OOSE Un caso di studio

Romina Eramo

romina.eramo@univaq.it

Sommario

- Argomento di studio
  - Requisiti
    - Lo sviluppo di una visione
    - casi d'uso Modellazione
    - Stabilire il modello di dominio
  - Analisi
    - Esecuzione di analisi dei Casi d'Uso
  - Design
    - Progettare l'architettura
    - Esecuzione di progettazione di dettaglio
    - Progettare l'interfaccia utente
  - Implementazione
  - Test e la distribuzione

Argomento di studio

- Questo caso di studio è un'applicazione che consente di automatizzare qualcosa con cui tutti sono a conoscenza: **una biblioteca**
- Fornisce funzionalità per la ricerca, di riserva, e prendere in prestito gli elementi, così come le possibilità di supporto per la gestione delle scorte e gli utenti della biblioteca

[UML2 Toolkit capitolo 11]

Requisiti

- Vari mezzi per suscitare, catturare e comunicare meno i requisiti del sistema
- Tra i mezzi più importanti sono
  - la visione del sistema
    - comunicata in un documento
  - casi d'uso del sistema
    - comunicata in UML come casi d'uso modelli
  - altri requisiti comuni artefatti
    - un glossario dei termini
    - specifiche complementari

Lo sviluppo di una visione

- Un insieme di requisiti di alto livello costituiscono la visione del sistema
  - E 'prodotto prima di scavare in requisiti dettagliati che prescrivono ciò che il sistema deve

Prima visione della Biblioteca

- Si tratta di un sistema di supporto per una libreria.
- La biblioteca presta libri e riviste per i mutuatari, che sono registrati nel sistema, come lo sono i libri e riviste.
- La biblioteca gestisce l'acquisto di nuovi titoli per la biblioteca. titoli popolari sono acquistati in copie multiple. Vecchi libri e riviste vengono rimossi quando sono fuori di data o in cattive condizioni.

- Il bibliotecario è un dipendente della biblioteca, che interagisce con i clienti (debitori) e il cui lavoro è supportato dal sistema.
- Un mutuatario può prenotare un libro o una rivista che non è attualmente disponibile nella libreria, in modo che quando è tornato o acquistati dalla biblioteca, che il mutuatario è notificato. La prenotazione viene annullata quando il mutuatario estrae il libro o una rivista o attraverso una procedura di cancellazione esplicita.

- Il bibliotecario può facilmente creare, aggiornare ed eliminare le informazioni sui titoli, mutuatari, prestiti e riserve nel sistema.
- Il sistema può essere eseguito su tutte le piattaforme popolare browser web (Internet Explorer 5.1+, Netscape 4.0+, e così via).
- Il sistema è facile da estendere con nuove funzionalità.

Modellazione casi d'uso

- attori
  - che interagiscono con il sistema
- Casi d'uso
  - ciò che il sistema libreria fornisce in termini di funzionalità a quegli attori

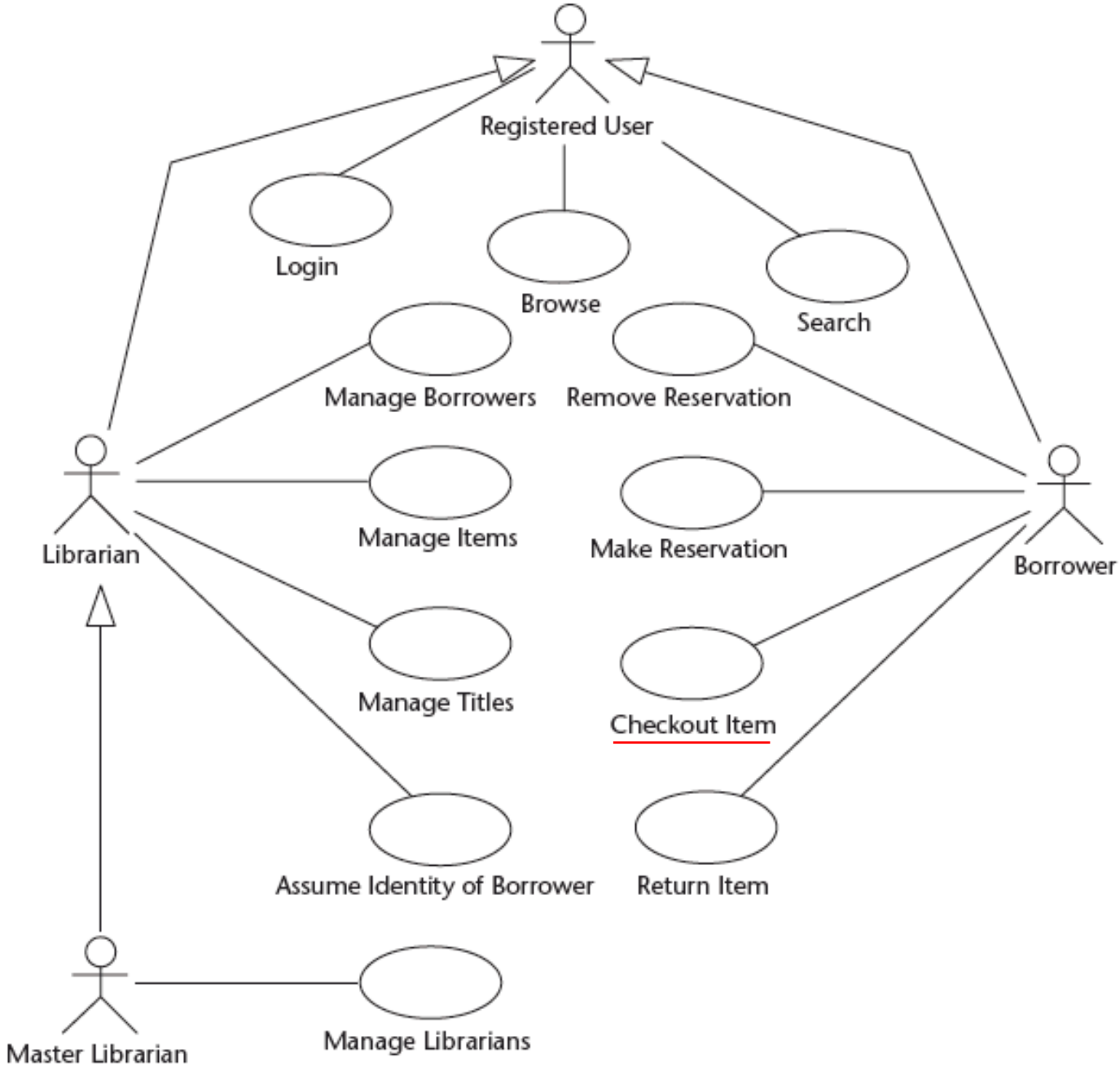
Modellazione casi d'uso

- Il **attori** sono bibliotecari e i mutuatari, perché entrambi sono gli utenti del sistema
  - I bibliotecari hanno capacità di gestione per aggiungere mutuatari, titoli e articoli
  - I mutuatari sono persone che il check out e di libri e riviste di riserva
  - Di tanto in tanto una biblioteca bibliotecario o di un altro può essere un mutuatario
  - Infine, un attore Maestro bibliotecario per la gestione dei bibliotecari pure
    - E 'possibile aggiungere un titolo al sistema prima che la libreria ha una copia (un oggetto), per consentire ai mutuatari di effettuare prenotazioni

Modellazione casi d'uso

- Il **casi d'uso** nel sistema di libreria sono i seguenti:
  - Accesso
  - Ricerca
  - Navigare
  - Prenotare
  - rimuovere la prenotazione
  - Pagamento
  - ritorno Articolo
  - gestire Titoli
  - gestire Articoli
  - gestire mutuatari
  - gestire bibliotecari
  - Assumere l'identità di mutuatario

Modellazione casi d'uso



Modellazione casi d'uso

Lo schema del flusso di base per il caso d'uso Pagamento :

1. Il mutuatario sceglie di eseguire un “Cerca” per i titoli desiderati.
2. Il sistema richiede al mutuatario di immettere i criteri di ricerca.
3. Il mutuatario specifica i criteri di ricerca e presenta la ricerca.
4. Il sistema individua i titoli corrispondenti e li visualizza per il mutuatario.
5. Il mutuatario sceglie un titolo da verificare.
6. Il sistema visualizza i dettagli del titolo, così come se c'è o non c'è

   è un elemento a disposizione per essere estratto.
7. Il mutuatario conferma che lui o lei desidera alla cassa la voce.
8. Il sistema controlla fuori l'oggetto.
9. Procedura 1 a 8 può essere ripetuto con la frequenza desiderata dal mutuatario.
10. Il mutuatario completa cassa
11. Il sistema avvisa un bibliotecario che il mutuatario ha concluso la sessione il Pagamento e visualizza le istruzioni per il mutuatario per raccogliere il contenuto.

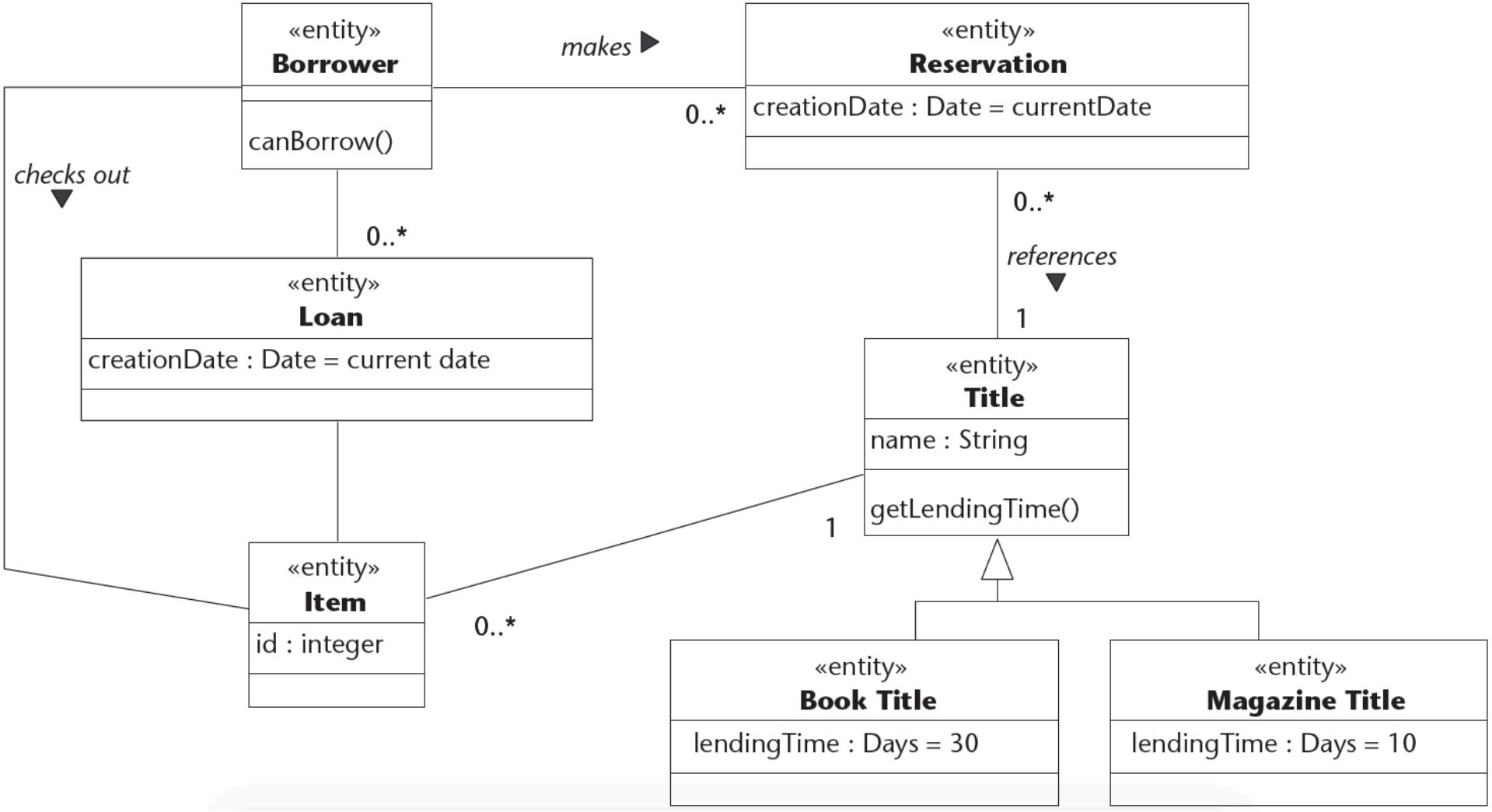
Stabilire il modello di dominio

- Un modello di dominio precoce è utile stabilire un nucleo di classi che rappresenta le cose nello spazio attività del sistema da costruire

