Introduzione a UML
Linguaggio di Modellazione Unificato

Corso di Ingegneria del Software

Anno Accademico 2012/13

Che cosa è UML?

- **UML (Unified Modeling Language) è un linguaggio grafico per:**
  - **specificare**
  - **visualizzare**
  - **realizzare**
  - **documentare i manufatti di un sistema software**
- **Adottato come standard da OMG (Object Management Group) dal 1997 (UML 1.1)**
- **La versione attuale è UML 2.0 (2004)**
- **Creatori: Grady Booch, Ivar Jacobson, Jim Rumbaugh**

Obiettivi UML

- **Definire un linguaggio di modellazione semplice ma ricco di concetti**
- **Unificare i linguaggi di modellazione OO: Booch, OMT, Objectory e gli altri presenti sul mercato**
- **Adottare le "best practices" dell'industria**
- **Considerare le tematiche attuali dello sviluppo del software:**
  - **Scalabilità, distribuzione, concorrenza, ecc**
- **Abilitare l'interscambio dei modelli OO e aiutare la definizione di *repository* per i modelli**

UML Modeling Language

- **Un linguaggio fornisce un vocabolario e le regole per combinare le parole del vocabolario**
- **Linguaggio:**
  - **Sintassi: regole con cui gli elementi del linguaggio (ad es. le parole) sono raggruppati in espressioni (ad es. le frasi)**
  - **Semantica: regole con cui viene assegnato un significato alle espressioni sintattiche**

UML: Modeling Language

- **Un *modeling language* è un linguaggio in cui il vocabolario e le regole si focalizzano sulla rappresentazione concettuale e fisica del sistema**
- **UML:**
  - **Sintassi: simboli usati nel modello**
  - **Semantica: set di regole che definiscono come usare la notazione**

UML: Modeling Language

- **Un *modeling language* come UML definisce come creare e leggere modelli ben formati ma non quali modelli creare**
- **Modello ben formato: modello (o parte di un modello) che rispetta tutte le regole semantiche e sintattiche che gli si applicano**
- **Tutti gli elementi e i diagrammi UML sono basati sul paradigma object-oriented**

Modelli ben formati

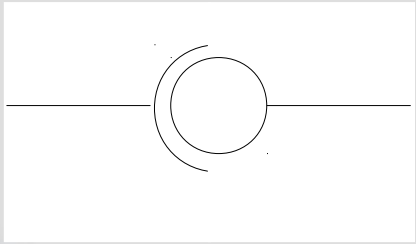
- **UML permette di specificare regole (tra l'altro) per:**
  - **Dare i nomi**
  - **Controllare la visibiltà dei nomi**
  - **Generare automaticamente porzioni di codice**
- **Durante lo sviluppo iterativo e incrementale, è possibile che i modelli siano incompleti e inconsistenti**

Regole comuni a tutti i diagrammi

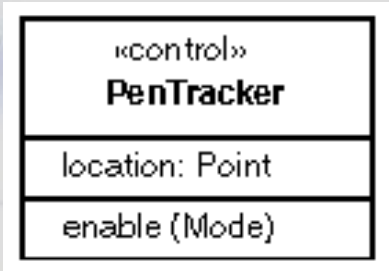
- **I diagrammi UML sono grafi contenenti nodi e connessioni**
- **Vi sono tre relazioni topologiche importanti:**
  - **La connessione (forme bidimensionali collegate con linee)**
  - **Il contenimento**
  - **La prossimità (di due simboli)**
- **Di solito, le dimensioni non hanno importanza**

I quattro costrutti grafici di UML

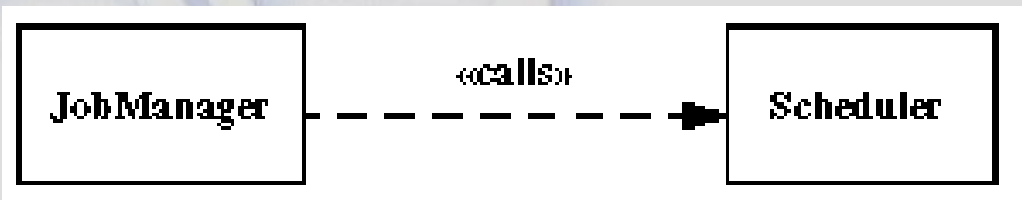
- **Icone:**



- **Simboli bidimensionali:**



- **Collegamenti:**



- **Stringhe:**

integrate (f: Function,from: Real, to: Real)

Costrutti grafici di UML

- **Le icone hanno dimensioni e forme fisse, e non contengono nulla**
- **I simboli bidimensionali:**
  - **hanno dimensioni variabili**
  - **possono essere espansi e contenere altre entità**
  - **possono essere suddivisi in *compartimenti***
- **I collegamenti sono linee curve o spezzate che terminano con connessioni a due simboli**
- **Le stringhe hanno molti usi e possono apparire entro liste**

