Ingegneria del Software

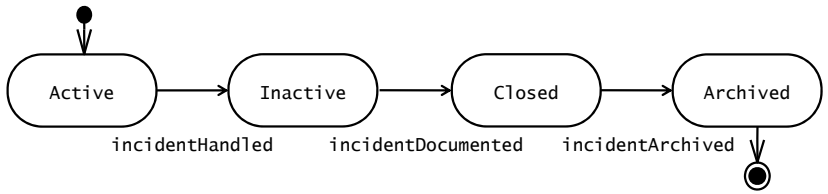
UML
Diagrammi degli Stati e delle Attività

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

Diagrammi degli Stati

- Descrivono la sequenza di stati di un oggetto in seguito all'occorrenza di eventi esterni
- Sono estensioni del modello della macchina a stati finiti
  - Forniscono le notazioni per innestare stati e macchine degli stati
  - Forniscono notazioni per legare le transizioni con gli invii di messaggi e le condizioni sugli oggetti
- Uno stato è una condizione soddisfatta dagli attributi di un oggetto
  - Ad esempio, un oggetto *Incident* in FRIEND può trovarsi in uno di quattro stati: Active, Inactive, Closed, Archived
    - Stato Active: situazione che richiede una risposta
    - Stato Inactive: situazione gestita ma i cui rapporti non sono stati ancora scritti
    - Stato Closed: situazione gestita e documentata
    - Stato Archived: incidente in stato Closed la cui documentazione è stata archiviata

Diagramma degli stati per la classe *Incident*

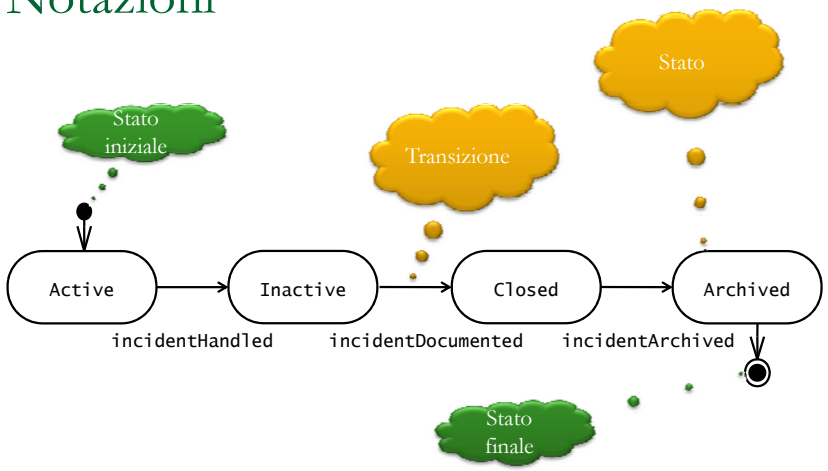


- I quattro stati possono essere rappresentati con un singolo attributo, ***status***, nella classe Incident che può assumere uno dei quattro valori {*Active, Inactive, Closed, Archived*}
  - In generale, uno stato può essere determinato dai valori di più attributi

Transizioni e notazioni

- Una transizione rappresenta un cambiamento di stato attivato da eventi, condizioni o dal tempo
- Uno stato è rappresentato da un rettangolo arrotondato
- Una transizione è rappresentata da frecce che connettono due stati
- Gli stati sono etichettati col proprio nome
- Un circoletto nero rappresenta lo stato iniziale
- Un circoletto che circonda un circoletto nero rappresenta uno stato finale

