UML2

Attività di Progettazione

Andrea Polini

Ingegneria del Software
Corso di Laurea in Informatica

Attività di Progettazione

La progettazione riguarda quelle attività che mirano a derivare un modello di progetto di dettaglio sufficiente a poter essere utilizzato per le attività di implementazione.

Tali attività diventano predominanti nelle ultime iterazioni della fase di elaborazione e nelle prime della fase di costruzione.

Obiettivo fondamentale è il raffinamento dei diagrammi di analisi al fine di aggiungere informazioni rilevanti e che tengano in considerazione anche possibili scelte derivanti da strumenti, linguaggi, framework di sviluppo utilizzati.

Modello di progettazione

Il modello di progettazione <<traccia>> il modello di analisi

- sottosistemi di progettazione (tracciano i package di analisi)
- classi di progettazione (tracciano le classi di analisi)
- interfacce
- progettazione e realizzazione dei casi d'uso
- diagramma di deployment

Le classi di progettazione contengono tutte le informazioni ed ornamenti che si ritengono necessari per poter sviluppare la classe e.g. le segnature sono precisamente definite.

Gestione dei modelli di analisi e di progettazione

Possibili scelte:

- trasformare modello di analisi in quello di progettazione
- trasformare e poi usare strumenti per la produzione di una vista di analisi
- congelare modello di analisi. Usare copia per derivare modello di progettazione
- mantenere i due modelli allineati

Gestione dei modelli di analisi e di progettazione

È necessario manutenere i modelli di analisi e di progettazione contemporaneamente? Vanno tenuti allineati?

Perchè un modello di analisi può essere utile:

- introdurre nuove risorse sul progetto
- comprensibilità del progetto
- tracciamento requisiti e funzionalità implementate
- interventi di manutenzione
- comprensione dell'architettura logica
- outsourcing dello sviluppo

