

**Dispensa su Analisi dei Requisiti e
Ciclo di vita del Software
Corso di Ingegneria del Software
Dipartimento di Matematica – Università della Calabria
Docente: Giorgio Terracina**

IL PROCESSO DI SVILUPPO DEL SOFTWARE

Quando si ha a che fare con progetti di grandi dimensioni, è necessario organizzare lo sviluppo del progetto con molta attenzione. Nell'ambito dei sistemi informativi, infatti, si hanno innumerevoli esempi di progetti falliti, tempi e costi non rispettati, soluzioni scorrette, sistemi non gestibili, ecc. In questo contesto, è necessario trovare una risposta alle seguenti domande:

- Quali sono le cause, e quali i sintomi della presenza di un problema in un progetto?
- Come devono essere trattati?

Lo sviluppo del software è più vicino ad un'arte che ad un processo manifatturiero ripetitivo. Le fonti di difficoltà associate allo sviluppo del software possono essere raggruppate in tre categorie:

- Stakeholder (le persone interessate al progetto)
- Processi
- Linguaggi e strumenti di modellazione

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 mettere insieme piccole parti di un sistema aziendale coerente.

E' importante sviluppare sistemi complessi 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, i costrutti concettuali (i modelli) devono essere creati per ogni sistema sviluppato da zero in modo che possano soddisfare le necessità proprie di un'organizzazione: è improbabile che un'organizzazione possa trovare sul mercato un package per automatizzare le sue attività fondamentali (di core business).

In seguito dettaglieremo le tre categorie evidenziate sopra.

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 clienti 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.

PROCESSO

Il processo di sviluppo software definisce le attività e le procedure organizzative per incrementare 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 attività 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 definisce regole precise per la valutazione della qualità del processo. Molte aziende attualmente ricercano la certificazione ISO 9000 in quanto 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.)

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 bisogno 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

LIVELLI GESTIONALI DI UN'ORGANIZZAZIONE

Nell'ambito di un organizzazione, possono essere individuati sostanzialmente tre livelli decisionali: strategico, tattico e operativo. Nello sviluppo di un progetto software è fondamentale considerare il modo in cui è gestita l'organizzazione e, soprattutto, il livello in cui il software avrà il suo impatto.

I vari livelli e le tipologie di applicazioni corrispondenti sono schematicamente mostrati nella seguente tabella.

Livello decisionale	Focalizzazione delle decisioni	Applicazioni dei Sistemi Informativi	Soluzioni di Information Technology
Strategico	Strategie di supporto agli obiettivi organizzativi di lungo periodo	Analisi del mercato e delle vendite. Pianificazione di prodotto, valutazione delle prestazioni	Data Mining, Gestione della conoscenza aziendale
Tattico	Politiche di supporto agli obiettivi di breve periodo e all'allocazione delle risorse	Analisi di bilancio. Previsioni retributive. Ordinamento delle giacenze e servizi al cliente	Data warehouse, elaborazione analitica, fogli elettronici
Operativo	Attività giornaliere di staff e supporto alla produzione	Stipendi, fatturazioni, acquisti, contabilità	Database, elaborazione transazioni, generatori di applicazioni

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 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 Architettuale* 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 architettuale
- Progetto di dettaglio
- Implementazione
- Integrazione
- Manutenzione

Parallelamente a queste sette fasi, particolare importanza rivestono le fasi di *Pianificazione delle Attività* e di *Testing*.

In seguito analizzeremo ciascuna di queste fasi più in dettaglio.

DETERMINAZIONE DEI REQUISITI

E' forse una delle fasi più importanti e delicate del processo. L'obiettivo è quello di fornire una definizione 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 tecniche 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 p 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 specifiche. Di solito vengono realizzati in aggiunta alle interviste. Alcune regole di base per i questionari sono: evitare i quesiti aperti; utilizzare quesiti 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 prototipi evolutivi.

Sviluppo cooperativo delle applicazioni (JAD): in questo caso 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 tecniche: Prototipi evolutivi, Strumenti CASE, Gruppi di specialisti con strumenti avanzati di sviluppo, JAD interattivo, Time boxing.

SPECIFICA DEI REQUISITI

Questa fase comprende la modellazione dei requisiti acquisiti nella fase precedente e la definizione formale dei vari aspetti del sistema.

Tra i modelli di specifica, occorre prevedere: 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 valore 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 operativa 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 diagrammi di stato (statechart) che specificano stati e transizioni (causate da eventi).

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

IMPLEMENTAZIONE

In questa fase, scelta l'architettura, definite le classi e le loro interrelazioni, si rende tutto operativo. Spesso in questa fase risulta cruciale il progetto del database.

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

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 porlo ai livelli richiesti; Manutenzione evolutiva, che fa fronte a nuove richieste o esigenze dei clienti.

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 operazionale, 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

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
-