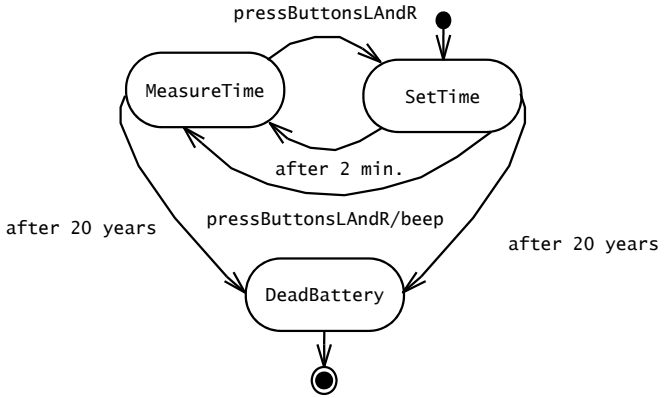
Notazioni



Esempio 2Bwatch

- Al più alto livello di astrazione, 2Bwatch ha due stati, *MeasureTime* e *SetTime*
- 2Bwatch cambia stato quando l'utente preme e rilascia entrambi i pulsanti simultaneamente
- Durante la transizione dallo stato SetTime allo stato MeasureTime, 2Bwatch emette un beep
  - Indicato dall'azione /beep sulla transizione
- Quando 2Bwatch e acceso per la prima volta si trova nello stato *SetTime*
  - Indicato dallo dal cerchietto nero che indica lo stato iniziale
- Quando la batteria si esaurisce, l'orologio non funziona
  - Cerchietto nero circondato da un altro cerchietto, lo stato finale

Esempio 2Bwatch *SetTime*



Azioni

- Le azioni sono piccoli comportamenti atomici eseguiti in punti specifici nella macchina a stati
- Le azioni richiedono per l'esecuzione una breve quantità di tempo e non possono essere interrotte
- Le azioni possono verificarsi in tre luoghi
  - durante una transizione
  - quando si entra in uno stato
  - quando si esce da uno stato

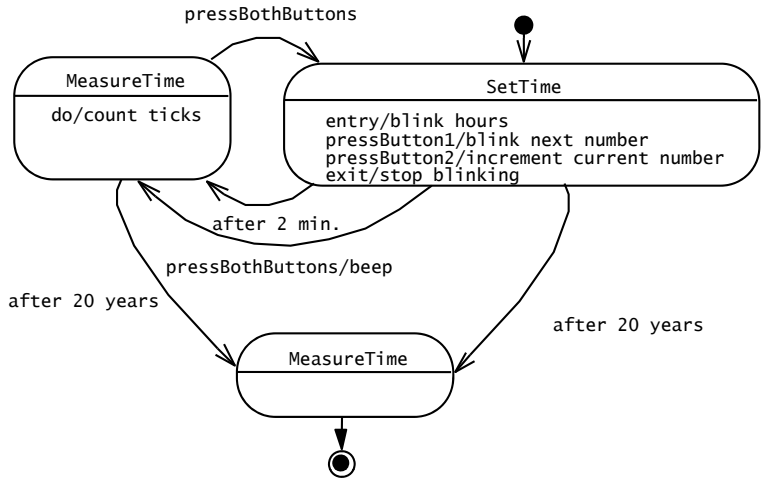
Transizioni interne

- Durante una transizione, le azioni di uscita dello stato sorgente sono eseguite per prime, poi sono eseguite le azioni associate con le transizioni e poi sono eseguite le azioni di entrata dello stato di destinazione
- Una **transizione interna** è una transizione che non lascia lo stato
  - Sono causate da eventi e possono avere delle azioni associate
  - L'attivazione di una transizione interna non comporta alcuna azione di uscita o entrata

Attività

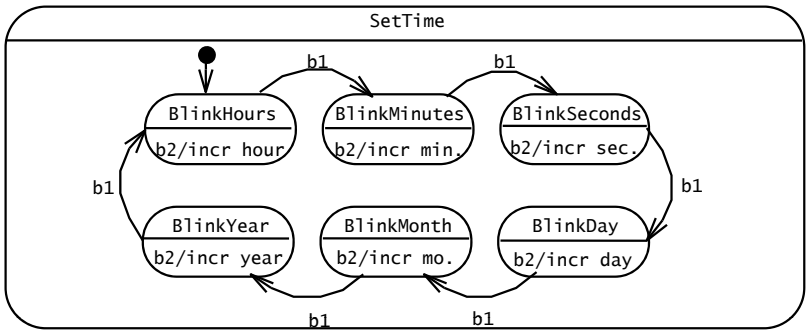
- Un'attività è un comportamento che è eseguito fintantoché un oggetto si trova in un dato stato
  - Un'azione è breve e non interrompibile, un'attività può richiedere un certo quantitativo di tempo ed è interrotta quando ha inizio una transizione che esce da uno stato
- Le attività sono rappresentate con l'etichetta *do* posizionata all'interno dello stato in cui è eseguita

