

Reúso de software (repaso)



Reúso

- ¿qué es lo que reusamos?
- ¿por qué el reúso es un tema importante en Ing. De Software?
- ¿qué dificultades encontramos?
- ¿qué alternativas de reúso tenemos?

¿qué reusamos?

- Aplicaciones completas que adaptamos e integramos
 - P.e., Drupal/Wordpress; GoogleApps, Minecraft, ...
- Componentes y Servicios (que invocamos)
 - P.e., Twitter API, Facebook API, Google Maps...
- Funciones y estructuras de datos
 - P.e., de Colecciones, de Fechas, de Archivos,
- Diseños o estrategias de diseño (que imitamos, implementamos)
 - P.e., Patrones de diseño, algoritmos de búsqueda, arquitecturas...
- Conceptos e ideas que funcionan
 - P.e., Compartir en las redes sociales; ayuda de contexto...

¿qué reusamos?

- Aplicaciones completas que adaptamos e integramos
 - P.e., Drupal/Wordpress; GoogleApps, Minecraft, ...
- **Componentes y Servicios (que invocamos)**
 - P.e., Twitter API, Facebook API, Google Maps...
- Funciones y estructuras de datos
 - P.e., de Colecciones, de Fechas, de Archivos,
- Diseños o estrategias de diseño (que imitamos, implementamos)
 - P.e., Patrones de diseño, algoritmos de búsqueda, arquitecturas...
- Conceptos e ideas que funcionan
 - P.e., Compartir en las redes sociales; ayuda de contexto...

¿qué reusamos?

- Aplicaciones completas que adaptamos e integramos
 - P.e., Drupal/Wordpress; GoogleApps, Minecraft, ...
- Componentes y Servicios (que invocamos)
 - P.e., Twitter API, Facebook API, Google Maps...
- **Funciones y estructuras de datos**
 - P.e., de Colecciones, de Fechas, de Archivos,
- Diseños o estrategias de diseño (que imitamos, implementamos)
 - P.e., Patrones de diseño, algoritmos de búsqueda, arquitecturas...
- Conceptos e ideas que funcionan
 - P.e., Compartir en las redes sociales; ayuda de contexto...

¿qué reusamos?

- Aplicaciones completas que adaptamos e integramos
 - P.e., Drupal/Wordpress; GoogleApps, Minecraft, ...
- Componentes y Servicios (que invocamos)
 - P.e., Twitter API, Facebook API, Google Maps...
- Funciones y estructuras de datos
 - P.e., de Colecciones, de Fechas, de Archivos,
- Diseños o estrategias de diseño (que imitamos, implementamos)
 - P.e., Patrones de diseño, algoritmos de búsqueda, arquitecturas...
- Conceptos e ideas que funcionan
 - P.e., Compartir en las redes sociales; ayuda de contexto...

¿qué reusamos?

- Aplicaciones completas que adaptamos e integramos
 - P.e., Drupal/Wordpress; GoogleApps, Minecraft, ...
- Componentes y Servicios (que invocamos)
 - P.e., Twitter API, Facebook API, Google Maps...
- Funciones y estructuras de datos
 - P.e., de Colecciones, de Fechas, de Archivos,
- Diseños o estrategias de diseño (que imitamos, implementamos)
 - P.e., Patrones de diseño, algoritmos de búsqueda, arquitecturas...
- **Conceptos e ideas que funcionan**
 - P.e., Compartir en las redes sociales; ayuda de contexto...

¿por qué?



- Para aumentar la productividad
 - Reducir los costos de desarrollo
 - Reducir el riesgo y la incertidumbre (mas vale conocido..)
 - Reducir los tiempos de entrega y puesta en el mercado



- Para aumentar la calidad
 - El reúso aprovecha mejoras y correcciones – el tiempo da madurez y confiabilidad
 - Se reaprovecha el conocimiento de los especialistas (que queda encapsulado en componentes reusables)
 - Ayuda a la implementación e imposición de estándares

¿por qué?



- Para aumentar la productividad
 - Reducir los costos de desarrollo
 - Reduce el riesgo y la incertidumbre (mas vale conocido..)
 - Menores tiempos de entrega y puesta en el mercado



- Para aumentar la calidad
 - El reúso aprovecha mejoras y correcciones – el tiempo da madurez y confiabilidad
 - Se reaprovecha el conocimiento de los especialistas (que queda encapsulado en componentes reusables)

¿qué dificultades tenemos?

