****

# ISW – Gruppo 01 – TeamSoftwareRevolution

Quesito CVS

Redattori: Carta Stefano 48554;

Carta Fabio 65006;

Argiolas Alessandro 49062;

Bertulu Giovanni 49044;

Desogus Omar 48975;

Fadda Luca 49006;

Zucca Luigi 65053;

# Sommario

1. Introduzione
2. Progettare un software
   1. UML
   2. Processo
   3. Ciclo di vita di un software
3. I modelli
   1. Modello a spirale
   2. Modello agile
   3. Modello FDD
4. Confronti

1.Introduzione

Nel presente elaborato, viene analizzato il ciclo di vita del software e successivamente definiti diversi modelli di progettazione software.

Conclusa l’analisi, vengono portate in evidenza alcune peculiarità dei modelli messi a confronto tra loro.

2.Progettare un software

La maggior parte delle metodologie per la creazione di un prodotto software consistono in un **linguaggio di modellazione** e in un **processo**.

Il linguaggio di modellazione è usato per esprimere le caratteristiche del progetto.

Processo: un insieme strutturato e organizzato di attività che si svolgono per raggiungere un risultato.

Elementi di un processo:

* + Attività
  + Persone (ruoli, sviluppatori)
  + Risorse (computer, strumenti, rete,...)
  + Manufatti (requisiti, progetti, codice, test,...)
  + Eventi

#### 2.1 UML

**UML** è un linguaggio di modellazione usato per capire e descrivere le caratteristiche di un nuovo sistema o di uno esistente, indipendentemente dal processo di sviluppo.

La forza dell’Unified Modeling Language consiste nel fatto che il processo di disegno del sistema può essere effettuata in modo tale che i clienti, gli analisti, i programmatori e chiunque altro sia coinvolto nel sistema di sviluppo possa capire ed esaminare in modo efficiente il sistema e prendere parte alla sua costruzione in modo attivo.

Un sistema software grazie al linguaggio UML viene **disegnato professionalmente** e documentato ancor prima che ne venga scritto il relativo codice, da parte degli sviluppatori. Si sarà così in grado di conoscere in anticipo il risultato finale del progetto su cui si sta lavorando.

Grazie alla documentazione del linguaggio UML diviene ancora più **facile effettuare eventuali modifiche future** al codice. Questo, ancora, a tutto beneficio dei costi di mantenimento del sistema.

## 2.2 Il processo

Chi lavora allo sviluppo o alla modifica di un software, adotta necessariamente un certo modo di *relazionarsi con i propri clienti/utenti* per organizzare il proprio lavoro e per la scelta delle tecniche da utilizzare.

Quindi, adottano un processo.

Ogni team di lavoro ha un proprio processo di sviluppo, cioè un proprio modo di lavorare, basato sulla propria esperienza, cultura e sul contesto organizzativo in cui si trova ad operare.

## 2.3 Ciclo di vita di un software

Ogni processo di sviluppo ha i propri **pregi** ed i propri **limiti**.

Un processo di sviluppo che non soddisfa le esigenze del progetto è destinato a fallire o comunque a non soddisfare il cliente/utente.

Il ciclo di vita di un software definisce un modello per il software, dalla sua concezione iniziale fino al suo sviluppo completo, al suo rilascio, alla sua successiva evoluzione, fino al suo ritiro.

Ogni fase prevede il completamento di 1 o più deliverable.

Deliverable è un prodotto tangibile e verificabile (Es: documento o

componente informatico).

Il ciclo di sviluppo del software, nella maggior parte dei casi è iterativo e ogni iterazione produce una sua release:

* + Specifica dei requisiti
  + Studio di fattibilità
  + Analisi e Design
  + Implementazione
  + Integrazione e testing

3. I modelli

Modelli di Processo Software (talora detti ancora CVS) definiscono la struttura di massima di un processo software, indicando le fasi in cui si articola e i criteri di successione.

