Elementi di Ingegneria del Software *Sommerville*

Alessio Veni

7 aprile 2025

1 Riassunto Capitolo 1 - Sommerville *Introduction to Software Engineering*

1.1 Cos'è l'ingegneria del software

L'ingegneria del software è una disciplina ingegneristica che si occupa di tutti gli aspetti della produzione del software: dalla definizione dei requisiti fino alla manutenzione del sistema dopo la sua consegna. È diversa dall'informatica perché si focalizza sulla realizzazione di sistemi software pratici, affidabili e gestibili.

1.2 Costi e sfide

Il software è spesso la componente più costosa di un sistema informatico. Le sfide principali sono:

- Diversità dei dispositivi e delle piattaforme
- Necessità di sviluppo rapido
- Affidabilità, sicurezza e manutenzione a lungo termine

1.3 Tipi di prodotti software

- **Prodotti generici**: software commerciali venduti a chiunque (es. software di grafica).
- **Prodotti personalizzati**: sviluppati su richiesta di un cliente specifico (es. sistemi di controllo).

1.4 Attributi del buon software

Un buon software deve essere:

- **Manutenibile**: facilmente aggiornabile
- **Affidabile e sicuro**
- **Efficiente**: uso ottimizzato delle risorse
- **Accettabile**: facile da usare, compatibile

1.5 Attività principali

Tutti i progetti software includono:

1. Specifica dei requisiti
2. Progettazione e sviluppo
3. Validazione del sistema
4. Evoluzione e manutenzione

1.6 Questioni generali

I software devono:

- Essere distribuiti e interoperabili
- Adattarsi a cambiamenti rapidi (sociali, tecnologici)
- Scalare da piccoli dispositivi a sistemi globali

1.7 Software per il web

Le applicazioni web hanno introdotto:

- Riuso di componenti e servizi web
- Sviluppo incrementale e metodi Agile
- Interfacce avanzate (AJAX, HTML5)

1.8 Etica e responsabilità professionale

Gli ingegneri del software devono agire eticamente. Temi chiave:

- Riservatezza delle informazioni
- Operare entro i propri limiti di competenza
- Rispetto della proprietà intellettuale
- Non abusare delle proprie competenze tecniche

Il codice etico ACM/IEEE include 8 principi che guidano il comportamento professionale, tra cui:

- Interesse pubblico
- Integrità professionale
- Qualità del prodotto
- Collaborazione e aggiornamento continuo

2 Capitolo 2 - Sommerville *Software Processes*

2.1 Cos'è un processo software

Un processo software è un insieme strutturato di attività per sviluppare un sistema software. Le attività chiave sono:

- **Specifica**: definire cosa deve fare il sistema.
- **Progettazione e implementazione**: costruzione tecnica del sistema.
- **Validazione**: verificare che il sistema soddisfi i requisiti.
- **Evoluzione**: adattare il sistema a nuove esigenze.

Un *modello di processo* è una rappresentazione astratta del processo, utile per guidare lo sviluppo.

2.2 Modelli di processo

- **Waterfall**: modello a fasi sequenziali, adatto a requisiti stabili.
- **Incrementale**: sviluppo graduale, con feedback continuo.
- **Integrazione e configurazione**: riuso di componenti esistenti, rapido ma meno flessibile.

2.3 Attività principali

Ingegneria dei requisiti

Include raccolta, specifica e validazione dei requisiti.

Progettazione e implementazione

Definizione dell'architettura, interfacce e database, seguita dalla programmazione.

Validazione

- **Test dei componenti**
- **Test del sistema**
- **Test col cliente**

Evoluzione

Il software deve essere aggiornato nel tempo per restare utile.

2.4 Gestione del cambiamento

- **Anticipazione**: uso di prototipi per prevedere modifiche.
- **Tolleranza**: sviluppo incrementale per adattarsi facilmente.

2.5 Prototipazione

Il prototipo è una versione semplificata del sistema:

- Aiuta a chiarire i requisiti e testare interfacce.
- Può essere scartato dopo l'uso.
- Migliora usabilità e qualità del design.

2.6 Consegna incrementale

Il sistema viene rilasciato a fasi:

- Ogni incremento ha valore reale per l'utente.
- I requisiti si definiscono nel tempo.
- Riduce i rischi di fallimento.

2.7 Miglioramento dei processi

Due approcci principali:

- **Maturità**: si migliora la gestione con buone pratiche.
- **Agile**: sviluppo veloce e iterativo, con meno burocrazia.

Modello di Maturità CMM

1. Iniziale
2. Ripetibile
3. Definito
4. Gestito
5. Ottimizzato

3 Capitolo 3 - Sommerville *Reflections*

3.1 Esempi in Java sulle Reflections

Disponibili nella cartella `reflection` del repository GitHub.

4 Capitolo 4 - Sommerville *Requirements Engineering*

4.1 Cos'è l'ingegneria dei requisiti

L'ingegneria dei requisiti è il processo con cui si definisce cosa deve fare un sistema software e sotto quali vincoli deve operare. È fondamentale, perché ogni errore in questa fase può compromettere l'intero progetto.

4.2 Tipi di requisiti

- **Requisiti funzionali**: descrivono i servizi specifici che il sistema deve offrire, cioè il comportamento atteso in certe condizioni.
- **Requisiti non funzionali**: definiscono vincoli generali (es. performance, usabilità, sicurezza). Spesso sono più critici dei funzionali.
- **Requisiti di dominio**: derivano dal contesto operativo del sistema (es. vincoli etici o legali).

4.3 Stakeholder

Gli stakeholder sono tutte le persone coinvolte o interessate al sistema:

- utenti finali
- manager
- tecnici IT
- sviluppatori
- enti esterni o regolatori

4.4 Requisiti e approcci Agile

Nei metodi Agile, i requisiti si scrivono in forma di *user stories* e si aggiornano man mano. Questo funziona bene nei sistemi aziendali ma è problematico per sistemi critici o distribuiti.

4.5 Raccolta dei requisiti (Elicitation)

È il processo di scoperta dei requisiti parlando con gli stakeholder. Tecniche principali:

- **Interviste**: domande aperte e chiuse per capire necessità e problemi.
- **Etnografia**: osservazione diretta del lavoro per cogliere aspetti impliciti.
- **Scenari e storie d'uso**: esempi realistici per stimolare la discussione.

Problemi comuni:

- stakeholder non sempre sanno cosa vogliono
- linguaggio tecnico o ambiguo
- requisiti in evoluzione continua

4.6 Specifica dei requisiti

Una volta raccolti, i requisiti devono essere documentati. Formati possibili:

- linguaggio naturale (frasi numerate)
- modelli grafici (es. UML)
- specifiche strutturate o tabellari

Devono essere chiari per i clienti e precisi per gli sviluppatori.

4.7 Validazione dei requisiti

Si verifica che i requisiti siano:

- **Validi**: rispondono ai bisogni reali
- **Coerenti**: senza contraddizioni
- **Completi**
- **Realistici**: realizzabili
- **Verificabili**: testabili

Strumenti: revisioni, prototipi, test-case.

4.8 Gestione dei cambiamenti

I requisiti cambiano nel tempo. È necessario un processo per:

- valutare l'impatto di ogni modifica
- tenere traccia delle dipendenze
- aggiornare documenti e design

4.9 Il documento dei requisiti

Documento ufficiale che descrive tutto ciò che il sistema deve fare. Contiene:

- introduzione e glossario
- requisiti utente e di sistema
- modelli e architettura
- appendici tecniche e indici

È una guida per sviluppatori e clienti.