Ingegneria del Software

Modellazione mediante
Unified Modeling Language (UML)
Diagrammi dei casi d'uso

Antonino Staiano
e-mail: **antonino.staiano@uniparthenope.it**

Introduzione

- Le notazioni consentono l'articolazione di idee complesse in modo sintetico e preciso
- In progetti che coinvolgono molte persone di background tecnici e culturali diversi l'accuratezza e la chiarezza sono critici
  - I costi delle incomprensioni crescono rapidamente
- Perché una notazione consenta un'accurata comunicazione è necessario che abbia
  - Una semantica ben definita
  - Deve essere adeguata per rappresentare un dato aspetto di un sistema
  - Deve essere ben compresa da tutti i partecipanti al progetto
- UML fornisce uno spettro di notazioni per rappresentare diversi aspetti di un sistema ed è stato accettato come notazione standard nelle industrie

Unified Modeling Language (UML)

- linguaggio (e notazione) universale, per rappresentare qualunque tipo di sistema (software, hardware, organizzativo, …)
- Standard OMG (Object Management Group), dal nov.1997
- Creatori:
  - Grady Booch, Ivar Jacobson e Jim Rumbaugh
  - noti come "los gringos" dell'ingegneria del sw.

Cos'è UML

- E' un linguaggio di modellazione e specifica di sistemi, basato sul paradigma object-oriented
- Serve a specificare le caratteristiche di un nuovo sistema, oppure a documentarne uno già esistente
- E' uno strumento di comunicazione tra i diversi ruoli coinvolti nello sviluppo e nell'evoluzione dei sistemi
- UML svolge un'importantissima funzione di lingua franca nella comunità della progettazione e programmazione a oggetti
- Gran parte della letteratura di settore usa UML per descrivere soluzioni analitiche e progettuali in modo sintetico e comprensibile a un vasto pubblico

Cosa NON è UML

- Non è una "metodologia"
- E' un linguaggio non un metodo completo
  - Notazione, sintassi e semantica sono standard
- UML non è legato ad uno specifico processo, e non fornisce indicazioni sul proprio utilizzo
- Quindi è utilizzato da persone e gruppi che seguono approcci diversi (è "indipendente dai metodi")
- UML è indipendente dal linguaggio di programmazione utilizzato

Breve Storia (I)

- I linguaggi per la modellazione object-oriented iniziano a svilupparsi in diversi contesti negli anni '80.
  - Si trattava di notazioni di varia natura per descrivere la struttura di un sistema software a oggetti (in termini di classi e relazioni fra classi) ed il suo comportamento dinamico
- La proliferazione di queste notazioni diede luogo a quelle che furono poi battezzate "guerre dei metodi" (method wars), con diverse organizzazioni, che adottavano e sostenevano una particolare notazione a scapito delle altre
- Nella metà degli anni '90 diversi metodi e linguaggi iniziarono a fondersi, e si iniziò a delineare la possibilità di una integrazione dei principali formalismi in una notazione universalmente accettabile.
- Fra le metodologie e le notazioni più apprezzate e diffuse del periodo spiccavano OMT (ObjectModeling Technique) di Rumbaugh, e il cosiddetto metodo Booch di Grady Booch, entrambi ricercatori presso Rational Software.
- Il lavoro di unificazione iniziò con loro; in seguito si unì a questo sforzo Jacobson con la sua software house Objectory
  - Il primo risultato congiunto di questo team fu OOSE (Object Oriented Software Engineering).

Breve Storia (II)

- Mentre "i tre gringos" operavano per unificare i propri approcci all'analisi e alla progettazione a oggetti, il progetto fu accolto sotto l'egida dell'OMG (Object Management Group), un consorzio fondato con l'obiettivo di creare e gestire standard nel contesto dello sviluppo del software a oggetti.
  - Nel 1995, l'OMG raccolse tutti i principali metodologisti del settore in un incontro internazionale per discutere della notazione unificata.
  - Nel 1996 l'OMG emise una RFP (Request forProposal) per tale notazione.
  - Nello stesso anno, Booch, Rumbaugh e Jacobson misero a punto le release 0.9 e 0.91 di UML.
- Il progetto fu ben accolto dalla comunità internazionale e innumerevoli grandi organizzazioni si unirono a Rational per proseguirlo (per esempio Digital, HP, IBM, Microsoft, Oracle e Unisys)
- Questo gruppo esteso realizzò, nel 1997, UML 1.0, che fu sottoposto alla OMG come risposta alla RFP dell'anno precedente.
- La release 1.1 di UML contribuì a consolidare la semantica del linguaggio e incluse elementi tratti da una proposta avanzata indipendentemente all'OMG da un gruppo composto da IBM, ObjectTime, Ptech e altre.

