Degradación de Código

Antonio Fontaine, Agustin Gomez,
Diego Sinay

Estructura

- Definición
- Software Entropy
- Technical Debt
- Feature Creep
- Software Bloat
- Software Brittleness
- Otros tipos de degradación de código

Degradación de código

Deterioro del rendimiento del *software* a medida que pasa el tiempo

El *software* comienza a tener fallas, ser inutilizable, o se convierte en “legacy”

El *software* no se deteriora per se, sino que deja de ser mantenido con respecto a un ambiente cambiante

Software Entropy

2da Ley de la Termodinámica:

- La cantidad de entropía del universo tiende a incrementar en el tiempo

Ivar Jacobson(UML,RUP) propuso lo siguiente en relación a softwares

- Cuando un sistema es modificado, su entropía siempre aumenta

Software Entropy

Manny Lehman extendió el concepto a estas dos leyes.

- Un software que es utilizado se modificará
- Cuando un programa es modificado, su complejidad aumentará, al menos que se trabaje activamente contra esto

Software Entropy Control

Andrew Hunt y David Thomas hacen una analogía con la teoría de las ventanas rotas

- Prevenir pequeños problemas en un entorno ayuda a crear una atmósfera que previene problemas mayores

Lección → No vivir con ventanas rotas

- No dejar malos diseños, malas decisiones o mal código sin arreglar

Technical Debt

El trabajo extra de desarrollo que surge cuando se utiliza código fácil de implementar en vez de la mejor solución

No necesariamente es algo malo, ya que mueve los proyectos hacia adelante

La “deuda técnica” va aumentando “cobrando intereses” a medida que avanza un proyecto

4 Tipos de Technical Debt

	Inconsiderado	Prudente
Deliberado	**“No tenemos tiempo para el diseño”**	**“Debemos entregar ahora y lidiar con las consecuencias”**
Inadvertido	**“¿Qué es *layering*”**	**“Ahora sabemos como lo teníamos que hacer”**

Causas y consecuencias de Technical Debt

Causas:

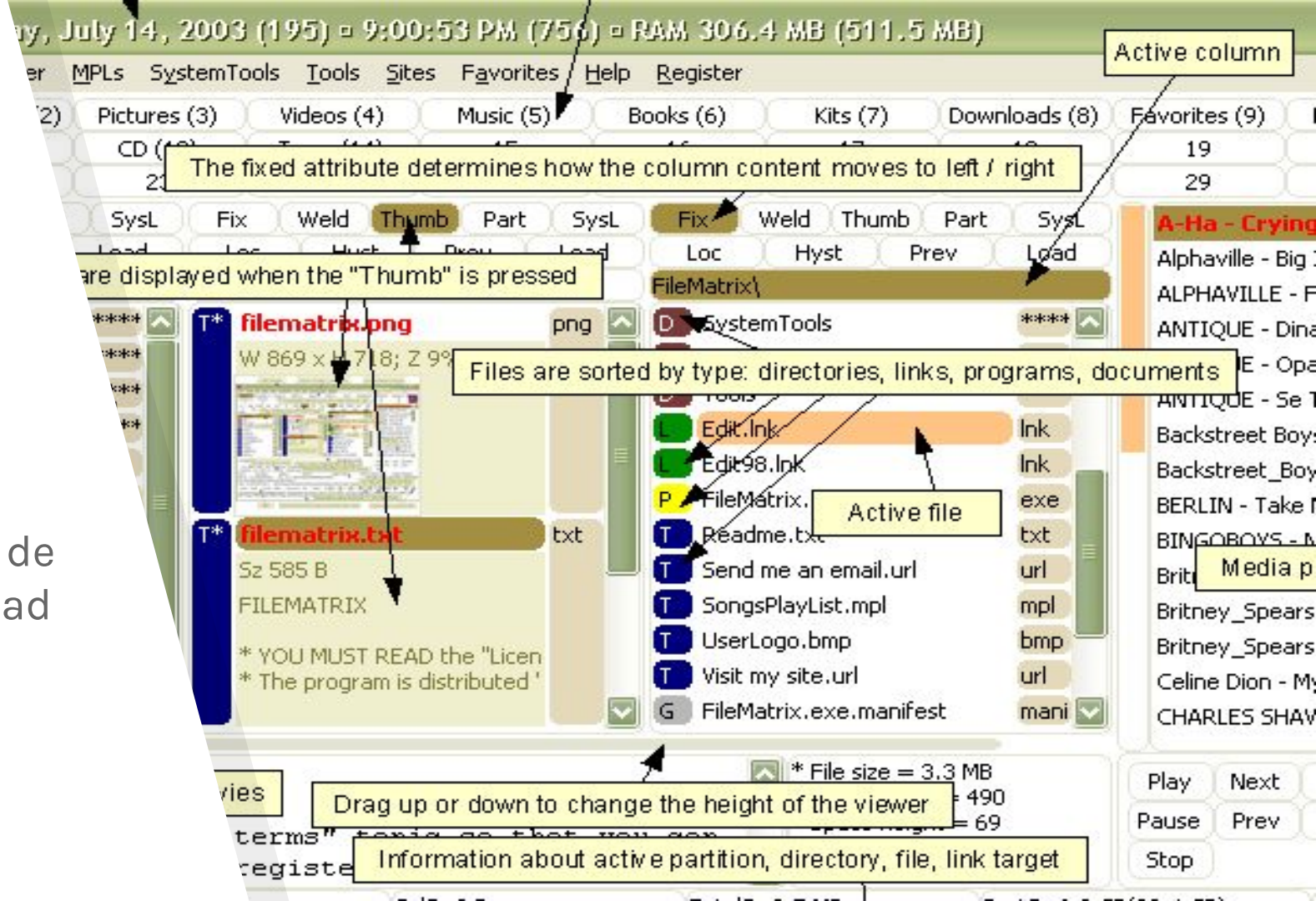
- Presiones de negocio
- Falta de entendimiento

