

# Ingegneria del Software

## *Processi di Ingegneria dei requisiti (III)*

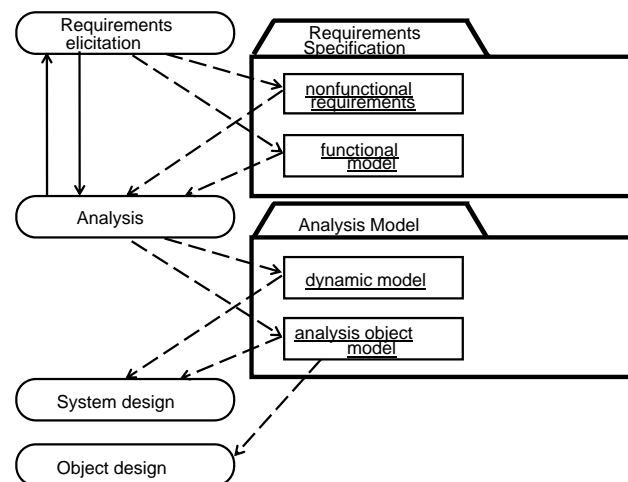
**Antonino Staiano**

e-mail: [antonino.staiano@uniparthenope.it](mailto:antonino.staiano@uniparthenope.it)

## Analisi

- L'obiettivo della fase di analisi è produrre un modello del sistema che sia corretto, completo, consistente e non ambiguo
- Gli sviluppatori formalizzano la specifica dei requisiti prodotta durante la fase di scoperta
  - esaminano più nel dettaglio le condizioni limite ed i casi eccezionali
- Il modello di analisi ed i requisiti non funzionali sono usati per predisporre l'architettura del sistema durante la fase di progettazione ad alto livello

## Prodotti delle fasi di scoperta e analisi



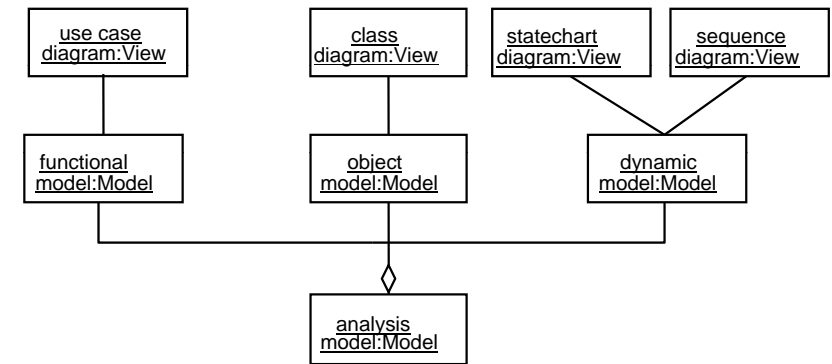
## Panoramica della fase di analisi

- L'analisi differisce dalla fase di scoperta
  - Gli sviluppatori si concentrano nel formalizzare e strutturare i requisiti scoperti a partire dagli utenti
- La formalizzazione fornisce informazioni aggiuntive e consente di individuare errori nei requisiti
  - Il modello di analisi potrebbe non essere compreso da utenti e clienti
  - Gli sviluppatori devono aggiornare la specifica dei requisiti per riflettere le informazioni aggiuntive e rivederla con clienti ed utenti
  - Alla fine del processo, i requisiti per quanto eventualmente numerosi, devono essere comprensibili a clienti ed utenti

## Modello di analisi

- Il modello di analisi è composto da tre singoli modelli
  - ❑ Il **modello funzionale**, rappresentato da casi d'uso e scenari
  - ❑ Il **modello ad oggetti di analisi**, rappresentato da diagrammi delle classi e degli oggetti
  - ❑ Il **modello dinamico**, rappresentato dai diagrammi di stato e delle sequenze

## Il modello di analisi



## Concetti di analisi

- Modelli ad oggetti di analisi e modelli dinamici
- Oggetti Entity, Boundary e Control
- Generalizzazione e specializzazione

