1 Introduzione

1.1 Sistemi software

Un *sistema software* è un insieme di componenti software che funzionano in modo coordinato allo scopo di informatizzare una certa attività. La realizzazione di un sistema software richiede l'impiego di un gruppo di lavoro, nel quale ogni persona ricopre un ruolo ben preciso e le attività dei vari gruppi vanno coordinate, e tempo da dedicare alle varie fasi di sviluppo.

Esistono due categorie di sistemi software: i *sistemi generici*, definiti in base alle tendenze di mercato, e i *sistemi customizzati*, richiesti da uno specifico cliente (il committente).

1.2 Il processo software

Con *ingegneria del software* si intende l'applicazione del processo dell'Ingegneria alla produzione di sistemi software. Il processo è suddiviso in:

- specifica: definizione dei requisiti funzionali e non funzionali
- progettazione: si definiscono architettura, controllo, comportamento dei componenti, strutture dati, algoritmi, struttura del codice, interfaccia utente
- implementazione: scrittura del codice e integrazione dei moduli
- collaudo: si controlla se il sistema ha difetti di funzionamento e se soddisfa i requisiti
- manutenzione: modifiche del sistema dopo la consegna

1.3 Gestione del processo

L'ingegneria del software si occupa anche della gestione del progetto che si svolge in parallelo al processo software. Le principali attività di gestione sono l'*assegnazione* di risorse (umane, finanziarie...), la *stima del tempo* necessario per ogni attività, la *stima dei costi* e la *stima dei rischi*.

2 Specifica

La *specifica* è l'insieme di attività necessarie per generare il documento dei requisiti che descrive i *requisiti funzionali* e i *requisiti non funzionali*: descrive il "cosa" il sistema deve fare, non il "come". I requisiti servono per una proposta di contratto e modellare fasi successive del processo software.

2.1 Requisiti funzionali

I requisiti funzionali sono i servizi che il cliente richiede al sistema. Per ogni servizio si descrive:

- cosa accade nell'interazione tra utente e sistema
- cosa accade in seguito ad un certo input o stimolo
- cosa accade in particolari sistuazioni, ad esempio in caso di eccezioni

Non viene descritto come funziona internamente il sistema, in quanto è oggetto della successiva fase di progettazione.

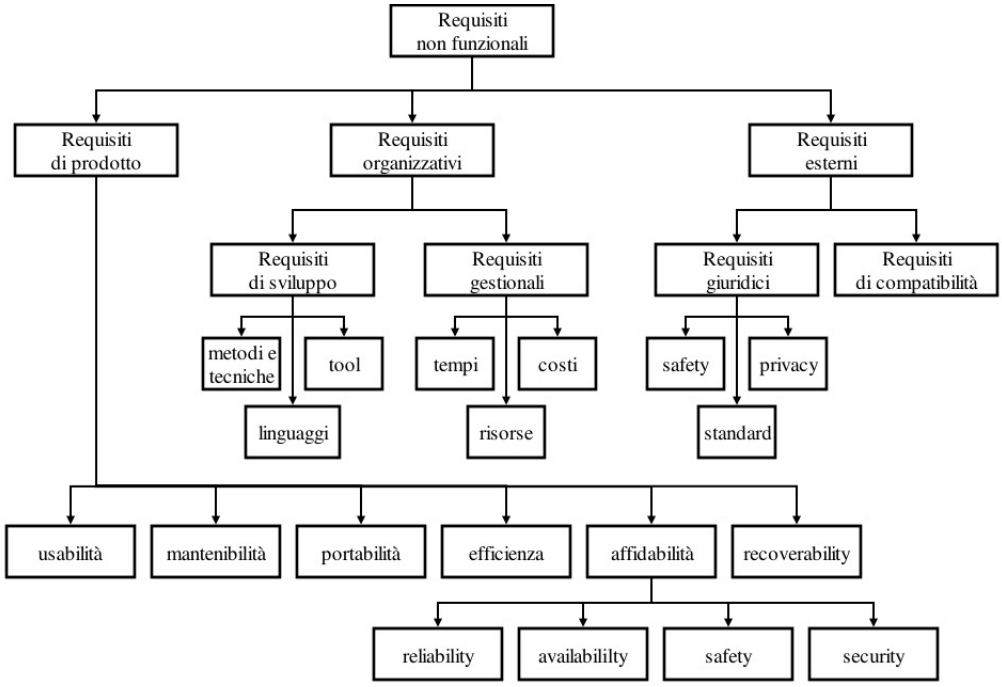
2.2 Requisiti non funzionali

I requisiti non funzionali sono divisi in tre categorie: *requisiti di prodotto*, *requisiti organizzativi* e *requisiti esterni*.

