Riassunto Ingegneria Del Software - Federico Pomponii

Modello - MOD.2

Per modello si intende una rappresentazione di un oggetto o di un fenomeno reale che riproduce caratteristiche o comportamentei ritenuto fondamentali per il tipo di ricerca che si sta svolgendo. Per l'Ingegneria del software un modello costituisce una visione semplificata di un sistema che rende il sistema stesso

- Più acessibile alla comprensione e alla valutazione
- Facilità il trasferimento di informazione e collaborazione tra persone

Modelli del sistema

Attraverso l'uso di diagrammi si cerca di rappresentare ***modelli del sistema*** per:

- Descrivere in modo conciso e preciso conoscenze sul problema
- Individuare rischi e scelte progettuali

I linguaggi per la descrizione dei modelli si basano su livello di astrazione più elevati rispetto ai comuni linguaggi macchina.

Tracciabilità

In qualsiasi direzione si percorra la sequenza di modelli generati, deve essere possibile mappare uno o più elementi in un modello in uno o più elementi in un altro.

Linguaggi di modellazione

Un linguaggio di modellazione è un linguaggi ***semi-formale*** che può essere utilizzato per descrivere un sistema di qualche natura. Quello che si esprime attraverso i diagrammi è una rappresentazione del modello creata attraverso l'uso di un linguaggio (Ad esempio ***UML, OPM o XML***)

Modelli e codice

Tipicamente il disallineamento tra modello e codice avviene già durante la fase di implementazione.

Modelli di processo

Un processo di sviluppo è un insieme ordinato di passi fine alla produzione dell'output desiderato a partire dai requisiti in ingresso. Le generiche fasi sono:

- ***Specifica*** : cosa il sistema dovrebbe fare e vincoli di sviluppo.
- ***Sviluppo*** : produzione del sistema software
- ***Validazione*** : testare che il sistema sviluppato sia quello che il committente voleva.
- ***Evoluzione*** : cambiamenti nel prodotto in accordo a modifiche dei requisiti o incremento delle funzionalità del sistema.

Modelli di processo di sviluppo

- Modello a cascata
- Modelli evolutivi
- Sviluppo incrementale-Iterativo

- Modello a spirale
- Modelli specializzati
    - Sviluppo a componenti
    - Modello dei metodi formali
    - Sviluppo aspect-oriented
    - Sviluppo model driven
    - Unified Process (UP - RUP)

Modello a cascata

Fasi distinte, in cascata tra loro, con retroazione finale. Il modello si fonda sul presupposto che ogni fase deve essere svolto in maniera esaustiva prima di passare alla successiva. Questo in quanto introdurre cambiamenti al software, in fasi avanzate dello sviluppo, ha costi elevati. Le uscite che una fase produce come ingresso per la fase successiva sono chiamate ***semilavorati*** I limiti di questo modello sono dati dalla sua *rigidità* in quanto ci sono due assunti di fondo:

- ***Immutabilità dell'analisi***
- ***Immutabilità del progetto***

Prototipo

Il prototipo ha l'obiettivo di essere mostrato al cliente per ottenere indicazioni sulle specifiche del progetto. Deve essere sviluppabile in tempi brevi e con costi minimi.

throw-away prorotyping

Prima di iniziare a lavorare sul sistema viene fornito, al cliente, un prototipo su cui definire le specifiche. Una volta esaurito il compito questo prototipo viene abbandonato.

Modelli evolutivi

Programmazione esplorativa

Il prototipo, progressivamente, fluisce nel prodotto finale. Questo presuppone un lavoro a stretto contatto con il cliente.

Esistono diversi tipo di modelli evolutivi, ma tutti in sostanza propongono un ciclo di sviluppo in cui un prototipo iniziale evolve, gradualmente, verso il prodotto finito. Il vantaggio è che ad ogni iterazione è possibile :

- ***Raffinamento dell'analisi*** : rivedere specifiche e funzionalità.
- ***Raffinamento del design*** : rivedere le scelte di progettazione.

