Progetto

Struttura del documento di specifica dei requisiti, Casi d’uso

manuel.comparetti@iet.unipi.it

Documenti da produrre

- Il progetto deve comprendere i seguenti documenti:
  - Documento di specifica dei requisiti (DSR)
    - È il prodotto della fase di analisi e specifica dei requisiti
  - Documento di specifica del progetto (DSP)
    - Specifica in termini di moduli software come deve essere implementato il sistema descritto dal DSR

Organizzazione di massima

- Documento di specifica dei requisiti:
  - Introduzione (al problema e al documento!)
  - Analisi del dominio/caratterizzazione di massima del sistema
  - Identificazione degli utenti (casi d'uso)
  - Specifica requisiti funzionali e non funzionali (...)
  - Interfaccia (verso l'utente, verso altri sistemi)
  - Scenari di utilizzo
- Documento di specifica del progetto:
  - Analisi architetturale
  - Analisi in dettaglio
- Letti entrambi i documenti un team di sviluppatori dovrebbe essere in grado di procedere alla fase di codifica (implementazione) dei moduli e produrre un sistema conforme con il DSR

Documento di specifica dei requisiti

- Deve essere:
  - Corretto (...)
  - Completo
  - Consistente
  - Organizzato in maniera logica
    - Ben strutturato
    - Ordinato (i requisiti devono avere un ordine)
    - Navigabile (tracciabile)
    - Facilmente modificabile (...)
  - Verificabile
- Deve comprendere
  - Un indice e un glossario
- Può utilizzare
  - UML e linguaggio naturale
    - Il linguaggio usato deve comunque essere univoco
    - Uso di termini chiave
  - Un linguaggio formale evita l'ambiguità ma può non essere compreso da chi legge il DSR ...

Documento di specifica dei requisiti

- Si riferisce allo spazio del problema (il DSP si riferisce allo spazio delle soluzioni)
- Definisce la vista (o le viste) del sistema che deve avere l'utente
- Quindi è sbagliato in questa fase introdurre aspetti implementativi
- Si pensi sempre che contestualmente al DSR può essere prodotto il *manuale utente*
- *Letto il documento di specifica dei requisiti un utente dovrebbe essere in grado di utilizzare il sistema*
- *Letto il documento di specifica dei requisiti un team di progetto dovrebbe avere tutte le informazioni per procedere alla stesura del DSP*

Documento di specifica dei requisiti

- 1. Introduzione: di cosa stiamo parlando? A che (e a chi) serve?
- 2. Può essere utile un'analisi del dominio
  - Raccoglie le informazioni relative a descrivere il dominio del problema
  - Dipende dall'applicazione
  - Serve a farsi un'idea delle entità e dei concetti con cui hanno a che fare il sistema e l'utente
  - Può prevedere uno schema (di massima e ad alto livello) del sistema o uno schema a blocchi (o delle classi) in cui si esemplificano le varie entità in gioco
- 3. Caratterizzazione di massima del sistema (tipo di elaborazione, ruolo dell'applicazione)

Documento di specifica dei requisiti

- Identificazione degli utenti (delle classi di utenza)
- Ogni classe di utenza ha uno scopo per cui utilizza il sistema
- Identificare le diverse classi di utenza permette di farsi un'idea delle differenti modalità di interazione con il sistema
- Come interagisce un utente con un sistema? Descrizione dinamica/particolare -> scenario

