Presentazione Metodologie di sviluppo del software

Competenza | Qualità | Competitività

Ingegneria Del Software

Glossario dell'IEEE («IEEE Standard Glossary of Software Engineering»):

- **Ingegneria del software**: «applicazione di un approccio sistematico, disciplinato e quantificabile allo sviluppo, all'operativita` e alla manutenzione del software»

- **Software**: «i programmi, le procedure, e l'eventuale documentazione associata e i dati relativi all'operativita` di un sistema di elaborazione»

L'ingegneria del software è quindi la disciplina tecnologica e gestionale che riguarda la produzione sistematica e la manutenzione dei software che vengono sviluppati e modificati **entro i tempi e i costi preventivati** ovvero è una disciplina che studia i metodi di produzione, le teorie alla base dei metodi, e gli strumenti di sviluppo e misura della qualità dei sistemi software.

E' importante sottolineare come l'ingegneria del software è anche una disciplina di tipo «empirico» ovvero basata sull'esperienza e sulla storia dei progetti realizzati.

Principali Caratteristiche Del Software

- ✓ **Manutenibilità (Maintainability):** Deve essere possibile intervenire sul software nel breve tempo sia al fine di soddisfare nuovi requisiti sia al fine di riportarlo in uno stato tale da garantire la continuità del servizio per il quale è stato prodotto in caso di problemi.

- ✓ **Affidabilità (Reliability):** Il Software deve dare garanzia di funzionamento continuativa e duratura nel tempo con l'obiettivo di NON produrre danni fisici o economici (sia in caso di guasto e/o malfunzionamenti del software stesso che dell'ambiente in cui esso viene processato)

- ✓ **Efficienza (Efficiency)**: Il software deve fare un uso ottimizzato ed efficiente delle risorse di sistema quali Supporti (ottici, magnetici etc..), CPU, RAM, etc.. in quanto un uso indiscriminato potrebbe non solo penalizzare il software stesso ma anche influire, ad esempio, su altri sistemi software che condividono le medesime risorse con ovvi e inevitabili danni (anche economici).

- ✓ **Usabilità (Usability)**: Il software deve presentare un'interfaccia che sia facilmente usufruibile e comprensibile per l'utilizzatore (c.d. interfacce «amichevoli»); inoltre le funzioni esposte nel software devono prevedere una documentazione appropriata, chiara e sintetica che ne descriva il corretto uso e la relativa operabilità.

- ✓ **Sicurezza (Security):** Il software deve garantire la sicurezza del dato sia in termini di «protezione/resistenza» ad attacchi o minacce informatiche sia in termini di politiche di visibilità delle informazioni in funzione dei profili utenti.

Ciclo Di Vita Del Software

Glossario dell'IEEE («IEEE Standard Glossary of Software Engineering"):

- **Ciclo di vita del software**: «framework contenente i processi, le attività e i compiti nello sviluppo, gestione e manutenzione di un prodotto software e che attraversa la vita del sistema dalla definizione dei suoi requisiti alla cessazione del suo uso»

Un «Processo Software», definito «ciclo di vita del software» (termine mutuato dalla biologia), è nient'altro che un insieme organizzato di attività che sovrintendono alla costruzione dei software da parte del team (o dei team) di sviluppo utlizzando metodi, tecniche, metodologie e strumenti e organizato secondo un predeterminato schema di riferimento (cha a sua volta può essere informale, semi-formale o formale).

Tipicamente i differenti processi software si distinguono in base all'organizzazione delle varie fasi di sviluppo; quest'ultime possono essere racchiuse sommariamente in:

- ✓ **Raccolta Specifiche** (Software Specification)
- ✓ **Design e Sviluppo** (Software Design and Implementation)
- ✓ **Validazione** (Software Validation And Acceptance Criteria)
- ✓ **Evoluzione** (Software Evolution: Corrective)