Problemi dei modelli evolutivi

- Il processo di sviluppo non è visibile.
- Il sistema è poco strutturato.
- E' richiesta una particolare abilità nella programmazione.

Modelli ibridi

Si tratta di sistemi composti da sotto-sistemi. Per ogni sotto-sistema è possibile adottare un diverso modello di sviluppo.

Sviluppo incrementale

- Si costruisce un sistema sviluppandone sistematicamente e in sequenza parti ben definite.
- **Una volta costruita una parte essa non viene più modificata**

Sviluppo iterativo

Si effettuano molti passi dell'intero cicli di sviluppo del software, per costruire, iterativamente tutto il sistema.

Sviluppo incrementale - iterativo

- Si individuano sottoparti relativamente autonome
- Si realizza il prototipo di una di esse
- Si continua con altre parti
- Si aumenta, progressivamente, l'estensione e il dettaglio dei protitipi.

RUP - Rational Unified Process

Non definisce un singolo, specifico processo, bensì un framework adattabile che può dar luogo a diversi processi in diversi contesti (per esempio in diverse organizzazioni o nel contesto di progetti con diverse caratteristiche). E' pensato per progetti di grandi dimensioni.

RUP individua tre diverse versioni del processo di sviluppo:

- Una prospettiva dinamica che mostra le fasi del modello nel tempo
- Una prospettiva statica che mostra le attività del processo coinvolte
- Una prospettiva pratica che suggerisce le buone prassi da seguire durante il processo

PROSPETTIVA DINAMICA

Inception(Avvio) - Generalizzazione dell'analisi di fattibili. Lo scopo principale è quello di delinare nel modo più accurato il business case ovvero:

- Comprendere il *tipo di mercato* al quale il progetto afferisce e identificare gli elementi importanti affinché esso conduca a un successo commerciale.
- Identificare tutte le *entità esterne* che interagiranno con il sistema e definire tali interazioni.

Elaboration(Elaborazione) - definisce la struttura complessiva del sistema. Comprende l'analisi di dominio e una prima fase di progettazione dell'architettura. L'elaborazione deve soddisfare i seguenti criteri :

- Modello dei casi d'uso completo all'80%
- Descrizione dell'architettura del sistema
- Sviluppo di un'architettura esegubile che dimostri il completamento degli use case significativi
- Revisione del business case e dei rischi
- Pianificazione del progetto complessivo

Construction(Costruzione) - Progettare, programmare e testare il sistema:

- Le diverse parti del sistema vengono sviluppate parallelamente e poi integrate
- Al termine della fase si dovrebbe avere un sistema software funzionane e la relativa documentazione pronta

Transition(Transizione) - Il sistema passa dall'ambiente di sviluppo a quello del cliente finale:

- Vengono condotte attività di training degli utenti e beta testing.
- Si deve in particolare verificare che il prodotto sia conforme alle aspettative descritte nella fase di inception.

PROSPETTIVA STATICA

La prospettiva statica di RUP si concentra sulle attività di produzione del software (*** workflow ***). RUP è stato progettato insieme ad UML quindi, la descrizione dei workflow, è orientata ai modelli UML.

- **WORKFLOW PRINCIPALI**
    - ***Modellazione delle attività principali***: i processi aziendali sono modellati utilizzando il business case.
    - ***Requisiti***: vengono identificati gli attori che interagiscono con il sistema e sviluppati i casi d'uso per modellare i requisiti.
    - ***Analisi e progetto***: viene creato e documentato un modello di progetto.
    - ***Implementazione***: i componenti del sistema sono implementati e strutturati.
    - ***Test***
    - ***Rilascio***: viene creata una release del prodotto.
- **WORKFLOW DI SUPPORTO**
    - ***Gestione della configurazione e delle modifiche***: workflow di supporto che gestisce i cambiamenti del sistema.
    - ***Gestione del progetto***: gestisce lo sviluppo del sistema.
    - ***Ambiente***: rende disponibili al team di sviluppatori gli strumenti adeguati