Stabilire il modello di dominio

- Le classi di dominio nel sistema bibliotecario sono:
  - mutuatario
  - Titolo
  - Titolo del libro
  - Magazine Titolo
  - Articolo
  - Prenotazione
  - Prestito

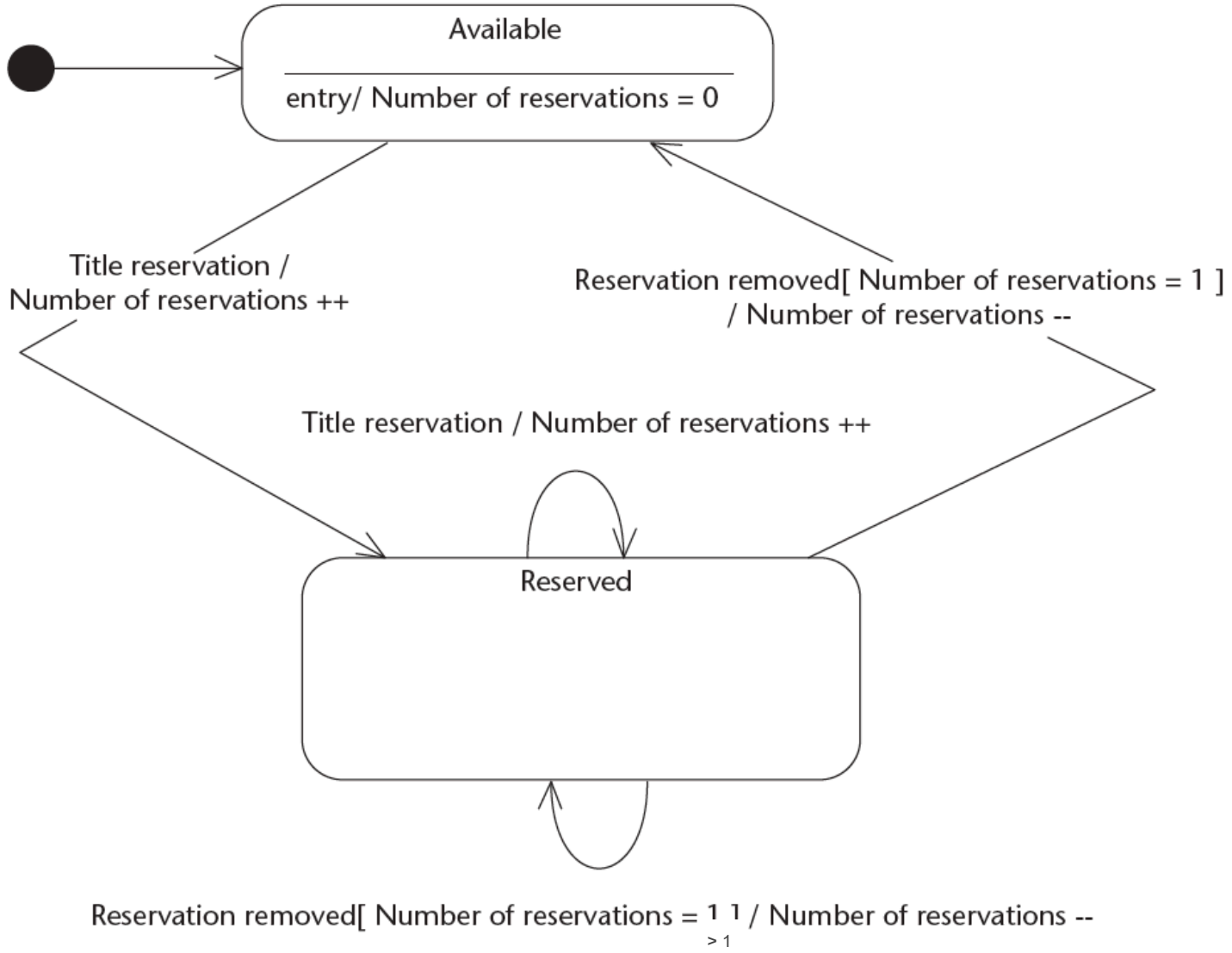
Stabilire il modello di dominio



Stabilire il modello di dominio

- Costruire UML statali diagrammi di macchine per le classi con gli stati interessanti
- I diagrammi mostrano i diversi stati che gli oggetti di queste classi possono avere, insieme con gli eventi che li rendono cambiare il loro stato

Stabilire il modello di dominio



Schema di macchina a stati per la classe Titolo

Analisi

- Mentre ci si sposta verso lo sviluppo di una soluzione al tuo problema, si inizia a
  - sfruttare più di UML per rappresentare senza ambiguità
  - e costantemente la vostra comprensione dello spazio problema in modi che facilmente si traducono in soluzioni proposte
- È possibile eseguire l'analisi in diversi modi
  - una tecnica nota come analisi caso d'uso

Esecuzione di analisi dei Casi d'Uso

- Un'analisi di dominio fornisce un semplice insieme di classi che cominciano a dimostrare le classi di analisi coinvolti nel sistema
- diagrammi statici (per esempio, i diagrammi di classe) diagrammi e dinamici (ad esempio, i diagrammi di sequenza) illustrano come le classi e le relative istanze collaborano per eseguire le azioni richieste dai casi d'uso

- Analisi classi sono in genere organizzati come una delle seguenti:

  - **<< soggetto >>** classi di entità rappresentano quelle classi che hanno un significato nel dominio e sono probabilmente lunghi tempi di vita nel vostro sistema

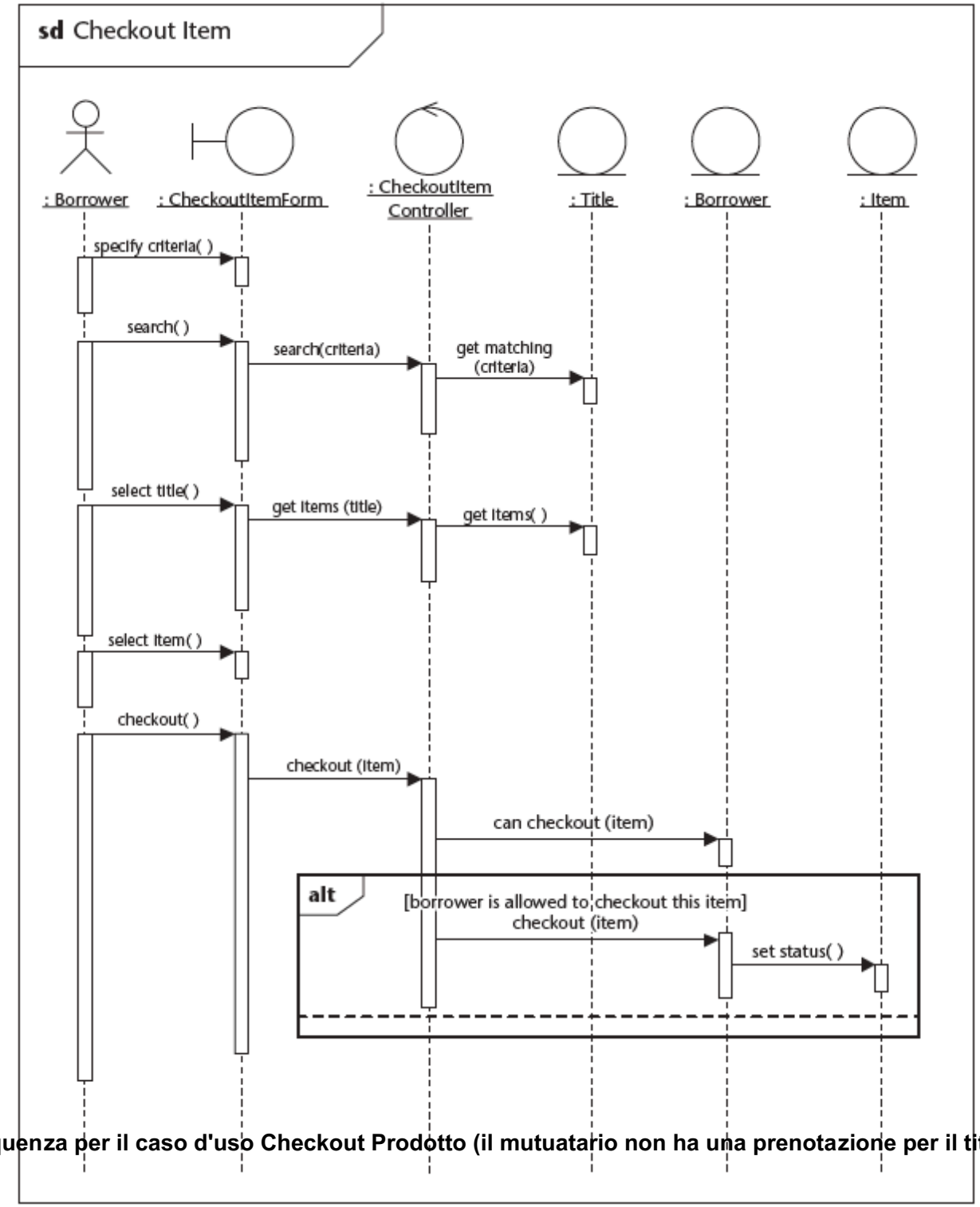
  - **<< confine >>** classi di interfaccia sono utilizzati per rappresentare le cose che sono necessarie per fornire un mezzo per attori di comunicare con il sistema

  - **<< controllo >>** classi di controllo sono utilizzati come segnaposto della logica coarsegrained di un caso d'uso

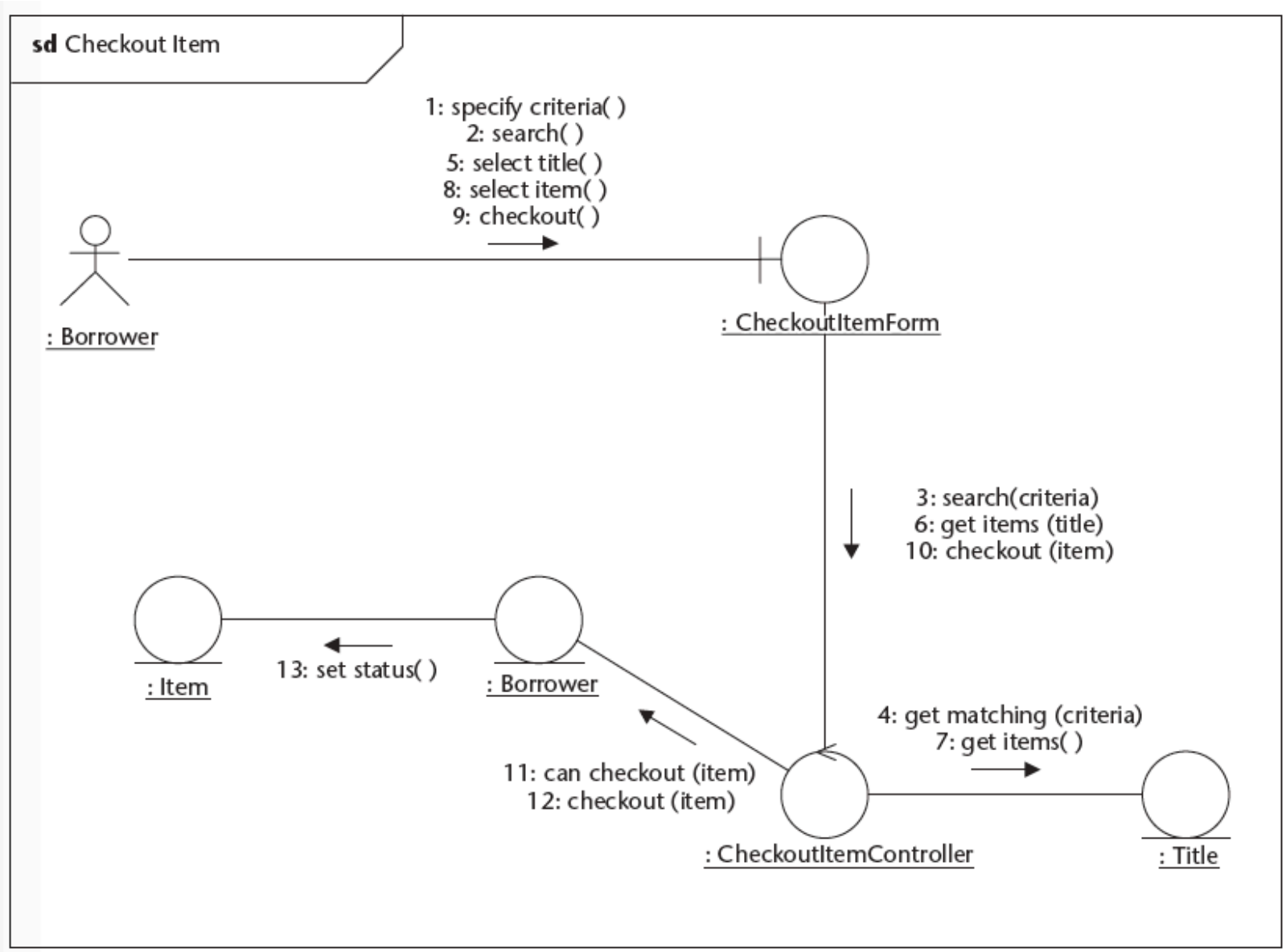
  [Object-Oriented Software Engineering, capitolo 5]