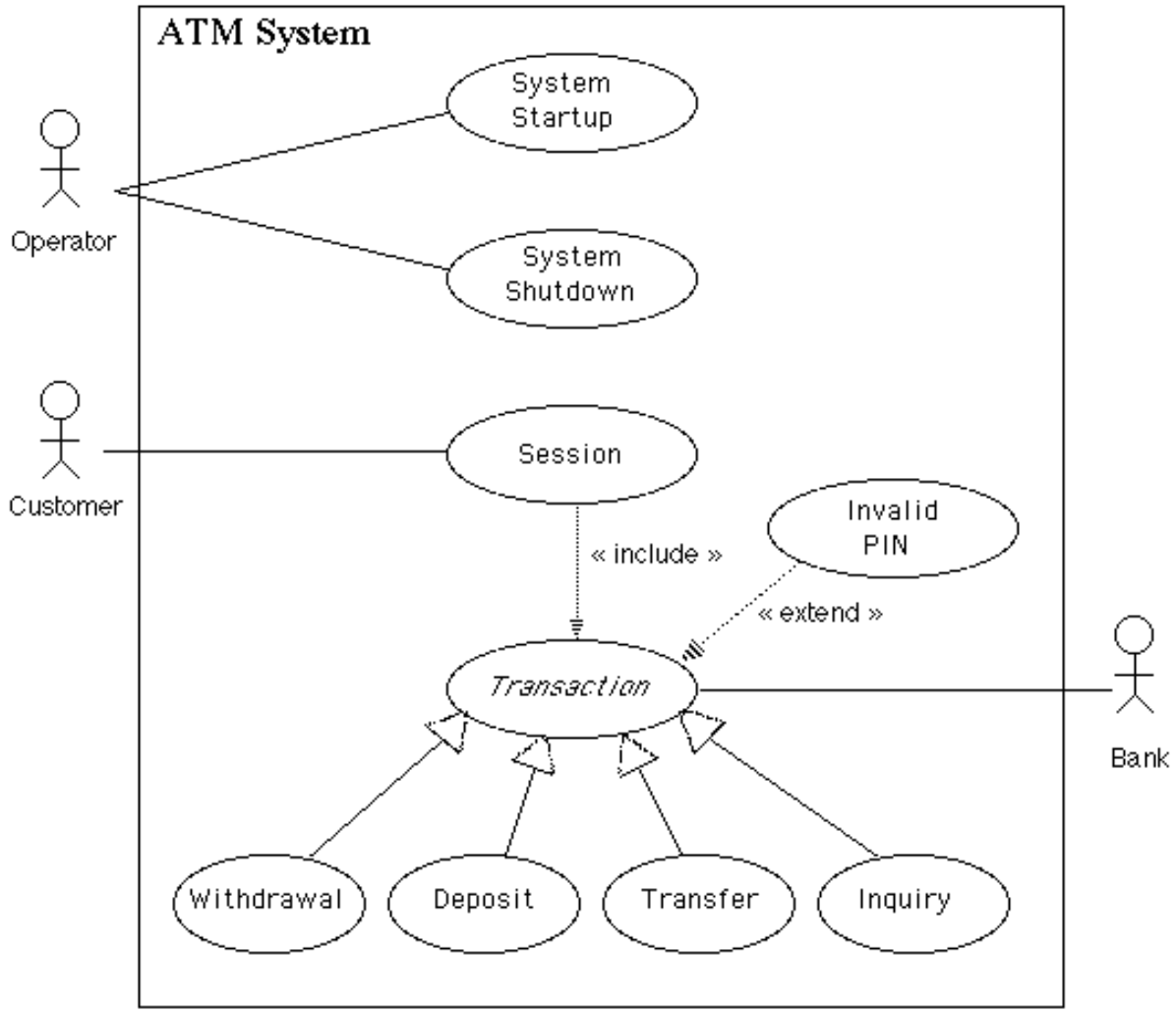
Astrazione: casi d’uso

- Definizione: uno **use case** è un insieme di scenari collegati da uno scopo comune dell’utente
- Gli use case sono utili per definire le classi di utenza
- Gli use case si usano anche per organizzare i requisiti dell’utente contestualizzandoli, in base alle tipologie di utilizzo (uno use case si può riferire a più requisiti o viceversa)
- L’attore principale di ogni caso d’uso è colui che richiede il servizio al sistema e, spesso, colui che inizia lo scenario con la prima interazione
- Per gli use case esiste una notazione UML

