Ingeniería de Software

¿Qué es ingeniería de software?

Es una disciplina de la ingeniería que estudia todos los aspectos de la producción de software, desde las primeras etapas de su desarrollo hasta el mantenimiento del mismo una vez que está en ejecución.

Con *disciplina de ingeniería* nos referimos al trabajo de un ingeniero. Un ingeniero hace que las cosas funcionen. Aplican métodos, teorías y herramientas según convenga con el fin de resolver un problema, siempre cumpliendo con limitaciones de tiempo y económicas. A veces, para cumplir con los plazos, se deben sacrificar otros parámetros.

Con *todos los aspectos de la prod de software* nos referimos a la administración del proyecto, a los procesos técnicos de desarrollo de software, desarrollo de herramientas, métodos y teorías que sostienen la producción de software.

¿Por qué ingeniería de software?

El mundo actual está empapado de software y lo usa de forma intensiva. Dentro de los ejemplos encontramos servicios públicos, industrias musicales, cine y televisión, entretenimiento, etc. Por todo esto, la ingeniería de software es esencial para el funcionamiento de la sociedad. Gracias una buena ingeniería de software fue posible explorar el espacio, tener Internet, etc.

La ingeniería de software está destinada al desarrollo de software profesional y su venta. Para esto, plantea técnicas que favorecen la especificación, el diseño y evolución de un programa, cuestiones que no importan en el desarrollo de software personal.

La ingeniería de software no sólo hace referencia a los programas en sí, sino también a la documentación y datos de configuración necesarios para que los mismos funcionen, como por ejemplo archivos de instalación, guías para usuarios, sitios web donde usuarios pueden ver información actual del producto, etc.

Como vemos, la principal diferencia entre desarrollo de software profesional y aficionado, es que para el aficionado, sólo importan los programas, mientras que para el profesional juega un rol importante todo lo “extra”. La documentación y datos de configuración son esenciales para la vida de un software.

La calidad de un software está dada por no sólo su correcto funcionamiento, sino también por su documentación, datos de configuración, tiempo de respuesta, comprensibilidad de código. A su vez, la calidad de un software dependerá de la aplicación que este tenga, por ejemplo, un sistema bancario necesita ser seguro, mientras que un juego online, necesita respuesta rápida.

Tipos de productos de software

- Prod genéricos: una organización de desarrollo elabora un sistema independiente, el cual se vende a cualquier persona que desee comprarlo. Ej: aplicaciones para PC, como un navegador, un editor de texto, etc.
- Prod personalizados: un sistema está destinado a un cliente en particular.

Una diferencia importante entre tipos de productos de software es que en prod genéricos la organización desarrolladora controla la especificación del producto, mientras que en prod

personalizados la organización que compra el software desarrolla y controla la especificación. Sin embargo, hoy en día la diferencia es cada vez menos pronunciada, ya que muchos sistemas se piensan genéricos como base, y luego se adaptan según los requerimientos del cliente.

El software es un bien intangible. A diferencia de otros bienes, no encuentra restricciones de propiedades de materiales, físicas ni de procesos de fabricación. Esto es la causa de que, al no basarse en la física, pueden volverse complejos, difíciles de entender y costosos de cambiar. El software se mantiene y cambia a lo largo del tiempo.

Si bien hay distintos sistemas de software, todos van a requerir ingeniería de software, sólo que distinta según el caso.

Si bien es mucha la gente que hace programas para bien propio y específico, el desarrollo de software es una actividad profesional, donde uno de sus propósitos es el negocio. El **software profesional** no se desarrolla por un individuo, sino por un equipo y está destinado a usarse por alguien distinto al propio desarrollador. Un sistema de software profesional no consta sólo de programas.

<u>Problemas del software</u>

- Aumento de demandas: como la tecnología avanza rápidamente, los sistemas deben construirse y distribuirse muy rápido, a pesar de tener que ser complejos. Los sistemas que se basen en tecnología vieja, quedan inútiles.
- Expectativas bajas (poca fe en el sistema): los proyectos no aplican la ingeniería de software desde el minuto cero, probablemente porque no proyectan que las necesidades serán mayores y se conforman con un rendimiento justo frente a los requerimientos actuales.

