Sviluppo Applicazioni Software

Luca Barra

Anno accademico 2023/2024

Indice

Capitolo 1

Processi per lo sviluppo software

1.1 Introduzione

In questa sezione verranno mostrati, anche in chiave storica, i principali processi per lo *sviluppo software*, *modelli di processi software*, sviluppo *iterativo* ed evolutivo, sviluppo *agile*.

Definizione 1.1.1: Software di qualità

- Non è un semplice programma o gruppo di programmi;
- Include *documentazione*, *test*, *manutenzione*, *aggiornamenti*;

Corollario 1.1.1 Caratteristiche essenziali

⇒ *Mantenibilità*: il software deve evolversi in base alle necessità dei clienti[a];

⇒ *Fidatezza*: il software non dovrebbe causare danni fisici o economici;

⇒ *Efficienza*: il software deve fare un uso efficiente delle risorse;

⇒ *Accettabilità*: il software deve essere comprensibile, usabile e compatibile con altri sistemi.

[a] Da questo si hanno i maggiori introiti.

Note:-

A volte può convenire vendere il software "sottoprezzo" per poi guadagnare con la manutenzione.

Question 1

Cosa descrive un processo software?

Risposta: descrive ***chi*** fa ***che cosa***, ***come*** e ***quando*** per raggiungere un *obiettivo*.

Definizione 1.1.2: Un processo per lo sviluppo software

Un processo software descrive un approccio *disciplinato* alla *costruzione*, al *rilascio* ed eventualmente alla *manutenzione* del software.

Si possono distinguere quattro attività di processo comuni:

⇒ *Specifiche del software:* clienti e sviluppatori definiscono le funzionalità del software (e i relativi vincoli);

⇒ *Sviluppo del software:* il software viene progettato e sviluppato;

⇒ *Convalida del software:* il software viene convalidato per garantire che soddisfi le specifiche del cliente;

⇒ *Evoluzione del software:* il software viene modificato per riflettere i cambiamenti nei requisiti del cliente e del mercato.

1.1.1 Specifica dei requisiti

Anche detta "*ingegneria dei requisiti*", è l'attività per capire e definire quali sono i requisiti richiesti dal sistema e identificare i vincoli all'operabilità e allo sviluppo del sistema.

Le fasi principali di questa attività sono:

⇒ *Deduzione e analisi dei requisiti:* osservazione di sistemi esistenti, discussioni con possibili utenti, analisi, etc.

⇒ *Specifica dei requisiti:* si traducono le informazioni raccolte in un *documento*;

⇒ *Convalida dei requisiti:* si controlla che i requisiti siano realistici, coerenti e completi.

1.1.2 Sviluppo del software

Anche detta "*progettazione e implementazione del software*", è l'attività di conversione delle specifiche del software in un sistema da consegnare al cliente. Nelle *metodologie agili* la progettazione e l'implementazione sono spesso *integrate* e, tipicamente, non producono documenti formali.

Le fasi principali di questa attività sono:

⇒ *Progettazione dell'architettura:* identifica la struttura complessiva del sistema, dei componenti, delle loro relazioni e della loro distribuzione;

⇒ *Progettazione del database:* si progetta la rappresentazione delle strutture dati che verranno utilizzate e la loro rappresentazione in un database[1];

⇒ *Progettazione dell'interfaccia:* definisce l'interfaccia utente e le modalità di interazione con il sistema[2];

⇒ *Progettazione e scelta dei componenti:* si ricercano i componenti riutilizzabili o vengono progettati nuovi componenti.

Note:-

La scelta dei componenti è particolarmente facile nel caso di linguaggi object-oriented.

1.1.3 Convalida del software

L'attività di verifica e convalida serve a dimostrare che un sistema sia *conforme* alle specifiche e che *soddisfi* le esigenze del cliente. La convalida richiede anche attività di *controllo*, *ispezione* e *revisione* a ogni stadio del processo di sviluppo. In alcune meteodologie agili si scrivono i test prima di scrivere il codice (extreme programming).

Note:-

In questo corso ci si concentrerà sul processo di testing. Per una modo formale di verificare la correttezza di un sistema si può fare riferimento al corso "Metodi formali dell'informatica".

I test possono essere:

⇒ *Test di unità (o dei componenti):* i componenti vengono testato singolarmente[3];

⇒ *Test del sistema:* si testa il sistema nel suo complesso;

⇒ *Test del cliente:* il sistema viene testato dal cliente con i propri dati.

[1] Non verrà trattata in questo corso. È stata parzialmente trattata nel corso "Basi di dati".
[2] Non verrà trattato lo sviluppo di un'interfaccia. È stato parzialmente trattata in "Programmazione III".
[3] Visti nel corso "Algoritmi e strutture dati".

1.1.4 Evoluzione del software

Anche detto "*manutenzione del software*", è l'attività di modifica durante o dopo lo sviluppo di un sistema software. La distinzione (storica) tra sviluppo e manutenzione è sempre più irrilevante. L'ingegneria del software è un unico processo evolutivo.

Note:-

Può capitare che si debba far fronte a cambiamenti improvvisi per esigenze di mercato o per incomprensioni con il cliente.

Bisogna *ridurre* i costi di rilavorazione:

⇒ *Anticipazione dei cambiamenti:* si possono prevedere o anticipare eventuali cambiamenti prima di una richiesta di rilavorazione;

⇒ *Tolleranza ai cambiamenti:* si progetta il sistema in modo da rendere facili eventuali cambiamenti.