Use case e UML

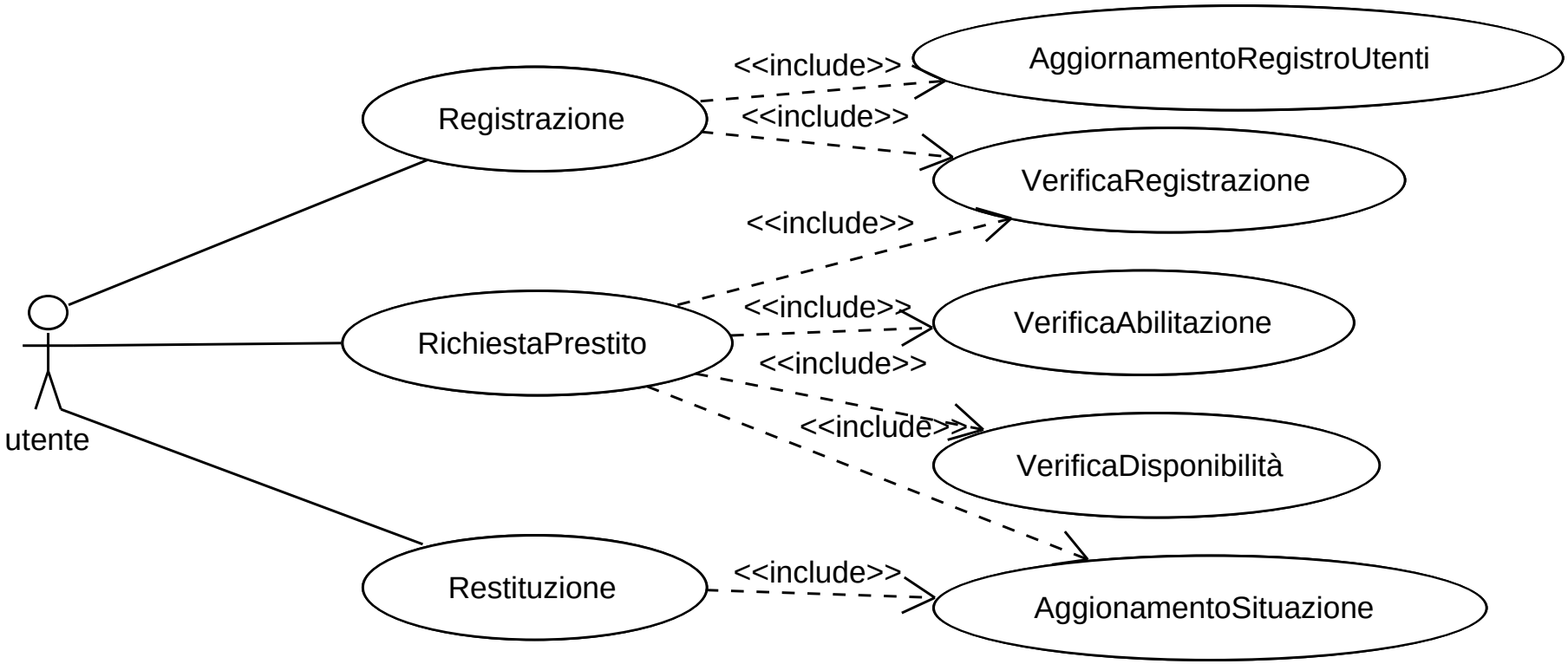
- UML definisce una notazione per descrivere gli use case, come sono collegati alle tipologie di utente e come sono organizzati tra di loro
- Per ogni use case, però, è necessaria una descrizione (testuale) a corredo, su cui non è definito uno standard preciso

Esempio di grafico UML



Tratto dal materiale online per il corso “CPS211: Object-Oriented Software Development” 2009, Russell C. Bjork, Gordon College”
http://www.math.gordon.edu/courses/cs211/