Esecuzione di analisi dei Casi d'Uso

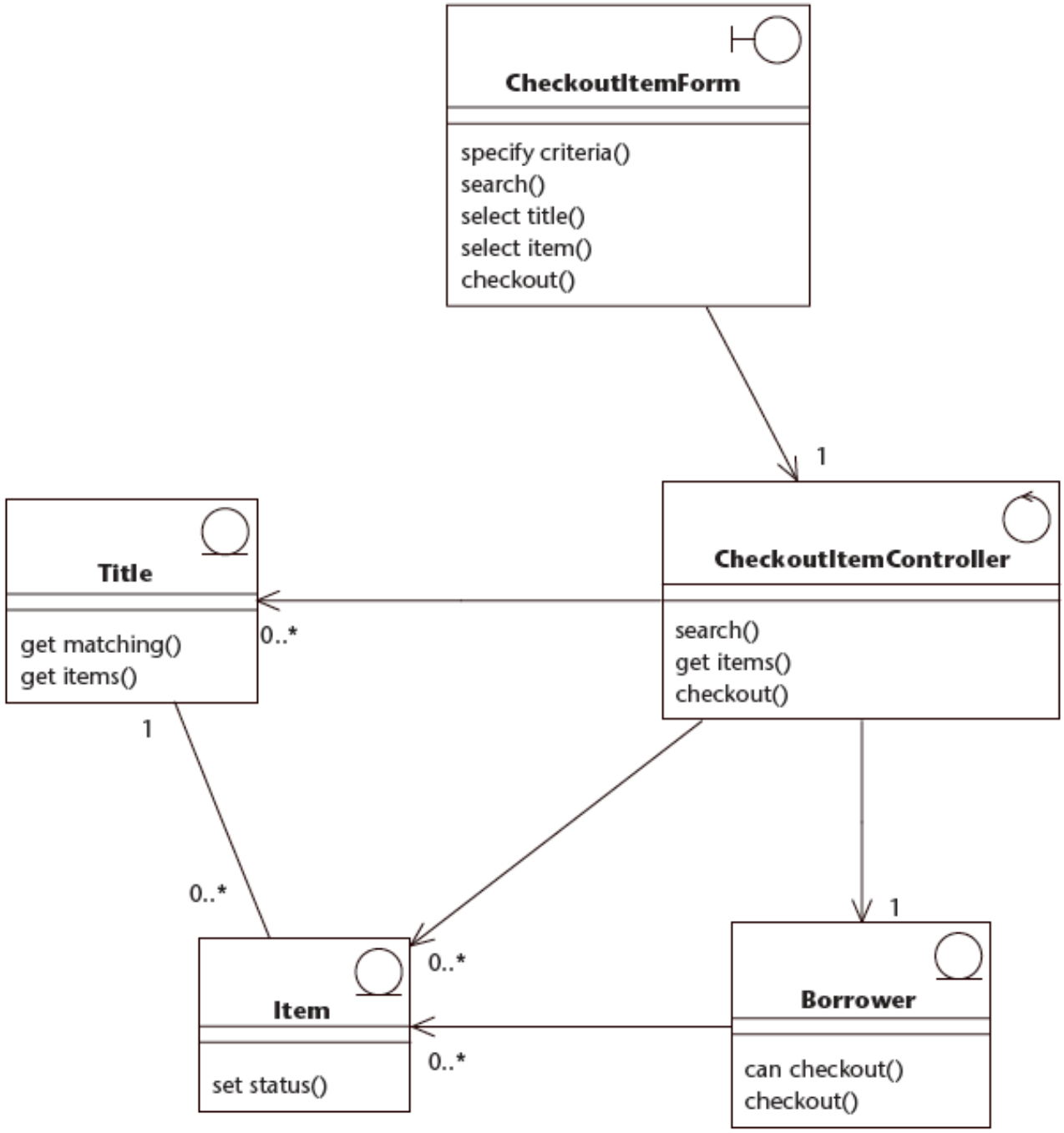
- Si raccomanda di sviluppare un diagramma di sequenza per ciascun flusso primario (o base) di un caso d'uso e di supporto a quelle con diagrammi occasionali alternativa interessante o potenzialmente mal capito o flussi eccezionali
- Questi possono essere integrate da diagrammi di comunicazione che meglio dimostrano i collegamenti richiesti tra gli oggetti per supportare il caso d'uso
- Ad esempio, i seguenti dati mostrano un diagramma di comunicazione di equivalente ad un diagramma di sequenza e parallelo una vista diagramma classi participanting



Un diagramma di sequenza per il caso d'uso Checkout Prodotto (il mutuatario non ha una prenotazione per il titolo)



Un diagramma di comunicazione per il caso d'uso Checkout Prodotto (il mutuatario non ha una prenotazione per il titolo)



Veduta delle classi Partecipare (VOPC) per l'utilizzo caso Checkout Articolo

Design

- La disciplina del design e il modello UML risultante espande e dettagli del modello di analisi tenendo conto di tutte le implicazioni tecniche e le restrizioni
- Lo scopo del progetto è quello di specificare una soluzione di lavoro che può essere facilmente tradotto in codice di programmazione
- Il disegno può essere diviso in due segmenti:
  - **Progettazione architettonica.** design di alto livello che definisce i pacchetti (sottosistemi), comprese le dipendenze e meccanismi di comunicazione principali tra i pacchetti
  - **Design dettagliato:** Dettagli Questa parte il contenuto dei pacchetti, in modo che tutte le classi sono descritte in dettaglio sufficiente per dare specifiche chiare per il programmatore che i codici della classe

Progettare l'architettura

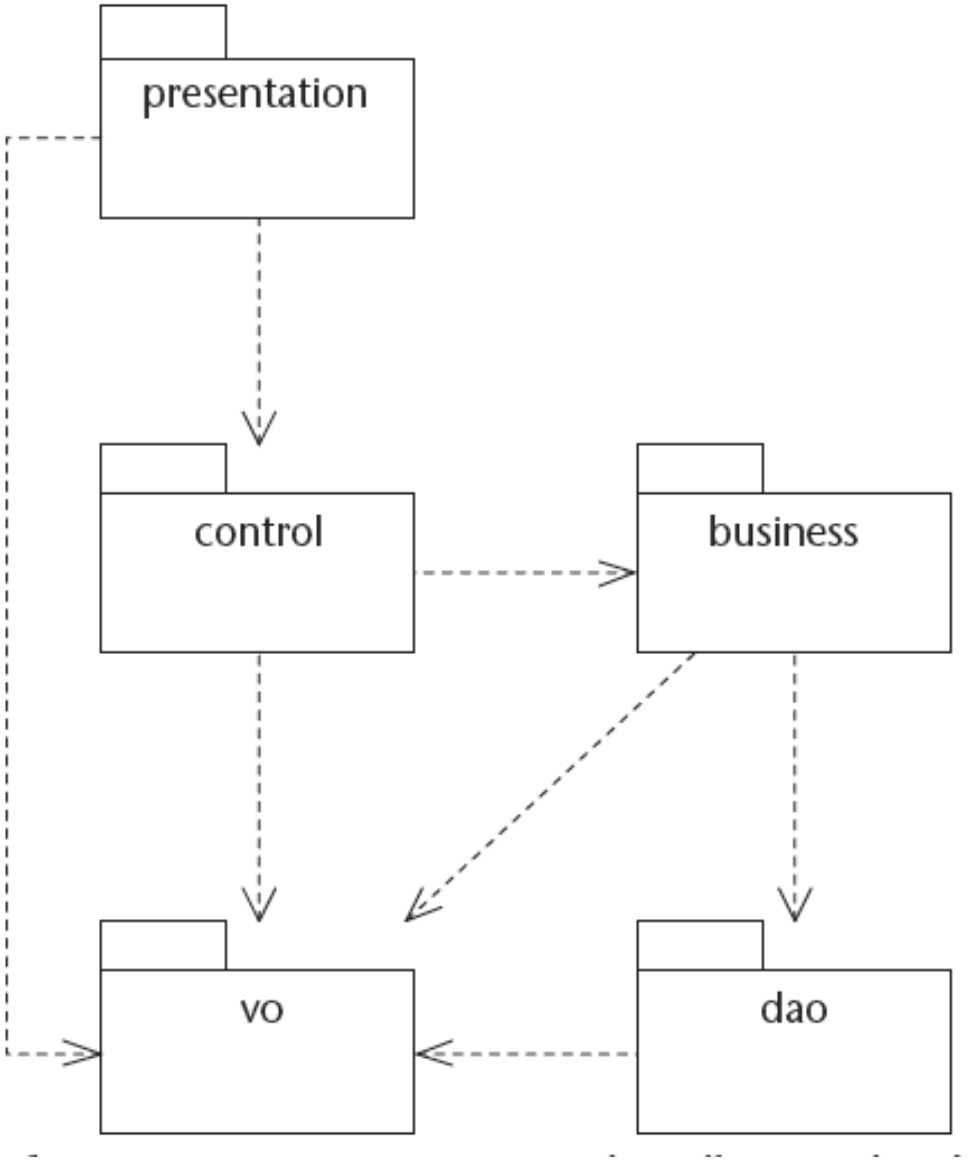
- Un'architettura ben progettato è la base per un sistema estensibile e di facile manutenzione
- Esempi di responsabilità architettonici critici:
  - Definizione pacchetto di chiavi e struttura di strato
  - La selezione o la progettazione di componenti riutilizzabili su larga scala
  - Impegnandosi a modelli chiave per funzionalità critiche e ripetuto
    - come il meccanismo di persistenza o modello di sicurezza
  - Guidando il design e gli standard di codifica per essere accolta nel corso della durata del progetto

Struttura del sistema

- Separando il sistema in pacchetti che servono sia strato e partizionare il sistema in fase di sviluppo

Struttura del sistema

- I pacchetti che forniscono stratificazione per il sistema bibliotecario



Struttura del sistema

- **Presentazione.** Questo contiene classi per l'intera interfaccia utente, per consentire all'utente di visualizzare i dati dal sistema e per immettere nuovi dati. Questo pacchetto coopera con gli oggetti di business (attraverso il valore Objects) ei pacchetti di controller. Il pacchetto interfaccia utente utilizza gli oggetti di valore ei dati di messaggi in modo che possa essere inviato al pacchetto di controllo.

- **Controller.** Le classi in questo pacchetto sono responsabili per la maggior parte del controllo delle applicazioni e forniscono anche meccanismi di “vigile” per aiutare le dipendenze delegare a loro e separare tra i pacchetti di presentazione e di dominio.
- **Attività commerciale.** Questo include le classi di dominio dal modello di analisi, come mutuatario, Titolo, Oggetto, prestito, e così via. Queste classi sono dettagliate nel disegno in modo che le loro operazioni sono completamente definiti. Il pacchetto oggetto di business collabora con il pacchetto del database tramite un rapporto associativo.

