Extreme programming e metodologie agili

Università degli Studi di Brescia, 8 Giugno 2007

Ing. Daniele Armanasco – daniele@armanasco.it
Ing. Emanuele DelBono – emanuele@codiceplastico.com

Enti organizzatori

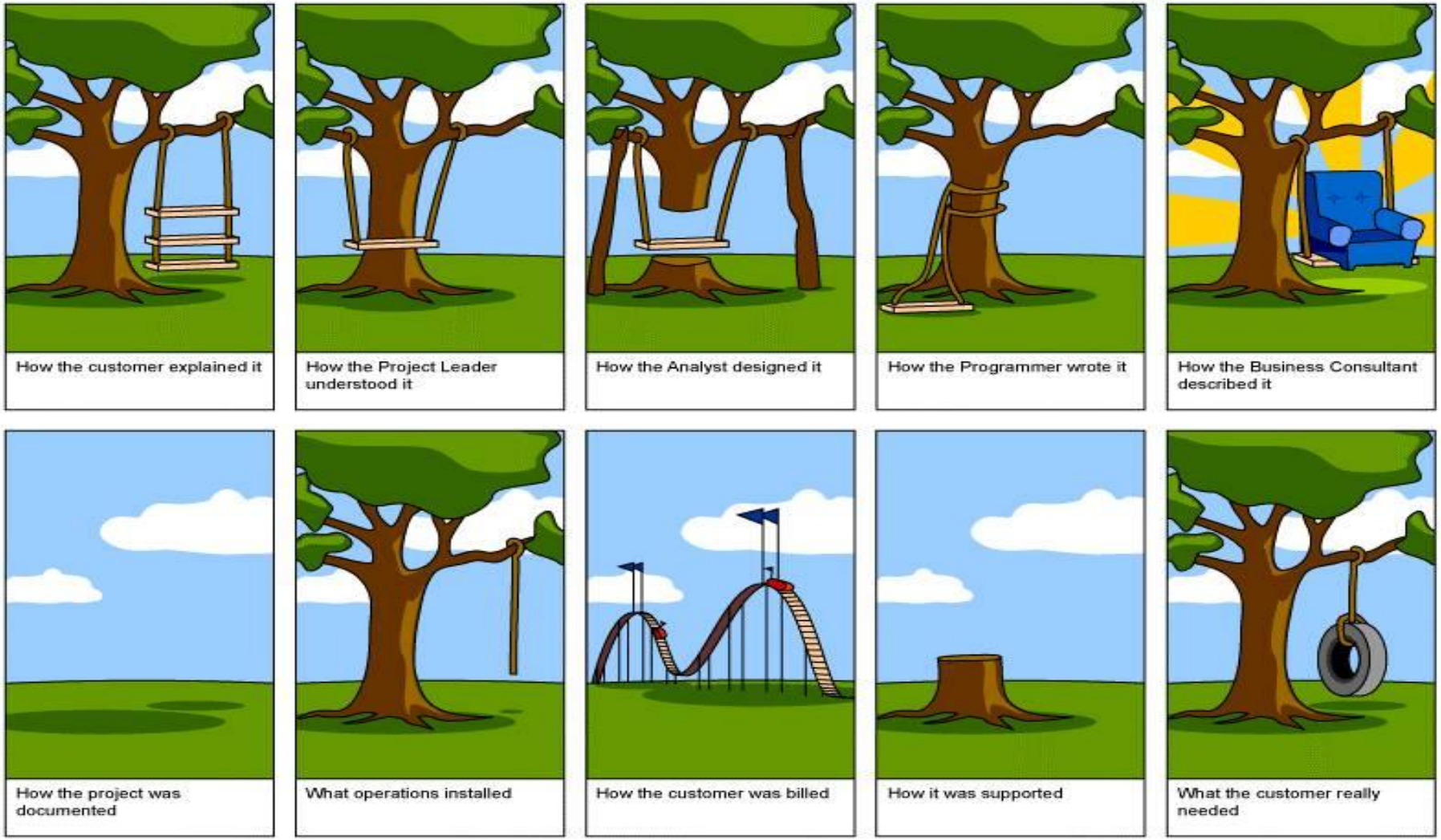
- Facoltà di Ingegneria dell'Università degli Studi di Brescia
- Commissione per l'ingegneria dell'informazione dell'Ordine degli Ingegneri di Brescia
  - www.ordineingegneri.bs.it
- Associazione GIB
  - Corso Creazione d'Impresa in collab. con l'ISU
  - Corso di inglese tecnico
  - www.gib.bs.it

Relatori

- Daniele Armanasco
  - Membro Commissione per l'ingegneria dell'Informazione dell'ODI
  - Consigliere e Delegato per l'Università del GIB
  - www.armanasco.it
  - daniele@armanasco.it
  - http://blogs.ugidotnet.org/DanBlog

- Emanuele DelBono:
  - Webcast Microsoft: Design Patterns by example
  - www.codiceplastico.com
  - emanuele@codiceplastico.com
  - http://blogs.ugidotnet.org/BlogEma

