INGEGNERIA DEL SOFTWARE - A.A. 2009/10

CORSO DI LAUREA IN INFORMATICA

FACOLTÀ DI S.M.F.N.

Prof. Marco Antonio Mastratisi

Esercitatrice: Dott.ssa Susanna Cozza

Lezione su Ciclo di Vita del Software e Metodologie di Sviluppo

1 Introduzione

L’espressione “ciclo di vita del software” fù coniata pressappoco in contemporanea con la nascita della disciplina della Ingegneria del Software, tra la fine degli anni sessanta del 1900 e l’inizio della decade successiva. Fù allora che si postulò la esigenza di un cambio di prospettiva riguardo alla realizzazione di programmi e sistemi software, con il passaggio dallo sviluppo del software inteso come attività “artiginale”, affidata prevalentemente alla creatività dei singoli sviluppatori, alla individuazione di approcci di tipo industriale, consistenti in processi complessi e richiedenti pianificazione, controllo e documentazione appropriati.

Questa transizione fu generata principalmente dalla diffusione in diversi campi delle applicazioni software e dall’incremento delle loro potenzialità. Si determinarono una aumentata complessità dei sistemi, l’avvento di un vero mercato del software, e conseguentemente la formulazione di nuovi e più stringenti requisiti di qualità, legati ad esempio all’uso di sistemi informatici in contesti critici (centrali energetiche, sistemi aerospaziali, settore militare in genere, etc..). La risposta al nuovo contesto fu la teorizzazione e sperimentazione, tanto in ambito accademico quanto in quello industriale, di metodologie o processi di sviluppo che scompongono la realizzazione di prodotti software in attività fra loro coordinate, il cui risultato finale è non solo il prodotto stesso ma anche tutta la documentazione ad esso associata. La specifica sequenza di attività individuate nel contesto di una determinata metodologia prende il nome di “ciclo di vita”, ad indicare che si tratta delle operazioni da compiersi nell’arco della intera vita del software di interesse, dalla sua creazione in base ad assegnati requisiti fino alla sua realizzazione e manutenzione postuma.

Pur potendosi paragonare il tipo di evoluzione descritta a quanto avvenuto, con tempi diversi, in altri settori più maturi della ingegneria va rimarcato il fatto che la produzione del software presenta ancora oggi caratteristiche proprie e non del tutto assimilabili ad altre produzioni industriali. Si può asserire che in molti casi essa appare più prossima ad un’arte che ad un processo manifatturiero ripetitivo. D’altro canto va sottolineato che in tempi recenti lo sviluppo di sistemi complessi viene spesso condotto a partire da package (pacchetti) adattabili. In questo ambito, l’enfasi si sposta dalla progettazione e costruzione del software, all’adattamento di software preesistente. Tuttavia algoritmi, librerie di codice, classi riusabili, componenti software eccetera, costituiscono soluzioni incomplete alle necessità che sorgono durante lo sviluppo di sistemi informativi. La sfida è pertanto mettere insieme piccole parti di un sistema aziendale coerente. Inoltre i costrutti concettuali (i modelli) devono essere creati per ogni sistema sviluppato da zero in modo che possano soddisfare le necessità peculiari della organizzazione destinataria del sistema.

2 Fonti di difficoltà associate allo sviluppo del software

Le fonti di difficoltà associate allo sviluppo del software possono essere raggruppate in tre categorie:

- Stakeholder (le persone interessate al progetto)
- Processi (le metodologie utilizzate per la realizzazione)
- Linguaggi e strumenti di modellazione (gli strumenti per rappresentare le scelte progettuali)

2.1 *Stakeholder*

Uno stakeholder è una persona che ha un ruolo in un progetto software. Possibili stakeholder sono i clienti e gli sviluppatori. Le principali cause di fallimento di un progetto di sviluppo software sono legate agli stakeholder e possono essere così semplificate:

- Le necessità del cliente sono mal comprese
- I requisiti del cliente cambiano troppo frequentemente
- I clienti non forniscono sufficienti risorse ai progetti
- I clienti non vogliono cooperare con gli sviluppatori
- I clenti hanno attese non realistiche
- Il sistema non porta benefici ai clienti

Naturalmente è importante fare costantemente attenzione a questi possibili sintomi di difficoltà durante tutta la fase dello sviluppo del software ed, eventualmente, correre ai ripari.

