Scrum

Introduzione

Obiettivi:

- Comprendere come utilizzare e adottare il framework Scrum

Scrum rappresenta un insieme di paradigmi utilizzati all'interno di team di sviluppo software, dove l'intento consiste nella creazione di sistemi che abbiano il più alto valore possibile. Un framework simile è adottato in differenti contesti, ma tramite l'ingegneria del software trova il suo massimo grado di espressione; di seguito è riportata una breve definizione che possa raffigurare l'importanza di uno strumento del genere.

Definizione informale
Scrum è un framework con cui membri di team di sviluppo possono affrontare problemi complessi in maniera adattiva, approciando ad un comportamento più creativo e produttivo possibile.

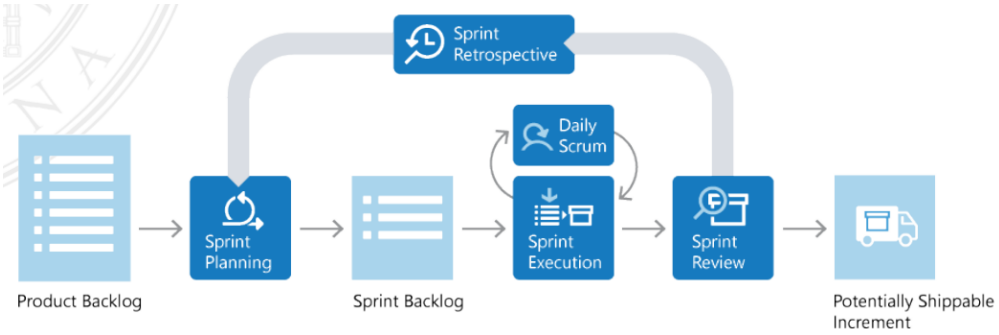
Il framework analizzato è spesso associato ad ulteriori metodiche implementative e di stesura del codice, poichè tipicamente il suo utilizzo è attuato per garantire una sequenzialità di pratiche gestionali di sistemi software, per cui adottato in concomitanza di strumenti maggiormente tecnici. E' fondamentale comprendere che *Scrum* non sia un metodo di sviluppo, ma si concentra solamente sulla pura progettazione della soluzione software dinnanzi ai requisiti funzionali richiesti, per questo spesso è adeguato con altri idiomi di sviluppo del mondo *Agile*, semplificando la realizzazione del prodotto finale.

Ogni progetto che adegua *Scrum* è caratterizzato da iterazioni denominate *sprint*; si riporta una definizione che possa descrivere al meglio l'entità fondamentale su cui si stabilisce l'intero meccanismo.

Definizione informale
Uno sprint è un'iterazione relegata da una durata predefinita, la quale non può più essere modificata qualora inizializzato il processo esecutivo. Ognuno di essi è preceduto da una riunione organizzativa, in cui si discutono gli obiettivi da portare a termine alla fine del vincolo temporale. In conclusione è ottenuto un prototipo incrementale, oppure meglio definito come *potentially shippable increment*, rappresentante un termine dell'insieme, poichè non dispone di tutte le funzionalità richieste, costituendo potenzialmente una parte della soluzione.

Scrum lifecycle

In questa sezione è riportata una raffigurazione dell'approccio organizzativo garantito da *Scrum*, in cui sono presenti differenti elementi che ne contraddistinguono la natura, i quali di seguito saranno meglio approfonditi.



Rispetto alla raffigurazione introdotta, sono stabiliti due insiemi che raggrappano gli elementi illustrati, denominati corrispettivamente *artefatti* ed *eventi*. Di seguito è anticipato l'insieme degli *artefatti*.

- *Product backlog* consiste in una lista di *things*, tipicamente ordinata in base all'esigenze del *product owner*, che devono essere riportate all'interno della soluzione software e conseguite, anch'esse suddivise in cinque categorie:
  - *Requisiti funzionali*, ossia i punti salienti individuati dal team di sviluppo durante le prime fasi di approccio al sistema software, tendenzialemente descritti come *user stories*
  - *Bug fixes*, illustrano i *difetti* di cui soffre la soluzione software, si ricorda che un *bug* oppure un *defect* indica un errore logico posto all'interno dell'architettura, il quale non sempre coincide con evidenti *malfunzionamenti*, per cui è necessario adeguare parallelamente allo sviluppo, azioni mirate al *testing* delle funzionalità introdotte
  - *Requisiti non funzionali*, interpretabili come un sottoinsieme dei requisiti funzionali ma non fondamentali al conseguimento della soluzione software. In questo ambito sviluppatori devono essere coscienti del *design smell needless repetition*, ciò implica che sviluppatori aggiungano layer di funzionalità ulteriori rispetto alle reali pretese degli utenti finali, riportando un erroneo trade-off tra spesa sostenuta e introito ricevuto
  - *Requisiti tecnologici*, nuovamente interpretabili come un sottoinsieme derivato dei requisiti funzionali, i quali sono stabiliti come conseguenza della traduzione delle richieste originarie rispetto ad un linguaggio di programmazione che dovrà essere attuato, per cui in relazione anche alle tecniche adoperate pur di conseguire nell'intento
  - *Chores*, attività utili per il team di sviluppo, soprattutto in relazione alla decisione di quali strumenti utilizzare, ad esempio realizzare un database relazionale oppure una base di dati No-SQL, azioni che non producono alcun valore aggiunto ma che possono aumentare la produttività interna del team
- *Sprint backlog* stabilisce un'evoluzione dell'entità che lo procedono, in cui non sono più trattate *user stories*. La *lista antecedente* diviene un insieme di *tasks*, determinate dai membri del team di sviluppo. Le mansioni devono essere concluse entro un certo *sprint*, per cui ad ognuna di esse è associato un valore, espresso in ore richieste per completare lo specifico compito, obbligatoriamente minore di un giorno lavorativo; qualora *task* siano associate ad un valore maggiore occorre suddividerla in due differenti mansioni

