Arquitectura de Software

Vista de módulos

Índice

- ¿Qué es la vista de módulos?
- ¿Qué es un módulo?
- Relaciones entre módulos
- Propiedades de los módulos
- ¿Para qué sirve una vista de módulos?
- ¿Cómo documentar la vista de módulos?
- Relación con otras vistas
- Discusión y resumen

¿Qué es la vista de módulos?

- La que describe las estructuras que implementan el sistema (módulos) y sus relaciones
- Como mínimo determina cómo se estructura el código fuente

¿Qué es un módulo?

- Una unidad de implementación (parte del *código fuente)* que proporciona un conjunto coherente de *responsabilidades*
- Ejemplos:
  - Unidades de código fuente: paquete de java
  - Unidades de lenguajes de programación: un programa C, una clase java, clase C++
- Las responsabilidades definen la funcionalidad del módulo (qué hace) y qué conocimiento mantiene

Principios de Parnas

- Ocultación de la información
  - El diseñador de un módulo debe proporcionar al usuario del mismo toda la información necesaria para que haga el mejor uso del módulo, pero no más información
  - Al desarrollador de un módulo se le debe dar toda la información necesaria para implementar las responsabilidades del módulo, pero no más información

Principios de diseño de módulos

- Acoplamiento
  - El acoplamiento nos dice el grado de interdependencia entre los módulos del sistema
  - Buscamos bajo acoplamiento
- Cohesión
  - Nos dice en qué medida están funcionalmente relacionados los elementos de un módulo. En qué medida el módulo realiza una única tarea
  - Buscamos alta cohesión

Relaciones entre módulos

- Tres tipos: Es parte de, Dependencia, Is-a
- Es parte de:
  - Lo que conocemos en orientación a objetos como agregación, composición
  - El módulo agregado (el todo) está formado por sub-módulos (las componentes) que a su vez son módulos
  - Dará lugar al "estilo de descomposición"

Relaciones entre módulos ...

- Dependencia. Hay tres tipos:
  - *Uso*. El módulo A usa al módulo B
    - Dará lugar al "estilo de uso"
  - *Permite su uso*
    - Dará lugar al "estilo de capas"
  - *Transversal*
    - Dará lugar al "estilo de aspectos"

Relaciones entre módulos ...

- Is a:
  - Como la especialización/generalización en orientación a objetos
  - El módulo hijo puede ser utilizado allí donde puede ser utilizado el módulo padre
  - Dará lugar al "estilo de generalización"

Relaciones para crear el modelo de datos.

  - *Asociación, etc.* Utilizaremos toda la riqueza vista en BDR e IS

Propiedades de los módulos

- Es lo que debemos documentar
- Nos sirven para la *implementación* y para *analizar* el sistema
- *Nombre*
  - Debe ser significativo
  - Refleja su posición en la jerarquía de descomposición
- *Responsabilidades*
  - ¿Nivel de detalle? El que creamos adecuado para que el lector entienda perfectamente qué hace el módulo

Propiedades ...

- *Visibilidad de las interfaces*
  - En un sub-módulo distinguimos dos tipos de interfaces
    - Internas
      - Usadas sólo por los "hermanos", no tienen relación con el padre
    - Tienen relación con el padre
      - Encapsulación. El padre proporciona las interfaces y mapea las peticiones a los hijos
      - Las interfaces del agregado son un subconjunto de las interfaces de los sub-módulos que lo componen

Propiedades ...

- *Información de implementación*
  - No pertenece a la arquitectura, pero es un buen sitio donde guardarla
  - Tipos de información
    - Lista de los ficheros que forman el módulo
    - Información para probar el módulo. Ejemplo: casos de prueba
    - Información de gestión. Ejemplos: Presupuesto, fecha de finalización
    - Restricciones de implementación
- *Cualquier otra información útil*

¿Para qué sirve?

- *Plan de acción* para construir el software
- Realizar análisis de impacto
- Realizar el seguimiento de los requisitos
- Explicar la funcionalidad del sistema y la estructura del código fuente

¿Para qué sirve? ...

- Facilitar la asignación de trabajo
- Planificar el desarrollo de manera incremental (Figura 2.6)
- Mostrar la estructura de la información persistente q modelo de datos
- ¿Para qué no nos sirve?

¿Cómo documentar la vista?