Cosa accade spesso…

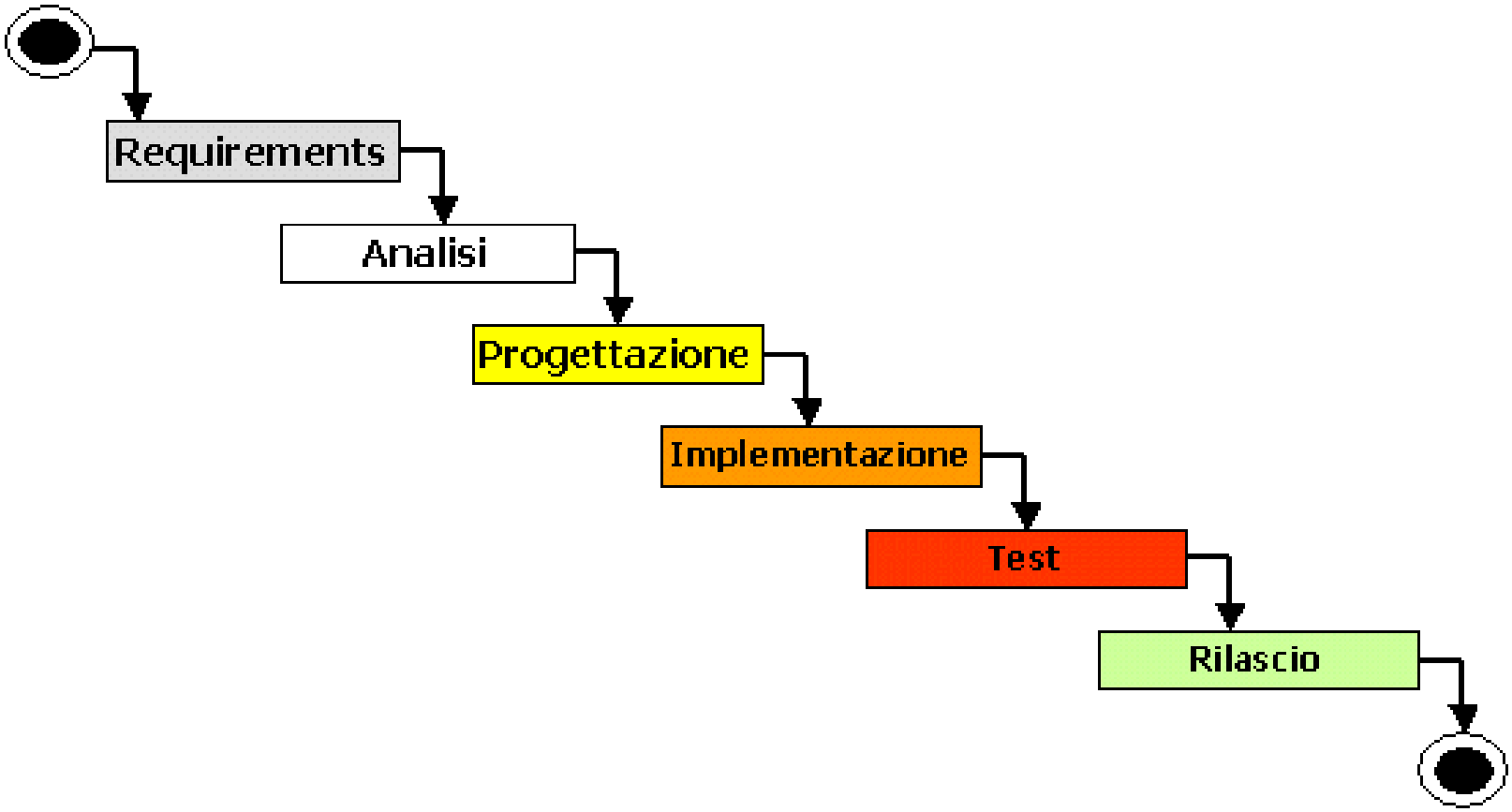


Processi per lo sviluppo del software

- Storicamente:
  - Code & fix
  - Metodologie ingegneristiche (a cascata, incrementale, a spirale, …)
    - Predire tutto (possibile se i requisiti sono stabili e identificabili)
    - Evitare il cambiamento
- Dagli anni ’90:
  - Metodologie agili
    - Adattive invece che predittive
    - Accettano e non ostacolano il cambiamento
    - Orientate all’individuo
    - Meno orientate alla documentazione

