Programma previsto per il corso di Ingegneria del Software

Docente: Massimo Cossentino

Anno Accademico 2015-16

1 Introduzione alla Ingegneria del Software

Concetti di: progetto, attività, risorsa, task, workproduct, sistema, modello, documento, obiettivi (goal), requisiti, vincoli, notazioni, metodi e metodologie.

Le fasi principali dello sviluppo: raccolta requisiti, analisi dei requisiti, progetto di sistema, progetto esecutivo o degli oggetti, implementazione, gestione del progetto, testing, ciclo di vita del software.

2 Linguaggi di Modellazione del Software

Introduzione all'UML, diagrammi principali:

1. diagrammi dei casi d'uso,
    a. Relazioni tra casi d'uso: include, extend, generalize
    b. Relazioni tra casi d'uso e attori: communicate
2. diagrammi delle classi,
    a. classi entity, boundary, control
    b. relazioni del diagramma delle classi
3. diagrammi di sequenza,
    a. Strutturazione dei diagrammi di sequenza, ordine oggetti,
    b. Strutture a forchetta e scala.
4. diagrammi di stato,
5. diagrammi di attività,
6. diagrammi dei componenti e di dislocazione.

Organizzazione dei diagrammi, estensione dei diagrammi, concetti di sistema, modello e vista. Modellazione object-oriented.

3 Analisi dei Requisiti

3.1 Raccolta dei requisiti

Il ciclo di vita del software. Introduzione alla raccolta dei requisiti.

Tecniche per la raccolta dei requisiti: questionari, analisi dei task, scenari, casi d'uso

Definizione di scenario.

Attività della raccolta dei requisiti:

- identificazione degli attori,
- euristica per l'identificazione degli scenari.

Il processo di raccolta e analisi dei requisiti

Concetti fondamentali:

- requisiti funzionali,
- requisiti non funzionali e pseudo-requisiti,
- principali requisiti non funzionali,
- livelli di descrizione,
- attributi delle specifiche: correttezza, completezza, consistenza, chiarezza, realismo, verificabilità e tracciabilità.

Documentazione della raccolta dei requisiti (template).

3.2 Analisi dei requisiti

Introduzione. Modelli di analisi: funzionale, degli oggetti, dinamico.
Modellazione funzionale. Dagli scenari ai casi d'uso:

a. identificazione dei casi d'uso,
b. documentazione dei casi d'uso (template)
c. affinamento dei casi d'uso,
d. identificazione delle relazioni tra attori e casi d'uso (relazione di comunicazione, estensione, inclusione, generalizzazione).
e. identificazione dei requisiti non funzionali.

Modellazione del sistema: oggetti di tipo entità, di confine e di controllo.
Attività:

a. dai casi d'uso agli oggetti (tecnica di Abbot)
b. identificazione degli oggetti entità,
c. identificazione degli oggetti di confine,
d. identificazione degli oggetti di controllo,
e. modellazione delle interazioni,
f. Identificazione degli attributi e operazioni,
g. identificazione delle associazioni (aggregazione, ereditarietà, generiche)
h. modellazione del comportamento degli oggetti:
    i. Diagrammi di interazione e stato.
    ii. Strutturazione dei diagrammi di sequenza, ordine oggetti, strutture a forchetta e scala. Eventi e gerarchie di eventi
i. Validazione dei requisiti e verifica delle transizioni tra modelli

Diagramma di interconnessione delle attività di analisi
Esempio di analisi dei requisiti (Toy car)
Documentazione dell'analisi dei requisiti (template).