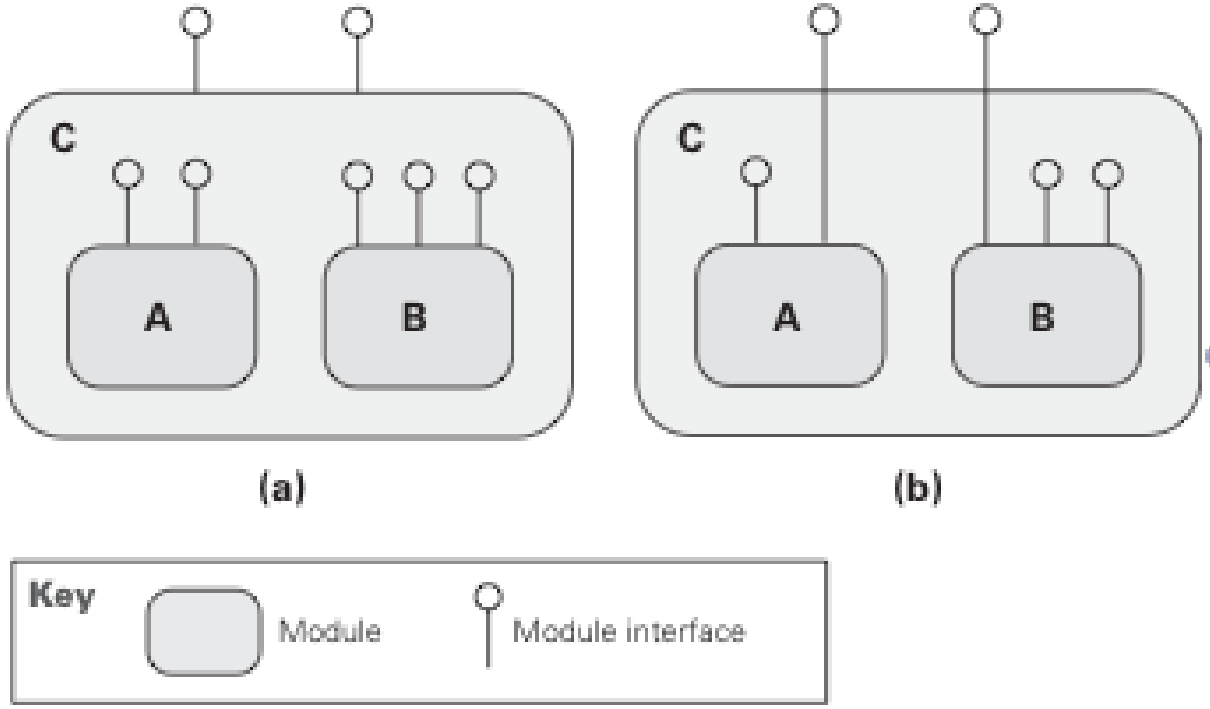
- Notaciones informales
- Matriz de dependencias
- UML
- Diagrama entidad-relación

Notaciones informales

- Gráfica
  - Cajas y líneas
    - Ejemplo: Figura 1.1
- Textual
  - Listar los módulos con la descripción de sus responsabilidades
  - Relaciones
    - Es parte de: sangrar, numerar,...
    - Otras relaciones: *keywords*

Notación informal gráfica

Figure 1.1
(a) Module C provides its own interface, hiding the interfaces of modules A and B. (b) Module C exposes a subset of the interfaces of modules A and B as its interface.



Matriz de dependencias

- Matriz de dependencias
  - http://www.ndepend.com/doc_matrix.aspx
  - http://www.lattix.com/technology

UML

- Lo hemos visto para modelado orientado a objetos. Es un lenguaje de modelado de propósito general
- Ejemplos:
  - Figuras 1.2, 1.3 y 1.4

UML ...