2.2 *Processo*

Il processo di sviluppo software definisce le attività e le procedure organizzative per incremetare la collaborazione del gruppo di sviluppo in modo da fornire al cliente un prodotto di qualità. Un modello di processo:

- Stabilisce un ordine di esecuzione delle attività
- Specifica quali elaborati dello sviluppo devono essere forniti e quando
- Assegna le attivivtà e gli elaborati da sviluppare agli sviluppatori
- Fornisce criteri per monitorare il progresso del processo, per misurarne i risultati e per pianificare i progetti futuri

Esistono varie possibili metodologie per la formalizzazione del processo di sviluppo del software. Esse differiscono le une dalle altre per il livello di accuratezza con cui valutano i progressi del processo. E’ importante valutare la dimensione del processo per scegliere l’opportuna formalizzazione del processo stesso. Alcune formalizzazioni per i processi di sviluppo sono:

- Il processo iterativo ed incrementale, che segue le varie fasi dello sviluppo con verifiche periodiche dei progressi ed eventuali revisioni delle scelte fatte precedentemente
- Il Capability Maturity Model (CMM); esso si concentra sulla valutazione della qualità del processo attualmente utilizzato, più che definire un processo vero e proprio. In particolare, in seguito ad un’analisi CMM del processo attuale è possibile categorizzarlo su uno dei seguenti livelli:
    - iniziale - situazione caotica dello sviluppo del progetto

- ripetibile – la gestione del progetto è ripetibile, nel senso che segue delle regole precise seppur rudimentali
- definito – uso di metodo e strumenti per lo sviluppo e la gestione del processo
- gestito – si ha la possibilità di valutare il processo
- ottimizzato – miglioramento continuo del processo

La maggior parte dei processi reali si trovano al primo livello (iniziale), alcuni al secondo, quasi nessuno al quinto.

– ISO 9000. E' uno standard che devinisce regole precise per la valutazione della qualità del processo. Molte aziende attualmente ricercano la certificazione ISO 9000 in quando indice di buona qualità di controllo del software prodotto. Si noti che anche in questo caso, la ISO 9000 non definisce come fare ma cosa (per esempio cicli di verifica periodici, documentazioni, testing etc.)

2.3 *Linguaggi e strumenti di modellazione*

Uno dei principali problemi nello sviluppo del software è la comunicazione tra i clienti e gli sviluppatori. Questi, infatti, parlano spesso due linguaggi distinti, proveniendo da background molto lontani. E' necessario, quindi, l'utilizzo di un linguaggio ed una terminologia espressiva comune ad entrambi per definire le specifiche del progetto. Tale linguaggio deve permettere la costruzione di modelli a vari livelli di astrazione, dovrebbe avere una forte componente visuale (una figura vale più di mille parole), ed una robusta semantica dichiarativa.

Gli sviluppatori hanno bisogni di CASE Tools (Computer-Assisted Software Engineering) per memorizzare, manipolare e realizzare i modelli rappresentativi delle specifiche di progetto. Infine, deve sussistere una forte correlazione tra il modello e lo sviluppo del software. Uno dei linguaggi che attualmente soddisfa tutte queste caratteristiche è UML (Unified Modeling Language). Esso possiede le seguenti caratteristiche:

– E' indipendente da qualunque processo di sviluppo
– Segue un approccio object-oriented allo sviluppo del software
– E' visuale
– E' indipendente dalle tecnologie per l'implementazione del software
– Consente la modellazione della struttura statica e del comportamento dinamico di un sistema. In particolare fornisce:
    - Modelli dello stato – per descrivere la struttura statica dei dati
    - Modelli del comportamento – per definire la collaborazione tra gli oggetti
    - Modelli del cambiamento di stato – per indicare gli stati consentiti

3 Fasi del ciclo di vita del software

Per ciclo di vita, si intende l'insieme ordinato di attività gestite nel corso di ogni progetto di sviluppo. In linea di massima, si possono individuare tre macro-fasi del ciclo di vita di un software: l'*analisi*, il *progetto* e l'*implementazione*. Per *analisi* si intende l'analisi dei requisiti del sistema; in questa fase, i requisiti sono individuati e specificati formalmente. Il *progetto* include sostanzialmente due sottofasi principali: il *progetto architetturale* ed il *progetto di dettaglio*. Queste fasi sono importanti per la definizione della comprensibilità, manutentibilità e della scalabilità del sistema. Infine, l'*implementazione* prevede l'effettiva realizzazione del sistema.