Blocchi base di UML

- **Elementi: astrazioni che rappresentano concetti di prima classe del modello.**
  - **Classi, interfacce, componenti, casi d'uso, ecc.**
- **Relazioni: relazioni tra le cose**
  - **Associazioni, generalizzazioni, dipendenze, ecc.**
- **Diagrammi: raggruppano collezioni interessanti di elementi.**
  - **Diagrammi delle classi, dei casi d'uso, d'interazione, ecc.**

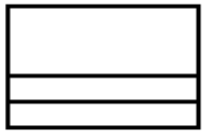
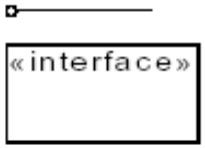
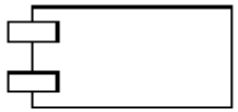
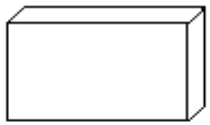
Modelli UML

- **Modello Statico:**
  - **Rappresenta gli elementi strutturali di un sistema software (la struttura dati e l'architettura) e le loro relazioni statiche (cioè invarianti al trascorrere del tempo).**
- **Modello Dinamico:**
  - **Rappresenta il comportamento di un sistema software al trascorrere del tempo.**

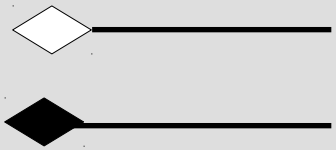
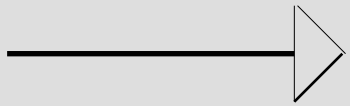
Modellazione Strutturale

- **Rappresenta una vista di un sistema software basata sulla *struttura* degli oggetti (classi di appartenenza, relazioni, attributi, operazioni)**
- **E' rappresentata dal modello statico**
- **Il *diagramma* delle classi è il principale diagramma UML per rappresentare un modello statico**

Modellazione Strutturale: Entità Fondamentali

Costrutto	Descrizione	Sintassi
Classe	La descrizione di un insieme di oggetti con gli stessi attributi, operazioni, relazioni e comportamento	
Interfaccia	Un insieme di operazioni che caratterizzano il comportamento di un elemento	
Componente	Una parte modulare, sostituibile e significativa del sistema che incapsula un'implementazione e espone un insieme di interfacce	
Nodo	Un oggetto fisico che rappresenta una risorsa computazionale quando il sistema è in funzione	

Modellazione Strutturale: Relazioni Fondamentali

Costrutto	Descrizione	Sintassi
Associazione	Una relazione tra due o più tipi che comporta un collegamento tra le loro istanze	
Aggregazione	Una forma di aggregazione che specifica una relazione tutto-parti tra le istanze	
Generalizzazione	Una relazione di classificazione tra un tipo più specifico ed uno più generale	
Dipendenza	Una relazione tra due elementi, per cui un cambiamento dell'elemento indipendente influenza l'elemento dipendente	

Modellazione Strutturale: Relazioni Fondamentali

Costrutto	Descrizione	Sintassi
Realizzazione	Una relazione tra una specifica e la sua implementazione	
Annidamento		

I diagrammi di modellazione strutturale

- **Principali tipi di diagrammi:**
  - **Composite structure diagram**
  - **Diagrammi strutturali statici:**
    - **Diagrammi di classe**
    - **Diagrammi degli oggetti**
  - **Diagrammi d'implementazione:**
    - **Diagrammi dei componenti**
    - **Diagrammi di dispiegamento (deployment)**
    - **Diagramma dei package**

Diagrammi di struttura statica

- **Mostrano le entità del sistema connesse secondo le relazioni statiche che le caratterizzano**
- **Diagrammi di classe:**
  - **Mostrano le classi e le loro relazioni**
  - **Sono a un livello di astrazione alto**
  - **Sono utilizzabili per generare il codice con le strutture dati e le dichiarazioni delle funzioni**
- **Diagrammi degli oggetti**
  - **Mostrano istanze delle classi durante uno scenario di funzionamento del sistema**

Diagrammi comportamentali

- **Rappresentano caratteristiche comportamentali di un sistema o di un processo di business.**
- **Mostrano informazione dipendente dal tempo**
- **Principali tipi di diagrammi:**
  - **Diagramma dei casi d'uso**
  - **Diagramma di attività**
  - **Diagramma delle macchine a stati**
  - **Diagrammi di interazione: enfatizzano le interazioni tra oggetti**
    - **Diagramma di sequenza**
    - **Diagramma di interazione**

Diagrammi d'implementazione

- Mostrano aspetti dell'implementazione di un modello:
  - Struttura del codice sorgente
  - Struttura del sistema in esecuzione.