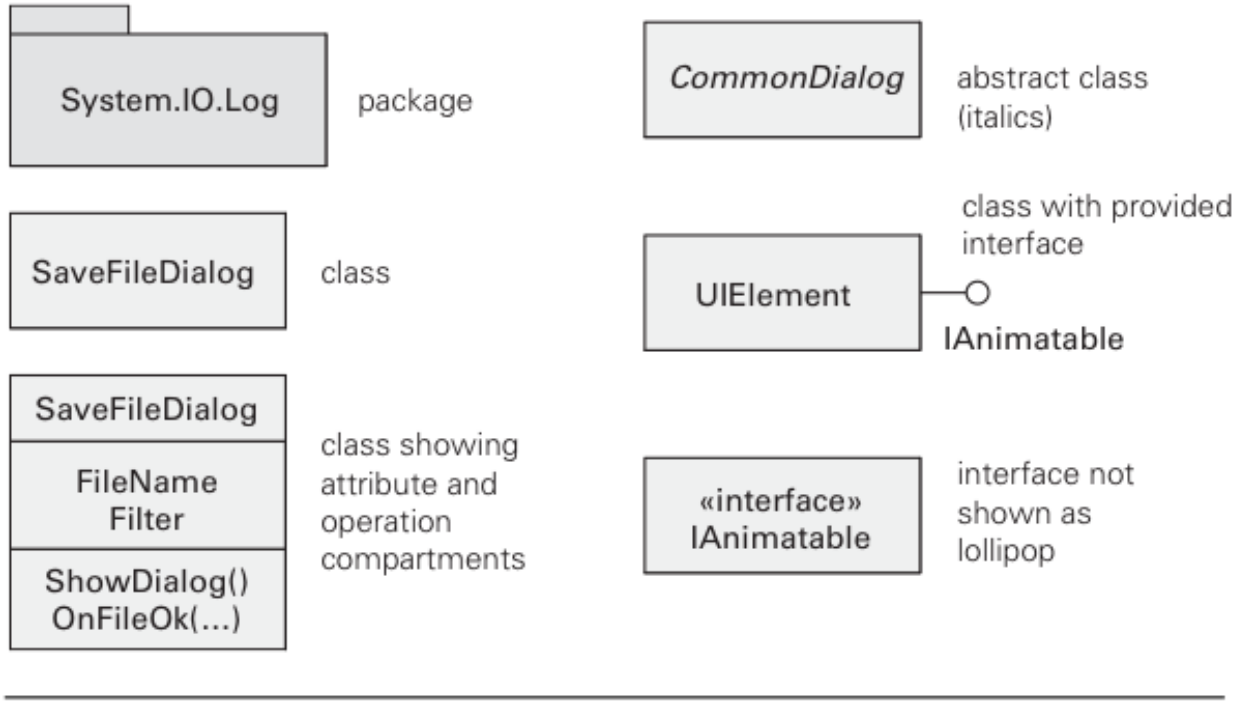


Figure 1.2
Examples of module notation in UML. A module may be represented as a class or a package. More specific types of modules can be indicated with stereotypes (as in Figure 1.4).

UML ...

