Extreme Programming

Xp nasce intorno al 1987. Si propone come una metodologia di sviluppo lightweight di sviluppo di software:

- Centrato sul codice che viene scritto
- Dedicato a progetti di piccola o media dimensione
- Orientato alla soddisfazione del cliente

È definito come un insieme di pratiche che caratterizzano lo sviluppo del software: descrive in modo rigoroso tutte e sole le attività di processo da mettere in pratica per sviluppare software. E' un approccio alla programmazione molto appropriato all'uso per piccoli team(2-10 programmatori),progetti con alto rischio e quindi requisiti instabili, che siano improntati sulla testabilità

XP valori

- ***Communication:*** durante ogni attività è necessario favorire la comunicazione tra i membri del team di sviluppo e con il committente/utente.

  XP enfatizza il valore della comunicazione in molte delle sue pratiche:

  1. Whole team
  2. Planning game
  3. Metaphor
  4. Onsite customer
  5. Pair programming
  6. Small releases
  7. Coding standard
  8. Collective ownership
  9. Daily standup meetings

  XP impiega un coach il quale lavoro è notificare quando le persone non stanno comunicando e reintrodurli.

- ***Feedback:*** sviluppare sulla base di ipotesi e requisiti non compresi comporta numerosi problemi. Feedback immediati e frequenti permettono di capire come funziona il sistema e di soddisfare il committente.

  - Feedback a diversa scala temporale.
  - Unit test indicano lo stato del sistema ai programmatori.
  - Quando i clienti scrivono nuove storie utente, i programmatori stimano il tempo necessario per apportare le modifiche.
  - I programmatori producono nuove versioni ogni 2-3 settimane affinché i clienti possano esaminarle.
  - Feedback oriented pratiche:
    1. Whole team
    2. Planning game
    3. Onsite customer
    4. Test-driven development
    5. Continuos integration
    6. Small releases

- ***Simplicity:*** si deve lavorare sul task più semplice possibile, senza anticipare esigenze future.

  Il coach può dire "fai la cosa più semplice che potrebbe funzionare" quando vede un sviluppatore XP fare qualcosa di inutilmente complicato.

  Simplicity-oriented pratiche:

1. Metaphor
2. Simple design
3. Refactoring

- ***Courage:*** durante un progetto spesso bisogna prendere delle decisioni: XP incoraggia l'adozione di misure drastiche quando necessario.

  - Il coraggio di comunicare e accettare il feedback.
  - Il coraggio di buttare via il codice (prototipi).
  - Il coraggio di ristrutturare l'architettura del sistema.
  - Courage-oriented pratiche:
    1. Collective ownership
    2. Sustainable pace
    3. Onsite customer
    4. Refactoring

XP Pratiche

- ***Whole team:*** in XP chiunque lavori ad un progetto è parte di un team globale; ne fa parte anche il customer le cui esigenze devono essere soddisfatte. ***Quindi*** **un rappresentante del cliente deve essere sempre a disposizione.**

  Standup meeting giornalieri

  - La comunicazione tra l'intero team è lo scopo della riunione in piedi.
  - Una riunione in piedi ogni mattina viene utilizzata per comunicare problemi, soluzioni e promuovere gli obbiettivi del team.
  - Tutti si alzano in cerchio per evitare lunghe discussioni:
    i. È più efficiente tenere una breve riunione alla quale tutti sono tenuti a partecipare rispetto a molte riunioni con pochi sviluppatori ciascuna.
    ii. Tutti partecipano, incluso il cliente.
    iii. Tutti possono parlare.
    iv. Di durata massima di 15 minuti.

- ***Metaphor:*** descrivere il modo in cui il sistema funziona mediante una "storia" semplice e condivisa tra i membri del team.