Breve Storia (III)

- La versione 2.0 di UML, ufficializzata da OMG nel 2005, presenta numerose novità rispetto alla precedente versione 1.5, che resta comunque quella più diffusamente supportata dai tool di modellazione e citata nella letteratura.
- Principali novità di UML 2.0 :
  - ampliamento, razionalizzazione e revisione del metamodello, permettendo una maggiore flessibilità
  - introduzione del linguaggio formale OCL come parte integrante di UML
  - numerosi nuovi elementi per la costruzione dei diagrammi tradizionali
  - alcuni nuovi tipi di diagrammi

Aspetti della Modellazione

- Lo sviluppo di sistemi focalizza l'attenzione su tre modelli differenti di un sistema
  - **Modello funzionale**: descritto in UML dai diagrammi dei casi d'uso in cui si evidenziano le funzionalità del sistema dal punto di vista utente
  - **Modello ad oggetti**: rappresentato in UML con diagrammi delle classi, descrive il sistema in termini di oggetti, attributi, associazioni e operazioni
  - **Modello dinamico**: rappresentato in UML con diagrammi di interazione, diagrammi di attività. Descrive il comportamento interno del sistema

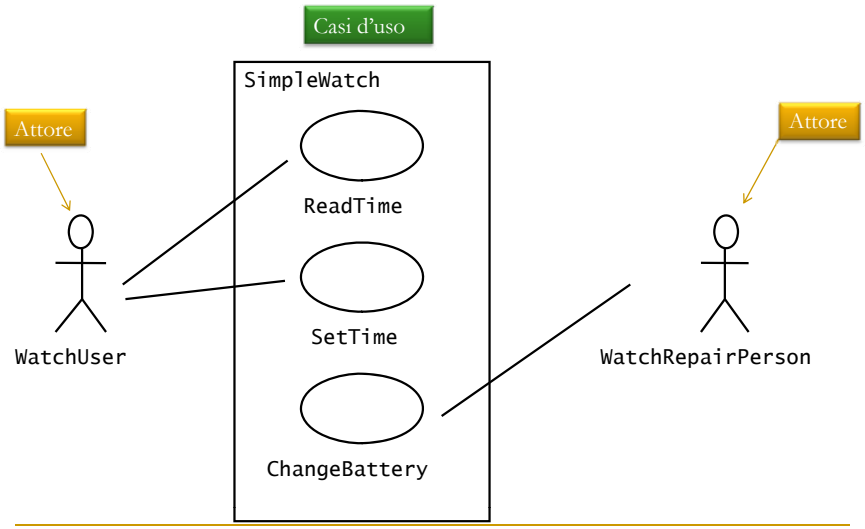
Panoramica di UML

- Diagrammi dei casi d'uso
- Diagrammi delle classi
- Diagrammi di interazione
- Diagrammi di stato
- Diagrammi delle attività

Diagrammi dei casi d'uso

- Sono usati durante la fase di ingegneria dei requisiti per rappresentare le funzionalità del sistema
- Si focalizzano sul comportamento del sistema da un punto di vista esterno
- Un caso d'uso descrive una funzione fornita dal sistema che porta ad un risultato visibile per un attore
- Un **attore** descrive una qualsiasi entità che interagisce con il sistema (un utente, un altro sistema, l'ambiente fisico del sistema)
- L'identificazione degli attori e dei casi d'uso definisce il confine del sistema
  - Differenzia i task eseguiti dal sistema e quelli eseguiti dal suo ambiente
  - Gli attori sono al di fuorri del confine mentre i casi d'uso sono all'interno del confine del sistema

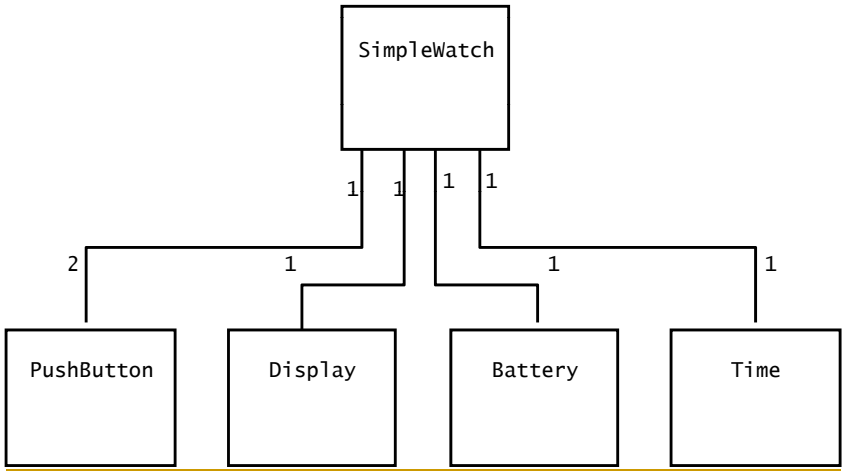
Diagramma casi d'uso per un orologio



Diagrammi delle classi

- Sono usati per descrivere la struttura del sistema
- Le classi sono astrazioni che specificano la struttura comune ed il comportamento di un insieme di oggetti
- Gli oggetti sono istanze delle classi e possono essere creati, modificati ed eliminati durante l'esecuzione del sistema
- Un oggetto ha uno stato che include i valori dei suoi attributi e collegamenti con altri oggetti
- I diagrammi delle classi descrivono il sistema in termini di oggetti, classi, attributi, operazioni e le loro associazioni

Diagramma delle classi di un orologio



Diagrammi di interazione

- Sono usati per formalizzare il comportamento dinamico del sistema e per visualizzare la comunicazione fra gli oggetti
- Utili per identificare oggetti aggiuntivi che partecipano nei casi d'uso
  - Gli oggetti coinvolti in un caso d'uso sono chiamati **oggetti partecipanti**