Più in dettaglio, è possibile individuare sette fasi del ciclo di vita del software:

- Determinazione dei requisiti
- Specifica dei requisiti
- Progetto architetturale
- Progetto di dettaglio
- Implementazione
- Integrazione
- Manutenzione

Parallelamente a queste sette fasi, particolare importanza rivestono le fasi di Pianificazione delle Attività e di Testing. Di seguito analizzeremo ciascuna di queste fasi più in dettaglio.

3.1 *Determinazione dei requisiti*

E' forse una delle fasi più importanti e delicate del processo. L'obiettivo è quello di fornire unadefinizione informale dei requisiti, funzionali e non, che gli stakeholder si aspettano di trovare nel sistema. Alcuni dei requisiti più importanti sono:

- Servizi attesi dal sistema (le funzionalità)
- Vincoli cui il sistema deve sottostare
- Requisiti di sistema (lo scopo)
- Requisiti Funzionali (funzioni di business)
- Requisiti Informativi (strutture dati)

Esistono diverse tecnice di raccolta dei requisiti. Un problema comune in tutte le tecniche è far parlare al cliente ed all'analista lo stesso linguaggio. Tra i *Metodi Tradizionali* di raccolta dei requisiti citiamo:

- *Interviste*: spesso i clienti possono avere solo una vaga idea dei propri requisiti e non essere in grado di esprimere i propri requisiti in termini comprensibili. A tale scopo possono essere utilizzate sia Interviste strutturate (formali) che prevedono l'uso di domande (aperte o chiuse) predeterminate, sia Interviste non strutturate realizzate tramite incontri informali.
- *Questionari*: servono a chiarire questioni specifice. Di solito vengono realizzati in aggiunta alle intervista. Alcune regole di base per i questionari sono: evitare i quesiti aperti; utilizzare questiti a scelta multipla o quesiti a punteggio o quesiti con ordinamento.
- *Osservazioni*: esse consistono nell'osservare le attività da implementare nel sistema. E' possibile realizzare sia osservazioni passive, in cui l'analista osserva le attività senza

interruzioni o coinvolgimento, oppure osservazioni attive in cui l'analista partecipa direttamente alle attività.
- *Studio dei documenti e dei sistemi software*. In questo caso si studia l'esistente.

Tra i *Metodi Moderni,* invece, citiamo:

- *Prototipi:* si realizzano sistemi dimostrativi "sporchi e veloci" che presentano una soluzione e simulano il comportamento del sistema relativo a vari eventi tramite GUI. A questo proposito si possono utilizzare prototipi "usa e getta" (consigliati) o prodotipi evolutivi.
- *Sviluppo* cooperativo delle applicazioni (JAD): in questo casi si mettono insieme tutti gli stakeholder (Leader, segretario, clienti, utenti, manager e sviluppatori) e si cerca di creare una sinergia di gruppo per lo sviluppo del software.
- *Sviluppo rapido delle applicazioni (RAD)*: combina cinque teniche: Prototipi evolutivi, Strumenti CASE, Gruppi di specialisti con strumenti avanzati di sviluppo, JAD interattivo, Time boxing.

