Università degli Studi di Udine

Metodologie Agili

prof. Maurizio Pighin

Dipartimento di Matematica e Informatica

Criticità dei metodi di sviluppo tradizionali

- Sequenzialità del processo
  - Requisiti inizialmente confusi e/o non noti
  - Requisiti instabili nel tempo
  - Il committente non ha le idee ben chiare sul prodotto che desidera
  - Gli ingegneri del software non sono esperti del dominio applicativo
  - Testing effettuato alla fine del processo di sviluppo

- Mentalità burocratica (documentazione e processi rigidi)

The Agile Manifesto

February 2001, 17

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.

Kent Beck, James Grenning, Robert C. Martin, Mike Beedle, Jim Highsmith, Steve Mellor, Arie van Bennekum, Andrew Hunt, Ken Schwaber, Alistair Cockburn, Ron Jeffries, Jeff Sutherland, Ward Cunningham, Jon Kern, Dave Thomas, Martin Fowler, Brian Marick

Agile principles

The Agile Manifesto is based on twelve principles:

1) Customer satisfaction by rapid delivery of useful software
2) Welcome changing requirements, even late in development
3) Working software is delivered frequently (weeks rather than months)
4) Close, daily cooperation between business people and developers
5) Projects are built around motivated individuals, who should be trusted
6) Face-to-face conversation is the best form of communication (co-location)
7) Working software is the principal measure of progress
8) Sustainable development, able to maintain a constant pace
9) Continuous attention to technical excellence and good design
10) Simplicity—the art of maximizing the amount of work not done—is essential
11) Self-organizing teams
12) Regular adaptation to changing circumstances

I principi del SW management

Approccio Convenzionale

I requisiti prima della progettazione

La progettazione prima dello sviluppo

Uso di linguaggi ad alto livello

Test di modulo prima di integrazione

Tracciabilità dettagliata tra documenti

Gestione della documentazione di progetto

Controllo della qualità esterno al processo

Controllo esteso a tutto

Pianificazione ampia e precisa

Controllo rigoroso dei sorgenti

Approccio Agile

Prima di tutto, l'architettura

Approccio iterativo per comprendere il problema

Sviluppo basato su componenti

Ambiente in grado di gestire i cambiamenti

Incoraggiamento e supporto dei cambiamenti

Progettazione basata su modelli

Controllo di qualità integrato nel processo

Verifica dell'avanzamento con dimostrazioni

Pianificazione di rilasci con livelli di dettaglio crescenti

Processi scalabili e configurabili

Metodologie agili

- Individui e interazioni vs. processi e strumenti
- Disponibilità di SW funzionante vs. ampia documentazione
- Collaborazione con il cliente vs. negoziazione contratti
- Pronta risposta ai cambiamenti vs. esecuzione di un piano
- Esistono vari modelli
  - *Extreme programming (XP)*
  - *Adaptive Software Development (ASD)*
  - *Dynamic System Development Method (DSDM)*
  - *Feature Driven Development (FDD)*
  - *Agile Modeling (AM)*
  - *Scrum (Mischia)*
  - *Crystal*

Processo aziendale

- Relazioni forti e continue tra soggetti
  - *All'interno dell'azienda*
    - Direzione Generale
    - Direzione commerciale
    - Team di analisi
    - Team di sviluppo
    - ...
  - *Con l'esterno dell'azienda*
    - Clienti
    - Committenti
    - Utenti finali

Variabili del progetto

COSTO

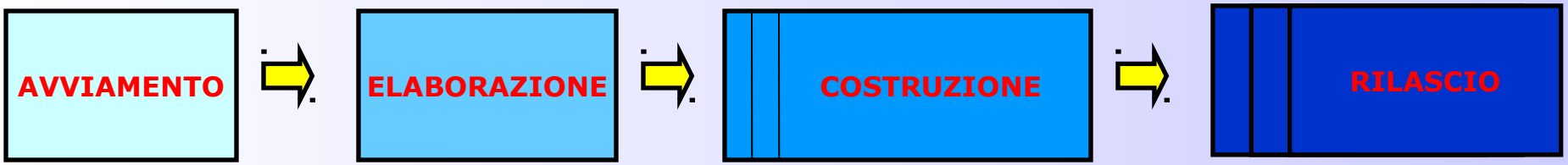
TEMPO

QUALITA'

DIMENSIONE

Processo di sviluppo di un progetto

Processo **iterativo** ed **incrementale**



Valori principali

SEMPLICITA'

COMUNICAZIONE

FEEDBACK

CORAGGIO

RISPETTO

Avvio del progetto

- Raccolta requisiti (ragionevolmente) dettagliati
- Analisi ad alto livello per stabilire l'architettura di base
- Si decide insieme la dimensione come numero e valore delle funzionalità
- Si crea insieme un piano di massima per la costruzione del software

Pianificazione: Planning game

- Ogni rilascio deve essere il più piccolo possibile e deve contenere le funzioni di maggior valore
- Piccoli rilasci che portano sempre un valore aggiunto al business
- Forma semplice di pianificazione e controllo
- Dialogo in evoluzione tra possibile e desiderabile
- Il piano complessivo viene migliorato continuamente in periodi di tempo sempre più brevi

Pianificazione: Strategie di base

- Obiettivo
  - *Massimizzare il valore del software prodotto dal gruppo*
- Strategia
  - *Investire il meno possibile*
    - Mettere in produzione velocemente le funzionalità più importanti riducendo al minimo il rischio
- Componenti
  - *I task da eseguire*
- Attori
  - *Il gruppo di sviluppo ed il management*

Pianificazione: Strategie di gestione

- **Gestione**
  - *Metriche*
  - *Redditività della commessa*
  - *Ruoli*
    - "Coach" e "Tracker"
  - *Interventi*
    - Riconoscere i momenti in cui il problema non può essere risolto dal gruppo di lavoro
    - Coraggio di accettarlo e comunicarlo
    - Capacità di decidere per intraprendere strade alternative
    - L'umiltà è la regola da adottare nel giorno dell'intervento.