Transizioni interne associate con lo stato *SetTime*



Diagrammi di stato annidati

- Riducono la complessità. Possono essere usati in alternativa alle transizioni interne
- Ad esempio, il numero corrente è modellato come stato annidato
  - Le azioni corrispondenti alla modifica sono modelltate usando transizioni interne



Applicazione dei diagrammi di stato

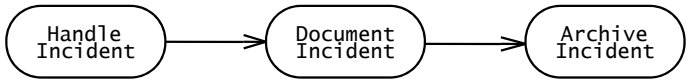
- Usati per rappresentare comportamenti non banali di un sottosistema o un oggetto
- A differenza dei diagrammi di interazione, che si focalizzano sugli eventi che influenzano il comportamento di un insieme di oggetti, i diagrammi di stato mettono a fuoco quale o quali attributi influenzano il comportamento di un singolo oggetto
- Usati per identificare attributi di oggetti e di rifinire tutte le descrizioni dei comportamenti di un oggetto
  - Invece, i diagrammi di interazione sono usati per identificare gli oggetti partecipanti ed i servizi che forniscono

Diagrammi delle attività

- Forniscono la sequenza di operazioni che definiscono un'attività più complessa
- Permettono di rappresentare processi paralleli e la loro sincronizzazione
- Possono essere considerati Diagrammi di stato particolari
  - Ogni stato contiene (è) un'azione
- Un diagramma delle attività può essere associato
  - A una classe
  - All'implementazione di un'operazione
  - Ad un caso d'uso

Esempio

- Un'azione è un'attività atomica: è rappresentata nel diagramma con un rettangolo arrotondato ed è identificata con una voce verbale che descrive l'azione stessa
  - *HandleIncident*: il *Dispatcher* riceve i rapporti e alloca risorse
  - *DocumentIncident*: tutti i *FieldOfficer* partecipanti e i *Dispatcher* documentano l'incidente dopo che è stato chiuso
  - *ArchiveIncident*: archiviazione delle informazioni relative all'incidente su dispositivi di memorizzazione



Diagrammi delle attività

- Ogni diagramma delle attività ha due nodi particolari: Inizio e Fine
  - Inizio è il punto di partenza del diagramma, indica la prima azione da eseguire ed è rappresentato da un cerchio con solo archi in uscita
  - Fine: indica la conclusione dello scenario descritto e ha solo archi in entrata. L'azione finale del diagramma deve puntare sempre al nodo finale

Diagrammi delle attività

- Un arco, o flusso, collega tra loro i nodi
  - L'insieme degli archi del diagramma rappresenta il flusso di esecuzione complessivo
  - E' rappresentato con una freccia
- Il flusso delle azioni progredisce solo nel momento in cui l'azione considerata è completata
- Una funzionalità complessa non è costituita da una semplice successione di azioni in sequenza: può presentare azioni da eseguire contemporaneamente o sotto particolari condizioni

