40 ore a settimana

- Si lavora 8 ore al giorno
- VIsto che ci sono molte cose si riconosce il fatto che se si è stanchi non si lavora bene e quindi non si trovano errori in maniera semplice
- Se ci si accorge che per andare avanti nella scrittura del codice per le storycard bisogna lavorare
 delle ore extra (superando il tempo stimato sulla storycard) vuol dire che la stima non è stata
 corretta perchè si sono verificati degli imprevisti (cambio piattaforma di sviluppo, librerie non
 disponibili ecc..).
 - In questo caso ci si riunisce e si prendono le eventuali decisioni per affrontare i problemi sorti

Cliente in sede

Il cliente è insieme agli sviluppatori quindi si trova in sede. In questo modo può *rispondere in qualsiasi momento* alle eventuali domande dei programmatori e questo compensa la poca documentazione.

Il dialogo quindi spiega eventuali dubbi sul codice scritto

- In vari momenti durante la giornata il cliente è disponibile per chiarire dubbi ai programmatori (di qualsiasi ruolo) e, nel tempo libero, può svolgere il proprio lavoro.
- La presenza del cliente in sede, quindi, rappresenta complessivamente un vantaggio molto strategico per lo sviluppo del software.
- Stabilisce priorità e fornisce il contesto per le decisioni dei programmatori

Standard di codifica

- Quando si sviluppa in team bisogna stabilire delle convenzioni per la codifica (parentesi graffe nella stessa linea oppure subito sotto, posione dei commenti, scelta dei nomi delle calssi/metodi,
 CamelCase o snake_case, spaziatura) e questo facilita la comprensione del codice da parte dei programmatori e del cliente.
- Il team che partecipa allo sviluppo del software si mette d'accordo sugli standard da adottare durante tutto il processo di sviluppo
- Costruzioni complicate (per il design) NON sono permesse

Convenzioni linguaggio Java

Refactoring

- Refactoring vuol dire ristrutturare il codice senza cambiarne la funzionalità.
- Si ha un miglioramento della STRUTTURA del codice e quindi una maggior modularità
- Se si aggiungono molte linee di codice, allora si fa di nuovo refactoring
- I test possono essere scritti in maniera più semplice visto che i metodi conterranno poche linee di codice
- Dopo il refactoring bisogna verificare che il comportamento del codice sia identico alla versione precedente e, per questo motivo, si scrivono dei test prima di fare refactoring.
- Si punta a codice senza ripetizioni

Considerazioni di XP

- Si focalizza sul codice: alta qualità, facile da comprendere, molti test che documentano quello che si deve fare
- Si orienta sulla gente e mette le persone al primo posto e la loro collaborazione (anche perchè il cliente è in sede)
- E' leggero: si pianifica il lavoro per il breve termine e *NON si spreca tempo per il lungo termine*.

 Se si spreca del tempo per una storycard esso è un tempo breve. Rimuove anche costi aggiuntivi e crea un prodotto di qualità tramite TEST RIGOROSI
- I principi di XP non sono nuovi perchè queste pratiche erano già adottate nel passato

XP è stato descritto nel 1999 e, di seguito, sono elencati gli aggiornamenti più significativi:

- Documentazione: nel 2008 StackOverflow
- Metodologia: nel 2014 Mob Programming (ci sono molte persone che lavorano allo stesso codice e solo uno di loro scrive mentre gli altri guardano e ogni 15 minuti si scambiano i ruoli)
- Metodologia: dal 2020 circa, è possibile instaurare una sessione di Remote Pair Programming, per esempio usando Live Share di VSCode (la produttività si abbassa)
- Metodologia: dal 2023 circa, ChatGPT (asssitente di capacità enormi) -> Pair Programming (il programmatore esperto deve essere la PERSONA FISICA per EVITARE di essere rimpiazzati da ChatGPT)

Scrum

E' un processo di sviluppo (inventato nel 2010) che, se usato da solo, non risponde a molte domande/esigenze (tipo la frase "*Usiamo Scrum*"). Un'azienza che usa solo Scrum non vuol dire nulla (a differenza di "Sto usando XP e Scrum" che ha più senso).

Da una terminologia e una direzione per orientarsi nella programmazzione

XP e Scrum affrontano aspetti diverse dell'organizzazione del lavoro e **non affrontano lo stesso problema**.