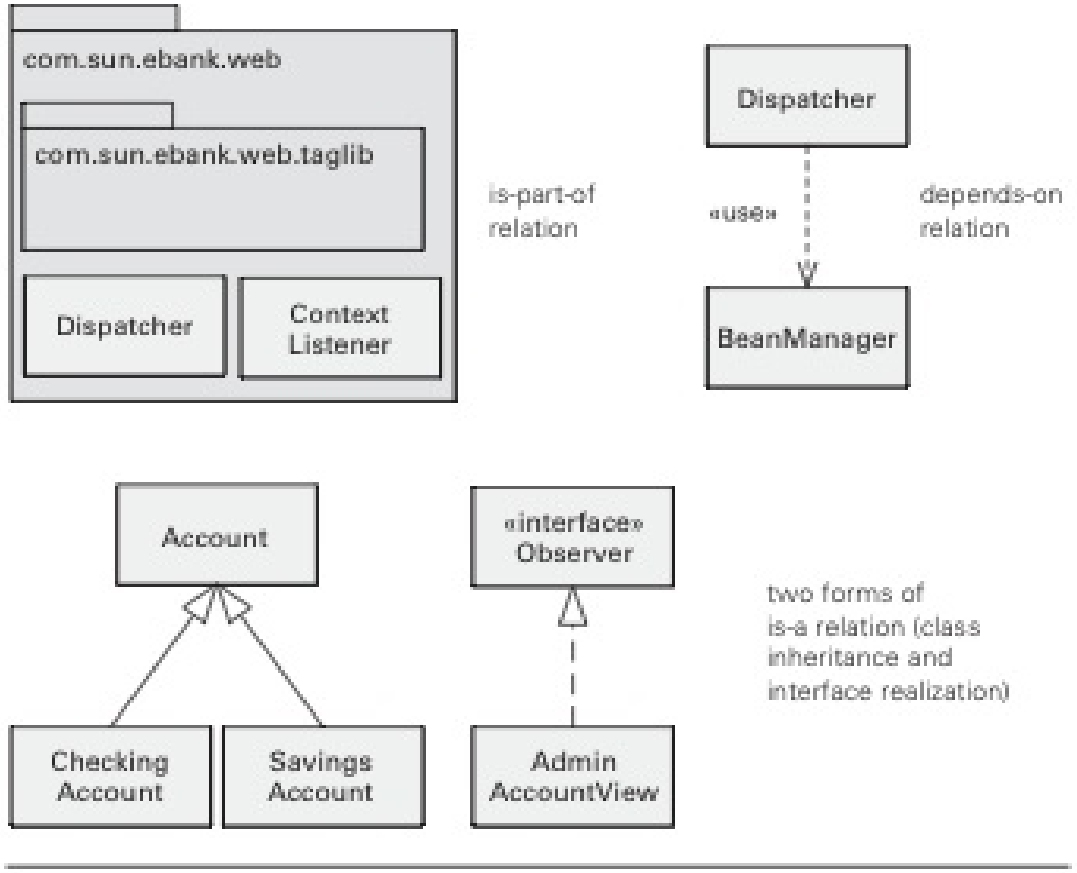
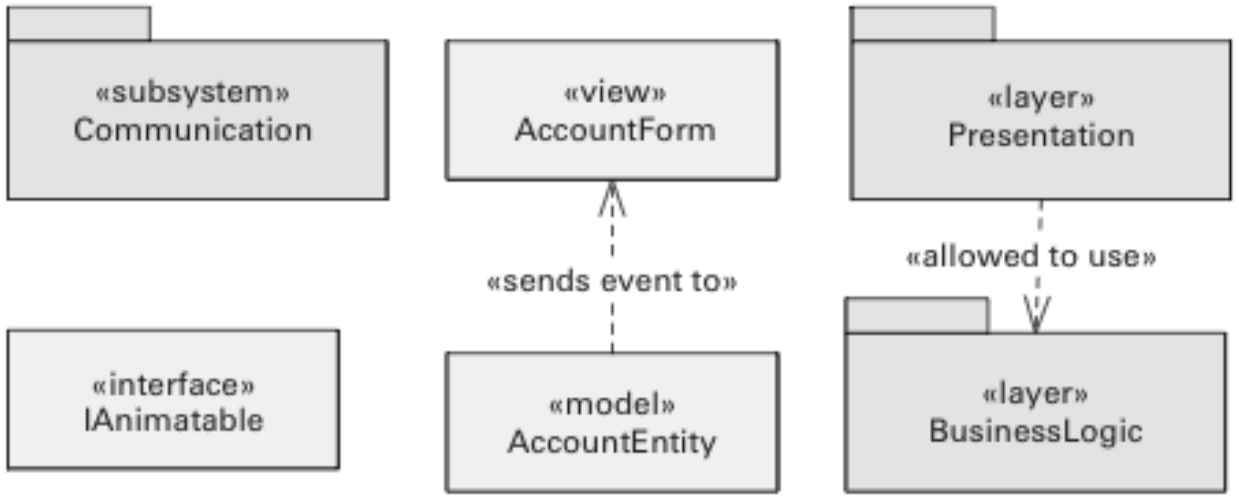


Figure 1.3
Examples of module relations in UML

UML ...

Figure 1.4
Examples of UML elements and relations with stereotypes



Relación con otras vistas

- La vista de módulos tiene correspondencia con la de C&C
  - Un módulo puede convertirse en un componente en ejecución o en varios
  - Un componente puede venir de varios módulos
- No mostrar relaciones que se producen en tiempo de ejecución

Puntos a discutir

1. ¿Cuál debería ser el nivel de detalle en la vista de módulos?
2. ¿Cual es la diferencia entre las responsabilidades de un módulo y los requisitos que debe satisfacer?
3. ¿Todos los módulos agregados son módulos?
4. ¿Mostrarías las librerías de las que dependen tus módulos como módulos en tu vista de módulos?
5. ¿Qué tipos de dependencias pueden expresarse en la vista de módulos?
6. ¿Qué podemos decir sobre el flujo de control en la vista de módulos? ¿sobre el flujo de datos?

Resumen

1. La vista de módulos *es arquitectural* q Los módulos nos dicen cómo se descompone el sistema en unidades de implementación
2. Los módulos se relacionan entre sí
3. *Plan de acción* para el código fuente y el modelo de datos
4. Tendremos al menos una vista de módulos
5. La propiedad de *responsabilidad* es la más importante
6. Documentar la interfaz del módulo
7. Esta vista tiene correspondencia con la de C&C