

Ingegneria del Software

UML

Diagramma delle Classi e di Interazione

Antonino Staiano

e-mail: antonino.staiano@uniparthenope.it

Ingegneria del Software, a.a. 2009/2010 – A. Staiano

Diagrammi delle classi

- Classi ed oggetti
 - I diagrammi delle classi descrivono la struttura del sistema in termini di classi ed oggetti
 - Le **classi** sono astrazioni che specificano gli attributi ed il comportamento di un insieme di oggetti
 - Una classe è una collezione di oggetti che condividono un insieme di attributi che contraddistinguono gli oggetti come membri della collezione
 - Gli **oggetti** sono entità che incapsulano lo stato ed il comportamento
 - Ogni oggetto ha un'identità mediante la quale ci si riferisce ad esso individualmente e che lo distingue dagli altri oggetti
 - In UML, classi ed oggetti sono rappresentati da riquadri composti da tre compartimenti
 - Parte alta: nome della classe o dell'oggetto
 - Centro: attributi
 - Parte bassa: operazioni
 - Le parti relative agli attributi e alle operazioni possono essere omesse per chiarezza

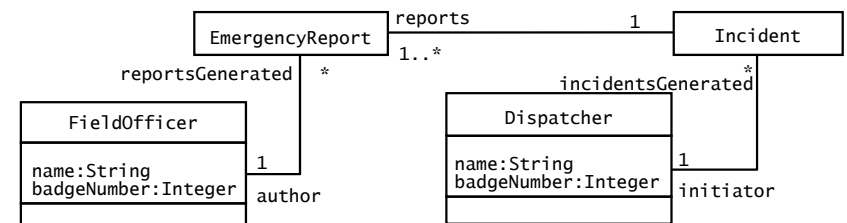
Ingegneria del Software, a.a. 2009/2010 – A. Staiano

Convenzioni UML per classi ed oggetti

- I nomi degli oggetti sono sottolineati per indicare che sono istanze
- I nomi delle classi iniziano con lettere maiuscole
- Agli oggetti, nei diagrammi degli oggetti, possono essere assegnati dei nomi (seguiti dalla loro classe) per semplificarne il riferimento
 - In questo caso, i nomi iniziano con lettere minuscole

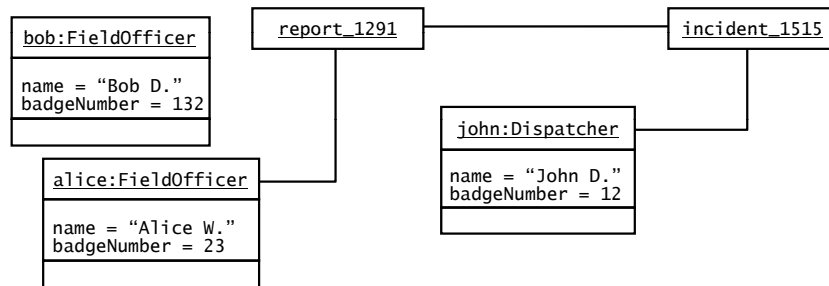
Ingegneria del Software, a.a. 2009/2010 – A. Staiano

Esempio di diagramma delle classi: classi partecipanti nel caso d'uso *ReportEmergency*



Ingegneria del Software, a.a. 2009/2010 – A. Staiano

Esempio di diagramma degli oggetti: oggetti partecipanti nello scenario *warehouseOnFire*

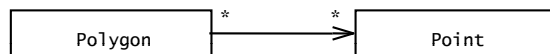


Associazioni e link

- Un **link** rappresenta una connessione tra due oggetti
- Le **associazioni** sono relazioni tra classi e rappresentano gruppi di link
- Nell'esempio *FRIEND*, ogni oggetto *FieldOfficer* ha anche una lista di *EmergencyReport* che sono stati scritti dal *FieldOfficer*
 - Nell'esempio del diagramma delle classi, la linea tra la classe *FieldOfficer* e la classe *EmergencyReport* è un'associazione
 - Nell'esempio del diagramma degli oggetti, la linea tra l'oggetto *alice:FieldOfficer* e l'oggetto *report_1291:EmergencyReport* è un link
 - Rappresenta uno stato del sistema per cui *alice:FieldOfficer* ha generato *report_1291:EmergencyReport*