Due diagrammi:

- Diagramma dei componenti (*component diagram*)
- Diagramma di dispiegamento (*deployment diagram*)

Diagramma dei componenti

- Mostra l'organizzazione delle componenti software e le loro dipendenze
- Le componenti possono essere:
  - specificate tramite classi (ad es. le classi d'implementazione)
  - implementate da artefatti (ad es. file rilocabili, eseguibili o di comandi)
- Non è particolarmente usato o significativo

Componenti

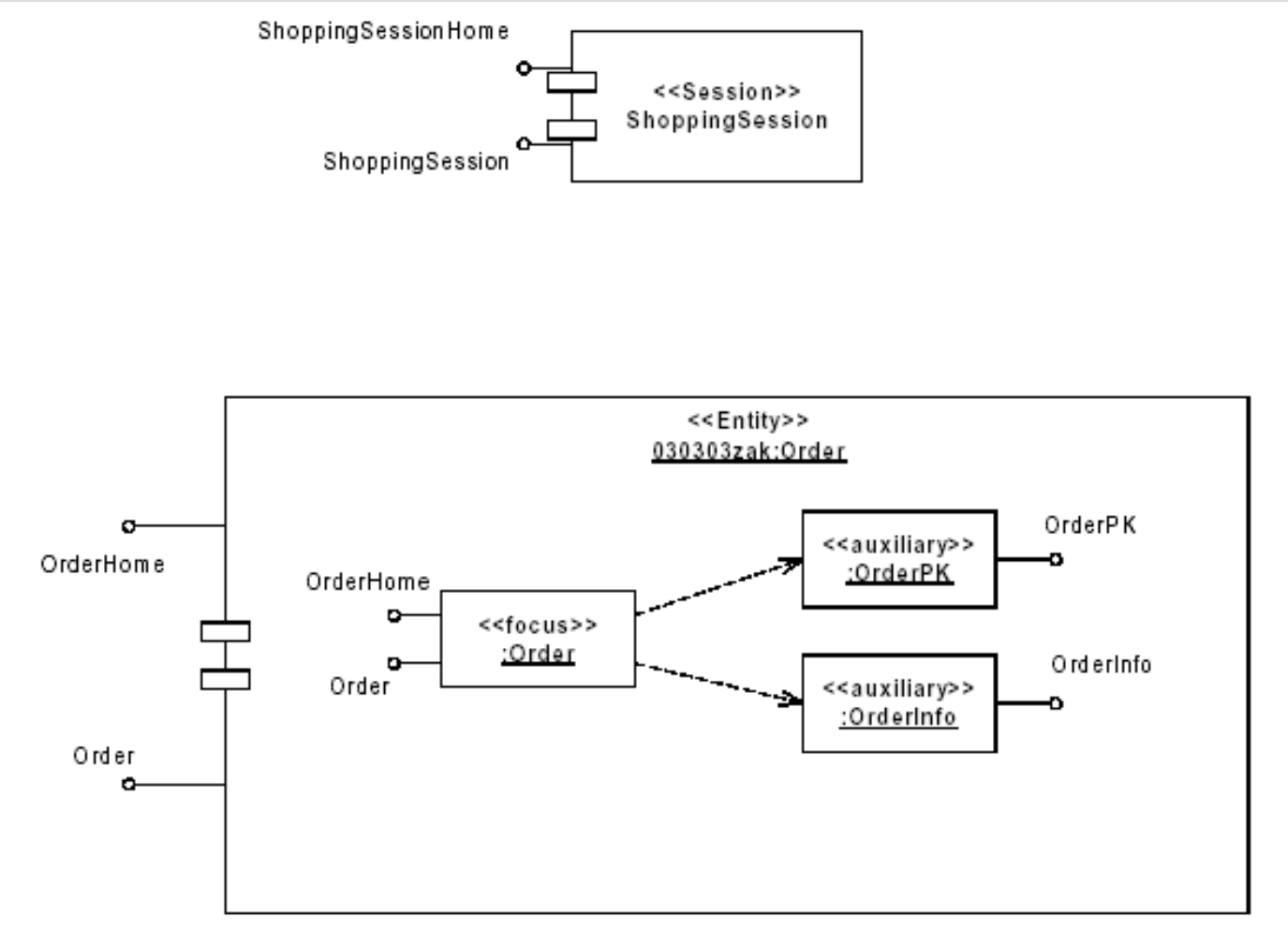
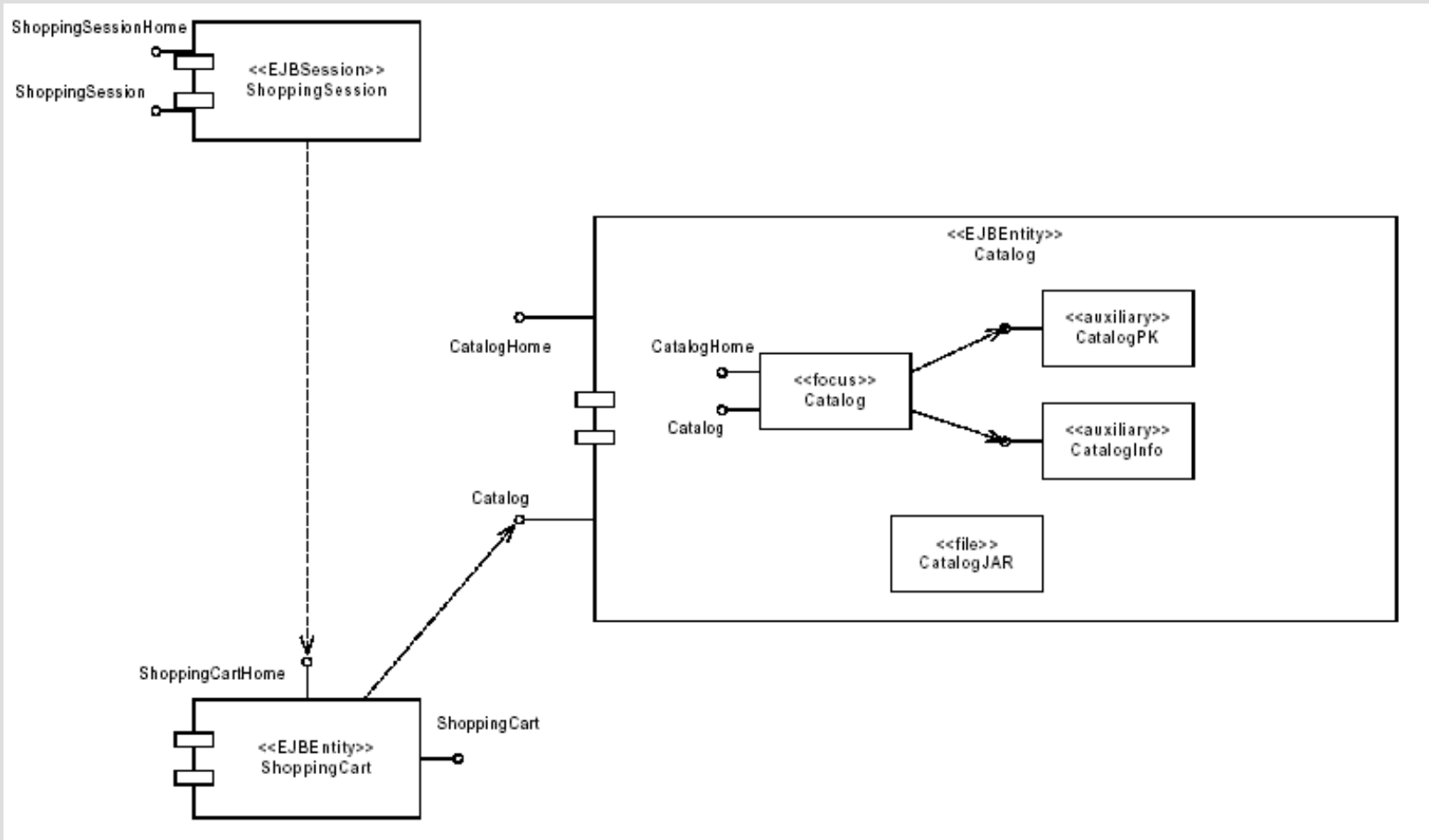