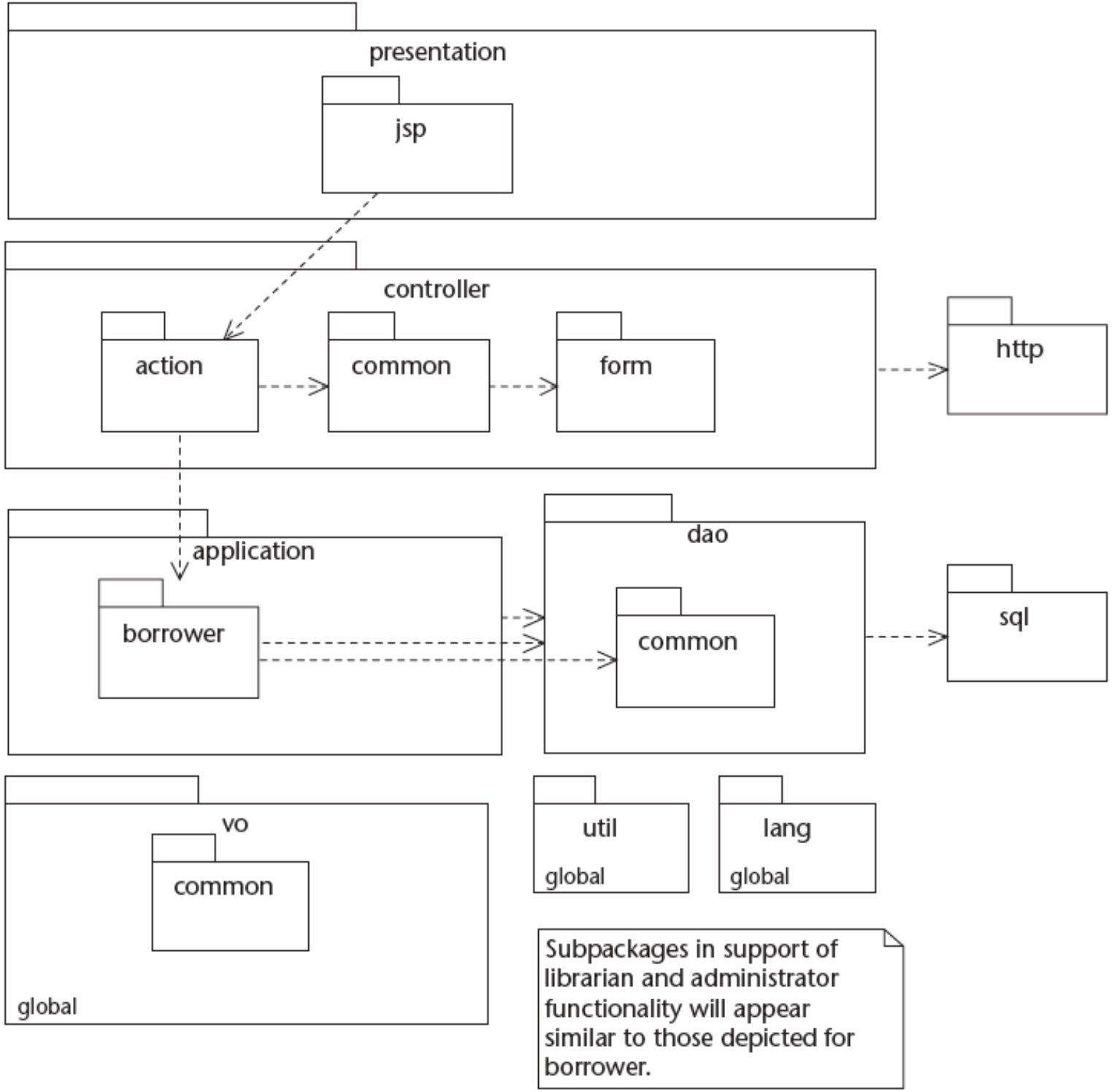
- **Dao.** Di nome *dao* perché contiene un Data Access Object (DAO) che rappresenta un modello di progettazione di uso comune che racchiude tutte le attività i dati relativi in classi discrete. Questo è il mezzo per firewall o limitare la conoscenza di come le cose sono persistenti.
- **Vo.** Il *vo* il pacchetto è così chiamato in quanto contiene semplici oggetti di valore, che sono rappresentazioni leggere di cose dominio legate. Il valore dell'oggetto (VO) è ancora un altro modello comune utilizzato per limitare il passaggio di oggetti pesanti e del traffico attraverso i sistemi aziendali.

Struttura del sistema

- Diverso livello di dettaglio può essere fornita
  - Pacchetti espandono sia internamente che attraverso i rapporti con machanisms implementazione.
    - ad esempio, preesistente Java standard e pacchetti J2EE

"Java" è sia il nome del linguaggio di programmazione orientato agli oggetti e un termine contenitore per "tutto Java". La maggior parte delle persone significa "core Java" o "JSE" (Java Standard Edition) quando dicono "Java".

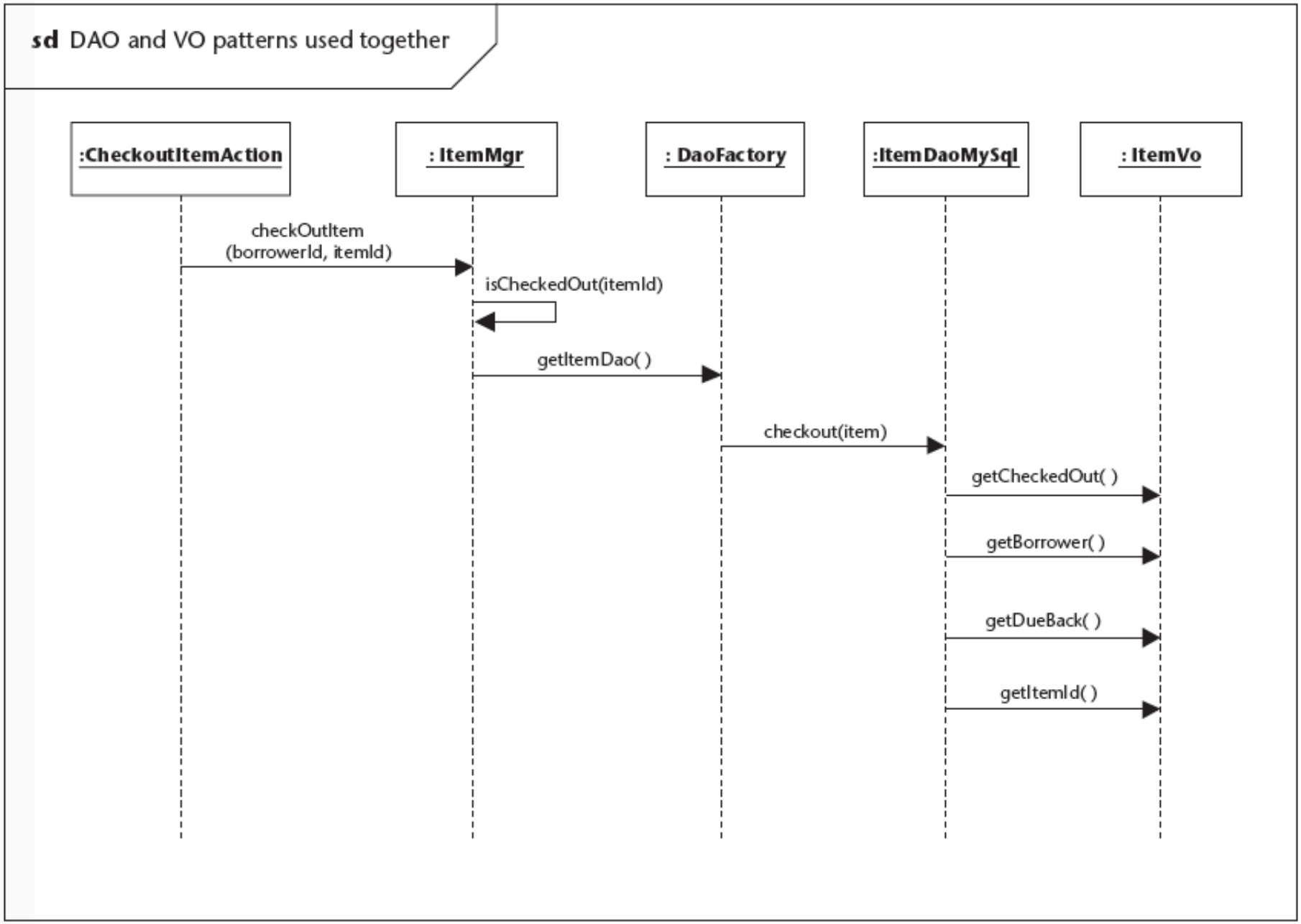
"J2EE" (o semplicemente "JEE" ora) è Java Enterprise Edition, che si compone di nucleo Java con un potente set di librerie. Queste librerie sono per lo più utile se le applicazioni che stai in via di sviluppo devono essere a più livelli, fault-tolerant e / o distribuito, in esecuzione su server applicativi.


presentation
jsp
controller
action
common
form
http
application
borrower
dao
common
sql
vo
common
global
util
global
lang
global
Subpackages in support of librarian and administrator functionality will appear similar to those depicted for borrower.

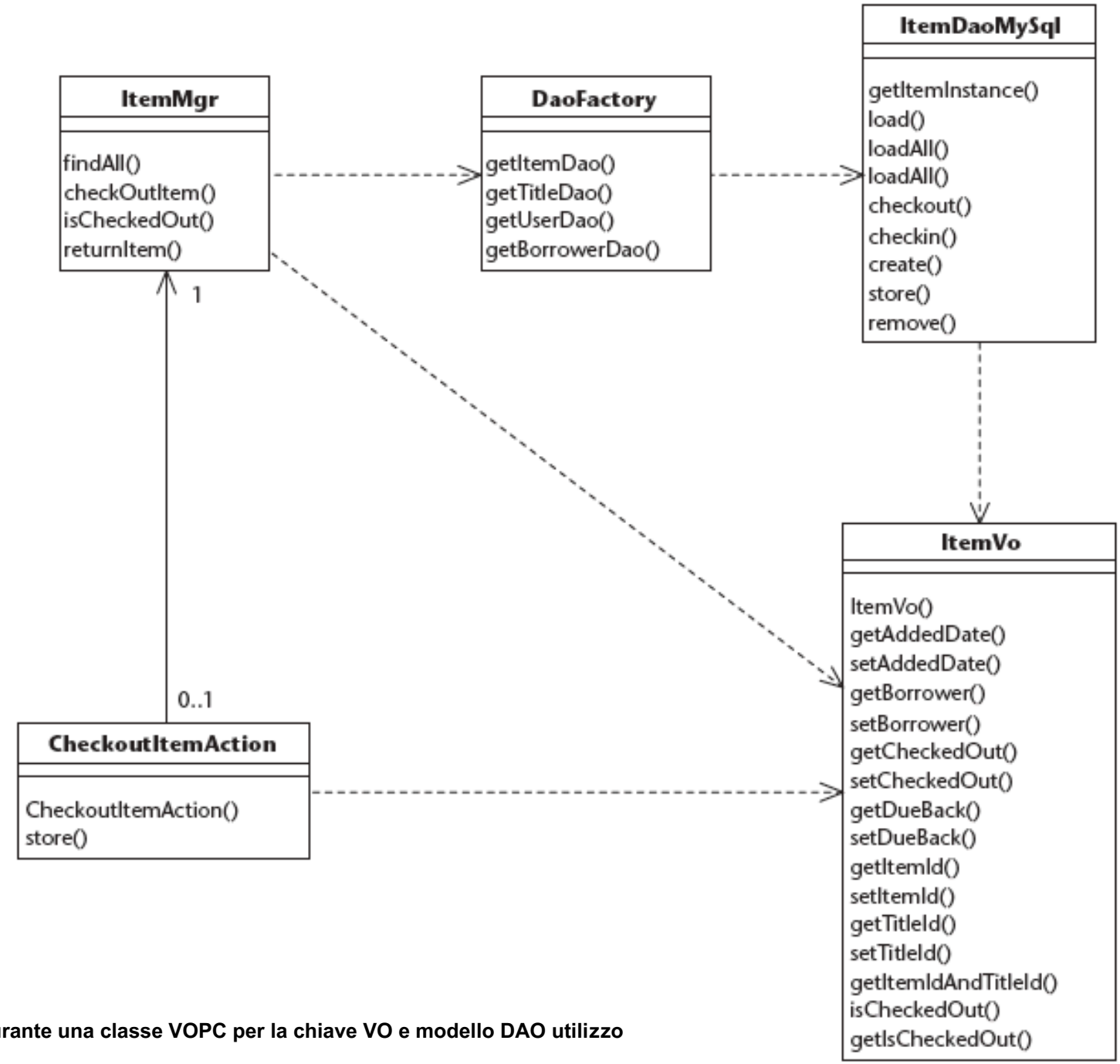
Meccanismi di architettura

- meccanismi architettonici descrivono le tecniche per affrontare, tra le altre cose, la persistenza, sicurezza, gestione degli errori, e interfacce

- meccanismi architettonici devono essere modellate per dare l'architetto (s) e sviluppatori una vista su meccanismi comuni che trascendono comunque utilizzabile una sola



Un diagramma di sequenza che evidenzia sia uso VO e DAO sul sistema libreria



Un diagramma raffigurante una classe VOPC per la chiave VO e modello DAO utilizzo

Modelli di progettazione

- Design / modelli architettonici sono, tra le altre cose, ben collaudati soluzioni riutilizzabili utilizzati per risolvere un particolare problema
  - **Model-View-Controller (MVC)**

  Il pattern MVC è più di un modello architettonico che utilizza modelli di progettazione per imporre il suo modello di alto livello. L'architettura MVC consente una netta separazione della logica di business, dati e logica di presentazione. Il modello contiene i dati aziendali, la vista genera contenuto del modello, e il controllore definisce il comportamento dell'applicazione.

