseen la característica de repetirse a lo largo del tiempo y pueden ser rastreados a los inicios de la computación. Una de las razones por las cuales estos mitos son tan populares radica en que parecen lógicos y en ocasiones son empleados por expertos en el tema.

Actualmente:

> *“Estos mitos están identificados y marcados como malas practicas en relación al software, pero pese a ello, es muy difícil erradicar la presencia de estos mitos del imaginario colectivo.”*

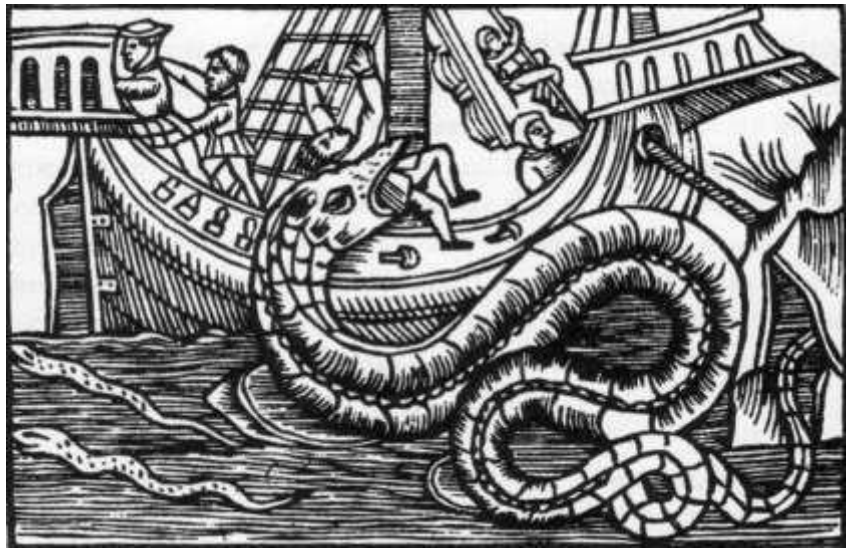
Tipos de mitos

Se tienen identificados 3 categorías de mitos asociados al software.

Mitos de la Administración

Mitos del Cliente

Mitos del Desarrollador.



Mitos de la **Administración**.

Los administradores de proyectos de software normalmente deben preocuparse por garantizar que se cumplan los itinerarios, que se mantengan los costos y que todo funcione como fue planeado. Lo anterior genera una serie de presión que muchas veces provoca que ellos se aferren a mitos a manera de salvavidas liberador de estas situaciones estresantes.

MITO 1: Se tiene un libro con estándares y procedimientos para el desarrollo de software. Esto proporcionara todo el conocimiento necesario a mi personal?.

Realidad: Se puede tener el libro, pero se esta empleando? Los desarrolladores conocen su existencia? Esta actualizado?. Es claro? . Esta orientado al alcance de la calidad?.

Mitos de la **Administración**.

MITO 2: Si tienes un retraso en el itinerario es factible contratar mas programadores para terminar a tiempo. (Horda Mongoliana).

Realidad: El desarrollo de software no es un proceso mecánico que permita adicionar mas personas para acelerar su desarrollo. De hecho es posible que vincular nuevo personal al proyecto provoque mayores contratiempos y retrasos , considerando el tiempo de capacitación y el acople al equipo del personal nuevo.

Mitos de la **Administración**.

MITO 3: Si dejo el desarrollo del proyecto de software a un tercero(subcontrato), puedo relajarme y dejar que esa compañía lo construya.

Realidad: No se puede descuidar el proyecto aunque se subcontrate, si una compañía no comprende como administrar y controlar sus proyectos de software de forma interna, sin lugar a dudas se presentaran problemas cuando trate de efectuar una subcontratación.

Mitos de la **Administración**.

- Los estándares y procedimientos son toda la guía que los Ing. de Software necesitan.
- Si contamos con la última generación de computadoras tenemos todas las herramientas necesarias.
- Si fallamos en la planificación, podemos añadir más programadores y adelantar el tiempo perdido.
- La calidad cuesta dinero: es un gasto.