El proceso de un software

En la elaboración de un software ocurren cuatro actividades troncales:

1) Especificación del software: se discute qué se va a hacer, se consideran requerimientos y restricciones.
2) Desarrollo del software: se diseña y programa.
3) Validación del software: se prueba el software y se ve que satisfaga al cliente.
4) Evolución del software: en caso de que el cliente lo desee, se modifica el software para adaptarlo a nuevas demandas.

<u>Variados tipos de software</u>

La ingeniería de software se basa en metodos y tecnicas para el desarrollo de software.

1) Apps independientes: aplicaciones que corren en una sola computadora. No es necesaria una red para comunicarse con otras computadoras. Ej: calculadora, editor de fotos, etc.
2) Apps interactivas basadas en transacción: hay una aplicación que corre en una computadora remota y los usuarios acceden desde sus propios equipos.
3) Sist. de control embebido: software que permiten el funcionamiento de hardware.
4) Sist. de procesamiento en lotes: sistemas, llenos de procesos batch, para grandes empresas.

5) Sist. de entretenimiento: sistemas que buscan entretener al usuario. La prioridad en estos sistemas es la calidad de interacción con el usuario. Ej: juegos.
6) Sist. de modelado y simulación: sistemas científicos que involucran mucho cómputo y operaciones en paralelo.
7) SIst. de adquisición de datos: sistemas que recopilan información de su entorno y la envian a otros sistemas para su procesamiento.
8) Sist. de sistemas: sistemas que requieran de otros.

Las barreras entre clasificaciones son poco sólidas. Las tecnicas de ingenieria de software a aplicar dependerá del tipo de sistema en cuestión. Sin embargo, independientemente del tipo de software que se trate, siempre se debe cumplir que:

- El equipo de desarrolladores debe planear el proceso de desarrollo, saber qué se va a hacer, cómo y para cuándo
- El software debe ser eficaz, eficiente y, de ser posible, seguro.
- Se debe cumplir con todos los requerimientos, de clientes y usuarios.
- Debe aprovechar recursos, incluyendo reutilizar software.

Caso 1: sist. embebido
el software controla y est'a en un dispositivo hardware. Problemas: tamanio fisico, tiempo de respuesta, uso de la energia, etc.

Caso 2: sist de informacion
El principal proposito es gestionar acceso a una base de datos de informacion, garantizando seguridad, integridad, usabilidad, etc.

Caso 3: sist de adquisicion de datos
sistema que tiene como objetivo recolectar datos de sensores y procesarlos, aun si las condiciones no son favorables.

Clase 1

(Def)Software

Según la IEEE es el conjunto de:

1-programas,
2-procedimientos,
3-reglas,
4-documentación y
5-datos asociados

involucrados en un sistema de computación.

Software personalizado (*¿particular?*): sistemas hechos para un cliente en particular.
Software genérico: sistemas hechos para un mercado abierto.

Ejs de tipos de software: basados en la Web (sitios Web), científicos (cálculo numérico), de gestión (info comercial), etc.

El software no es un bien material, es lógico. No se fabrica, se desarrolla. Lo más caro del desarrollo es la ingeniería, no la producción.

El software es un bien que no se desgasta. No sufre el paso del tiempo, sino los cambios.

(Def)Ingeniería de Software

Es la ingeniería (hace que todo funcione) que estudia todos los aspectos de la producción de software (ej: gestión del proyecto), desde las primeras etapas de su desarrollo hasta el mantenimiento del mismo una vez que está en ejecución. El software debe desarrollarse de forma correcta dentro de un tiempo y costos estipulados. Para conseguir este objetivo tan difícil la ingeniería de software cuenta con métodos sistemáticos y estándares para organizar eficientemente el trabajo.

Sobre el ingeniero de software

Debe conocer del campo en el que está el software a desarrollar, debe saber de técnicas de trabajo, junto con sus ventajas y desventajas, para entender cuándo aplicarlas, así como también saber administrar proyectos. Un ingeniero de software debe combinar conocimientos de ciencias de computación (teóricos y prácticos) junto con problemas de un cliente para desarrollar el sistema que dicho cliente demanda.