3.2 *Specifica dei requisiti*

Questa fase comprende la modellazione dei requisiti acquisiti nella fase precedente e la definizione formale dei vari aspeti del sistema. Tra i modelli di specifica, occorre prevedere almeno: i modelli dello stato, i modelli del comportamento ed i modelli del cambiamento di stato. In particolare:

- *I modelli dello stato*: rappresentano i requisiti informativi; lo stato di un oggetto è determinato dal vaore dei propri attributi ed associazioni (ad esempio se il conto è in rosso). Essi forniscono una vista statica del sistema e prevedono la definizione delle classi per il dominio e degli attributi e relazioni tra le classi.
- *I modelli del comportamento*: specificano l'interazione tra gli oggetti; forniscono una vista operazionale del sistema e ne definiscono i casi d'uso. Altri aspetti importanti sono la definizione delle operazioni delle classi e lo scambio di messaggi tra oggetti (modellazione delle interazioni). Si noti che a questo livello si fissa uno stato del sistema e si ragiona su quello stato. I cambiamenti di stato sono gestiti dall'altro tipo di modello.
- *I modelli del cambiamento di stato*: rappresentano l'evoluzione del sistema. In genere si utilizzano dagrammi di stato (statechart) che specificano stati e transizioni (causate da eventi).

3.3 *Progetto architetturale e di dettaglio*

La distinzione tra analisi e progetto non è netta. Il progetto è costruito sulla piattaforma hardware/software di implementazione del sistema. Nella fase di progetto architetturale viene prodotta una descrizione del sistema in termini dei suoi moduli, mentre la descrizione dei componenti interni di ogni modulo viene realizzata nella fase di progetto di dettaglio. Possibili architetture generali per i sistemi software sono:

- Locale: tutto viene memorizzato ed eseguito su un singolo PC
- Client/Server
- Architettura distribuita di elaborazione – più computer sono collegati in rete e ciascuno può fungere da client e da server; i dati possono essere distribuiti

In questo contesto, è importante fare anche riferimento a possibili strategie di riuso:

- di classi
- di componenti (una parte fisica del sistema: eseguibile, libreria, tabelle di dati, file, documenti, ecc.)
- di strategie di risoluzione

3.4 *Implementazione*

In questa fase, scelta l'architettura, definite le classi e le loro interrelazioni, si rende tutto operativo.

3.5 *Integrazione*

In progetti complessi, i vari moduli sono progettati e realizzati separatamente. In questa fase, si integra il tutto per ottenere un unico sistema. Nella realizzazione dei singoli moduli spesso bisogna simulare le interazioni con gli altri moduli. In questi casi si fa uso di moduli particolari chiamati STUB. In questa fase, è molto delicato eliminare le simulazioni e sostituirle con le implementazioni reali.

3.6 *Manutenzione*

In questa fase si deve distinguere tra:

- Gestione ordinaria (manutenzione ordinaria);
- Manutenzione adattativa , che prevede il monitoraggio, la verifica e l'eventuale modifica del sistema per porarlo ai livelli richiesti;
- Manutenzione evolutiva, che fa fronte a nuove richieste o esigenze dei clienti.

3.7 *Pianificazione delle attività*

Per la buona riuscita di tutto il progetto, è importante avere una buona attività di pianificazione in tutte le fasi del processo di sviluppo. In particolare è fondamentale:

- Pianificare i tempi di sviluppo (valutando anche i rischi)
- Pianificare incontri periodici di valutazione
- Pianificare il processo di sviluppo (per moduli o fasi dipendenti tra loro)
- Pianificare le spese

Molto comune, in progetti di grandi dimensioni, è la realizzazione di un'analisi di fattibilità che vauta i rischi e l'effettiva praticabilità del progetto. Elementi di valutazione in questa analisi sono:

- Fattibilità operaizonale, essa valuta l'impatto del sistema sulle strutture organizzative
- Fattibilità economica, in cui si fa una stima dei costi/benefici
- Fattibilità tecnica, in cui si stima la possibilità di realizzare le soluzioni proposte in relazione alle risorse disponibili
- Fattibilità temporale, in cui si valuta la ragionevolezza delle scadenze

3.8 *Testing*

E’ una fase molto delicata da non fare solo alla fine dell’implementazione. E’ fondamentale riuscire a scoprire possibili guasti per tempo. Per garantire il giusto grado di oggettività, è opportuno che questa fase sia realizzata da terze parti e non dagli sviluppatori stessi. Esistono sostanzialmente due tipologie di test:

