Qualità del software – I problemi

La gestione delle release
Testing
La documentazione
L'etica
Il backup delle versioni
Le specifiche funzionali
I rapporti con il cliente
……
……

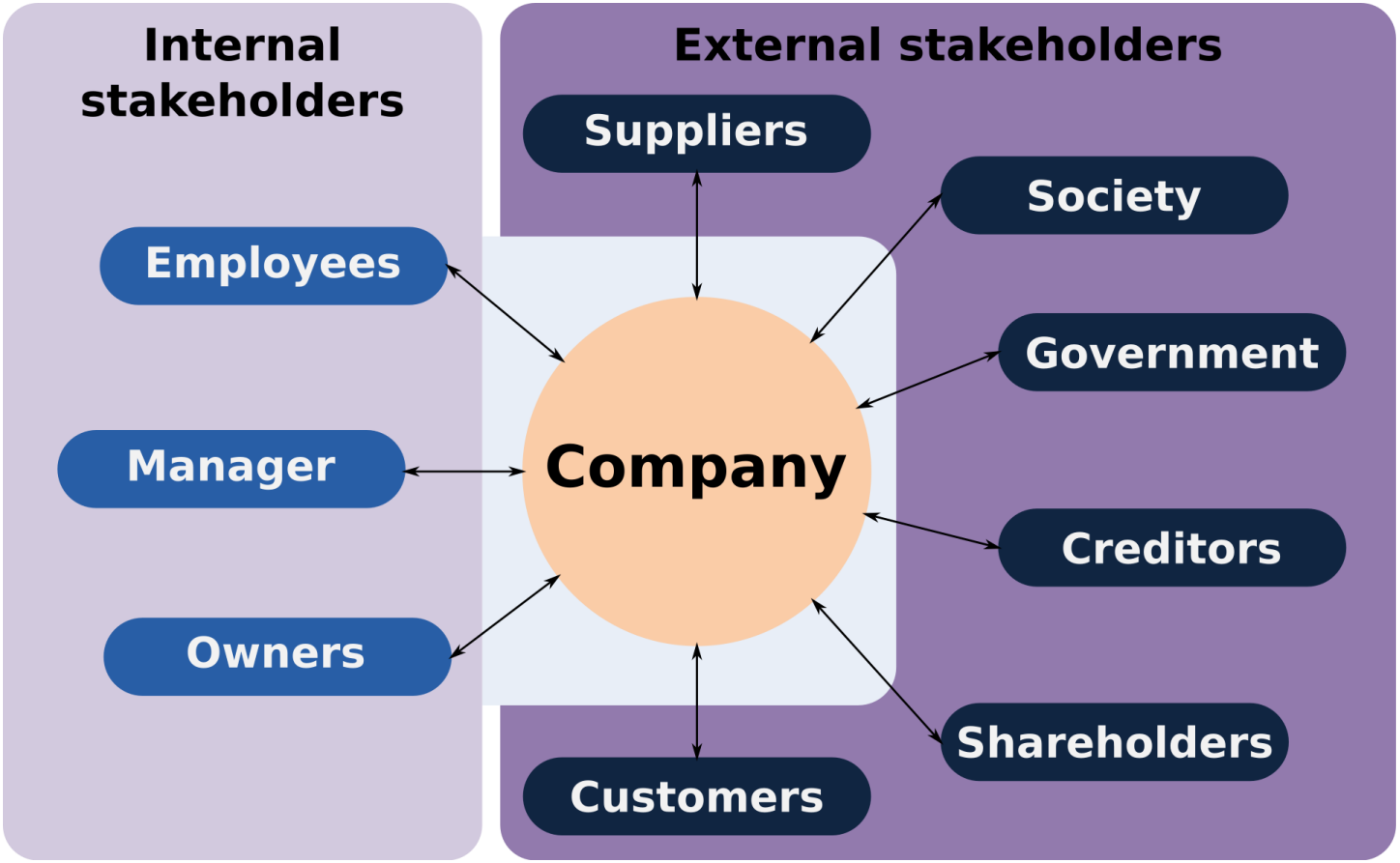
- Video etica digitale – Norberto Patrignani

https://www.youtube.com/watch?v=wgH3iXb9PLk

https://www.acm.org/code-of-ethics

Stakeholder?

Un soggetto (o un gruppo) *influente* nei confronti di una iniziativa economica, una società o un qualsiasi altro progetto.