  - **front controller**

  Questo modello di progettazione è fornito dal framework Struts che supporta il pattern MVC. Il front controller fornisce il primo punto di contatto per gestire le richieste in ingresso al ricorso.

  - **Data Access Object (DAO)**

  Un DAO viene utilizzato per astratto e incapsulare tutti gli accessi ad una fonte di dati. Questo ti permette di andare in un posto per fare le modifiche dei dati e non ha un impatto sui livelli di lavoro o di presentazione.

  - **Valore Object (VO)**

  Un VO viene utilizzato per incapsulare i dati aziendali.

Model-View-Controller (MVC)

- MVC è inventato da Trygve Reenskaug

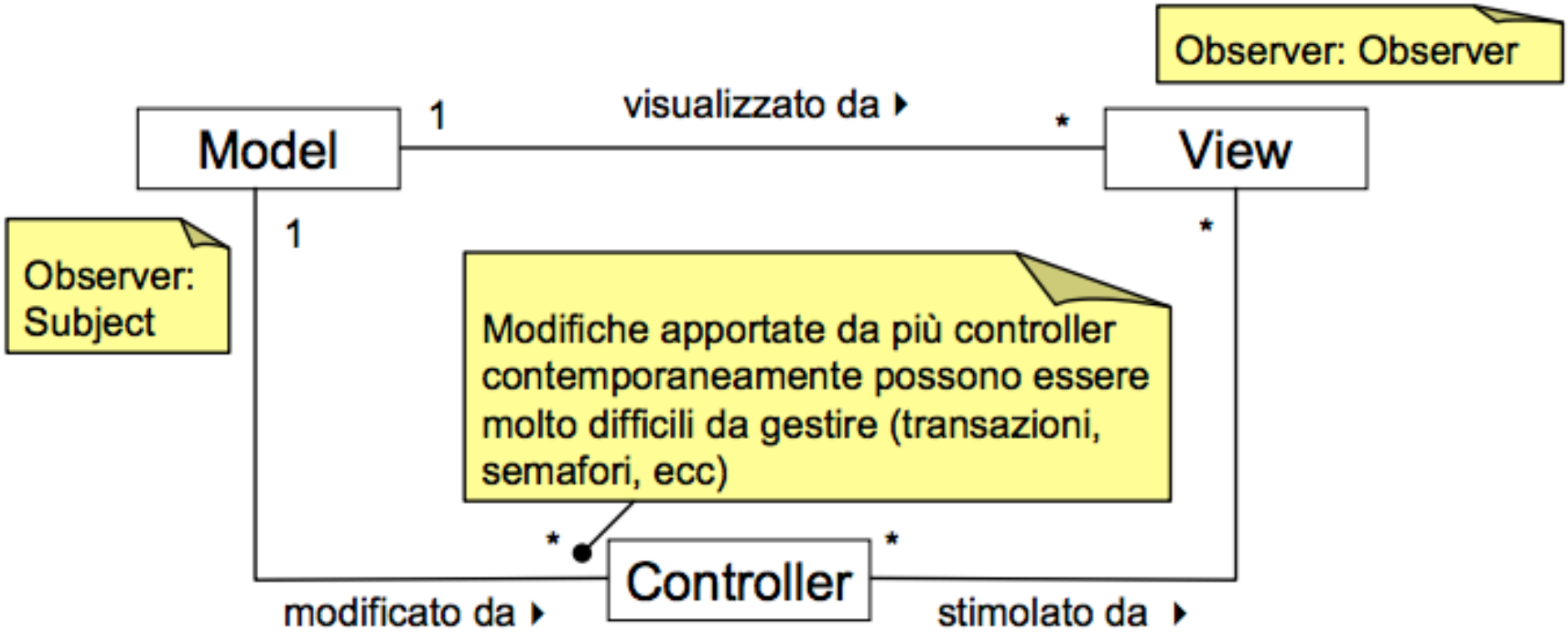
- E non è un modello di interfaccia utente: Si tratta di un modello architettonico.

- MVC è più su modello mentale e abbiamo concettualizzare il sistema con esso.

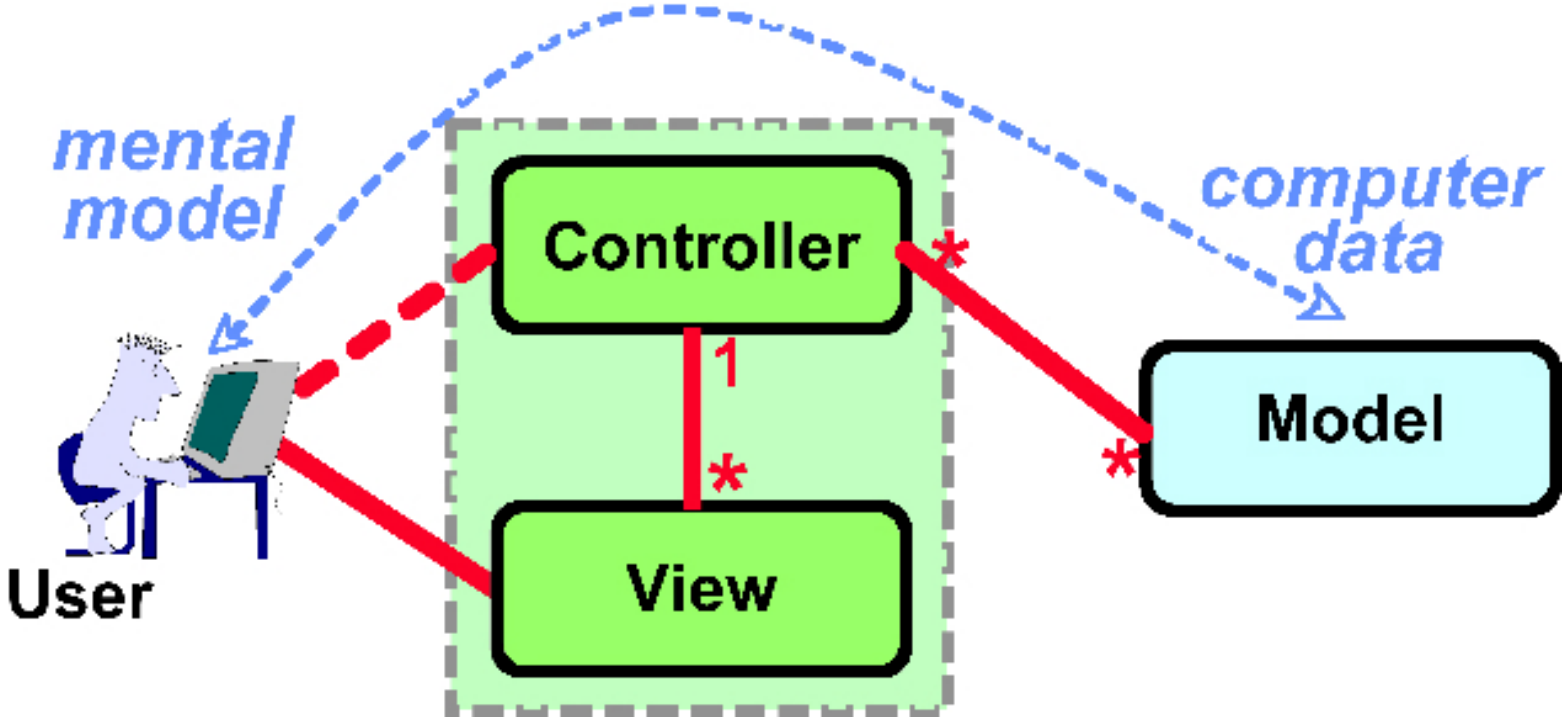
- Si tratta di vista del modello e l'utente mentale del sistema

- Vediamo questa volta e il tempo di nuovo modello in più punti. Ha una buona separazione degli interessi (SoC)

Model-View-Controller (MVC)



Model-View-Controller (MVC)