4 Architetture Software

Introduzione. Obiettivi della progettazione di sistema
Attività della progettazione di sistema:

1. identificazione degli obiettivi di progetto (criteri di performance, affidabilità, costo, manutenzione, utente finale), obiettivi degli stakeholder, compromessi di progetto.
2. Decomposizione in sottosistemi.
    a. Definizione di sottosistema, servizio, interfaccia del sottosistema, API
    b. Relazioni tra sottosistemi: coerenza e accoppiamento
    c. Esempi di stili architetturali: livelli e partizioni, virtual machine (aperta e chiusa), SOA, client-server, peer to peer, 3-tier, 4-tier, Model-View-Controller, repository, pipe and filter.
3. Identificazione della concorrenza
4. La mappatura dei sottosistemi software sulle componenti hardware: la selezione di una configurazione hardware e una piattaforma, l'allocazione degli oggetti e sottosistemi sui nodi.
5. Gestione dei dati permanenti: la definizione dei depositi di dati (file, DB relazionali, oggetti, File system vs Database).
6. Gestione delle risorse globali: controllo degli accessi, matrice di accesso, lista di controllo degli accessi.
7. Il flusso di controllo: procedure-driven, event-driven, threads.
8. Le condizioni di confine: inizializzazione, terminazione, guasto.

La documentazione della progettazione di sistema (template).

5 Progettazione del software e codifica

Introduzione, concetti della progettazione del modello ad oggetti.
La progettazione del modello oggetti; oggetti d'analisi e oggetti della soluzione; Tipi, signature, visibilità.
Attività della progettazione del modello ad oggetti:

- identificare e adattare i componenti,
- identificare e adattare pattern, definizione di design pattern.
- specifica di tipi, signature e visibilità;
- specifica dei vincoli (constraint);
- specifica delle eccezioni;
- identificazione degli attributi e delle operazioni mancanti;
- identificazione e adattamento delle librerie di classi: rivisitare le gerarchie di ereditarietà, collassare le classi;
- realizzazione delle associazioni (uno a uno unidirezionale e bidirezionale, una molti, molti e molti, associazioni come oggetti separati, associazioni qualificate).
- caching/ritardo dei calcoli complessi.
- Component-based software engineering, approccio COTS (Commercial-off-the-shelf),
- Riuso del codice: white box (ereditarietà) e black box (composizione).

Uso dell'ereditarietà: descrizione di tassonomie e specifica delle interfacce:

- Ereditarietà di implementazione e specifica. Delegazione al posto dell'ereditarietà di implementazione. Operazioni e classi astratte. Ereditarietà di implementazione: Metodi riscrivibili e ereditarietà stretta, contrazione.

Specializzazioni dell'attore Developer: Class user, implementor, extender.
Specificare le interfacce:

- implementazione in Java della visibilità UML,
- informatin hiding,
- specificare la signature.
- Contratti: invarianti, pre-condizioni, post-condizioni; rappresentare i contratti nei modelli UML, Object Constraint Language.
    - semplici predicati OCL: context, inv, pre, post.
    - Javadoc, commenti in javadoc,
    - contratti come insieme di commenti Javadoc.

Documentazione del progetto del modello ad oggetti (template).

6 Testing Verifica e Validazione

Introduzione;
Concetti di test: componente, guasto, errore, malfunzionamento,test case, test stub/driver, validazione, testing vs debugging;
Tecniche di controllo della qualità, tecniche per evitare guasti, tecniche per la scoperta dei guasti, tecniche per tollerare i guasti;
Modello del test (test driver, input data, oracle, test harness).
Model driven testing;
Il modello del test ad oggetti;
Attività di testing (unit, integration, system, acceptance testing).
Black box testing, White box testing, branch testing;
Attività di testing:

- ispezione dei componenti;

- unit testing (equivalence testing, boundary testing, path testing , state-based testing);
- test di integrazione: stub e driver, strategie del test di integrazione (big bang, bottom-up, top-down, sandwich, modified sandwich, continuous integration), i passi del test d'integrazione;
- Test di sistema: test funzionale, test di performance, test pilota (alpha test, beta test), test di accettazione e test di installazione.

Pianificazione del test; documentazione del test.

7 Project Management

Elementi fondamentali del project management

- Caratteristiche fondamentali del progetto
- Attività (ordinarie, di riepilogo, cardine)
- Struttura delle attività
- Relazioni tra attività
- Le risorse (il calendario, i costi)
- I costi fissi del progetto

I diagrammi fondamentali del project management

- Diagramma di Gantt
- Diagramma di Pert
- Report fondamentali di tempi, costi e risorse

La gestione del progetto

- Il progetto iniziale
- Il progetto con previsioni
- Il progetto con variazioni