- Problemas de mantenimiento si no tenemos control de los componente que reusamos
- Algunos programadores prefieren hacer todo ellos mismos (confianza en lo propio, búsqueda de desafíos)
- Hacer software reusable es mas difícil y costoso (paga a la larga y requiere procesos especiales)
- Encontrar, aprender a usar y adaptar componentes reusables requiere esfuerzo adicional

¿qué dificultades tenemos?

- Dificultades de mantenimiento si no tenemos control de los componente que reusamos
- Algunos programadores prefieren hacer todo ellos mismos (confianza en lo propio, búsqueda de desafíos)
- Hacer software reusable es mas difícil y costoso (paga a la larga y requiere procesos especiales)
- Encontrar, aprender a usar y adaptar componentes reusables requiere esfuerzo adicional

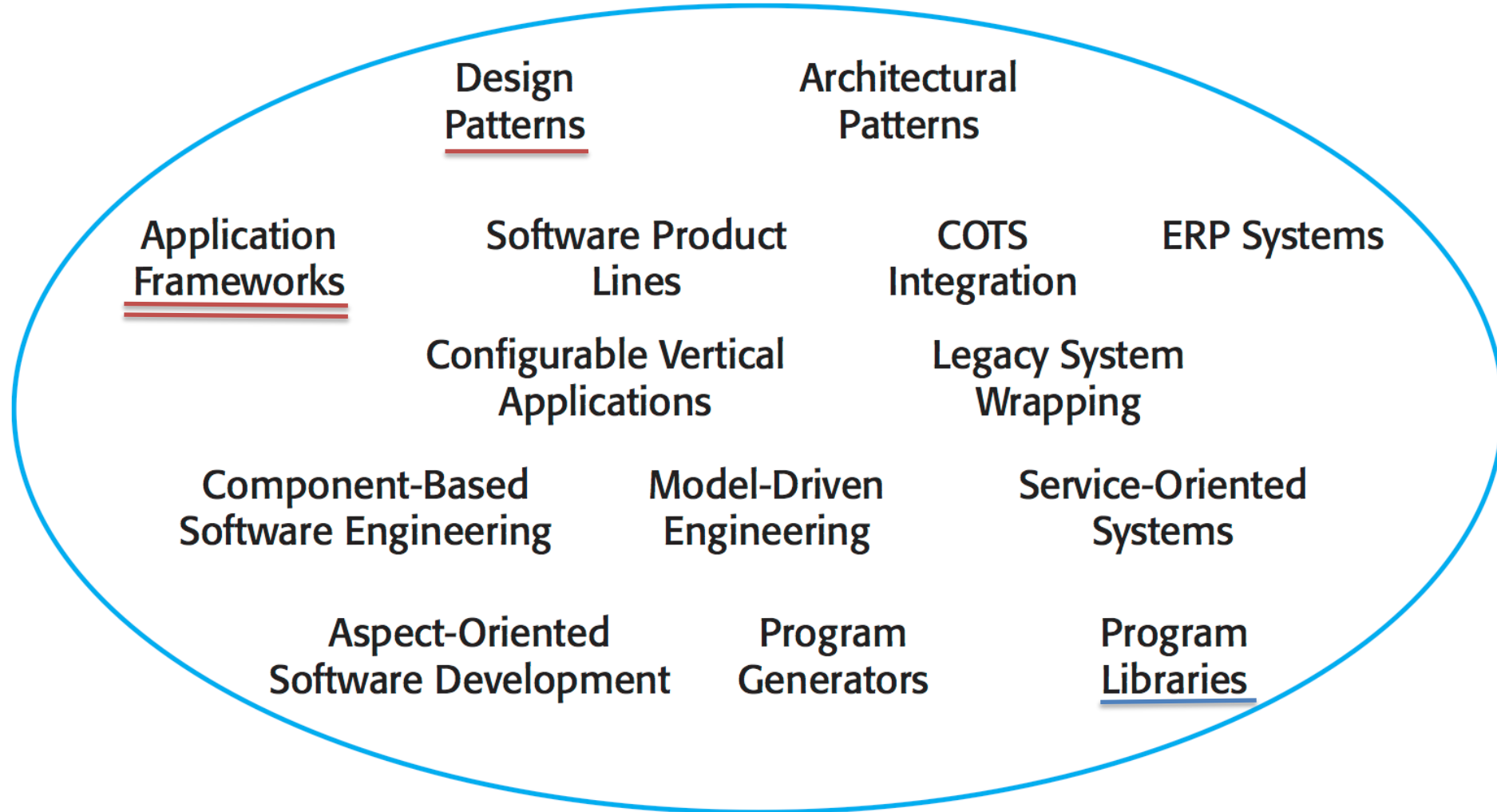
¿qué dificultades tenemos?