Mitos del **Cliente**

Los clientes pueden llegar de cualquier lugar y tienen características muy diferentes, llegan con creencias predefinidas y mitos arraigados que en muchas oportunidades se explica por el poco esfuerzo de los profesionales del software por corregir esta desinformación. La presencia de estos mitos en el cliente produce falsas expectativas e insatisfacción con el trabajo del desarrollador.

MITO 1: Una descripción general de los objetivos es suficiente para iniciar los trabajos de construcción del software, los detalles se afinaran mas adelante.

Realidad: No siempre se tendrá claridad con los objetivos, si estos presentan una ambigüedad producirán todo un desastre. La comunicación constante y efectiva entre el cliente y el desarrollador son la mejor manera de identificar los requerimientos del software.

Mitos del **Cliente**

MITO 2: Los requerimientos de un software cambian constantemente, pero esto no se considerara un problema y se ajustan rápidamente porque el software es flexible.

Realidad: Es verdad que los requerimientos del software cambian, pero el impacto de estos cambios depende mucho del momento en que ellos ocurran. En etapas tempranas el costo de asimilar los cambios no son tan altos, pero a medida que las etapas están mas adelantadas el cambio en los requerimientos puedo involucrar el adicionar mas recursos y tiempos, incluso cambiar todo el software.

Mitos del **Cliente**

- Una declaración general de los objetivos del cliente es todo lo necesario para empezar a programar.
- Los requisitos cambian continuamente, pero los cambios pueden acomodarse fácilmente porque el software es flexible.

Mitos del **Desarrollador**

Los diferentes mitos que acompañan a los programadores se han mantenido durante muchos años. El desprenderse de estos mitos se hace difícil pues se vuelven un elemento de costumbre en los programadores.

MITO 1: Cuando el programa ha sido escrito y se coloco a funcionar, el trabajo quedo terminado.

Realidad: Entre el 60 y 80 % del trabajo se realiza posterior a la entrega al cliente(de acuerdo a estudios).

Mitos del **Desarrollador**

MITO 2: Mientras el programa no se este ejecutando no hay forma de evaluar su calidad.

Realidad: El software se debe probar en cada una de sus etapas , esto con el fin de garantizar su calidad. Incluso desde el inicio del proyecto con las revisiones técnicas formales y la verificación de los requisitos dados por los clientes.

Mitos del **Desarrollador**

MITO 3: El único producto que debe entregarse para considerar un proyecto de software exitoso es el programa funcionando.

Realidad: El programa funcionando es solo una parte. La documentación del software permite garantizar su calidad, realizarle mantenimiento y transformarse en una guía para nuevos desarrolladores.

Mitos del **Desarrollador**

MITO 4: La ingeniería del software obliga a realizar documentación voluminosa he innecesaria, teniendo como resultado un proceso mas lento.

Realidad: La ingeniería del software no es realizar documentación , es la búsqueda de calidad y con la calidad se reducen los trabajos redundantes lo que permite un proceso mas ágil. Con ello el cliente no solo recibe a tiempo un producto si no tiene la garantía que el mismo es de calidad.

Mitos del **Desarrollador**

- Una vez que se escribió el programa y se lo hizo funcionar, el trabajo del Ing. de Software está terminado.
- No hay forma de comprobar la calidad del software hasta no poder ejecutarlo en alguna máquina.
- Lo único que se entrega al terminar el proyecto es el programa funcionando.

Reflexión

Si no hacen análisis, hay t…..
Si no diseñan antes de programar, hay t…..
Si programan sin hacer pruebas, hay t…..
Si no comentan el código, hay t…..
Si no emplean los estándares, hay t…..
Si cuelgan una aplicación por un ciclo infinito, hay t…..
Si reportan el mismo error dos veces, hay t…..
Si los casos de uso no están bien escritos, hay t…..
Si no planean, hay t…..
Si no gestionan proactivamente los riesgos, hay t…..
Si no versionan, hay t…..
Si preguntan por que todo esto, hay t…..