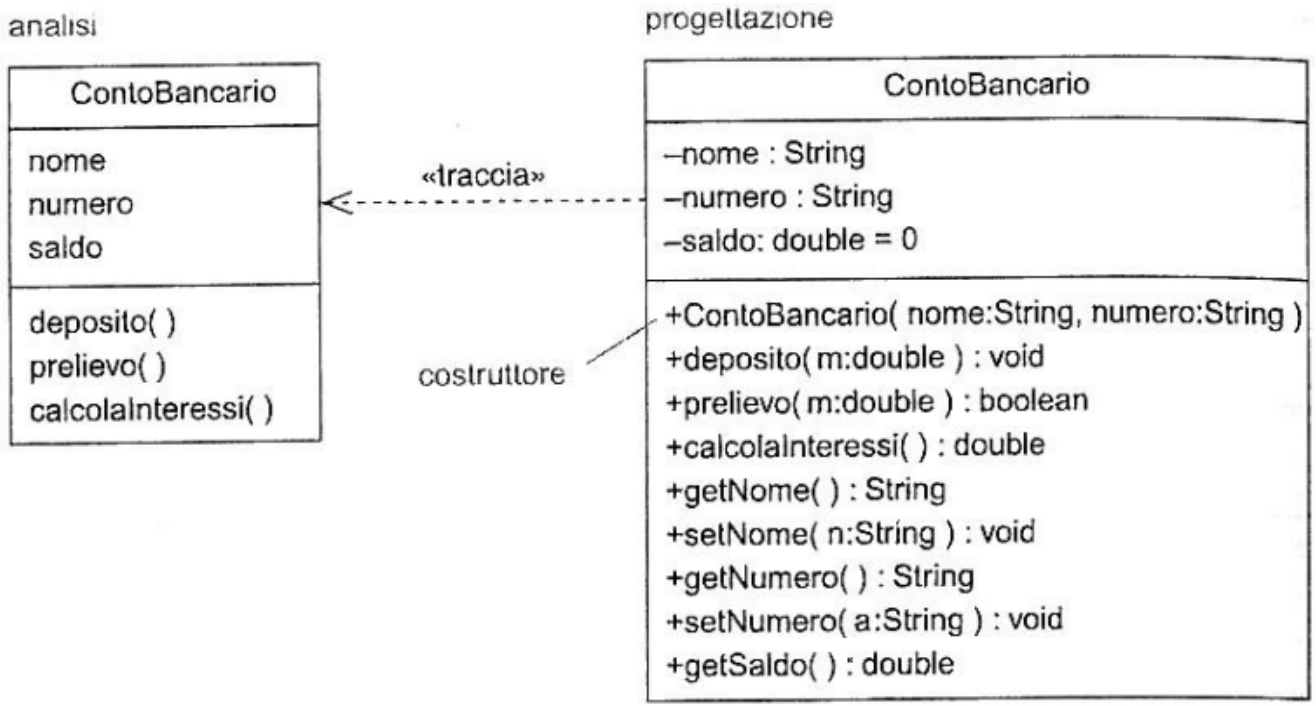
Classi di progettazione

Le classi di progettazione evolvono durante le attività di progettazione e le iterazioni in cui tali attività sono poste in essere. L'evoluzione viene interrotta quando si ritiene di avere una quantità di informazioni tale per cui si può passare all'implementazione. In un certo momento il modello di progettazione conterrà classi a diversi stati di "maturazione"

Origine delle classi di progettazione:

- dominio del problema
- dominio della soluzione: middleware, framework di sviluppo, framework per sviluppo GUI, Design Patterns, librerie di classi ed utilità