Diagramma delle sequenze di un l'orologio per il caso d'uso SetTime

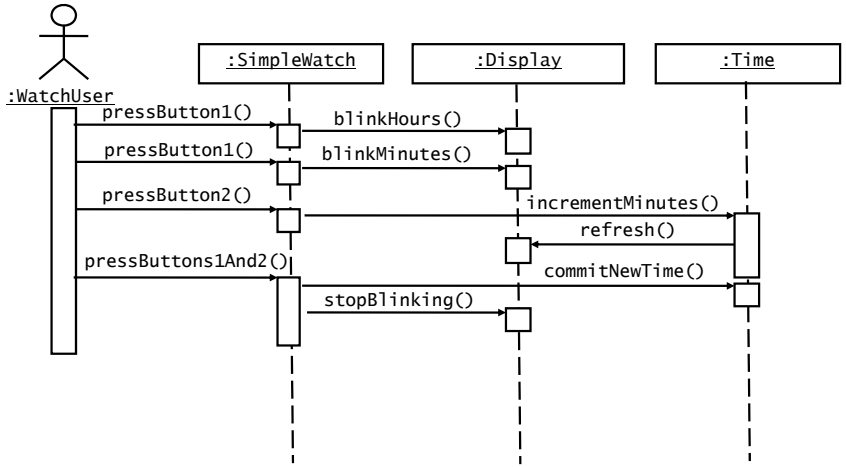


Diagramma degli stati

- I diagrammi degli stati descrivono il comportamento dinamico di un oggetto sulla base di un numero di stati e transizioni tra gli stati
- Uno stato rappresenta un particolare insieme di valori di un oggetto
  - Dato uno stato, una transizione rappresenta uno stato futuro in cui può trovarsi un oggetto e le condizioni associate con il cambiamento di stato

Diagramma degli stati per un orologio

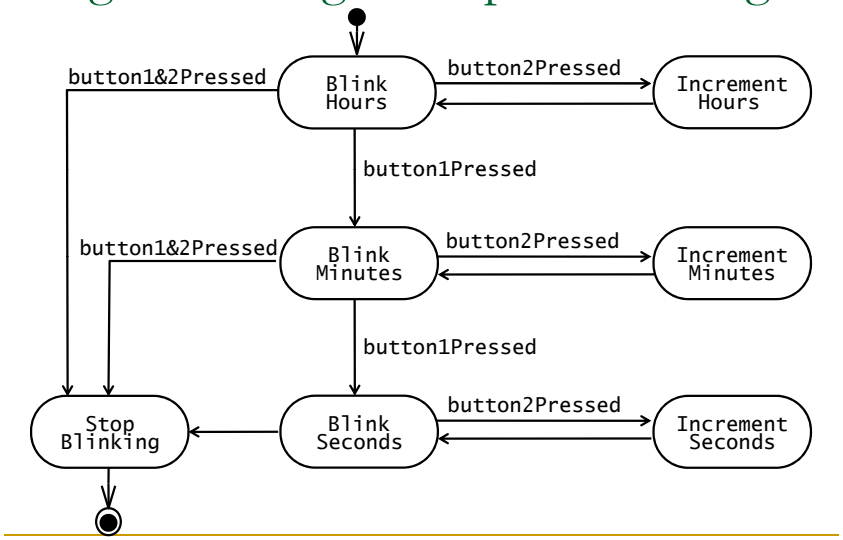
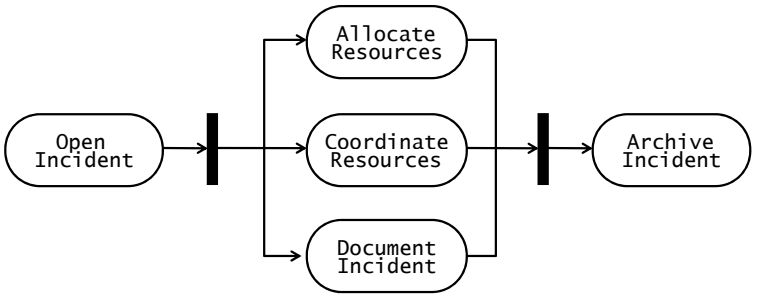


Diagramma delle attività

- Descrive il comportamento del sistema in termini di attività. Le attività sono elementi che rappresentano l'esecuzione di un insieme di operazioni
- Il completamento di queste operazioni fanno da trigger per transire in un'altra attività
  - Simili ai diagrammi di flusso e flusso di dati

Esempio di diagramma di attività



Concetti di Modellazione

- Un sistema è un insieme organizzato di parti comunicanti finalizzato ad uno scopo specifico
  - Una macchina, composta di quattro ruote, uno chassis, un corpo e un motore è progettata per trasportare persone
  - Un orologio, composta da una batteria, un circuito, ruote è progettato per misurare il tempo
- Le parti di un sistema possono essere considerate come sistemi più semplici chiamati sottosistemi
  - Il motore di una macchina, formato da cilindri, pistoni, un modulo ad iniezione e tante altri parti, è il sottosistema di una macchina