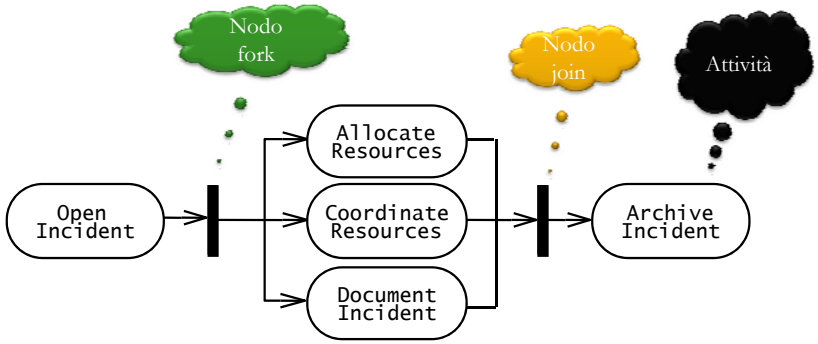
Nodi di controllo

- Fork
  - Descrive l'esecuzione in parallelo di più azioni: quelle puntate dal nodo fork sono avviate contemporaneamente ed in parallelo (rappresentato da una barra nera puntata da un solo arco e dalla quale partono due o più archi verso le azioni da eseguire in parallelo)
- Join
  - E' il duale del fork. Specifica che un'azione è eseguita solo nel momento in cui le azioni precedenti hanno terminato la propria esecuzione. E' definito anche sincronizzazione perché sincronizza due rami svolti parallelamente, generando un unico flusso di esecuzione. E' rappresentato da una barra nera

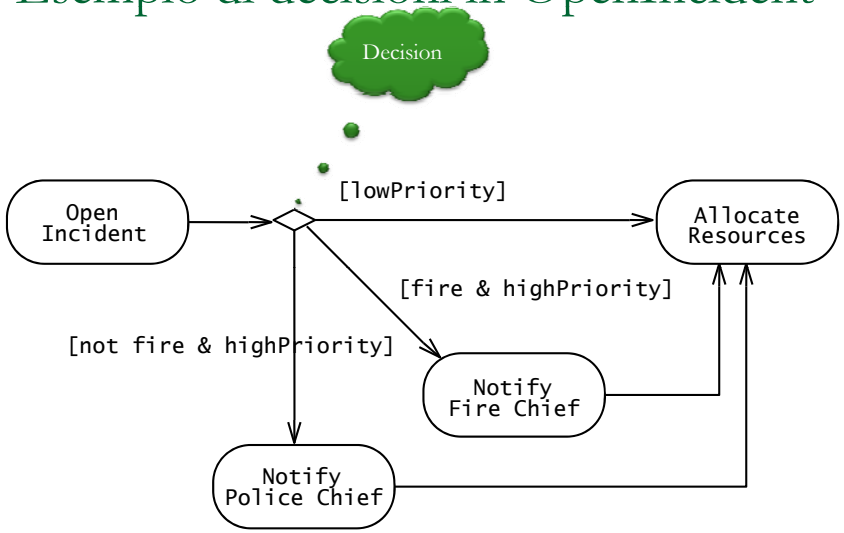
Nodi di controllo

- Decision
  - Indica che l'esecuzione di un'azione dipende dal verificarsi di una determinata condizione chiamata guard. Descrive il fatto che al termine di una particolare azione, lo scenario può proseguire in modo diverso con azioni che dipendono da specifiche situazioni che si vengono a verificare. E' rappresentato con un diamante puntato da un arco e dal quale escono almeno due archi.
- Merge
  - Duale del nodo decision e descrive un punto del processo in cui si ricongiungono due o più flussi alternativi. Rappresntato anch'esso da un diamante

Esempio di diagramma delle attività



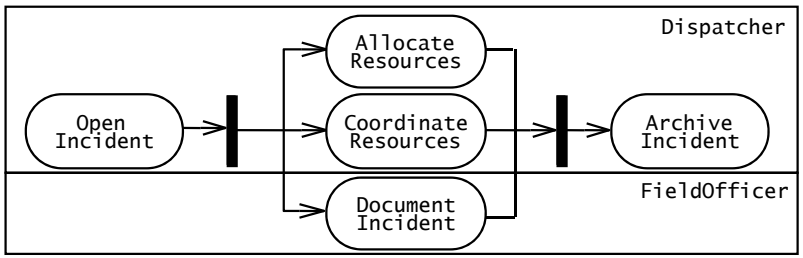
Esempio di decisioni in OpenIncident



Swimlane

- Le azioni possono essere raggruppate in ***swimlane*** per denotare l'oggetto o sottosistema che implementa le azioni
  - Le swimlane sono rappresentate come rettangoli che racchiudono un gruppo di azioni.
  - Le transizioni possono attraversare le swimlanes