- Il **PRODUCT OWNER** è il proprietario del software che dovrà riceverlo. <u>Prende scelte sul prodotto e sulle caratteristiche e i requisiti</u>. *Quindi elenca il lavoro da realizzare*. I requisiti da realizzare vengono raccolti nel **PRODUCT BACKLOG** che viene **riempito più volte**.
- SCRUM TEAM è il team di sviluppatori che adottano le tecniche Srum che prende i requisiti e
 produce il software.
- Il prodotto viene realizzato in **più iterazioni** (più passate di produzione) e ognuna di esse viene chiamata **SPRINT** che **dura 4 settimane** (in XP durava 2 settimane).
- Il team valuta il da farsi per i prossimi sprint insieme al Product Owner e insieme ad altre persone (STAKEHOLDER) che hanno interesse nello sviluppo dello stesso prodotto.

Principi di Scrum

E' semplice ma incompleto. Si basa sui seguenti termini:

• EMPIRISMO; si deve osservare la realtà e prendere scelte basate su quello che avviene nella realtà. Nelle aziende non sempre si ha la capacità di osservare la realtà (per esempio nella stima dei tempi di consegna del prodotto -> si stima un tempo brevissimo ma in realtà ne serve molto di più)

 LEAN: letteralmente vuol dire snello/agile e significa ridurre lo spreco e concentrarsi sull'essenziale (si produce solo quello che serve subito).

Nella guida di Scrum non vi è la parola agile perchè compare lean

Pilastri di Scrum

Scrum si sostiene sui seguenti pilastri (devo avere la prima per la seconda e devo avere la seconda per avere la terza: trasparenza -> ispezione -> adattamento):

- **TRASPARENZA**: Tutto il codice prodotto derivato dal Product Backlog (che *può venire* prodotto in *qualsiasi modo*, in genere con le storycard) deve essere visibile a tutti e di facile consultazione. Se tale pilastro è soddisfatto, allora posso avere il successivo.
- **ISPEZIONE**: Si vanno a cercare i problemi molto frequenteente per scoprire potenziali problemi. Scrum fornisce la cadenza delle ispezioni tramite 5 **eventi**.
- **ADATTAMENTO**: Si vanno a cercare le soluzione per un determinato problema ispezionato. L'adattamento si applica subito per evitare ulteriori divergenze dei problemi



Il successo nell'uso di Scrum dipende da 5 valori :

- Commitment: il team si impegna a raggiungere gli obiettivi e a supportarsi a vicenda
- Focus: concentrazione sul lavoro da fare senza farsi distrarre da nuovi avvenimenti durante lo sviluppo
- Openness: il team può cambiare direzione durante lo sviluppo e il lavoro può essere riprogrammato in base alle nuove idee che vengono fuori
- Respect: ogni componente del team rispetta gli altri (modo di procedere, punto di vista)
- Courage: il team prende con coraggio delle decisioni importanti di comune accordo. Il team lavora su problemi difficili

Eventi di Scrum

Ci sono degli eventi importanti:

- Sprint: iterazioni del lavoro. SI ha un Product Backlog e da esso si selezionano le cose da fare subito e si fa il primo sprint.
- Sprint Planning: Prima dello sprint si fa una pianificazione del lavoro (8 ore al massimo) per un mese (4 settimane). Partecipano tutti i membri del team e il Product Owner
- Daily Scrum: giornalmente c'è un incontro che dura 15 minuti e si fa all'inizio della giornata. Serve per scambiarsi informazioni tecniche sul lavoro svolto e il lavoro da svolgere. Si programma un po' il da farsi durante la giornata. Si affronta in piedi per avere la sensazione che il tempo stia scorrendo e quindi per sbrigarsi (se ci si siede si ha le sensazione di poter perdere molto più tempo).
 - SPRINT BACKLOG -> si estraggono, dal Product Backlog, i requisiti da portare a termine nel singolo Sprint. Lo Sprint Backlog include anche attività che non sono relative per ottenere nuove funzionalità e quindi sono solamente attività da svolgere
- Sprint Review: si fa dopo la conclusione dello sviluppo delle 4 settimane e dura al massimo 4 ore.
 Serve per capire se si sono raggiunti i risultati posti come obiettivi all'inizio dello Sprint. Si aggiungono eventuali cose da fare al Product Backlog

- Sprint Retrospective: si ragiona su come si è operato per il singolo sprint e si va a valutare l'organizzazione adottata per tentare di migliorarla. (si sono incontrate difficoltà nell'utilizzo di una libreria)
 - **Definition of Done**: si deve capire bene e **DEFINIRE** quando si reputa che **effettivamente si è finito il lavoro** (è *PERSONALE* e può essere: "ho finito quando ho completato i test", "ho completato i test senza errori", "ho consegnato il codice al cliente")

Produet Backles =

Artefatti di Scrum →

4 lose prodotte

Ci sono varie cose che possono essere prodotte (Artefatti):

- Product Backlog: Lista dei requisiti del prodotto che servono per migliorarlo. E' la sorgente del lavoro del team
- Product Goal: obiettivo e si trova nel backlog e descrive lo stato futuro del prodotto
- Sprint Backlog: obiettivo complessivo del singolo sprint.
 - Sprint Goal è l'obiettivo dello Sprint
- Increment è qualcosa che porta verso il Goal. Quando il Product Backlog soddisfa la Definition of Done allora si ha un nuovo incremento
- CScrum Master: è una figura che fa da allenatore per il team di Scrum che ha più esperienza con la tecnica di Scrum e quindi dà consigli su come attuare questi principi e di come rendere più snello il lavoro da fare.