I requisiti di prodotto sono attributi che definiscono la qualità del sistema. Una *proprietà complessiva* riguarda il sistema nel suo complesso; una *proprietà emergente* è una proprietà che "emerge" dal funzionamento del sistema, dopo che è stato implementato.

I requisiti organizzativi sono caratteristiche riguardanti le fasi del processo software o la gestione del progetto. I *requisiti di sviluppo* sono i metodi e le tecniche di sviluppo utilizzati; i *requisiti gestionali* sono le risorse utilizzate.

I requisiti esterni derivano da fattori esterni al sistema e al processo software. Essi sono i requisiti di compatibilità con altri sistemi e aspetti giuridici.



Usabilità

L'*usabilità* è il grafo di facilità con cui l'utente riesce a comprendere l'uso del software. Il sistema deve avere un'interfaccia utente intuitiva ed curata, in quanto è fattore critico per il successo di un prodotto. L'uso del sistema deve essere ben documentato, per permettere all'utente di apprendere velocemente l'uso del prodotto. Il *training* degli utenti può migliorare l'usabilità del prodotto.

Mantenibilità

la *mantenibilità* è il grado di facilità di manutenzione. Le cause della manutenzione sono molteplici, e deve essere possibile l'evoluzione del software per soddisfare i requisiti nel tempo.

Portabilità

La *portabilità* è la capacità di migrazione da un ambiente ad un altro.

Recoverability

La *recoverability* è la capacità di ripristinare lo stato e i dati del sistema dopo che si è verificato un fallimento.

Efficienza

L'*efficienza* è il livello di prestazioni del sistema, e può essere misurato in vari modi: tempo di risposta, numero medio di richieste...

Affidabilità

L'*affidabilità* è il grado di fiducia con cui si ritiene che il sistema svolga in modo corretto la propria funzione. Ci sono varie misure di affidabilità:

- *reliability*: capacità di fornire i servizi in modo continuativo per una certa durata di tempo
- *availability*: capacità di fornire i servizi nel momento richiesto
- *safety*: capacità di operare senza causare danni materiali
- *security*: capacità di proteggersi da intrusioni e attacchi

Un sistema è definito *critico* quando il suo non corretto funzionamento può provocare conseguenze "disastrose" a persone e ambiente (*safety critical system*) o perdite economiche (*business critical system*). Il costo cresce in modo esponenziale rispetto al grado di affidabilità richiesto.

2.3 Processo di specifica

Il *processo di specifica* è il processo per generare il documento dei requisiti, ed è diviso in più fasi. Lo stesso requisito viene definito con due gradi di dettaglio diversi. Il *requisito utente* è descritto ad alto livello, in linguaggio naturale, ed è il risultato della deduzione dei requisiti. Il *requisito di sistema* è descritto dettagliatamente, fornendo tutti i dettagli necessari per la fase di progettazione, ed è il risultato dell'analisi dei requisiti.

Studio di fattibilità

Lo *studio di fattibilità* è la valutazione della possibilità di sviluppare il sistema e dei suoi vantaggi per il committente. Si decide se la costruzione del sistema è fattibile date le risorse disponibili e se il sistema è effettivamente utile al cliente. Per svolgere lo studio si raccolgono informazioni e si prepara un rapporto di fattibilità, che contiene la valutazione della possibilità di costruire un sistema e dei vantaggi che possono derivare dalla sua introduzione.

Deduzione dei requisiti

La *deduzione dei requisiti* è la raccolta di informazioni da cui dedurre quali sono i requisiti. Le informazioni si possono raccogliere mediante uno studio del dominio applicativo del sistema richiesto, il dialogo con stakeholder, studio di sistemi simili già realizzati e studio di sistemi con cui dovrà interagire quello da sviluppare.

Il *dominio applicativo* è l'insieme di entità reali su cui il sistema software ha effetto.

Uno *stakeholder* è, in ambito economico, il soggetto che può influenzare il successo di un'impresa o che ha interessi nelle decisioni dell'impresa; in ambito del processo software sono persone che possono influenzare il processo o che hanno interesse nelle decisioni assunte in esso.

È possibile dialogare con gli stakeholder tramite *interviste*, nelle quali viene chiesto di raccontare attraverso degli esempi reali come l'attività lavorativa funziona realmente, e tramite *etnografia*, l'osservazione dei potenziali utenti nello svolgimento delle loro mansioni.

Il dialogo con gli stakeholder presenta vari problemi, in quanto non sono in grado di indicare chiaramente cosa vogliono dal sistema, omettendo informazioni ritenute ovvie ed utilizzando terminologia non adatta. Inoltre, lo stesso requisito può essere espresso da più stakeholder in maniera differente, ed addirittura essere in conflitto.

