ExTreme Programming

Le dodici regole (o pratiche) che sono alla base di *Extreme Programming* possono essere raggruppate in quattro aree.

Feedback a scala fine

- *Pair programming* - Due programmatori lavorano insieme su una sola workstation, uno sta alla tastiera e l'altro ragiona sull'approccio e pensa se può funzionare. Questo rende il codice prodotto di migliore qualità. I due programmatori devono avere la stessa esperienza.
- *Planning Game* - è una riunione di pianificazione che avviene una volta per iterazione, tipicamente una volta a settimana.
- *Test driven development* - i test automatici (sia unitari che di accettazione) vengono scritti prima di scrivere il codice.
- *Whole Team* - in XP, il "cliente" non è colui che paga il conto, ma la persona che realmente utilizza il sistema. Il cliente deve essere presente e disponibile a verificare (sono consigliate riunioni settimanali).

Processo continuo

- *Continuous integration* - Integrare continuamente i cambiamenti al codice eviterà ritardi più avanti nel ciclo del progetto, causati da problemi d'integrazione.
- *Refactoring* o *Design Improvement* - riscrivere il codice senza alterarne le funzionalità esterne, cambiando l'architettura, in modo da renderlo più semplice e generico.
- *Small Releases* - consegna del software avviene tramite frequenti rilasci di funzionalità che creano del valore concreto.

Comprensione condivisa

- *Coding standards* - Scegliere ed utilizzare un preciso standard di scrittura del codice. Questo rappresenta un insieme di regole concordate, che l'intero team di sviluppo accetta di rispettare nel corso del progetto.
- *Collective code ownership* - significa che ognuno è responsabile di tutto il codice; quindi contribuisce alla stesura chiunque sia coinvolto nel progetto.
- *Simple design* - i programmatori dovrebbero utilizzare un approccio del tipo "*semplice è meglio*" alla progettazione software. Progettare al minimo e con il cliente.
- *System metaphor* - descrivere il sistema con una metafora, anche per la descrizione formale. Questa può essere considerata come una storia che ognuno - clienti, programmatori, e manager - può raccontare circa il funzionamento del sistema.

Benessere dei programmatori

- *Sustainable pace* - il concetto è che i programmatori o gli sviluppatori software non dovrebbero lavorare più di 40 ore a settimana.

Modello dei processi

James Donovan Wells individua quattro linee guida:

- *comunicazione* (tutti possono parlare con tutti, persino l'ultimo dei programmatori con il cliente);

- *semplicità* (gli analisti mantengano la descrizione formale il più semplice e chiara possibile);
- *verifica* (sin dal primo giorno si prova il codice);
- *coraggio* (si dà in uso il sistema il prima possibile e si implementano i cambiamenti richiesti man mano).

Individua inoltre quattro fasi di progetto, ognuna delle quali con le sue regole interne:

- *pianificazione* (*user stories*, *release planning*, *small releases*, *project velocity*, *load factor*, *iterative development*, *iteration planning*, *move people around*, *daily stand up meeting*, *fix XP*);
- *progettazione* (*simplicity*, *system metaphor*, *never add early*, *refactoring*);
- *sviluppo* (Customer Always Available, Standards, Unit Test First, Pair Programming, Sequential Integration, Integrate Often, Collective Code Ownership, Optimize Last, No Overtime);
- *collaudo* (*unit test framework*, *bug's found*, *functional test* od *acceptance tests*).