## Concetti di analisi: **modelli ad oggetti di analisi** e modelli dinamici

- Il modello di analisi rappresenta il sistema in corso di sviluppo dal punto di vista dell'utente
- Il modello ad oggetti di analisi è parte del modello di analisi e focalizza i concetti manipolati dal sistema, le loro proprietà e le loro relazioni
- Il modello ad oggetti di analisi è descritto in UML dai diagrammi delle classi ed include classi, attributi e operazioni
  - ❑ E' un dizionario visuale dei concetti principali visibile all'utente

## Concetti di analisi: modelli ad oggetti di analisi e modelli dinamici

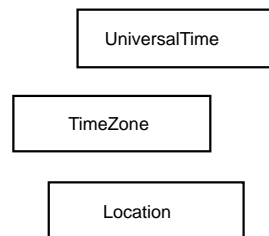
- Il modello dinamico è incentrato sul comportamento del sistema
  - Descritto da diagrammi delle sequenze e degli stati
- I diagrammi delle sequenze rappresentano le interazioni tra un insieme di oggetti durante un caso d'uso singolo
- I diagrammi degli stati rappresentano il comportamento di un singolo oggetto (o un gruppo di oggetti fortemente accoppiati)
  - Servono per assegnare le responsabilità a ciascuna classe e, nel processo, per identificare nuove classi, associazioni, e attributi da aggiungere al modello ad oggetti di analisi

## Concetti di analisi: modelli ad oggetti di analisi e modelli dinamici

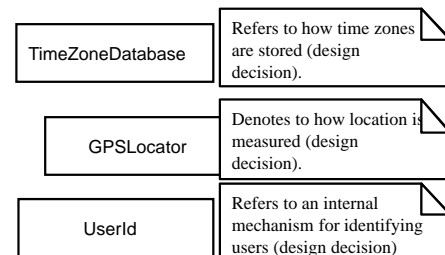
- E' da sottolineare che il modello ad oggetti di analisi ed il modello dinamico, rappresentano concetti a livello utente e non le classi o componenti effettivi del software
  - Ad esempio, concetti quali *Database*, *Subsystem*, *SessionManager*, *Network* non dovrebbero apparire nel modello di analisi
    - Gli utenti sono completamente schermati da quei concetti
  - Le classi di analisi dovrebbero essere viste come astrazioni ad alto livello che saranno dettagliate in seguito

## Esempi e contro esempi di classi nel modello ad oggetti di analisi di *SatWatch*

Concetti del dominio che dovrebbero essere rappresentati nel modello ad oggetti di analisi



Classi software che non dovrebbero essere rappresentate nel modello ad oggetti di analisi



## Oggetti Entity, Boundary e Control

- Il modello ad oggetti di analisi consiste di oggetti entity, boundary e control
  - Gli oggetti entity rappresentano l'informazione persistente della quale il sistema tiene traccia
  - Gli oggetti boundary rappresentano le interazioni tra tra gli attori del sistema
  - Gli oggetti control, realizzano i casi d'uso
  - Esempio *2Bwatch*
    - *Year*, *Month* e *Day* sono oggetti entità;
    - *Button* e *LCDDisplay* sono oggetti boundary
    - *ChangeDateControl* è un oggetto control che rappresenta l'attività del cambio data premendo una combinazione di pulsanti

## Oggetti Entity, Boundary e Control

- Gli oggetti entity, boundary e control forniscono semplici euristiche per distinguere concetti differenti ma collegati
  - Ad esempio, l'orario misurato da un orologio ha proprietà differenti rispetto al display che lo visualizza
    - Gli oggetti boundary e entity forzano la distinzione
      - L'orario misurato dall'orologio è rappresentato dall'oggetto *Time*
      - Il display è rappresentato da *LCDDisplay*
  - L'approccio basato sui tre tipi di oggetti da luogo ad oggetti più piccoli e specializzati