Un sistema bibliotecario



"Baresi, Lavazza, Pianciamore Dall'idea al codice con UML 2*"*

Una possibile descrizione testuale di uno use case

- **Nome**
- **Tipo** *primario/secondario*
- **Precondizione** (e *trigger): cosa deve assicurare il sistema perché lo use case possa avere luogo*
- **Svolgimento normale:** *organizzato per fasi*
- **Svolgimento(i) alternativo(i):** *casi particolari, eccezioni, casi di errore*
- **Post condizione** (svolgimento normale): *descrive il nuovo stato del sistema*
- **Descrizione** *(ad es. su cosa agisce lo use case, quali sono i rapporti con gli altri use case)*

Esempio: registrazione

- **Nome**: registrazione.
- **Tipo**: primario.
- **Precondizione**: l'utente richiede di essere registrato.
- **Svolgimento normale**: l'utente non risulta essere registrato, quindi viene registrato. Il sistema comunica all'operatore il numero identificatore assegnato al nuovo utente della biblioteca.
- **Svolgimento(i) alternativo(i)**: l'utente risulta già essere registrato, quindi il sistema non effettua alcuna modifica, ma si limita a comunicare all'operatore il numero identificatore dell'utente presso la biblioteca.
- **Post condizione**: l'utente è registrato, cioè compare nel registro utenti con un identificatore univoco.
- **Descrizione**: per prima cosa si verifica se l'utente è già registrato (caso d'uso **verifica registrazione**). Se l'esito è positivo il registro utenti non viene modificato, altrimenti viene aggiornato con l'aggiunta di una registrazione corrispondente al nuovo utente (caso d'uso **AggiornamentoRegistroUtenti**). In ogni caso il sistema comunica all'operatore il numero identificatore dell'utente presso la biblioteca.

Questo e il successivo esempio di casi d'uso seguenti sono stati tratti da.

"Baresi, Lavazza, Pianciamore dall'idea al codice con UML 2."

- *Nota: gli use case possono avere livellli diversi.*

Esempio: VerificaRegistrazione

- **Nome**: VerificaRegistrazione.
- **Tipo**: secondario *(non è sollecitato direttamente dall'utente).*
- **Precondizione**: qualche funzione sollecitata dall'utente (richiesta registrazione, richiesta prestito) ha riscontrato la necessità di effettuare un controllo della registrazione *(trigger).*
- **Svolgimento normale**: l'utente è registrato, quindi la verifica ha esito positivo. In uscita viene comunicato il numero identificatore dell'utente.
- **Svolgimento alternativo**: l'utente non è registrato, quindi la verifica ha esito negativo. In uscita viene comunicato l'esito negativo della verifica.
- **Post condizione** (svolgimento normale): la registrazione è stata verificata e l'esito comunicato. Il registro utenti non ha subito variazioni.
- **Descrizione**: questa operazione è funzionale* rispetto a varie operazioni. Viene eseguita quando si verificano le condizioni per effettuare un controllo della registrazione. La verifica si basa sul contenuto del registro utenti: se l'utente è presente la verifica ha esito positivo, altrimenti ha esito negativo.

* Non si tratta di funzioni interne al sistema! Se lo use case viene specificato significa comunque che c'è interazione con l'utente (...).

Estensioni di uno use case

- **Acquisto azioni sul web**
- **Attore primario**: `cliente`
- **Precondizione**: `il cliente dispone di un pacchetto azionario`
- **Scenario principale**:
  - `1. L'utente seleziona l'acquisto di azioni sul web`
  - `2. Il sistema acquisisce dall'utente il nome del sito web da utilizzare (E*Trade, Schwabb etc.)`
  - `3. Il sistema apre la connessione web al sito`
  - `4. L'utente naviga il sito e acquista le azioni`
  - `5. Il sistema intercetta la risposta del sito web`
  - `6. Il sistema mostra la situazione aggiornata del pacchetto azionario dell'utente`
- **Estensioni**:
  - `2a. L'utente richiede un sito web non supportato dal sistema`
  - `2a1. Il sistema richiede un nuovo sito all'utente, che eventualmente può annullare l'operazione`
  - `3a. Fallimento della connessione al sito`
  - `3a1. Il sistema riporta il fallimento riportandosi alla schermata precedente`
  - `3a2. L'utente ritenta la connessione o eventualmennte annulla l'operazione`
  - `4a. Crash del terminale dell'utente`
    `... Cosa si può fare qui? ...[TBD]`

il DSR viene prodotto in maniera incrementale! Ma la versione finale del documento deve essere completa