- Dificultades de mantenimiento si no tenemos control de los componente que reusamos
- Algunos programadores prefieren hacer todo ellos mismos (confianza en lo propio, búsqueda de desafíos)
- Hacer software reusable es mas difícil y costoso (paga a la larga y requiere procesos especiales)
- Encontrar, aprender a usar y adaptar componentes reusables requiere esfuerzo adicional

¿qué dificultades tenemos?

- Dificultades de mantenimiento si no tenemos control de los componente que reusamos
- Algunos programadores prefieren hacer todo ellos mismos (confianza en lo propio, búsqueda de desafíos)
- Hacer software reusable es mas difícil y costoso (paga a la larga y requiere procesos especiales)
- Encontrar, aprender a usar y adaptar componentes reusables requiere esfuerzo adicional

¿que alternativas tenemos?



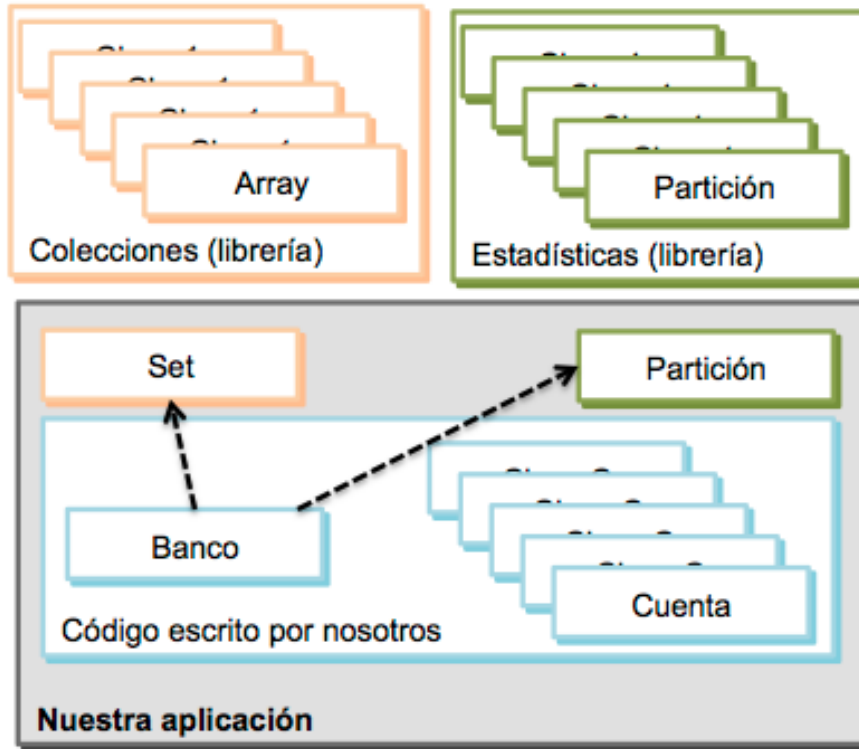
Librería de clases

- Resuelven problemas comunes a la mayoría de las aplicaciones
 - P.e., manejo de archivos, funciones aritméticas, colecciones, fechas,
- Cada clase en la librería resuelve un problema concreto, es independiente del contexto de uso, no espera nada de nuestro código, y generalmente independiente de otras clases en la librería
- Nuestro código controla/usa a los objetos de las librerías

