UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II
**Facoltà di Ingegneria**
**Corso di Studi in Ingegneria Informatica**



tesi di laurea

# Modelli e strumenti per la generazione automatica di codice

Anno Accademico

## 2005-2006

**relatore**
Ch.mo prof. Porfirio Tramontana

**candidato**
Valerio Lombardi
Matr. 534/237

# Contesto e Contributo

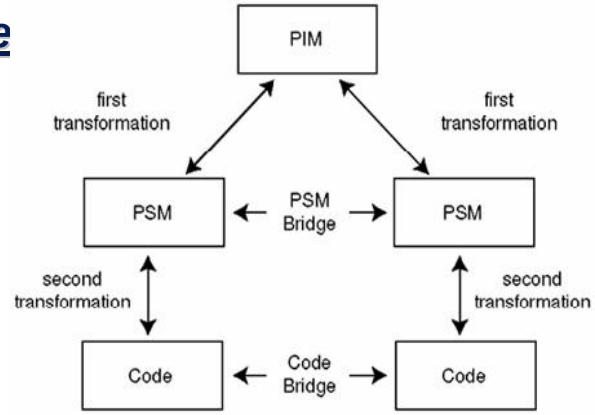
- **Fusione tra il mondo della modellazione e della programmazione nello sviluppo del software**

- **Scopo Tesi:**
  - **Studio dell'approccio Model Driven Architecture per la generazione di applicazioni attraverso trasformazione di modelli**
  - **Analisi dei plugin open source di Eclipse**
    - **Eclipse Modeling Framework**
    - **Graphical Editing Framework**
    - **Graphical Modeling Framewok**
  - **Generazione automatica codice attraverso EMF e valutazione dei pregi e difetti del plugin**

# Introduzione al problema

## *OMG - Model Driven Architrecture*

- **Il concetto chiave è la Modellazione**




- **Esistenza di due tipi di modelli descritti in UML:**
  - **Platform Independent Modeling (PIM)**
  - **Platform Specific Modeling (PSM)**
- **L'automazione della trasformazione tra PIM e PSM permette di passare da un livello d'astrazione molto alto, ad un livello molto vicino all'implementazione**
- **Legame molto stretto tra documentazione e codice**
- **Divisione modelli →interoperabilità & portabilità**

# Strumenti per la generazione automatica di codice

## Caratteristiche dei MDA-support tools

| MDA tools | Tool vendor | Model to Model | Model to Code | Availability | Support for (PSM) |
|---|---|---|---|---|---|
| **Optimalj** | *Compuware* | *X* | *X* | *Developer Edition $1900*<br>*Professional Edition $5000*<br>*Architecture Edition $10000* | *J2EE* |
| **OlivaNova Model Execution System** | *Sosy Inc.* | | *X* | *OlivaNova MES* | *.NET*<br>*COM*<br>*EJB* |
| **Together** | *Borland* | *X* | *X* | *for Visual Studio $300*<br>*for JBuilder X $ 700*<br>*Developer for Eclipse $ 1000*<br>*Designer for Eclipse $1500* | *J2EE*<br>*.NET* |
| **Kennedy Carter** | *Kennedy Carter* | | *X* | *Kennedy Carter* | *EJB*<br>*.NET* |
| **Eclipse Modeling Framework** | *IBM* | | *X* | *EMF 2.2.0 Edition Open Source* | *J2EE* |

- **La maggioranza dei tool MDA supporta principalmente la trasformazione Model to Code, sono prodotti commerciali e per J2EE e .NET**

# Eclipse's Plugin EMF – GEF – GMF

1) **Eclipse Modeling Framework: automaticamente genera codice e un editor per un modello definito dall'utente**

2) **Graphical Editing Framework: open source per lo sviluppo di rappresentazioni grafiche di modelli esistenti, è basato sul pattern MVC**
   - **Non genera codice**
   - **Utilizza codice java scritto e funzionante**
   - **Non può importare altri tipi di file come UML**

3) **Graphical Modeling Framework: open source per lo sviluppo di editor grafici di modelli**
   - **Partendo da un modello di dominio ottenuto attraverso EMF, esegue un mapping guidato di modelli intermedi, per arrivare alla generazione automatica dell'editor**
   - **Rispetto a GEF possiede una serie di tool e comandi più avanzati**


# Perché Eclipse Modeling Framework ?

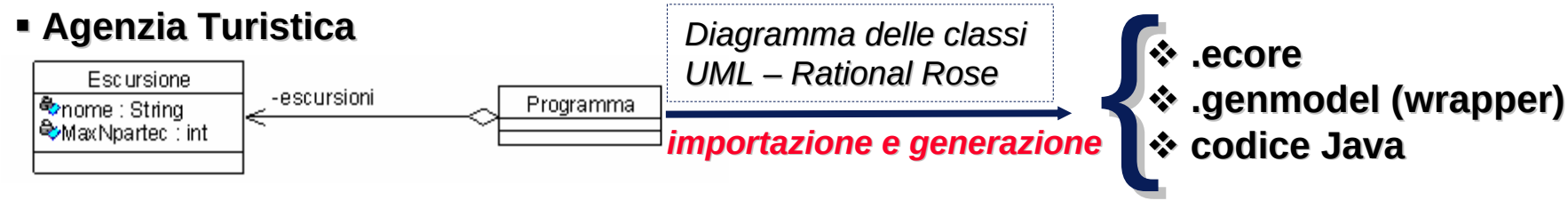
- **Eclipse fornisce:**
  - **una piattaforma open source**
  - **un'architettura estensibile sui plugin**
  - **disponibilità per un vasto range di sistemi operativi**

