Programación Orientada a Objetos

Tema 5. Reutilización del Software

POO: TEMAS DE TEORÍA

TEMA 1: ABSTRACTION & SOFTWARE DESIGN

TEMA 2: SOFTWARE DE CALIDAD

TEMA 3: DESCOMPOSICIÓN MODULAR

TEMA 4: TDD vs OOD. ESPECIFICACIÓN E IMPLEMENTACIÓN

TEMA 5: REUTILIZACIÓN

TEMA 6: 4 PILARES DE LA TECNOLOGÍA OO

TEMA 7: PATRONES DE DISEÑO

Programación Orientada a Objetos

Tema 5. Reutilización del Software

Reutilización del Software

- Desde los 60's se habla de reutilizar software
- En la actualidad se tiende cada vez más hacia *“el desarrollo de software como industria basada en componentes”*
- La reutilización veremos que:
  - Beneficia creadores
  - Beneficia consumidores
  - Beneficia la calidad del producto final

La reutilización beneficia a toda la industria

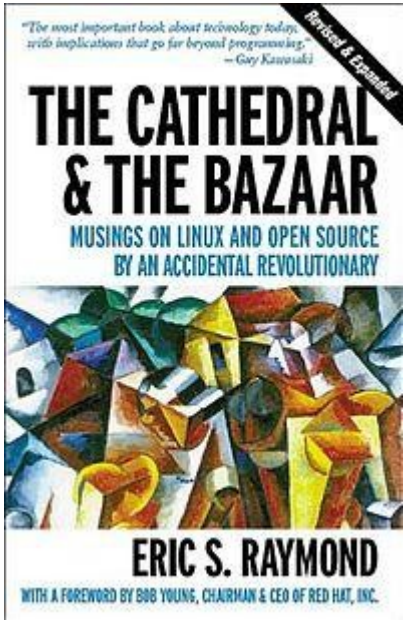
Beneficios (esperados)

Los beneficios más destacables

1. Rapidez de desarrollo:
   - Los plazos de entrega y la “oportunidad” de nuestro software mejorará
2. Fiabilidad
3. Eficiencia
4. Menor mantenimiento
5. Mayor consistencia en nuestros desarrollos
   - Pensarlo y diseñarlo con idea de que van a ser reutilizados nos hará hacer mejores piezas de software
6. Mejora costes. Y es una inversión rentable
7. Se espera la mejoría de **todos los factores de calidad, de y todos criterios, reglas, principios de descomposición modular**

Mejora de la calidad

The essay's central thesis is Raymond's proposition that "given enough eyeballs, all bugs are shallow" (which he terms Linus's law): the more widely available the source code is for public testing, scrutiny, and experimentation, the more rapidly all forms of bugs will be discovered.



No estamos hablando de copiar y pegar el código,
eso **no es suficiente.**
El acceso al código fuente **no es suficiente**

Productor - Consumidor

La reutilización tiene 2 aspectos:

- Producción (creación, autoría)
  - Desarrollar componentes es una disciplina difícil
  - Antes de desarrollar hay que consumir y aprender
- Consumo
  - Estudia, aprende, ….. e imita:
    - Estilo
    - Diseño
    - Propiedades de un componente
    - Etc...

Cómo y qué reutilizar

- El personal de desarrollo (no propiamente software)
- Metodologías de programación:
  - POO: objetos/clases, herencia, polimorfismo, templates, etc.
  - También en otras metodologías
- Los diseños
  - **Patrones de diseño** (los estudiaremos en un tema posterior)
- Se hacen mejoras constantes en estructuras de reutilización:
  - En lenguajes de programación
  - Repositorios
  - Colecciones de componentes
  - Comunidades de desarrolladores

La repetición

En una presencia habitual durante el desarrollo

- Detectarla (también factorizando comportamientos comunes)
- Entenderla en su sentido abstracto
- Describirla bien
- Para, a continuación, **desarrollarla como módulo**
- Diseñarla:
  - Desde cero.
  - O teniendo en cuenta la posibilidad de usar uno o varios **patrones de diseño**

Módulo ideal para la reutilización

- Módulo
- Desempeña una única tarea, clara, concreta y bien descrita
- Suficiente nivel de abstracción que permita su instanciación en otros problemas: oculta detalles irrelevantes
- Facilidad de uso
- Tamaño razonable (más bien pequeño)
- Debe ser buena pieza de desarrollo en si misma (de buena calidad)
- Debe cumplir los factores de calidad
- Debe cumplir los criterios, reglas y principios de una **buena descomposición modular**

Obstáculos

- Falta de formación
- Falsos mitos:
  - “Mejor hacerlo nosotros”
  - “Es más barato hacerlo nosotros”
  - “Nosotros todo, para que el cliente dependa de nosotros”
- Fallo en el diseño y gestión de un buen repositorio de componentes

Más obstáculos

- Reticencias a la reutilización más allá de nuestro propio equipo de desarrollo
  - Desconfianza de otros desarrollos
  - Reticencias a distribución del fuente
  - Muchas veces el esfuerzo es la interfaz no los módulos

Comunidades de desarrolladores

- Ventajas:
  - Simplifican el desarrollo
  - Detección/corrección de errores y mantenimiento
  - Portabilidad a otras plataformas
  - Mejora de la funcionalidad
  - Mejora de la interfaz
  - Mayor difusión… clientes
  - Aprendizaje, etc.

Estructuras de reutilización

- Rutinas, bibliotecas de rutinas o librerías
  - Agrupación de unidades (función, subrutina, subprograma)
  - Adaptables solo mediante argumentos
  - Problemas muy concretos y complejos pero de reducido tamaño
  - … necesitamos más estructuras de reutilización...

no incorporan elementos potentes de reutilización,
no facilitan la derivación de nuevos trabajos

Estructuras de reutilización

- Clases
  - Las clases son piezas modulares reutilizables por definición
  - Una o varias pueden formar un módulo o librería
  - Integradas en la tecnología OO: herencia, polimorfismo, etc.

Estructuras de reutilización

- Paquete
  - Más avanzado que rutina
  - Utilidades de todo tipo sobre un tema
  - Rutinas, variables, tipos, declaraciones, espacios de nombres, distintos componentes, etc.
  - También permiten un uso parcial de su funcionalidad
  - Solución completa o amplia a un problema
  - Ofrecen poco más allá de su funcionalidad
  - Ej: visión artificial, estadística, control de usuarios, encriptación, etc.

Estructuras de reutilización

- Framework/entorno de desarrollo
  - Facilitan el desarrollo de cualquier aplicación (dentro de su ámbito)
  - Puede incorporar elemento estructurales de diseño de aplicaciones.
  - Pueden especializarse en plataformas: web, disp. móviles, etc.
  - Ej.: Rails, Django, Symfony, Flask, JDK, Android, Apple, etc.

La POO y la reutilización

Existen diversos elementos específicos de la POO centrados en la reutilización.

Hemos estudiado:

- Las clases y los objetos
- La herencia
- El polimorfismo.
- Genericidad (*templates* de función y de clase)
- Patrones de diseño

Resumen

- *“El desarrollo de software como industria basada en componentes”*
- La POO:
  - Objetos y clases para una programación más efectiva y natural.
  - Estructuras de reutilización para sacar provecho de los beneficios de la reutilización.

Fin