Laboratorio di Sviluppo delle Applicazioni Software

Progettazione

T3 - Prof. Mirko Polato

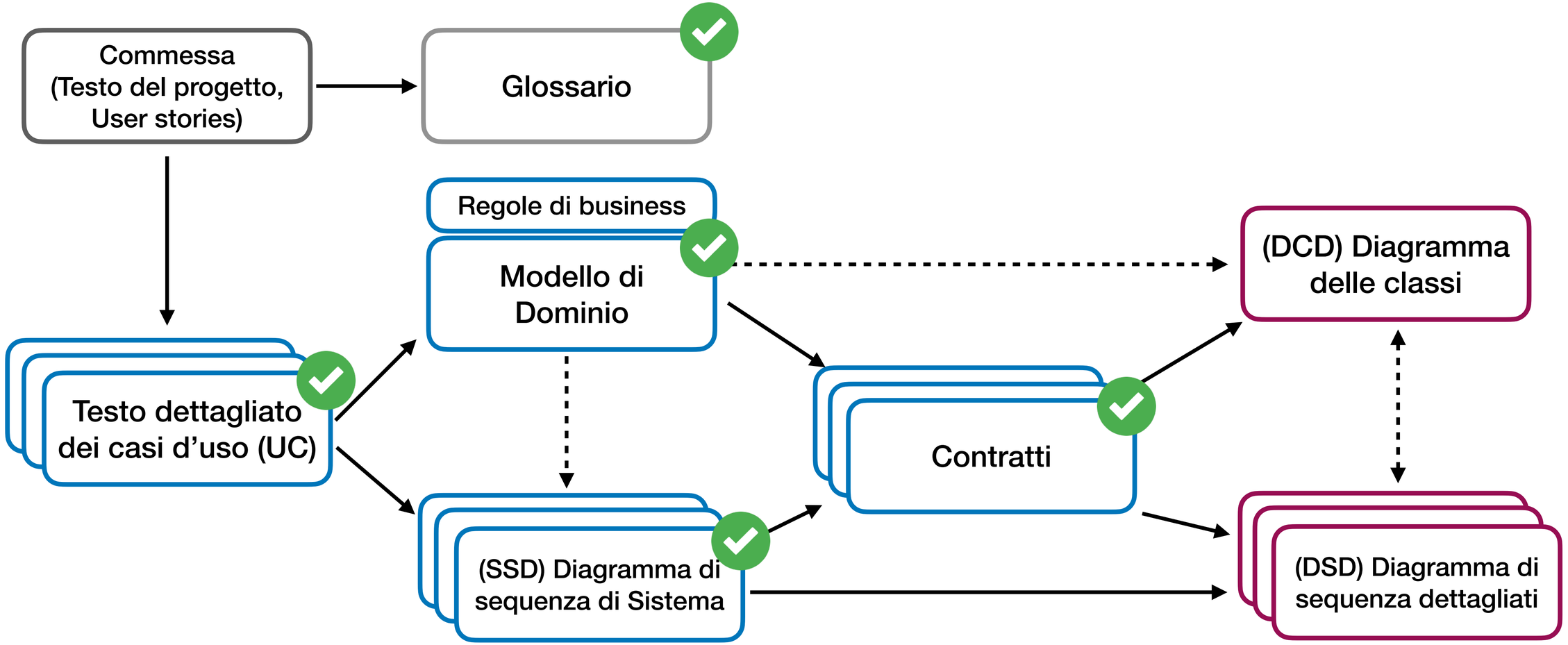
29/04/2024

Punto della situazione

Da dove partiamo

Documentazione (codice escluso)