Un Modello di processo software è una rappresentazione astratta di un processo, fornisce una descrizione del processo da una particolare prospettiva.

Il processo di sviluppo deve essere modellato esplicitamente, per poter essere gestito e monitorato.

I modelli di processo software sono descrizioni precise e formalizzate delle attività, delle trasformazioni, dei deliverables e degli eventi per realizzare e/o ottenere l’evoluzione del software.

## 3.1 Modello a spirale

Prova a unire i vantaggi del modello evolutivo con quelli del modello a cascata.

E’ un modello nel quale il processo viene diviso in quattro fasi principali.

Ogni fase inizia con un obiettivo di design, e termina con un’interazione col cliente, che rivede il progresso ottenuto.

Un momento di analisi e di ingegnerizzazione è presente in ogni fase del progetto.

Ripetizione ciclica di task regions:

* Determinazione degli obiettivi
* Identificazione e riduzione dei rischi
* Sviluppo e verifica della fase
* Pianificazione della fase successiva

Determinazione degli obiettivi:

Committente e fornitore lavorano a stretto contatto per definire in maniera corretta e univoca cosa bisogna realizzare e come.

Identificazione e riduzione dei rischi:

Si identificano e si analizzano i problemi e i rischi associati al progetto al fine di determinare delle strategie per controllarli.

Sviluppo e verifica della fase:

Questa fase è caratterizzata dalla vera e propria realizzazione del progetto, essa comprende la codifica e la verifica dei requisiti.

Pianificazione della fase successive:

Questa fase richiede l’approvazione del lavoro da parte del committente. Se il prodotto non soddisfa effettivamente i requisiti richiesti, è necessario impostare un nuovo ciclo di attività.

Una caratteristica importante di questo modello è il fatto che i rischi vengono presi seriamente in considerazione e che ogni fine ciclo produce una deliverables. In un certo senso può **essere visto come un modello a cascata iterato più volte.**

Vantaggi:

Rende esplicita la gestione dei rischi focalizzando l’attenzione sul riuso.

Aiuta a considerare gli aspetti della qualità e integra sviluppo e manutenzione.

Svantaggi:

Aumento nei tempi di sviluppo e delle persone con capacità di identificare i rischi. Richiede una gestione maggiore del team di sviluppo e quindi anche un costo maggiore.

## 3.2 Modello agile

Con la necessità di produrre in tempi sempre più ridotti risultati che possano migliorare la competitività, anche in presenza di budget decrescenti, viene favorito lo sviluppo di nuove metodologie.

I pricipi :

* + Massima priorità soddisfare il cliente rilasciando software di valore, fin da subito e in maniera continua.
  + Accogliere cambiamenti nei requisiti, anche a stadi avanzati dello sviluppo.
  + I processi agili sfruttano il cambiamento a favore del vantaggio competitivo del cliente.
  + Consegniamo frequentemente software funzionante, preferendo i periodi brevi.
  + Committenti e sviluppatori devono lavorare insieme quotidianamente per tutta la durata del progetto.
  + Fondiamo i progetti su individui motivati e confidiamo nella loro capacità di portare il lavoro a termine
  + Una conversazione faccia a faccia è il modo più efficiente e più efficace per comunicare con il team ed all'interno del team.
  + Il software funzionante è il principale metro di misura di progresso. Sviluppatori e utenti dovrebbero essere in grado di mantenere indefinitamente un ritmo costante.
  + La continua attenzione all'eccellenza tecnica e alla buona progettazione esaltano l'agilità.
  + La semplicità - l'arte di massimizzare la quantità di lavoro non svolto - è essenziale.
  + Le architetture, i requisiti e le progettazioni migliori emergono da team che si auto-organizzano.
  + A intervalli regolari il team riflette su come diventare più efficace, dopodiché regola e adatta il proprio comportamento di conseguenza.