Ci sono due metodi per far fronte ai cambiamenti:

⇒ *Prototipazione del sistema:* il sistema viene sviluppato rapidamente per verificare i requisiti del cliente. Ciò consente eventuali modifiche prima di sviluppare il sistema completo;

⇒ *Consegna incrementale:* vengono consegnati al cliente parti del sistema in modo incrementale in modo che il cliente possa provarlo e commentarlo.

Note:-

Il refactoring è un importante meccanismo per supportare la tolleranza ai cambiamenti

1.2 Modelli di processo software

Esitono veri modelli di processo software: *cascata*, *Unified Process*, *Scrum*, *XP*, *RUP*, *RAD*, *Spirale*, etc. Le quattro attività fondamentali sono organizzate in modo diverso in ciascun modello: in sequenza nel modello a cascata e intrecciate negli altri (modelli incrementali). Un ulteriore modello è il modello a integrazione e configurazione che però è poco trattato a livello ingegneristico.

Definizione 1.2.1: Paradigma di processo

Il modello di processo software è una rappresentazione semplificata di un processo software. Sono strutture di processo da *estendere* e *adattare* per soddisfare le esigenze specifiche di un progetto.

Note:-

Non esiste un modello di processo software "universale", ma la scelta del modello dipende dai requisiti del cliente:

⇒ i software a sicurezza critica richiedono un modello a cascata per via delle analisi e della documentazione;

⇒ i software per il mercato richiedono un modello incrementale;

⇒ i sistemi aziendali richiedono un modello a configurazione e integrazione.

Inoltre, in grandi sistemi, si possono combinare più modelli.

1.2.1 Modello a cascata

Definizione 1.2.2: Modello a cascata

Il modello a cascata è un modello di processo software in cui le fasi di sviluppo sono viste come *fasi distinte* e *non sovrapposte*.
Questo modello era l'unico modello utilizzato fino agli anni '80.

Note:-

Si contrappone ai modelli incrementali in cui le fasi di sviluppo sono sovrapposte e iterate.

Corollario 1.2.1 Fasi del modello a cascata

⇒ All'inizio si definiscono i requisiti;

⇒ All'inizio si definisce un piano temporale;

⇒ Si progetta e modella il sistema;

⇒ Si crea un progetto completo del software;

⇒ Si inizia la programmazione del sistema;

⇒ Si testa il sistema, si rilascia e si prosegue con la manutenzione.

Il modello a cascata:

⇒ Non è adatto allo sviluppo in team;

⇒ Si dovevano definire spesso modelli matematici;

⇒ Costava molto in termini di tempo e denaro.

1.2.2 Modello incrementale

Definizione 1.2.3: Modello incrementale

Il modello incrementale è un modello di processo software in cui il sistema viene sviluppato in *incrementi* (o *iterazioni*). Si effettuano *feedback veloci* e *rilasci*.

Note:-

Negli anni '80 e '90 molte persone si avvicinano al mondo della progettazione e nasce la necessità di sviluppare software in modo incrementale.

Corollario 1.2.2 I casi d'uso

I casi d'uso sono il modo migliore per definire i requisiti: il cliente racconta una storia e il programmatore la traduce in un caso d'uso.

Lo sviluppo incrementale:

⇒ È un approccio *plan-driven*, *agile* o una combinazione di questi approcci;

⇒ Se *plan-driven*, si pianificano in anticipo gli incrementi;

⇒ Se *agile*, si identificano gli incrementi iniziali ma si dà priorità al rilascio di incrementi che soddisfano i requisiti più importanti;

⇒ Il costo di implementazione di modifiche è ridotto;

⇒ È più facile ottenere un feedback dal cliente;

Note:-

Tuttavia si devono avere consegne regolari e frequenti, la struttura dei sistemi tende a degradarsi e richiede pianificazione in anticipo per grandi team.

1.2.3 Integrazione e configurazione

Definizione 1.2.4: Riutilizzo del software

⇒ Dagli anni 2000 si sono diffusi software che riutilizzano software già esistente;

⇒ Collezioni di oggetti che sono sviluppati come un componente o un pacchetto da integrare tramite framework;

⇒ Servizi web che possono essere integrati in un sistema.

Le fasi principali sono:

⇒ *Specifica dei requisiti*;

⇒ *Ricerca e valutazione del software:* se esiste un software che soddisfa i requisiti;

⇒ *Perfezionamento dei requisiti:* utilizzando le informazioni trovate nella ricerca;

⇒ *Configurazione del sistema di applicazioni*;

⇒ *Adattamento e integrazione:* si integra il sistema con i componenti riutilizzabili.

Note:-

Questo approccio riduce la quantità di software da sviluppare, riducendo i costi e i rischi. Però bisogna scendere a compromessi con i requisiti e si perdwe il controllo sull'evoluzione del sistema.

1.2.4 Sviluppo incrementale, iterativo ed evolutivo

Questo modello è:

- **Incrementale:** si incrementa il codice man mano che si sviluppa;
- **Iterativo:** si sviluppa il software in cicli (iterazioni);
- **Evolutivo:** si sviluppa il software in modo che possa evolvere a ogni iterazione richiedendo un feedback.

Definizione 1.2.5: Approccio iterativo

Nell'approccio iterativo:

⇒ lo sviluppo è organizzato in mini-progetti brevi (le iterazioni);

⇒ il risultato di ogni iterazione è un sistema parzialmente funzionante (testato e integrato);

⇒ ogni iterazione dura poche settimane e comprende le proprie attività di analisi, sviluppo, etc.;

⇒ si ottiene un feedback a ogni iterazione.

Note:-

Git supporta lo sviluppo incrementale, iterativo ed evolutivo.

1.3 Sviluppo agile

1.3.1 I principi dello sviluppo agile

1.3.2 Estreme Programming (XP)

1.3.3 Scrum