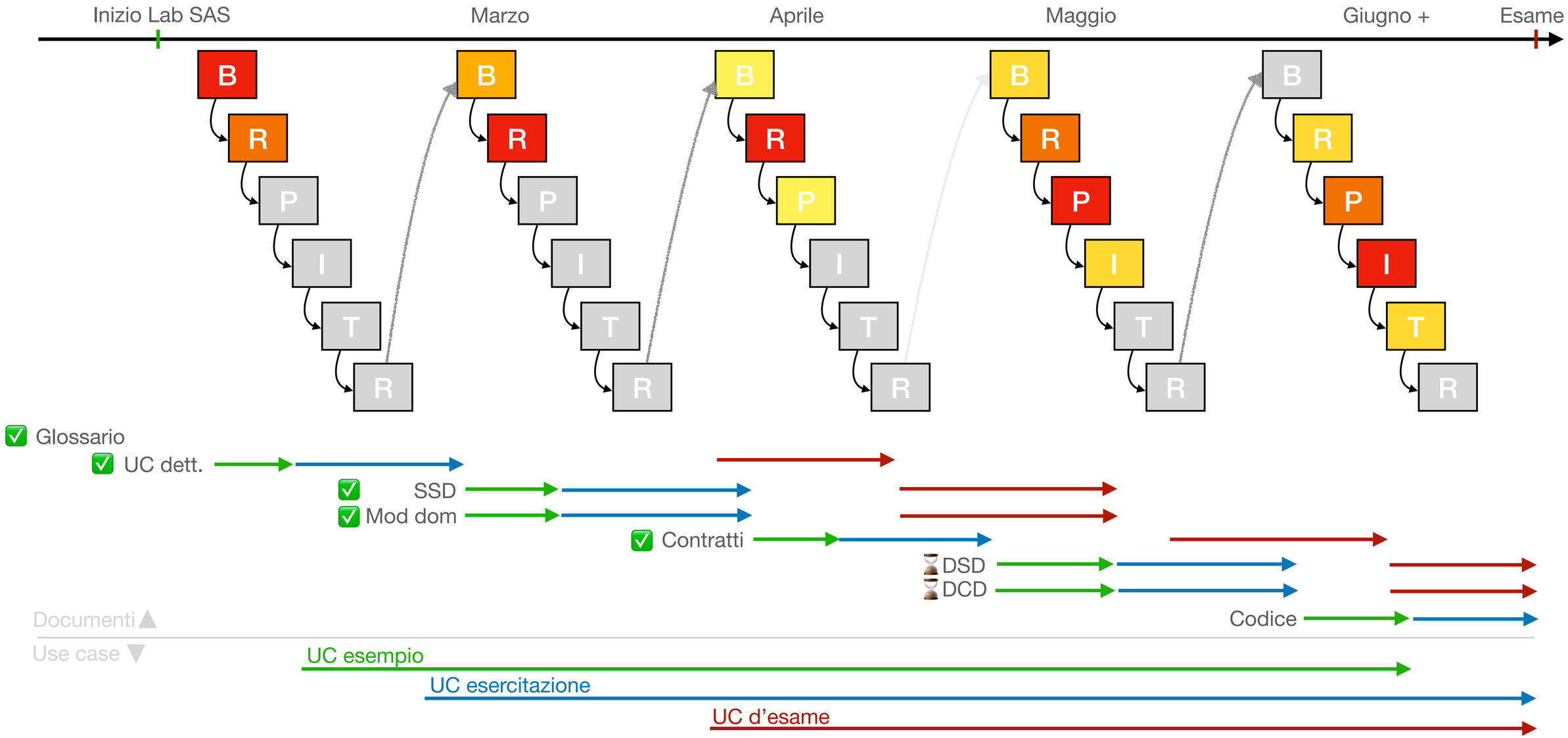
Artefatti della metodologia UP



Requisiti

Progettazione

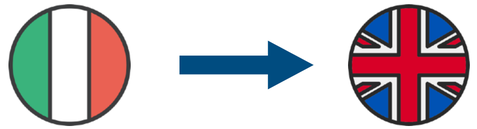
Evoluzione degli artefatti di UP

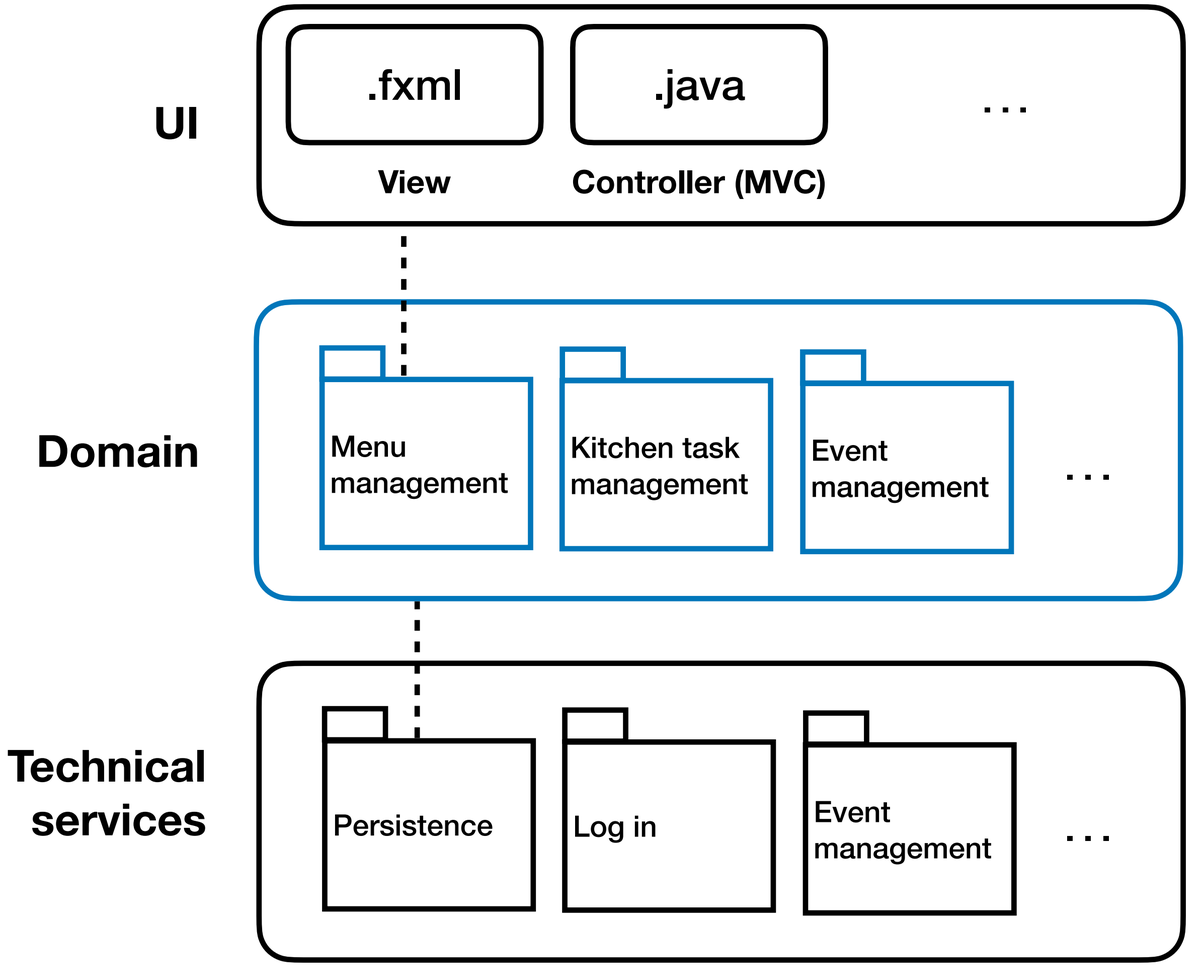


Progettazione

Parte I

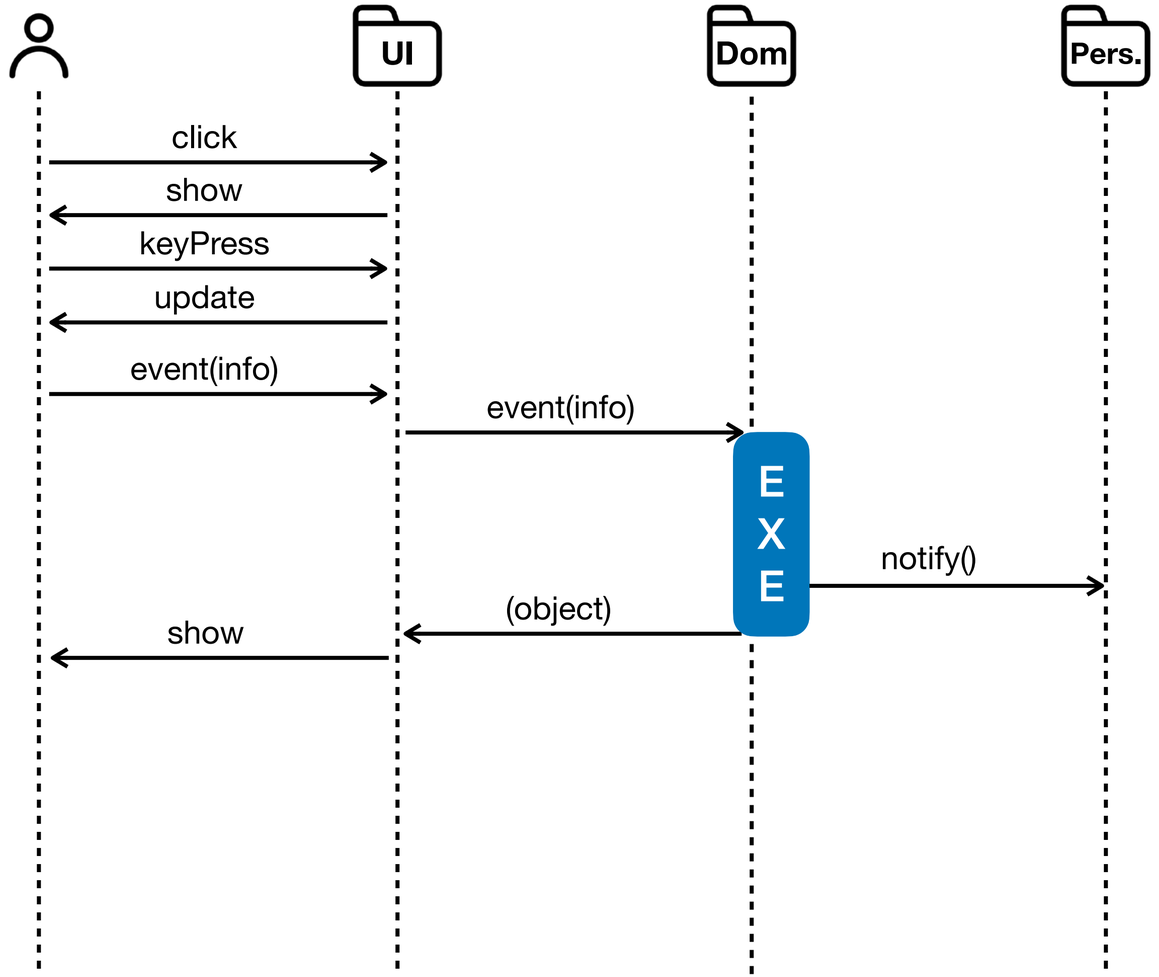
Cat&Ring - Architettura statica





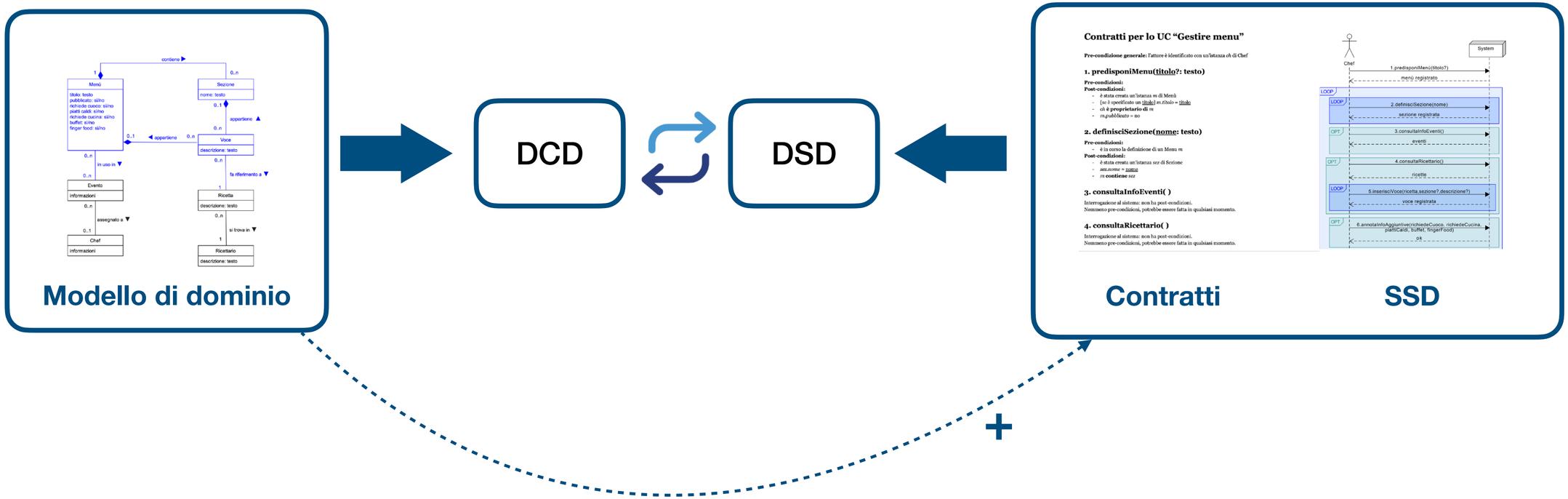
Cat&Ring - Architettura dinamica

