

Materiale, link utili, avvisi

<http://www.dmi.unict.it/~tramonta/se>

Forum su SDAI

<http://www.sdai.unict.it>

leggere gli avvisi

partecipare alle discussioni

fare domande

## Materiali utili

---

Libri consigliati, poiché le slide da sole non sono sufficienti :-)

- Sommerville. *Software Engineering*. Pearson Addison-Wesley  
*oppure*
- Pressman. *Principi di Ingegneria del Software*. McGraw-Hill
  
- Fowler. *UML Distilled*. Pearson
  
- Gamma, Helm, Johnson, Vlissiders. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley

Ambienti di sviluppo

Eclipse ([www.eclipse.org](http://www.eclipse.org)), non basta Scite!

Jude, ArgoUML

## Lezioni

---

- Due per settimana in aula 2
  - lunedì 10.30 -- 13.30
  - mercoledì 10.30 -- 13.30
  - il 22 marzo ed il 24 marzo non ci sarà lezione
- Coprono tutto il programma del corso
- Consiglio vivamente di seguire *jugyou ni kite kudasai!*
  - Si impara di più, e si ascolta da un esperto
  - E' possibile fare domande ed ottenere risposte!
  - Si è aggiornati sulle ultime novità
  - Si possono guadagnare vari bonus
  - Sarà più facile sostenere gli esami

## Modalità Esami

---

- Esame finale
  - Mini-progetti individuali
  - Test a risposte multiple + 2 domande aperte su tutto il programma del corso
  - Domande orali
- Durante il periodo di lezioni
  - Mini-progetti individuali facoltativi (è un bonus)
  - Prove in itinere tramite test

# Obiettivi del corso

- Descrivere in dettaglio il processo di sviluppo del software
  - Consistente delle fasi di: analisi, progettazione, implementazione, testing, manutenzione
- Esaminare tecniche allo stato dell'arte per le suddette fasi
- Per le fasi di progettazione ed implementazione
  - Si farà riferimento alla progettazione ad oggetti
  - Si studierà lo standard UML
  - Si userà il linguaggio Java

# Contesto

Progetti piccoli	Progetti grandi
Uno sviluppatore	Team di sviluppatori
Lo sviluppatore decide cosa fare	I clienti decidono cosa fare
Un prodotto	Famiglie di prodotti
Pochi cambiamenti	Tanti cambiamenti contemporanei
Vita breve	Lunga vita
Poco costoso	Molto costoso
Poche conseguenze	Grandi conseguenze

← Programmare → Ingegneria del Software

# Cosa è l'ingegneria del software

- L'ingegneria del software si occupa della creazione di soluzioni economiche ed efficienti
  - per problemi pratici
  - applicando conoscenze scientifiche
  - per costruire prodotti software
  - con benefici per i clienti ed anche per la società
- La differenza tra ingegneria del software e computer science (informatica)
  - L'informatica si orienta sulla teoria ed i fondamenti
  - L'ingegneria del software si orienta ai problemi pratici di sviluppo e consegna di prodotti software utili

**Riferimenti**  
Pressman, capitoli 1 #1.5, 2.1, 9.2, 15.1, 27.7;  
Sommerville, capitolo 1

# Definizioni e Costi

- Processo software
  - Un insieme di attività ed i loro risultati che permettono di produrre un sistema software
- Modello software
  - Una descrizione semplificata di un processo software sotto una particolare prospettiva
    - Modello workflow, modello del flusso dei dati
- Costi orientativi per lo sviluppo di un sistema software
  - Dipendono dal modello adottato
  - Per lo sviluppo tramite il modello a cascata: 15% specifiche, 20% design, 18% codice, **47%** integrazione e test
  - Per sistemi di lunga durata: 25% sviluppo, **75%** evoluzione

# Ingegneria del software

- Negli anni '60 i sistemi software diventarono sempre più grandi (controllo aereo, prodotti commerciali)
- Nel '68 in un convegno NATO fu coniato il termine "crisi del software", questo termine significa:
  - Produzioni in ritardo, costi maggiori di quelli preventivati, mancata affidabilità dei sistemi software prodotti
- Definizioni di Ingegneria del software
  - *L'approccio sistematico e disciplinato allo sviluppo, all'operatività ed alla manutenzione del software; ovvero l'applicazione dell'ingegneria al software [IEEE]*
  - *La produzione di varie versioni di programmi da parte di tante persone [Parnas, 1974]*

