💚

APS - Processi software

Created	@May 23, 2024
Tags	APS

2.1 Processi per lo sviluppo del software

Un processo per lo sviluppo del software (o processo software) definisce un approccio disciplinato per la costruzione, il rilascio e la manutenzione del software.

1. Definisce chi fa che cosa, quando e come per raggiungere un certo obiettivo
2. “Cosa” sono le attività? “Chi” sono i ruoli? “Come” sono le metodologie? “Quando” riguarda l’organizzazione temporale delle attività?

Intuitivamente, tutti i processi software sono organizzati attorno a uno stesso numero di attività fondamentali:

1. Specifica dei requisiti
2. Analisi
3. Progettazione
4. Implementazione
5. Validazione e verifica
6. Rilascio
7. Manutenzione
8. Gestione del progetto.

2.2 Tipologie di processi software

2.2.1 Processi agili e basati sul piano

- I processi orientati al piano sono processi in cui tutte le attività del processo sono pianificate in anticipo e i progressi sono misurati rispetto a questo piano.
- Nei processi agili, la pianificazione è incrementale ed è più facile modificare il processo per riflettere le mutevoli esigenze dei clienti.

In pratica, la maggior parte dei processi pratici include elementi di entrambi gli approcci, quello guidato dal piano e quello agile.
Non esistono processi software giusti o sbagliati.

2.2.2 Modelli di processo software

- **Il modello a cascata**
  Modello guidato dal piano. Fasi separate e distinte di specifica e sviluppo.
- **Sviluppo incrementale**
  Specifica, sviluppo e validazione si alternano. Può essere guidato dal piano o agile.
- **Integrazione e configurazione**
  Il sistema viene assemblato a partire da componenti configurabili esistenti. Può essere pianificato o agile.
  Il riutilizzo è oggi l'approccio standard per la costruzione di molti tipi di sistemi aziendali.

In pratica, la maggior parte dei sistemi di grandi dimensioni viene sviluppata utilizzando un processo che incorpora elementi di tutti questi modelli.

Quale Scegliere?

Ecco alcuni dei motivi per cui esistono diversi modelli di processi di sviluppo software:

1. Complessità del progetto: La complessità del progetto influenza il livello di formalità e struttura necessari nel processo di sviluppo.
2. Dimensione del team: I team di grandi dimensioni hanno bisogno di un processo più formale rispetto ai team di piccole dimensioni.
3. Budget e tempistiche: I progetti con budget e tempistiche limitati hanno bisogno di un processo più snello rispetto ai progetti con budget e tempistiche flessibili.
4. Rischio: I progetti ad alto rischio hanno bisogno di un processo più rigoroso rispetto ai progetti a basso rischio.

2.2.2.1 Il processo a cascata

Il processo software con ciclo di vita a **cascata** (o sequenziale) è, in prima approssimazione, basato su uno svolgimento sequenziale delle diverse attività dello sviluppo del software.

Il modello a cascata prevede fasi distinte:

- Analisi e definizione dei requisiti
- Progettazione del sistema e del software
- Implementazione e test delle unità
- Integrazione e test di sistema
- Funzionamento e manutenzione

Il principale svantaggio del modello a cascata è la difficoltà di accogliere i cambiamenti dopo che il processo è stato avviato. In linea di principio, una fase deve essere completata prima di passare alla fase successiva.

In media, il 45% delle caratteristiche e delle funzionalità identificate usando un approccio a cascata per i requisiti non viene mai utilizzato, e le stime iniziali a cascata dei tempi e dei costi variano fino al 400% dai valori effettivi finali.

La suddivisione inflessibile del progetto in fasi distinte rende difficile rispondere alle mutevoli esigenze del cliente (Figura 2.1)

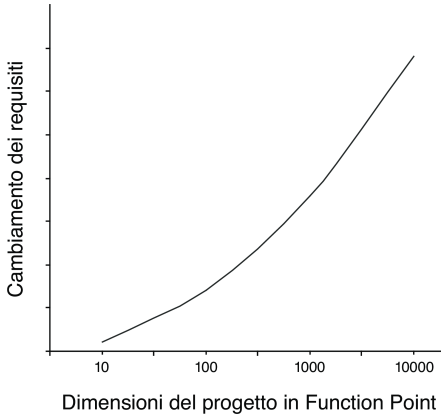


Figura 2.1: Percentuale di cambiamento in progetti software al variare delle dimensioni.

