SOFTWARE AS AN ENGINEERING DISCIPLINE
ASW SEMINARIO

Sara Inés Bolado – UO277494

Andrea Fuertes Carral – UO276299

Marcos González García - UO282587

INTRODUCCIÓN

- Diferencia entre ingeniería de software y desarrollo
- Importancia del diseño y la planificación
- Tipos de complejidad
- Diseñar para el cambio: la clave de un software sostenible
- Diseño iterativo y el costo de la revisión tardía
- Refactorización y retrabajo (rework)
- Calidad: un trabajo de todos
- Rol de Ingeniero Jefe
- Expectativas de los consumidores
- Aplicar los principios de la ingeniería de software a nuestro contexto

DIFERENCIA ENTRE INGENIERÍA SOFTWARE Y DESARROLLO SOFTWARE

- Parte 'científica' vs Parte 'aplicada'
- RIGOR
- Fácil ponerse a escribir código y hacer una web...
- ... pero ¿realmente funciona?

IMPORTANCIA DEL DISEÑO Y LA PLANIFICACIÓN

- Mundo ágil -> resistencia a principios rigurosos
- Ignorar documentación
- No planificar
- Ir directamente al código

Error -> costoso cambiar el rumbo

- Estimar tiempo y costes
- Diseñar
- Pensar y planificar
- Documentar

Acierto

TRES TIPOS DE COMPLEJIDAD

- Complejidad objetiva: objetivos de los clientes **¿requisitos = lo que quieren los clientes?**
- Complejidad de requisitos -> mejorada gracias a metodología ágil
- Complejidad de solución: cómo se estructura el software

Necesitamos **CONTROL**, gestionar complejidad

Aprender a controlar cambios

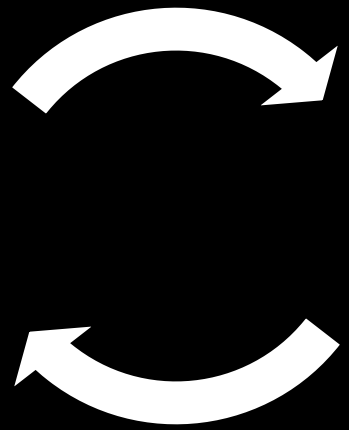
Ingeniería del software -> hacer sistemas mantenibles que puedan crecer

DISEÑAR PARA EL CAMBIO

- Clave de un Software Sostenible
- Sistemas escalables y manejables
- Comparativa → cambio de un calentador
- Modificar sin complicaciones

DISEÑO ITERATIVO Y COSTO DE LA REVISIÓN TARDÍA

- El diseño del software no es lineal
- Retroalimentación de otros ingenieros
- Comparativa con construcción
- Hay que ajustar el diseño durante el proceso



REFACTORIZACIÓN Y RETRABAJO

- Refactorización ➜ mejorar el código sin cambiar su comportamiento
- Rework ➜ cambios mayores (mal diseño inicial)
- Un buen diseño facilita la refactorización
- Los hotfixes no resuelven problemas a largo plazo
- El diseño inicial debe permitir la evolución

CALIDAD

- Importante integrarla en cada parte del proceso
- Todo el equipo es responsable
- Control de calidad → evaluación después
- Aseguramiento de calidad → integración desde el inicio

EL ANDON CORD DE TOYOTA

- Cuerda que tenían en las fábricas de Toyota
- Cuando alguien detectaba un problema, tiraba de la cuerda
- Se paraba la cadena de producción y todo el mundo colaboraba para resolver el problema
- La producción no continuaba hasta resolver el problema

EL ROL DE INGENIERO JEFE

VISIÓN GENERAL

CONOCIMIENTO AMPLIO DEL ÁMBITO

CAPACIDAD DE ADAPTACIÓN

MAYORES EXPECTATIVAS

- El paso del tiempo implica mayores exigencias de los usuarios
- Mayores exigencias implican más calidad y funcionalidad requerida en el software
- Los mínimos permitidos han aumentado mucho los últimos años

APLICAR PRINCIPIOS DE LA INGENIERÍA DEL SOFTWARE A NUESTRO CONTEXTO

- Contratar buenos líderes
- Invertir tiempo en entrenar a nuevas incorporaciones