Sistemi Informativi:

Il processo software

La metodologia “Agile”

- All’inizio c’era il programmatore solitario
- Per fare un po’ di ordine furono introdotti
  - i processi di costruzione del sw, con
  - strumenti CASE (Computer Aided Software Engineering) di supporto
- Come reazione ai grossi sistemi CASE, si sono evolute metodologie *agili* (leggere) di produzione del sw.

Il vecchio e il nuovo modo

- Vecchio:
  - enfasi sul processo (complicato, burocratico)
  - indipendente dalle persone
  - molta documentazione prodotta
  - predittivo: stabilire il futuro, resistenza al cambiamento
  - successo come conformita' al piano originale
- Il modo "agile"
  - enfasi sulla capacita' del team di sviluppo
  - dipendente dalle persone
  - il codice sorgente e' la documentazione piu' importante
  - adattivo
  - successo come utilita' per il cliente

la metodologia “agile”

- cicli di verifica brevi
- sostituire il costo fisso con intervallo
- importanza agli individui (non intercambiabili)
  - I programmatori sono individui responsabili
  - il manager accetta le loro decisioni
  - difficolta’ di misurazione (fallisce se si trascura anche un solo fattore importante)
  - accesso continuo alle competenze del cliente
- costante revisione ed adattamento a ogni ciclo

quando usare “agile”

- avere un team che vuole provarla
- il cliente vuole collaborare
- iniziare con un progetto piccolo ma importante
- requisiti in evoluzione

eXtreme Programming

- Una delle metodologie Agili (non per SI, in generale)
- Il target e' un team di programmazione piccolo (da 2 a 10 programmatori)
- Il problema ha requisiti che sono vaghi e in evoluzione
- Basata sulle pratiche migliori di programmazione
- *Implementare* (codificare) e' l'attività chiave del progetto sw.

Le quattro variabili

- *Costo*, *tempo*, *qualita'*, e *scope* (ambito di applicabilita' della soluzione)
- Prima il manager le stabiliva tutte e 4
- Nel modo eXtreme: dal di fuori ne sono stabilite 3 la quarta e'scelta dagli sviluppatori.
- In generale la qualità deve essere alta. Costo e tempo sono stabiliti esternamente, per cui lo sviluppatore decide lo scope

Non chiarezza (softness) dei requisiti

- In questo contesto è considerata positiva, perchè permette di variare l' ampiezza delle funzionalita'
- Sviluppando possono cambiare i requisiti.
- Fissate le altre tre variabili si può lavorare su questa
- Questo procedimento tollera i *cambiamenti* facilmente, ed il progetto cambierà direzione spesso.

Come soddisfare il cliente

(tradizionalmente)

- requisiti progettazione testing implementazione produzione
- costo dei cambiamenti
- Tempo

– Extreme

- Scegliere prima le funzionalita’ importanti
- Fare pratica nello stimare il tempo necessario a realizzare le funzionalita’ richieste

Le tecniche di XP

- Semplice progettazione, tecnologie quali:
  - Programmazione ad oggetti
  - DB orientati ad oggetti (non spendere troppo tempo nella traduzione dello schema)
- Test automatizzati
- Pratica nel modificare la struttura (refactoring)
  - Muovere funzionalità
  - Fare ogni decisione velocemente (ma avere test automatizzati per verificarne la correttezza)
- Programmazione in coppia

Il vecchio e XP

- Vecchio:
  - Lunga analisi dei requisiti
  - Lungo periodo di implementazione
  - Alla fine si può essere perso tempo in caratteristiche non importanti o già sorpassate
- Il modo eXtreme
  - Un semplice prototipo **subito**
  - Sviluppo incrementale, con feedback rapido
  - Il cliente può cambiare i requisiti

Confronto tra processi di sviluppo SW

Waterfall Model	Rapid Prototype Model	Spiral Model
Pro	**Pro**	**Pro**
Documentation	Allows frequent changes	Risk analysis preceding each phase
Maintenance easier	Helps define user requirements	Allows for changing requirements
Quality product at finish	Rapid return on investment	Allows prototyping
Con	**Con**	**Con**
Specification document	Increased maintenance costs	Once risk cannot be mitigated the project is terminated
Have to get it right first time	How do you know you are finished?	Not effective for large-scale projects
Does not allow for prototype	Build-and-fix	
Time consuming		
Costly		
Hard to accommodate		
Doesn't accommodate new requirements		

SW e SI

- un SI e’ un tipo di SW dove la parte di modellazione dei dati prevale sulla parte di scrittura degli algoritmi

La filosofia Open Source

- Un buon prodotto sw inizia con un problema personale di uno sviluppatore
- I *bravi* programmatori sanno cosa scrivere. I *migliori* sanno cosa riscrivere.
- Quando un programma non ti interessa piu' e' tuo dovere passare le consegne ad un successore competente
- Trattare gli utenti come sviluppatori e' il modo migliore per ottenere debugging efficace e miglioramenti del codice
- Distribuisci *presto* e *spesso* (e ascolta gli utenti).

Il processo Open Source

- Il processo e' "pubblico"
- Le implementazioni sono controllate da un board che revisiona e testa il codice proposto
- Il codice e' prodotto attraverso:
  - Modifiche moderate
  - Build frequenti
  - E' proprietà collettiva
- Non esiste il mantenimento

Manifesto Agile

- The Manifesto for Agile Software Development
-
- We are uncovering better ways of developing software by doing it and helping others do it.
- Through this work we have come to value:
-
- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan
-
- That is, while there is value in the items on the right, we value the items on the left more.

Principii Agile 1

- Our highest priority is to satisfy the customer
- through early and continuous delivery
- of valuable software.
-
- Welcome changing requirements, even late in
- development. Agile processes harness change for
- the customer's competitive advantage.
-
- Deliver working software frequently, from a
- couple of weeks to a couple of months, with a
- preference to the shorter timescale.
-
- Business people and developers must work
- together daily throughout the project.

Principii Agile 2

- Build projects around motivated individuals.
- Give them the environment and support they need,
- and trust them to get the job done.
-
- The most efficient and effective method of
- conveying information to and within a development
- team is face-to-face conversation.
-
- Working software is the primary measure of progress.
-
- Agile processes promote sustainable development.
- The sponsors, developers, and users should be able
- to maintain a constant pace indefinitely.
-
-

Principii Agile 3

- Continuous attention to technical excellence
- and good design enhances agility.
-
- Simplicity--the art of maximizing the amount
- of work not done--is essential.
-
- The best architectures, requirements, and designs
- emerge from self-organizing teams.
-
- At regular intervals, the team reflects on how
- to become more effective, then tunes and adjusts
- its behavior accordingly.
-
-

Limitazioni

- Team grossi (> 20)
- Team distribuiti
- SW “Mission critical”

Morale

- Non affezionarsi a nessuna metodologia
- Scegliere di volta in volta quella piu' adatta in base ai vincoli che impone
- Si possono ibridare
- Iniziare con le retrospettive
- Mai smettere di sperimentare
- Imparare dai propri errori