Sviluppo Applicazioni Software 13/14 - Riassunto su UP

Mattia Cerrato

2 giugno 2014

1 Purpose

Riempire i buchi nelle slides, un po' troppo sintetiche

2 Ingegneria del Software

Produre software un'operazione complessa e differente da esperienza ad esperienza, ma esistono diverse organizzazioni e aziende che si preoccupano di definire standard per i processi di sviluppo: il loro scopo Ë quello di migliorare la qualit‡† del prodotto finale e del procedimento stesso di creazione.
L'organizzazione **Object Management Group** - OMG raccoglie al suo interno i principali sviluppatori di software: tra i suoi contributi c'Ë anche UML. **UML** è una notazione visuale per la specifica di software, e un modello UML si compone di un insieme di diagrammi. È uno standard di OMG dal 97.

3 Unified Process

3.1 Introduzione

UP, Unified Process, non è invece uno standard ma un processo di sviluppo, che al suo interno utilizza alcuni tra i diagrammi UML. Per processo di sviluppo si intende un'idea o schema generale di procedimenti da seguire passo passo per sviluppare il software, che però aziende e sviluppatori sono invitati a seguire. Le caratteristiche salienti di UP sono:

- È iterativo ed incrementale: le iterazioni sono le unità di tempo fondamentali di UP. All'interno di un'iterazione gli sviluppatori dovranno compiere certe operazioni decise a priori. È fondamentale che le iterazioni non superino l'ordine delle settimane se si vuole mantenere un approccio agile.
- Le iterazioni iniziali sono guidate dal rischio (nel senso: si prova ad eliminarlo), dalle richieste del cliente e dall'architettura che desideriamo per il software: inoltre si possono cominciare a programmare le caratteristiche che il cliente ritiene più importanti.
- È personalizzabile e flessibile, e **può** essere applicato con un approccio cosiddetto agile.

3.2 Agilità

Riguardo l'**agilità**: è vero che UP incoraggia l'utilizzo di pratiche agili, che sono state introdotte però da altre metodologie. Sarebbe bene come già detto utilizzare iterazioni corte e progettate a monte, in cui c'è un progressivo raffinamento della documentazione, in modo da evitare una mentalità waterfall. Inoltre si possono utilizzare i principi di Scrum, in cui il lavoro è auto-organizzato programmatore per programmatore, o ancora la programmazione a coppie e lo sviluppo guidato dai test di XP. Quando c'è buona comprensione del software, è meno fondamentale la documentazione.

3.3 Rischio

Di cosa si parla quando si parla di **rischio**? Si intende l'insieme dei problemi da risolvere all'interno del software, che se non si risolvessero potrebbero portare a uno software di scarso successo che non risolve i problemi del cliente. In un'applicazione web per la gestione di un conto corrente bancario, la sicurezza e la buona fondatezza delle operazioni sono fondamentali, ma sono rischi anche questioni interne al team di sviluppo, come la necessità di utilizzare tool conosciuti dai programmatori per evitare il rischio di dover fare altre assunzioni o contratti a progetto.

3.4 Di cosa si compone UP

UP si compone di alcune indicazioni di massima su come affrontare lo sviluppo del progetto:

- Organizzazione del piano di sviluppo per fasi sequenziali
- Indicazioni sulle **attività** da svolgere, a quali **discipline** queste appartengono e sulle relazioni tra tali attività e discipline
- Un insieme di ruoli da far assumere ai componenti del team
- Un insieme di artefatti documentativi da produrre

4 Le fasi di UP

Famosamente, le fasi di UP sono quattro.

- Ideazione: dove il team formerà una prima visione approssimativa del software finale, facendo uno studio economico e di portata, dunque studiando i costi e i tempi di sviluppo. La milestone raggiunta sono gli **Obiettivi**, artefatto che riassume quali sono i problemi che risolverà il software.
- Elaborazione: dove il team raffinerà la sua visione del software e comincerà con iterazioni implementative delle funzionalità più importanti, risolvendo dunque i rischi maggiori. Inevitabilmente si assumerà più coscienza degli effettivi requisiti e della portata. La milestone qui è di tipo **Architetturale**, nel senso che l'architettura generale del software dovrà essere ben definita.
- Costruzione: dove il team implementerà con iterazioni successive gli elementi che rimangono, che però sono quelli con rischio minore. Inoltre si preparerà alla fase di rilascio e deployment. La milestone qui è la **Capacità Operazionale**, la capacità del software di funzionare secondo le specifiche richieste dal cliente.
- Transizione: dove il team effettuerà il beta testing e rilascierà il prodotto. La milestone qui è la **Release**.