- La **UI non è responsabile della logica delle operazioni**, ma ha “conoscenza” del dominio
- Lo strato di dominio è responsabile della logica
- La **UI conosce il GRASP controller** che funge da “interfaccia” tra UI e Dom
- Esiste una dipendenza forte (inevitabile) tra UI e dominio
- Mentre c’è una dipendenza debole (o inesistente) tra dominio e strato tecnologico



DCD e DSD

Come procedere…



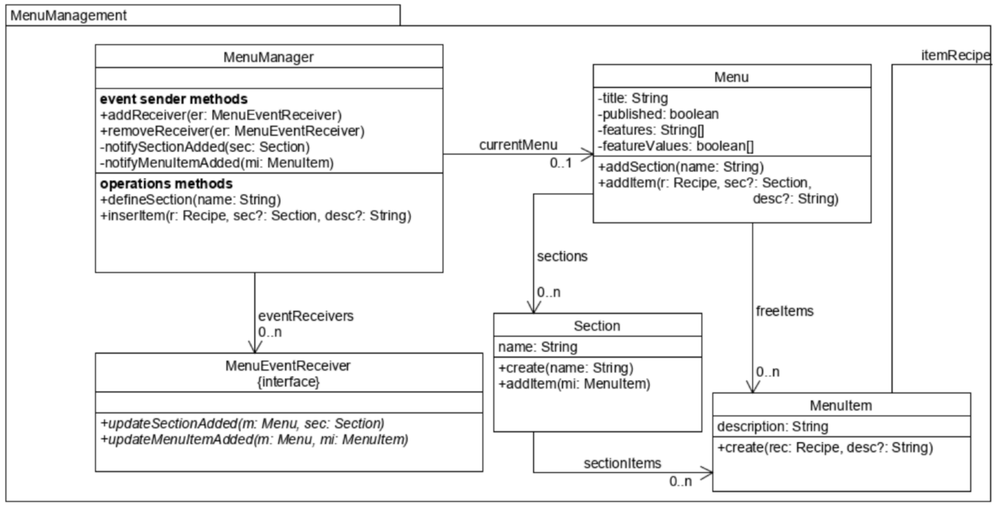
Design Class Diagram (DCD)