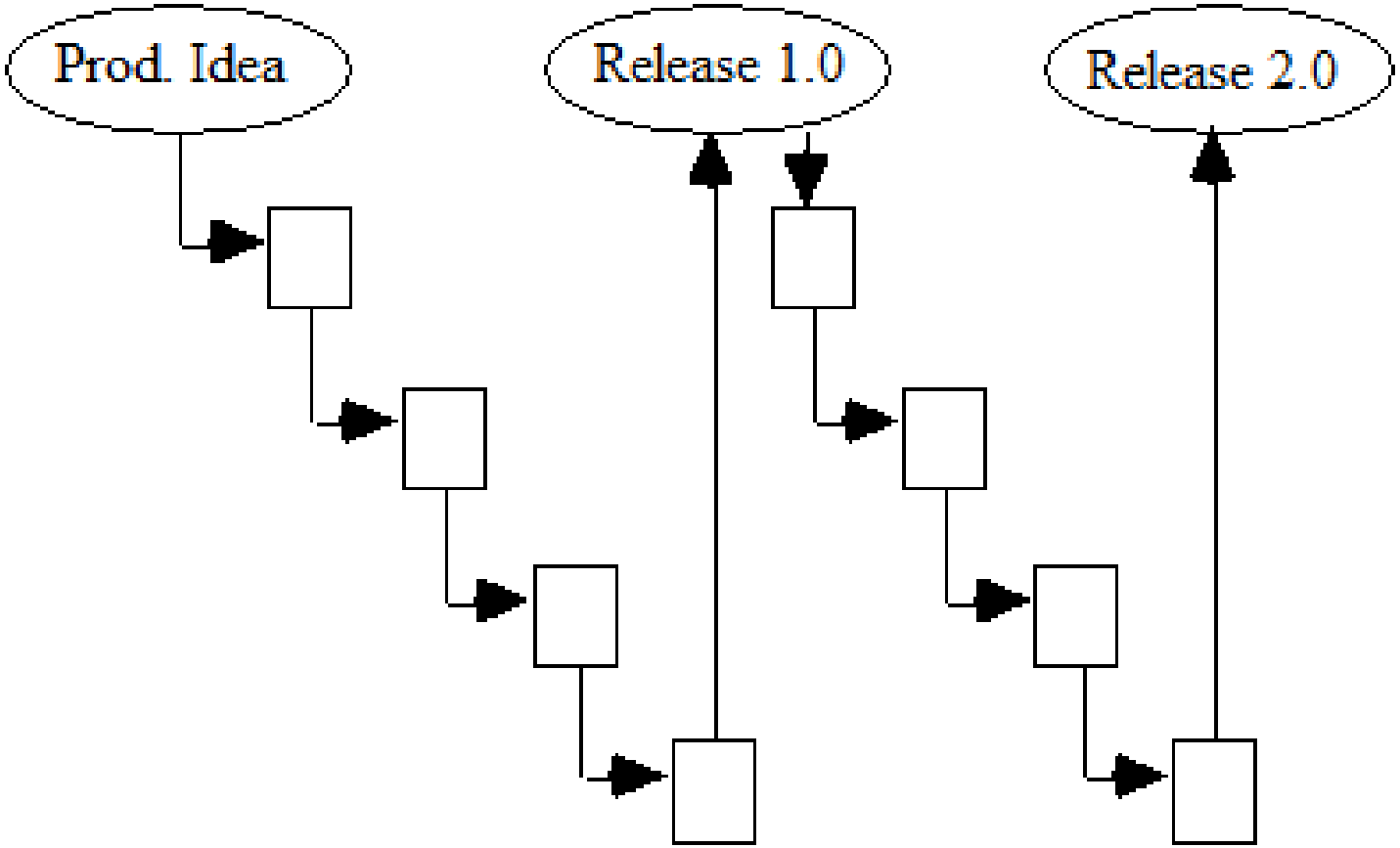
Metodologie ingegneristiche

- A cascata



Metodologie ingegneristiche

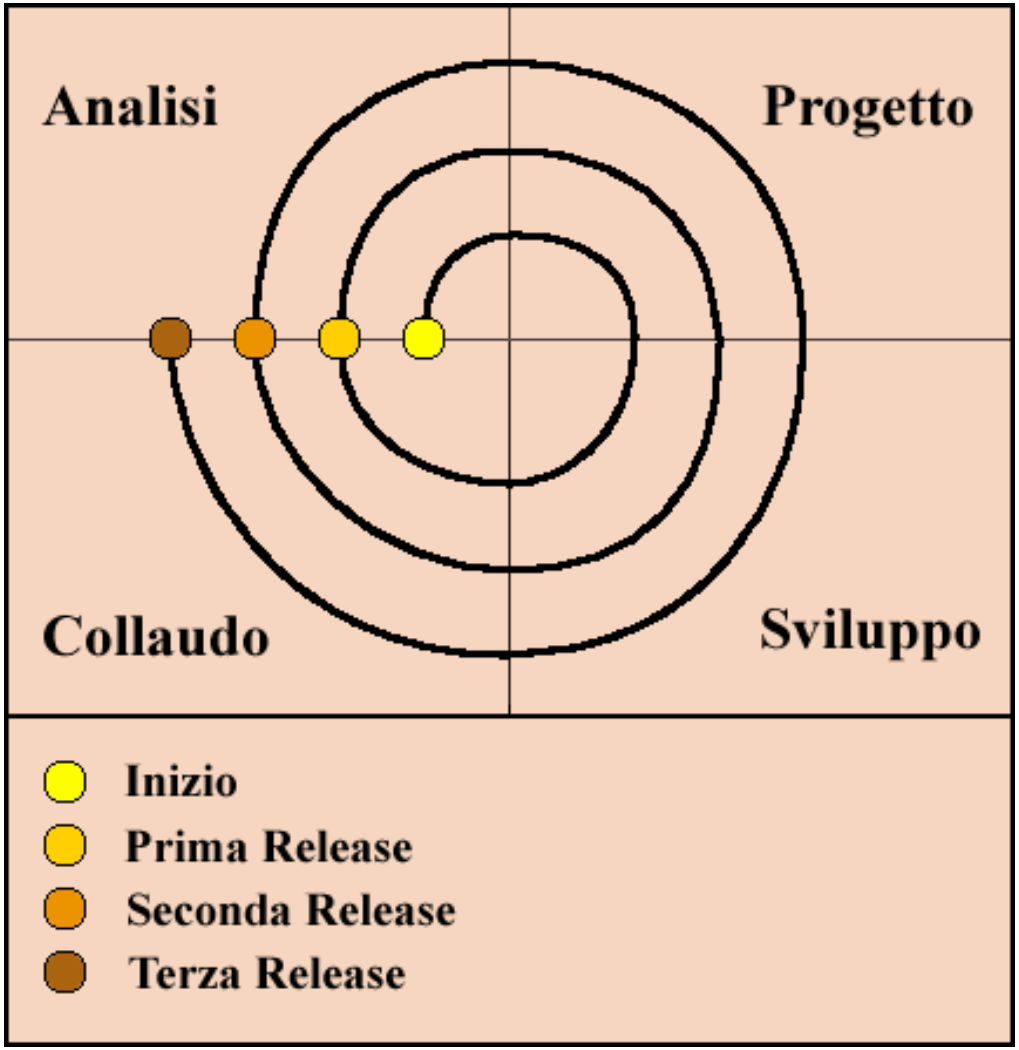
- Incrementale



Metodologie ingegneristiche

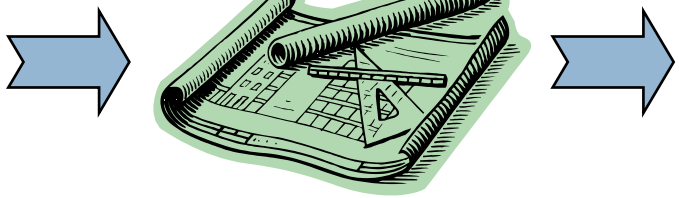
- A spirale

Raggio = costo accumulato
Lunghezza = progresso



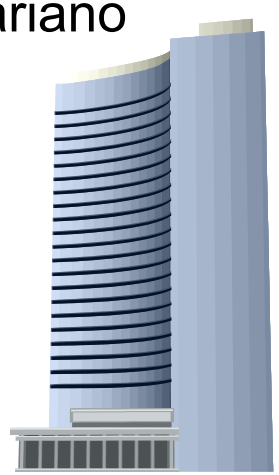
Funzionano bene se …

- La progettazione
  - è costosa perché fatta da personale specializzato
  - produce un risultato facilmente implementabile
  - rappresenta una piccola parte di tutta l'attività


Nello sviluppo del software:

- Difficile e costoso ottenere un progetto passabile direttamente ai programmatori
- Il tempo per la programmazione è poco rispetto a quello per la progettazione
  - Jack Reeves: la realizzazione è a tempo zero, perché consiste nell'utilizzo del compilatore e linker
- I requisiti cambiano continuamente
  - Nell'economia moderna le esigenze del committente variano velocemente
  - Difficile valutare una funzionalità finché non la si usa



Metodologie agili (“leggere”)

- “Guidare non significa mettere la macchina nella giusta direzione. Significa fare sempre attenzione, facendo piccole correzioni in questa o in quella direzione”

  La mamma di Kent Beck

- Adattività consente di controllare l’imprevedibilità
- Serve un meccanismo di feedback attendibile

Breve storia dei metodi agili

- I primi metodi agili sono nati negli anni ’90
- Agile Manifesto:
  - Nel 2001 17 esponenti di tali metodi, tra cui Kent Beck (XP), Martin Fowler e Ward Cunningham (wiki), si sono riuniti e hanno adottato il termine “metodi agili” e creato l’Agile Manifesto con l’intenzione di diffondere un nuovo modo di sviluppare applicazioni software
  - www.agilemanifesto.org

Feedback: sviluppo iterativo

- Idea già presente in passato (sviluppo a spirale, evolutivo)
- realizzare frequentemente versioni funzionanti del sistema finale che possiedano un sottoinsieme delle funzionalità richieste
- gli errori (sia in termini di malfunzionamenti che in termini di una cattiva comprensione o della variazione dei requisiti) emergono realmente quando si utilizza un sistema

Quando utilizzare lo sviluppo iterativo

- Come tutte le cose non è sempre adatto …

  … usatelo solo nei progetti che volete abbiano successo!

"You should use iterative development only on projects that you want to succeed"

Martin Fowler

Cliente adattivo

- Non tutti i clienti si prestano all'adozione di un processo agile
- Adattare un contratto a prezzo fisso ad un processo agile può generare conseguenze dolorose per tutti