Tratto e adattato da “Cockburn, *Writing effective use cases*”

Relazioni tra use case

- Inclusione (**<<include>>** , **<<uses>>** in standard UML precedenti) si utilizza per descrivere meglio use case complicati, identificando interazioni di base con il sistema, o specificare una sola volta operazioni comuni a più use case
- Estensione (**<<extend>>**) cattura scenari alternativi a partire da uno o più extension point. Modella eccezioni, casi di errore, condizioni inattese, help utente e quant'altro. In generale dovrebbe implicare un esito differente dello use case

Relazioni tra use case

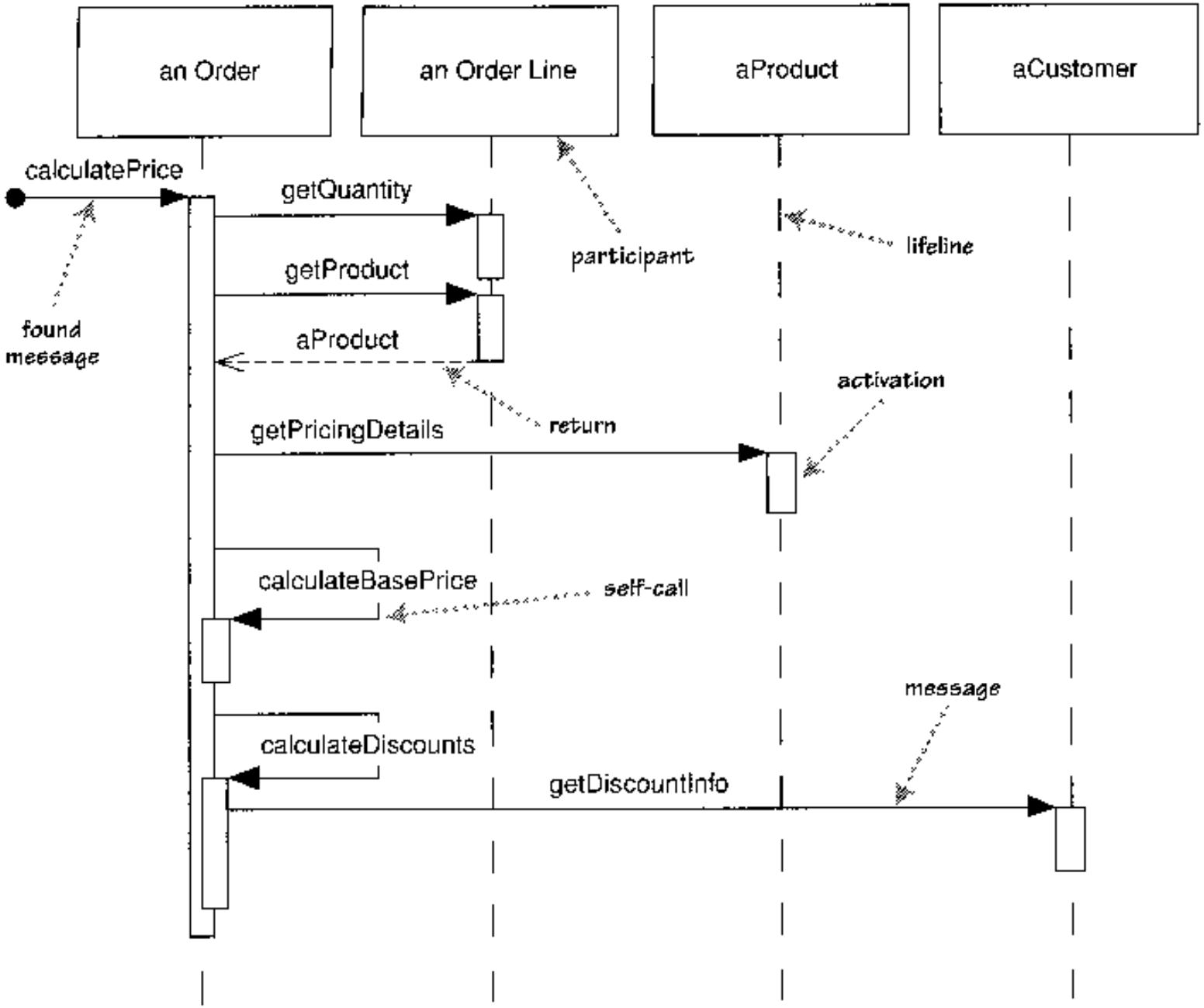
- Generalizzazione: (stessa notazione dei class diagram) uno use case specifico è simile a uno generale, ma comporta un comportamento aggiuntivo. Significato e scopo dello use case arricchito rimangono gli stessi.
- Invece l'estensione aggiunge un comportamento opzionale in una fase precisa dello use case base, e solo se sono soddisfatte determinate condizioni.
  - Il risultato finale dello use case esteso può essere diverso (spesso un annullamento dell'operazione in caso di errore)
- Spesso le differenze possono essere molto sfumate, sta a chi scrive il documento di specifica organizzare gli use case nella maniera più chiara ed efficace possibile.