Complessità ed astrazione

- Molti sistemi sono composti da numerosi sottosistemi interconnessi in modo complesso
  - Nessuno sviluppatore da solo è in grado di gestire una tale complessità nella sua interezza
- La modellazione è un modo per trattare con la complessità
  - I sistemi complessi sono generalmente descritti da più di un modello, ognuno focalizzato su differenti aspetti o livelli di accuratezza
  - Modellare significa costruire un'astrazione di un sistema che mette a fuoco gli aspetti interessanti ed ignora dettagli irrilevanti
    - Cosa è rilevante o meno dipende dal problema

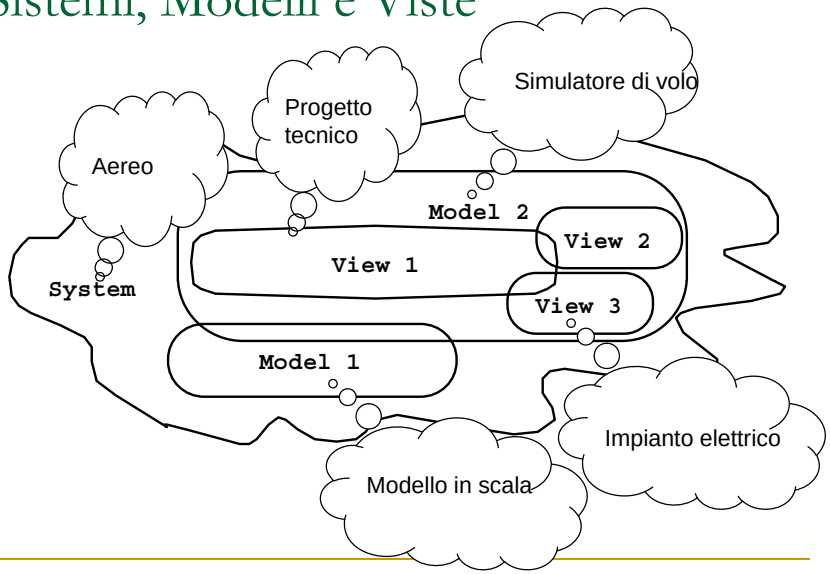
Perché modellare il software

- Il Software sta diventando sempre più complesso
  - Windows XP > 40 milioni di linee di codice
  - Un singolo programmatore non può gestire una tale quantità di codice
- Il codice non è semplice da capire dagli sviluppatori che non lo hanno scritto
- Abbiamo bisogno di rappresentazioni più semplici per i sistemi complessi
  - La modellazione è un modo per trattare la complessità

Sistemi, Modelli e Viste

- Un **modello** è un'astrazione che descrive un sottoinsieme di un sistema
- Una **vista** illustra degli aspetti specifici di un modello
- Una notazione è un insieme di regole grafiche o testuali per illustrare le viste
- Viste e modelli di un singolo sistema possono sovrapporsi
- Esempi:
  - Sistema: Aereo
  - Modelli: Simulatore di volo, modello in scala
  - Viste: tutti i modelli, impianto elettrico, impianto carburante

Sistemi, Modelli e Viste

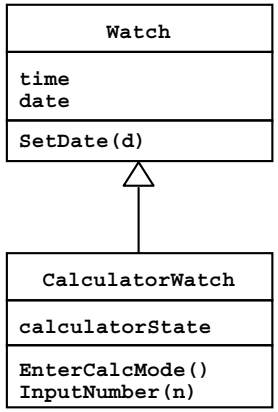


Tipi di dati, Tipi di dati Astratti e Istanze

- Un tipo di dato è un'astrazione nel contesto dei linguaggi di programmazione
  - Ha un nome unico che lo distingue da altri tipi di dati
  - Denota un insieme di valori membri del tipo di dato (vale a dire le **istanze** del tipo di dato)
  - Definisce la struttura e le operazioni valide per tutte le istanze del tipo di dato
    - Ad esempio, `int` in Java corrisponde a tutti gli interi con segno da -2^{32} a $2^{32}-1$. Le operazioni valide sono tutte le operazioni aritmetiche intere e tutte le funzioni e metodi che hanno un intero come argomento
- Un tipo di dato astratto è un tipo di dato definito da una specifica indipendente dalla implementazione
  - Consente agli sviluppatori di ragionare in termini di istanze senza preoccuparsi di una specifica implementazione
    - Ad esempio, il tipo di dato astratto Person può definire le operazioni getName(), getSocialSecurityNumber(), getAddress(). Il fatto che il social security number di una persona è memorizzato come una stringa o un numero non è visibile.