## 3.3 Modello FDD

FDD è un processo di sviluppo iterativo ed incrementale guidato da modelli, caratterizzato da brevi iterazioni, che suddivide il progetto in cinque fasi, dette processi.

Propone una robusta fase di analisi e progettazione integrata con un modello di sviluppo agile.

Il processo di sviluppo software di FDD consiste di 5 attività principali. Le prime tre servono a stabilire la forma del modello; le rimanenti due lo realizzano per ciascuna Feature. Per un accurato monitoraggio delle attività di progetto, l'avanzamento lavori di ogni singola Feature viene monitorato attraverso appositi Milestones.

1. Sviluppo del Modello

Il progetto ha inizio con un esame approfondito dell'ambito e del contesto del sistema, a cui fa seguito un'analisi dettagliata di ciascuna dominio. I modelli così realizzati, suddivisi in piccoli gruppi per ogni dominio, vengono sottoposti ad una revisione. I modelli rivisti confluiscono in un modello generale, la cui forma è comunque suscettibile di aggiustamenti successivi.

2. Lista delle Feature

Le informazioni raccolte nel corso della precedente fase sono utilizzate per definire la lista delle funzionalità (features) da sviluppare, ottenute decomponendo i vari domini in aree, ciascuna contenenti attività di business. La lista delle Features si trova a quest'ultimo livello.

3.Pianificazione per Feature

Ottenuta la lista delle Features, il passo successivo è la definizione del piano di sviluppo,suddividendo le Features in Classi e assegnando ogni classe ad un capo programmatore.

4. Disegno per Feature

Per ogni Feature viene prodotto un documento di Disegno. Il capo programmatore sviluppa il piano di dettaglio per ciascuna Feature e perfeziona il modello complessivo. Il disegno così realizzato viene infine sottoposto a revisione.

5. Costruzione per Feature

Per ogni Feature, dopo il disegno si procede alla produzione del codice relativo. Dopo aver superato i test e l'ispezione del codice, ogni singola Feature viene caricata sulla Build principale dell'applicazione.

Per assicurare un efficace controllo dell'avanzamento dei lavori di ogni singola feature, per ciascuna vengono definite sei **Milestones** sequenziali, le prime tre da completare entro la quarta fase (Disegno per Feature) e le restanti nell'ultima fase (Costruzione per Feature).

Per facilitare il controllo dell'avanzamento, ad ogni Milestone viene assegnata una percentuale di avanzamento:

* Analisi di dominio 1%
* Disegno 40%
* Revisione del disegno 3%
* Programmazione 45%
* Revisione del codice 10%
* Fine dello sviluppo 1%

4. Confronti

Il modello a Spirale è spesso definito anche meta-modello perché sfrutta sia il modello a cascata che quello prototipale. Qui troviamo una differenza fondamentale rispetto all’Agile: lo sviluppo non è incrementale, ma vengono sfruttati uno o più cicli basati su prototipi per abbassare i rischi e poi passare, solo successivamente, allo sviluppo vero e proprio che avviene in chiave waterfall.

A differenza del modello a Spirale, lo sviluppo in chiave Agile è Iterativo ed Incrementale, basato su iterazioni/sprint brevi (tipicamente 2-4 settimane) ed è incentrato sui Valori e sui Principi del Manifesto Agile.

Al di là delle differenze, esistono comunque alcuni punti di incontro tra il modello a Spirale e l’approccio Agile:

* In entrambi le stime (costi, tempi, ecc..) sono aggiornate ripetutamente
* La gestione del rischio è uno dei fattori chiave di entrambi i processi
* I Cambiamenti sono accettati con più elasticità
* Il monitoraggio delle attività è flessibile e dinamico

Per quanto riguarda il modello Feature Driven Development, esso rientra nella categoria dei metodi Agili.

Questo, in relazione con il modello a Spirale, comporta tutte le differenze e e le concomitanze precedentemente trattate.