5 Le discipline di UP

Le varie attività compiute dal team sono organizzate in **discipline**: ogni artefatto che produciamo o azione che compiamo ricade all'interno di una certa disciplina. Questo è utile perchè capiremo che in ogni fase sarà bene dedicare più tempo ad alcune discipline e meno ad altre.

- Discipline ingegneristiche: Modellazione del Business (modellare il problema e il suo ambito), Requisiti, Progettazione (analisi dei requisiti e progetto architetturale), Implementazione, Testing e Rilascio

- Discipline di Supporto: Gestione delle configurazioni e del cambiamento (manutenzione del progetto durante il progetto), Gestione del progetto (pianificazione delle attività future), Infrastruttura-Environment (supportare il team di sviluppo riguardo processi e tool utilizzati).

Se le fasi sono assolutamente sequenziali e ad una iterazione di Ideazione certamente non segue una di Costruzione, le discipline non sono sequenziali: ad ogni iterazione svolgeremo idealmente ognuna di esse, magari in quantità diverse.

6 UP e UML

È vero che UP usa UML come linguaggio di modellazione: ma non è necessario usare tutti i diagrammi. UP è personalizzabile, e i vari diagrammi possono essere aggiornati di iterazione in iterazione. Quello che ci dice UP è semplicemente quando utilizzare un diagramma. Il documento che riporterà le scelte del team in merito a quali diagrammi UML utilizzare è lo Scenario di Sviluppo.

7 Cosa non è UP

Alcuni controesempi per vedere come non si usa UP.

- Il team definisce, o ci prova, tutti i requisiti del software prima di cominciarne la progettazione o l'implementazione
- Il team passa giorni e giorni a modellare con UML prima di programmare
- Il team crede che l'ideazione sia solamente la fase di raccolta dei requisiti, l'elaborazione solamente quella di progettazione, la costruzione solamente quella di scrittura del codice
- Il team usa iterazioni lunghe mesi
- Il team tenta di pianificare il progetto nei minimi dettagli sin dall'inizio, provando a prevedere tutte le iterazioni ed il loro contenuto.

8 La fase di Ideazione

L'obiettivo della fase di Ideazione è quello di stabilire una visione comune del problema e di capire la portata del progetto, attraverso lo studio di fattibilità. La durata dell'ideazione è breve, il tempo di un workshop o un paio di incontri. Cosa si fa durante l'ideazione?

- Si analizza circa il 10% dei casi d'uso nel dettaglio
- Si analizzano i requisiti non funzionali più critici
- Si realizza uno studio economico, per capire la portata del progetto e i costi
- Si prepara l'ambiente di sviluppo

E quali sono gli artefatti da produrre?

1. Modelo dei casi d'uso: costituisce i requisiti funzionali. Si identificano la maggior parte dei nomi dei casi d'uso, dettagliandone circa il 10%
2. Specifiche supplementari: servono ad evidenziare i requisiti non funzionali
3. Visione e studio economico: si riassumono gli obiettivi e i vincoli, si fa uno studio economico dei costi, si riassume il progetto
4. Glossario: dove vengono riportati i termini rilevanti e di significato peculiare al progetto
5. Lista dei rischi e piano di gestione dei rischi
6. Prototipo e proof-of-concept
7. Piano dell'iterazione: linee di massima su cosa si farà all'interno della prossima iterazione
8. Piano delle fasi e piano di sviluppo del software: linee guida su quanto durerà ogni fase UP.
9. Scenario di sviluppo: scelta dei diagrammi UML da utilizzare e personalizzazione di UP

Sono molti! Si possono scegliere solo quelli che si ritiene aggiungano valore al progetto. Inoltre, UP è incrementale: non è necessario terminarli tutti durante l'ideazione.

9 Requsiti Evolutivi

I requsiti evolutivi sono le capacità alle quali il sistema deve essere conforme, o per meglio dire alcuni principi da tenere in mente durante lo sviluppo. In UP sono presenti delle

best pracices per affrontare i requisiti che cambiano nel tempo. Tale modello è detto FURPS+:

- Funcional: caratteristiche comportamentali, capacità
- Usability: fattori umani, help, documentazione
- Reliability: frequenza dei guasti, capacità di riparazione
- Performance: tempo di risposta, uso delle risorse, throughput
- Supportability: adattamento, manuntenzione, configurabilità
- +: requisiti complementari e secondari (risorse, linguaggi, interfacce, problemi legali)