Consecuencias:

- Entregas atrasadas por tener que pagar la “deuda”
- Dificulta el desarrollo

Feature Creep

Expansión desmedida de funcionalidades en un producto.



Feature Creep

Causa:

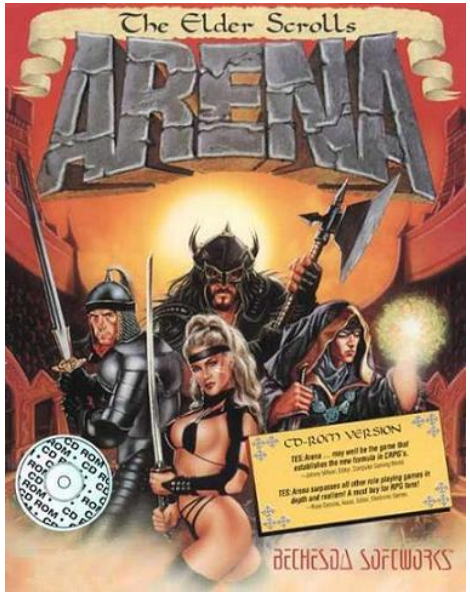
- Clientes similares pero con requerimientos sutilmente distintos.
- Intención desmedida de diferenciarse de la competencia o de sacar nuevas versiones.

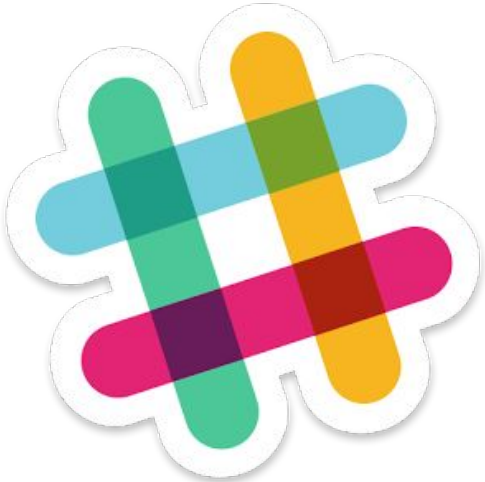
Feature Creep

Consecuencias:

- Retrasos
- Confunde al usuario
- Código lleno de “parches”
- Cambio del plan original

Feature Creep





Feature Creep

Soluciones:

- Tener distintos productos (versiones)
- Refactoring
- Ordenar features por importancia

Software Bloat

Versiones sucesivas de programas son más lentas, usan más recursos, o tienen mayores requerimientos sin mejoras sustanciales.

También es usado como *software* preinstalado o programas empaquetados no deseados

Ejemplos de Software Bloat

Apple’s iTunes

- pasar de un reproductor de música a un e-commerce y plataforma de publicidad

Microsoft Windows

- Compatibilidad con versiones anteriores
- Programas pre-instalados

Software Brittleness

La dificultad de arreglar *software* antiguo que parece confiable pero falla con data inusual o con pequeños cambios.

Como un metal, el *software* es maleable al principio y cuando crece pasa ser más frágil y difícil de mantener sin quebrar todo el código.

Causas de Software Brittleness

- Usuarios resistentes al cambio
- Cambios en el equipo de desarrollo
- Arreglos parches
- Dependencias muy rígidas

Otros tipos de degradación

- Spaghetti code
- Big Ball of Mud
- Dependency Hell

Spaghetti Code

- Programas con una estructura de control de flujo compleja e incomprensible

- El flujo es como un plato de tallarines, enredado y retorcido.

Big Ball of Mud

- Un sistema de *software* que no tiene una arquitectura clara

Dependency Hell

- *Software* con dependencia a paquetes específicos
- Muchas dependencias
- Dependencias circulares
- Dependencias conflictivas

¿PREGUNTAS?