I parametri della qualità del software

- Parametri di qualità esterni

Sono quelli relativi alla percezione di qualità da parte degli utilizzatori del software

- Parametri di qualità interni

  Sono quelli relativi alla percezione di qualità da parte degli sviluppatori del software

I parametri della qualità del software

- Parametri di qualità esterni
  Correttezza
  Affidabilità
  Robustezza
  Efficienza
  Usabilità
  Ecocompatibilità
  Scalabilità

- Parametri di qualità interni
  Verificabilità
  Manutenibilità
  Riparabilità
  Evolvibilità
  Riusabilità
  Portabilità

Parametri di qualità esterni

Correttezza

- Un programma è corretto se fa esattamente quello che è stato progettato per fare

Affidabilità

- Un sistema è tanto più **affidabile** quanto più raramente, durante l'uso del sistema, si manifestano malfunzionamenti

Robustezza

- In termini generali, la **robustezza** di un sistema è la misura in cui il sistema si comporta in modo *ragionevole* in situazioni impreviste, non contemplate dalle specifiche

Efficienza

- Un sistema è **efficiente** se usa memoria, CPU e altre risorse in modo proporzionato ai servizi che svolge

Usabilità

- Un sistema è facile da usare se un essere umano lo reputa tale.
  È una qualità soggettiva!

Ecocompatibilità

- Le tecnologie dell'informazione e della comunicazione (TIC) sono responsabili di circa il 2% delle emissioni globali di anidride carbonica (CO2). Questo numero comprende soltanto la "fase di utilizzo" dell'hardware. Il software può contribuire a diminuire il consumo di energia (ovvero diventare più verde) in almeno due modi: rendendo il software più efficiente da un punto di vista energetico - quindi usando meno risorse e producendo meno emissioni di CO2 - e rendendo più sostenibili i processi supportati.

Scalabilità

- Un sistema è scalabile se può essere adattato a diversi contesti con forti differenze di complessità (per esempio database molto piccoli o molto grandi) senza che questo richieda la riprogettazione dello stesso sistema

Parametri di qualità interni

Verificabilità

Un sistema è verificabile se le sue proprietà:

- correttezza
- affidabilità

sono facili da verificare.

Manutenibilità

- Facilità di apportare modifiche a sistema realizzato. In origine si pensava che la manutenzione corrispondesse solo al bug fixing, ma oggi la situazione è più complessa; la manutenzione riguarda infatti ogni miglioramento del software e andrebbe indicata più precisamente come evoluzione del software

Riparabilità

- Un sistema è riparabile se la correzione degli errori è poco faticosa

Evolvibilità

- È necessario prevedere sin dall'inizio che il software può evolvere e progettare tenendo in mente questa evoluzione per sfruttare al meglio i costi sostenuti in passato.

