Analisi e Progettazione del Software

Avvertenze:

Questi appunti sono stati scritti per la prima volta in MarkDown con Typora, in base a come andranno (Se piaceranno o saranno comunque leggibili) probabilmente continuerò o meno. Potrei ritornare ad usare LaTeX.

Ovviamente la valutazione sarà anche da un mio punto di vista, se mi garba, continuo, all'incovercio, diobono cancello pure tutto Linux.

Software da avere/usare/FARE ESPLODERE:

IBM Rational Software Architect v7.0:

Sarebbe un fork **Eclipse** fatto male.. Praticamente prendere dello sterco, appiccicatelo di pacco su Eclipse, ottenete questo. Vi ricordate **Lispworks**? PEGGIO. No ok, peggio di Lispworks credo ci sia poco, ma ci siamo capiti insomma, no?

Un Client CVS o SVN:

C'è sia integrato in **Eclipse**, che in **Tortoise CVS** che in **Smart CVS**

Obbiettivi del corso

Stando alle slides puntano tutto sul capire come sviluppare software, non in senso di codice, ma tutto ciò che ne viene alla base del suddetto. Banalmente ci spiegano come si progetta il software **DOPO** averci fatto fare un progetto (di LP)... Un po' discutibile, no?

Let's start bois ->

Definizione di Software

Ma davvero ogni volta dobbiamo ripartire da capo con queste definizioni? AD OGNI MODO:

- Un software è un programma/insieme di programmi con **documentazione** ad esso associata, tra cui:
    - Analisi dei Requisiti

    - Modelli di Progetto
    - Manuale per l'utente

- E come si categorizza il software? (Prendete il corso di Reti e Sistemi, lì c'è vita morte e miracoli)
    - Generici
    - Personalizzati

> Un software **generico** è un software "uguale per tutti" (***Urss anthem intensifies***), prendete tipo Word, Excel, Firefox. Roba che chiunque può avere,
>
> Un software **personalizzato** è invece un software che un'azienda sviluppa per un cliente specifico, ad esempio i software di gestione aziendale dei **ticket** o gestione dei vari server aziendali.

Un nuovo software può nascere completamente da 0, ma volendo potrebbe essere un fork di un altro, o anche solo implementarne una parte, vedi GNU Public License