Consecuencias por fallas del software

Se pueden clasificar en:

- Consecuencias inmediatas y efectos directos
  - Son los perjuicios ocasionados mientras dura la caída de los sistemas.
  - En sistemas de misión crítica (sistema bancario) se generan perdidas realmente significativas.
  - Los costos de estos fallos son relativamente predecibles dado que dependen directamente del tiempo que dure la interrupción de la operación.
- Consecuencias a mediano y largo plazo, y efectos indirectos
  - Son los perjuicios posteriores a la caída de los sistemas.
  - Las consecuencias varían, desde la restauración de los datos, propaganda negativa, pérdida de clientes hasta juicios en contra.
  - Es difícil de predecir el costo real a mediano y largo plazo.

¿Qué es Ingeniería?

Construir una casa para FIDO

Puede hacerlo una sola persona
Requiere:
- Modelado mínimo
- Proceso simple
- Herramientas simples

Vs.

Construido eficientemente y en un tiempo razonable por un equipo
Requiere:
- Modelado
- Proceso bien definido
- Herramientas más sofisticadas

¿Qué es Ingeniería?

APLICACIÓN de conjunto de conocimientos y técnicas científicas

¿Qué es software?

Elemento lógico de la computadora

Ciencia Vs Ingeniería

Modelado: Modelar los objetos es una tarea difícil

El buen modelado de objetos implica el dominio de conceptos complejos, la terminología y las convenciones.

También requiere una experiencia considerable y, a veces subjetiva en un proceso fuertemente basado en la experiencia.

Cuidado con la falsa creencia de que:

1. La tecnología puede sustituir a la habilidad y,
2. Esa habilidad es un reemplazo para el pensamiento.

Definiciones de Ingeniería de Software

La ingeniería de software es el establecimiento y uso de principios robustos de la ingeniería a fin de obtener económicamente software que sea fiable y que funcione eficientemente sobre máquinas reales [Bauer].

Ingeniería del software es la aplicación práctica del conocimiento científico en el diseño y construcción de programas de computadora y la documentación asociada requerida para desarrollar, operar y mantenerlos. Se conoce también como desarrollo de software o producción de software [Bohem].

La ingeniería de software es el estudio de los principios y metodologías para desarrollo y mantenimiento de sistemas de software [Zelkovitz].

Ingeniería de software: (1) La aplicación de un enfoque sistemático, disciplinado y cuantificable hacia el desarrollo, operación y mantenimiento del software; es decir, la aplicación la de la ingeniería al software. (2) El estudio de enfoques como en (1) [IEEE].

La Ingeniería de Software es una disciplina de la Ingeniería que concierne a todos los aspectos de la producción de software [Sommerville].

Ingeniería de Software Pressman

La ingeniería de software es el establecimiento y uso de principios fundamentales de la ingeniería con objeto de desarrollar en forma económica software que sea confiable y que trabaje con eficiencia en máquinas reales.

Ingeniería de Software

Ingeniería de Software

Colección de técnicas, metodologías y herramientas para ayudar con la producción de:

1. Un software de alta calidad
2. Con un presupuesto dado
3. Antes del plazo determinado
4. Mientras el cambio se produce

Ingeniería de Software Cont.

- Entender el problema antes de dar una solución.
- El diseño es una actividad crucial de la ingeniería de software.
- El software debe tener alta calidad.