Limitaciones del Ing

A veces las limitaciones que puede encontrar un ingeniero de software son de carácter económico, social y legal.
Debe obrar éticamente, esto es:

-La información entre Ing y cliente es confidencial
-Saber hasta dónde llegan sus conocimientos, si es competente con el proyecto
-Conocer sobre derechos de propiedad intelectual, como patentes.
-No sacar ventajas con mala intención de sus conocimientos.

Su trabajo no debe sobrepasar estos límites.

(Def) Proceso de software

Un proceso de software es el conjunto de las actividades (y sus resultados) que producen un producto de software.
Todo proceso tiene 4 actividades importantes:

1-Especificar qué se va a hacer
2-Diseñar cómo se va a hacer
3-Una vez hecho asegurar que el software satisfaga al cliente.
4-Mantener el software (con o sin cambios).

(Def)Modelo de proceso (o ciclo de vida del software)
Un modelo de proceso es una forma simplificada de entender un proceso.
Modelos básicos de procesos:
-Cascada
-Iterativo
-IS basada en componentes
Los modelos de proceso buscan a que el software que se desarrolle mediante determinadas acciones sea de calidad: debe poder ser modificado, ser confiable, eficiente y fácil de usar.
Los modelos pueden ser: prescriptivos, visión teórica de un proceso, o descriptivos, visión real de un proceso.

Modelo en cascada
Se toman los requerimientos, se diseña el software, se implementa, se prueba que funcione correctamente, se activa y mantiene. El modelo en cascada asume que cada etapa inicia sólo cuando la anterior se concreta. Se considera que el flujo de etapas es lineal. Errores en etapas avanzadas provocan que se descarte trabajo, se pierda tiempo, se replanteen las bases. No responde bien a cambios. Es un modelo ineficiente, pero muy prolijo a la hora de saber qué etapas debe tener un proceso de software (no cómo).

Desarrollo por partes
Se diseña una implementación base, se consulta al cliente, y se siguen construyendo versiones hasta alcanzar el resultado esperado. Mientras una parte del sistema está en uso, se puede seguir construyendo otra.
Hay dos formas de desarrollo por partes:
1-Incremental: se agregan partes del sistema, a medida que se tienen.
2-Iterativo: el sistema tiene de entrada todas sus partes, sólo que funcionan las implementadas.

Modelo en V
Se plantean los requerimientos --- --- --- Se los valida con el cliente
Se diseña el sistema --- --- Se prueba el sistema
Se diseña el programa --- Se prueba
Codificación
Comienzo por el nivel 3 y desciendo sólo cuando consigo validar correctamente. Si una validación no se concreta, permanezco en el nivel actual.

Modelo de prototipos
Se hacen versiones incompletas del sistema con el fin de corroborar requerimientos o funcionamiento con el cliente.
Un prototipo puede ser evolutivo, si el sistema va ir corrigiendo errores a lo largo del tiempo, o descartable, si es sólo para tener una idea de que estamos haciendo las cosas bien.

Un prototipo no debe ser costoso, debe dejar que el usuario experimente, debe ser de rápida construcción y debe ser construido por pocos

Modelo en espiral

Es el más completo y complejo. Permite eliminar errores al inicio. Trata cuestiones de calidad y riesgos.

(Def)Metodologías ágiles

Se busca construir software de forma rápida, incremental/iterativo y simple. Para esto se requiere el constante contacto con el cliente. Los requerimientos y soluciones evolucionan a la par gracias al trabajo en grupo.

Una iteración es un software desarrollado en un tiempo determinado. Una iteración puede durar hasta un mes. A fin de cada iteración, las metas propuestas deben estar alcanzadas.

Promueven el trabajo de todos los miembros de un equipo. Los problemas de un miembro pueden resolverse por otro.

XP (eXtreme Programming)

Se programa en parejas. No se avanza hasta no solucionar todos los errores.

Planeación con usuario y plan de iteración
Diseño y prototipado
Codificación (en pareja)
Prueba
Lanzamiento

Scrum

Mejora la manera de trabajar en grupo. Método iterativo, donde las tareas se organizan por la prioridad que determina el cliente.