Scenari

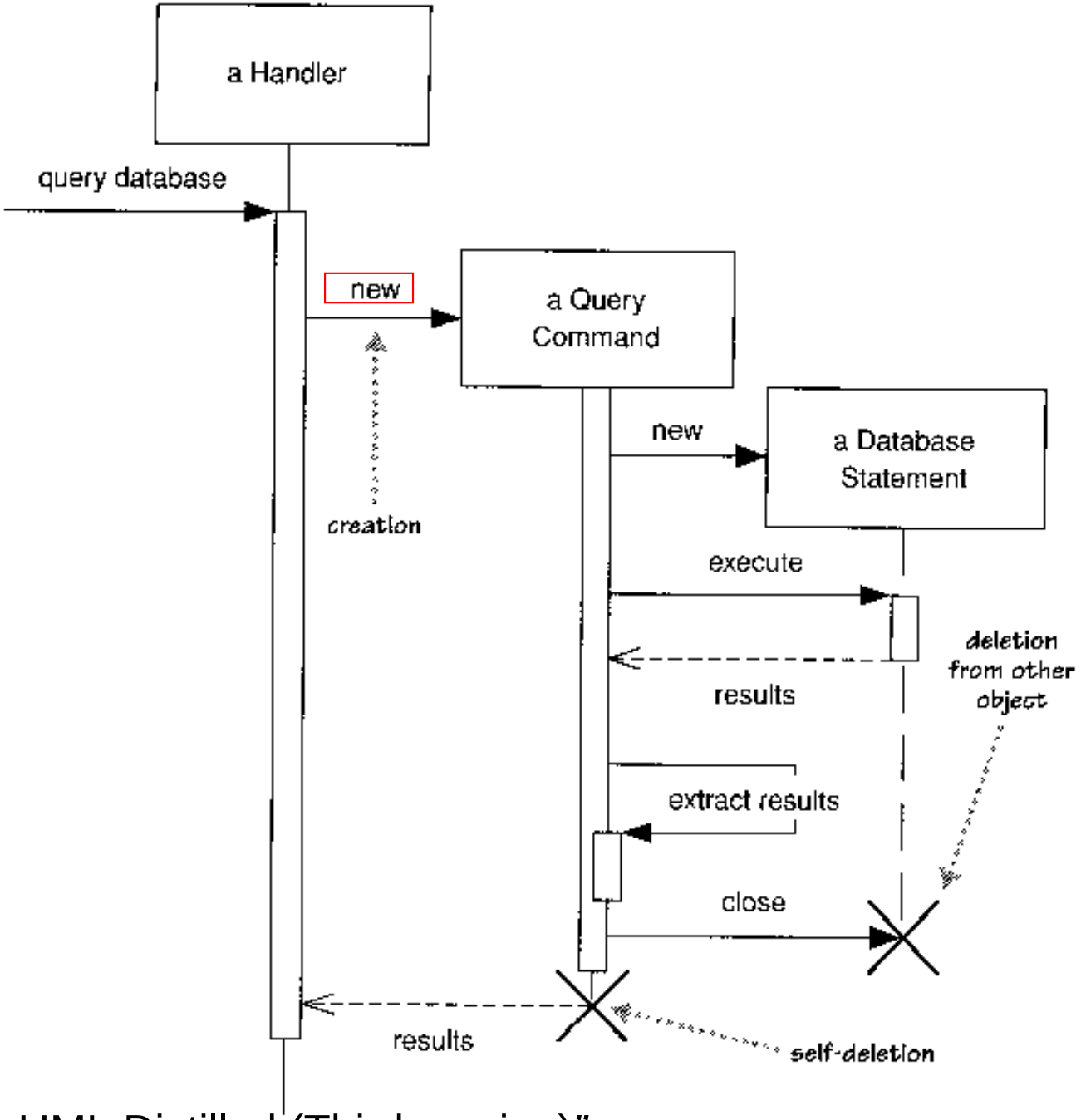
- Come precedentemente detto: uno **use case** è un insieme di scenari collegati da uno scopo comune dell'utente.
- Talvolta può risultare utile (e rendere più chiare le specifiche) esplicitare un particolare scenario di uno use case complesso.
- Ad esempio, nell'ambito delle specifiche dei requisiti del software, un modo di procedere può essere quello di dettagliare i vari use case, descrivendo in dettaglio il comportamento del sistema nei vari casi.
- In generale, un'alternativa alla descrizione testuale di uno scenario può essere rappresentata dai diagrammi di sequenza UML.
- I diagrammi di sequenza introdotti, come ogni diagramma UML che compare nel documento, dovrebbero sempre essere accompagnati da una descrizione sintetica in linguaggio naturale.