Classi e Oggetti

- Una classe è un'astrazione nella modellazione orientata agli oggetti e nei linguaggi di programmazione orientati agli oggetti
  - Incapsula sia la struttura che il comportamento
  - Possono essere definite in termini di altre classi usando l'ereditarietà
  - La classe CalculatorWatch è una specializzazione della classe Watch
- Il tipo di relazione tra una classe base e una classe più rifinita è chiamata **ereditarietà**
  - La classe di generalizzazione è chiamata **superclasse**
  - La classe di specializzazione è chiamata **sottoclasse**



Classi e Oggetti

- Una classe definisce un insieme di **operazioni** che possono essere applicate alle istanze
  - Le operazioni di una superclasse possono essere ereditate e applicate agli oggetti della sottoclasse
- Una classe definisce gli **attributi** che si applicano a tutte le istanze
  - Un attributo ha un nome unico nella classe ed un tipo
- Un **oggetto** è un'istanza di una classe
  - Ha un'identità e memorizza i valori degli attributi
  - Ogni oggetto appartiene ad esattamente una classe

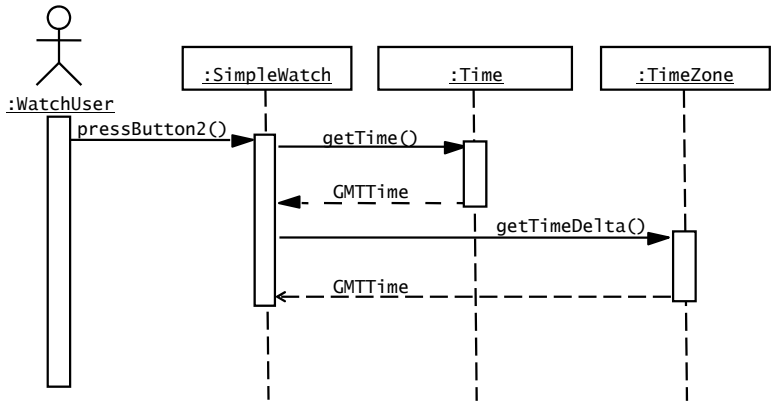
Convenzioni base di UML

- I rettangoli sono classi o istanze
- Gli ovali sono funzioni o casi d'uso
- Le istanze sono denotate con nomi sottolineati
  - `myWatch:SimpleWatch`
  - `Joe:Firefighter`
- I tipi sono denotati con nomi non sottolineati
  - `SimpleWatch`
  - `Firefighter`
- I diagrammi sono grafi
  - I nodi sono entità
  - Gli archi sono relazioni tra le entità

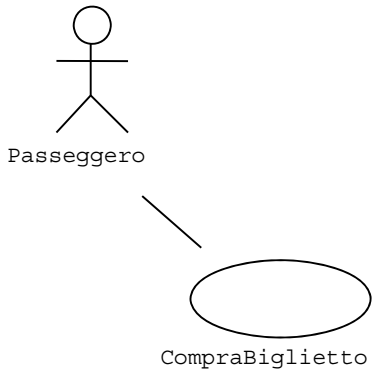
Classi di eventi, Eventi e Messaggi

- Le classi di eventi sono astrazioni che rappresentano un tipo di evento per cui il sistema ha una risposta comune
- Un evento, un'istanza di una classe di eventi, è un'occorrenza rilevante all'interno del sistema
  - Ad esempio, un evento è uno stimolo proveniente da un attore , un time out o l'invio di messaggi tra due oggetti
- Spedire un messaggio è il meccanismo per cui l'oggetto che invia richiede l'esecuzione di un'operazione nell'oggetto ricevente
  - Il messaggio è composto di un nome ed un numero di argomenti. L'oggetto ricevente passa gli argomenti all'operazione corrispondente. I risultati sono inviati al richiedente
- Eventi e messaggi sono istanze: rappresentano occorrenze concrete nel sistema

Invio di messaggi

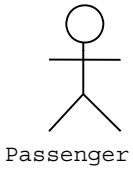


Diagrammi dei casi d'uso



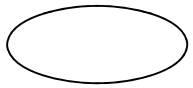
- Usati durante la fase di scoperta dei requisiti per rappresentare il comportamento esterno
- Gli ***attori*** rappresentano ruoli, cioè, un tipo di utente del sistema
- ***I casi d'uso*** rappresentano una sequenza di interazioni per un tipo di funzionalità
- Il modello dei casi d'uso è l'insieme di tutti i casi d'uso. E' una descrizione completa della funzionalità del sistema e del suo ambiente

Attori



- Un attore modella un'entità esterna che comunica con il sistema:
  - Utente
  - Sistema esterno
  - Ambiente fisico
- Un attore ha un nome unico e una descrizione opzionale
- Esempi:
  - Passeggero: Una persona nel treno
  - Satellite GPS: fornisce al sistema le coordinate del GPS

Caso d'uso



Un caso d'uso rappresenta una classe di funzionalità fornito dal sistema come un flusso di eventi

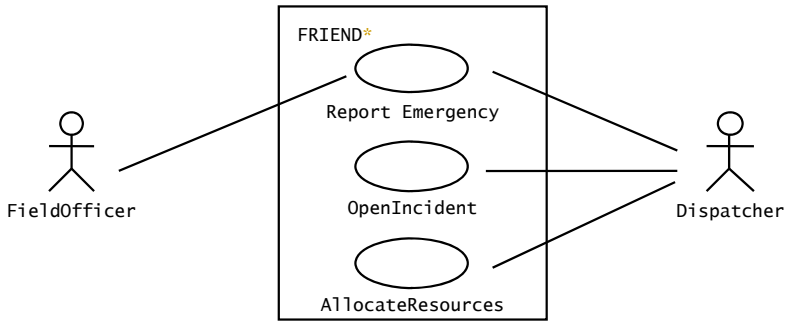
Un caso d'uso consiste di:

- Nome unico
- Attori partecipanti
- Condizioni d'entrata
- Flusso di eventi
- Condizioni di uscita
- Requisiti speciali

Esempio: sistema gestione incidenti

- I field officer (ad esempio, un poliziotto o un pompiere), hanno accesso ad un computer wireless che consente di interagire con un Dispatcher.
  - Il dispatcher può visualizzare lo stato corrente di tutte le risorse, come le macchine di polizia o autocarri, sul video del computer e spedire una risorsa inserendo comandi da una stazione di lavoro.
- In questo esempio FieldOfficer e Dispatcher sono attori.