Definita la categoria di *artefatti*, successivamente sono descritti gli *eventi* che caratterizzano *Scrum*.

- *Sprint planning* è un evento in cui il *product owner* definisce un obiettivo e presenta, in relazione al team di sviluppo e all'intento che voglia essere conseguito, quali temi siano caratterizzati da una maggiore importanza all'interno del *calderone*, ossia il *product backlog* di riferimento, ed ad ognuno di essi è associato il valore temporale necessario, previa precedente discussione, per compiere la *user story*, divenendo un *task*, affinchè sia popolato *sprint backlog*
- *Daily scrum* raffigura un breve incontro tra i membri del team di sviluppo, tendenzialmente di breve durata, in cui si discutono e si analizzano velocemente i progressi implementativi attuati fino ad un determinato momento. Ciò accade giornalmente, indipendentemente dalla difficoltà delle *tasks* e dallo stato di avanzamento della soluzione, potrebbe inoltre rappresentare un'occasione utile in cui si manifestano perplessità e problematiche inerenti allo sviluppo
- *Sprint review e Sprint retrospecting*, al termine di uno *sprint* sono attuate due differenti operazioni, suddivise in:
  - *Review*, consiste nell'analisi del prototipo software realizzato, rispetto al *goal* definito dal *product owner*, affinchè sia possibile osservare se risulti essere idoneo alla prospettiva iniziale, o più semplicemente se in grado di produrre il comportamento atteso. Partecipano tutti i membri del *team di sviluppo* e i *portatori di interesse*, in maniera tale che sia imbastita una discussione del prodotto realizzato, ottenendo il livello di compatibilità con l'aspettative degli utenti finali
  - *Retrospecting*, consiste in un'*analisi interna*, in cui sono coinvolti il *Scrum Master* e ogni membro del *team di sviluppo*, dove il tema centrale consiste nell'analisi dell'approccio adeguato in relazione all'obiettivo originario, permettendo di descrivere quali entità siano conformi all'ambiente di sviluppo e quali dovrebbero subire delle modifiche in modo tale che risultino affini al gruppo software

Scrum Roles

Trattandosi di un idioma sviluppato per favorire e semplificare la complessità progettuale, al suo interno sono stabiliti una serie di ruoli conformi all'intento di *Scrum*, in cui si contraddistinguono:

- *Product Owner*, figura indispendabile all'interno di tematiche relative all'implementazione di sistemi software, poichè rappresenta il *costumer*, ossia colui che richiede a team di sviluppo di adeguare le proprie richieste ad un meccanismo automatizzato. Trattandosi di una figura esterna all'effettività descritta, si occupa sostanzialmente di definire le *priorità*, le *scadenze* ed infine le *funzionalità* che la soluzione debba possedere.
- *Scrum Master*, il responsabile della corretta applicazione dei principi e pratiche di strumenti *agile*, promuove attività di supporto ai membri del team in difficoltà coordinando ed assegnando differenti metriche di implementazione, inoltre è garante del totale rispetto dei limiti temporali imposti dagli *artefatti* ed *eventi*, tipicamente organizzando riunioni e promuovendo la discussione tra sviluppatori affinchè siano condivise perplessità e problematiche, favorendo in tal modo il trasferimento di conoscenza

- *Development Team*, team di sviluppo composto da un limite minimo di cinque membri fino ad un totale di nove, qualora siano sviluppato progetti di piccola o media dimensione; la scelta di una cardinalità simile avviene per garantire scalabilità informativa, dato che il gruppo è responsabilizzato per la realizzazione del prodotto finale e può contare solamente su capacità e conoscenze interne pur di proseguire corettamente con l'implementazione