- Nell'approccio agile in genere si fissano il tempo e il prezzo; il risultato verrà stabilito man mano in accordo con il cliente
  - Non ha senso chiedersi se si sono rispettati tempi e costi
  - Ha senso chiedersi quale valore ha ottenuto il cliente

Vantaggi per il cliente

- Influisce maggiormente sull'andamento del progetto
- Se vi sono problemi vengono rilevati in anticipo
- Il progetto è guidato dal valore che le funzionalità assumono nei confronti dell'attività del cliente

L’individuo in primo piano

- Le persone sono parti del sistema fondamentali e difficili da sostituire
- se si considerano gli sviluppatori unità di programmazione intercambiabili si otterrà proprio questo, perdendo le persone competenti
- gli sviluppatori sono professionisti responsabili e sono i più adatti a determinare come svolgere il lavoro

Il management deve aver presente che …

- I dirigenti, anche se sviluppatori in passato, non hanno conoscenze tecniche aggiornate in un settore con alto ritmo di cambiamento
- La gestione basata sulle misure è più adatta per lavori semplici e ripetitivi
- Misurare le prestazioni è pericoloso se non si riescono a misurare tutti i fattori, perché i lavoratori adatteranno la propria modalità di lavoro per far rilevare le misure migliori, anche riducendo la reale efficacia
- Anche gli sviluppatori necessitano di uno stretto contatto con gli esperti aziendali

Inoltre

- Comunicazione:
  - Deve essere continua per informare tutti dei cambiamenti rapidi e continui
- Processo adattivo:
  - il processo deve essere adattato al compito del momento e deve evolvere durante il progetto e con l'esperienza del team
  - regolari (spesso ad ogni iterazione) revisioni del processo

Riepilogando

- Meglio un processo predittivo se:
  - Requisiti stabili
  - team di taglia superiore a 100 unità
  - prezzo fisso per fissata quantità di lavoro
- Meglio un processo agile se:
  - requisiti incerti o mutevoli
  - sviluppatori responsabili e motivati
  - cliente consapevole di questi aspetti

Nella pratica …

purtroppo

- nella pratica non funziona

purtroppo

- si tratta della solita utopia

purtroppo

- nessuno utilizza le metodologie agili

Italian Agile Day

- Alcuni numeri della giornata del 1 dicembre 2006:
  - 2 sessioni plenarie
  - 18 sessioni open space "pre-semi-organizzate" (prima esperienza di conferenza Open Space in Italia)
  - un on-going XP Game durato tutta la giornata di modo che il maggior numero possibile di persone potesse parteciparvi
  - molto spazio per discussioni spontanee
  - 180 partecipanti