Esempio: sistema di gestione degli incidenti



* First Responder Interactive Emergency Navigational Database

Template per i casi d'uso

- Per descrivere un caso d'uso è possibile usare un template composto da sei campi
  - **Nome** del caso d'uso: unico in tutto il sistema in modo da essere usato senza ambiguità da tutti i partecipanti al progetto
  - **Attori partecipanti**: sono gli attori che interagiscono con il caso d'uso
  - **Condizioni di entrata**: descrivono le condizioni che devono essere soddisfatte prima che il caso d'uso sia iniziato
  - **Flusso di eventi**: descrive la sequenza di interazioni del caso d'uso. Sono numerati per riferimenti successivi
    - I casi comuni e i casi eccezionali sono descritti separatamente per chiarezza
  - **Condizioni di uscita**: descrive le condizioni che sono soddisfatte dopo il completamento del caso d'uso
  - **Requisiti di qualità**: requisiti non legati alla funzionalità. Vincoli sulle prestazioni del sistema

Nome	ReportEmergency
Attori Partecipanti	Iniziato dal FieldOfficer Comunica con il Dispatcher
Flusso eventi:	1. Il FieldOfficer attiva la funzione "Report Emergency del terminale 2. FRIEND risponde presentando una form al FieldOfficer 3. Il FieldOfficer riempie la form selezionando il livello di emergenza, tipo, località, breve descrizione della situazione. Il FieldOfficer descrive anche possibili risposte alla situazione di emergenza. Appena finito Il FieldOfficer invia la form 4. FRIEND riceve la form e notifica al Dispatcher 5. Il Dispatcher rivede le informazioni sottomesse e crea un Incidente nel DB invocando il caso d'uso OpenIncident. Il Dispatcher seleziona una risposta e accetta il rapporto. 6. FRIEND visualizza l'accettazione e la risposta selezionata al FieldOfficer
Condizioni di entrata	Il FieldOfficer è loggato in FRIEND
Condizioni di uscita	Il FieldOfficer ha ricevuto un'accettazione e una risposta selezionata dal Dispatcher, OR Il FieldOfficer ha ricevuto una spiegazione indicante perché la transazione non è stata eseguita
Requisiti di qualità	Il rapporto del FieldOfficer è accettato entro 30 secondi La risposta selezionata arriva non più tardi di 30 secondi dopo che è stata inviata dal Dispatcher

Scenari

- Un caso d'uso è un'astrazione che descrive tutti i possibili scenari che coinvolgono la funzionalità descritta
- Uno scenario è un'istanza del caso d'uso che descrive un insieme concreto di azioni
  - Sono usati come esempi per illustrare i casi comuni. L'obiettivo è la comprensibilità.
  - I casi d'uso descrivono tutti i possibili casi. L'obiettivo è la completezza

Template per gli scenari

- Si usa un template con tre campi per descrivere uno scenario
  - Il **nome dello scenario**: consente di riferirlo in modo non ambiguo. Il nome è sottolineato per indicare che è un'istanza
  - Le **istanze degli attori partecipanti**: indica quali istanze degli attori sono coinvolte nello scenario. Anch'essi hanno nomi sottolineati
  - **Flusso di eventi**: descrive la sequenza di eventi passo dopo passo

Nome Scenario	warehouseOnFire
Istanze attori partecipanti	bob, alice: FieldOfficer john: Dispatcher
Flusso di eventi	1. Bob, guidando lungo la strada principale nella sua patrol, osserva del fumo proveniente da un magazzino. Il suo partner, Alice, attiva la funzione "Report Emergency" dal laptotp del FRIEND. 2. Alice immette l'indirizzo dell'edificio, una breve descrizione della sua ubicazione (angolo nord-ovest) ed un livello di emergenza. In aggiunta ad un'unità dei vigili del fuoco, richiede diverse unità paramediche poiché la zona non è molto trafficata. Conferma l'input ed attende l'accettazione. 3. John, il Dispatcher, è allertato per l'emergenza da un beep dalla sua stazione di lavoro. Rivede le informazioni sottomesse da Alice e accetta il rapporto. Alloca un'unità dei vigili del fuoco e due unità paramediche sul sito dell'Incidente ed invia la stima del tempo di arrivo (ETA) ad Alice. 4. Alice riceve l'accettazione e l'ETA.