- **EMF permette:**
  - **Generazione automatica del codice Java a partire dal modello**
    - **Importazione XML Schema, UML(es. Rational Rose), Java Annotated**
  - **Modifiche al codice generato mantenendo aggiornato il modello**
  - **Supporto XMI/XML per la (de)serializzazione dei dati, permettendo la persistenza degli oggetti**

# MDA tool - Eclipse Modeling Framework

## *Caso di studio*

- **Agenzia Turistica**




- **Le interfacce generate (Escursione) contengono metodi accessori che consentono l'accesso o la modifica del valore di ogni attributo e relazione:**

```
public interface Escursione extends EObject {
        String getNome();
        void setNome(String value);
        int getMaxNpartec();
        void setMaxNpartec(int value);
         }
```

- **Escursione non ha classe base esplicita**
- **EObject equivalente ECore di java.lang.Object**
- **Metodi accessori get() e set() per attributi semplici**

- **Per l'interfaccia di Programma invece:**

```
public interface Programma extends EObject {
        EList<Escursione> getEscursioni();
        }
```

- **EList equivalente Ecore di list supporta i template**

# MDA tool - Eclipse Modeling Framework

## *Caso di studio – implementazione delle classi*

- **Il generatore di codice produce anche un' implementazione delle classi:**

```
protected static final String NOME_EDEFAULT = null;
protected String nome = NOME_EDEFAULT;

public String getNome() {
                return nome;
        }

public void setNome(String newNome) {
                String oldNome = nome;
                nome = newNome;
                if (eNotificationRequired())
                        eNotify(new ENotificationImpl(this, Notification.SET,
                        progettotesiPackage.ESCURSIONE__NOME,
                        oldNome, nome));
        }
```

- **Se il valore di default non è specificato nel modello, EMF seleziona un valore adatto al tipo**
- **Il metodo get() restituisce una variabile che rappresenta l'attributo**
- **Settaggio della variabile nome**

- **Il metodo set deve trasmettere la notifica automatica agli observers dell'oggetto in caso di modifica attraverso il metodo eNotify().**

# MDA tool - Eclipse Modeling Framework

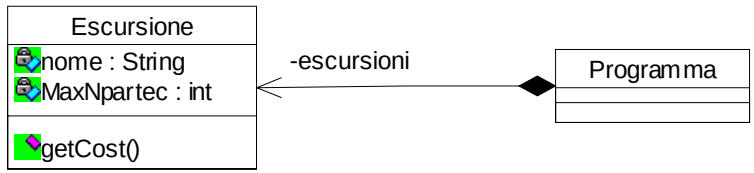
## *Caso di studio – relazioni e operazioni*

- **L'interfaccia e l'implementazione della classe generati contengono almeno un metodo accessorio per ciascuna relazione:**

```
protected EList<Escursione> escursioni = null;

public EList<Escursione> getEscursioni() {
      if (escursioni == null) {
          escursioni = new EObjectResolvingEList
          <Escursione>(Escursione.class, this, …);
                                }
                                return escursioni;
              }
```

- **Il metodo getEscurione() fa ritornare una speciale classe di implementazione EObjectResolvingEList**




- **Nel caso di composizione: EObjectContainmentEList**

- **Per il metodo getCost() EMF genera uno stub che lancia un eccezione:**

```
/**
 * @generated
 */
public double getCost() {
// TODO: implement this method
              throw new UnsupportedOperationException();
              }
```

- **Modificando il metodo dobbiamo cancellare @generated per non perdere il lavoro in seguito ad una rigenerazione anche parziale del modello**

- **EMF genera anche altre classi d' implementazione: una factory e un package**

# Conclusioni e sviluppi futuri

- **Qualità riscontrate nell'utilizzo di Eclipse Modeling Framewok:**
  - **buona modularità (separazione interfaccia e implementazione delle classi)**
  - **EMF patterns per la generazione del codice Java**
  - **modifiche al codice Java generato mantenendo aggiornato il modello**
  - **alta portabilità**
- **Limiti nell'utilizzo di EMF:**
  - **il codice automaticamente generato è unicamente Java**
  - **no repository-concept**
  - **no vincoli OCL**

- **In futuro si prevede sempre più integrazione tra la fase di design e implementazione, automatizzando le operazioni più noiose e ripetitive e quindi anche incline all'errore da parte del programmatore.**