Associazioni simmetriche e asimmetriche

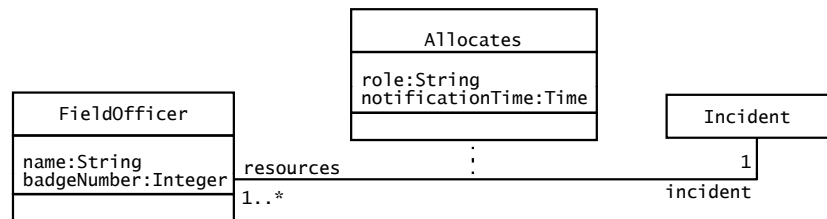
- Le associazioni possono essere bidirezionali (simmetriche) o unidirezionali (asimmetriche)
 - Nei diagrammi precedenti sono tutte simmetriche
- Una associazione asimmetrica è, ad esempio, quella tra le classi *Poligono* e *Punto*
 - La freccia di navigazione indica che il sistema supporta solo il verso da poligono a punto
 - Dato un poligono specifico, è possibile individuare tutti i punti che costituiscono il poligono. Dato un punto specifico, non è possibile individuare il poligono di cui il punto fa parte. Per convenzione, le associazioni senza frecce sono simmetriche



Classe di associazione

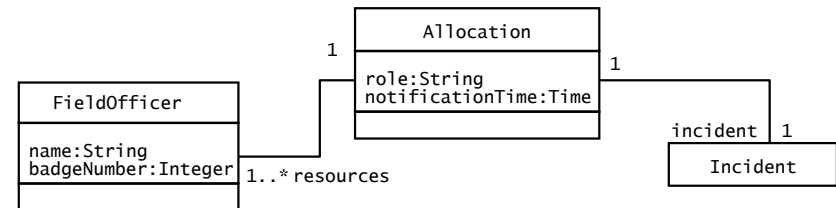
- Le associazioni sono simili alle classi poiché possono avere attributi ed operazioni
- Una tale associazione è chiamata classe di associazione
 - Disegnata con un simbolo di classe che contiene attributi e operazioni ed è connessa al simbolo di associazione con una linea tratteggiata
 - Ad esempio, l'allocazione dei *FieldOfficer* ad un Incidente è modellata come una classe di associazione con attributi ruolo (*role*) e ora di notifica (*notificationTime*)

Esempio di classe di associazione



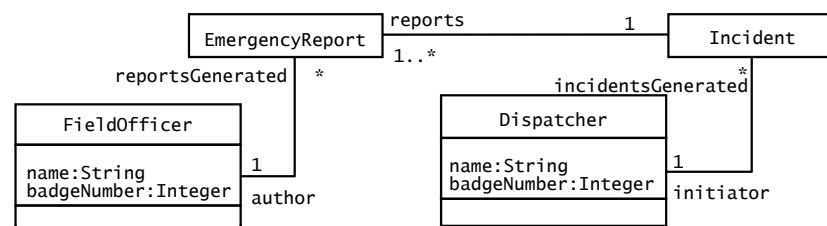
Modello alternativo per Allocation

Qualsiasi classe di associazione può essere trasformata in una classe e associazioni semplici



Ruoli

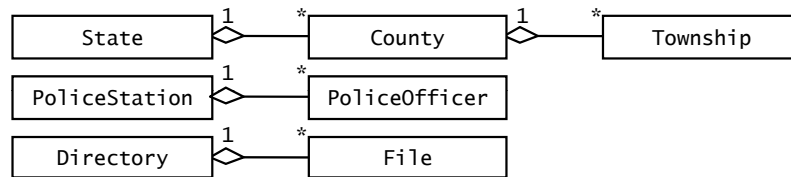
- Ciascuna estremità di un'associazione può essere etichettata con una stringa chiamata **ruolo**
- I ruoli dell'associazione tra le classi *EmergencyReport* e *FieldOfficer* sono autore (*author*) e rapporto generato (*reportGenerated*)
 - Etichettare le estremità delle associazioni con i ruoli consente di distinguere tra le multiple associazioni che si originano da una classe. Inoltre, i ruoli chiariscono lo scopo dell'associazione



Aggregazione

- Le associazioni sono usate per rappresentare un'ampia gamma di connessioni tra un insieme di oggetti
- Un tipo speciale di associazione si presenta frequentemente: le **aggregazioni** (denotate con una linea con testa di diamante)
 - Esempi:
 - uno *stato* contiene molti *paesi* che a loro volta contengono molte *città*
 - una *stazione di polizia* è costituita di un certo numero di *poliziotti*
 - una *directory* contiene un certo numero di *file*
 - Tali relazioni possono essere modellate con associazioni uno-a-molti
 - UML, invece, fornisce il concetto di aggregazione che consente di denotare aspetti gerarchici della relazione che può avere molteplicità sia uno-a-molti che molti-a-molti