Modelli di Processo Software

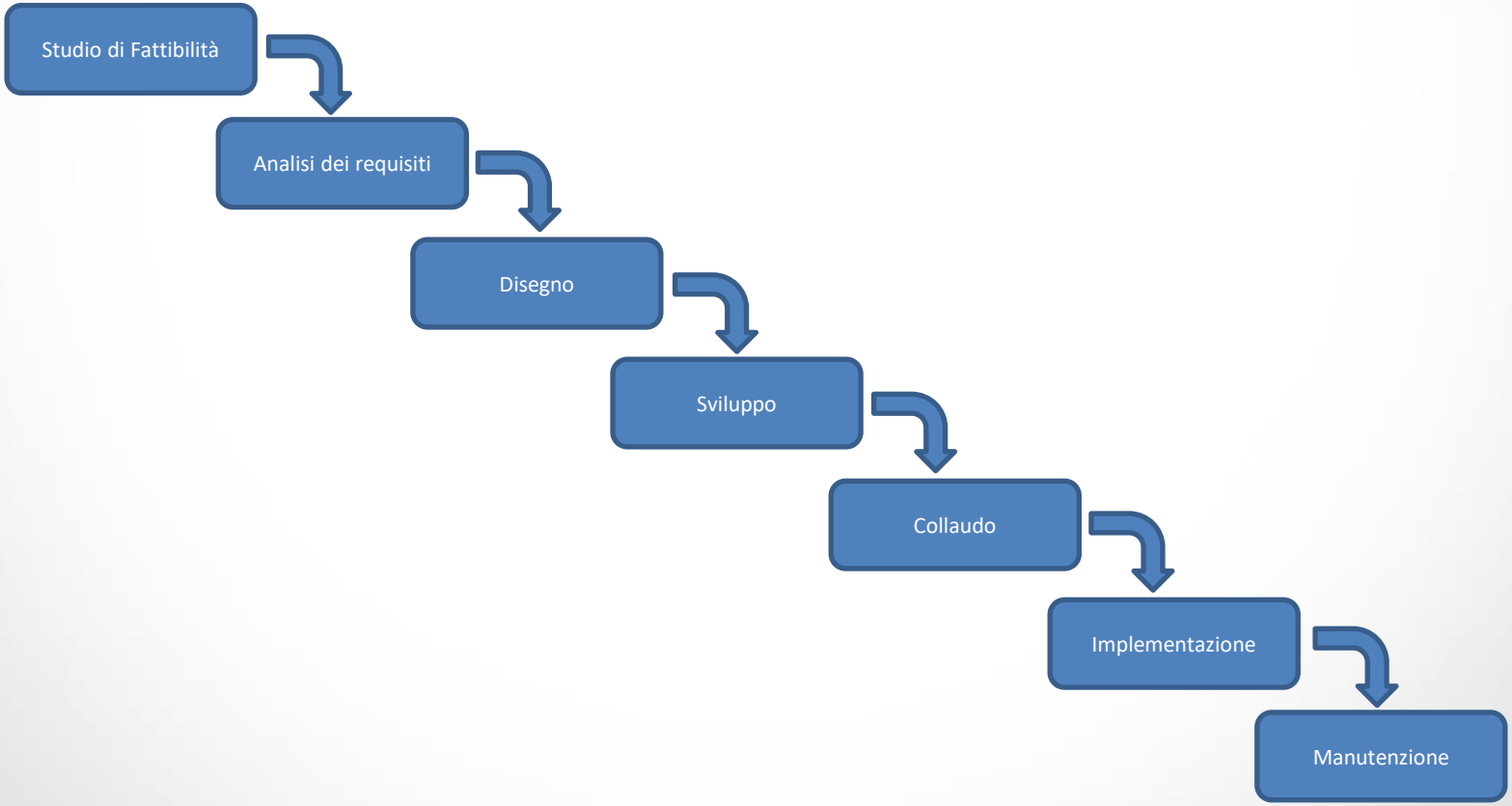
Un modello software fornisce una struttura, uno schema coerente per descrivere una caratteristica importante del processo di sviluppo; I modelli possono inoltre essere definiti a qualsiasi livello di astrazione (intero processo o singola attività strutturale).

Non esiste un modello che sia migliore di altri: un team di sviluppo può combinare tali modelli per realizzare un processo di sviluppo che risponda meglio alle esigenze del progetto; un modello non assicura che il prodotto sviluppato risponda a tutti i requisiti richiesti e che venga consegnato nei tempi previsti in particolare **ogni processo software va valutato in funzione del contesto**;