Roles en Scrum:

-Product Owner: decide las prioridades del proyecto
-Scrum Master: conoce la técnica y garantiza que se desarrolle completamente.
-Scrum Team: personas que llevan a código el proyecto a desarrollar
-Otros: usuarios (son importantes porque pueden aportar ideas o necesidades)

Product Backlog: lista de todos los requerimientos que debe cumplir el sistema
Sprint Backlog: lista de requerimientos que van a cumplirse en una iteración
Burndown chart: trabajo hecho día a día.
Scrum es ideal para proyectos dinámicos y difíciles.

Modelos PIM, PSM

PIM: modelo independiente de la plataforma
PSM: modelo de plataforma específica

Estos modelos permiten una mayor abstracción de un problema e independencia en las soluciones, mejoran la productividad, resisten cambios.

Preguntas Clase 1:

1)¿Entra historia de la ingeniería de software? Todo lo que fue la crisis del software, la OTAN, etc...

2)En lo referido a un "proceso de software"... ¿puede reemplazarse "producto de software" por "software"?

3)¿Podría tomarse como válida la definición de modelo de proceso?

4)¿Desarrollo por partes es sinónimo de desarrollo por fases?

5)¿Jerarquía, en cuanto a eficiencia, de modelos? Repasar ventajas y desventajas.

6)Los modelos PIM, PSM, etc, ¿son cosas independientes? ¿qué es MDD? ¿PIM, PSM están incluídos en MDD?

Clase 2

Al iniciar un proceso debemos conocer del cliente **para cuándo**, **qué**, **por qué**, **cómo y lo quiere**. Para conseguir esta información es necesaria la COMUNICACIÓN, que es la principal fuente de error en un proyecto.
Las *necesidades* en un proyecto, en términos técnicos se traducen a *requerimientos*.
Un *requerimiento* es una característica del sistema o descripción de algo que el sistema debe hacer, para cumplir con el objetivo de dicho sistema.
¿De dónde obtenemos los requerimientos?

1) de documentación
2) de sistemas similares
3) de stakeholders: cualquier persona o grupo que se relaciona con el sistema de alguna forma.

Los stakeholders se clasifican según su punto de vista como:

1) interactuadores: relación directa con el sistema
2) indirecto: no usan el sistema pero aportan requerimientos
3) dominio: restricciones del dominio que influyen en los requerimientos

(Def)Elicitación de requerimientos

Proceso de adquirir toda la información necesaria para modelar los requerimientos de un problema. Es un proceso más social que tecnológico. Intervienen los problemas de comunicación.
Técnicas de elicitación:

- Análisis de información existente: conocer los orígenes de un proyecto, errores pasados, documentos, etc.
- Visitas al sitio: conocer software similar
- Observar el trabajo: conocer dificultades de usuarios
- Cuestionarios: abiertos o cerrados. Recolectar información masivamente y de forma general

- Entrevistas: alcance reducido. personales. Conocer opiniones del entrevistado. Preguntas y respuestas con un objetivo específico. Costosa (en tiempos y recursos humanos). Estructurada (triángulo, embudo, diamante)
- Brainstorm: procedimiento para obtener ideas que ayuden a solucionar un problema. Se promueven soluciones creativas. Consta de etapas: se proponen ideas (mientras más mejor), se agrupan ideas, se eliminan los grupos que no solucionan el problema, se ordenan los grupos por prioridad.

Preguntas Clase 2
1)¿Entra JRP (Pág.53)? ¿Qué es?

Clase 3

Errores en la toma de requerimientos tienen alto costo a la hora de solucionarlos, dependiendo de qué tan avanzado esté un proyecto.

(Def)Ingeniería de requerimientos
Proceso que transforma las necesidades de un cliente en requerimientos.

6 Características de un requerimiento
-Necesario: sin él el software no funciona
-Conciso: se lee y se entiende
-Completo: no hay que aclarar nada más
-Consistente: no contradice a otro
-No ambiguo: única implementación
-Verificable: puede testearse

Un buen análisis de requerimientos facilita la planificación y disminuye errores de un proceso.
En base a los requerimientos se puede decidir si se encara y de qué forma un proyecto, o no.