Esempi di aggregazioni

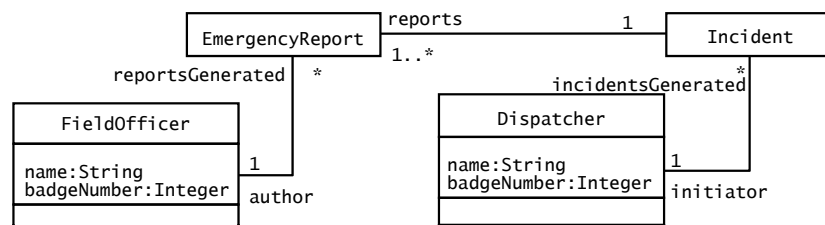


Molteplicità

- Ogni estremità di un'associazione può essere etichettata con un insieme di interi che indicano il numero di link che si originano da un'istanza della classe connessa all'estremità dell'associazione
- Questo insieme di interi è chiamato **molteplicità** dell'estremità dell'associazione

Esempio molteplicità

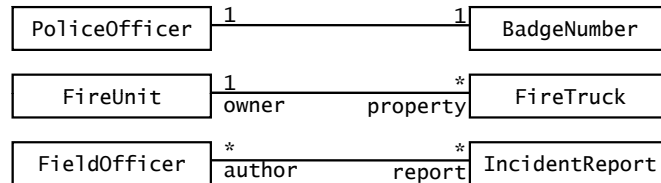
- L'estremità dell'associazione *author* ha una molteplicità pari ad 1
 - Significa che tutti gli *EmergencyReport* sono scritti esattamente da un *FieldOfficer* vale a dire, ogni oggetto *EmergencyReport* ha esattamente un link ad un oggetto della classe *FieldOfficer*
 - La molteplicità dell'estremità dell'associazione *reportsGenerated* è “molti” ed indicata con un asterisco (*) che indica 0..n



Tipi di molteplicità

- Le associazioni solitamente usate nei diagrammi sono di tre tipi:
 - **Associazione uno-a-uno**
 - Ha molteplicità 1 su entrambe le estremità: esiste esattamente un link tra le istanze di ogni classe
 - **Associazione uno-a-molti**
 - Ha molteplicità 1 su di una estremità e 0..n o 1..n dall'altra estremità: denota il rapporto di composizione tra due classi
 - **Associazione multi-a-molti**
 - Ha molteplicità 0..n o 1..n su ambo i lati: indica che un numero arbitrario di link posso esserci tra le istanze di due classi

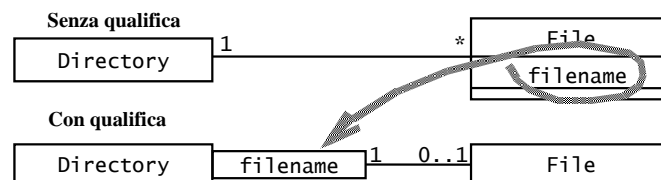
Esempi di molteplicità



Qualifica

- E' una tecnica per ridurre le molteplicità
 - Le associazioni con una molteplicità 0..1 o 1 sono più semplici da capire rispetto alle associazioni con molteplicità 0..n o 1..n
 - Spesso con le associazioni uno-a-molti, gli oggetti sul lato “molti” possono essere distinti l'un l'altro usando un nome
 - Esempio: in un file system gerarchico ogni file appartiene ad esattamente una directory. Ogni file è identificato univocamente da un nome nel contesto della directory. Molti file possono avere lo stesso nome nel contesto del file system; tuttavia due file non possono condividere lo stesso nome nella stessa directory