Esempio di swimlanes



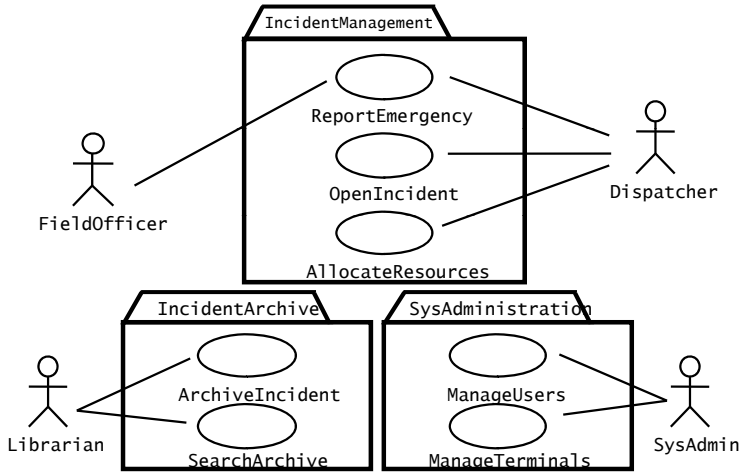
Applicazione dei diagrammi delle attività

- I diagrammi delle attività forniscono una visione task-centrica del comportamento di un insieme di oggetti
- Possono essere usati per descrivere vincoli di sequenza tra i casi d'uso, attività sequenziali tra gruppi di oggetti, o task di un progetto

Organizzazione dei diagrammi

- I modelli di sistemi complessi diventano velocemente anch'essi complessi quando gli sviluppatori li revisionano e rifiniscono
- La complessità dei modelli può essere gestita raggruppando elementi in relazione tra loro in **package**
- Un package è un raggruppamento di elementi, come casi d'uso, classi, o attività che semplificano la comprensione

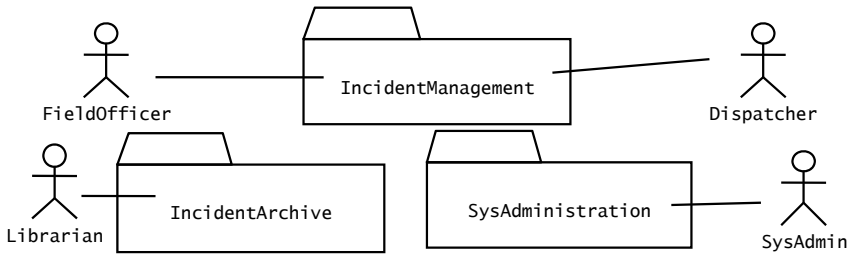
Esempio di package: casi d'uso di FRIEND organizzati per attori



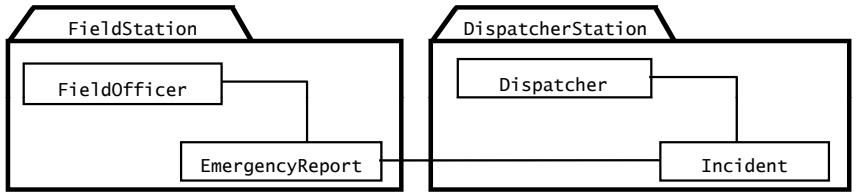
Package nei diagrammi delle classi

- Anche i diagrammi delle classi possono essere organizzati in package
- Le classi dal caso d'uso *ReportEmergency* sono organizzate in accordo al punto in cui gli oggetti sono creati
  - *FieldOfficer* e *EmergencyReport* sono parte del package FieldStation
  - *Dispatcher* e *Incident* sono parte di *DispatcherStation*

Esempio di package



Esempio di package



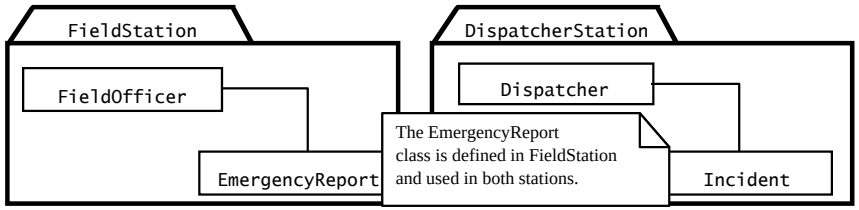
Ancora sui package

- I package possono essere usati per trattare la complessità alla stregua dell'organizzazione di file e directory di un generico utente
  - La differenza è che i package non sono necessariamente gerarchici: la stessa classe può trovarsi in più di un package
  - Per ridurre le inconsistenze, le classi sono possedute da esattamente un package e gli altri package fanno riferimento ad esso