Marlboro
vodafone
vodafone
FIAT

XP 2007

- 8th International Conference on Agile Processes in Software Engineering and Extreme Programming – Como, 18-22 Giugno 2007
- Keynote:
  - “The GranPrix starts at 2 o'clock: a race to race Software Development eXPerience” – Piergiorgio Grossi
  - “Ease at Work” – Kent Beck
- www.xp2007.org

Adobe

- La nuova Adobe Creative Suite 3 (CS3) è stata realizzata usando una metodologia agile
- Durante lo sviluppo era sempre disponibile una beta version funzionante
- Implementazione incrementale delle features
- Fase di test molto ridotta
  - http://www.regdeveloper.co.uk/2007/03/08/adobe_cs3_development/
  - http://blogs.adobe.com/jnack/2007/03/agile_development.html

Introduzione a eXtreme Programming

Università degli Studi di Brescia, 8 Giugno 2007

Ing. Daniele Armanasco – daniele@armanasco.it
Ing. Emanuele DelBono – emanuele@codiceplastico.com

Tutto cambia

- L’esperienza nella realizzazione di un’applicazione software rende evidente una realtà di cui noi dovremmo essere coscienti...
- La velocità del cambiamento sta aumentando ciò che era vero sei mesi fa potrebbe non essere più vero fra sei mesi
- Non possiamo basarci su ciò che abbiamo appreso in passato per agire in futuro
- Tutto cambia

Cos’è XP?

- Extreme Programming è un metodo di sviluppo software che vi permette di gestire il cambiamento mutando radicalmente l'approccio ad esso.
- Invece di evitarlo o ignorarlo, *abbracciarlo*!

Cos’è XP?

- XP è basata su un insieme di valori:
  - semplicità
  - comunicazione
  - feedback
  - coraggio
- Questi valori, che sembrano "astratti", sono fondamentali nel metodo: senza di essi e' difficile applicarlo.

Semplicità

- Di solito agli sviluppatori piace prevedere il futuro e per questo realizzano framework “general purpose”
- Il risultato sono programmi molto complessi, difficili da mantenere e con molte funzionalità inutilizzate
- XP dice: “Do The Simplest Thing That Could Possibly Work” [K. Beck]
- La cosa semplice non è “Copia&Incolla” ma la cosa semplice per chi domani dovrà leggere il codice

Comunicazione

- I membri del team devono comunicare senza paura
- Quando una nuova feature viene richiesta non ci si deve trattenere dal dire le proprie opinioni

Feedback

- In un mondo che cambia in fretta l'unico modo per non perdere la rotta è dare e ricevere continuamente feedback
- Chi da il feedback?
  - Gli sviluppatori
  - Il cliente quando verifica ciò che abbiamo realizzato
  - Il management
  - Il software stesso durante i test

Coraggio

- In molte occasioni durante lo sviluppo di un’applicazione si devono fare delle scelte sul design
- A volte bisogna avere il coraggio di cambiare una grossa parte di applicazione per migliorarne il design
- A volte bisogna avere il coraggio di buttare parte del codice scritto perché obsoleto

Cos'è XP?

- XP è una metodologia basata su un insieme di pratiche:
  - Whole Team
  - Planning Game
  - Customer Tests
  - Small Releases
  - Simple Design
  - Pair Programming
  - Test Driven Development
  - Design Improvement
  - Continuous Integration
  - Collective Code Ownership
  - Coding Standard
  - Metaphor
  - Sustainable Pace

Whole Team

- Il team in XP agisce come un unica entità!
- Attenzione perchè il cliente fa parte del team ed è suo compito:
  - fornire i requisiti
  - impostare le priorità
  - guidare il progetto
  - definire i test di accettazione
  - chiarire quando è necessario

Whole Team

- Ovviamente il team: ha dei programmatori ☺
- Può avere dei tester, che aiutano il cliente a definire i test di accettazione
- Ha un coach, che guida il team nello sviluppo
- Ci può essere un manager che gestisce risorse e comunicazione con l'esterno

Planning Game

- In XP l'attività di pianificazione è molto importante, ma invece di essere di tipo predittivo (sfera di cristallo) pone un enfasi molto più alta sulla possibilità di cambiare (*steering*).
- Il cliente ha la possibilità in ogni momento di cambiare idea e di scegliere di sviluppare la funzionalità X invece di quella Y.
- La pianificazione in XP avviene in due momenti:
  - Release Planning
  - Iteration Planning

Planning Game: Release Planning

- Nel release planning il cliente presenta le funzionalità richieste al team sotto forma di storie ("user stories"), delle quali il team fornisce una stima di massima. Il cliente, conoscendo i costi e sapendo cosa è più importante per il suo business, definisce un piano di massima del progetto.

Planning Game: release planning

- I release planning iniziali sono ovviamente imprecisi: le stime sono grossolane
- Non si sa ancora quale sarà la produttività del team
- Tuttavia sono molto utili! Danno un feedback immediato e aiutano a capire cosa deve fare il sistema. Il release planning periodicamente può essere
  - rivisto
  - riemesso ex-novo
- Questo dipende dal team: XP e' adattabile :-)

Planning Game: iteration planning

- L'iteration planning e' la pratica che permette ad un team XP di "sterzare" ogni tot settimane. Tipicamente un team XP realizza il suo software in iterazioni di due settimane, rilasciando software alla fine di ogni iterazione.