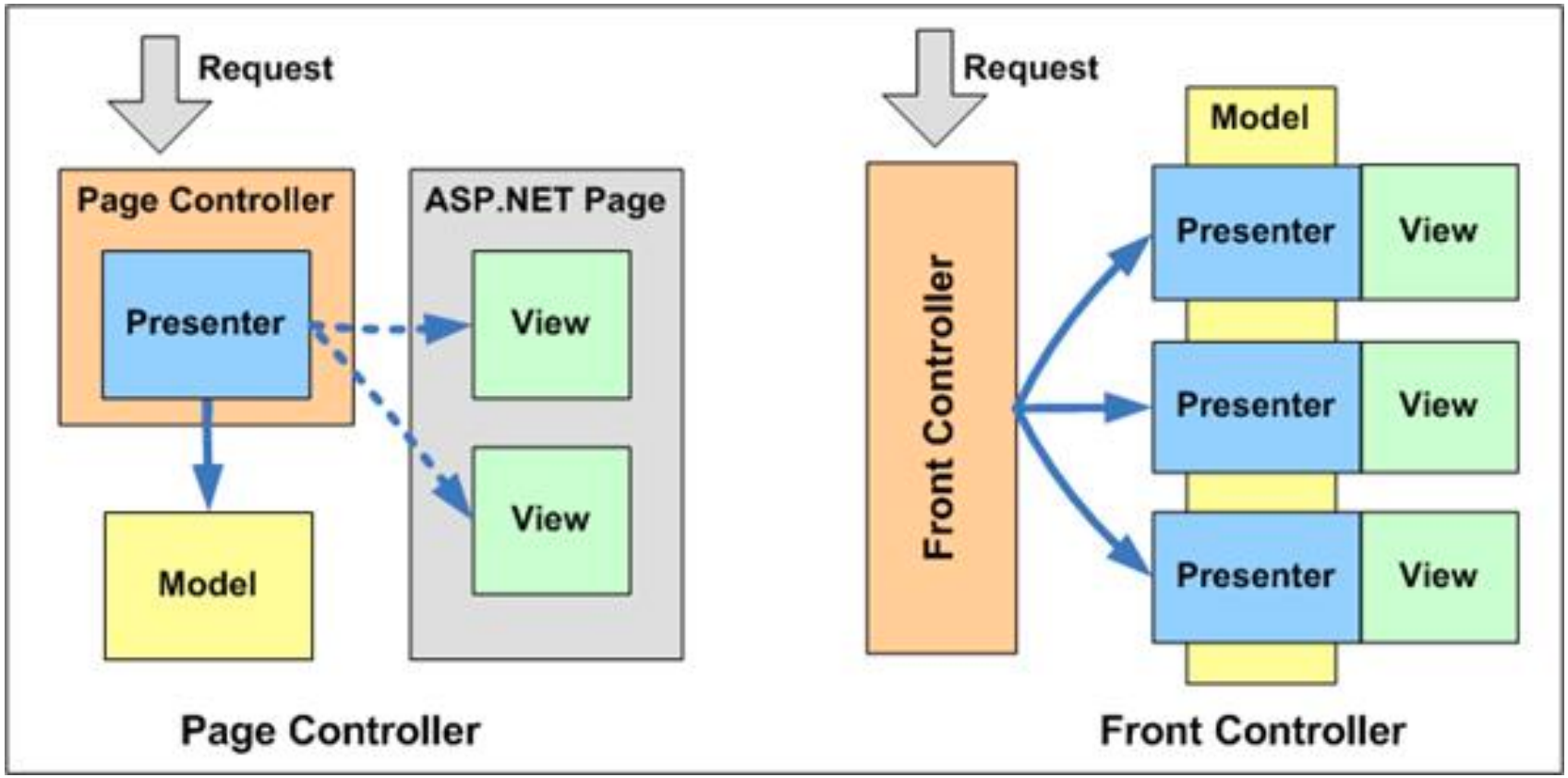


L'obiettivo del controllo è quello di coordinare viste multiple

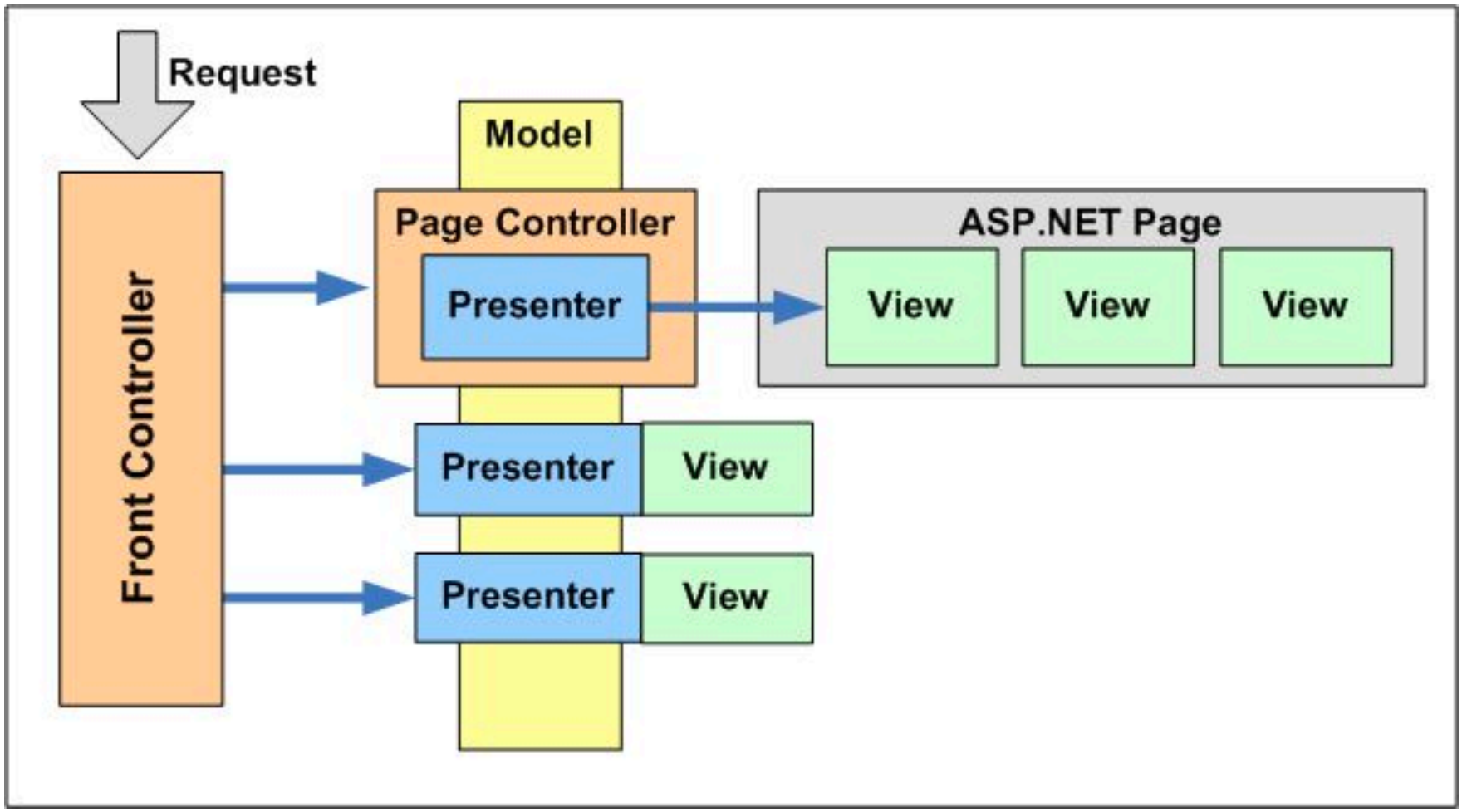
front controller

- Modello architetturale che sì appli alla Progettazione di Applicazioni web.
- Fornisce un punto di ingresso centralizzato per la Gestione delle Richieste
- framework per applicazioni web di Molti implementano il modello anteriore Controller, Tra cui:
  - Primavera, framework MVC Java
  - Struts, framework open source per lo Sviluppo di Applicazioni web su Piattaforma Java EE
  - Torta, codeigniter, Drupal, Symfony, Yii e Zend Framework, quadri Scritti in PHP

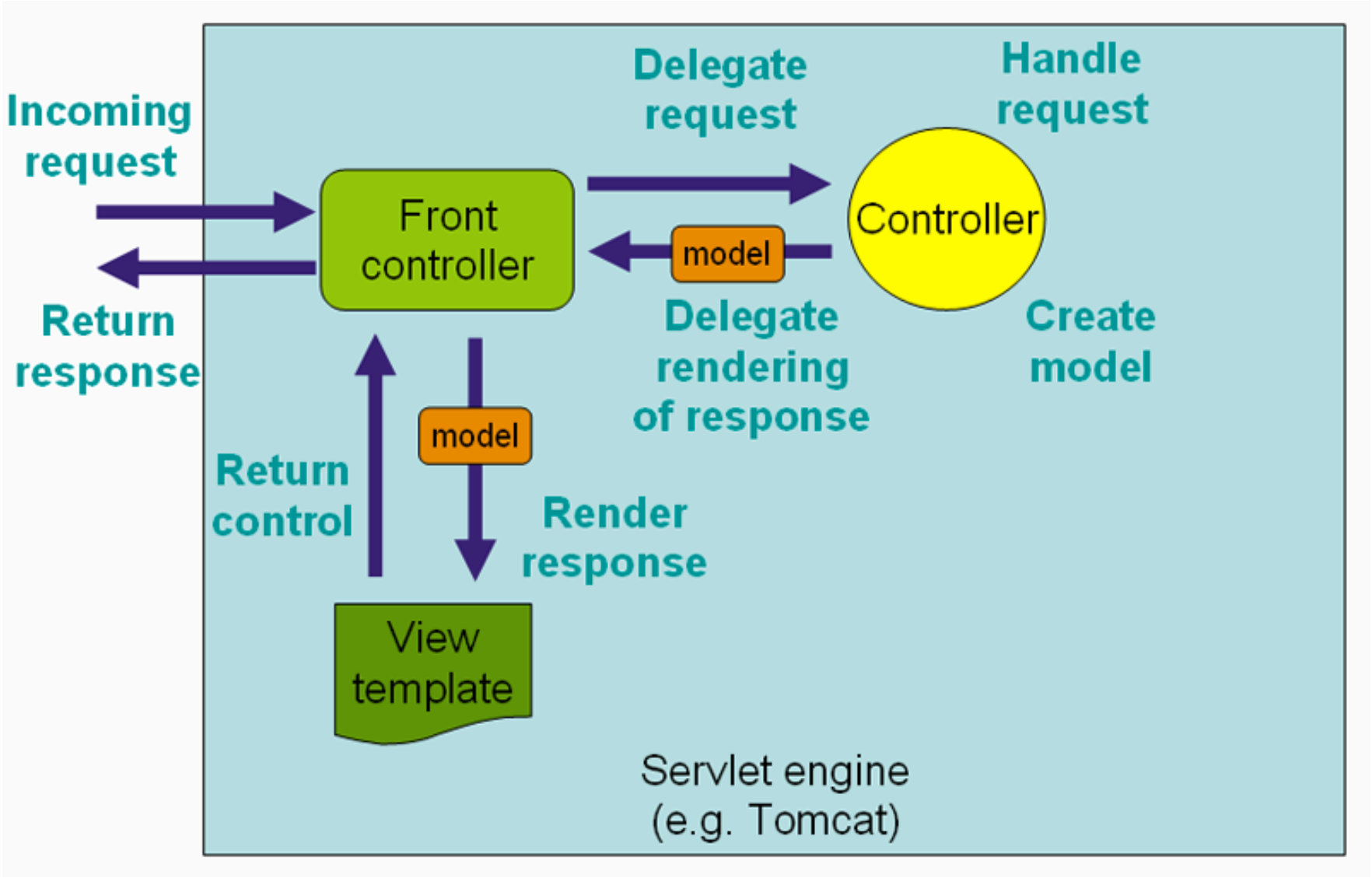
front controller



front controller



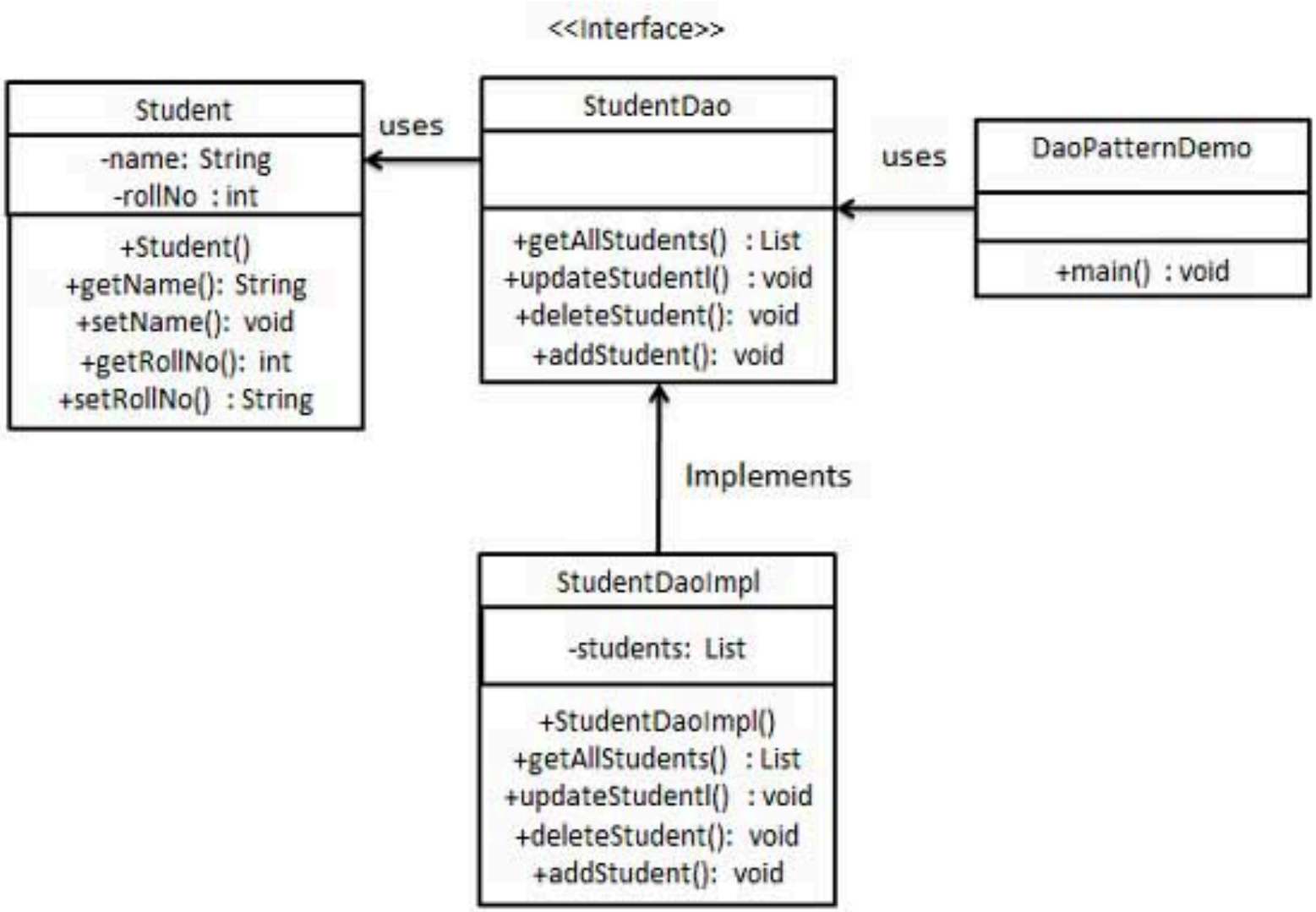
front controller



Data Access Object (DAO)

- architetturale del modello per la Gestione della persistenza:
  - Classi con Informazioni relative Metodi rappresentano Entità tabellare di un RDBMS
  - Usato principalmente in Applicazioni web di tipo Java EE, per stratificare e isolare l'ACCESSO at a tabella tramite query (poste all'interno dei Metodi della classe)
  - I Metodi del DAO con le rispettive interrogazione Dentro verranno richiamati Così Dalle classi logica di business della
- Il Vantaggio Relativo all'uso del DAO E dunque il mantenimento di Una rigida separazione Tra le Componenti di un'applicazione, le Quali potrebbero Essere il "Modello" e il "controllo" in un'applicazione Basata sul paradigma MVC.