Esempio: le associazioni qualificate riducono la molteplicità



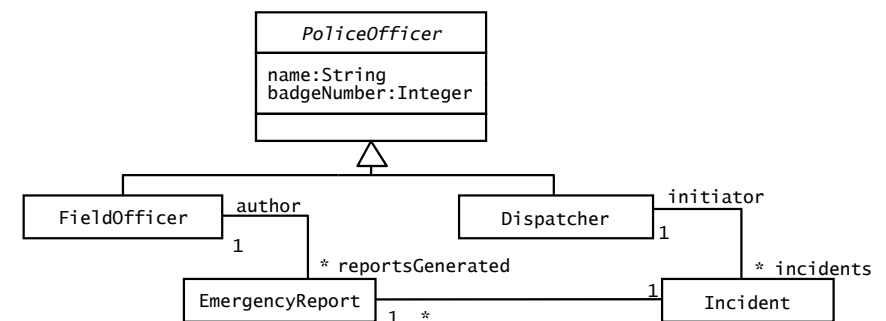
Qualificatori

- Senza qualifica l'associazione tra *Directory* e *File* ha una molteplicità 1 sul lato *Directory* e una molteplicità 0-a-molti sul lato *File*
- Possiamo ridurre la molteplicità sul lato *File* usando l'attributo *filename* come chiave, chiamata anche **qualificatore**. La relazione tra *Directory* e *File* è chiamata associazione qualificata
- E' sempre preferibile ridurre la molteplicità poiché il modello è reso più chiaro e devono essere presi in considerazioni meno casi
 - Gli sviluppatori dovrebbero esaminare ogni associazione che ha molteplicità uno-a-molti o molti-a-molti per verificare se aggiungere un qualificatore

Ereditarietà

- L'**ereditarietà** è la relazione tra una classe generale ed una o più classi specializzate
- L'ereditarietà consente di descrivere tutti gli attributi e le operazioni che sono comuni ad un insieme di classi
 - Esempio: *FieldOfficer* e *Dispatcher* hanno entrambi gli attributi *name* e *badgeNumber*. Tuttavia, *FieldOfficer* ha un'associazione con *EmergencyReport*, mentre *Dispatcher* ha un'associazione con *Incident*. Gli attributi comuni di *FieldOfficer* e *Dispatcher* possono essere modellati introducendo una classe *PoliceOfficer* che è specializzata da *FieldOfficer* e *Dispatcher*
 - *PoliceOfficer* è chiamata la **superclasse**; *FieldOfficer* e *Dispatcher* sono chiamate **sottoclassi**
 - Le sottoclassi ereditano gli attributi e le operazioni della loro superclasse

Esempio di generalizzazione



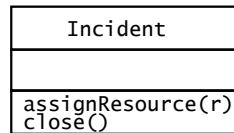
Classi astratte

- La classe *PoliceOfficer* è una classe astratta. Per distinguerla dalle classi concrete si scrive il nome in corsivo
- Le classi astratte sono utilizzate nella modellazione orientata agli oggetti per classificare concetti collegati riducendo, quindi, la complessità totale del modello
 - Eliminano la ridondanza

Oggetti e operazioni

- Il comportamento di un oggetto è specificato dalle **operazioni**
- Un oggetto richiede l'esecuzione di un'operazione ad un altro oggetto inviandogli un messaggio
- Il messaggio è confrontato con il metodo definito dalla classe a cui l'oggetto ricevente appartiene o da una qualsiasi sua superclasse
- I metodi di una classe in un linguaggio di programmazione orientato agli oggetti sono le implementazioni di queste operazioni

Esempio di operazioni della classe Incident



Applicazione dei diagrammi delle classi

- Usati per descrivere la struttura del sistema
- Durante la fase di analisi gli ingegneri costruiscono diagrammi delle classi per formalizzare la conoscenza del dominio dell'applicazione
- Le classi rappresentano gli oggetti partecipanti individuati nei diagrammi dei casi d'uso e di interazione
 - Descrivono i loro attributi e le operazioni
- Lo scopo dei modelli di analisi è di descrivere il proposito del sistema e scoprire i suoi confini
 - Ad esempio, un analista può esaminare la molteplicità dell'associazione tra *FieldOfficer* e *EmergencyReport* e chiedere all'utente se ciò è corretto
 - E' possibile avere più di un autore per l'*EmergencyReport*?
 - Sono previsti rapporti anonimi?
- La fase di analisi tiene fuori concetti implementativi
 - I diagrammi di classe sono rifiniti durante la fase di progettazione del sistema e degli oggetti

