Modelo para desarrollos software basados en componentes funcionales y no funcionales.

Miguel A. Pérez, Amparo Navasa, Juan M. Murillo
Departamento de Informática. Universidad de Extremadura.
{toledano, amparonm, juanmamu}@unex.es

1

Índice.

- Introducción.
- Objetivos
- Propuesta.
- Ejemplo.
- Líneas de trabajo.

2

Origen del problema.

- Introductcon.
- Objetivos.
- Documentacion.
- Ejemplo.
- Líneas futuras.

•El desarrollo de grandes sistemas software presenta los siguientes requisitos:

1. Minimizar el tiempo de desarrollo.
2. Incrementar la calidad de las aplicaciones software.
3. Minimizar los costes de mantenimiento.
4. Construir aplicaciones fácilmente adaptables al entorno cambiante.

•Por tanto, en la actualidad se trabaja sobre las ideas:

1. Reutilizar código.
2. Utilizar una filosofía plug-play.

3

Problema.

- Introducción.
- Objetivos.
- Propuesta.
- Ejemplo.
- Líneas futuras.

Reutilización de software y el modelo de componentes.

•Las dependencias entre el código de los componentes debido a los aspectos ortogonales.

Usar el paradigma de la separación de aspectos, para crear componentes funcionales más reusables.

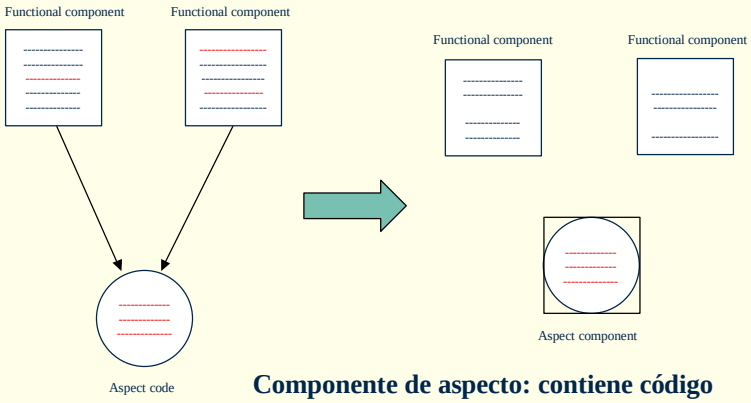
•El “código de aspecto” extraído del componente es difícil de reusar.

•A los componentes funcionales a los que se extrajo el código, se le debe aplicar el aspecto que les falta.

4

Componentes de aspecto.

- Introducción.
- Objetivos.
- Propuesta.
- Ejemplo.
- Líneas futuras.



Componente de aspecto: contiene código asociado a un aspecto (y sólo uno).

5

Ideas preliminares.

- Introducción.
- Objetivos.
- Propuesta.
- Ejemplo.
- Líneas futuras.

- Reutilizar el código de aspecto, documentando adecuadamente los componentes.
- Utilizar esos “componentes de aspecto” como los componentes funcionales utilizando la filosofía plug and play.
- Nos centraremos en los aspectos de sincronización, concurrencia, distribución y coordinación.
- Utilizar los componentes documentados para construir sistemas

6

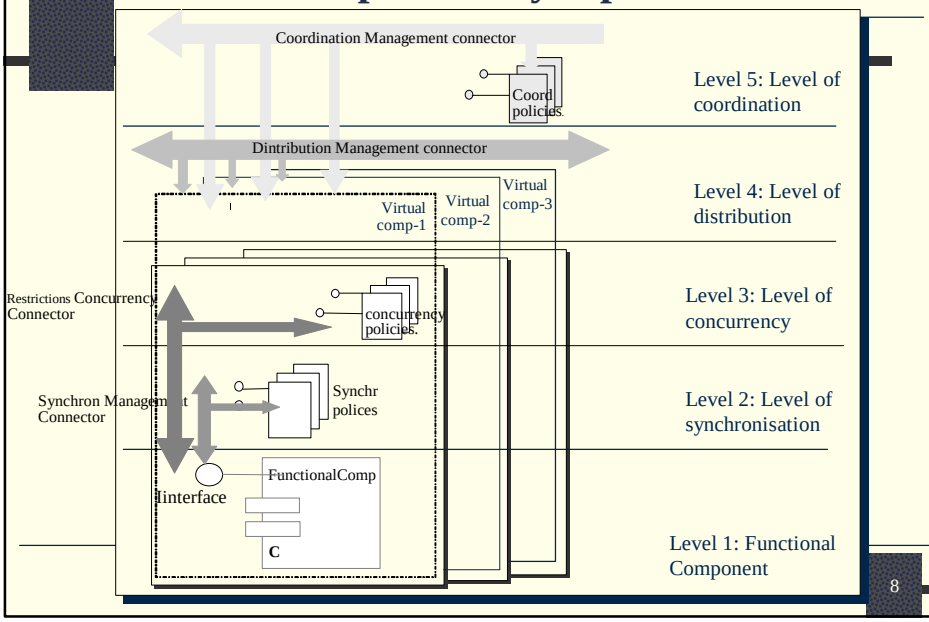
Objetivos

- Introducción.
- Objetivos.
- Propuesta.
- Ejemplo.
- Líneas futuras.