Anatomia delle classi di progettazione

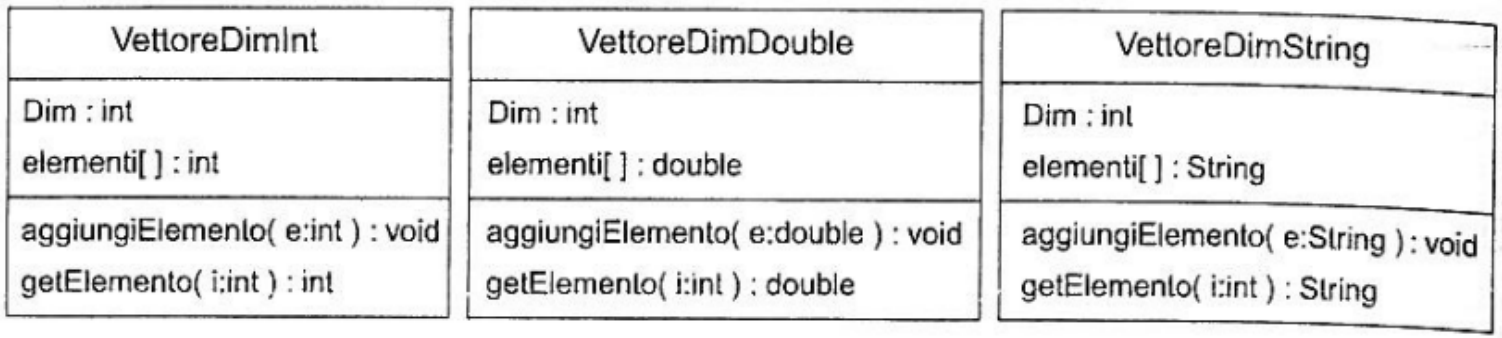


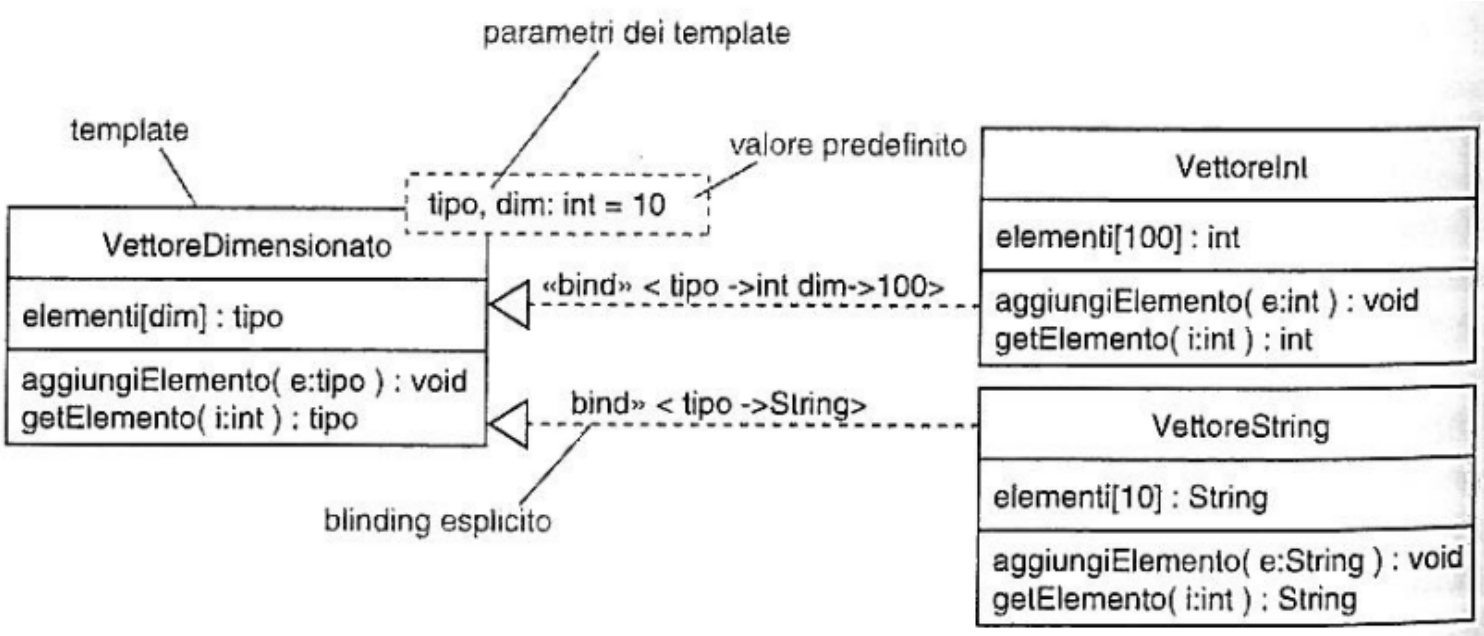
- Completezza e Sufficienza
- Essenzialità
- Massima coesione
- Minima interdipendenza

Ereditarietà

- Aggregazione o ereditarietà?
- Ereditarietà multipla
- Ereditarietà o interfacce

Classi template





Generics in Java

```
public class Entry<K, V> {

   private final K key;
   private final V value;

   public Entry(K k,V v) { key = k; value = v; }

   public K getKey() { return key; }

   public V getValue() { return value; }

   public String toString() {
      return "(" + key + ", " + value + ")";
   }
}
```

Classi annidate

Rappresentate con una relazione di contenimento

Visibile solo alla classe contenitore tipicamente molto usate nella gestione degli eventi per i framework di gestione delle interfacce.

