**LEZIONE 1: Software Processes**

**Software** = programma e documentazione associata

Possono essere:

* Generici = sviluppato per essere venduto ad un range ampio di persone
* Custom = fatto apposta per un cliente (specifiche molto precise)

**COS’E’ L’INGEGNERIA DEL SOFTWARE?**

* **E’ una disciplina** dell’ingegneria che si occupa dello sviluppo dei sfotware

L’Ingegneria si sviluppa più su roba pratica (anche sul costo che serve per sviluppare il tutto) per sviluppare software rispetto all’informatica che è più teorica

I**ngegneria dei sistemi e Ingegneria del software**

* Si occupa anche della parte hardware rispetto ad INGSW, come assemblarla e tutto il resto nel modo più efficiente
* L’INGSW è un sott’insieme di essa

**PROCESSO SFOTWARE**

* Una serie di attività che si occupano di sviluppare o migliorare un software

Ci sono 4 tipi:

* Specificazione: Cosa vuole il cliente che il programma faccia?
* Sviluppo: Produzione del software
* Validazione: Controlliamo se ciò che abbiamo fatto coincide con le richieste
* Evoluzione (?): Tenere in continua evoluzione il software

**MODELLI DI PROCESSO DI SVILUPPO SOFTWARE**

* Una rappresentazione semplificata di un processo software (es. Modello a cascata)

**DOVE SONO I COSTI DI UN SOFTWARE?**

* 60% costi di sviluppo
* 40% testing costs
* Molti costi verranno spesi nell’evoluzione del sistema (aggiornamenti ecc..)

**METODI DI INGSW**

* Processi strutturati per approcciare lo sviluppo di un software

**COME SVILUPPARE IN SOFTWARE IN MODO CORRETTO?**

* Il software dovrebbe rilasciare le funzionalità richieste con performance accettabili dall’utente, e deve essere
* Manutenibile: facile da aggiornarne e aggiungere cose senza ripercussioni sul resto del sof
* ,Affidabile: deve dare fiducia nell’utilizzo di un certo sfotware
* Efficiente: Non deve sprecare risorse dell’hardware
* Accettabilità: Chi lo utilizza lo deve accettare, in caso contrario non verrebbe utilizzato o potrebbe essere vista come un ostacolo (meno utilizzo = meno soldi)

**SFIDE CHE L’INGSW AFFRONTA**

* Eterogenuità: Sviluppare tecniche utilizzabili su varie piattaforme
* Delivery: Consegnare in tempo il sfotware
* Fiducia: Sviluppare un software con la sicurezza che funzioni

PROCESSI DI SVILUPPO SFOTWARE

* Processo software = abilità per sviluppare un software di sistema
* Modello a Cascata
* Sviluppo evolutivo

MODELLO A CASCATA

* Come una cascata, in questo modello il flusso è verso il basso
* 1. Analisi dei requisiti
* 2. System and software desing
* 3. Implementazione e unit testing
* 4. Integrazione e system testing
* 5. Operazioni e manutenzione

Per passare alla fase successiva, completare fase attuale

Un problema di questo modello è la **bassa flessibilità**, difficile rispondere ad eventuali cambiamenti richiesti dal cliente

Questo modello viene usato per progetti molto ampi

SVILUPPO EVOLUTIVO

Qui fasi eseguite più volte, per venire in contro a nuove richieste del cliente

Esplorazione insieme al cliente dei requisiti, tramite feedback ecc..

Da non confondere con il PROTOTIPO USA E GETTA (si usa Quando non si sa bene cosa fare)

Rispetto al modello a cascata ha dei vantaggi:

* Se c’è qualcosa da cambiare non lo portiamo fino alla fine
* Si usa per progetti non molto grandi, per non perdere visioni sul futuro(?)