- Construir sistemas software, a partir de una arquitectura considerando los aspectos de sincronización, concurrencia, coordinación y distribución.
- Desarrollar repositorios para componentes de aspecto.
- Construir herramientas para búsqueda y selección de componentes de aspecto.
- Generar sistemas complejos a partir de la unión de componentes funcionales y no funcionales.

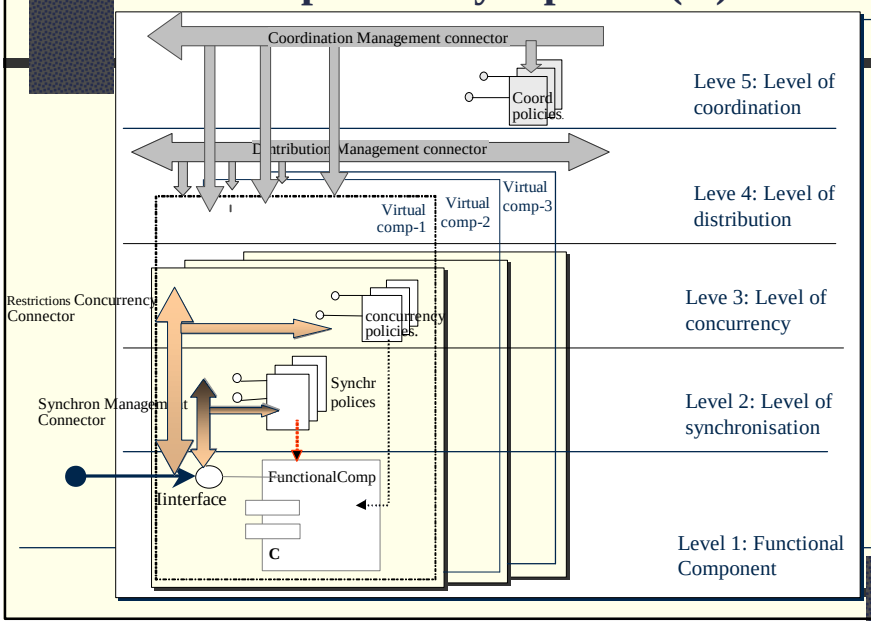
7

Arquitectura para integrar componentes y aspectos.



8

Arquitectura para integrar componentes y aspectos (II).



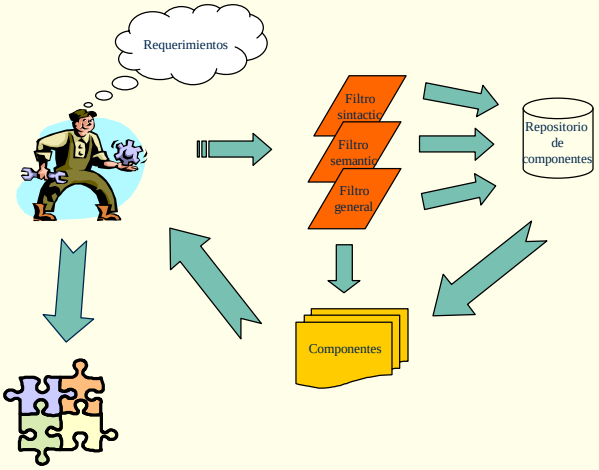
Documentación de componentes

- Introducción.
- Objetivos.
- Propuesta.
- Ejemplo.
- Líneas futuras.

- We need documentation for the aspect components that allows us to compare and make a better selection.
- The documentation is divided into three parts:
  1. Syntactic information (how do we use the component ?)
  2. General information (datos del componente)
  3. Semantic information(what does the component do?)
- The semantic documentation is the most complicated to explain and is different for each aspect.
- In this presentation, we will explain how to semantically describe coordination aspect components.

Selección de componentes

- Introducción.
- Objetivos.
- Propuesta.
- Ejemplo.
- Líneas futuras.



11

Diferencias componentes

- Introducción.
- Objetivos.
- Propuesta.
- Ejemplo.
- Líneas futuras.