Diagramma delle classi

Diagramma delle classi (DCD)

Visione statica della relazione tra le classi

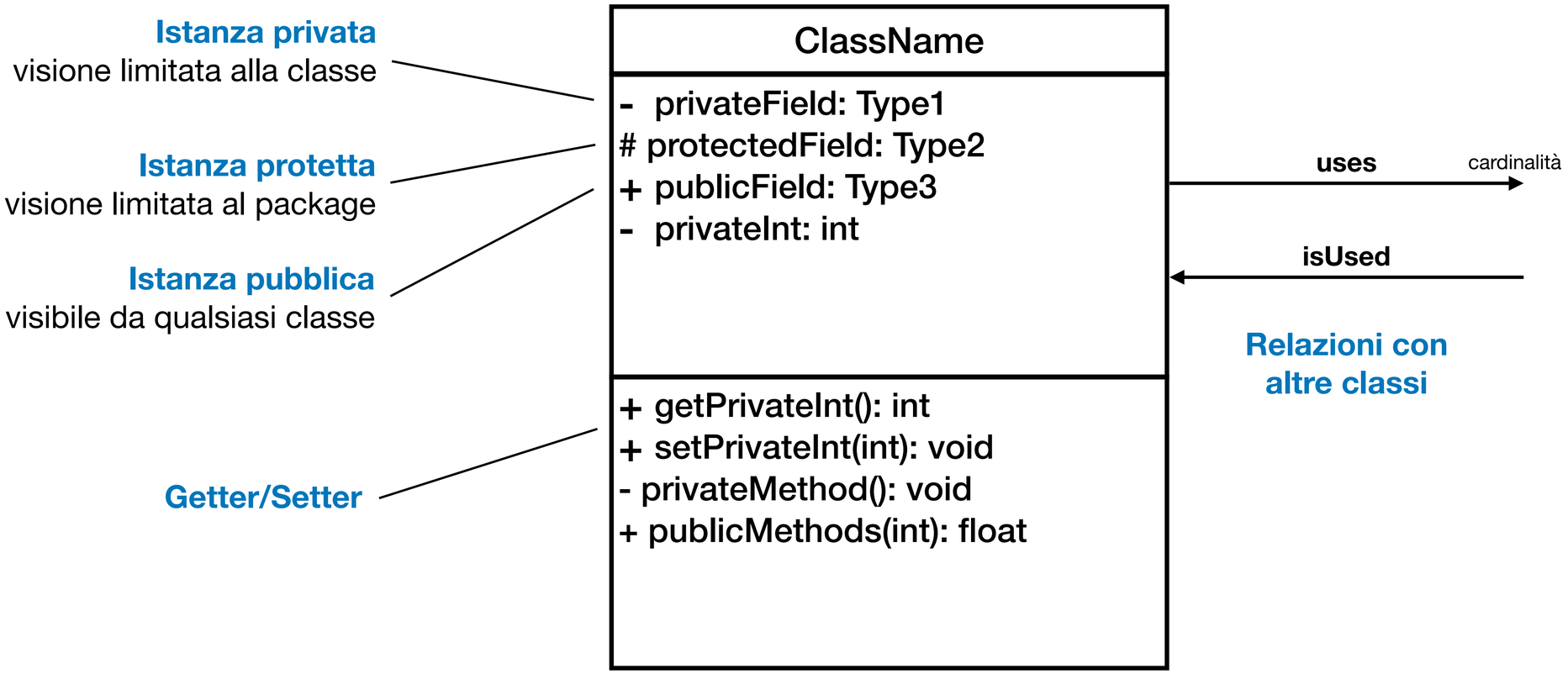
- La creazione dei DCD è parallela a quella dei DSD
- Il DCD **non** è una copia del modello di dominio
- Il **modello di dominio** funge da **ispirazione** per il DCD
- Utilizzare moduli/package per “isolare” i diversi UC
- “Sfruttare” i pattern GoF, quando possibile