INGSW BASATA SUI COMPONENTI

* Basata su riutilizzi sistematici di sistemi integrati da componenti già esistenti

PROCESSO ITERATIVO

* I Requisiti di un sistema evolvono sempre nel corso del progetto
* L’iterazione puo essere applicata su generici modelli

SVILUPPO A SPIRALE

* Qui si tiene in conto nel ripetere vari fasi, nel bilancio rischio/benefici
* Non ci sono fasi specifiche, dipende da cosa conviene fare prima
* Si decide obiettivo prossima iterazione, quelle che hanno meno svantaggio e meno rischi si fanno prima

SPECIFICHE SOFTWARE

* Le fasi sono:
* 1. Studio di fattibilità = cerca di capire se quello che si desidera si riesce a fare ( valutiamo costi, risorse, tempo ecc…)
* 2. Identificazione dei requisiti e analisi = Requisito è un vincolo di funzionamento per un programma, qui capiamo cosa vuole il nostro cliente (Qui si sceglie il system models)
* 3. Specifiche dei requisiti = Scrittura dei requisiti richiesti dal cliente
* 4. Validazione dei requisiti = check dei requisiti (Qui si scriverà il documento dei requisiti)

SOFTWARE DESIGN E IMPLEMENTAZIONE

* Processo il quale converte specifiche del sistema in un sistema eseguibile

STRUCTURED METHODS

* Approccio sistematico per lo sviluppo di un sfotware desing
* Il design è sempre documentato come una serie di modelli grafici, Modelli Possibili:

1. Object Model
2. Sequence Model

…

PROGRAMMING AND DEBUGGING

* Trasformare un design in un programma, rimuovendo errori da esso

VALIDAZIONE DEL SFOTWARE

* Verifica e validazione, cioè quando il sistema è conforme alle specifiche e requisiti del cliente
* Verifica= Abbiamo fatto bene quello che abbiamo implementato?
* Validazione= Coincide con le richieste del cliente?

TESTING STAGES

* Component o unit testing:
  + Componenti individuali testati indipendentemente
* System testing
  + Test del sistema nel complesso.

(DA FINIRE)

EVOLUZIONE DEL SOFTWARE

* Dopo i test, validazione ecc.. passiamo alla fase di evoluzione
* I requisti del sistema cambieranno sempre nel tempo, per questo deve essere felissibile al cambiamento

REQUISITI DEL SOFTWARE

* Cos’è un requisito? = Stato di un servizio o un vincolo di sistema di alto (astratto) o basso (funzioni matematiche) livello
* E’ ineviteabile che un requistio debba avere due funzioni:
  + La base per un offerta di contratto
  + Base del contratto stesso (definita in dettagli)

TIPI DI REQUISITI

* Requisito utente: Scritto in linguaggio naturale più diagrammi a seguito di affermazioni date dal cliente, questi sono scritti appunto per loro
* Requisiti di sistema: Scritti con un documento strutturato, con descrizioni dettagliate delle funzioni del sistema, definiscono cosa implementare

REQUISITI FUNZIONALI E NON FUNZIONALI

* Requisiti funzionali (VEDI SLIDE PER ESEMPI)
  + Descrive funzionalità o servizi del sistema
  + Come il sistema reagisce a particolari input
  + Come il sistema dovrebbe comportarsi in situazioni particolari
* Requisti non funzionali
  + Vincoli sui servizi o funzioni matematiche (timing constraints, vincoli sullo sviluppo del processo)

IMRECISIONI DEI REQUISTI

* Queste imprecisioni emergono a seguito di requistii non precisamente esposti
* Requisiti possono essere molto ambigui e quindi possono essere interpretati in modo differente

I Requisiti devono essere:

* Completi= devono includere descrizioni di tutte le strutture dei requisiti
* Consistenti = Non devono andare in conflitto o in contraddizione nella descrizione

REQUISTI NON FUNZIONALI

* Definiscono proprietà di sistema e vincoli
* Questi requisiti possono essere molto più critici rispetto a quelli funzionali, se non vengono rispettati il sistema è inutile