ShoppingSessionHome
<<Session>>
ShoppingSession
ShoppingSession
<<Entity>>
030303zak:Order
OrderHome
OrderHome
<<focus>>
:Order
Order
<<auxiliary>>
:OrderPK
OrderPK
<<auxiliary>>
:OrderInfo
OrderInfo
Order

Diagramma dei componenti



Un altro diagramma dei componenti

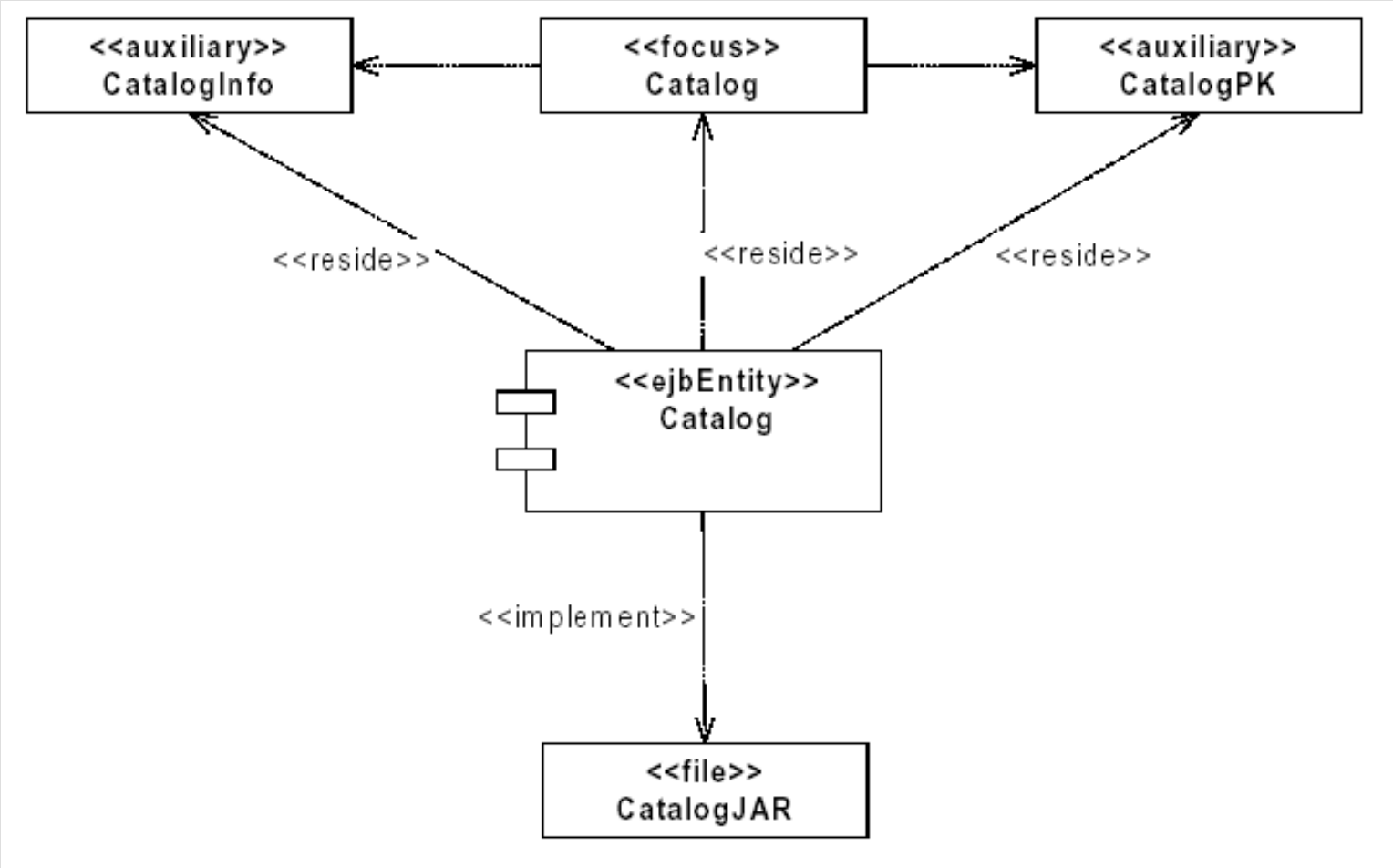
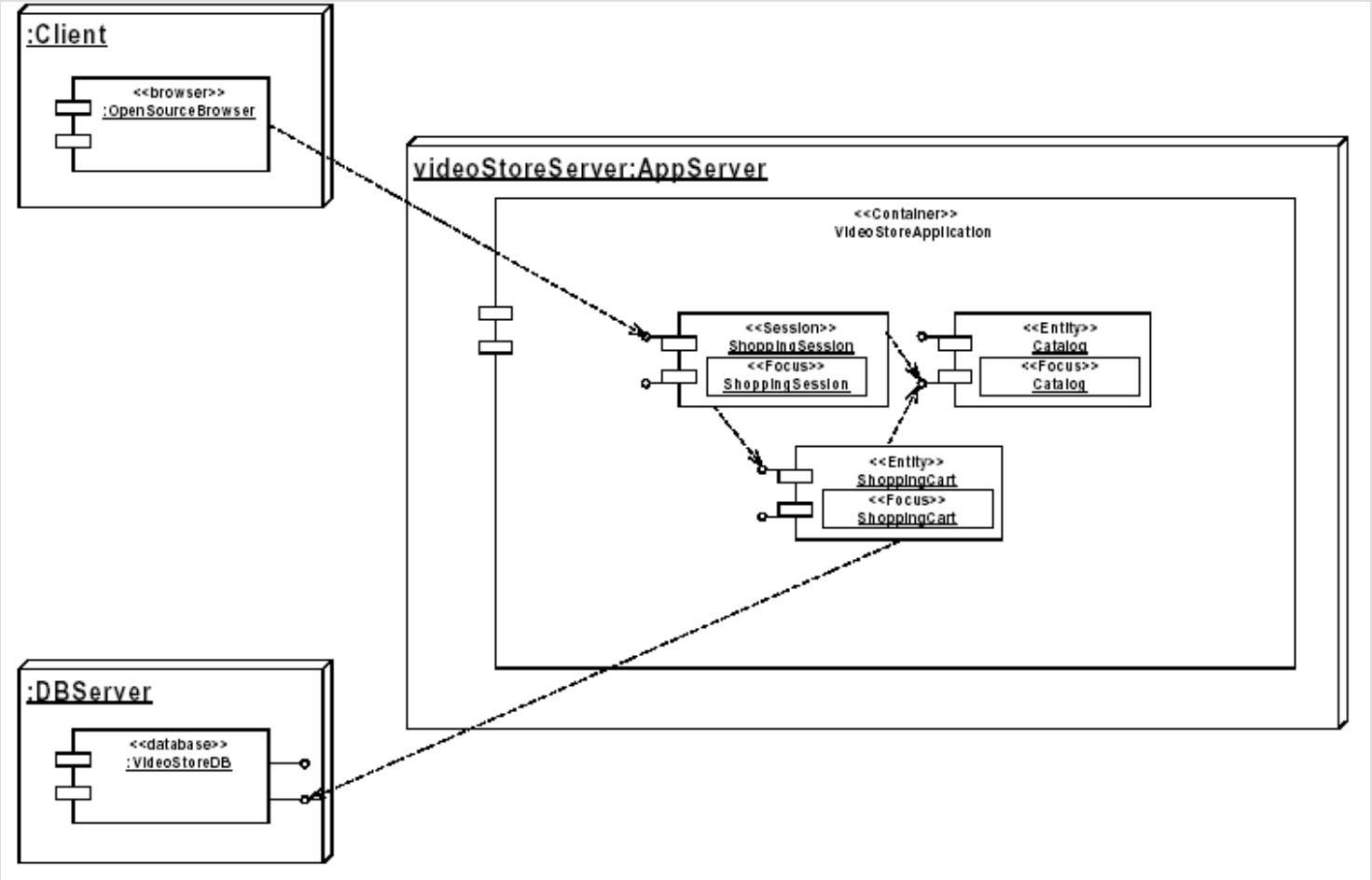


