SOFTWARE LEGACY

Carlos Fontela

Frase del día

The death of a program happens when the programmer team possessing its theory is dissolved. A dead program may continue (...) to produce useful results. The actual state of death becomes visible when demands for modifications of the program cannot be intelligently answered. Revival of a program is the rebuilding of its theory by a new programmer team.

(Peter Naur, Programming as Theory Building)

Software Legacy

... o software "heredado"

Difícil de mantener

- Difícil de extender,
- arreglar,
- mejorar

¿Antiguo o tecnológicamente atrasado?

- No necesariamente...

Por qué estudiarlos

Es un activo fundamental de las organizaciones

- *Por dinero invertido*
- *Por conocimiento embebido*

Mantenimiento costoso y riesgoso

Sistemas que requieren cambios constantes

=> círculo vicioso difícil de romper

Qué tipo de “legacy” nos interesa

Son exitosos

- *Hacen lo que quieren los usuarios*
- *Sin errores*

Evolucionan

- *… pero el costo de cambiarlos es alto*

Definición:

Sistemas de software que han sido desarrollados y han alcanzado un grado de éxito relativo, pero resultan difíciles de mantener porque se ha perdido el conocimiento que permitiría hacerlo.

Por qué se convierte en “legacy”

Desarrollados por equipos que ya no están en el proyecto (productos huérfanos).

No hay pruebas de aceptación ni técnicas que permitan probar el funcionamiento del código o entender cómo funciona (código inasequible).

Escasa documentación funcional, técnica y para usuarios (productos “flojos de papeles”).

Muchas funcionalidades desconocidas por los desarrolladores y usuarios (productos tipo “caja de sorpresas”).

Mantienen un gran volumen de deuda técnica que impide probarlos y comprenderlos.

La visión de tirar y reconstruir

Pros

Dificultad de cambiar el software

- *Más directa*
- *Deja un producto presuntamente adecuado*

Contras

Muy costosa

Se suele perder conocimiento almacenado en el código

... y algunas funcionalidades que pueden ser necesarias

La visión de Peter Naur (1985)

Programar es construir conocimiento

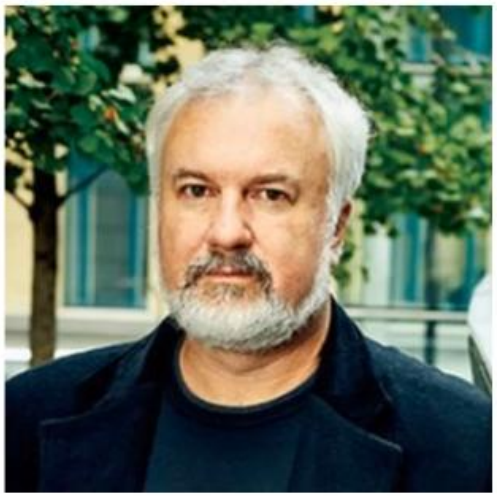
Situación: una persona que no trabajó en el desarrollo se incorpora al equipo

- *No alcanza con que pueda comprender el código*
- *Imposible restablecer el conocimiento reunido durante el desarrollo mediante la simple lectura de textos*
- *Podría haber personas que estén en posesión del conocimiento*

Si no queda nadie, el producto debe considerarse muerto

- *Lo único que queda por hacer es intentar revivirlo / resucitarlo*
- *... o recuperarlo con otro equipo de trabajo*

La visión de Feathers



Código legacy es el que carece de pruebas

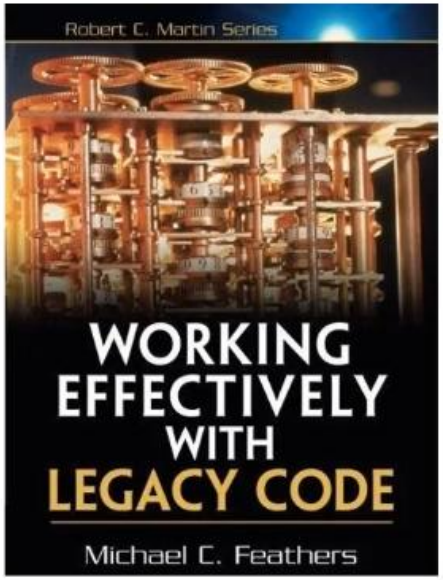
> *Se refiere a pruebas unitarias automatizadas*

Crea un catálogo de escenarios típicos

Recetas para ir rompiendo dependencias en el código mediante la introducción de pruebas automatizadas que lo cubran

Feathers es menos drástico que Naur: sugiere que lo que debemos reconstruir son las pruebas que nos dan pistas sobre cómo es el comportamiento del programa

> *En línea también con el planteo de Kent Beck: uno de los objetivos de las pruebas unitarias en código es brindar información sobre el comportamiento*



Recuperación del software heredado: una definición

Es la acción o conjunto de tareas que permiten llevar un producto de software que se ha vuelto imposible de mantener a uno que vuelve a ser posible de modificar, principalmente mediante la reconstrucción del conocimiento contenido en él.

Carlos Fontela, 2015

Naur diría “revivir el software”.

Algunas (buenas) viejas ideas

Mejorar la mantenibilidad del sistema a la vez que se mantiene la funcionalidad heredada.

Proceder con un enfoque incremental de reemplazo del software heredado por el nuevo.

Comenzar por las partes que requieren cambios

... que suelen ser las más utilizadas

Desde lo arquitectónico, envolver comportamiento observable con capas que expongan interfaces accesibles a las nuevas partes del sistema.

Algunas (buenas) propuestas de la comunidad ágil

Feathers es el más conocido

Stevenson & Pols: enfoque más centrado en la gente que en los documentos:

- *Con usuarios*
- *Con herramientas que comparen resultados entre el sistema anterior y el migrado*
- *Con pruebas de aceptación automatizadas introducidas durante el proceso*

Storytest-Driven Migration (STDM) - Universidad Aldo Moro de Bari:

- *Trabajo con pruebas de aceptación (especificaciones ejecutables)*
- *Deben funcionar tanto en el sistema heredado como en su reemplazo*
- *Revisión de las especificaciones por todos los interesados*

Recapitulación

El software legacy es un activo de las organizaciones

Se convierte en legacy por la pérdida de conocimiento

Naur habla de software muerto

Hay ideas de maneras de recuperarlo