Il Diagramma delle classi è unico per l’intero progetto!!

Diagramma delle classi (DCD)

Classe in UML



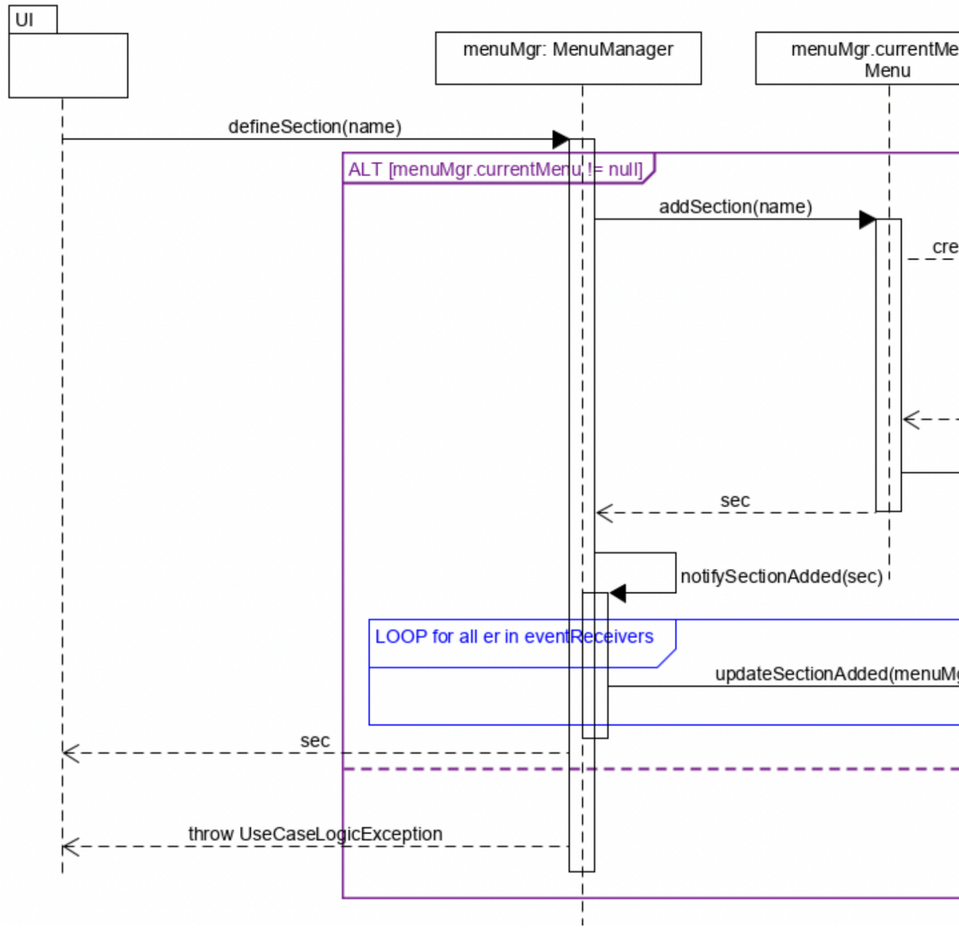
Detailed Sequence Diagrams

Diagrammi di sequenza dettagliati

Diagrammi di sequenza dettagliati (DSD)