Riusabilità

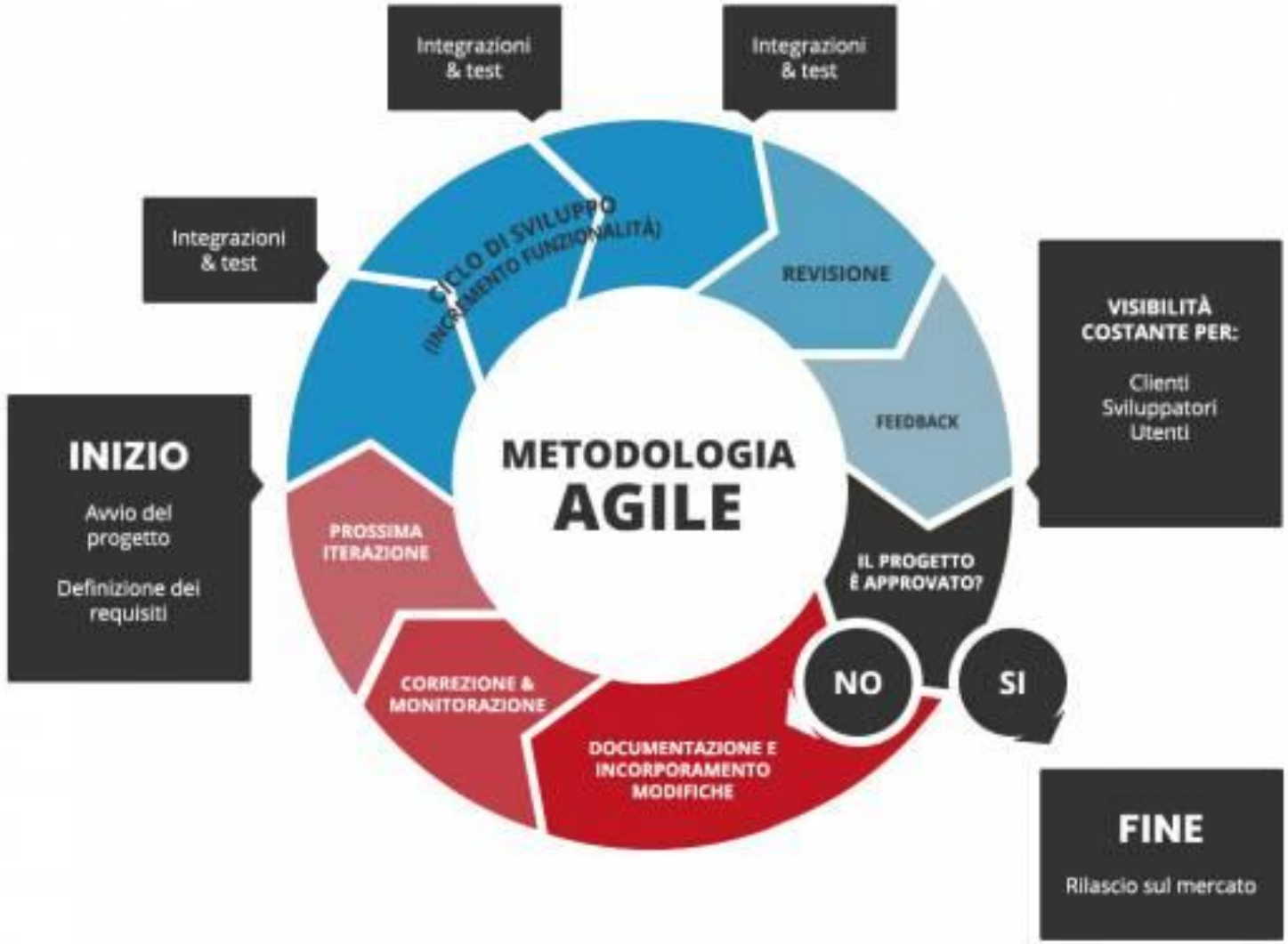
- Affine all'evolvibilità. Tuttavia si ha nell'evolvibilità una modifica per realizzare una nuova versione, mentre nella riusabilità si usano parti di sistema per realizzare un prodotto diverso

Portabilità

- Un sistema è portabile se è in grado di funzionare in ambienti diversi.

Metodologia Agile

come sviluppare
software in modo
più rapido,
snello ed efficiente.


Integrazioni & test
Integrazioni & test
Integrazioni & test
CICLO DI SVILUPPO (INCREMENTO FUNZIONALITÀ)
REVISIONE
FEEDBACK
VISIBILITÀ COSTANTE PER:
Clienti
Sviluppatori
Utenti
INIZIO
Avvio del progetto
Definizione dei requisiti
METODOLOGIA AGILE
PROSSIMA ITERAZIONE
IL PROGETTO È APPROVATO?
NO
SI
CORREZIONE & MONITORAZIONE
DOCUMENTAZIONE E INCORPORAMENTO MODIFICHE
FINE
Rilascio sul mercato

Metodologia Agile – gli obiettivi

]

L'obiettivo è la piena soddisfazione del cliente e non solo l'adempimento di un contratto.

Il corretto uso di queste metodologie, inoltre, può consentire di abbattere i costi e i tempi di sviluppo del software, aumentandone la qualità.

La metodologia prende spunto dai metodi applicati in piccole software house dopo la **Internet revolution**

https://it.wikipedia.org/wiki/Metodologia_agile

Metodologia Agile – I principi

]

I principi su cui si basa una metodologia agile che segua i punti indicati dall'Agile Manifesto, sono solo quattro:

•le persone e le interazioni sono più importanti dei processi e degli strumenti (ossia le relazioni e la comunicazione tra gli attori di un progetto software sono la miglior risorsa del progetto);

•è più importante avere software funzionante che documentazione (bisogna rilasciare nuove versioni del software ad intervalli frequenti, e bisogna mantenere il codice semplice e avanzato tecnicamente, riducendo la documentazione al minimo indispensabile);

•bisogna collaborare con i clienti oltre che rispettare il contratto (la collaborazione diretta offre risultati migliori dei rapporti contrattuali);

•bisogna essere pronti a rispondere ai cambiamenti oltre che aderire alla pianificazione (quindi il team di sviluppo dovrebbe essere pronto, in ogni momento, a modificare le priorità di lavoro nel rispetto dell'obiettivo finale).