  Users stories

  Il cliente contatta un gruppo di sviluppo XP per avviare un progetto. Un team XP insiste affinché il cliente sieda con il proprio team per tutto il tempo in cui si sta sviluppando. Un progetto XP in genere prevede tre fasi:
  - una fase di esplorazione, in cui il cliente scrive storie, i programmatori le stimano e il cliente sceglie quali storie saranno sviluppate;
  - una fase di iterazione, in cui il Cliente scrive test e risponde a domande, mentre i programmatori programmano
  - una fase di rilascio, in cui i programmatori installano il software e il cliente (si spera) accetta il risultato.

  Il cliente in XP ha frequenti opportunità di cambiare la direzione del team se le circostanze cambiano. Poiché i test sono così importanti, il cliente è a conoscenza del vero stato del progetto molto prima nel ciclo.

- ***Planning game:*** pianificare e guidare in modo semplice il progetto in termini di obbiettivi e tempi delle attività da svolgere.

  - Il cliente presenta un elenco delle funzionalità desiderate per il sistema.
  - Ogni funzione è scritta come una user story.
  - Gli sviluppatori stimano la quantità di sforzo necessaria per ogni storia e la quantità di sforzi che il team può produrre in un determinato intervallo di tempo (iterazione).
  - Velocità progetto = numero di giorni che possono essere assegnati a un progetto per settimana.
  - Date le stime degli sviluppatori e la velocità del progetto, il cliente dà la priorità alle storie da implementare.

- ***Simple design:*** concepire il sistema nel modo più semplice possibile per soddisfare i requisiti corretti.
  - Utilizzare sempre il design più semplice possibile per eseguire il lavoro.
  - I requisiti cambieranno domani, quindi fai solo ciò che è necessario per soddisfare i requisiti di oggi.

  - Secondo XP la fase di design è durante l'implementazione del codice non prima come nel waterfall(niente design up-front). Si preferisce anche la riduzione dell'uso dei commenti puntando direttamente sulla chiarificazione del codice stesso rendendolo auto esplicativo. La mancanza del design up-front funziona grazie alla presenza del cliente,dalle user stories, dalla grande comunicazione tra i membri del team e il refactoring continuo migliorano il codice base. L'enfasi su codice leggibile funziona grazie al pair programming che assicura codice Leggibile.

- ***Small releases:*** dare frequentemente visibilità al software prodotto.

  - Inizia con l'insieme di funzionalità utili più piccoli.
  - Rilasci prematuri e ricorrenti, aggiungendo qualche piccola funzionalità.
  - I rilasci possono essere identificati in base alla data o alla users stories.

- ***Continuous integration:*** effettuare integrazione e build del sistema più volte al giorno.
  - Gli sviluppatori dovrebbero integrare e rilasciare il codice nel repository di codici il più presto possibile. Una integrazione continua, spesso, evita sforzi di sviluppo divergenti o frammentati, in cui gli sviluppatori non comunicano tra di loro su ciò che può essere riutilizzato o ciò che potrebbe essere condiviso.
  - Tutti devono lavorare con l'ultima versione.
  - Non è necessario apportare modifiche al codice obsoleto.
  - Utilizzare gli unit test facilitano l'integrazione.

- ***Test-driven development:*** il testing va effettuato costantemente durante lo sviluppo.

  - Prova iniziale: prima di aggiungere una funzione, scrivi un test per essa.
    - ***i.*** se il codice non ha un caso di test automatizzato, si presume che non funzioni.
  - Quando la suite di test completa supera il 100%, la funzione viene accettata.
  - Quando crei il tuo test, prima di scrivere del codice, successivamente troverai molto più facile e veloce creare il tuo codice.
  - Il tempo necessario per creare uno unit test e scrivere il codice della feature in grado di passare lo unit test, è quasi lo stesso impiegato per codificare immediatamente la feature.
  - Quando viene trovato un bug, viene scritto un nuovo test unit per catturare quel bug in futuro.

Tipi di test

Uno **unit test** automatizza il test delle funzionalità mentre gli sviluppatori continuano a sviluppare. Ogni test unit in genere verifica solo una singola classe o un piccolo gruppo di classi. Gli esperimenti dimostrano che lo sviluppo guidato dai test riduce il tempo di debug. Aumenta la sicurezza che le nuove funzionalità funzionano e funzionino con tutto.