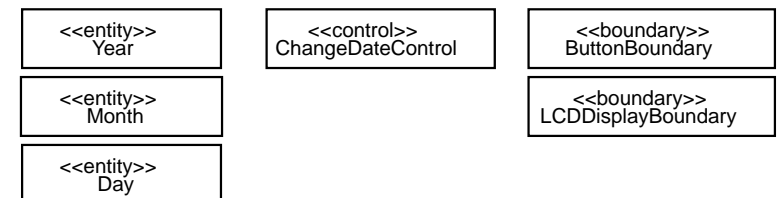
## Oggetti Entity, Boundary e Control

- L'approccio con tre tipi di oggetti è flessibile rispetto alle modifiche
  - E' più probabile che cambi l'interfaccia del sistema (gli oggetti boundary) rispetto alle funzionalità di base (oggetti entity e control)
- Separare l'interfaccia dalle funzionalità lascia gran parte del modello inalterata quando, ad esempio, cambia l'interfaccia ma non gli oggetti entità

## Stereotipi per entity, boundary e control

- Per distinguere gli oggetti entity, boundary e control, UML fornisce il meccanismo degli stereotipi
  - Ad esempio, possiamo allegare lo stereotipo `<<control>>` all'oggetto *ChangeDateControl*
  - E' possibile introdurre delle convenzioni sui nomi
    - Gli oggetti control, possono avere il suffisso Control nel proprio nome
    - Gli oggetti boundary possono avere un nome che denota una caratteristica di interfaccia (ad esempio, con un suffisso *Form*, *Button*, *Display*, etc.)
    - Gli oggetti entity non hanno alcun suffisso

## Classi di analisi per *2Bwatch*



## Generalizzazione e specializzazione

- L'ereditarietà consente di organizzare i concetti in gerarchie
  - In cima alla gerarchia c'è il concetto generale
  - In fondo alla gerarchia troviamo i concetti più specializzati
  - Nel mezzo possono esserci diversi livelli intermedi che rappresentano concetti più o meno specializzati
- Le gerarchie consentono di riferirsi a molti concetti in modo preciso. Ad esempio in FRIEND
  - Quando ci riferiamo ad un *Incident* intendiamo tutte le istanze del tipo *Incident*
  - Quando ci riferiamo a *Emergency* ci riferiamo solo ad un incidente che richiede una risposta immediata

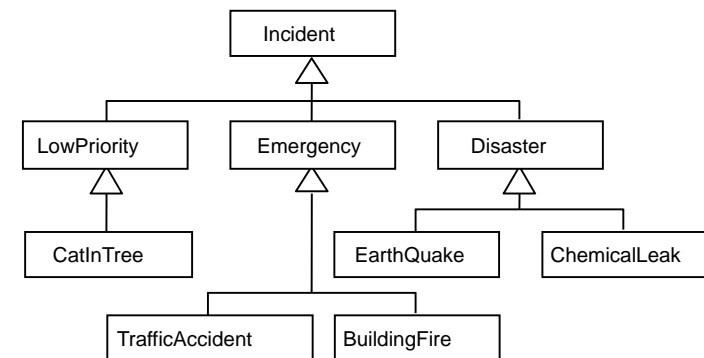
## Generalizzazione e specializzazione

- La generalizzazione è l'attività di modellazione che identifica i concetti astratti dai concetti di basso livello
- Esempio
  - Ingegnerizzando all'inverso (reverse engineering) un sistema di gestione emergenze, verifichiamo l'esistenza di schermi per gestire sinistri legati al traffico o ad incendi
    - Osservando le caratteristiche comuni tra i tre concetti, creiamo una classe astratta *Emergency* per descrivere le caratteristiche comuni (e generali) di incidenti legati al trasporto e agli incendi