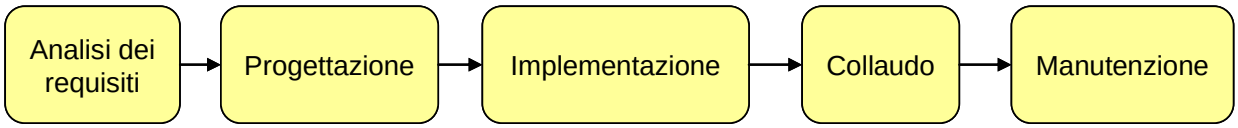
- Test delle specifiche , anche chiamato black box in cui si ignora il dettaglio implementativo del sistema e si considerano solo gli input e gli output del sistema.
- Test del codice, anche detto white box che analizza la logica del programma e studia gli input di test da fornire in modo opportuno.

Le qualità principali da testare in un sistema sono:

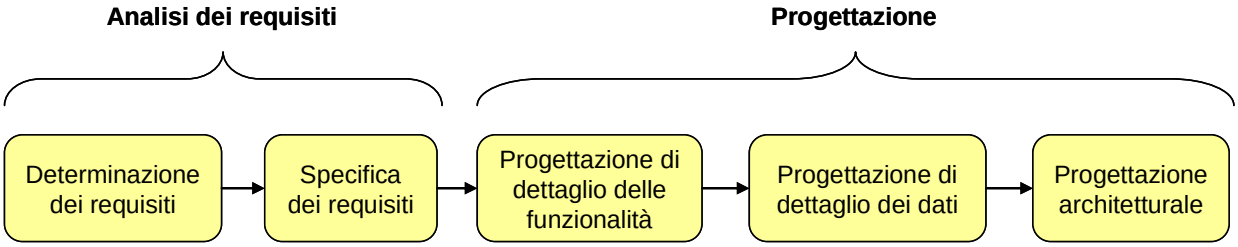
- La correttezza. Naturalmente il sistema deve funzionare correttamente.
- L’efficienza. Si deve fare riferimento sia all’efficienza in termini di tempo che in termini di spazio occupato.
- La robustezza, ovvero il comportamento del sistema in situazioni impreviste
- L’usabilità. Particolare attenzione va posta sulla flessibilità del sistema, sull’interfaccia che deve essere amichevole, sulla capacità di fornire aiuti e spiegazioni agli utenti e sulle verifiche degli input e richieste di conferme sui dati immessi.
- La modularità del sistema
- L’information Hiding, ovvero la trasparenza
- La riusabilità dei componenti
- L’interoperabilità, cioè la capacità di interagire con altri sistemi
- La manutentibilità. In questo caso bisogna verificare la possibilità di realizzare i tre tipi di manutenzione: correttiva, adattativa e perfettiva introdotte precedentemente
- La portabilità. Quanto il sistema può essere utilizzato su computer aventi piattaforme differenti.
- Leggibilità
- Completezza ed efficacia della documentazione

4 Una metodologia per la analisi e progettazione di sistemi software basata sull'uso (parziale) di UML

Si può fornire la seguente schematizzazione generale per le fasi più interessanti del processo di sviluppo software



Per gli scopi del presente corso di Ingegneria del Software si può fare riferimento alle sole prime due fasi, scomponibili in dettaglio come segue:



Qui di seguito per ognuna delle sotto-fasi è riportata una descrizione qualitativa e gli eventuali modelli (diagrammi) UML che è tipicamente previsto che vengano utilizzati:

Determinazione dei requisiti	Acquisizione delle specifiche, comprensione del dominio su cui il sistema software dovrà insistere.	
Specifica dei requisiti	Modellazione statica delle funzionalità previste per il software.	Diagrammi dei casi d'uso
Progettazione delle funzionalità	Modellazione dinamica delle funzionalità e dei processi in cui si articolarno.	Diagrammi di attività, Diagrammi delle sequenze.
Progettazione dei dati	Modellazione concettuale statica dei dati di interesse per il software, modellazione dinamica del ciclo di vita di oggetti interessanti.	Diagrammi delle classi, Diagrammi di stato.
Progettazione architetturale	Modellazione della distribuzione fisica di dispositivi hardware e componenti software.	Diagrammi di distribuzione.