- Diferentes modelos de separación de aspectos, incompatibles entre ellos.
- Cada aspecto presenta unas necesidades de documentación diferentes.
- Interface componentes no-funcionales muestra los requerimientos que necesita.
- Descripción semántica del componente imprescindible para recuperar componentes de aspecto.

12

Información sintáctica

- Introducción.
- Objetivos.
- Propuesta.
- Ejemplo.
- Líneas futuras.

- Descripción de eventos permitidos.
  - Tipo de evento.
  - Notificación.
  - Mensajes.
  - Estudio de parámetros.
- Información de interoperabilidad.
  - Descripción de interface.
  - Información sobre correspondencia entre parámetros.

13

Información general

- Introducción.
- Objetivos.
- Propuesta.
- Ejemplo.
- Líneas futuras.

- Nombre del componente.
- Datos de implementación.
- Datos sobre versionado.
- Información sobre dominios de utilización habituales.
- Datos fabricante.
- Información sobre los demás componentes.
- Restricciones.

14

Información semántica

- Introducción.
- Objetivos.
- Propuesta.
- Ejemplo.
- Líneas futuras.

- Cada aspecto presenta diferencias.
- Realizar la descripción a alto nivel.
- Descripción lo más precisa posible.
- Posibilidad de comparar especificaciones.
- Información intuitiva.
- Herramienta conocida por los usuarios finales.
- Información sobre cambios de estados en el componente.

15

Recuperación

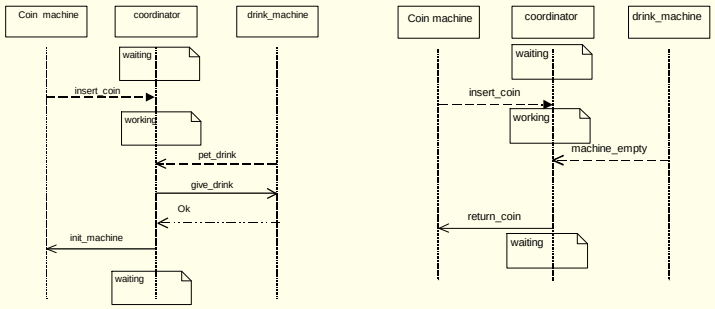
- Introducción.
- Objetivos.
- Propuesta.
- Ejemplo.
- Líneas futuras.

- La recuperación de un componente que cumpla exactamente los requerimientos de usuario es muy difícil.
- Proponemos el “trading” como forma de recuperación, utilizando la información disponible.
- La información semántica como forma de seleccionar, y también como fuente de información al usuario.
- Información adicional, con los diagramas de estado, no como herramienta de búsqueda sino como ayuda a la selección.

16

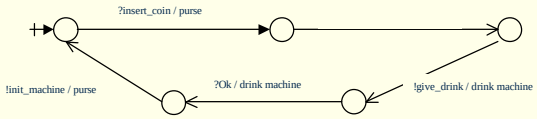
Ejemplo: máquina de bebidas

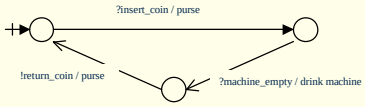
- Introducción.
- Objetivos.
- Propuesta.
- Ejemplo.
- Líneas futuras.



Autómatas

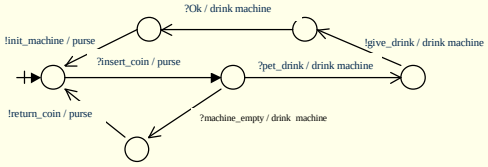
- Introducción.
- Objetivos.
- Propuesta.
- Ejemplo.
- Líneas futuras.

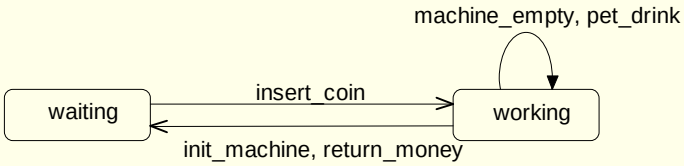




Resultado

- Introducción.
- Objetivos.
- Propuesta.
- Ejemplo.
- Líneas futuras.





Líneas de investigación

- Introducción.
- Objetivos.
- Propuesta.
- Ejemplo.
- Líneas futuras.

- Extender el modelo a otros aspectos.
- Construir repositorios de componentes de aspecto.
- Incluir en las metodologías de desarrollo orientado a componentes funcionales, referencias a los componentes de aspecto.
- Formalizar la arquitectura propuesta.
- Diseñar herramientas para búsqueda y selección de componentes de aspecto, desde repositorios.