In sintesi, l'Agile Manifesto, sottolinea l'importanza dei succitati principi fermo restando il valore di processi, strumenti, documentazione, contratti e pianificazione.

Metodologia Agile – Le pratiche

]

Automazione - Se l'obiettivo delle metodologie agili è concentrarsi sulla programmazione senza dedicarsi alle attività collaterali, allora queste ultime possono essere eliminate o automatizzate;;

Comunicazione stretta - Secondo Alistair Cockburn, probabilmente il primo teorico delle metodologie agili, questo è l'unico vero aspetto nodale che rende *agile* una metodologiaì

Coinvolgimento del cliente - Il coinvolgimento del cliente è qui indicato singolarmente perché ci sono differenti gradi di coinvolgimento possibiliì

Progettazione e documentazione - flessibile

Consegne frequenti - Effettuare rilasci frequenti di versioni intermedie del software permette di ottenere più risultati contemporaneamente: si ricomincia l'iterazione avendo già a disposizione un blocco di codice funzionante in tutti i suoi aspetti, si offre al cliente "qualcosa con cui lavorare" e lo si distrae così da eventuali ritardi nella consegna del progetto completo, si usa il cliente come se fosse un test

Metodologia Agile – Le pratiche

]

Gerarchia - La scelta di creare una struttura gerarchica all'interno del team di sviluppo dipende molto dall'approccio del project manager, in ogni caso si ha una conseguenza non secondaria facendo questa scelta; se si decide per una struttura gerarchica ad albero e frammentata si ottiene la possibilità di gestire un numero molto alto di programmatori e di lavorare a diversi aspetti del progetto parallelamente; se si decide per una totale assenza di gerarchia si avrà un team di sviluppo molto compatto e motivato, ma necessariamente piccolo in termini di numero di programmatori;

Pair Programming - Lo sviluppo viene fatto da coppie di programmatori che si alternano alla tastiera;

Re factoring - La ristrutturazione di parti di codice mantenendone invariato l'aspetto e il comportamento esterno;

Miglioramento della conoscenza - Nata con l'avvento della programmazione Object-Oriented, non è altro che la presa di coscienza della produzione di conoscenza che si fa in un'azienda man mano che si produce codice;

Retroingegneria - Ossia ottenere, spesso in maniera automatica, la documentazione a partire dal codice già prodotto

Metodologia Agile – Le pratiche

]

Semplicità - semplicità nel codice, semplicità nella documentazione, semplicità nella progettazione, semplicità nella modellazione

Formazione di una squadra e Proprietà del codice - La formazione del team di sviluppo è condizionata dalla scelta sulla gerarchia interna, ma segue regole precise che permettono di ottenere un team produttivo nell'ambito della metodologia scelta; la scelta dei membri del team è condizionata anche alla scelta della proprietà del codice, che può essere individuale o collettiva; nel primo caso la responsabilità sullo sviluppo è individuale, nel secondo dipende da tutto il team e quindi dal project manager;

Test - Pratica diffusissima anche prima della nascita delle metodologie leggere, ha prodotto una letteratura vastissima ed una serie di approcci differenti

Controllo della versione - Una delle conseguenze dirette dell'iterazione nella produzione è la necessità di introdurre un modello, un metodo, uno strumento, per il controllo delle versioni del software prodotto e rilasciato; uno degli strumenti più diffusi e maggiormente suggeriti per ottemperare automaticamente a questa pratica è il *CVS*.

Ieri- Makefile

```
edit : main.o kbd.o command.o display.o
    cc -o edit main.o kbd.o command.o display.o

main.o : main.c defs.h
    cc -c main.c
kbd.o : kbd.c defs.h command.h
    cc -c kbd.c
command.o : command.c defs.h command.h
    cc -c command.c
display.o : display.c defs.h
    cc -c display.c

clean :
     rm edit main.o kbd.o command.o display.o
```

Oggi

Integrated Development Environments

- Netbeans, Eclipse,
- GitHub (repository)

Che impatto hanno gli aggiornamenti automatici?

- esempio

Concludendo…
3 consigli

- Specifiche chiare (a partire dalle richieste del cliente)
- Buona gestione dei rapporti con il team
- Testing

Buon lavoro!