

Ingegneria del Software I

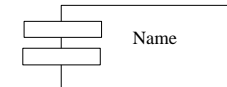
Unified Modelling Language

Modellare l'implementazione
Component Diagram
Deployment Diagram

1

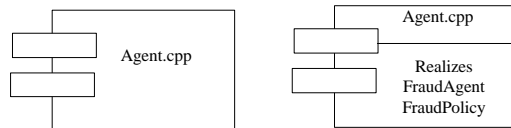
Componente in UML

- ◆ Un componente rappresenta un pezzo "fisico" dell'implementazione di un sistema
 - programma eseguibile (es. .exe): << executable>>
 - libreria statica (es. .h) o dinamica (es. .dll): << library>>
 - file di codice sorgente o dati (es. .cpp): << file>>
 - documento (es. .htm): << document>>
 - tabella di database: << table>>
 - ...
- ◆ Un componente ha un nome e una locazione



Componente in UML

- Un componente ha un nome e una locazione
 - è mostrato, tipicamente, con il solo nome; per le classi è possibile 'adornarlo' con compartimenti riportanti altri dettagli
 - è possibile indicare le relazioni tra component e class e/o interface che essi realizzano
- I componenti (come a livello logico le classi) possono essere raggruppati in package
 - Se i componenti sono file, i package sono cartelle/ directory



Component Diagram

- ◆ Rappresentano l'implementazione del sistema
 - Un componente rappresenta un pezzo "fisico" dell'implementazione di un sistema
- ◆ Definiscono le relazioni fra i componenti software che realizzano l'applicazione
 - sorgenti, binari, eseguibili, ...
- ◆ Parte della specifica architetturale ...
- ◆ Evidenziano l'organizzazione e le dipendenze esistenti tra componenti
 - I diversi componenti offrono e usano interfacce specifiche
 - Primo passo verso *Component Programming*

4

Modellare l'Implementazione

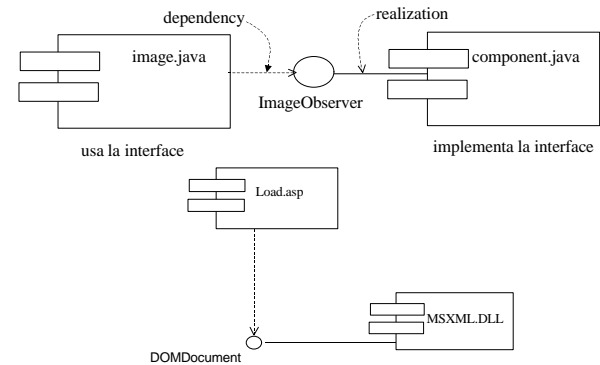
Ingegneria del Software I

Componenti ed interfacce

- ◆ Un componente può utilizzare le interfacce di altri componenti e realizzare un insieme di interfacce
 - “a component conforms to and provides the realization of a set of interfaces” (Booch '99)
- ◆ Le interfacce rappresentano una collezione di operazioni usate per specificare un servizio
 - Java e C++ più recente permettono di definire “interface”
- ◆ I sistemi di middleware più comuni COM+ (ora .NET), CORBA, Enterprise JavaBeans, sono basati su componenti e usano interfacce per legare le componenti tra loro

5

Componenti e Interfacce: Esempi



6

Componenti e classi

- ◆ Livelli di astrazione distinti
 - Le classi rappresentano astrazioni logiche mentre le componenti rappresentano cose fisiche (hanno una locazione e sono viste dal software di base)
 - I servizi di una classe sono direttamente accessibili da programma tramite operazioni pubbliche, mentre i servizi di un componente sono accessibili tramite le interfacce che implementa
- ◆ Le classi sono realizzate da componenti
 - Tipicamente, un component mappa una o più class, interface o collaboration
 - Spesso un componente coincide con un package che racchiude un insieme di classi ...
 - Importante mantenere la tracciabilità tra classi e componenti⁷