# Abilità degli ingegneri del software

- Conoscenza di
  - Algoritmi e strutture dati
  - Linguaggi di programmazione
- Capacità di modellare
  - Operare a vari livelli di astrazione
  - Capire i requisiti, scrivere specifiche
  - Costruire modelli e ragionare con essi
- Capacità sociali
  - Lavorare in team grandi
  - Comunicare con le persone del team e con i clienti
  - Gestire il tempo e le risorse

Caratteristiche che rendono i sistemi software diversi dagli altri prodotti

# Caratteristiche del software

- Caratteristiche
  - Complessità, Conformità, Modificabilità, Invisibilità
- Il software è un prodotto **complesso**, poiché:
  - Componenti tutte differenti
  - Numero di stati cresce in modo combinatorio
  - Grandi dimensioni
  - Astratto ed immateriale
  - Non esistono leggi naturali che lo regolano
  - Difficile da comprendere

## Conformità

- Il software si deve ***conformare*** (adeguare) all'ambiente esterno
  - Molte interfacce hardware
  - Vari utenti con profili differenti
  - Processi lavorativi predefiniti
- La conformità aggiunge complessità al software

## Modificabilità

- Se un sistema software è di successo esiste sempre la necessità di cambiarlo
  - Per adattarlo ad una realtà che cambia (mutate esigenze)
  - Le richieste di estensione aumentano all'aumentare del successo
  - Poiché di successo, il sistema software sopravvive all'hardware per cui era stato sviluppato, generando una nuova esigenza di adattamento [vedi lezione sull'evoluzione]

## Invisibilità

- Il software è invisibile e immateriale
  - Non può essere catturato completamente da un'unica rappresentazione geometrica
  - Alcune rappresentazioni mirano ad evidenziare
    - Flusso di controllo
    - Flusso di dati
    - Dipendenze di componenti e variabili
    - Sequenze temporali
  - [vedi lezioni su diagrammi UML]

## Qualità

- Le tecniche dell'ingegneria cercano di produrre sistemi software entro i ***costi*** e i ***tempi*** preventivati e con ***qualità*** accettabile
- Come si valuta la qualità del software?
  - Definizione: *Totalità di caratteristiche di un prodotto che si basano sulla abilità a soddisfare i bisogni espliciti ed impliciti [ISO]*
- Criteri per valutare la qualità
  - Aderenza allo scopo e conformità alle specifiche
    - Il sistema software fa quello che il cliente vuole?
    - Il sistema software soddisfa le specifiche che erano state raccolte?
  - Efficienza, Manutenibilità, Dependability, Usabilità

# Qualità

- Correttezza
  - Un sistema software è corretto se soddisfa (ovvero è conforme con) le specifiche funzionali, assumendo che tali specifiche esistano
  - Se le specifiche sono formali, la correttezza può essere definita formalmente
    - Può essere provata come un teorema
    - Può essere rifiutata trovando contro-esempi
  - Cosa succede se le specifiche sono errate?
    - Il software conforme a tali specifiche è inutile

# Qualità

- Efficienza
  - L'uso di risorse da parte del software dovrebbe evitare sprechi (memoria, processore, comunicazione)
- Manutenibilità
  - La facilità di cambiare il software per soddisfare esigenze che cambiano
- Dependability
  - Il software dovrebbe essere affidabile (reliability), sicuro (security, essenzialmente proteggere i dati; safety, proteggere l'hardware)
    - Affidabile: probabilità che il software operi come atteso in un certo intervallo di tempo. Come reagisce il software ad input non attesi?
- Usabilità
  - Il software dovrebbe essere usato facilmente dagli utenti per cui è stato progettato

# Obiettivi

- Obiettivi (sfide) dell'ingegneria del software
  - Affrontare sistemi legacy (ereditati), eterogenei e ridurre i tempi di consegna
  - Sistemi legacy
    - Sistemi software antichi ma di valore (indispensabili) che devono essere mantenuti ed aggiornati
  - Eterogeneità
    - I sistemi sono distribuiti e consistono di un mix di hardware e software
  - Consegna
    - C'è una pressione crescente per avere i sistemi software più velocemente

# Responsabilità

- Un ingegnere del software deve comportarsi in modo onesto ed eticamente responsabile
  - La confidenzialità di collaboratori e clienti deve essere rispettata
  - Il livello di competenza non deve essere falsato
  - Le leggi sulla proprietà intellettuale (copyright) devono essere conosciute e rispettate
  - Le capacità tecniche non devono essere impiegate in modo non appropriato (es. per diffondere virus, danneggiare sistemi altrui, etc.)