Raffinamento delle relazioni di analisi

Durante la fase di analisi tipicamente le relazioni vengono generalmente soltanto identificate (associazioni) ma non necessariamente dettagliate (aggregazione, composizione)

Molte associazioni definite nella fase di analisi non sono supportate dai linguaggi di programmazione:

- associazioni bidirezionali
- classi associazioni
- associazioni molti a molti

Raffinamento delle relazioni di analisi

Nella progettazione si opera sulle relazioni di analisi al fine di:

- trasformare le associazioni in relazioni di aggregazione o composizione
- implementare le classi associazione
- implementare le associazioni uno ad uno
- implementare le associazioni molti ad uno
- implementare le associazioni molti a molti
- implementare le associazioni bidirezionali

Le associazioni di progettazione dovranno:

- specificare la navigabilità
- specificare la molteplicità di entrambi gli estremi

Aggregazione vs. Composizione

- Aggregazione: *è una relazione del tipo tutto/parte dove le parti mostrano comunque una loro indipendenza e possono essere parti di più entità di aggregazione*
- Composizione: *è una relazione molto forte di tipo tutto/parte dove le parti non possono esistere se non come componenti di una sola entità più complessa*

Sono entrambe relazioni transitive ed asimmetriche

Composizione

Relazione estremamente forte che comporta:

- componente appartiene ad un solo composto
- composto è responsabile dei suoi componenti per i quali funge da meccanismo di accesso
- creazione e distruzione di un componente avviene contestualmente ad un composito
- componenti possono essere passati ad altri compositi che ne assumono la responsabilità

composizione vs. attributo

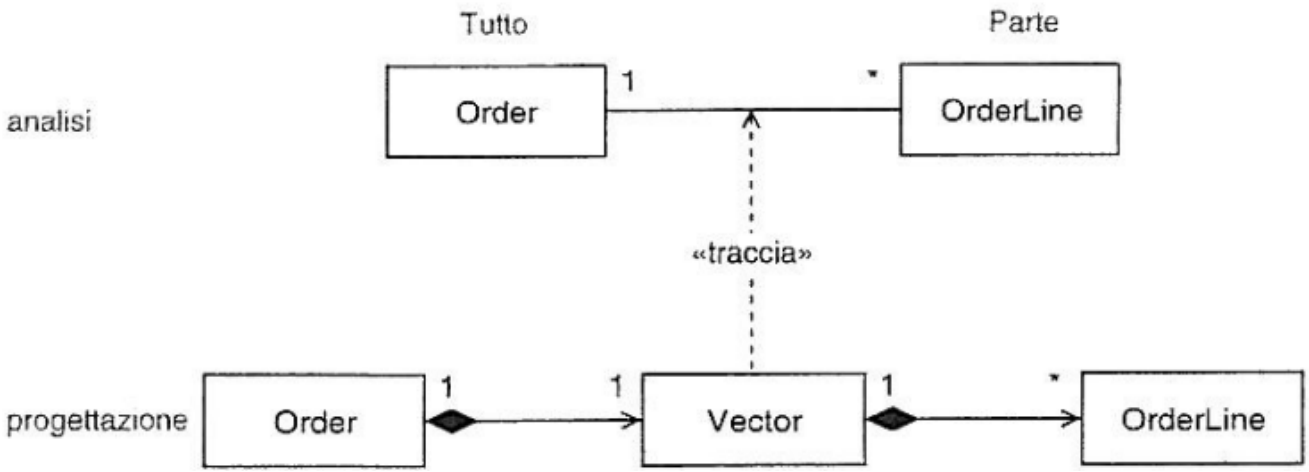
Raffinamento delle relazioni di analisi

In un diagramma di progetto l'unico caso in cui è lecito utilizzare un'associazione è il caso di un possibile ciclo nel grafo delle aggregazioni

- aggiunta molteplicità e nomi associazione/ruoli
- decidere quale estremo è la parte e quale il tutto
- aggiungere navigalità da tutto alla parte

