

Extreme Programming (XP)

- Principi di XP
 - Avere feedback rapidamente
 - Assumere la semplicità
 - Cambiamenti incrementali
 - Supportare i cambiamenti
 - Produrre lavoro di qualità

Pratiche di XP [Beck]

- Gioco di pianificazione
- Piccole release
- Metafora
- Testing
- Refactoring
- Pair Programming (programmazione a coppie)
- Cliente in sede
- Design semplice
- Possesso del codice collettivo
- Integrazione continua
- Settimana di 40 ore
- Usare gli standard per il codice

Gioco di pianificazione

- Storie utente = casi d'uso leggeri
- Descrizione storie: 2-3 frasi su una card che
 - Sono importanti per il cliente
 - Possono essere testate
 - Permettono di ricavare una stima del loro tempo di sviluppo
 - Possono essere associate a priorità

Story Card

Customer Story and Task Card BIW Development / COLA

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

STORY NUMBER: ~~1275~~ 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/frames 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 DEEnt Excess COLA

Date	Status	To Do	Comments	BIN

Board per Story Card



Ing. E. Tramontana - Processo XP - 30-Mar-07 5

Gioco di pianificazione

- Gli utenti scrivono storie
- Gli sviluppatori le stimano
- Gli utenti dividono, fondono e assegnano priorità alle storie
- Pianificare l'intera release (grossolanamente) e la nuova iterazione
 - Non pianificare troppo in avanti

Ing. E. Tramontana - Processo XP - 30-Mar-07 6

Piccole release

- Rendere ogni release il più piccola possibile
- Effettuare un design semplice e sufficiente per la release corrente
- Piccole release forniscono
 - Feedback rapidamente
 - Un senso di: "ho ottenuto qualcosa di valido"
 - Rischio ridotto
 - La fiducia del cliente
 - Possibilità di fare aggiustamenti per requisiti che cambiano

Ing. E. Tramontana - Processo XP - 30-Mar-07 7

Metafora

- Guidare il progetto con una singola metafora
 - Es.: "La UI è un desktop"
 - Deve rappresentare l'architettura
 - Rende le discussioni più semplici
 - Il cliente deve essere a suo agio con essa

Ing. E. Tramontana - Processo XP - 30-Mar-07 8

Design Semplice

- Il giusto design per il software
 - Passa i test
 - Non ha parti duplicate
 - Esprime ciascuna intenzione importante per i programmatori
 - Ha il numero più piccolo di classi e metodi
- Non preoccuparsi di dover apportare cambiamenti dopo
 - Fare la cosa più semplice che può funzionare
 - Paga quanto usi

Testing

- Si testa tutto ciò che potenzialmente può andar male, per tutto il tempo
- I test sono la specifica
 - Una specifica eseguibile
- Due tipi di test
 - Test funzionali
 - Unit Test

Test

- Test funzionali
 - Specificati dall'utente (punto di vista dell'utente)
 - Effettuati da utenti, sviluppatori e team di testing
 - Automatizzati
 - Eseguiti almeno giornalmente
 - Parte della specifica
- Unit Test
 - Scritti dagli sviluppatori (punto di vista del programmatore)
 - Scritti prima e dopo la codifica
 - Supportano design, codifica, refactoring e qualità

Pair Programming

- Ruolo di uno dei partner
 - Usa il mouse e la tastiera
 - Pensa al miglior modo di implementare il metodo
- Ruolo dell'altro
 - L'approccio funzionerà?
 - Pensa ai test
 - Potrebbe essere fatto più semplicemente?
- Scambio dei partner
- Pair programming aiuta la disciplina, sparge la conoscenza sul sistema



Possesso del codice collettivo

- Chiunque può aggiungere qualunque codice su qualunque parte del sistema
- Unit test proteggono le funzionalità del sistema
- Chiunque trova un problema lo risolve
- Ciascuno è responsabile per l'intero sistema

Integrazione continua

- Integrazione del codice testato ogni poche ore (max un giorno)
- Tutti gli unit test devono essere superati
- Se un test fallisce la coppia deve riparare il codice
- Se non può ripararlo, buttare il codice e ricominciare

40 ore a settimana

- Se per te non è possibile fare il lavoro in 40 ore, allora hai troppo lavoro
- 40 ore a settimana ti lasciano “fresco” per risolvere i problemi
- Previene l'inserimento di errori difficili da trovare
- Pianificazioni frequenti evita a ciascuno avere troppo lavoro
- Ore extra di lavoro è sintomo di un problema serio

Cliente sul sito

- Scrive i test funzionali
- Stabilisce priorità e fornisce il contesto per le decisioni dei programmatori
- Risponde alle domande
- Porta avanti il suo proprio lavoro

- Se non puoi avere il cliente sul sito, forse il progetto non è così importante?

Standard di codifica

- Costruzioni complicate non sono permesse
 - **Mantenere le cose semplici**
- Codice appare uniforme
 - **Più facile da leggere**
- Nessuna necessità di riformattare il codice
 - Spazi
 - Tab
 - Parentesi graffe

Refactoring

- Refactoring significa migliorare la struttura del codice senza influenzarne il comportamento
- Fatto in piccoli passi
- Supportato dagli unit test, design semplice e pair programming
- Puntare a codice senza ripetizioni
- Refactoring fatto in coppia dà più coraggio

Pianificazione

- Gli addetti al business prendono decisioni su
 - **Date**
 - **Contesto**
 - **Priorità**
- Sviluppatori prendono decisioni sullo sviluppo
 - **Stime**

Pianificazione release

- I clienti scrivono le storie
- Gli sviluppatori le stimano
 - **Se troppo complesse da stimare ritornare dal cliente e far dividere le storie**
- Il cliente dà le priorità alle storie e riempie 3 settimane scegliendo le storie
 - **Non preoccuparsi delle dipendenze**
- Fare una o due release in questo modo

Pianificazione iterazioni

- Per la release attuale
 - Gli sviluppatori dividono ciascuna storia in task
 - Stimano i task insieme
 - Ciascuno prende l'impegno di realizzare un task
- Quali task vengono prima?
 - Quelli più rischiosi

In breve

- XP focalizza sul codice
 - Fare solo le cose che sveltiscono la produzione del codice
 - Codifica e test
- XP si orienta sulla gente
 - La conoscenza è trasferita attraverso la comunicazione tra la gente
- XP è leggero
 - Rimuovere i costi aggiuntivi
 - Creare prodotti di qualità tramite test rigorosi
- I principi di XP non sono nuovi