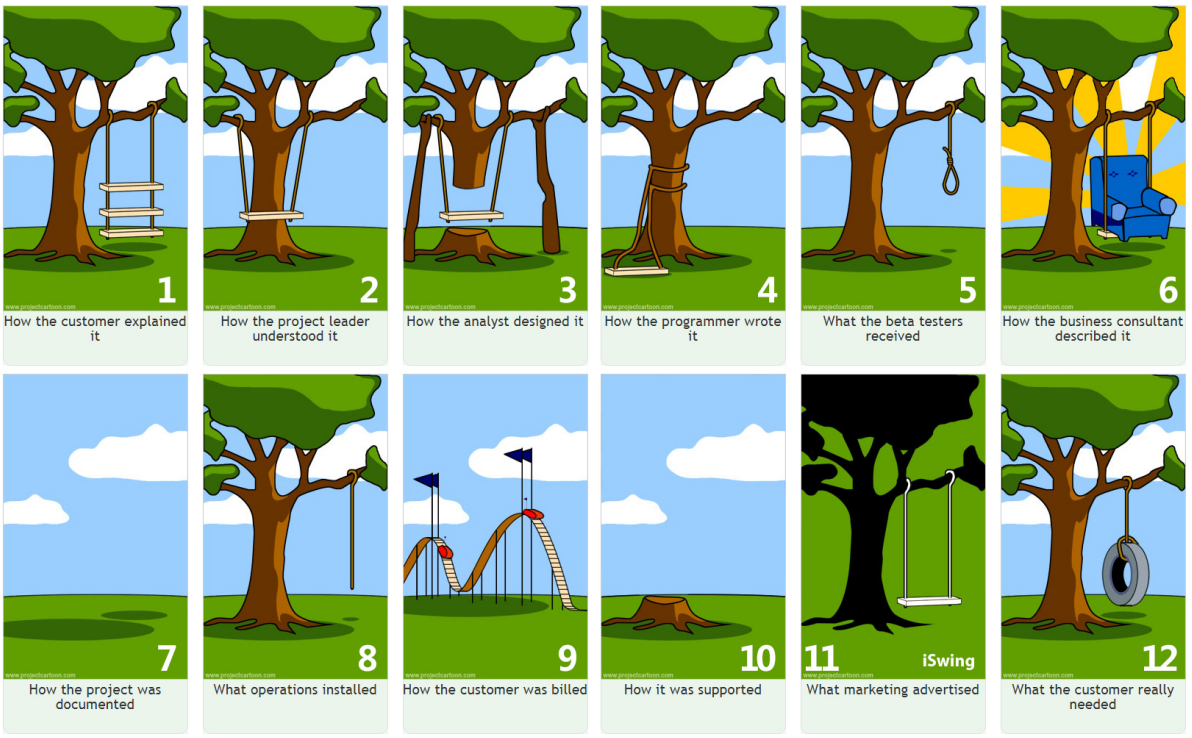
Ingegneria del Software

Comprende tutto ciò che sta attorno alla produzione di un qualsiasi software, mediante organizzazione del lavoro (con tecniche appropriate (ed approvate dalla PegPerego per farci giocare i bambini, naturalmente))

Su cosa si basa?

- Vincoli sulle modalità di sviluppo,
- Risorse disponibili

Ecco a grandi linee uno schema di come funziona il tutto:



Modello di Processo

- Alto livello
  - Raccolta di requisiti, progettazione, stesura di un piano di lavoro, supporto alle prime attività di processo
- Basso livello
  - Supporto alle ultime attività di processo, quindi ai programmatori, tester, debugger, bestemmiatori, infami, ignavi, satana, e reverse egineering

Com'è fatto un software buono?

Basta non essere come Lispworks, o IBM Rational Software Architect, ma più precisamente deve essere:

- Utilizzabile

  Già qui Lispworks ha perso in partenza, per utilizzabile si intende che dev'essere facile da utilizzare, nel senso di intuitivo, che non ti faccia venire voglia di maledire il carnevale ad ogni click

- Fidato

  Deve avere una probabilità di bug o malfunzionamenti bassa, il più ridotta possibile, non come DevC++ che è l'unico IDE che ha il debugger buggato.

- Efficiente

Non si devono sprecare risorse, deve essere efficace ed efficiente, porta a termine il tuo compito e fallo nel migliore tempo, con meno risorse possibili

- Mantenibile

  Dev'essere strutturato in maniera tale da poterlo aggiornare e mantenere in modo agevole.

  > In taluni casi, l'unico modo per risolvere i bug del software, è ritirandolo dal mercato

Accettabilità

Il software va accettato dagli utenti ai quali è destinato il software, quindi va scritto compatibile, semplice. In una sigla: KISS, Keep it simple stupid. Pure il più imbecille degli utenti deve essere in grado di usarlo.

Responsabilità Etica e Professionale

L'ingegneria del software che da ora scriverò SW perché sì, include responsabilità legate alla sicurezza dei dati su cui si opera.

Come deve essere il contraente con il committente?

Oltre rispettare la legge, nel senso che non devono esserci violazioni del Copyright tipo praticamente mezzo sistema operativo "RedStarOS", che è un clone fatto malissimo di MacOS PIENO di violazioni e... E' l'OS della Korea del Nord, io non andrei a reclamare, non so voi.

Il contraente deve garantire:

- Riservatezza
- Competenza
- Proprietà intellettuale: Ciò che ti do è fatto da me, se ti devo fare un sito non te lo faccio con Wordpress.
- Uso improprio del computer (Nel senso, se con un browser diffondi materiale illegale, non ne risente chi ha sviluppato il browser, ma chi ne fa uso)
-

Codice etico ACM/IEEE

E' uno standard di etica e pratica professionale unificato, tipo un manuale di comportamento

E' composto da 8 principi legati a tutto ciò che è legato al comportamento di ingegneri sw, sviluppatori, educatori, manager, supervisori e anche noi poveri studenti