Sostituibilità di un componente

- ◆ I componenti collaborano tra di loro tramite interfacce
 - Interfaccia esportata: l'interfaccia che il componente fornisce come servizio alle altre componenti
 - » Ci può essere più di un'interfaccia esportata
 - Interfaccia importata: l'interfaccia che il componente usa
- ◆ Un componente è sostituibile, ovvero è possibile sostituire un componente con un altro che è conforme alle stesse interfacce e che preserva l'interfaccia precedentemente esportata
 - È possibile estendere l'interfaccia esportata o aggiungerne nuove
 - E' possibile aggiungere nuove importazioni

8

Modellare l'Implementazione

Ingegneria del Software I

Tipi di Componente

Tre tipi di componenti:

- deployment components: i componenti necessari e sufficienti per formare un sistema eseguibile (DLL, programmi eseguibili, ...)
- work product components: componenti che non partecipano direttamente nel sistema eseguibile ma che sono frutto del lavoro fatto per creare il sistema eseguibile
 - sono, essenzialmente, il 'residuo' del processo di sviluppo (source code file, data file usati per la creazione di deployment components,)
- execution component: componenti creati come conseguenza del sistema eseguibile (es. COM+ object, JCL, ...)

9

Componenti e UML stereotypes

UML definisce 5 tipi di stereotypes standard per i component

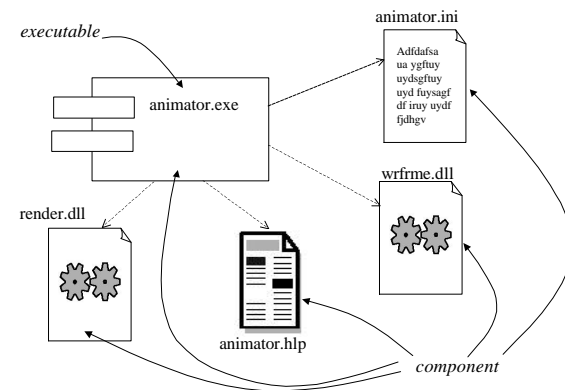
- <<executable>>: un component che può essere eseguito in un nodo
- <<library>>: una libreria statica o dinamica
- <<table>>: rappresenta una tabella di un database (relazionale)
- <<file>>: un component contenente codice sorgente o dati
- <<document>>: un component rappresentante un documento

non sono definite icone specifiche per tali stereotipi alcune tra le più usate sono:



10

Esempio



11

Deployment Diagram

- ◆ Anche detto diagramma di allocazione o di dislocazione
- ◆ Elementi: nodo, connessione tra nodi
- ◆ Permette di rappresentare, a diversi livelli di dettaglio, l'architettura fisica del sistema
- ◆ Permette anche di evidenziare la configurazione dei nodi elaborativi in ambiente di esecuzione (run-time), e dei componenti, processi ed oggetti ubicati in questi nodi

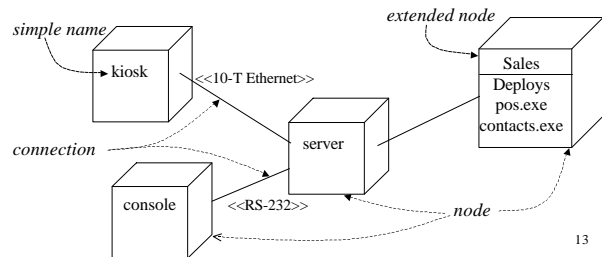
12

Modellare l'Implementazione

Ingegneria del Software I

Deployment diagrams

- Mostra la configurazione di run time dei nodi elaborativi e dei componenti ubicati in essi
- permette di rappresentare, a diversi livelli di dettaglio, l'architettura fisica del sistema

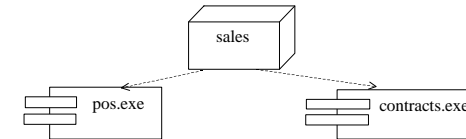


13

Deployment Diagram: Nodi

Ciascun nodo è identificato da un nome

- ◆ Un nodo può riportare ulteriori dettagli in compartimenti aggiuntivi o usando tagged value
- ◆ Un nodo può essere in connection con altri nodi, ed avere relationship con componenti e altri nodi
 - componenti sono things che partecipano nell'esecuzione di un sistema; nodi sono things che eseguono componenti
 - componenti rappresentano il packaging fisico di altri elementi logici; nodi rappresentano l'allocazione fisica di componenti



14

Modellare l'Implementazione