- ✓ **Modelli a Cascata**: Specifiche e sviluppo sono separati e distinti; si caratterizza per l'esecuzione in sequenza delle fasi di analisi, progetto, sviluppo, collaudo e manutenzione;
- ✓ **Modelli Iterativi**: Specifiche e sviluppo sono ciclici e sovrapposti; rispetto al modello a Cascata si introducono i cicli e si rafforzano le interazioni; si definiscono iterativi tutti i modelli di sviluppo che prevedono processi in grado di ritornare ad una fase del procedimento già affrontata in precedenza
- ✓ **Modelli Incrementali**: Definiti anche evolutivi, rappresentano un'ulteriore **strutturazione** dei modelli iterativi e sono particolarmente indicati quando si fa ricorso a prototipizzazione e sviluppo incrementale
- ✓ **Modello a Spirale**: è una ulteriore evoluzione del modello incrementale.
- ✓ **Modelli RAD e Agile**: Non sono pianificati e guidati dall'utente ma dai test focalizzati alla rapidità dei risultati e allo sviluppo in parallelo dei processi

Modello a Cascata

E’ un modello più tradizionale che prevede una sequenza di fasi, ciascuna delle quali produce un preciso output che viene utilizzato come input per la fase successiva (da cui la metafora della cascata); pur essendo stato oggetto di profonde critiche e revisioni, resta il metodo di riferimento universalmente accettato, rispetto al quale tutti gli altri risultano delle varianti piuttosto che delle vere alternative.



Modello a Cascata – Valutazioni

Il modello è tutto orientato alla data di rilascio; per questo motivo le correzioni di eventuali errori commessi nelle fasi iniziali (requisiti errati o inadeguati) potranno essere implementate solo nella fase di manutenzione facendo seguire alla prima consegna un successivo progetto che recepisca tutte le correzioni maturate nel frattempo.

La segnalazione degli errori durante le fasi di test e la loro correzione devono svolgersi nella giusta sequenza, secondo un processo lineare ma ciò si traduce ovviamente in un allungamento dei tempi.

Ogni fase viene congelata quando si passa alla fase successiva; questo approccio non consente ripensamenti, se non su base eccezionale e sopportandone il costo, sia in termini di tempo che di budget. Ad esempio non è possibile un'interazione tra clienti e sviluppatori durante la fase di sviluppo, perché i requisiti sono ormai definiti e non possono essere più modificati (blindatura dei requisiti).

Nel corso degli anni il modello ha subito delle evoluzioni che si sono tradotte in approcci diversificati alla metodologia da adottare al fine di migliorare gli aspetti negativi del modello (es. Modello a V, Sawtooth etc..)

Modello a Cascata – Valutazioni (segue)

Vantaggi:

- ✓ **Metodologia:** Facile Comprensione e applicabilità
- ✓ **Requisiti:** Essendo un modello rigoroso, si applica bene in qualunque contesto in cui i requisiti sino stabili e chiari sin dall'inizio

Svantaggi:

- ✓ **Stime:** Difficoltà a stimare in modo accurato i costi e le risorse necessarie nella fase iniziale del progetto, quando ancora mancano sufficienti elementi di dettaglio ma è già necessario definire budget e piano di lavoro.
- ✓ **Standard:** Necessità di aderire a precisi standard nella produzione dei documenti di progetto, con il rischio di introdurre una eccessiva burocratizzazione delle attività.
- ✓ **Cliente e Documentazione:** L'interazione con il cliente avviene solo all'inizio e alla fine del ciclo; Il documento che specifica i requisiti non sempre soddisfa le effettive esigenze degli utenti perché spesso gli utenti stessi non sono in grado di conoscere e quindi di descrivere con efficacia tutti i requisiti dell'applicazione («l'utente sa quello che vuole solo quando lo vede»). Considerando che la documentazione vincola il prodotto da sviluppare, la sua rigidità sin all'inizio del processo rappresenta un limite piuttosto significativo per la qualità del prodotto finale.