## Generalizzazione e specializzazione

- La specializzazione è l'attività che identifica concetti più specializzati da quelli ad alto livello
  - Esempio: stiamo costruendo un sistema di gestione emergenze cominciando da zero e stiamo discutendo le funzionalità con il cliente
    - Il cliente ci introduce il concetto di incidente e poi descrive tre tipi di incidenti (*Incidents*)
      - Disastri (*Disasters*), che richiede la collaborazione di diverse agenzie
      - Emergenze (*Emergencies*), che richiede una risposta immediata ma può essere gestita da una singola agenzia
      - Incidenti con priorità bassa (*LowPriorityIncidents*), che non deve essere necessariamente gestita se le risorse sono richieste da altri incidenti con maggiore priorità

## Esempio di gerarchia di generalizzazione



## Attività di Analisi: dai casi d'uso agli oggetti

- Descriveremo le attività che trasformano i casi d'uso e gli scenari prodotti nella scoperta dei requisiti in un modello di analisi
  - Identificazione degli oggetti entity
  - Identificazione degli oggetti boundary
  - Identificazione degli oggetti Control
  - Mapping dai casi d'uso agli oggetti con i diagrammi delle sequenze
  - Identificazione delle associazioni
  - Identificazione degli aggregati
  - Identificazione degli attributi
  - Modellazione del comportamento dipendente dagli stati di un oggetto
  - Modellazione delle relazioni di ereditarietà
  - Revisione del modello di analisi

Ingegneria del Software, a.a. 2008/2009 – A. Staiano

## Identificare gli oggetti entity

- Gli oggetti partecipanti costituiscono la base del modello di analisi
- Gli oggetti partecipanti sono individuati esaminando ogni caso d'uso ed identificando gli oggetti candidati
- L'analisi basata sul linguaggio naturale fornisce un insieme di euristiche (Abbott, 1983) per identificare oggetti, attributi, e associazioni a partire da una specifica dei requisiti
  - Il linguaggio naturale prevede di mappare parti del parlato (sostantivi, verbi, etc.) in componenti del modello (oggetti, relazioni etc.)
  - Sebbene ci sia il vantaggio di mettere a fuoco i termini dell'utente, ci sono anche diverse limitazioni
    - Dipendenza dallo stile di scrittura dell'analista
    - Ci sono numerosi sostantivi rispetto alle classi rilevanti

Ingegneria del Software, a.a. 2008/2009 – A. Staiano

## Mapping parti del parlato ai componenti del modello

<b>Parte del parlato</b>	<b>Componente del modello</b>	<b>Esempio</b>
Nome proprio	oggetto	Alice
Nome comune	classe	Field officer
Verbo fare	metodo	Crea, seleziona
Verbo essere	ereditarietà	È-un (tipo-di)
Verbo avere	aggregazione	Ha, consiste
Verbo modale	vincolo	Deve essere
aggettivo	attributo	descrizione incidente

Ingegneria del Software, a.a. 2008/2009 – A. Staiano

## Euristiche per identificare gli oggetti entity

- E' possibile utilizzare le euristiche seguenti insieme con quelle di Abbott
  - Termini che gli sviluppatori o gli utenti devono chiarire per capire il caso d'uso
  - Sostantivi ricorrenti nei casi d'uso
  - Entità del mondo reale che il sistema deve considerare (FiledOfficer, Dispatcher, Resource)
  - Attività del mondo reale che il sistema deve considerare (EmergencyOperationPlan)
  - Sorgenti o contenitori di dati (es, Printer)

Ingegneria del Software, a.a. 2008/2009 – A. Staiano

## Identificare gli oggetti entity

- Gli sviluppatori nominano e descrivono brevemente gli oggetti, i loro attributi e responsabilità non appena li identificano
  - Dare nomi univoci invoglia l'uso di una terminologia standard
  - Per gli oggetti entity è raccomandato di usare sempre nomi usati dagli utenti finali e dagli specialisti del dominio applicativo
  - Descrivere gli oggetti, anche brevemente, consente di chiarire i concetti usati e limitare le incomprensioni
  - Non è necessario impiegare troppo tempo nel dettagliare oggetti e attributi poiché siamo in fase di analisi
  - Gli sviluppatori dovrebbero documentare attributi e responsabilità se non sono ovvi, altrimenti è sufficiente un nome ed una breve descrizione per ogni oggetto