Librería de clases

- Resuelven problemas comunes a la mayoría de las aplicaciones
 - P.e., manejo de archivos, funciones aritméticas, colecciones, fechas,
- Cada clase en la librería resuelve un problema concreto, es independiente del contexto de uso, no espera nada de nuestro código y generalmente independiente de otras clases en la librería
- Nuestro código controla/usa a los objetos de las librerías

Librería de clases



comunes a la mayoría de las
funciones aritméticas,
resuelve un problema
fuera del contexto de uso, y
dependiente de otras clases en la

- Nuestro código controla/usa a los objetos de las librerías

Algunas librerías de clases

<i>Java</i>	<i>Pharo (Cap. 3 de ColaborActiveBook)</i>
Codec (apache common) General encoding/decoding algorithms (for example phonetic, base64, URL).	Collections-* packages Multiple collection implementations
Compress (apache common) Defines an API for working with tar, zip and bzip2 files.	Compression package Multiple classes to work with Zip, Gzip and tar files
Math (apache common) Lightweight, self-contained mathematics and statistics components.	Graphic-Files package Classes for reading and writing image files (jpg, png, bmp)
fastutil extends the Java™ Collections Framework by providing type-specific maps, sets, lists and queues with a small memory footprint and fast access and insertion	Chronology (Kernel) package Classes to model dates, times, calendars, etc.

Frameworks Orientados a Objetos

- Un framework es una *aplicación “semi-completa”, “reusable”, que puede ser especializada para producir aplicaciones a medida...*
- *...un conjunto de clases concretas y abstractas, relacionadas para proveer una arquitectura reusable para una familia de aplicaciones relacionadas...*