Refactoring

- Refactoring è il processo che cambia un sistema software in modo che il comportamento esterno
 del sistema non cambi, ovvero i requisiti funzionali soddisfatti sono gli stessi, per far sì che la
 struttura interna sia migliorata
- Si migliora la progettazione a poco a poco, durante e dopo l'implementazione del codice

Esempio semplice: consolidare (ovvero eliminare) frammenti di codice duplicati all'interno di rami condizionali

Vantaggi del refactoring:

- Spesso la dimensione del codice si riduce
- Si comprende meglio il codice
- Le strutture complicate si trasformano in strutture pipiù semplici da capire e mantenere
- Si evita di introdurre un debito tecnico all'interno della progettazione

Tecnica Estrai metodo

```
public void printDovuto(double amount) {
   printBanner();
   System.out.println("nome: " + nome);
   System.out.println("tot: " + somma);
}

• Diventa
public void printConto(double somma) {
   printBanner();
   printDettagli(somma);
}

public void printDettagli(double amount) {
   System.out.println("nome: " + nome);
   System.out.println("tot: " + somma);
}
```

A livello di progettazione:

- printDovuto() si serve di un metodo di più basso livello perchè lo richiama (printBanner()) e implementa istruzioni di basso livello
- Si rende di livello più alto se si trasportano le System.out in un metodo apposito e il metodo printConto() fa due macro-attività.
- printConto() ha un livello di astrazione più preciso perchè prima faceva sia cose elementari che non elementari.
- Adesso, se voglio stampare di nuovo i dettagli posso solo richiamare la funzione printDettagli piuttosto che chiamare printDovuto()

Motivazioni Estrai Metodo

Si applica:

- Quando si hanno metodi lunghi (più di 15 linee)
- Quando il codice non è comprensibilie
- Quando i metodi si hanno commenti all'interno del codice (avere commenti è sintomo di refactoring).
 Il codice sotto il commento diventa una nuova funzione e si applica la tecnica Estrai Metodo. Ogni commento da anche il nome del futuro nuovo metodo che viene inventato da applicare
- Per avere metodi piccoli: in caso di gerarchia di classi si fa facilmente OVERRIDE.

Meccanismi

Come applicare la tecnica Estrai Metodo: (**ESAME**):

- Creare un nuovo metodo il cui nome comunica l'intenzione del ù
- Copiare il codice estratto dal metodo sorgente al nuovo
- Guardare se il codice estratto ha variabili locali al metodo sorgente e far diventare tali variabili parametri del metodo
- Se alcune variabili sono usate solo all'interno del codice estratto farle diventare variabili temporanee
 del puovo
- Se una variabile locale al metodo sorgente è modificata dal codice estratto, vedere se è possibile far sì che il metodo estratto sia una query e assegnare il risultato alla variabile locale del metodo
- Sostituire nel metodo sorgente il codice estratto con una chiamata al metodo nuovo

Se il metodo estratto deve ritornare più cose ci sono diverse alternative:

- uno di questi valori viene tornato e gli altri vengono assegnati ad attributi
- più valori di ritorno li metto negli attributi e il valore di ritorno diventa void
- si crea un tipo apposito che contiene tutti i valori di ritorno (coppia <String, Integer> e simili)

```
public class Ordine {
    private double importo;
    public double getImporto() {
        return importo;
    }
}

public class Stampe {
    private ArrayList=Ordine> ordini;
    private String nome = "Mike";

public void printDovuto() {
        private String nome = "Mike";

    public void printDovuto() {
        Iterator=Ordine> i = ordini.iterator();
        double tot = getTotale();
        printDettagli(tot);

        // scrivi banner
        System.out.println(""-cliente Dare -");
        System.out.println(""-cliente Dare -");
        System.out.println(""-cliente Dare -");
        System.out.println(""-cliente Dare -");
        }

        // calcola totale
    while (i.hasNext()) {
            Ordine o = i.next();
            tot += o.getImporto();
        }

        // scrivi dettagli
        System.out.println("-cliente Dare -");
        Syst
```