Ingeniería de Software – Aspectos Importantes

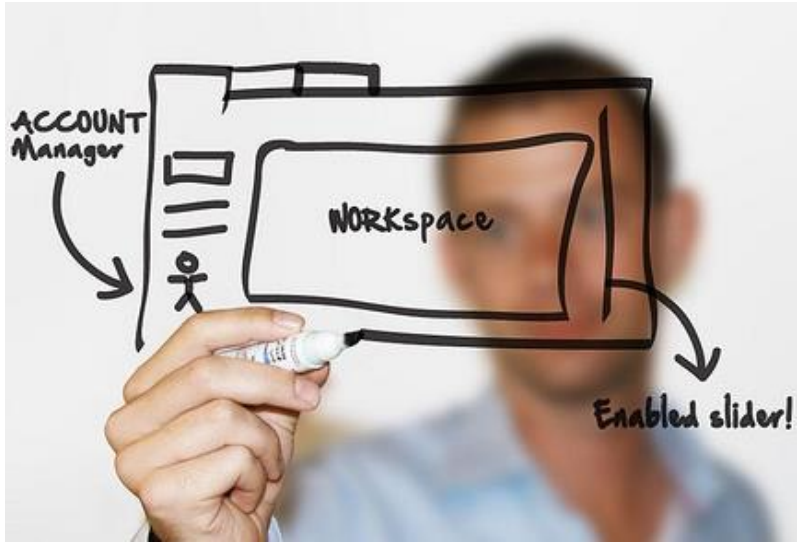
- Las economías de TODAS las naciones desarrolladas son dependientes de software y sistemas informáticos.
- Cada vez más sistemas son controlados por software.
- La Ingeniería de Software concierne teorías, métodos y herramientas para el desarrollo profesional de software.
- Los gastos en software representan una fracción significativa del PBI en todos los países desarrollados.

Ingeniería de Software: Una actividad de resolver problemas

Ingeniería de Software

Colección de técnicas, metodologías y herramientas para ayudar con la producción de:

1. Un software de alta calidad.
2. Con un presupuesto dado.
3. Antes del plazo determinado.
4. Mientras el cambio se produce.



Ingeniería de Software: Una actividad de resolver problemas

Ingeniería de software es “La aplicación de un enfoque sistemático, disciplinado y cuantificable para el desarrollo, operación y mantenimiento del software”.

“*IEEE Standard Glossary of Software Engineering Terminology*” (Stad. 610.12-1990) ha desarrollado esta definición más completa para ingeniería del software.

¿Por qué estudiar Ingeniería de Software?

1. Adquirir habilidades para desarrollar programas grandes.
2. El crecimiento exponencial del nivel de complejidad y dificultad con el tamaño.
3. El enfoque ad hoc se rompe cuando el tamaño del software incrementa

Para resolver un problema se usa:

1. Técnicas (métodos): Procedimientos formales para producir resultados usando alguna notación bien definida (ej. técnicas de programación)
2. Metodologías (Proceso y Modelos): Colección de técnicas aplicadas a través del desarrollo de software y unificado bajo un enfoque filosófico (Iterativo, incremental, RAD, RUP, XP, etc.)
3. Herramientas: instrumentos o sistemas automatizados para llevar a cabo una técnica

Cuál es la diferencia entre Ingeniería de Software y Computación?

- La computación comprende teorías y fundamentos de cualquier sistema de cómputo; a la Ingeniería de Software le concierne los aspectos prácticos del desarrollo y entrega de software útil.

- Las teorías de la computación aún son insuficientes para respaldar completamente a la Ingeniería de Software (indistintamente a, ejemplo, la física y la ingeniería eléctrica).

Ingeniería De Software Vs Ingeniería De Sistemas

Ingeniería de Sistemas

- Se ocupa de todos los aspectos del desarrollo de sistemas basados en computadores, incluyendo el hardware, software y la ingeniería de procesos.

- La Ingeniería de Software es parte de este proceso, en términos de arquitectura, control, bases de datos en el sistema (Diseño, integración y despliegue).

Cuál es la diferencia entre Ingeniería de Software e Ingeniería de Sistemas?

- A la ingeniería de sistemas le competen todos los aspectos de desarrollo de sistemas basados en cómputos, incluyendo hardware, software y procesos de ingeniería. La Ingeniería de Software es parte de este proceso, haciendo referencia al desarrollo de la infraestructura del software, aplicaciones y bases de datos en el sistema.

- Los ingenieros de sistemas están involucrados con la especificación del sistema, diseño arquitectónico, integración y despliegue del mismo.

Cuáles son los desafíos primordiales frente a la Ingeniería de Software?

Heterogeneidad

Técnicas de desarrollo para la construcción de software que puedan encararse con plataformas heterogéneas y ambientes de ejecución apropiados.

Entrega

Técnicas de desarrollo que lleven a una entrega de software más rápida;

Confianza

Técnicas de desarrollo que demuestren que el software es de confianza para con sus usuarios

Cuáles son los desafíos primordiales frente a la Ingeniería de Software?