Associazioni uno ad uno $\Longrightarrow$ composizione (o attributo)
Associazioni molti ad uno $\Longrightarrow$ aggregazione
Associazioni uno a molti $\Longrightarrow$ uso di classi collezione

classi collezione



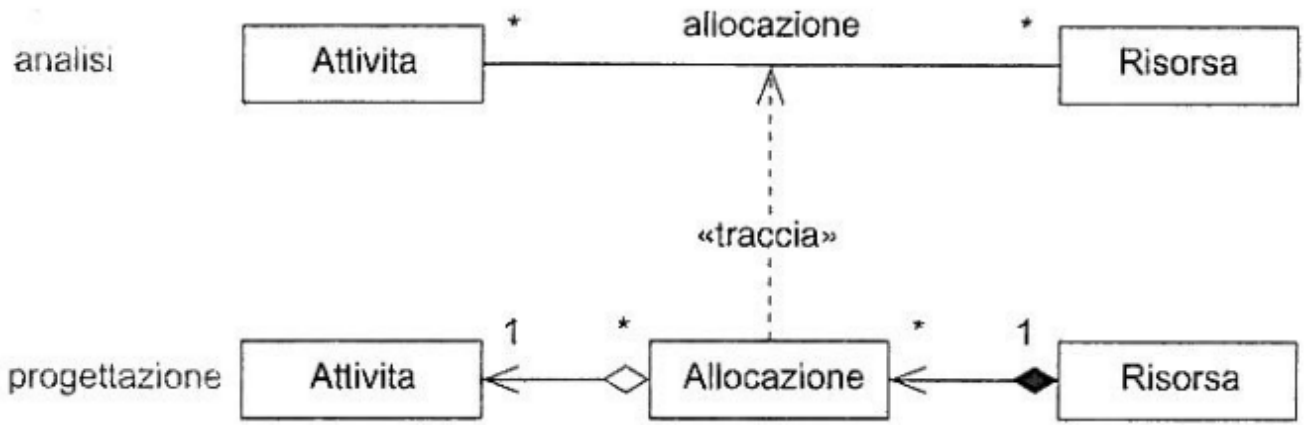
Strategie di specifica per classi collezione:

- specifica esplicita
- uso di valori etichettati
- aggiunta di proprietà alla relazione
  - *ordinata*
  - *non-ordinata*
  - *univoca*
  - *non-univoca*
- scelta lasciata ai programmatori