Data Access Object (DAO)



Valore Object (VO)

- Creare un oggetto che è un attributo di un altro oggetto, senza alcuna identità relativo al dominio e che può essere intercambiabili tra gli oggetti che utilizzano lo stesso attributo.

- Avete una persona e una persona ha un indirizzo. Hai una società e una società ha un indirizzo. Il database può avere un tavolo e un tavolo PersonAddress companyAddress. Tuttavia, non v'è alcuna necessità di creare un oggetto PersonAddress e un oggetto Azienda Indirizzo.

Valore Object (VO)

- Oggetti di valore sono più importanti: VO sono probabilmente gli oggetti più importanti voi ragazzi dovrebbe creare

- Sono piccoli, altamente focalizzata, oggetti immutabili

- Sono sempre tipo di valore. Se avete bisogno di confrontare due VO di (per l'uguaglianza), il contenuto decideranno la sua uguaglianza

Valore Object (VO)

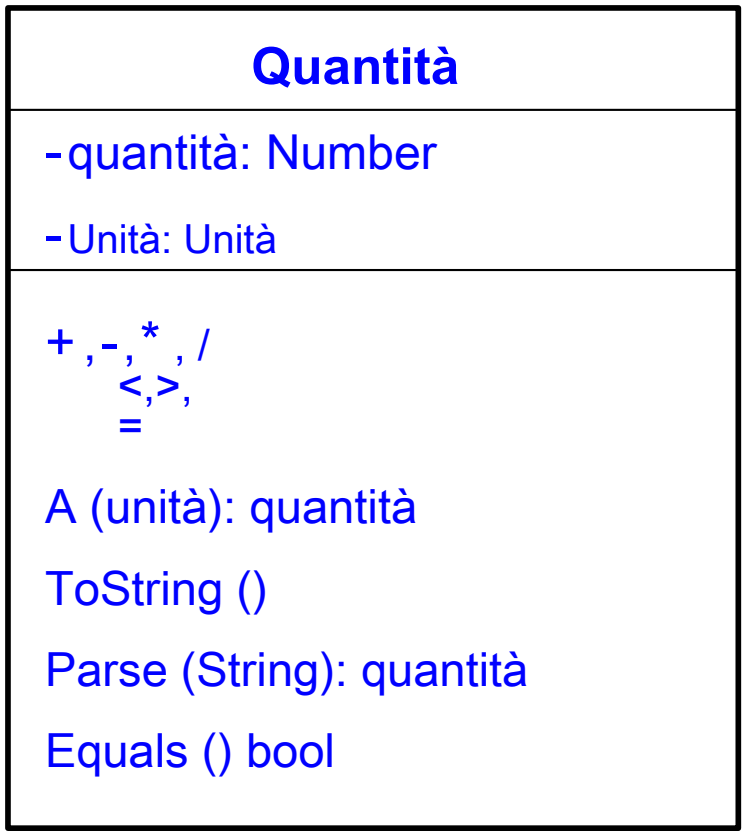
- VO il tuo grado di migliorare il vostro disegno, la riutilizzabilità e la qualità del codice

- ad esempio denaro, Indirizzo, Gamma, Quantità, dimensioni, data, Point, Identità (Id), Chiave e Enums

- Idealmente ci sarà mai alcun oggetto di int tipo, long, stringhe, doppie, decimali in vero e proprio business.

Valore Object (VO)

Valore Object (VO): ***Quantità***



design Policies

- Stabiliamo alcune politiche sul progetto di far rispettare i principi della progettazione architettonica
- accoppiamento lasco tra gli strati architettonici:
  - **Un cambiamento a un livello non richiede un cambiamento in altre parti dell'applicazione.**
  - **Separazione degli interessi. Questo consente agli sviluppatori e ai progettisti di concentrarsi su ciò che sanno fare meglio.**

  - Tutte le chiamate verso i DAO sono fatte solo tramite l'oggetto di business adeguato. Ad esempio, per accedere al BorrowerDao, si chiama l'oggetto business mutuatario.
  - JSP accedere soltanto il modello (nel pattern architetturale MVC) tramite un oggetto valore. JSP non accedono direttamente oggetti connessi alle imprese.
  - JSP contengono solo codice di presentazione-related. No funzionalità di business è contenuto nel JSP.
  - JSP non contiene alcun codice scriptlet Java. Invece, usano librerie di tag per mostrare o nascondere alcuni dati di presentazione relativi condizionalmente.
  - classi azioni sostenute dal framework Struts contengono solo codice relativo ai estrarre i dati dagli oggetti di richiesta, passando i dati per l'oggetto di business adeguato, e tornando alla JSP appropriata. Non esiste un codice di business in queste classi di azione. Ciò consente di supportare più di un tipo di cliente (diverso da un browser Web).

  - Dal momento che FormBeans sono un meccanismo di implementazione del framework Struts, abbiamo scelto di collegare le FormBeans per l'oggetto valore e poi agire sul valore dell'oggetto in nessun'altra parte dell'applicazione. In questo modo se avete bisogno di usare un quadro diverso, è possibile farlo con un impatto minimo.

Esecuzione di Progettazione dettagliata

- Lo scopo del progetto dettagliato è quello di impiegare la progettazione architettonica strategico e tutta la sua guida implicita associati e aderire ad esso, mentre evolvendo i risultati delle analisi, o di iterazioni anche prima, in una rappresentazione precisa e semanticamente ricco di ciò che si vuole costruire in codice

- Progettazione esecutiva comprende
  - affinando le specifiche di classe individuali per garantire che essi siano completi
  - soddisfare le politiche del progetto per la propagazione eccezione
  - segnalazione, la gestione generale, e così via
- Dopo un paio di iterazioni, il modello di progettazione deve essere sufficientemente dettagliata in modo da poter consegnare il modello a uno strumento
- Le parti visive UML sono insufficienti per descrivere tutto quello che serve

  - la combinazione di elementi grafici e non grafici UML elementi sono sufficienti per raggiungere l'obiettivo

Esecuzione di Progettazione dettagliata

- Le parti visive UML sono insufficienti per descrivere tutto quello che serve
  - la combinazione di elementi grafici e non grafici UML elementi sono sufficienti per raggiungere l'obiettivo
- Questa attività può concentrarsi su una base chiave package-by-package
  - per esempio, i pacchetti di business e di presentazione descritti in alcune delle decisioni di progettazione dettagliata sulla base di analisi originale

Progettare l'interfaccia utente

- Iniziata durante l'analisi, viene eseguita separatamente ma in parallelo all'altro lavoro di progettazione
- La decisione dovrebbe sempre garantire che i requisiti per l'applicazione sono soddisfatte
- L'interfaccia utente dell'applicazione Biblioteca è composta da
  - una pagina principale, con un menu e una grafica appropriata, da cui tutte le altre pagine dell'applicazione può essere raggiunto
  - le altre pagine tipicamente presentano una notifica del ricorso e sono associati a un particolare caso d'uso
  - in alcuni casi, una singola interfaccia utente, o una pagina, potrebbero mappare a più casi d'uso

Implementazione

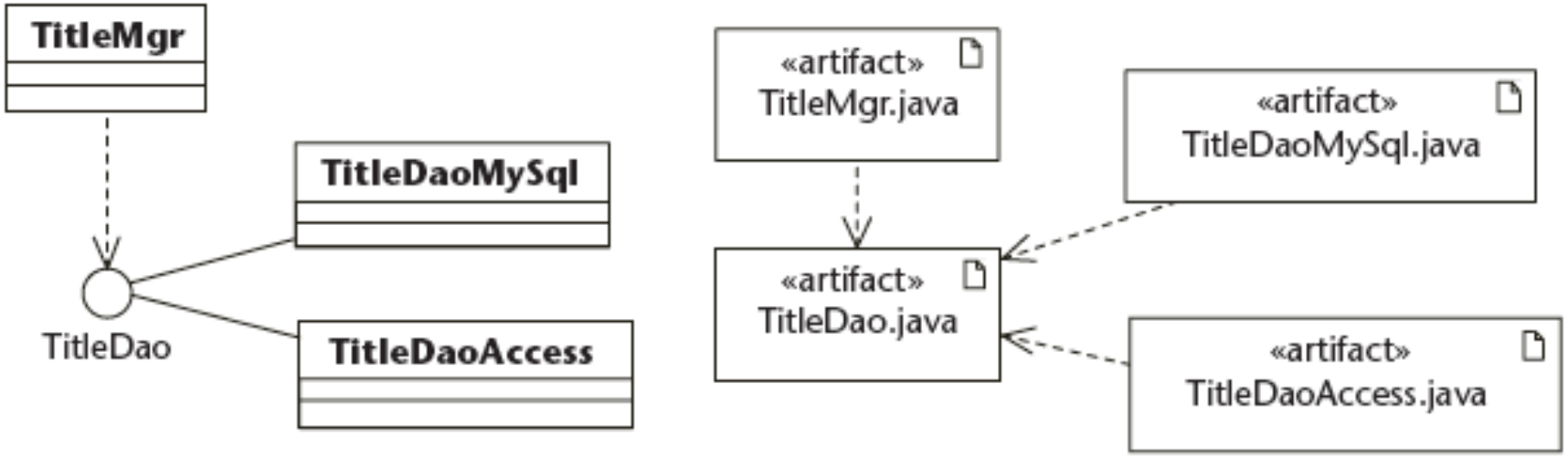
- Le attività di implementazione costituiscono la programmazione effettiva delle classi
- Java consente una progressione naturale dalle classi logiche che sono stati creati nel progetto per i componenti di codice in fase di attuazione

  - perché c'è una corrispondenza uno-a-uno di una classe in un file di codice Java (e una mappatura uno-a-uno ad un eseguibile Java

    .file di classe)
  - Java specifica inoltre che il nome del file sia la stessa di quella della classe Contiene

Implementazione

- Un diagramma componente aiuta a visualizzare la rappresentazione fisica come mappata dal modello disegno
- Un diagramma di classe e il corrispondente diagramma componente, che mostra i manufatti che manifestano queste classi (dipendencies tra un numero di classi titolo)



Implementazione

- Per la codifica, le specifiche sono raccolti dai seguenti diagrammi nel modello di progettazione:
  - **diagrammi delle classi.** I diagrammi delle classi in cui la classe è presente, mostrando la sua struttura statica e relazione con altre classi.
  - **diagramma di macchina a stati.** Un diagramma di macchina a stati per la classe, mostrando i possibili stati e le transizioni che devono essere gestiti (insieme con le operazioni che attivano le transizioni).
  - **diagrammi dinamici (sequenza, comunicazione e attività) in cui sono coinvolte oggetti della classe.** Diagrammi che mostrano la realizzazione di un metodo specifico nella classe o come altri oggetti utilizzano oggetti della classe.

  - **Utilizzare case-diagrammi e specifiche.** Diagrammi che mostrano il risultato del sistema di dare agli sviluppatori ulteriori informazioni su come il sistema deve essere utilizzato quando lui o lei potrebbe essere perdersi in Dettagli- perdere di vista il contesto generale.