Relazioni tra i casi d'uso

- Quando i casi d'uso e gli attori si scambiano informazioni, si dice che essi comunicano
- Gli scambi di informazione possono essere rappresentati con le relazioni di comunicazione
- I diagrammi dei casi d'uso possono includere quattro tipi di relazioni: **comunicazione**, **inclusione, estensione** ed **ereditarietà**

Relazioni di comunicazione

- Attori e casi d'uso comunicano quando si scambiano informazioni
- Le relazioni di comunicazione sono illustrate con linee solide tra i simboli dell'attore e del caso d'uso
- In riferimento al sistema FRIEND di gestione incidenti, gli attori FieldOfficer e Dispatcher comunicano con il caso d'uso ReportEmergency. Solo l'attore Dispatcher comunica con i casi d'uso OpenIncident e AllocateResources
- Le relazioni di comunicazione tra attori e casi d'uso possono essere usate per denotare funzionalità di accesso
  - FieldOfficer e Dispatcher sono forniti di interfacce differenti al sistema ed hanno accesso a funzionalità differenti

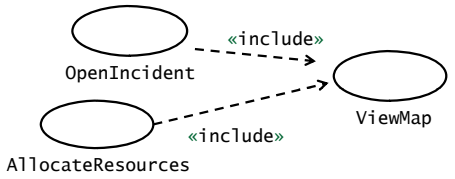
Relazioni di inclusione

- E' possibile ridurre la complessità di un modello identificando delle cose in comune tra differenti casi d'uso
  - Esempio: assumiamo che il Dispatcher premendo un tasto abbia accesso ad una mappa stradale.
    - Ciò può essere modellato da un caso d'uso ViewMap incluso nei casi d'uso OpenIncident e AllocateResources
  - Il modello risultante descrive solo la funzionalità ViewMap una volta, riducendo di fatto la complessità del modello dei casi d'uso complessivo

Relazioni di inclusione

- Due casi d'uso sono legati da una relazione di inclusione se uno di essi include il secondo nel suo flusso di eventi
- In UML, le inclusioni sono rappresentate da frecce tratteggiate aperte che si originano dal caso d'uso che include
  - Le relazioni di inclusione sono etichettate con <<include>>

Esempio: relazione «include»



Esempio: relazione «include» testuale

Nome	AllocateResources
Attori Partecipanti	Iniziato da Dispatcher
Flusso eventi:	…
Condizioni di entrata	Il Dispatcher apre un Incidente
Condizioni di uscita	Risorse aggiuntive sono assegnate ad un incidente Le risorse ricevono notizie circa nuovi assegnamenti Il FieldOfficer che ha a carico l'incidente riceve notizia circa le nuove risorse
Requisiti di qualità	In qualsiasi punto del flusso di eventi, questo caso d'uso può **includere** il caso d'uso ViewMap. Il caso d'uso ViewMap è iniziato quando il Dispatcher invoca la funzione "map". Quando invocato all'interno di questo caso d'uso, il sistema scorre la mappa in modo che il luogo dell'incidente sia visibile al Dispatcher

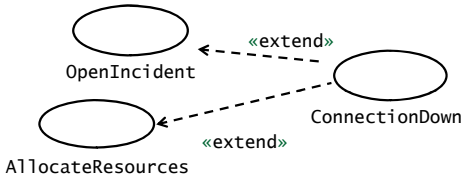
Relazioni di estensione

- Le relazioni di estensione sono un modo alternativo per ridurre la complessità in un modello dei casi d'uso
- Un caso d'uso può estendere un altro caso d'uso aggiungendo eventi
- Una relazione di estensione indica che un caso d'uso estende un secondo caso d'uso (caso d'uso base) quando descrive in modo più ampio e dettagliato una variante del'attività del caso d'uso base
- Una tipica applicazione di relazione di estensione è la specifica di un comportamento eccezionale

Esempio

- Assumiamo che la connessione di rete tra il Dispatcher e il FieldOfficer si può interrompere in qualsiasi momento (ad esempio quando il FieldOfficer entra in un tunnel)
- Il caso d'uso ConnectionDown descrive l'insieme di eventi che riguardano il sistema e gli attori quando la connessione è persa
  - ConnectionDown estende i casi d'uso OpenIncident e AloccateResources
- Separare comportamenti eccezionali da comportamenti comuni consente di scrivere casi d'uso più brevi e maggiormente focalizzati

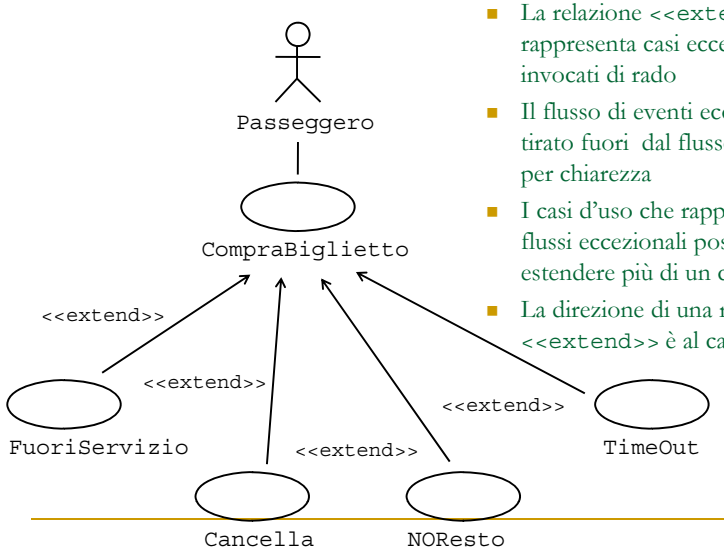
Esempio: relazione «extend»