Heterogeneidad

Técnicas de desarrollo para la construcción de software que puedan encararse con plataformas heterogéneas y ambientes de ejecución apropiados;

Entrega

Técnicas de desarrollo que lleven a una entrega de software más rápida;

Confianza

Técnicas de desarrollo que demuestren que el software es de confianza para con sus usuarios

Por que el software es tan complejo?

El dominio del problema es difícil
El dominio del problema es a veces difícil, simplemente porque no somos expertos en ello.
Es decir, puede que no sea un desafío intelectual, sino porque uno no es un experto en ello → hay que aprenderlo.

El software ofrece una flexibilidad extrema
Podemos cambiar casi todo lo que hemos diseñado en el software.
Si bien es difícil de cambiar el diseño de una lavadora, es muy fácil de cambiar el programa antes de ejecutarlo.

Por que el software es tan complejo? Cont.

El proceso de desarrollo es muy difícil de gestionar
Uno de los supuestos que los administradores han hecho en el pasado, es que el desarrollo de software puede ser gestionado como un conjunto de pasos en forma lineal, por ejemplo: Especificación de Requisitos, seguido de Diseño de Sistemas seguido de Ejecución seguido de pruebas y de entrega.

En realidad esto no es tan fácil. Desarrollo de software no se sigue un proceso lineal. Es altamente no lineal. Hay dependencias entre la forma en que se diseña un sistema y la funcionalidad. Por otra parte, y eso hace que sea muy difícil, algunas de estas dependencias no se pueden formular a menos que pruebe el diseño.

Por que el software es tan complejo? Cont. 2

El software es un sistema discreto (formas particulares de codificación)

Cuando uno está sentado en un avión en un asiento de la ventana, y oprime un botón para llamar a la azafata para tomar una bebida, no espera que el sistema de un giro y usted termine en la tierra.

Esto puede suceder con los sistemas digitales. Una de las razones:

Si bien se puede descomponer el sistema en subsistemas (Azafata, Control de Vuelo), si usted no sigue las buenas reglas de diseño, que podrían haber utilizado alguna variable global para cada uno de estos subsistemas.

Una de estas variables utilizadas por el subsistema de control de vuelo podrían haber sido sobrescritos por el módulo de Azafata.

Capas de la Ingeniería de Software (Pressman)

EL PROCESO DEL SOFTWARE

El proceso de software forma la base para el control de la administración de proyectos de software, y establece el contexto en el que se aplican métodos técnicos, se generan productos del trabajo (modelos, documentos, datos, reportes, formatos, etc.), se establecen puntos de referencia, se asegura la calidad y se administra el cambio de manera apropiada.

"Un proceso define quién hace qué, cuándo y cómo, para alcanzar cierto objetivo."