Planning Game: iteration planning

- Durante l'iteration planning: il cliente presenta le funzioni che desidera nelle prossime 2 settimane (si basa sulle stime del release planning)
- I programmatori le splittano in "task"
- Le stimano con maggiore precisione
- Ad ogni iterazione si misura il lavoro svolto nella precedente e si raffinano le stime per le prossime iterazioni.
- Il cliente ne ha evidenza del lavoro svolto
- Eventualmente il cliente può cancellare *presto* il progetto senza aspettare che continui a succhiare risorse all'infinito!

Planning Game

- Quando il cliente presenta le storie definisce uno o più test di accettazione. I test devono essere eseguibili in modo automatico, i test manuali sono frequentemente saltati.
- I migliori team XP definiscono questi test sotto forma di codice eseguibile, trattato alla stessa stregua dei test realizzati durante lo sviluppo.

Small Releases

- Il team rilascia software funzionante ogni iterazione, fornendo il valore richiesto dal cliente, che puo' decidere se:
  - valutarlo da solo
  - usarlo come prototipo
  - renderlo disponibile agli utenti finali
- Sembra impossibile lavorare così in fretta? Eppure si può, grazie all'insieme delle pratiche che formano XP.

Simple Design

- In XP non c'è una "fase" di design, o comunque design up-front: il design e' continuo! Il design in un progetto XP è esattamente quello che serve per quel sistema, nulla di più e nulla di meno.
- Si fa design quando si scrivono i test, cercando di identificare i servizi corretti (e solo quelli) che servono ai nostri oggetti

Simple Design

- Si fa design durante il refactoring, eliminando le duplicazioni e facendo emergere quelle strutture di collaborazione fra oggetti che alla fine riconosciamo come pattern. E' necessario che il design sia più che buono, che il codice sia malleabile e che non ci siano rigidità che impediscano di gestire il cambiamento.
- Il design è continuo: la maggior parte del tempo che trascorrete in un team XP in realtà è dedicato al design.

Pair Programming

- Beh, e' molto semplice: tutto il software di un team XP è creato da coppie di programmatori, che siedono fianco a fianco alla stessa macchina.
- "guidano" (usano mouse e tastiera) uno alla volta
- Le coppie cambiano spesso durante la giornata
- I singoli trasmettono la conoscenza al team
- Gli esperti possono skillare i nuovi membri
- Il code-review è continuo
- Sembra inefficiente? Beh, non lo è. Dopo che vi sarete abituati ad avere certi risultati con PP, sarà difficile accontentarsi se mai tornerete indietro.

Pair Programming

- Quando programmate da soli a volte sentite che state lavorando ad una velocità super, vero? Poi, all'improvviso, perdete il "fuoco" e cominciate a divagare...
- Il vostro pair fa in modo che invece la vostra efficienza sia costante!

Pair Programming

- Studi hanno dimostrato che:
  - i pair non usano più ore uomo dei singoli
  - i pair creano un numero inferiore di difetti
  - i pair producono meno linee di codice
  - i pair si divertono di più

Test Driven Development

- Il Testing non è una fase a se che viene fatta alla fine dello sviluppo
- Il Testing viene fatto continuamente e in XP nascono prima i test (Unit Test) del codice da testare
- Il codice della nostra applicazione è guidato dai test

TDD e ritmo

- Il flusso è questo:
  - Scrivo un test
  - Verifico che fallisce
  - Scrivo il codice che soddisfa il test
  - Verifico che il test passi
  - Faccio refactoring
  - Verifico che il test passi
- Faccio uso di framework (*Unit) per automatizzare l'esecuzione dei test

Refactoring

- Cosa significa?
  - rimuovere le duplicazioni
  - incrementare la coesione dei componenti
  - abbassare l'accoppiamento fra le componenti
- Appena abbiamo qualcosa che funziona rifattorizziamo:
  - con piccoli passi (5 minuti)
  - lanciando i test e mantenendoli verdi

Refactoring

- E' richiesto in XP non fare il checkin del codice fino a che:
  - tutti i test passano
  - non ci sono duplicazioni
  - il codice è il più espressivo possibile
  - il codice è il più semplice possibile

Refactoring

- I commenti “puzzano”: dovrebbero essere minimizzati perché spesso non dicono la verità
- Costituiscono un ammissione che non siamo riusciti ad esprimere con il codice le nostre idee
- In generale il codice deve essere "speaking": leggendolo si deve capire al volo cosa fa, deve essere espressivo

Continuous integration

- In XP il codice si integra più volte al giorno, e in una giornata si realizzano *diverse* build.
- Tutti sono in grado di farlo
- I bug diminuiscono perché il sistema viene testato nella sua interezza molto spesso, e i malfunzionamenti si rilevano presto
- Non ci sono parti "congelate" del codice, sulla quale delle persone lavorano (da mesi?) e che rendono impossibile consegnare il sistema

Continuous integration

- In poche parole: tutti possono modificare qualunque parte del codice in qualunque istante
- Nessuno è il "proprietario" di qualcosa
- Tutto il codice viene posto all'attenzione di tutti
- Tutti conoscono il codice
  - si evitano duplicazioni
  - si evita di mettere le cose nel posto sbagliato

Collective Code Ownership

- Un team XP deve seguire uno standard di codifica condiviso: "quale" non è importante, è importante che tutto il codice segua quello stabilito tutti devono concordare sulle convenzioni
- lo standard "emerge" col passare del tempo
- ... e si evolve continuamente
- L'unica regola da rispettare? Il codice deve essere speaking, vi deve parlare!

Metaphor

- Un team XP deve condividere una metafora, una visione comune di come il sistema funziona. Una metafora di solito non viene individuata durante lo svolgimento del progetto, ed evolve insieme ad esso.
- In ogni caso un team XP usa un sistema di nomi comune per essere sicuro che ciascuno capisca come il sistema funziona e dove cercare per trovare una funzionalità.