Esempio: relazione «extend» testuale

Nome	ConnectionDown
Attori Partecipanti	Comunica con FieldOfficer e Dispatcher
Flusso eventi:	…
Condizioni di entrata	Questo caso d'uso **estende** i casi d'uso OpenIncident e AllocateResources. E' iniziato dal sistema quando la connessione di rete tra il FieldOfficer e il Dipsatcher è persa.
Condizioni di uscita	…

La relazione <<extend>>



- La relazione <<extend>> rappresenta casi eccezionali o invocati di rado
- Il flusso di eventi eccezionali è tirato fuori dal flusso principale per chiarezza
- I casi d'uso che rappresentano flussi eccezionali possono estendere più di un caso d'uso
- La direzione di una relazione <<extend>> è al caso d'uso esteso

Relazioni <<include>> e <<extend>>

- La differenza tra le relazioni di inclusione e di estensione sta nel luogo della dipendenza
  - Assumiamo di aggiungere molti nuovi casi d'uso per l'attore Dispatcher, ad esempio UpdateIncident e ReallocateResources.
  - Se modellassimo il caso d'uso ConnectionDown con la relazione include, gli autori dei casi d'uso UpdateIncident e ReallocateResources dovrebbero conoscere e di includere il caso d'uso ConnectionDown
  - Con la relazione di estensione, invece, solo il caso d'uso ConnectionDown deve essere modificato per estendere i casi d'uso aggiuntivi
- In generale, i casi relativi ad eccezioni come help, errori e condizioni inattese, sono modellati con relazioni di estensione
- I casi d'uso che descrivono comportamenti condivisi da un insieme limitato di casi d'uso sono modellati con la relazione di inclusione

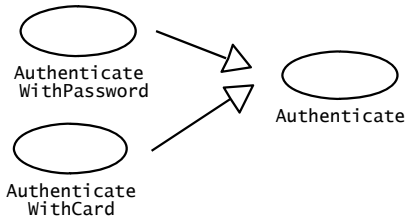
Relazioni di ereditarietà

- Terzo meccanismo per ridurre la complessità di un modello
- Un caso d'uso può specializzare un altro caso d'uso più generale aggiungendo più dettaglio
- Ad esempio, ai FieldOfficer è richiesto di autenticarsi prima di usare FRIEND
  - Durante le prime fasi delle scoperta dei requisiti, l'autenticazione è modellata come un caso d'uso ad alto livello Authenticate
  - Successivamente, gli sviluppatori descrivono Authenticate con maggior dettaglio e consentono diverse piattaforme hardware
  - Il risultato sono due ulteriori casi d'uso, AuthenticateWithPassword, che consente ai FieldOfficer di autenticarsi senza hardware ausiliari e AuthenticateWithCard che consente di autenticarsi con una smart card

Relazioni di ereditarietà

- I due nuovi casi d'uso sono rappresentati come specializzazioni del caso d'uso Authenticate
- Nella rappresentazione testuale, i casi d'uso specializzati ereditano l'attore che inizia e le condizioni di entrata e uscita dal caso d'uso generale

Esempio: relazione di generalizzazione



Esempio: relazione di ereditarietà testuale

Nome	AuthenticateWithCard
Attori Partecipanti	**Ereditato** dal caso d'uso Authenticate
Flusso eventi:	1. Il BankCustomer inserisce la propria carta nel Bancomat 2. Il Bancomat accetta la carta e richiede all'attore il PIN 3. Il BankCustomer immette il PIN con il tastierino numerico 4. Il Bancomat controlla il PIN immesso con il PIN memorizzato sulla carta. Se i PIN sono uguali, il BankCustomer è autenticato. Altrimenti, il Bancomat rifiuta il tentativo di autenticazione
Condizioni di entrata	**Ereditato** dal caso d'uso Authenticate
Condizioni di uscita	**Ereditato** dal caso d'uso Authenticate

Extend e inheritance

- Le relazioni extend e inheritance sono differenti:
  - In extend ogni caso d'uso descrive un differente flusso di eventi per fare un compito differente
    - Il caso d'uso OpenIncident descrive le azioni che si verificano quando il Dispatcher crea un nuovo incidente, mentre il caso d'uso ConnectionDown descrive le azioni che occorrono durante l'interruzione della connessione
  - Nel caso della relazione di ereditarietà, invece, AuthenticateWithPassword e Authenticate entrambe descrivono le azioni che si verificano durante l'autenticazione, benché a differenti livelli di astrazione

Applicazione dei diagrammi dei casi d'uso

- I casi d'uso e gli attori definiscono i confini del sistema
- Sono sviluppati durante la fase di scoperta dei requisiti, spesso con i clienti e gli utenti
- Durante la specifica, i casi d'uso sono rifiniti e corretti poiché sono rivisti anche dagli sviluppatori e validati rispetto ai casi reali