UML: Sequence diagrams

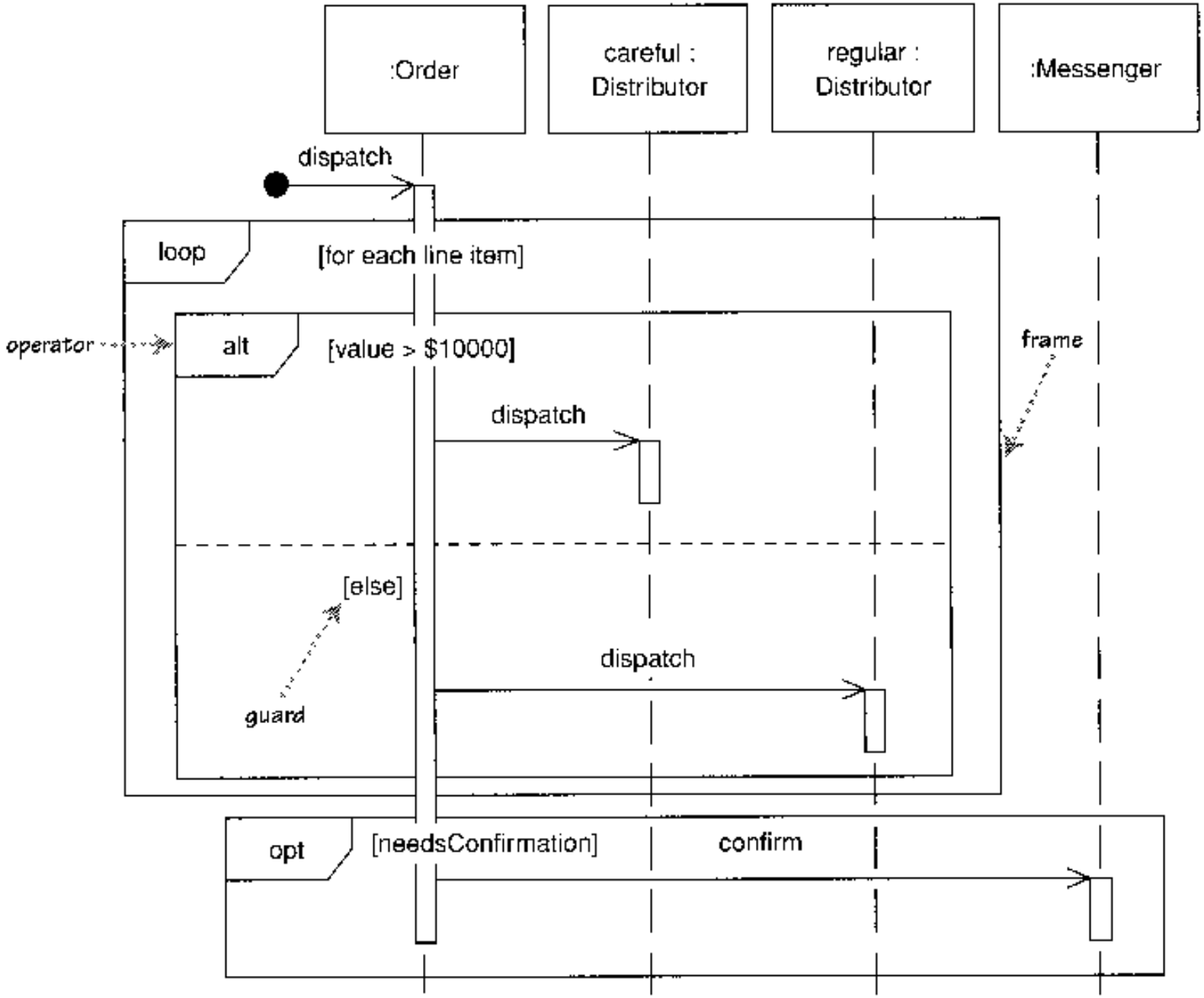
- I diagrammi di sequenza possono essere utilizzati (sebbene con una semantica e un dettaglio differenti) sia nella fase di definizione dei requisiti che nel progetto
- Da UML Distilled: *"*[...] *a sequence diagram captures the behavior of a single scenario* [...] *you should use sequence diagrams when you want to look at the behavior of several objects within a single use case. Sequence diagrams are good at showing collaborations among the objects; they are are not so good at precise definition of the behavior"*
- I sequence diagram non devono essere utilizzati per descrivere genericamente un algoritmo. Sebbene UML 2.0 offra formalismi di controllo (loop, sezioni alternative o opzionali) questi vanno usate con attenzione



"M. Fowler, UML Distilled (Third version)*"*



"M. Fowler, UML Distilled (Third version)*"*



"M. Fowler, UML Distilled (Third version)*"*

UML: Sequence diagrams

- In un DSR un sequence diagram dettaglia una particolare sequenza di eventi di uno use case;i “participant” sono entità generiche percepite dall’utente, ad alto livello -> non hanno necessariamente un corrispettivo definito sw/hw che le implementa.
- In un DSP un sequence diagram descrive un caso particolare di interazione tra classi; i participant devono essere istanze ben precise di classi descritte nel documento e i messaggi devono corrispondere a metodi presenti nell’interfaccia delle classi.

Riferimenti

Fowler, *UML Distilled*, 3° edition

Bruegge, Dutoit *Object-Oriented Software Engineering Using UML, Patterns and Java*

Baresi, Lavazza, Pianciamore *Dall'idea al codice con UML 2*

Cockburn*, Writing effective use cases*

Russell C. Bjork, Gordon College : CPS211: *"Object-Oriented Software Development" 2009* (materiale online del corso)