Los requerimientos pueden ser funcionales o no funcionales. Los funcionales son los que marcan qué debe hacer un sistema, hablan de su funcionalidad. Los no funcionales son restricciones que tiene un sistema, pueden ser restricciones ajenas al sistema

ERS: documento donde se especifican los requerimientos de un sistema.

Cuando se desarrolla el sistema es necesario verificarlo y validarlo. Verificar es ver que el sistema esté correcto, que no falle. Validar es ver que el sistema cumpla con lo que el cliente desea.

Gestión de requerimientos

Existen requerimientos duraderos y volátiles. Los que demandarán más atención son los volátiles.

Técnicas de especificación de requerimientos: estáticas, donde se muestra relación de objetos (Ej: definición axiomática, como los TAD en Pascal), y dinámicas, donde se muestra un sistema y sus respuestas a estímulos (Ej:tablas de decisión, máquinas de estado finito, redes de Petri).

Tablas de decisión

Tenemos CONDICIONES, ACCIONES y REGLAS
Las condiciones sólo toman valor Verdadero o Falso (1 o 0 respectivamente).
Hay el doble de reglas que condiciones.

Las variables son las condiciones, las salidas son las acciones. *¿Y las reglas?*
Hay redundancia si para iguales condiciones tengo iguales acciones
Hay contradicción si para iguales condiciones tengo distintas acciones

Ejercicio Tabla Decisión

En el sistema de un videoclub se quiere modelizar el subsistema de alquileres. Sólo se permite alquilar películas si la copia de la película está disponible. Un socio puede tener en su poder 3 películas. Si el socio tiene 3 películas en el momento de alquilar otra película, no se permite alquilar la película. Si tiene películas vencidas sin devolver, no se le permite alquilar y se le cancela momentáneamente la cuenta. Si no está disponible la copia de la película, no tiene 3 películas en su poder y no tiene alquileres vencidos se le reserva la película

Condiciones:
A: Copia de película está disponible
B: Socio tiene 3 películas en su poder
C: Socio tiene película vencida

Acciones:
Z_0: Se alquila película
Z_1: Se cancela la cuenta
Z_2: Se reserva película

A	B	C	Z_0	Z_1	Z_2
0	0	0	0	0	1

0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	0	1	0
1	0	0	1	0	0
1	0	1	0	1	0
1	1	0	0	0	0
1	1	1	0	1	0

Debido a la sencillez de la tabla, no hace falta realizar ningún diagrama de Karnaugh para ver que:

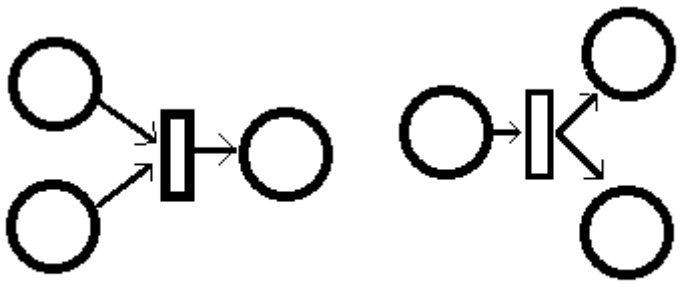
- Se alquila película sólo si hay copia de película disponible, el socio no tiene vencida alguna y no tiene 3 en su poder.
- Se cancela la cuenta momentáneamente siempre que el socio tenga película vencida
- Se reserva película si al momento del pedido el socio no tiene 3 películas en su poder, no tiene películas vencidas y no hay copia de película disponible.

Máquinas de estado finito

Describe al sistema como un conjunto de estados, donde el paso de uno a otro (transición) se da con la llegada de un estímulo.
Para realizar el esquema:
1)Definir estados (si alguno es complejo puede dividirse)
2)Partir de un estado inicial y conectarlos mediante flechas
3)Determinar condiciones que llevan de uno a otro
4)Verificar que pueda llegar a todos los estados, ver que no pueda quedarme atrapado en un estado. Debería poder controlar todos los estados posibles.