Molti problemi scaturiscono dal *linguaggio naturale*, in quanto una descrizione ad alto livello di un requisito può generare confusione. La soluzione è quella di utilizzare il linguaggio in modo coerente, evitando gergo tecnico ed illustrando i requisiti tramite semplici diagrammi.

Analisi dei requisiti

La *analisi dei requisiti* è l'organizzazione, negoziazione e modellazione dei requisiti. Comprende:

- *classificazione e organizzazione dei requisiti*
- *assegnazione di priorità ai requisiti*: si stabilisce il grado di rilevanza di ogni requisito
- *negoziazione dei requisiti*
- *modellazione analitica dei requisiti*: produzione di modelli che rappresentano o descrivono nel dettaglio i requisiti

I *requisiti di sistema* sono l'espansione dei requisiti utente, e formano la base per la progettazione. Il linguaggio naturale non è adatto alla definizione di un requisito di sistema, quindi è necessario usare template, modelli grafici o notazione matematica.

Il *modello data-flow*, detto anche pipe & filter, permette di modelalre il flusso e l'elaborazione dei dati, ma non prevede la gestione degli errori. L'elaborazione è di tipo batch: input $\rightarrow$ elaborazione $\rightarrow$ output.

I requisiti non funzionali si possono specificare definendone delle misure quantitative:

- *efficienza*: tempo di elaborazione delle richieste, occupazione di memoria
- *affidabilità*: probabilità di malfunzionamento, disponibilità
- *usabilità*: tempo di addestramento, aiuto contestuale

Il *documento dei requisiti* contiene il risultato della deduzione e dell'analisi, ed è la dichiarazione ufficiale di ciò che si deve sviluppare. Il documento contiene una breve introduzione che descrive le funzionalità del sistema, un glossario contenente le definizione di termini tecnici, i requisiti utente e i requisiti di sistema, correlati con modelli UML. Il documento è letto da tutte le figure coinvolte nella realizzazione del progetto.

Validazione dei requisiti

La *validazione dei requisiti* è la verifica del rispetto di alcune proprietà da parte del documento dei requisiti, serve ad evitare la scoperta di *errori di specifica* durante le fasi successive del processo software. Sono da verificare le segueni proprietà:

- *completezza*: tutti i requisiti richiesti dal committente devono essere documentati
- *coerenza*: la specifica dei requisiti non deve contenere definizioni tra loro contraddittorie
- *precisione*: l'interpretazione di una definizione di requisito deve essere unica

- *realismo*: i requisiti devono essere implementati date le risorse disponibili
- *tracciabilità*

Quando si modifica un requisito bisogna valutarne l'impatto sul resto della specifica: è quindi necessario tracciarlo. Vari tipi di *tracciabilità*:

- *tracciabilità della sorgente*: reperire la fonte d'informazione relativa al requisito
- *tracciabilità dei requisiti*: individuare i requisiti dipendenti
- *tracciabilità del progetto*: individuare i componenti del sistema che realizzano il requisito
- *tracciabilità dei test*: individuare i test-case usati per collaudare il requisito

Per validare i requisiti si può impegare un gruppo di *revisori* che ricontrolli i requisiti e *costruire dei prototipi*.

3 Progettazione

3.1 Attività di progettazione

Durante la fase di *progettazione architetturale* viene definita la struttura del sistema, come questo verrà distribuito e come il sistema si dovrà comportare. Sono inoltre progettate le strutture dati, gli algoritmi e la GUI.

3.2 Progettazione architetturale

Strutturazione

Il sistema può essere strutturato in vari sottosistemi (strati), tipicamente tre. Ogni strato interagisce con gli strati adiacenti. Gli strati sono:

- *presentazione*: l'interfaccia utente, raccoglie i dati dall'utente
- *elaborazione*: elabora i dati in input e produce dati in output
- *gestione dei dati*: il database

Un *sottosistema* è la parte del sistema dedicata a svolgere una certa attività, mentre un *modulo* è la parte del sottosistema dedicata a svolgere particolari funzioni legate all'attività del sottosistema.

Deployment

Con *deployment* si intende la distribuzione dei componenti in vari dispositivi hardware. Nel *deployment a 1-tier* i tre stati del sistema sono concentrati su un dispositivo, in quello a *2-tiers* su due e in quello a *3-tiers* su tre.

Con deployment a 2-tiers si possono avere due soluzioni: *fat client* e *thin client*. Nel modello *thin client*, il server si occupa dell'elaborazione e della gestione dei dati, mentre il client si occupa della presentazione. Nel modello *fat client*, il server si occupa della gestione dei dati, mentre il client si occupa della presentazione e dell'elaborazione.

Metodo di controllo