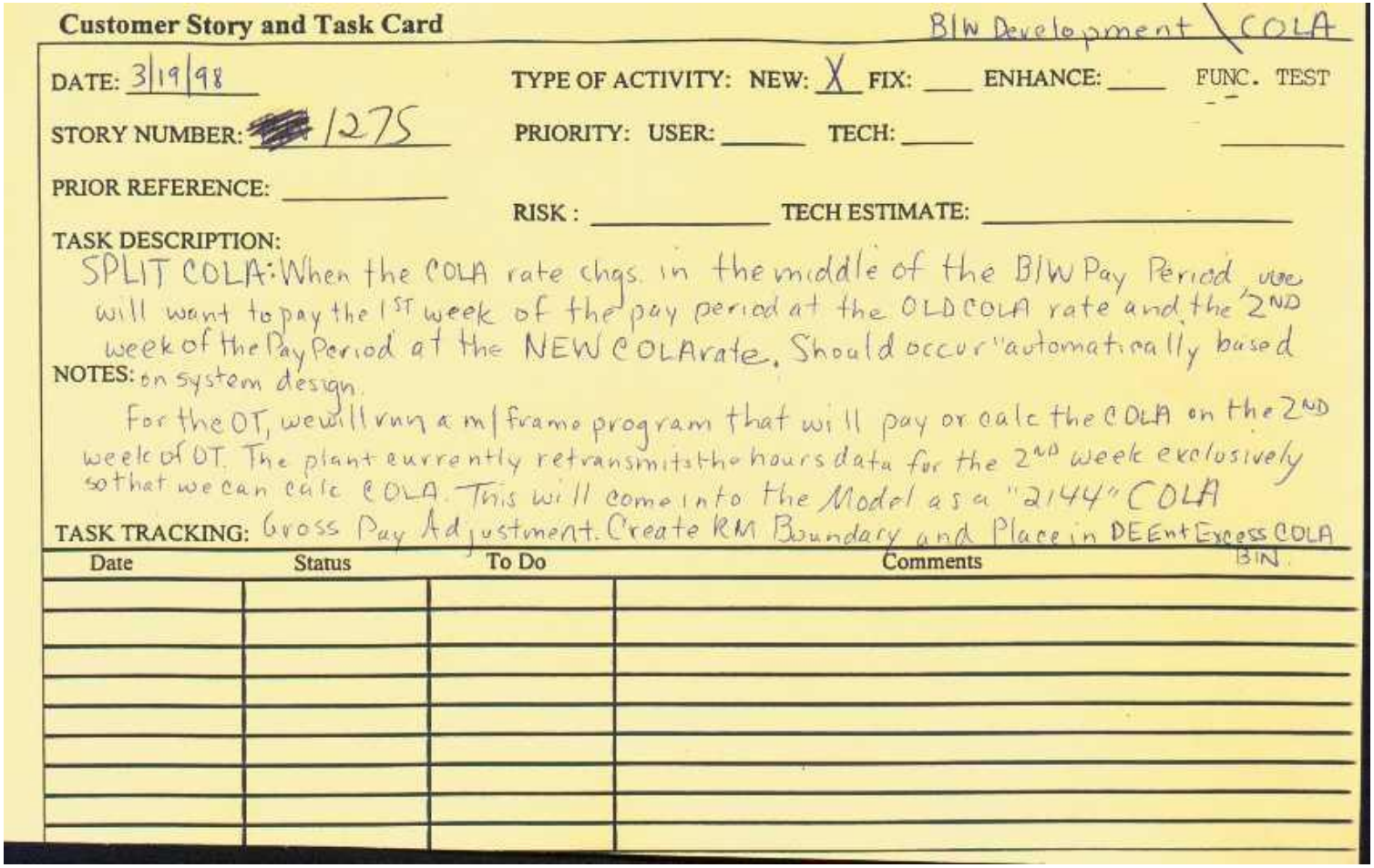
Sustainable Pace

- Già nota come "40 hours week" questa pratica indica semplicemente che l'overtime deve essere occasionale e solo quando sia assolutamente necessario
- Non puoi dare il massimo se sei stanco
- I problemi che causi influiscono su tutto il team
- Devi essere riposato
- Un team XP quando e' sul pezzo lavora duro, non molla un attimo, per questo deve lavorare ad un ritmo accettabile.
- "I team XP sono lì per vincere, non per morire"

Quindi…

- XP e' costituita da condivisione di valori e da semplici pratiche, che però non sono semplici affatto da applicare!
- Inoltre:
  - ci vuole molta disciplina
  - ci vuole pratica nel processo
  - ci vuole conoscenza tecnica
  - ci vogliono tool adatti, veloci e versatili
- Ma soprattutto ci vuole un team!

Story Card



Customer Story and Task Card — BIW Development \ COLA

DATE: 3/19/98 — TYPE OF ACTIVITY: NEW: X FIX: ___ ENHANCE: ___ FUNC. TEST

STORY NUMBER: [redacted] 1275 — PRIORITY: USER: ___ TECH: ___

PRIOR REFERENCE: ___ — RISK: ___ TECH ESTIMATE: ___

TASK DESCRIPTION:

SPLIT COLA: When the COLA rate chgs. in the middle of the BIW Pay Period we will want to pay the 1ST week of the pay period at the OLD COLA rate and the 2ND week of the Pay Period at the NEW COLA rate. Should occur "automatically based on system design.