- Pertanto, questo modello è appropriato solo quando i requisiti sono ben compresi e le modifiche saranno piuttosto limitate durante il processo di progettazione.

- Pochi sistemi aziendali hanno requisiti stabili.

2.2.2.2 Sviluppo iterativo, incrementale e evolutivo

Una pratica fondamentale di molti processi software moderni (come UP e Scrum) è lo **sviluppo iterativo**. Il sistema cresce in modo incrementale nel tempo, iterazione dopo iterazione, e pertanto questo approccio è noto anche come **sviluppo iterativo e incrementale**. Poiché il feedback e l'adattamento fanno evolvere le specifiche e il progetto, è noto anche come **sviluppo iterativo ed evolutivo**.

In questo approccio al ciclo di vita, lo sviluppo è organizzato in una serie di mini-progetti brevi, di lunghezza fissa (per esempio di tre settimane) chiamati **iterazioni**; il risultato di ciascuna iterazione è un sistema eseguibile, testato e integrato, anche se parziale.
Ciascuna iterazione comprende le proprie attività di analisi dei requisiti, progettazione, implementazione e test.

Nel tempo, attraverso il feedback (provenienti dalle attività di sviluppo, test, sviluppatori e clienti) iterativo e l'adattamento, il sistema sviluppato evolve e converge verso i requisiti corretti e il progetto più appropriato. L'instabilità dei requisiti e del progetto tende a diminuire nel tempo. Nelle iterazioni finali è difficile, ma non impossibile, che si verifichi un
cambiamento significativo dei requisiti.

Vantaggi	**Svantaggi**
- Minor probabilità di fallimento del progetto - Miglior produttività - Percentuali più basse di difetti	- Il processo non è visibile. - La struttura del sistema tende a degradarsi con l'aggiunta di nuovi incrementi.

Una buona pratica dello sviluppo iterativo è il **timeboxing**: le iterazioni hanno una lunghezza fissata.

- Molti processi iterativi raccomandano una lunghezza da 2 a 6 settimane
- Se troppo lungo, aumenta il rischio del progetto

- Se troppo corto, non consentono di completare del lavoro sufficiente per ottenere feedback significativi

La durata di un'iterazione, una volta fissata, non può cambiare – un'iterazioni di lunghezza fissata è detta **timeboxed**

L'adozione dello sviluppo iterativo richiede che il software venga realizzato in modo flessibile affinché l'impatto dei cambiamenti sia basso o tale che l'impatto dei cambiamenti sul sistema sia basso
A tal fine il codice sorgente deve essere facilmente
modificabile, **comprensibile** e **flessibile**.

Durante ciascuna iterazione, i requisiti su cui operare vengono prima **fissati** (durante la pianificazione iterativa) e poi bloccati (non possono cambiare durante l'interazione).

Durante ciascuna iterazione, il team di sviluppo può lavorare al suo meglio

- Poiché i requisiti sono bloccati, il team non può essere interrotto o disturbato durante l'iterazione
- I committenti possono interagire con il team di sviluppo a termine di ogni iterazione

Il team se termina tutte le task può cambiare il piano dell'iterazione.

2.3 Unified Process

Il processo unificato (UP) è un processo iterativo diffuso per lo sviluppo del software orientato agli oggetti. UP è pilotato dai casi d'uso (requisiti) e dai fattori di rischio, incentrato sull'architettura, iterativo, incrementale e evolutivo.

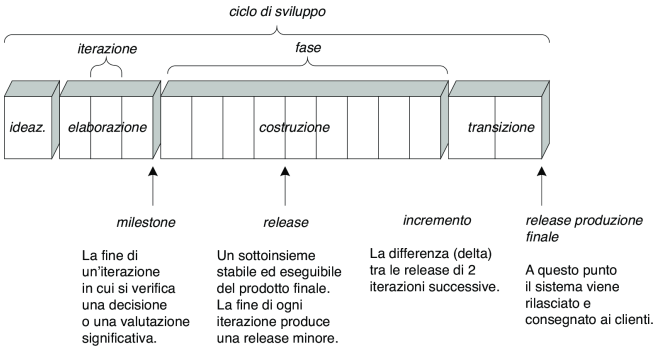
2.3.1 Pratiche fondamentali di UP

1. Affrontare le problematiche di rischio maggiore e valore elevato nelle interazioni iniziali
2. Impegnare gli utenti continuamente sulla valutazione, il feedback e i requisiti
3. Creare un'architettura coesa nelle interazioni iniziali
4. Verificare continuamente la qualità; testare presto, spesso in modo realistico
5. Fare della modellazione visuale con UML
6. Gestire attentamente requisiti
7. Gestire richieste di cambiamento e le configurazioni