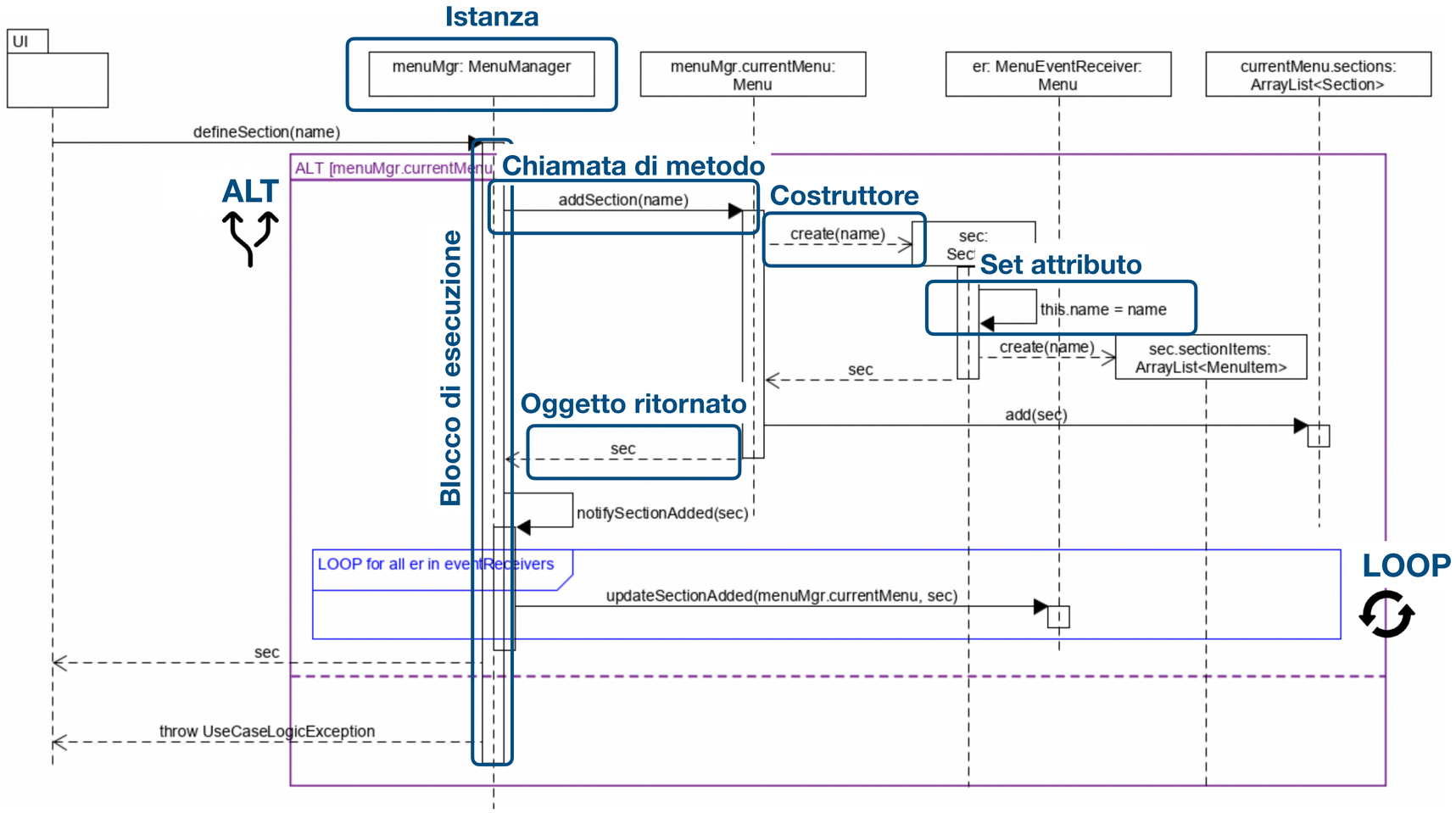
Visione dinamica della relazione tra le classi

- Modella le interazioni tra gli oggetti in un UC
- Esiste un DSD per ogni operazione definita nei contratti
- Nella pratica, si definiscono DSD solo di operazioni critiche e importanti
- **Buona idea**: partire da DSD di operazioni “impattanti”, ovvero che vi costringono a fare delle scelte
- La prima chiamata di metodo avviene sempre tra UI e Dominio tramite il **controller GRASP**
- **Sfruttate i pattern GRASP (information expert, creator)**



Detailed Sequence Diagram (DSD)