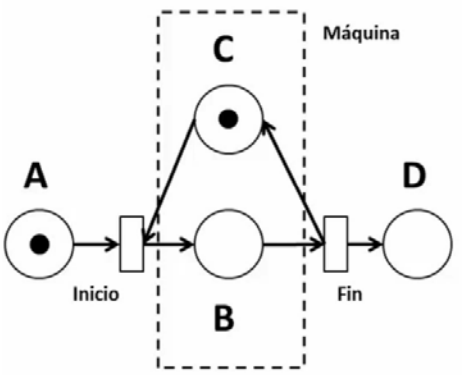
Redes de Petri

Consiste en un diseño de sistema que se basa en cuatro símbolos: plaza(estado), transición(evento. relacionan eventos), flecha(indican el sentido del flujo), token(se ubican en las plazas). Hay un evento cuando un token viaja de una plaza a otra por medio de una transición. Un token sólo puede viajar de una plaza a otra por medio de una transición.



Sincronización Y **Concurrencia**

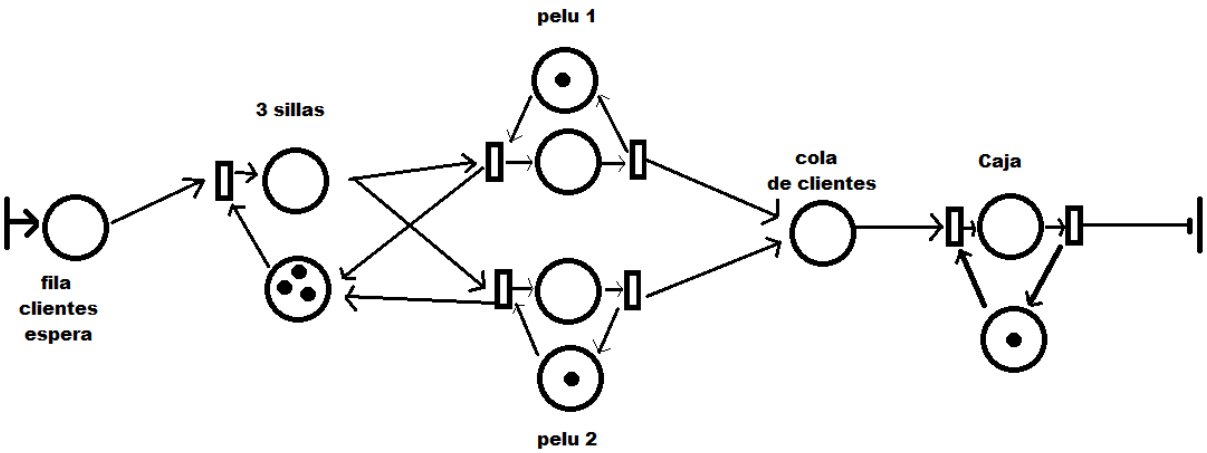
La sincronización apunta a que dos eventos a la vez generan una transición
La concurrencia apunta a uno o más eventos que parten de un único estímulo



Este esquema garantiza que A deba esperar que B esté libre para poder pasar. Es una combinación de las dos anteriores: sincronización y concurrencia.
En redes de Petri no hay un orden de eventos. Son asincrónicas.

Ejercicio red Petri

Dos peluqueros trabajan en una peluquería. La peluquería cuenta con una sala de espera con sólo 3 sillas para que los clientes esperen por ser atendidos. Cuando alguno de los peluqueros se libera atiende a uno de los clientes de cualquiera de las sillas para cortarle el cabello, liberando la silla de la sala de espera, para que se siente un nuevo cliente. Una vez que terminó de cortarle el cabello el peluquero es liberado y puede atender a otro cliente. Finalmente los clientes deben pasar por la caja en la cual se atiende a un cliente por vez.
NOTA: cuando llegan personas y las tres sillas están ocupadas forman una única fila en la puerta de la peluquería.



Las 3 sillas funcionan como una gran máquina sincrónica y concurrente.

Preguntas Clase 3

1)¿Qué son las técnicas de especificación de requerimientos? Me hace ruido el término “especificación”
2)¿Qué serían las reglas? ¿Está aprobada la forma de explicar y hacer una tabla de decisión?
3)En redes de Petri, ¿Cómo puedo hacer para diseñar una máquina que pueda almacenar hasta 2 tokens, partiendo de la idea de la sincronización y concurrencia?