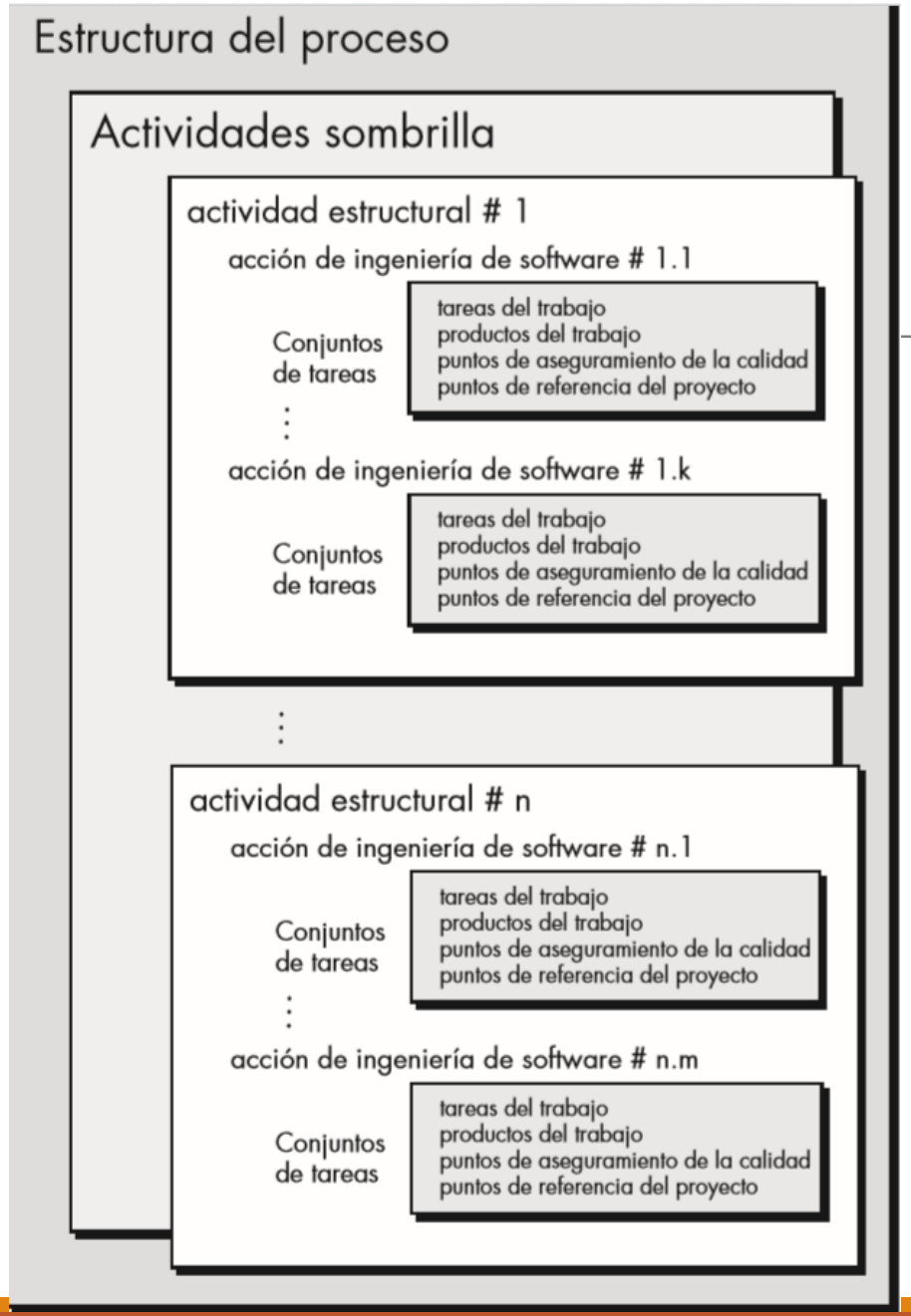
EL PROCESO DEL SOFTWARE Cont.

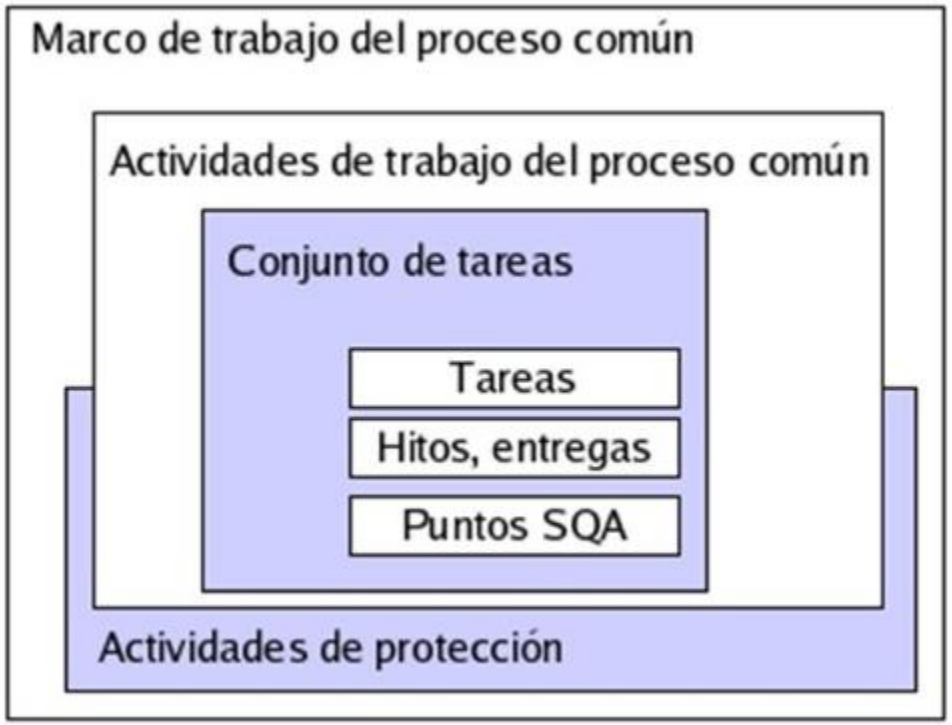
- El proceso de software no es único.
- No existe un proceso de software universal que sea efectivo para todos los contextos de proyectos de desarrollo.
- Debido a esta diversidad, es difícil automatizar todo un proceso de desarrollo de software.