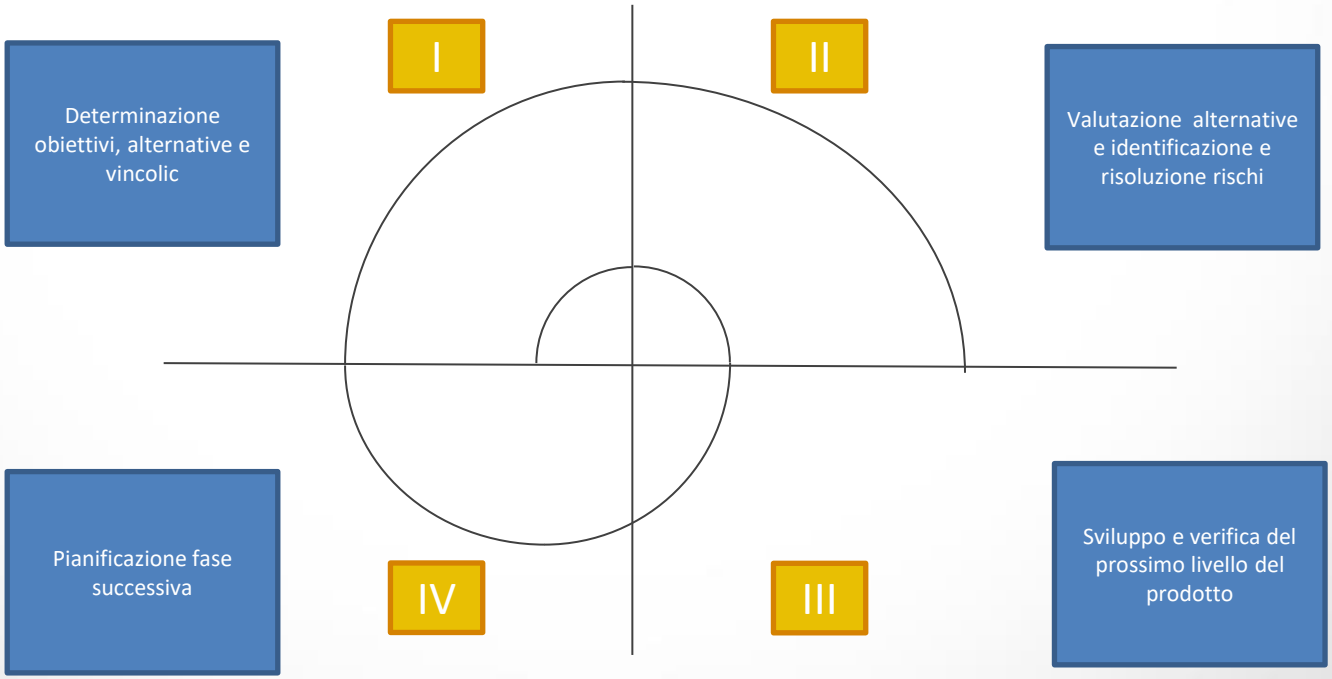
In conclusione, il modello a cascata **favorisce un incremento complessivo dei costi di produzione** del software a causa dei numerosi interventi successivi che si rendono necessari per adeguare le funzionalità del sistema alle effettive esigenze degli utenti, che le specifiche di progetto non sono state in grado di intercettare a causa della rigidità con cui il modello le gestisce. Così, quello che si è evitato di spendere durante il progetto accettando una variazione dei requisiti lo si ritrova, ad un costo superiore, come integrazione da sviluppare nel periodo successivo all'implementazione.

Modello a Spirale

Il processo è visto come una spirale dove ogni ciclo viene diviso in quattro fasi:

- ✓ **Determinazione degli obiettivi** della fase
- ✓ **Identificazione e riduzione dei rischi**, valutazione delle alternative
- ✓ **Sviluppo e verifica** della fase
- ✓ **Pianificazione** della fase successiva

Proposto per supportare l' analisi dei rischi è un modello di tipo ciclico, dove il raggio della spirale rappresenta il costo accumulato durante lo svolgimento



Modello a Spirale – Valutazioni

Ogni ciclo della spirale rappresenta una fase del processo di sviluppo ove il più interno può essere lo studio di fattibilità, il successivo la definizione dei requisiti, il successivo ancora la progettazione, etc.

Non sono definite fasi a priori e man mano che si cicla nella spirale, i costi aumentano.

Ogni ciclo della spirale passa attraverso i quadranti del piano che rappresentano i seguenti passi logici:

I. Determinazione di obiettivi, alternative e vincoli
II. Valutazione di alternative, identificazione e risoluzione di rischi (ad esempio sviluppo di un prototipo per validare i requisiti
III. Sviluppo e verifica del prossimo livello del prodotto.
    - modello evolutivo, se i rischi riguardanti l'interfaccia utente sono dominanti;
    - modello cascata se è l'integrabilità del sistema il rischio maggiore;
    - modello trasformazionale, se la sicurezza è più importante
IV. Pianificazione della fase successiva;
    - si decide se continuare con un altro ciclo della spirale

Una caratteristica importante di questo modello è il fatto che i rischi vengono presi seriamente in considerazione e che ogni fine ciclo produce una deliverables (release funzionanti alle quali mancano alcune funzionalità).

In un certo senso può essere visto come un modello a cascata iterato più volte.