DSD Cheatsheet

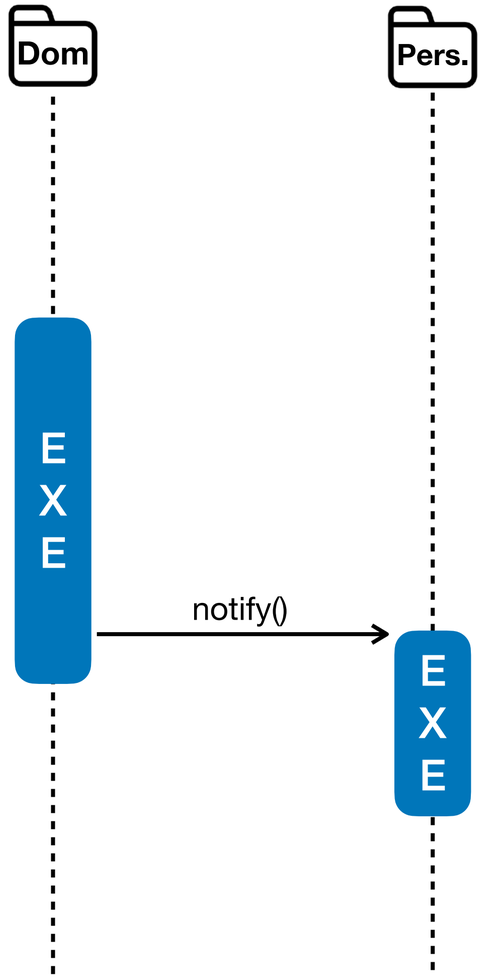


DCD - Parte II

Pattern EventReceiver, post-condizioni, ereditarietà…

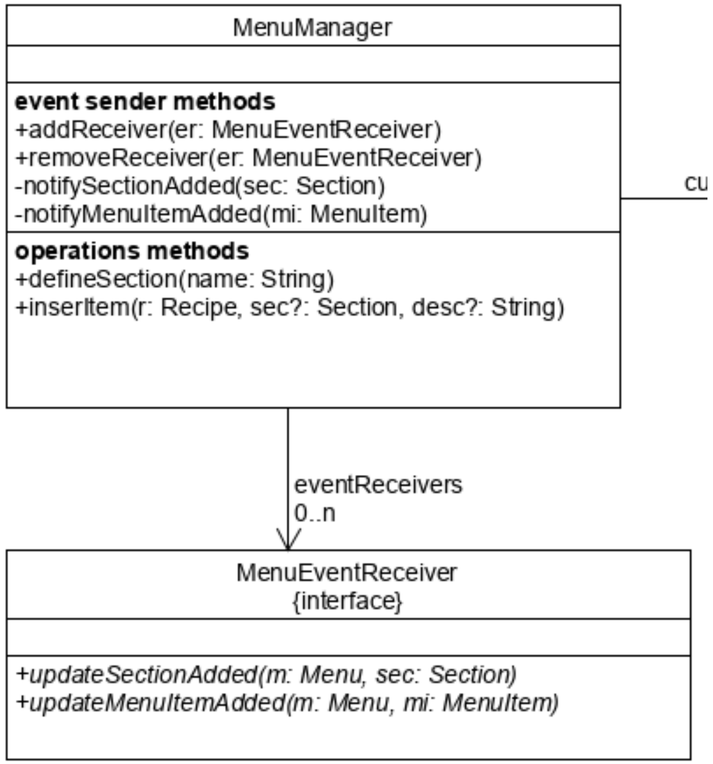
Pattern Event Receiver (Observer)

- La persistenza sarà gestita utilizzando il pattern Event Receiver (**Observer**)
- Lo strato di persistenza (una sua classe, *observer*) viene **notificato** dallo strato di dominio (*subject*) che il suo stato è cambiato
- Lo strato di dominio deve solamente conoscere chi sono gli osservatori e non la loro logica interna
- Lo strato di dominio e di technical service sono debolmente accoppiati



Pattern Event Receiver (Observer)

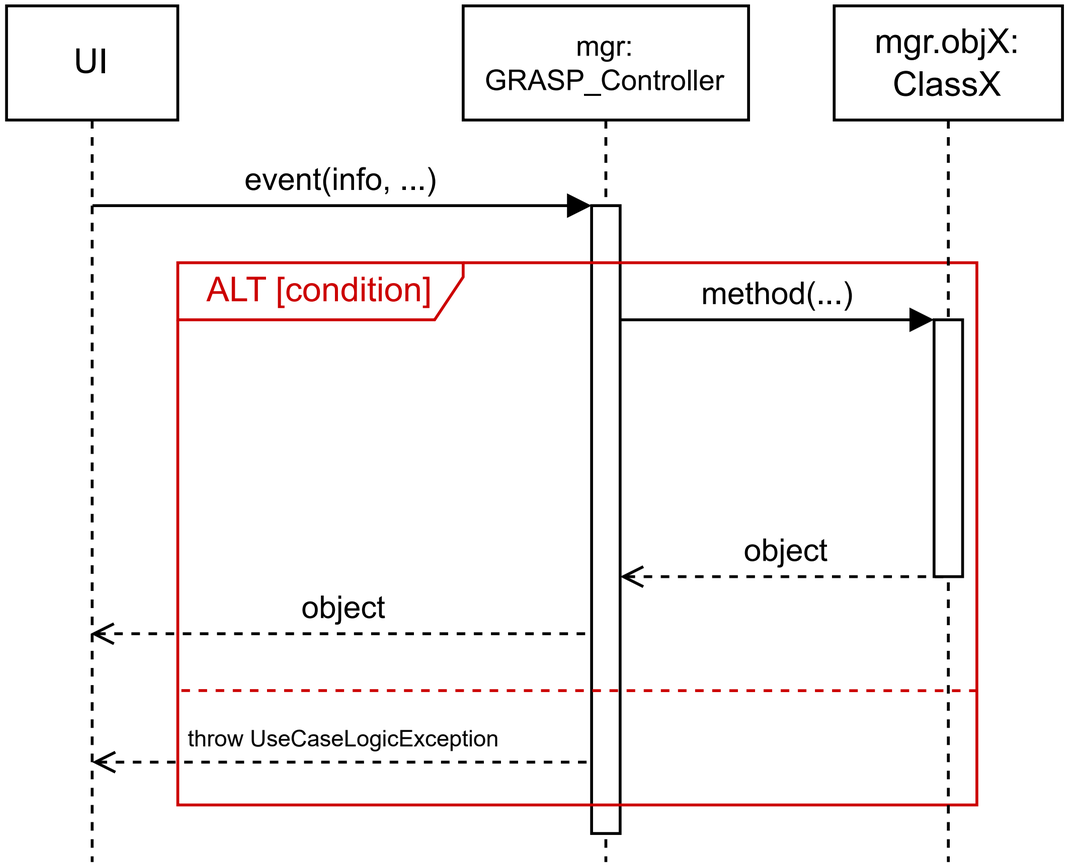
- L’oggetto osservato (**Subject**) dovrà avere la lista dei riferimenti degli osservatori (**Observers/ EventReceiver**)
- Gli EventReceivers implementano l’interfaccia corrispondente
- Il Subject notifica gli osservatori (metodi **notify**) richiamando i corrispondenti **metodi update**



DSD - Precondizioni

Come gestirle

- Le **precondizioni**, spesso, si traducono in un **blocco ALT**
- Se la precondizione rappresenta un **vincolo logico** legato al caso d'uso allora si lancia un'eccezione del tipo **UseCaseLogicException**
- Tale eccezione indica che non è rispettata una **condizione necessaria perché tale operazione possa essere effettuata** (e.g., attore non autenticato come Chef)
- Se invece, la precondizione riguarda la **business logic** (e.g., il Menù non può essere cancellato perché in uso), allora il tipo dell'eccezione sarà diverso (e.g., MenuException)