Qué es un proceso de software?

Un conjunto sistemático de actividades cuya meta es el desarrollo o la evolución del software.
A pesar de la variedad de propuestas de proceso de software, existe un conjunto de actividades genéricas en todos los procesos de software son:

- Especificación – lo que el sistema debería hacer y sus restricciones de desarrollo
- Desarrollo – producción del sistema software
- Validación – comprobando que el software es lo que el cliente quiere
- Evolución – cambios y mantenimiento en el software con relación a los cambios en los requerimientos y demandas.


Estructura del proceso
Actividades sombrilla
actividad estructural # 1
acción de ingeniería de software # 1.1
Conjuntos de tareas
tareas del trabajo
productos del trabajo
puntos de aseguramiento de la calidad
puntos de referencia del proyecto
acción de ingeniería de software # 1.k
Conjuntos de tareas
tareas del trabajo
productos del trabajo
puntos de aseguramiento de la calidad
puntos de referencia del proyecto
actividad estructural # n
acción de ingeniería de software # n.1
Conjuntos de tareas
tareas del trabajo
productos del trabajo
puntos de aseguramiento de la calidad
puntos de referencia del proyecto
acción de ingeniería de software # n.m
Conjuntos de tareas
tareas del trabajo
productos del trabajo
puntos de aseguramiento de la calidad
puntos de referencia del proyecto


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

Actividades del Proceso de la IS

Comunicación: comunicarse y colaborar con el cliente (y con otros participantes)
Planeación: plan del proyecto de software, describir las tareas técnicas por realizar, los riesgos probables, los recursos que se requieren, los productos del trabajo que se obtendrán y una programación de las actividades.
Modelado: Crea un “bosquejo”
Construcción: Generación de código (ya sea manual o automatizada) y las pruebas que se requieren para descubrir errores en éste.
Despliegue: El software (como entidad completa o como un incremento parcialmente terminado) se entrega al consumidor que lo evalúa y que le da retroalimentación.

Actividades Sombrilla.

- Seguimiento y control del proyecto de software
- Administración del riesgo
- Aseguramiento de la calidad del software
- Revisiones técnicas
- Medición
- Administración de la configuración del software
- Administración de la reutilización
- Preparación y producción del producto del trabajo