Ingegneria del Software, a.a. 2008/2009 – A. Staiano

## Esempio

- Dopo un primo esame del caso d'uso ReportEmergency si impiegano la conoscenza del dominio applicativo e le interviste con gli utenti per identificare gli oggetti Dispatcher, EmergencyReport, FieldOfficer e Incident
  - E' da osservare che l'oggetto EmergencyReport non è menzionato esplicitamente nel caso d'uso ReportEmergency
    - Il passo 5 del caso d'uso si riferisce ai rapporti di emergenza come “informazioni sottomesse dal FieldOfficer”.
    - Dopo una revisione con il cliente si scopre che a questa in formazione ci si riferisce solitamente come “emergency report” e si decide quindi di chiamare il corrispondente oggetto EmergencyReport

Ingegneria del Software, a.a. 2008/2009 – A. Staiano

Nome	ReportEmergency
Attori Partecipanti	Iniziato dal FieldOfficer Comunica con il Dispatcher
Flusso eventi:	<ol style="list-style-type: none"><li>1. Il FieldOfficer attiva la funzione “Report Emergency” del terminale</li><li>2. FRIEND risponde presentando una form al FieldOfficer. <i>La form include un menu del tipo di emergenza (emergenza generale, incendio, trasporto) e la località, la descrizione dell'incidente, la richiesta di risorse, i campi dei materiali nocivi.</i></li><li>3. Il FieldOfficer riempie la form <i>specificando al minimo il tipo di emergenza e i campi di descrizione.</i> Il FieldOfficer descrive anche possibili risposte alla situazione di emergenza e <i>può richiedere risorse specifiche.</i> Appena finito Il FieldOfficer invia la form</li><li>4. FRIEND riceve la form e notifica al Dispatcher <i>con un finestra pop-up.</i></li><li>5. Il Dispatcher rivede le informazioni sottomesse dal FieldOfficer e crea un Incidente nel DB invocando il caso d'uso OpenIncident. <i>Tutte le informazioni contenute nella form del FieldOfficer sono incluse automaticamente nell'Incident. Il Dispatcher seleziona una risposta allocando le risorse all'Incidente (con il caso d'uso AllocateResources) e notifica il rapporto di emergenza inviando un breve messaggio al FieldOfficer.</i></li><li>6. FRIEND visualizza l'accettazione e la risposta selezionata al FieldOfficer</li></ol>
Condizioni di uscita	Il FieldOfficer riceve l'accettazione e la risposta selezionata

Ingegneria del Software, a.a. 2008/2009 – A. Staiano

## Oggetti partecipanti al caso d'uso ReportEmergency

<b>Dispatcher</b>	Ufficiale di Polizia che gestisce gli Incident. Un Dispatcher apre, documenta, e chiude gli incidenti in risposta ad un EmergencyReport e altre comunicazioni con i FieldOfficer. I Dispatcher sono identificati con numeri di badge.
<b>EmergencyReport</b>	Rapporto iniziale su un Incident da un FieldOfficer ad un Dispatcher. Un EmergencyReport solitamente innesca la creazione di un Incident dal Dispatcher. Un EmergencyReport è composto da un livello di emergenza, un tipo (fuoco, incidente stradale, altro), una località e una descrizione.
<b>FieldOfficer</b>	Ufficiale di Polizia o dei Vigili del fuoco in servizio. Un FieldOfficer può essere allocato al più ad un incidente alla volta. I FieldOfficer sono identificati dai numeri di badge.
<b>Incident</b>	Situazione che richiede l'attenzione di un FieldOfficer. Il rapporto su un Incident può essere inserito nel sistema da un FieldOfficer o chiunque altro esterno al sistema. Un Incident è composto da una descrizione, una risposta, uno stato (aperto, chiuso, documentato), una località, e un numero di FieldOfficer.