2.3.2 Fasi di UP

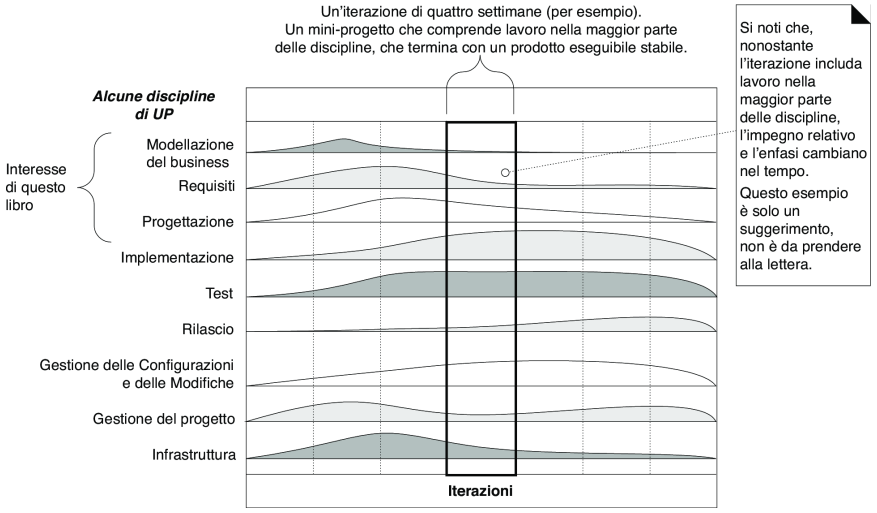
Un progetto UP organizza il lavoro e le iterazioni in quattro fasi temporali principali successive.

1. **Ideazione**. Visione approssimativa, studio economico, portata, stime approssimative dei costi e dei tempi.
2. **Elaborazione**. Visione raffinata, implementazione iterativa del nucleo dell'architettura, risoluzione dei rischi maggiori, identificazione della maggior parte dei requisiti e della portata, stime più realistiche.
3. **Costruzione**. Implementazione iterativa degli elementi rimanenti, più facili e a rischio minore, e preparazione al rilascio.
4. **Transizione**. Beta test, rilascio.



Termini di UP orientati all'organizzazione temporale dei progetti.

Anche se ogni iterazione può prevedere tutti i flussi di lavoro, la collocazione dell'iterazione all'interno del ciclo di vita del progetto determina una maggiore enfasi su uno dei flussi di lavoro.



Discipline di UP.

Tra gli elaborati e le pratiche di UP, quasi tutto è opzionale. Ciò premesso, alcune pratiche e principi di UP sono fissi, come lo sviluppo iterativo e guidato dal rischio e la verifica continua della qualità.

Tuttavia, esaminando a fondo UP si nota che tutte le attività e gli elaborati (modelli, diagrammi, documenti, ...) sono *opzionali*, con l'ovvia esclusione del codice.

N.B: UP non è stato pensato per essere pesante o non agile. Molti elementi sono opzionali. UP può essere applicato in modo agile (UP agile).

2.4 Agile

I metodi per lo **sviluppo agile** di solito applicano lo sviluppo iterativo ed evolutivo, con iterazioni brevi e timeboxed, fanno uso della pianificazione adattiva, promuovono le consegne incrementali e comprendono altri valori e pratiche che incoraggiano l'*agilità*, ovvero una risposta rapida e flessibile ai cambiamenti.

Non è possibile dare una definizione precisa di **metodo agile**, poiché le pratiche adottate variano notevolmente da metodo a metodo. Tuttavia, una pratica di base condivisa dai vari metodi è quella che prevede iterazioni brevi, con un raffinamento evolutivo dei piani, dei requisiti e del progetto.

Gli analisti e i modellatori esperti conoscono il *segreto della modellazione*:

> Lo scopo della modellazione (abbozzare UML, ...) è principalmente quello di comprendere, non di documentare.