Modello a Spirale – Valutazioni (segue)

Vantaggi:

- ✓ **Rischi:** Rende esplicita la gestione dei rischi
- ✓ **Riuso:** Focalizza l'attenzione sul riuso
- ✓ **Errori:** Determina errori nelle fasi iniziali (consentendone quindi la gestione e risoluzione in corso d'opera)
- ✓ **Qualità:** Aiuta a considerare gli aspetti della qualità
- ✓ **Integrazione:** Integra sviluppo e Manutenzione

Svantaggi:

- ✓ **Rischi:** Richiede persone in grado di valutare i rischi
- ✓ **Costi:**
  - ✓ Richiede un aumento nei tempi di sviluppo
  - ✓ Richiede un aumento delle persone con capacità di identificare i rischi
  - ✓ Una gestione maggiore del team di sviluppo (a sua volta un costo maggiore)
- ✓ **Cliente:** Può creare problemi nella definizione del contratto col cliente;
- ✓ **Adattamento:** La sua adozione comporta un suo stesso adattamento alla realtà aziendale e/o al team.

In conclusione è compito di chi gestisce (il manager) di minimizzare i rischi (personale non adeguato, scheduling, budget non realistico, sviluppo del sistema sbagliato etc..); ma è necessario comunque considerare che Il rischio è insito in tutte le attività umane ed è una misura dell'incertezza sul risultato dell'attività: alti rischi provocano ritardi e costi imprevisti.

Il rischio è collegato alla quantità e qualità delle informazioni disponibili: meno informazione si ha, più alti sono i rischi

Modello Agile – Introduzione

Le metodologie «**Agile**» si rifanno ai principi definiti dalla cosiddetta «**Agile Alliance**» un gruppo di sviluppatori che nel 2001 ha pubblicato «**Agile Manifesto**», un insieme di valori, principi e pratiche per la modellazione del software secondo criteri di maggior flessibilità rispetto ai metodi di sviluppo tradizionali.

1. La nostra massima priorità è soddisfare il cliente rilasciando software di valore, fin da subito e in maniera continua.
2. Accogliamo i cambiamenti nei requisiti, anche a stadi avanzati dello sviluppo.
3. I processi agili sfruttano il cambiamento a favore del vantaggio competitivo del cliente.
4. Consegniamo frequentemente software funzionante, con cadenza variabile da un paio di settimane a un paio di mesi, preferendo i periodi brevi.
5. Committenti e sviluppatori devono lavorare insieme quotidianamente per tutta la durata del progetto.
6. Fondiamo i progetti su individui motivati. Diamo loro l'ambiente e il supporto di cui hanno bisogno e confidiamo nella loro capacità di portare il lavoro a termine.
7. Una conversazione faccia a faccia è il modo più efficiente e più efficace per comunicare con il team ed all'interno del team.
8. Il software funzionante è il principale metro di misura di progresso. I processi agili promuovono uno sviluppo sostenibile. Gli sponsor, gli sviluppatori e gli utenti dovrebbero essere in grado di mantenere indefinitamente un ritmo costante.
9. La continua attenzione all'eccellenza tecnica e alla buona progettazione esaltano l'agilità.
10. La semplicità - l'arte di massimizzare la quantità di lavoro non svolto - è essenziale.
11. Le architetture, i requisiti e la progettazione migliori emergono da team che si auto-organizzano.
12. A intervalli regolari il team riflette su come diventare più efficace, dopodiché regola e adatta il proprio comportamento di conseguenza.

Modello Agile – Manifesto

Stiamo scoprendo modi migliori di creare software,
sviluppandolo e aiutando gli altri a fare lo stesso.
Grazie a questa attività siamo arrivati a considerare importanti:

Gli individui e le interazioni più che i processi e gli strumenti
Il software funzionante più che la documentazione esaustiva
La collaborazione col cliente più che la negoziazione dei contratti
Rispondere al cambiamento più che seguire un piano

Ovvero, fermo restando il valore delle voci a destra,
consideriamo più importanti le voci a sinistra

Modello Agile – Metodi

Di seguito vengono riportati una serie di framework inerenti alle metodologie Agile:

- ✓ Scrum
- ✓ XP (Extreme Programming)
- ✓ Agile Unified Process (Derivato dal RUP)
- ✓ Feature Driven Development
- ✓ Dynamic System Development Process

Modello Agile – Valutazioni

I potenziali benefici dell'approccio Agile sono:

- ✓ Maggiore attenzione ai risultati di business – l'utente è molto più coinvolto.
- ✓ Maggiore time to market – l'approccio iterativo accelera lo sviluppo.
- ✓ Maggiore efficienza organizzativa e team con morale più alto - motivazione.
- ✓ Maggiore produttività e costi più bassi - cooperazione esperto e utente.
- ✓ Potenziale qualità più alta - il test è parte integrante dello sviluppo.