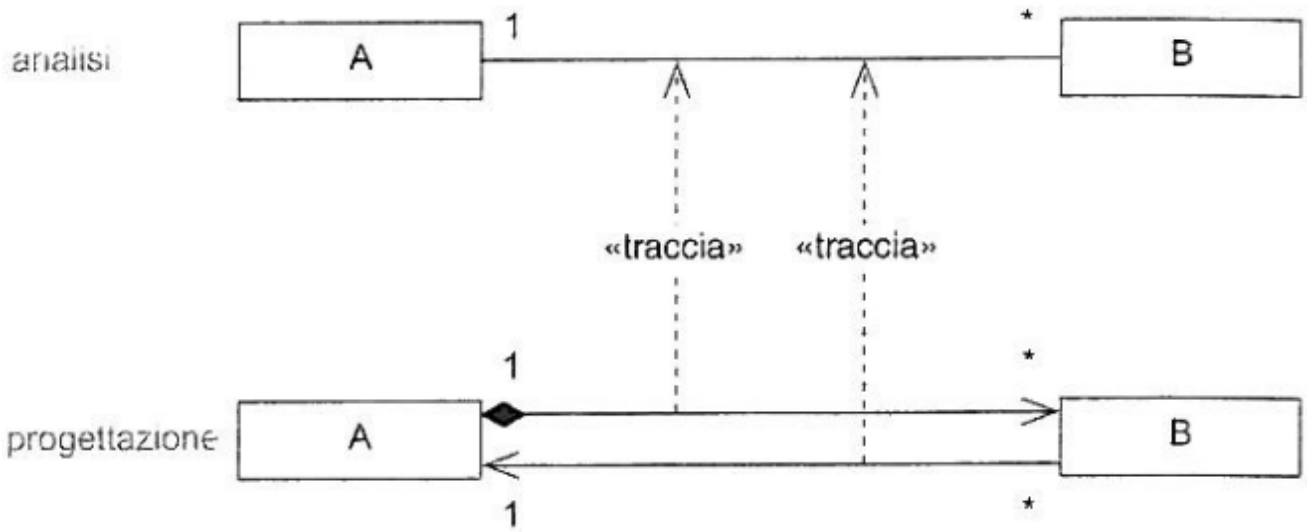
Relazioni Reificate

- associazioni molti a molti
- associazioni bidirezionali
- classi associazione

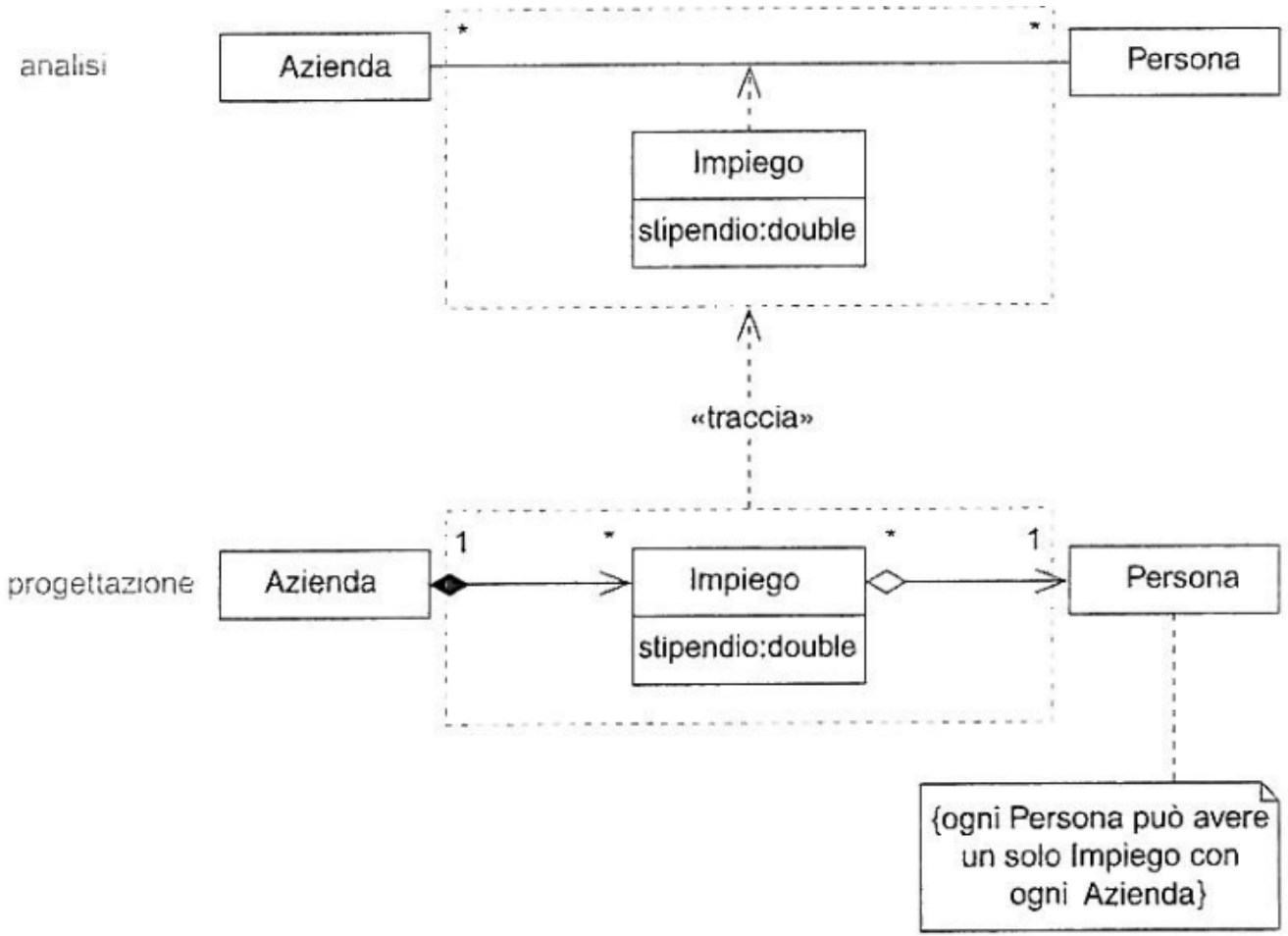
Relazioni Reificate – associazione molti a molti