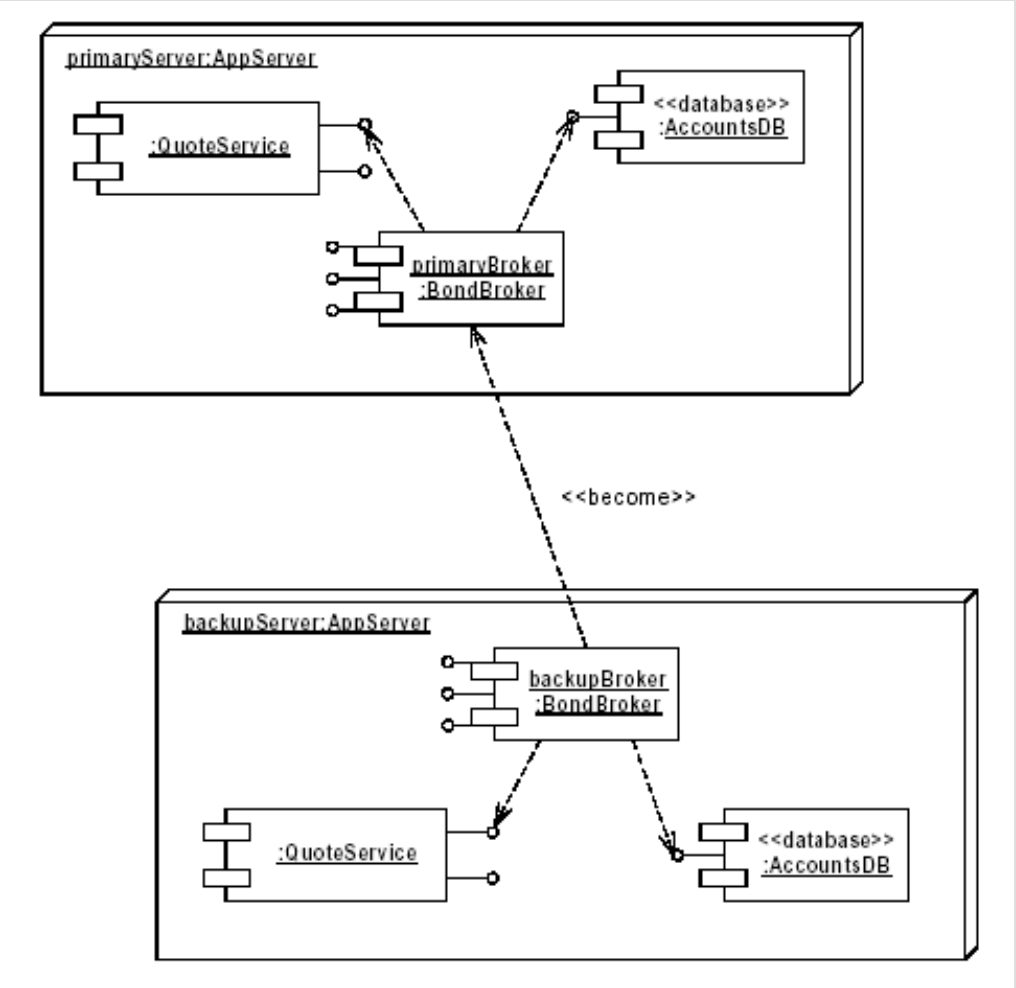
Diagramma di dispiegamento

- Mostra la configurazione di ogni elemento a run-time, oltre che delle componenti software, dei processi, e degli oggetti in esso presenti
- Può essere usato per mostrare il nodo in cui ogni componente entra in esecuzione
- Non è particolarmente usato o significativo

Un diagramma di dispiegamento



Un diagramma di dispiegamento



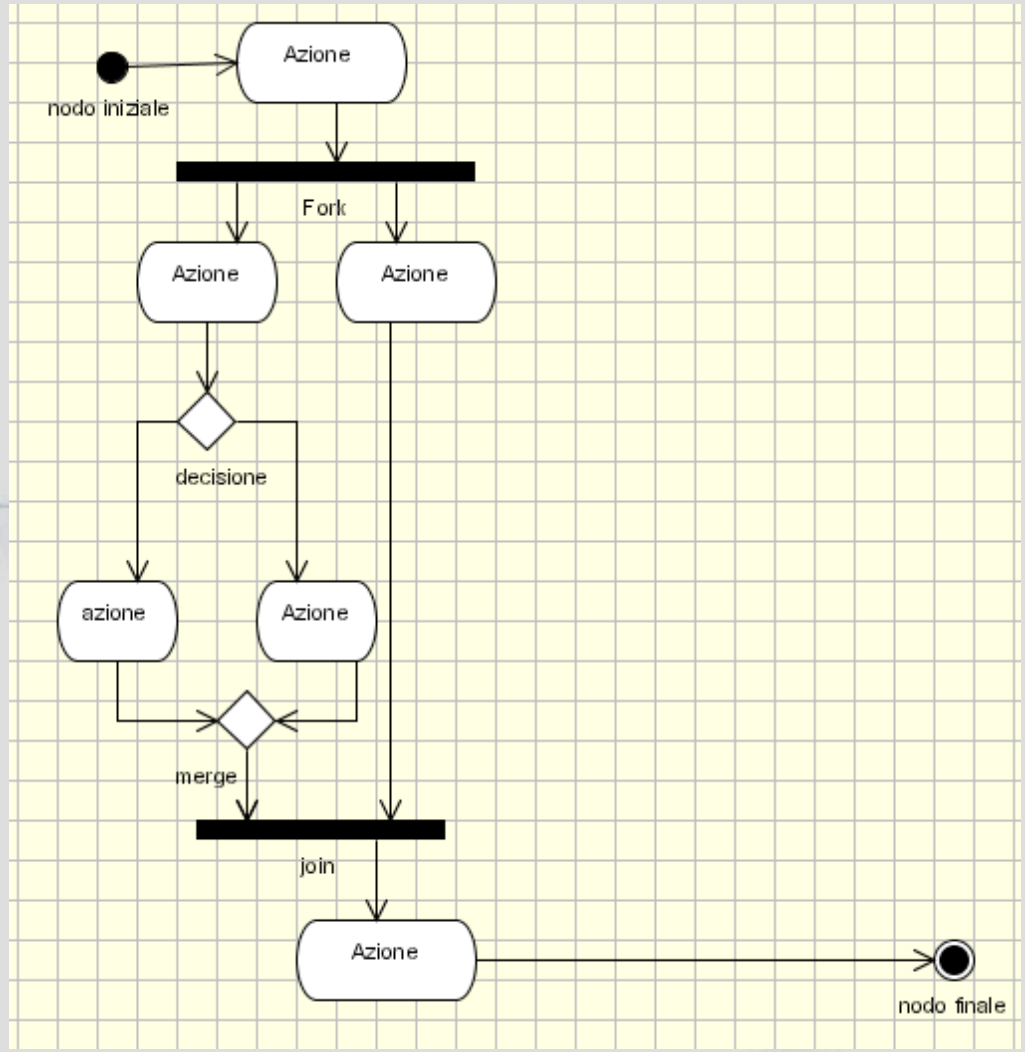
Diagrammi di attività

- Servono a descrivere
  - Workflow
  - Logica procedurale
- Simili ai diagrammi di flusso, con la differenza fondamentale che supportano la rappresentazione di elaborazione parallela
- Sono utilizzati nelle fasi di
  - **Analisi** per ridurre la complessità di un caso d'uso
  - **Design** di metodi complessi