Ingegneria del Software, a.a. 2008/2009 – A. Staiano



## Esempio: Flusso di eventi

- Jim Smith (cliente) entra in un negozio con l'intenzione di comprare un giocattolo per la propria figlia che ha n anni
- Deve essere disponibile un aiuto in meno di un minuto
- Il proprietario del negozio consiglia il cliente. Il consiglio è dipende dall'età della bimba e gli attributi del giocattolo
- Il cliente seleziona un giocattolo pericoloso di un tipo non adatto alla bimba
- Il proprietario del negozio raccomanda una bambola

<b>Parte del parlato</b>	<b>Componente del modello</b>	<b>Esempio</b>
Nome proprio	oggetto	Jim Smith
Nome comune	classe	Giocattolo, bambola
Verbo fare	metodo	Compra, raccomanda
Verbo essere	ereditarietà	È un tipo-di
Verbo avere	aggregazione	Ha un
Verbo modale	vincolo	Deve essere
aggettivo	attributo	3 anni
Verbo transitivo	metodo	entra
Verbo intransitivo	Metodo (evento)	Dipende da

## Identificare gli oggetti boundary

- Gli oggetti boundary rappresentano l'interfaccia del sistema con gli attori
- In ciascun caso d'uso, ogni attore interagisce con almeno un oggetto boundary
- L'oggetto boundary raccoglie le informazioni dagli attori e le traducono in una forma che è possibile usare con ambo gli oggetti entity e control

## Euristiche per identificare gli oggetti boundary

- *Identificare i controlli dell'interfaccia utente di cui l'utente necessita per iniziare il caso d'uso (es., `ReportEmergencyButton`)*
- *Identificare le form con le quali l'utente immette dati nel sistema (es., `EmergencyReportForm`)*
- *Identificare avvisi e messaggi che il sistema usa per rispondere all'utente (`AcknowledgmentNotice`)*
- *Quando multipli attori sono coinvolti in un caso d'uso, identificare i terminali degli attori (es., `DispatcherStation`) per riferirsi all'interfaccia utente in questione*
- *Non modellare aspetti visuali dell'interfaccia con oggetti boundary*
- *Usare sempre i termini dell'utente finale per descrivere le interfacce; non usare termini dai domini implementativi*



## Oggetti boundary del caso d'uso *ReportEmergency*

<b>AcknowledgmentNotice</b>	Avviso usato per visualizzare l'accettazione del Dispatcher al FieldOfficer.
<b>DispatcherStation</b>	Computer usato dal Dispatcher.
<b>ReportEmergencyButton</b>	Pulsante usato dal FieldOfficer per iniziare il caso d'uso ReportEmergency
<b>EmergencyReportForm</b>	Form usata per inserire il ReportEmergency. Questa form è presentata al FieldOfficer sulla FieldOfficerStation quando è selezionata la funzione "Report Emergency". La EmergencyReportForm contiene i campi per specificare tutti gli attributi di un rapporto di emergenza e un pulsante (o altro controllo) per sottomettere la form completa
<b>FieldOfficerStation</b>	Computer portatile usato dal FieldOfficer
<b>IncidentForm</b>	Form usata per la creazione di Incidenti. Questa form è presentata al Dispatcher sul DispatcherStation quando l'EmergencyReport è ricevuto. Il Dispatcher usa anche questa form per allocare le risorse a per accettare il rapporto del FieldOfficer

Ingegneria del Software, a.a. 2008/2009 – A. Staiano

## Oggetti boundary del caso d'uso *ReportEmergency*

- *IncidentForm* non è menzionato esplicitamente nel caso d'uso *ReportEmergency*
  - L'oggetto è stato identificato osservando che il Dispatcher ha bisogno di una interfaccia per vedere il rapporto di emergenza sottomesso dal FieldOfficer e per spedire un'accettazione
- I termini usati per descrivere gli oggetti boundary nel modello di analisi dovrebbero seguire la terminologia dell'utente