I **test di accettazione** (o test funzionali) sono specificati dal cliente per verificare che l'intero sistema funzioni come specificato. Quando tutti i test di accettazione vengo superati, la storia dell'utente viene considerata completa.

- ***Refactoring:*** ristrutturare il sistema quando necessario per migliorare il design.

  - Noi programmatori manteniamo gli stessi vecchi design del software a lungo anche quando essi sono diventati obsoleti.

    i. Continuiamo a utilizzare e riutilizzare il codice che non è più gestibile perché funziona ancora in qualche modo e abbiamo paura di modificarlo.

  - Extreme programming (XP) prende una posizione differente:
    i. Quando rimuoviamo la ridondanza, eliminiamo la funzionalità inutilizzata e ringiovaniamo i progetti obsoleti eseguiamo il refactoring.
    ii. Il refactoring durante l'intero ciclo di vita del progetto consente di risparmiare tempo e migliorare la qualità.
    iii. Puoi eseguire il refactoring con sicurezza utilizzando gli unit test per verificare le modifiche effettuate.
  - Eseguire il refactoring per mantenere un design semplice.
  - Mantenere il codice pulito e coinciso.
  - Assicura che tutto sia espresso una volta e una sola volta.

- ***Pair programming:*** tutto il codice viene scritto da due programmatori che lavorano in coppia alla stessa macchina.

  Tutto il codice da includere in una versione di produzione è creato da due persone che lavorano insieme su un singolo computer. È contro intuitivo, ma due persone che lavorano su un singolo computer aggiungeranno le stesse funzionalità di due persone che lavorano separatamente, tranne per il fatto che le funzionalità aggiunte saranno di qualità molto più elevata. Con una maggiore qualità arrivano grandi risparmi più avanti nel progetto.
  Il modo migliore per programmare in coppia è semplicemente sedersi fianco a fianco davanti al monitor. Una persona digita e pensa tatticamente al metodo che viene creato, mentre l'altra pensa strategicamente al modo in cui quel metodo si adatta alla classe.

- ***Collective code ownership:*** chiunque faccia parte del team può cambiare (migliorare) qualsiasi parte del codice quando vuole.
- ***Coding standard:*** adottare al livello di team uno standard comune (come se il codice fosse scritto da un unica entità).
- ***Sustainable pace:*** per produrre software di qualità il team deve lavorare ad un ritmo sostenibile.

  40 ore a settimana

  - Il lavoro straordinario non è la soluzione ai cattivi sforzi di sviluppo.
  - Regola XP: "Non ci sono due settimane di straordinari di fila".
  - Gli sviluppatori esausti non sono efficienti.

XP - ciclo di vita

Exploration: si scrivono le user stories, si esplorano le tecnologie disponibili, si vagliano le alternative architetturali e le rispettive possibili implementazioni.
Planning: si esegue il planning game, si concorda il più piccolo e significativo insieme di stories da completare entro una certa data.
Iteration to first release: lo schedule è frazionato in iterazioni da 1-4 settimane, ogni iterazione produce un insieme di casi di test funzionali per ciascuna delle storie schedulate in quella iterazione. I test funzionali vengono eseguiti alla fine di ciascuna iterazione.
Productionizing: si compiono iterazioni più frequenti (1 settimana) per aumentare il feedback, si certifica che il software è pronto per la produzione, si implementa un nuovo test e si mette a punto il sistema.
Maintenance: contemporaneamente, si producono nuove funzionalità, si mantiene funzionante il sistema esistente, nuovi membri si aggiungono al team. Se necessario, si opera il refactoring o si migra verso una nuova tecnologia. Si sperimentano nuove idee architetturali.
Death: il progetto cessa di esistere (es. il customer è soddisfatto e non richiede altre funzionalità, lo sviluppo non è più sostenibile economicamente, …).