Relazioni Reificate – associazioni bidirezionali



Relazioni Reificate – classi associazione



ATTENZIONE: vincolo sull'unicità della coppia viene perso

Modellazione della Concorrenza

Progettare la “Concorrenza” significa stabilire come e quando più parti del sistema possano procedere in parallelo. La concorrenza è un aspetto sempre più rilevante ma estremamente complesso.

Legge di Murphy

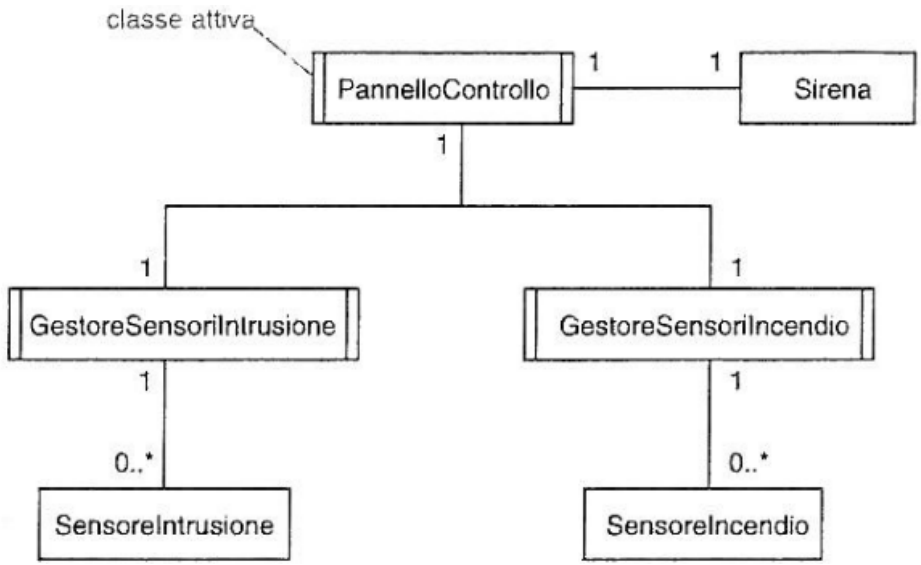
Se qualcosa può andar storto . . . ??

UML 2 mette a disposizione diversi strumenti per la progettazione della concorrenza che comunque è un aspetto che non dovrebbe far parte del modello di analisi

- classi attive
- biforcazioni e ricongiunzioni nei diagrammi di attività
- operatore `par` dei diagrammi di sequenza
- numero di sequenza nei diagrammi di comunicazione
- origini multiple nei diagrammi di temporizzazione
- stati compositi ortogonali nelle macchine a stati