PROSPETTIVA PRATICA

La prospettiva pratica di RUP descrive la buona prassi che si consiglia di utilizzare nello sviluppo dei sistemi. Le pratiche fondamentale sono sei:

- ***Sviluppare il software ciclicamente***: Sviluppare e consegnare le funzioni con la priorità più alta all'inizio del processo di sviluppo.
- ***Gestire i requisiti***: documentare esplicitamente i requisiti del cliente e i cambiamenti effettuati
- ***Usare architetture basate sui componenti***: strutturare l'architettura del sistema con approccio a componenti.
- ***Creare modelli visivi del software***: usare modelli grafici UML
- ***Verificare la qualità del software***: assicurarsi che il software raggiunga gli standard qualitativi
- ***Controllare le modifiche del software***: gestire i cambiamenti del software usando un sistema per la gestione delle modifiche.

ADT - MOD.1-3

Concetti base di classi astratte e programmazione OO

Analisi dei requisiti MOD.2-3

I requisiti di un sistema rappresentano la descrizione

- Dei servizi forniti
- Dei vincoli operativi
- **Requisiti utente**: dichiarano quali servizi il sistema dovrebbe fornire e i vincoli sotto cui deve operare.
    - Sono requisiti molto astratti e di alto livello
    - Tipicamente sono espressi in linguaggio naturale e corredati da qualche diagramma.
- **Requisiti di sistema**: definiscono le funzioni, i servizi e i vincoli del sistema in modo dettagliato
    - il ***Documento dei Requisiti del sistema*** deve essere preciso e definire esattamente cosa deve essere sviluppato.

Requisiti di sistema

I requisiti di sistema, solitamente, sono divisi in: - Requisiti funzionali - Requisiti non funzionali - Requisiti di dominio

Requisiti funzionali

Descrivono quello che il sistema "dovrebbe fare". Sono elenchi di servizi che il sistema dovrebbe fornire e per ogni servizio dovrebbe essere indicato:

- Come reagire a particolari input
- Come comportarsi in particolari situazioni
- In alcuni casi specificare cosa il sistema non dovrebbe fare

Le specifiche dei requisiti funzionali dovrebbero essere: - **Complete** - **Coerenti**

Requisiti non funzionali

I principali tipi di requisiti non funzionali sono:

- **Requisiti del prodotto**: specificano o limitano le proprietà complessive del sistema.
    - *affidabilità, prestazioni, protezione dei dati, disponibilità dei servizi,tempi di risposta, occupazione di spazio, capacità dei dispositivi di I/O, rappresentazione dei dati nelle interfacce, etc.*
- **Requisiti organizzativi**: possono violare anche il processo di sviluppo adottato.
    - *politiche e procedure dell'organizzazione cliente e sviluppatrice, specifiche degli standard di qualità da adottare, uso di un particolare CASE tool e linguaggi di implementazione, limiti di budget, requisiti di consegna e milestones..*

- **Requisiti esterni**: si identificano tutti i requisiti che derivano da fattori non provenienti dal sistema e dal suo processo di sviluppo.

Requisiti di dominio

Derivano dal dominio di applicazione del sistema e solitamente si riferiscono ai suoi concetti. L'analisi deve coinvolgere gli esperti del dominio per chiarire ogni dubbio sulla terminologia.

Raccolta dei requisiti

- L'obiettivo è raccogliere tutte le informazioni su cosa il sistema deve fare secondo le intenzioni del cliente. Non prevede passi formali in quano dipende dal particolare tipo di problema
- **Risultato**
  - Un documento scritto dall'analista, discusso e approvato dal cliente.
  - Una versiona iniziale del glossario contente la descrizione *precisa e non ambigua* di tutti i termini e i concetti utilizzati
- **Tipologia di persone coinvolte**
  - Analista
  - Utente
  - Esperto del dominio (non indispensabile)
- **Metodi utilizzati**
  - Interviste, questionari
  - Studio di doumenti che esprimono i requisiti in forma testuale
  - Osservazione passiva o attiva del processo da modellare
  - Studio di sistemi software esistenti
  - Prototipi

Analisi dei requisiti

Validazione dei requisiti

Ogni requisito deve essere validato con i clienti prima di essere inserito nel documento dei requisiti.