DCD - Parametri opzionali

Come gestirli

- **I parametri opzionali** si indicano con il **?**
- **Java non permette(*) di avere parametri opzionali**
- I parametri opzionali si traducono in **overload dei metodi**, uno per ogni combinazioni di parametri

```
method(param1?, param2?, param3?)
```

(**)

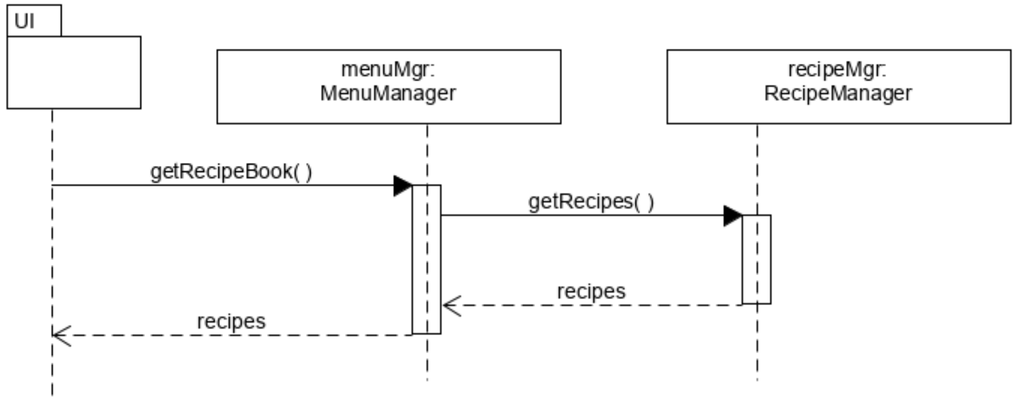
```
method(param1)

method(param2)

method(param3)

method(param2, param3)

method(param1, param2)

method(param1, param3)

method(param1, param2, param3)
```

(*) Esistono in realtà strategie per gestirli come, ad esempio, la classe contenitore Optional e il Builder Pattern.
(**) Non è sempre possibile creare tutti gli overload nel caso i tipi siano gli stessi

DSD & DCD - Ordinamento e consultazioni

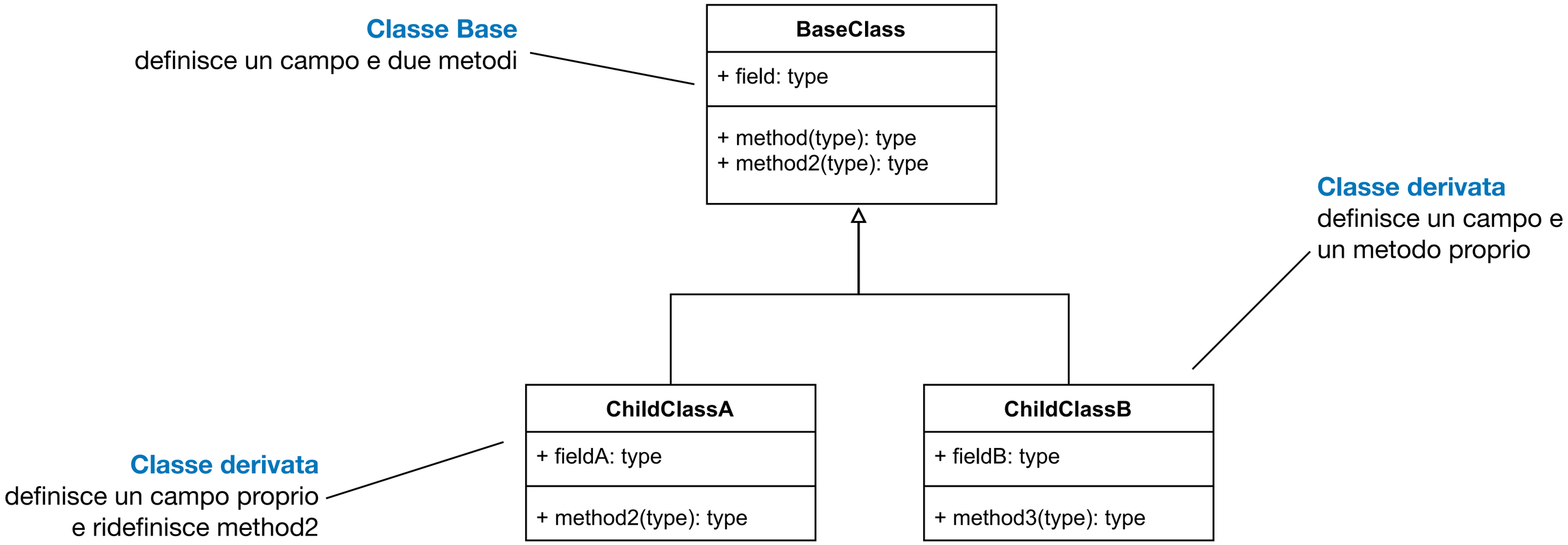
Come gestirli

- **L'ordinamento** di oggetti all'interno di liste va gestito più **a basso livello** rispetto a UC dettagliato/Contratti
- Si deve **immaginare il tipo di interazione** dell'attore con il sistema per effettuare tale ordinamento, e.g., drag&drop dell'oggetto all'interno della lista
- L'interazione **definisce quali informazioni deve offrire l'attore al sistema**, e.g., oggetto + posizione finale
- Le operazioni di **consultazione** di solito non hanno **ne pre ne post** condizioni e quindi si traducono in semplici DSD



DCD - Ereditarietà

Come implementare una gerarchia di classi



Progettazione

DCD e DSD