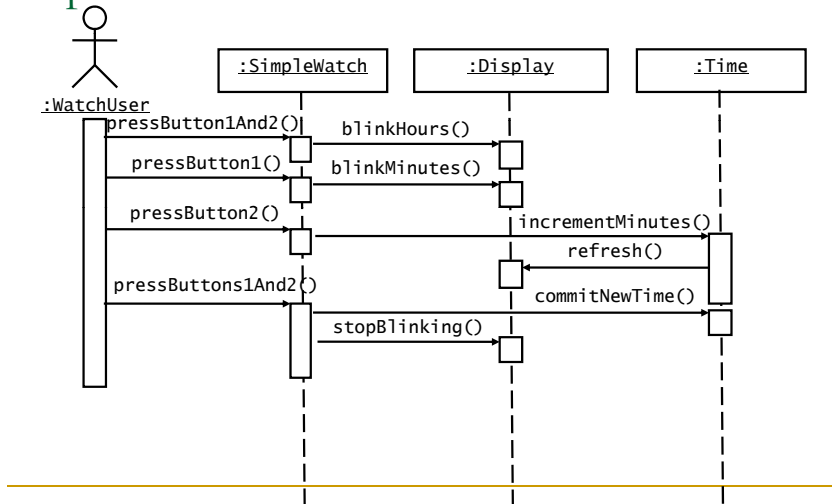
Diagrammi di Interazione

- Descrivono le modalità di comunicazione tra un insieme di oggetti interagenti
- Un oggetto interagisce con un altro oggetto inviando messaggi
 - La ricezione di un messaggio da un oggetto aziona l'esecuzione di un metodo che a sua volta può inviare messaggi ad altri oggetti
 - Possono essere inviati degli argomenti insieme al messaggio compatibilmente con i parametri del metodo di cui si richiede l'esecuzione

Diagrammi delle sequenze

- I diagrammi delle sequenze rappresentano orizzontalmente gli oggetti partecipanti nell'interazione e verticalmente il tempo
- Esempio
 - Un orologio con due pulsanti (2Bwatch)

Esempio diagramma di sequenze: impostazione ora sul 2Bwatch



Ingegneria del Software, a.a. 2009/2010 – A. Staiano

Notazioni

- Le colonne rappresentano gli oggetti che partecipano nell'interazione
- Le frecce rappresentano i messaggi
 - Le etichette rappresentano i nomi che possono contenere argomenti
- I rettangoli verticali rappresentano le attivazioni (esecuzione dei metodi)
- L'attore che inizia l'interazione è rappresentato nella prima colonna a sinistra
- I messaggi provenienti dall'attore rappresentano le interazioni descritte nei diagrammi dei casi d'uso
 - Se altri attori comunicano con il sistema durante il caso d'uso, questi attori sono rappresentati sul lato destro e possono ricevere messaggi

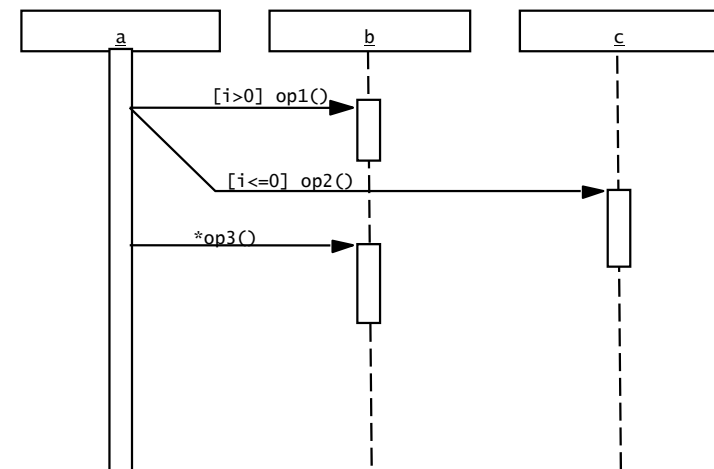
Ingegneria del Software, a.a. 2009/2010 – A. Staiano

Condizioni ed Iterazioni

- I diagrammi delle sequenze possono essere usati per descrivere una sequenza astratta (tutte le possibili interazioni) o sequenze concrete (una possibile interazione)
- Sono disponibili anche notazioni per esprimere condizioni o iterazioni (quando si descrivono tutte le possibili interazioni)
 - Una **condizione** su un messaggio è rappresentata da una espressione tra parentesi quadre prima del nome del messaggio. Se la condizione è vera il messaggio è inviato
 - Una invocazione **ripetitiva** di un messaggio è denotata da un '*' prima del nome del messaggio

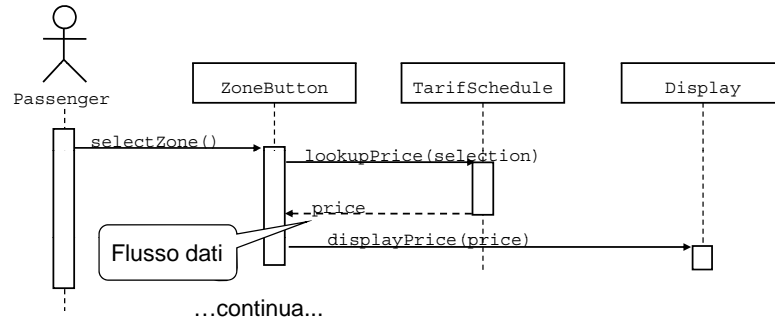
Ingegneria del Software, a.a. 2009/2010 – A. Staiano