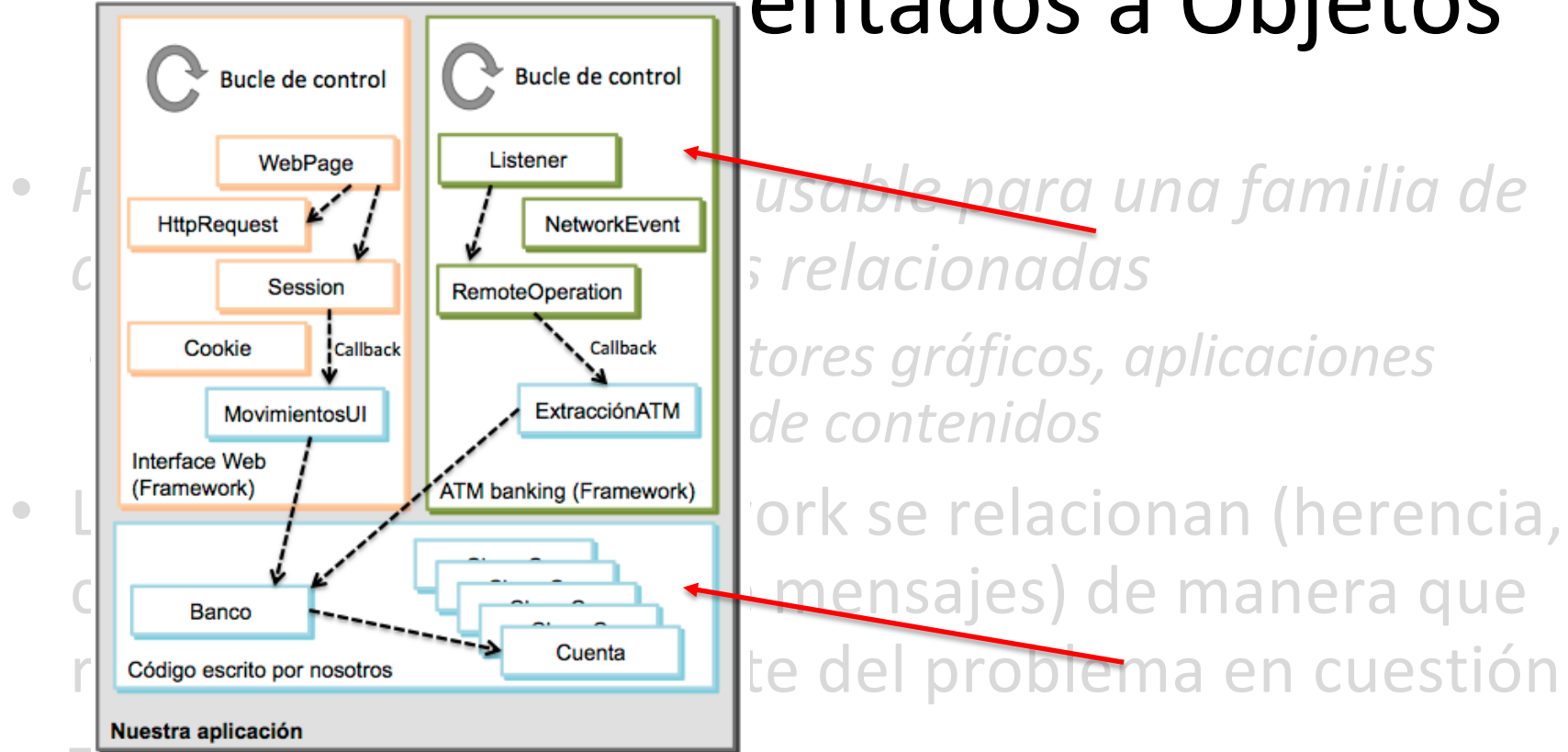
Frameworks Orientados a Objetos

- *Proveer una solución reusable para una familia de aplicaciones/problemas relacionadas*
 - *P.e., interfaces web, editores gráficos, aplicaciones colaborativas, gestores de contenidos*
- Las clases en el framework se relacionan (herencia, conocimiento, envío de mensajes) de manera que resuelven la mayor parte del problema en cuestión
 - conforman un todo
- El código del framework controla al nuestro

Frameworks Orientados a Objetos

- *Proveer una solución reusable para una familia de aplicaciones/problemas relacionadas*
 - *P.e., interfaces web, editores gráficos, aplicaciones colaborativas, gestores de contenidos*
- Las clases en el framework se relacionan (herencia, conocimiento, envío de mensajes) de manera que resuelven la mayor parte del problema en cuestión
 - conforman un todo
- El código del framework controla al nuestro

Frameworks Orientados a Objetos



- El código del framework controla al nuestro:

INVERSIÓN DE CONTROL

Frozenspot vs Hotspots

- Frozenspots de un framework
 - Todas las aplicaciones construídas con un mismo framework tienen aspectos en común que no podemos cambiar
- Hotspots de un framework
 - El framework ofrece puntos de extensión que nos permiten introducir variantes y así construir aplicaciones diferentes

Caja blanca vs Caja negra

- Los puntos de extensión pueden implementarse en base a herencia o en base a composición
- A los frameworks que utilizan herencia en sus puntos de extensión, les llamamos de Caja Blanca (Whitebox)
- A los que utilizan composición les llamamos de Caja Negra (blackbox)

Material adicional

Frameworks: Guía de lectura

TOC

Frameworks

Guía de lectura

Preguntas de autoevaluación

<< < ^ > >>

Frameworks

Un framework se puede ver como el resultado de refactorizar una aplicación, no solo para mejorar su calidad interna sino para abstraer lo que es común a toda una familia de aplicaciones similares. En ese proceso de refactoring se identifica lo que se mantendrá constante y será, a partir de ese momento, responsabilidad de los desarrolladores del framework y lo que podrá variar de aplicación a aplicación y será responsabilidad de los desarrolladores de aplicaciones que usen el framework.

Para separar la implementación de la parte constante (a lo que llamamos **frozenspot**) y de las partes que varían (a las que llamamos **hotspots**), se introducen en el framework puntos de extensión. Estos tienen la forma de plantillas que ofrecen ganchos para que, mediante herencia o composición, los programadores de aplicaciones definan el comportamiento variable. Algunos patrones de diseño (particularmente el template method y el strategy) son frecuentemente utilizados para introducir esos puntos de extensión.

La construcción y uso de frameworks es una estrategia para modularizar y reutilizar, que mejora la calidad del software y aumenta la productividad de quienes lo hacen. Un framework, visto como módulo de software, no solo encapsula operaciones y estructuras de datos reusables sino que las combina para resolver, con demostrada eficacia, los requerimientos comunes a una familia de aplicaciones. En pocas palabras, encapsula