Pianificazione: Coach e Tracker

- **Ruoli**
  - ***Coach*** *dirige*
  - ***Tracker*** *controlla*
  - *Il coach non fa il capo progetto ma*
    - è sempre disponibile come compagno di sviluppo
    - ha la visione a lungo termine degli obiettivi
    - esalta le caratteristiche dei programmatori
    - spiega il processo ai dirigenti

Costruzione: Strategia di sviluppo

- **Semplicità**
  - *Fare solo lo stretto indispensabile nella maniera più lineare*
- **Visibilità nel breve-medio**
  - *Si realizza attentamente oggi una soluzione per i problemi di oggi*
  - *Bisogna essere in grado domani di risolvere i problemi di domani*

Costruzione: Principi di sviluppo

- **Proprietà collettiva**
  - *In XP ogni soggetto che contribuisce al progetto è una parte integrante dell'intero team*
  - *Chiunque veda l'opportunità di migliorare una parte qualsiasi del codice deve poterlo fare, in qualunque momento*

Costruzione: Principi di sviluppo

- **Programmazione a coppie**
  - *Il codice scritto sulla stessa tastiera su un unico terminale*
- **Perché?**
  - *Ci sono poche forme di comunicazione più intense del faccia a faccia*
  - *Le coppie scrivono assieme, raggiungendo lo stesso livello di comprensione del progetto*
  - *Il dialogo migliora il processo di sviluppo del software*

Costruzione: Principi di sviluppo

- Refactoring
  - *I programmatori ristrutturano il sistema per migliorarlo senza cambiarne il comportamento*
  - *I programmatori si assicurano la possibilità di aggiungere sempre la funzionalità successiva con uno sforzo ragionevole*
  - *Il primo obiettivo è la rimozione della duplicazione*
  - *Ristrutturazione è più facile quando il progetto entra nel vivo perché si intuiscono meglio le chiavi del progetto*

Costruzione: Principi di sviluppo

- Metafora
  - *Il team condivide una semplice "storia" sul funzionamento del sistema*
  - ***Metafora***
    - Nell'XP la metafora sostituisce di molto "l'architettura". Con l'uso di una metafora si ha una buona probabilità di ottenere un'architettura facile da comunicare e da elaborare

Costruzione: Principi di sviluppo

- Settimana di 40 ore
  - *Ogni programmatore lavora con un ritmo che può sostenere nel tempo senza problemi*
  - *Le persone tollerano carichi di lavoro diversi. Nessuno può lavorare per 60 ore per settimane e settimane*

Costruzione: Principi di sviluppo

- Test
  - *Test funzionali*
    - Definiti col cliente
    - Utili per stabilire se il software è implementato correttamente
  - *Test unitari*
    - Si scrive il codice del test prima di scrivere il codice
  - *Automazione*
    - sotto pressione i test manuali vengono saltati
  - *I test*
    - Si conservano per tutta la durata del sistema
    - Durante lo sviluppo si eseguono frequentemente, per provare le evoluzioni del progetto
  - *Alcune fasi del test si fanno con o dal cliente*

Costruzione: Principi di sviluppo

- Standard di sviluppo
  - *Tutti lavorano in un ambiente prefissato*
  - *Le metodologie di interazione (interfaccia) sono fissate dall'ambiente*
  - *La parte funzionale deve rispettare standard di semplicità per poter essere capita gestita da tutti*
  - *Chi si occupa di "ambiente" non si occupa anche di "applicazioni"*

Rilascio: Principi generali

- Rilasci brevi e continui (integrazione continua)
  - *Piccole dimensioni dei rilasci*
  - *Poco impatto con il cliente*
  - *Maggiore facilità di interventi nel caso di errori*
  - *Visibilità di "crescita continua" del sistema*

Ogni singola pratica di sviluppo non funziona bene da sola

- Planning game
- Ciclo breve di rilascio
- Metafora
- Semplicità
- Testing
- Refactoring
- Programmazione a coppie
- Proprietà collettiva
- Integrazione continua
- Settimana di 40 ore
- Cliente sul posto
- Standard di codifica

Conclusioni

- Perché XP
  - *Cultura di produzione "just-in-time"*
  - *Liquidità diffusa*
    - Dominare il cambiamento è una abilità che consente di sopravvivere
  - *Cambio di prospettiva*
    - Centralità del cliente vs Centralità del prodotto
  - *Valorizzazione del patrimonio di conoscenza*

Criticità dei metodi di sviluppo agili

- Scalabilità su progetti di grandi dimensioni
- Difficoltà di comunicazione in team numerosi (documentazione scarsa e non standardizzata)
- Problematiche di rotazione e cambiamenti delle risorse del team
- Gli stakeholder potrebbero non volere (o potere) partecipare alle fasi dello sviluppo
- Difficoltà nell'esternalizzare il lavoro
- Il successo dipende fortemente dalle skill “agili” dei membri del team di sviluppo
- Poco adatti allo sviluppo di applicazioni safety-critical (aerospazio, militare, nucleare, …)

Scelta della metodologia di sviluppo

Non c'è una metodologia “migliore” tra tecniche agili e tradizionali

- La scelta della metodologia di sviluppo dipende da:
  - Dominio applicativo
  - Committente
  - Dimensioni del progetto
  - Innovatività del prodotto software
  - Team di sviluppo
  - Risorse del team di sviluppo
  - Risorse del committente
  - … !