Classi Attive

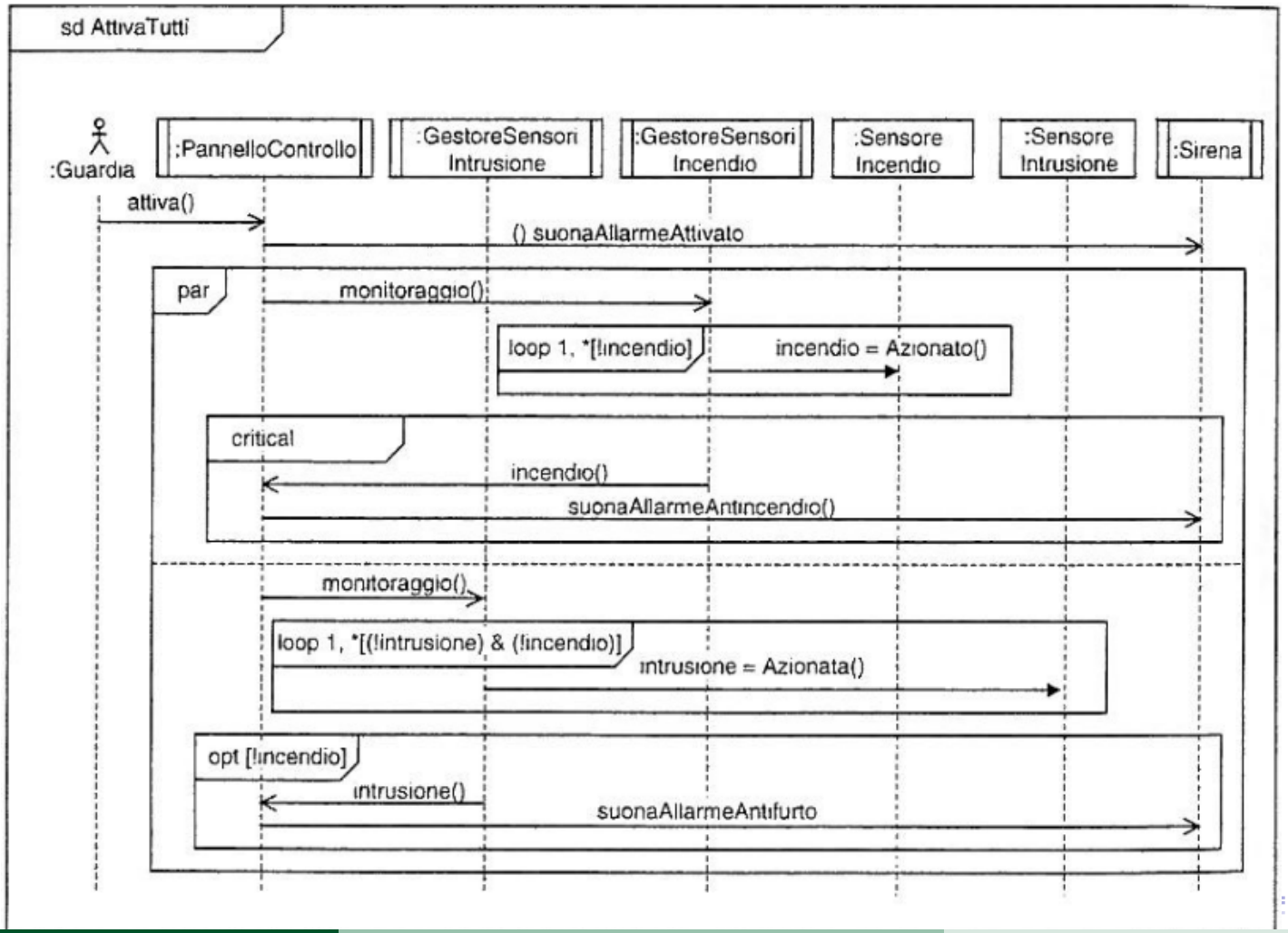
Le classi attive permettono di rappresentare classi che hanno autonomia di comportamento. Meccanismo dei *Thread*. Esempio di un sistema di sicurezza per il controllo di incendi ed intrusioni.



Sono necessarie in particolare nella progettazione dei sistemi embedded dove il software lavora principalmente con risorse limitate e controlla, fa interagire dispositivi hardware. In tal caso le componenti hardware sono una buona sorgente per l'identificazione di classi di progettazione.

Concorrenza nei diagrammi di sequenza

Diagramma di sequenza per il sistema di sicurezza di attivazione dei sensori:



Versione con molti sensori