Questo modo di vedere, coerente con i metodi agili, è stato chiamato **modellazione agile.** Esso è basato su un insieme di pratiche e valori:

1. Adottare un metodo agile non significa evitare del tutto la modellazione;
2. Lo scopo della modellazione e dei modelli è principalmente quello di agevolare la *comprensione* e la *comunicazione*, non di documentare.
3. La modellazione non va fatta da soli, ma piuttosto a coppie (o in tre) alla lavagna, tenendo presente che lo scopo della modellazione è scoprire, capire e condividere ciò che si è compreso.
4. Si tenga presente che tutti i modelli saranno imprecisi, e il codice o il progetto finali differiranno, talvolta anche in modo notevole, dal modello abbozzato.

2.4.1 Scrum

Scrum è un metodo agile che consente di sviluppare e rilasciare prodotti software con il più alto valore per i clienti, nel più breve tempo possibile.

Scrum è un approccio iterativo e incrementale allo sviluppo del software. Ciascuna iterazione, chiamata uno **Sprint**, ha una durata fissata, per esempio di due settimane. Le iterazioni sono timeboxed, e dunque non vengono mai estese.

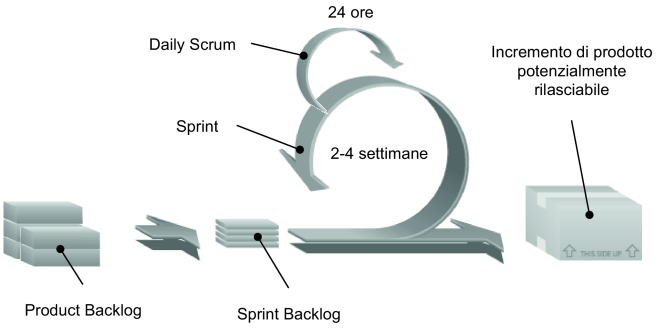
In Scrum ci sono solo tre ruoli.

1. Il **Product Owner** (il "proprietario" del prodotto) definisce le caratteristiche del prodotto software da realizzare e specifica le priorità tra queste caratteristiche. Il suo obiettivo è massimizzare il valore del prodotto.
   Il Product Owner definisce le caratteristiche del prodotto da realizzare nel **Product Backlog. D**escrive tutte le caratteristiche del prodotto; di iterazione in iterazione questo elaborato viene aggiornato, e descrive le cose che devono essere ancora fatte per completare il prodotto.
2. Il **Development Team** (la "squadra" di sviluppo, chiamata anche solo Team) è composto di solito da una manciata di persone, che possiedono le competenze necessarie per sviluppare il software. Il Team è auto-organizzato e auto-gestito, e opera con un grado di autonomia e libertà molto alto.
3. Lo **Scrum Master** (il "maestro" di Scrum) aiuta l'intero gruppo ad apprendere e applicare Scrum, al fine di ottenere il valore desiderato. Lo Scrum Master non è il manager del Team; è piuttosto un istruttore e una guida, che serve, aiuta e protegge il Team.

Complessivamente, il Team di sviluppo, il Product Owner e lo Scrum Master formano un **Team Scrum**.

All'inizio di ciascuno Sprint, il Team seleziona dal Product Backlog un insieme di voci da sviluppare durante quell'iterazione. Questa scelta definisce lo **Sprint Goal**, ovvero l'obiettivo di sviluppo del Team per questo Sprint. Inoltre il Team compila lo **Sprint Backlog** ("lavoro arretrato" dello Sprint), che specifica l'insieme dei compiti

dettagliati da svolgere per raggiungere lo Sprint Goal.



Sviluppo iterativo e incrementale con Scrum

Ogni giorno, all'inizio della giornata, il Team si riunisce brevemente in un **Daily Scrum** (Scrum "giornaliero") per verificare i propri progressi e per decidere i passi successivi necessari per completare il lavoro rimanente. Dopodiché, il Team si mette al lavoro e compie tutte le attività necessarie per l'analisi, la progettazione, la costruzione, l'integrazione e la verifica del software.

Il risultato di ciascuno Sprint deve essere un prodotto software funzionante, chiamato "incremento di prodotto potenzialmente rilasciabile".

Alla fine dello Sprint, nella **Sprint Review** ("revisione" dello Sprint) il Product Owner e il Team presentano alle diverse parti interessate l'incremento di prodotto software che è stato sviluppato, e ne fanno una dimostrazione. Lo scopo della Sprint Review è ottenere un feedback su quanto è stato fatto, anche per poter decidere che cosa è utile fare nel prossimo Sprint.