Ingegneria del Software L-A

Corso di Laurea Triennale in
Ingegneria Informatica
III anno – A.A. 2009/2010

Docente: Giuseppe Bellavia
Collaboratore: Gabriele Zannoni

Premessa

Una domanda fondamentale

- Che cosa significa scrivere del buon software?

- Risposta del programmatore C:

  "Scrivere del buon software significa ottimizzare ogni istruzione, in modo da ottenere il codice più compatto ed efficiente possibile"

- Risposta del programmatore Visual Basic:

  "Scrivere del buon software significa fornire le funzionalità richieste dall'utente nel minor tempo possibile e col minor costo possibile, indipendentemente da come si arriva al risultato"

Risposta dell'ingegnere del software

Scrivere del buon software significa

- Trovare il miglior equilibrio fra diversi fattori:
  - La soddisfazione dell'utente
  - La facilità di estensione dell'applicazione
  - La comprensibilità delle soluzioni adottate
- Adottare tecniche adeguate a gestire la crescente complessità delle applicazioni

Risposta dell'ingegnere del software

Scrivere del buon software significa

- Utilizzare al meglio l'investimento, spesso ingente, necessario per produrre un'applicazione, garantendo in particolare:
  - Il maggior tempo di vita possibile
  - Il riutilizzo in altri progetti di parte del codice prodotto

Ingegnere del Software

Compiti principali

- Affrontare in modo sistematico e misurabile:
  - il progetto
  - la realizzazione
  - l'utilizzo
  - la manutenzione

  dei prodotti software
- Studiare le strategie per realizzare il punto precedente

Ingegneria del Software

Di cosa si occupa

- ✓ Gestione del processo di sviluppo del software
- ✓ Attività di analisi
- ✓ Attività di progettazione
- ✓ Attività di codifica
- ✓ Attività di verifica e convalida (testing)
- Attività di tipo gestionale
  - Stime dei costi (e dei tempi)
  - Gestione dei progetti (delle persone, pianificazione)
  - Gestione dei rischi
  - Gestione della qualità
- Metriche

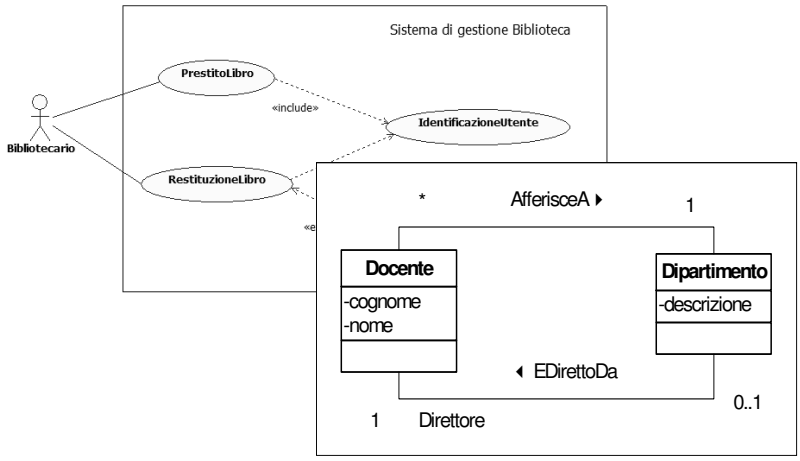
Obiettivi del corso

- Fornire i concetti di base dell'ingegneria del software
- Esaminare tecniche *object-oriented* per l'analisi, il progetto e la realizzazione di applicazioni
- Fornire nozioni di base su
  - Ambiente .NET
  - Linguaggio C#

Linguaggio di modellazione

- Durante il processo di sviluppo del software è indispensabile poter utilizzare un linguaggio per costruire i modelli da discutere con il cliente e gli altri sviluppatori
- Il linguaggio dovrebbe
  - essere visuale (una figura è meglio di mille parole)
  - possedere capacità dichiarative
- UML - *Unified Modeling Language*

Linguaggio di modellazione



Design Pattern

- Durante la progettazione è indispensabile conoscere e utilizzare i design pattern, al fine di:
  - risolvere problemi progettuali specifici
  - rendere i progetti *object-oriented* più flessibili e riutilizzabili
- Ogni design pattern
  - cattura e formalizza l'esperienza acquisita nell'affrontare e risolvere uno specifico problema progettuale
  - permette di riutilizzare tale esperienza in altri casi simili

Testi consigliati

- Slide viste a lezione

- *C. Larman*, **Applicare UML e i pattern**
  Analisi e progettazione orientata agli oggetti
  (terza edizione), Prentice Hall, 2005
- *E. Gamma, R. Helm, R. Johnson, J. Vlissides*, **Design Patterns – Elements of Reusable Object-Oriented Software**, Addison Wesley, 1998

Bibliografia

- *M. Fowler*, **UML Distilled** (edizione italiana), Addison Wesley, Settembre 2000
- *S. Bennett, J. Skelton, K. Lunn*, **Introduzione a UML**, McGraw-Hill, 2002
- *W. Zuser, S. Biffl, T. Grechenig, M. Köhle*, **Ingegneria del software con UML e Unified Process**, McGraw-Hill, 2004
- *J. Arlow, I. Neustadt*, **UML e Unified Process** – Analisi e progettazione object-oriented, McGraw-Hill, 2003
- *L. A. Maciaszek*, **Sviluppo di sistemi informativi con UML**, Addison Wesley, Aprile 2002

Bibliografia

- *I. Sommerville*, **Ingegneria del software** (settima edizione), Addison Wesley, 2005
- *R. S. Pressman*, **Principi di Ingegneria del Software** (quinta edizione), McGraw-Hill, 2008
- *R. C. Martin and M. Martin*, **Agile Principles, Patterns, and Practices in C#**, Prentice Hall, 2006
- *M. Fowler*, **Refactoring – Improving the Design of Existing Code**, Addison Wesley, 1999

Software

- MSDN Academic Alliance

Utilizzabile per la modellazione UML

Esiste anche una versione Express più 'leggera' – scaricabile gratuitamente dal sito MS

Orario delle lezioni

Lunedì 14-17	14.15 – 15.45 16.00 – 16.45
Martedì 16-18+	16.15 – 17.45

Esame = Progetto + Orale;

Progetto

- Dato da un generico documento dei requisiti (scelta libera):
  - Analizzare i requisiti
  - Progettare l'applicazione
  - Realizzare un prototipo
- Singolarmente
- Team di 3 persone

Progetto

- **Da consegnare**
  - Relazione cartacea e pdf
- **Discussione del progetto**
  - Presentazione (ad es. ppt)
  - Prototipo dell'applicazione

Orale

- **Domande**
  su tutto il programma del corso
- **Test – quiz a scelta multipla**
  in sostituzione dell'orale
  - Non è possibile utilizzare appunti o altro
  - Solo negli appelli di gennaio e febbraio

Date esami marzo e aprile

- Test
  - gennaio 2010 – dalle ore 14.30
  - febbraio 2010 – dalle ore 14.30
- Discussioni e orali
  - In date da definire