NOTES:

For the OT, we will run a m/frame program that will pay or calc the COLA on the 2ND week of OT. The plant currently retransmits the hours data for the 2ND week exclusively so that we can calc COLA. This will come into the Model as a "2144" COLA

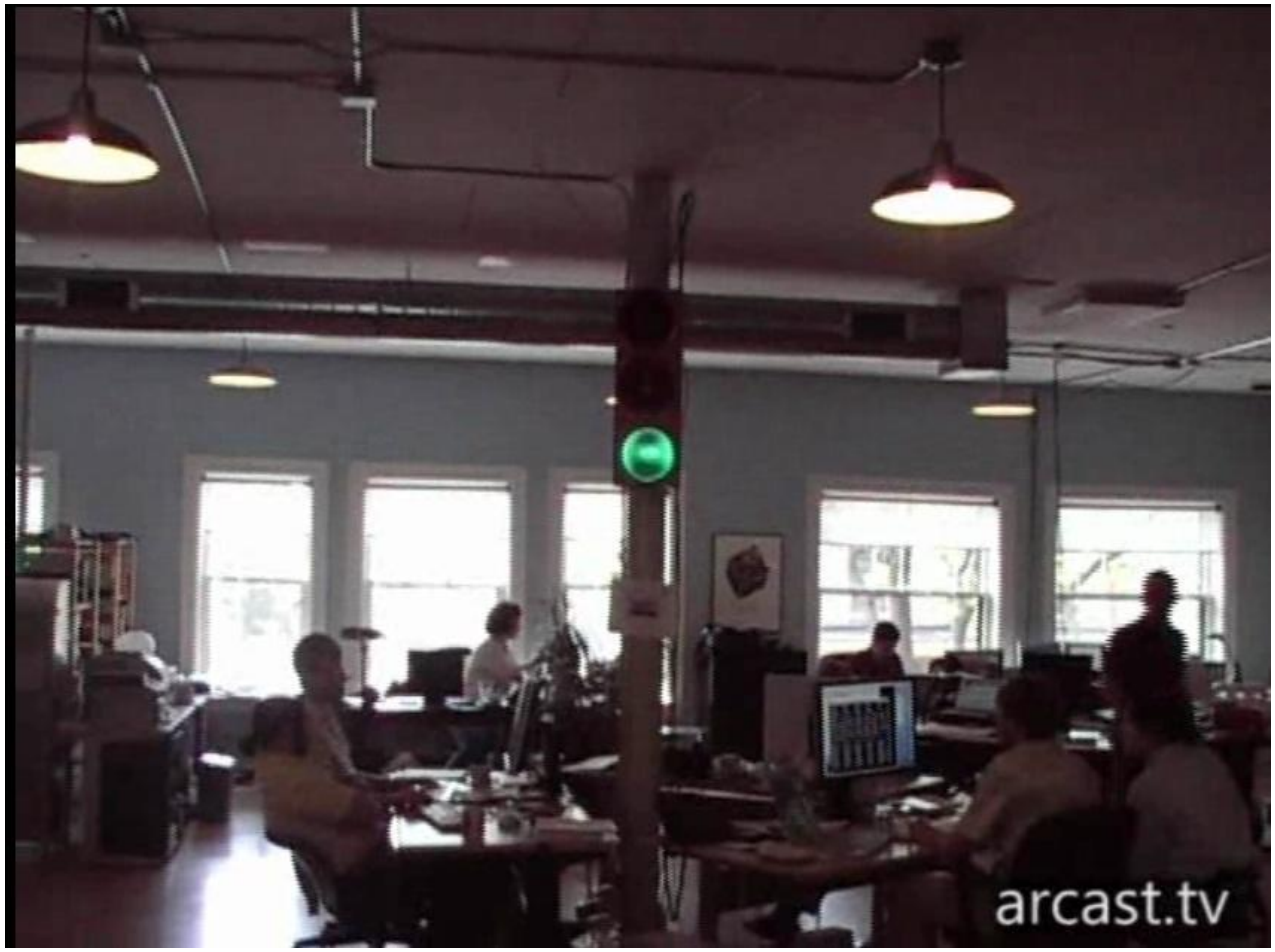
TASK TRACKING: Gross Pay Adjustment. Create RM Boundary and Place in DEEntExcessCOLA BIN.

Date	Status	To Do	Comments

La lavagna delle story card

Planning game

Un ufficio “agile”



Riferimenti

- Altre metodologie agili:
  - Scrum http://www.controlchaos.com/
  - Crystal http://alistair.cockburn.us/index.php/Crystal_methodologies
  - Dynamic System Development Method http://www.dsdm.org/
  - Feature Driven Development http://www.featuredrivendevelopment.com

- La nuova metodologia – articolo di M. Fowler scaricabile all'indirizzo http://www.martinfowler.com/articles/newMethodology.html
- An Introduction to Agile Methods – Steve Hayes e Martin Andrews www.wrytradesman.com/articles/IntroToAgileMethods.pdf
- http://en.wikipedia.org/wiki/Extreme_Programming
- http://www.extremeprogramming.org
- http://www.xprogramming.com/

Riferimenti 2

- http://www.agilemovement.it/ la community italiana dedicata alle metodologie agili fondata da Marco Abis
- http://www.agileday.it/ la conferenza italiana dedicata alle metodologie agili in cui incontrarsi, confrontarsi, condividere esperienze e conoscere esperti ed aziende di primo piano
- http://www.extremeprogramming.it/ lo user group italiano di extreme programming, qui trovi le persone più attive su XP in italia
- http://www.agilemanifesto.org/ il manifesto contiene i valori e i principi comuni a tutti i metodi agili
- http://www.extremeprogramming.org/ una spiegazione introduttiva chiara e sintetica su XP
- http://www.quinary.it/pagine/innovation/res.htm la descrizione di un caso applicato di XP
- http://www.frappr.com/italianagilemovement la mappa degli agilisti italiani
- http://it.groups.yahoo.com/group/extremeprogramming-it/ Mailing list italiana su XP