Modelado del Software

Comunicación

Planeación

Modelado

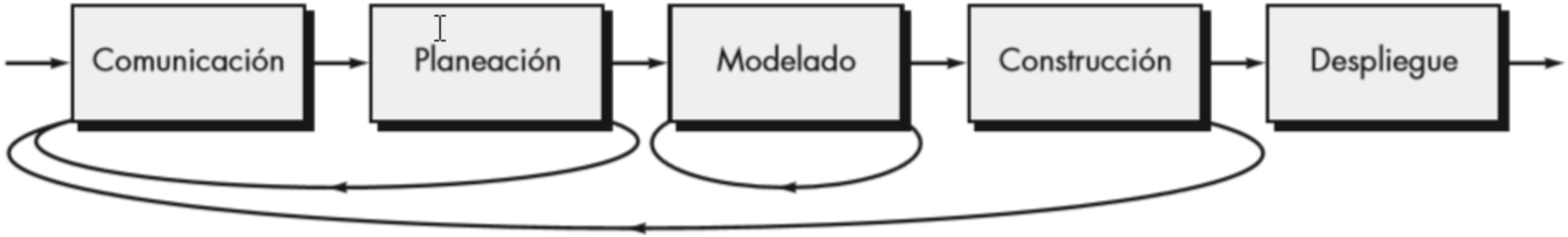
Construcción

Despliegue

Flujo del Proceso Lineal

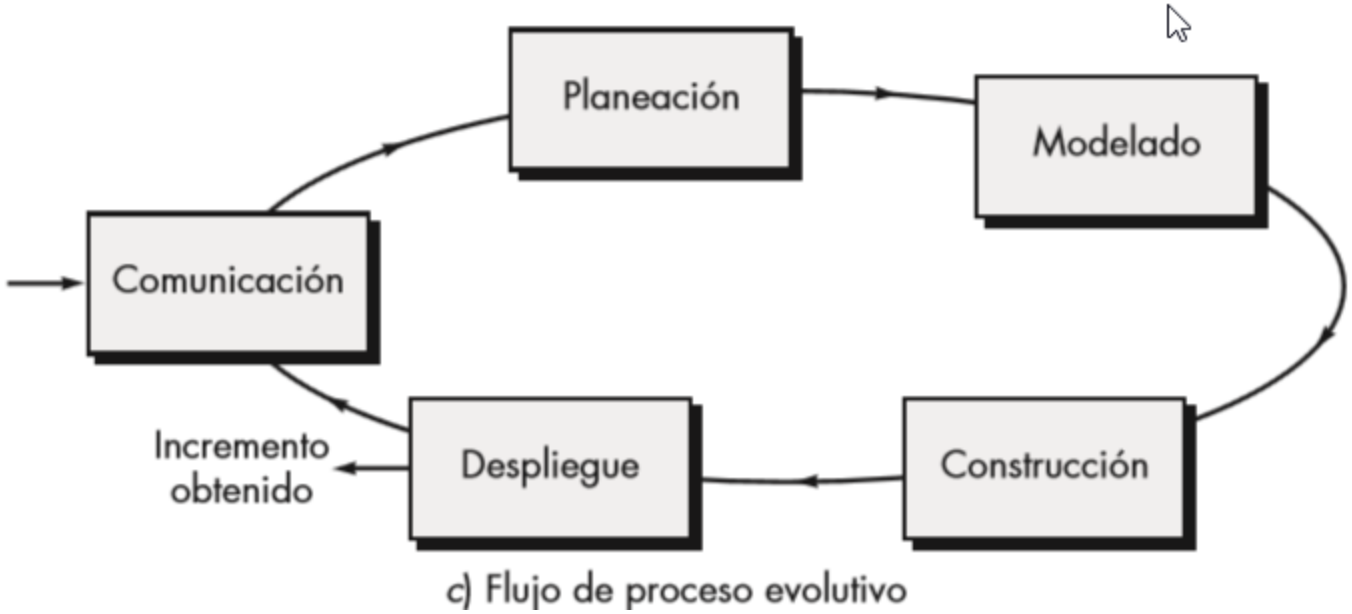
a) Flujo de proceso lineal

Flujo de Proceso Iterativo



b) Flujo de proceso iterativo

Flujo de Proceso Iterativo



c) Flujo de proceso evolutivo

Modelos de proceso software

Sommerville define el modelo de proceso de software como:
"Una **representación simplificada de un proceso de software**, representada desde una perspectiva específica. Por su naturaleza los modelos son simplificados, por lo tanto un modelo de procesos del software **es una abstracción de un proceso real**."
Los modelos genéricos no son descripciones definitivas de procesos de software; sin embargo, son abstracciones útiles que pueden ser utilizadas para explicar diferentes enfoques del desarrollo de software

Modelos de proceso software

Algunos modelos son:

1. Codificar y corregir
2. Modelo en Cascada
3. Desarrollo Evolutivo
4. Desarrollo Incremental
5. Desarrollo en Espiral