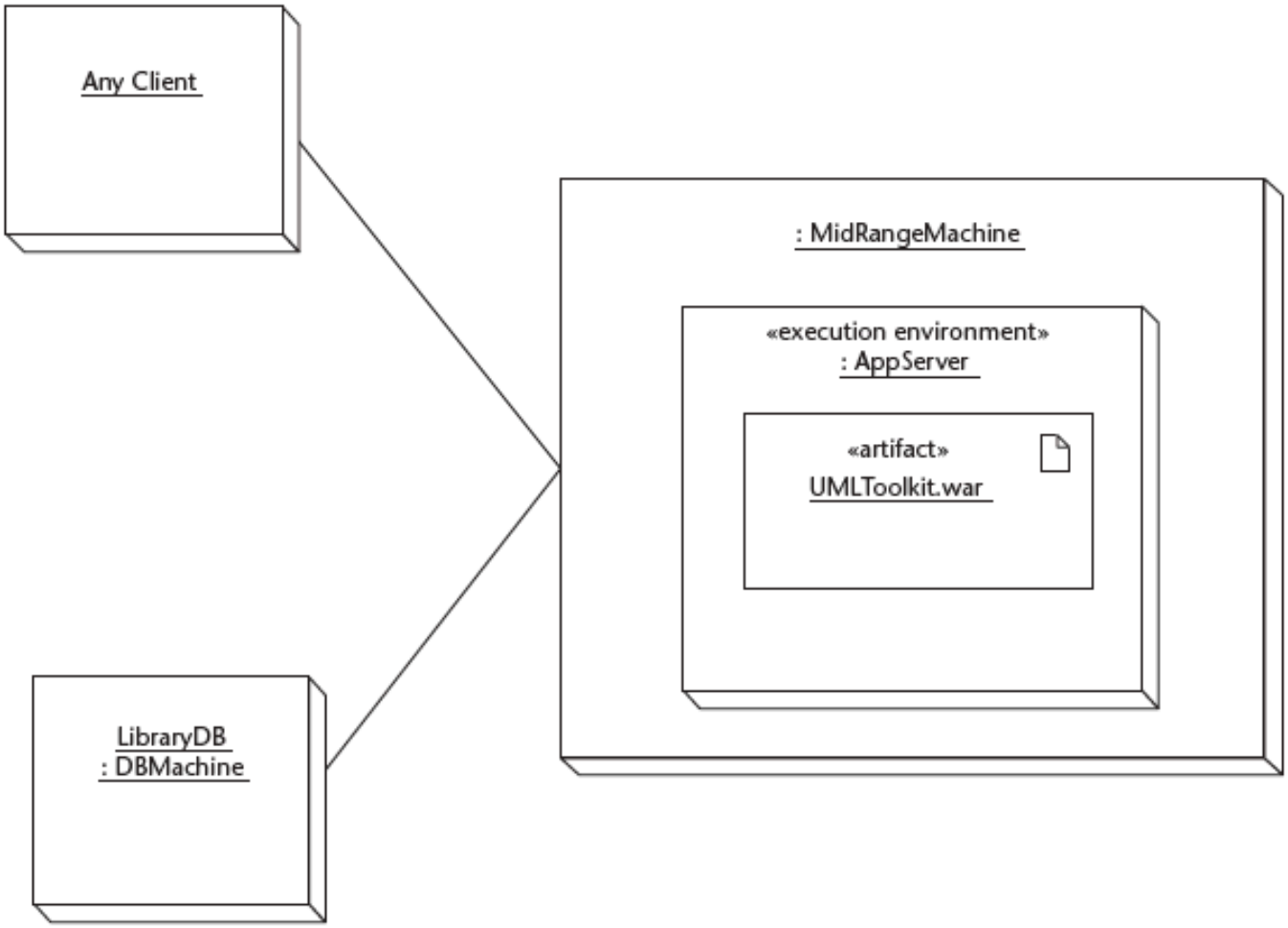
Test e distribuzione

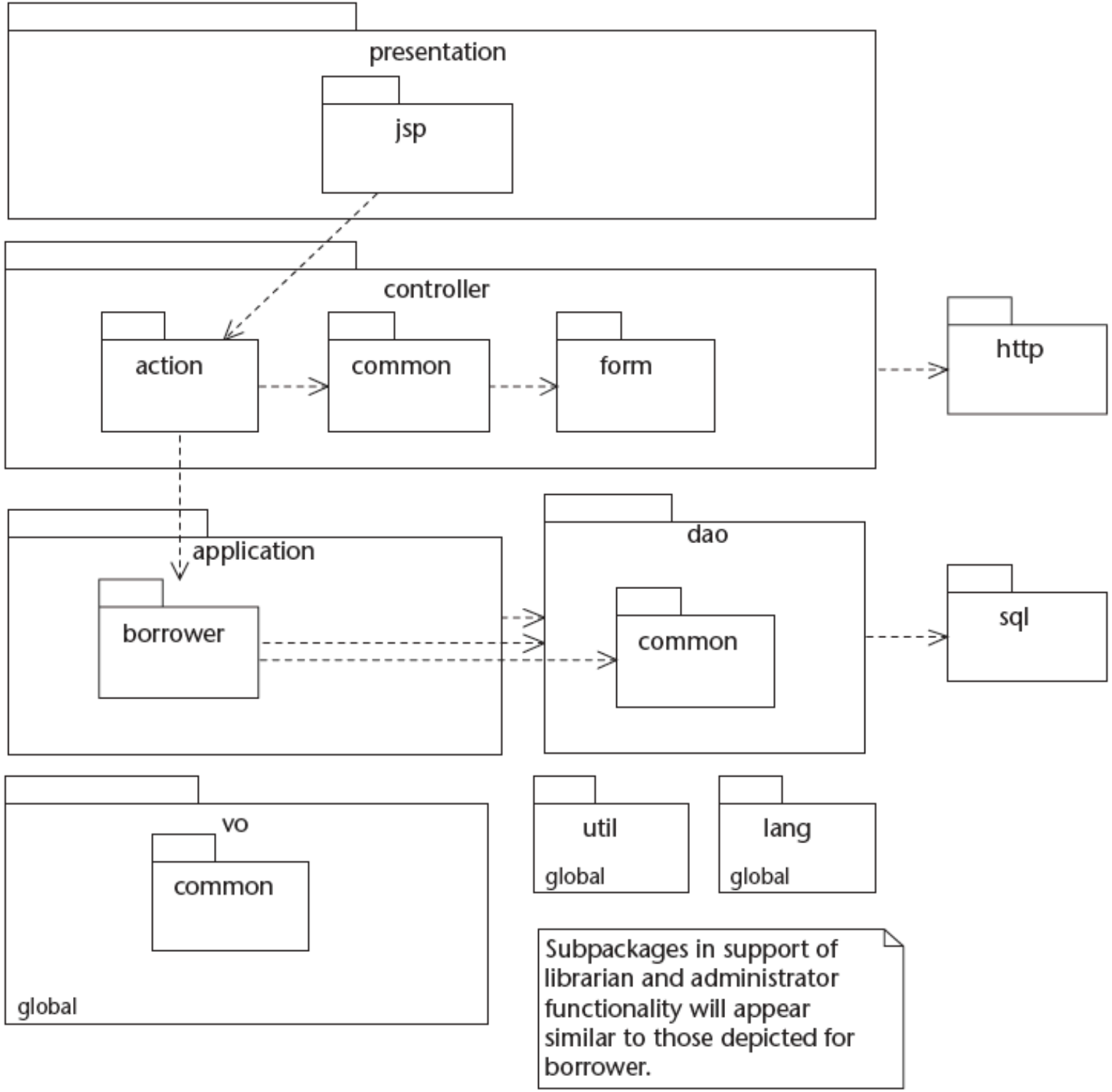
- Un'applicazione più vasta scala richiede specifiche formali più contro il quale il software è testato e un difetto del sistema di monitoraggio

- L'implementazione del sistema è la consegna effettiva, compresa la documentazione.
- In un progetto di vita reale, manuali d'uso e le descrizioni di marketing sono in genere parte del lavoro di documentazione

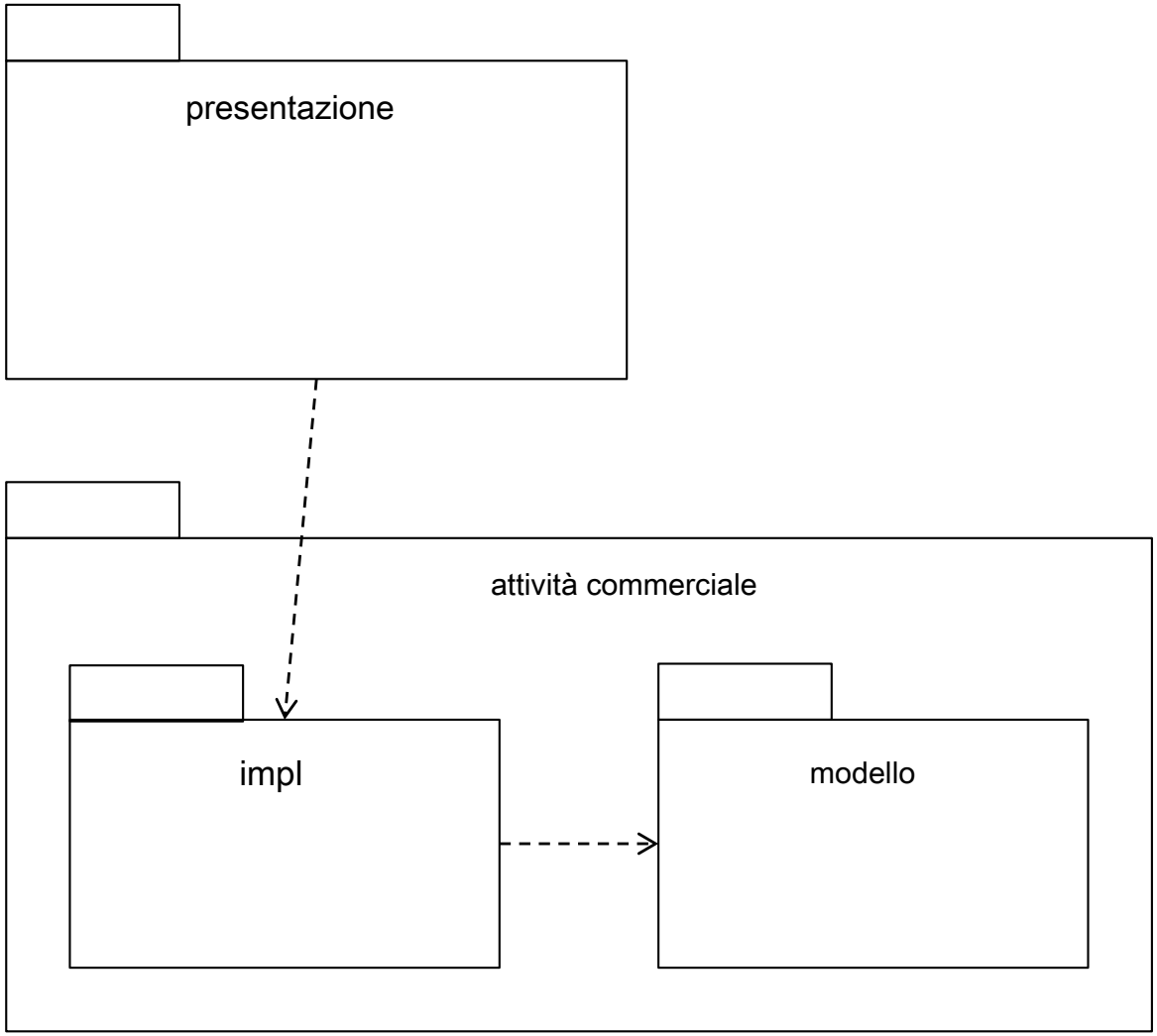
Test e distribuzione

- Un diagramma di distribuzione (per il sistema libreria) dovrebbe anche essere disegnato dell'architettura fisica




presentation
jsp
controller
action
common
form
http
application
borrower
dao
common
sql
vo
common
global
util
global
lang
global
Subpackages in support of librarian and administrator functionality will appear similar to those depicted for borrower.

Biblioteca



biblioteca Web