Note

- Una **nota** è un commento allegato al diagramma
- Sono usate dagli sviluppatori per allegare informazioni ai modelli e agli elementi del modello
- E' un meccanismo per registrare questioni rilevanti di un modello, per chiarire un punto complesso e per tener traccia di un punto da svolgere in futuro

Esempio di nota. Le note possono essere allegate ad un elemento del diagramma specifico



Estensione dei diagrammi

- L'obiettivo di UML consiste nel fornire un insieme di notazioni per modellare un'ampia classe di sistemi software
- Tuttavia, un insieme prefissato di notazioni non potrebbe consentire di perseguire un tale scopo
  - È impossibile anticipare tutte le necessità che si verificano in tutti i domini applicativi
- UML fornisce un numero di meccanismi che consentono di estendere il linguaggio

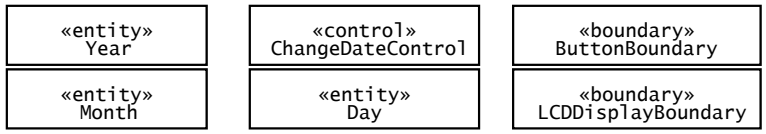
Stereotipi

- Uno **stereotipo** è un meccanismo di estensione che consente di classificare gli elementi del modello in UML
- E' rappresentato da una stringa racchiusa tra parentesi angolari <<>> e allegate all'elemento a cui si applica, come una classe o un'associazione
- Formalmente, uno stereotipo corrisponde, semanticamente, a creare una nuova classe nel meta-modello UML (ovvero, il modello che rappresenta i costrutti di UML)

Stereotipi

- Esempio: durante l'analisi, classifichiamo gli oggetti in tre categorie: *entity*, *boundary* e *control*
  - Hanno la stessa struttura ma scopi diversi
- Il linguaggio base di UML include solo un tipo di oggetto. Per rappresentare le tre categorie usiamo gli stereotipi `<<entity>>`, `<<boundary>>` e `<<control>>`

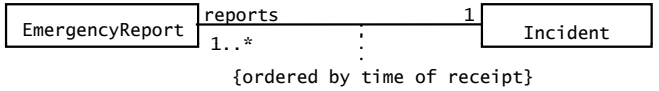
Esempio di stereotipi



Vincoli

- Un vincolo è una regola che viene allegata ad un elemento restringendone la semantica
- Consente di rappresentare fenomeni che non potrebbero essere espressi altrimenti in UML
- Esempio
  - Un incidente (*Incident*) può essere associato con uno o più *EmergencyReport* prodotti dal luogo dell'evento
  - I *Dispatcher* devono essere in grado di esaminare i rapporti in ordine cronologico
  - L'ordinamento cronologico degli *EmergencyReport* a *Incident* è espresso dal vincolo {*ordinato in base all'ora di ricezione*}
  - I vincoli possono essere espressi in linguaggio naturale o mediante OCL (Object Constraint Language)

Esempio di vincolo



Strumenti per UML

Programma	Descrizione	http
DIA	Gnome Visio-like diagram tool with aUML template	http://www.gnome.org/projects/dia/
Poseidon for UML community	UML diagrams, code generation for Java	http://www.gentleware.com/index.php?id=ce
ArgoUML	Open-source project, written in Java	http://argouml.tigris.org/
Umbrello	KDE-based open source written in C++	http://uml.sourceforge.net/index.php
Visual Paradigm for UML community	Free version with restrictions	http://www.visual-paradigm.com/product/vpuml/communityedition.jsp

Attività di sviluppo e relativi prodotti