Cambiamento dei requisiti

- Requisiti esistenti possono essere modificati o rimossi
- Nuovi requisiti possono essere aggiunti in una qualunque fase del ciclo di sviluppo
- Il costo del cambiamento è proporzionato all'avanzamento dello sviluppo
- Ogni cambiamento deve essera accuratamente analizzato

Analisi del dominio

- **Obiettivo** definire la porzione del mondo reale, rilevante per il sistema.
- *Principio Fondamentale* : **Astrazione**
- **Risultato**: prima versione del *vocabolario* partendo dai sostantivi che si trovano nei requisiti

Analisi e gestione dei rischi

- Analisi completa di tutti i possibili rischi che posso fare fallori o intralciare la realizzazione del sistema

- Ogni rischio presenta due carratteristiche:
    - Probabilità che avvenga
    - Costo

Le tipologie di rischi sono:

- **Rischi relativi ai requisiti**
- **Rischi relativi alle risorse umane**
- **Rischi relativi alla protezione e privacy dei dati**
- **Rischi tecnologici**
- **Rischi politici**

Strategie risolutive:

- **Strategia reattiva**
- **Strategia preventiva**
    - Si mette in moto prima che inizi il lavoro tecnico
    - Si individuano rischi potenziali, se ne valutano le probabilità e si stabilisce un ordine di importanza
    - Si predispone un piano che permetta di reagire in modo controllato ed efficace.

Casi d'uso e scenari

I casi d'uso e i relativi scenari permettono di:

- formalizzare i requisiti funzionali
- di comprendere meglio il funzionamento del sistema
- di comunicare meglio con il cliente

L'insieme di casi d'uso costituisce l'immagine del sistema con l'esterno.

1. Individuare il confine del sistema
2. Individuare gli attori
    - Ogni attore modella il ruole interpretato da un utente(*persona o sistema esterno*)
3. Individuare i casi d'uso
4. Disegnare i diagrammi dei casi d'uso
5. Descrivere i dettagli di ogni singolo caso d'uso mediante scenari
6. Ricontrollare e validare i casi d'uso insieme al cliente

Un caso d'uso

- viene sempre avviato, direttamente o indirettamente, dall'intervento di un attore che si pone un obiettivo.
- Si conclude con successo quando l'obiettivo viene raggiunto
- Si conclude con fallimento quando l'obiettivo non viene raggiunto

Un caso d'uso viene sempre descritto dal punto di vista di un attore e comprende

- **0+ Precondizioni** - Condizioni che devono essere tutte verificate prima che il caso d'uso possa essere eseguito
- **1+ Scentari - sequenze di passi** - descrivono le interazioni tra l'attore e il sistema

- **0+ Postcondizioni** - Condizioni che devono essere tutte vere quando il caso d'uso termina l'esecuzione con successo.

Ogni sequenza di passi deve essere scritto in una forma narrativa strutturata e utilizzare il vocabolario di dominio.

Un caso d'uso comprende

- **1 SCENARIO PRINCIPALE**
- **0+ SCENARI ALTERNATIVI**

Generalizzazione

Si utilizza quando un caso d'uso è simile ad un altro, ma fa qualcosa in più

Inclusione <>

Si utilizza quando un caso d'uso utilizza almeno una volta un altro caso d'uso

Estensione <>

Si utilizza quando è necessario aggiungere un comportamento opzionale a un caso d'uso esistente

Diagrammi UML - MOD 2.5

E' un *linguaggio* che serve per visualizzare, specificare, costruire e documentare un sistema e gli elaborati prodotti durante il suo sviluppo.