Esempi di condizioni e iterazioni

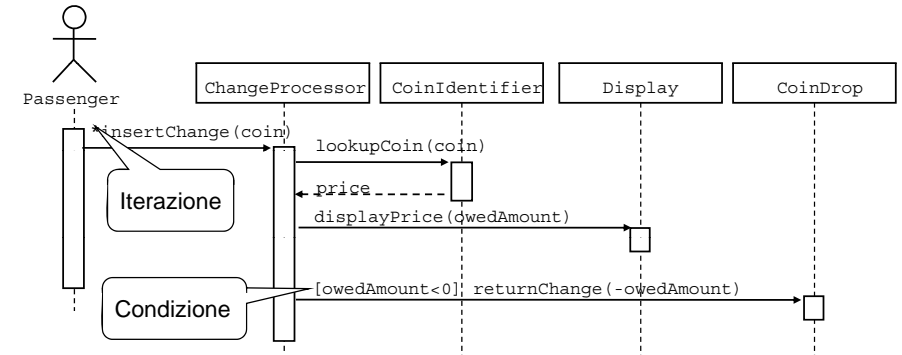


Ingegneria del Software, a.a. 2009/2010 – A. Staiano

Messaggi nidificati



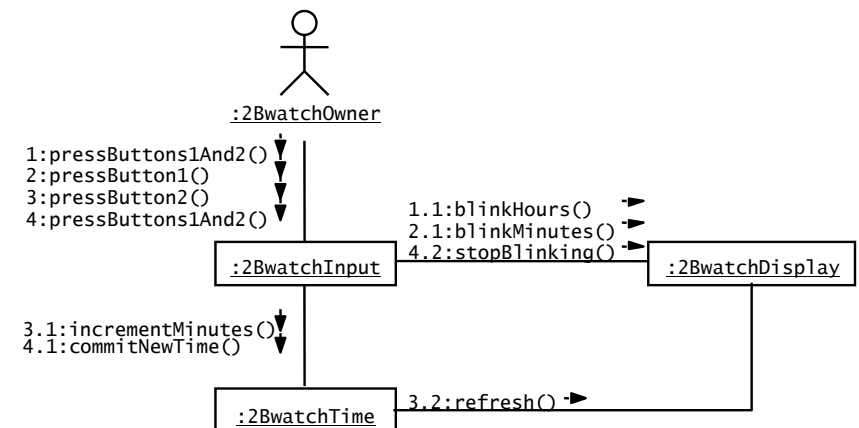
Iterazioni e Condizioni



Diagrammi delle collaborazioni

- Descrivono le stesse informazioni dei diagrammi delle sequenze
 - Rappresentano le sequenza dei messaggi numerando le interazioni
 - Ciò elimina la necessità di vincoli geometrici sugli oggetti con il risultato di diagrammi più compatti
 - Tuttavia, la sequenza dei messaggi diventa più difficile da seguire

Esempio: diagramma collaborazioni per 2Bwatch



Applicazione dei diagrammi di interazione

- Descrivono le interazioni tra diversi oggetti
- Portano alla luce le responsabilità delle classi nei diagrammi delle classi e scoprono, eventualmente, nuove classi
 - Aiutano gli sviluppatori nel decidere quali oggetti richiedono metodi particolari
 - Tipicamente c'è un diagramma di interazione per ogni caso d'uso focalizzandosi sul flusso di eventi
 - Lo sviluppatore identifica gli oggetti che partecipano al caso d'uso ed assegnano pezzi del comportamento del caso d'uso agli oggetti sotto forma di operazioni
 - Diagrammi delle classi e il corrispondente diagramma delle interazioni sono costruiti in tandem dopo che è stato definito un diagramma delle classi iniziale
 - Questo processo porta spesso a raffinamenti dei casi d'uso

Esercizi

- Disegnare un diagramma delle classi per i riferimenti bibliografici
- Disegnare un diagramma delle sequenze per lo scenario warehouseOnFire. Includere gli oggetti bob, alice, john, FRIEND, e istanze di altre classi di cui si ha bisogno. Disegnare solo i primi invii di messaggi