Ingegneria del Software, a.a. 2008/2009 – A. Staiano

## Identificare gli oggetti control

- Gli oggetti control sono responsabili per coordinare gli oggetti boundary ed entity
  - Solitamente non hanno una controparte concreta nel mondo reale
  - Spesso esiste una relazione stretta tra un caso d'uso e un oggetto control
  - Un oggetto control solitamente è creato all'inizio di un caso d'uso e cessa di esistere alla fine
  - E' responsabile di raccogliere informazioni dagli oggetti boundary agli oggetti entity

Ingegneria del Software, a.a. 2008/2009 – A. Staiano

## Identificare gli oggetti control

- Inizialmente, si modella il flusso di controllo del caso d'uso *ReportEmergency* con un oggetto control per ogni attore
  - *ReportEmergencyControl* per il *FieldOfficer* e *Manage-Emergency-Control* per il *Dispatcher*
- La decisione di modellare il flusso di controllo del caso d'uso *ReportEmergency* con due oggetti control scaturisce sapendo che *FieldOfficerStation* e *DispatcherStation* sono due sottosistemi che comunicano su un link asincrono
  - Rendendo questo concetto visibile nel modello di analisi consente di mettere a fuoco comportamenti eccezionali come la perdita di comunicazione tra ambo le stazioni

Ingegneria del Software, a.a. 2008/2009 – A. Staiano

## Oggetti control del caso d'uso *EmergencyReport*

<b>ReportEmergencyControl</b>	Gestisce la funzione di reporting del ReportEmergency sulla FieldOfficerStation. Questo oggetto è creato quando il FieldOfficer seleziona il pulsante "Report Emergency". Esso dopo crea una EmergencyReportForm e la presenta al FieldOfficer. Dopo aver sottomesso la form, questo oggetto dopo raccoglie le informazioni dalla form, crea un EmergencyReport, e lo invia al Dispatcher. L'oggetto control dopo aspetta un'accettazione proveniente dalla DispatcherStation. Quando è ricevuta l'accettazione, l'oggetto ReportEmergencyControl crea un AcknowledgeNotice e la visualizza al FieldOfficer.
<b>ManageEmergencyControl</b>	Gestisce la funzione di reporting del ReportEmergency sulla DispatcherStation. Questo oggetto è creato quando è ricevuto un EmergencyReport. Esso dopo crea una IncidentForm e la visualizza al Dispatcher. Una volta che il Dispatcher ha creato un Incident, allocato Resource e sottomesso un'accettazione, ManageEmergencyControl invia l'accettazione alla FieldOfficerStation

Ingegneria del Software, a.a. 2008/2009 – A. Staiano

## Euristiche per identificare gli oggetti control

- *Identificare un oggetto control per caso d'uso*
- *Identificare un oggetto control per attore nel caso d'uso*
- *La durata di vita di un oggetto control dovrebbe coprire l'estensione del caso d'uso o l'estensione di una sessione utente. E' difficile identificare l'inizio e la fine dell'attivazione di un oggetto control, il caso d'uso corrispondente probabilmente non ha condizioni di entrata e di uscita ben definite*

Ingegneria del Software, a.a. 2008/2009 – A. Staiano

## Identificare gli oggetti control

- Nel modellare il caso d'uso ReportEmergency, è stata modellata la stessa funzionalità usando gli oggetti entity, boundary e control
- Spostandosi dalla prospettiva del flusso di eventi ad una prospettiva strutturale, si è incrementato il livello di dettaglio della descrizione e selezionato termini standard per riferirsi alle entità principali del dominio applicativo e al sistema

Ingegneria del Software, a.a. 2008/2009 – A. Staiano