Diagrammi di attività

- Il diagramma rappresenta una singola attività composta da azioni
- Le azioni possono essere implementate come
  - sotto-attività
  - metodi di classe

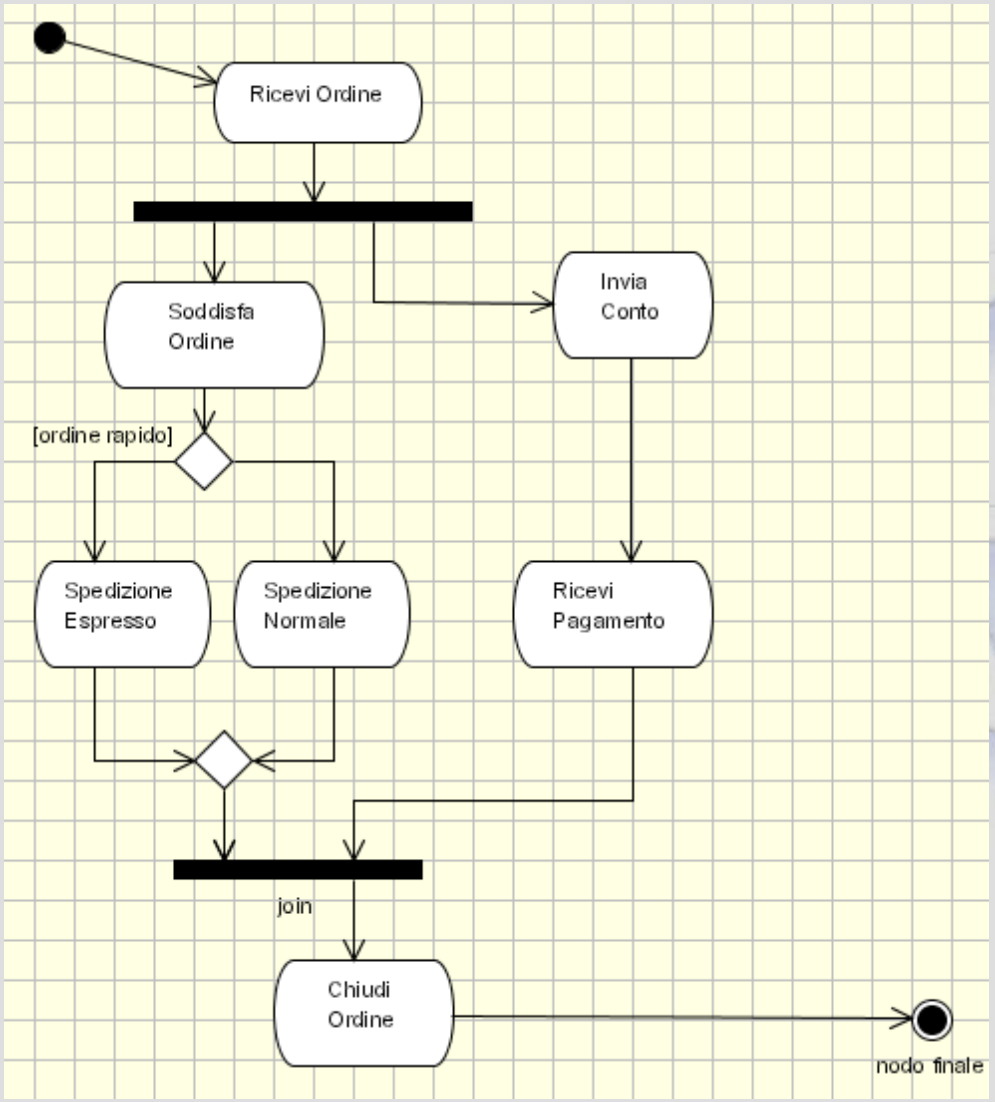
Diagrammi di attività - Notazione



Diagrammi di attività - Notazione

- Fork
  - Ha un solo flusso in ingresso e più flussi in uscita
- Join
  - Ha più flussi in ingresso e un solo flusso in uscita
- Comportamento condizionale
  - Decisione (Branch in UML 1)
    - Un singolo flusso in ingresso e due flussi in uscita
  - Merge
    - Più flussi in input e un solo output
    - Segnala la fine di un comportamento condizionale

Esempio



Le azioni SoddisfaOrdine, InviaConto e le successive sono svolte in parallelo.