Difficoltà dell'approccio Agile:

- ✓ Non è facile creare un ambiente realmente collaborativo.
- ✓ E' necessario un forte coinvolgimento del management dell'organizzazione (collaborazione dello Sponsor, cambiamento culturale, adeguata formazione dei partecipanti).
- ✓ Il tipo di business può presentare dei vincoli (leggi particolari, controlli specifici).
- ✓ Bisogna ridisegnare l'approccio al project management (disegno della propria metodologia, con un approccio più sofisticato per soddisfare le esigenze del business).

Non esiste un modo standard di avviare lo sviluppo «Agile». A volte possono sorgere ostacoli insormontabili, per cui si deve ricorrere a formalità tipiche del project management tradizionale

Scelta del modello di processo

Di seguito le linee guida che in generale vengono suggerite per la determinazione del modello da adottare.

- ✓ Per sistemi o sottosistemi di cui si hanno requisiti stabili e competenze forti sulle tecnologie da impiegare si può adottare il modello a cascata
- ✓ Sistemi critici per le persone o cose (safety critical) con requisiti stabili, possono essere sviluppati con modelli a cascata.
- ✓ Sistemi con requisiti altamente instabili possono utilizzare modelli evolutivi.
- ✓ Sistemi di grosse dimensioni, non critici, possono essere sviluppati con modelli incrementali

Ma è sufficiente ciò per comprendere quale strada sia giusto perseguire?

Scelta del modello di proceso (segue)

Circa lo sviluppo software, tra le due tesi estreme Waterfall e Agile esistono infinite soluzioni intermedie molto più aderenti alla realtà dei clienti. E' quindi necessario chiarire i vantaggi e gli svantaggi delle due soluzioni, anche se:

- I sostenitori di «Agile» credono che essere «Agile» significhi adottare metodologie come Scrum o Extreme Programming e che non ci sia affatto bisogno di altri processi di project management.
- I sostenitori dello sviluppo waterfall (a cascata) credono che l‟unico modo di gestire i progetti di sviluppo software sia il metodo tradizionale con un piano dove tutto viene previsto a priori.

Si contrappone l'agilità al controllo, la creatività alla prevedibilità, mentre in realtà le aziende hanno bisogno sia di processi di project management tradizionali sia di principi delle filosofie «Agile».

In sostanza, occorre essere flessibili e prendere il meglio dalle due discipline.

In queste aree di conoscenza non si finisce mai di imparare; bisogna seguire l'evoluzione della disciplina imparando ad agire in tutti gli ambienti attraverso il tradizionale approccio basato sulla pianificazione preventiva e sull'approccio più flessibile del project management adattivo.

E' necessario comprendere che alcune soluzioni rispondono meglio in determinati ambienti e non in altri.

«il segreto del successo non sarà la bontà della soluzione, ma semplicemente la capacità di comprendere le caratteristiche dell'ambiente» (fonte esterna)

Scrum (Estratto della guida ufficiale)

Scrum è un framework per sviluppare e sostenere prodotti complessi che si basa sulla teoria del controllo empirico dei processi o empirismo. L'empirismo afferma che la conoscenza deriva dall'esperienza e *che le decisioni si basano su ciò che si conosce*.

Sono tre i pilastri che sostengono ogni implementazione del controllo empirico di processo

- ✓ **Trasparenza**
- ✓ **Ispezione**
- ✓ **Adattamento**

Scrum utilizza un metodo iterativo ed un approccio incrementale per ottimizzare la prevedibilità ed il controllo del rischio.

Scrum è Un framework che consente alle persone di risolvere problemi complessi di tipo adattivo e, al tempo stesso di creare e rilasciare prodotti in modo efficace e creativo dal più alto valore possibile. Scrum è:

- ✓ **Leggero**
- ✓ **Semplice da comprendere**
- ✓ **Molto difficile da padroneggiare**

Il framework Scrum è costituito dagli Scrum Team e dai ruoli, gli eventi, gli artefatti e le regole ad essi associati e ogni parte del framework serve a uno specifico scopo ed è essenziale per il successo e l'utilizzo di Scrum.

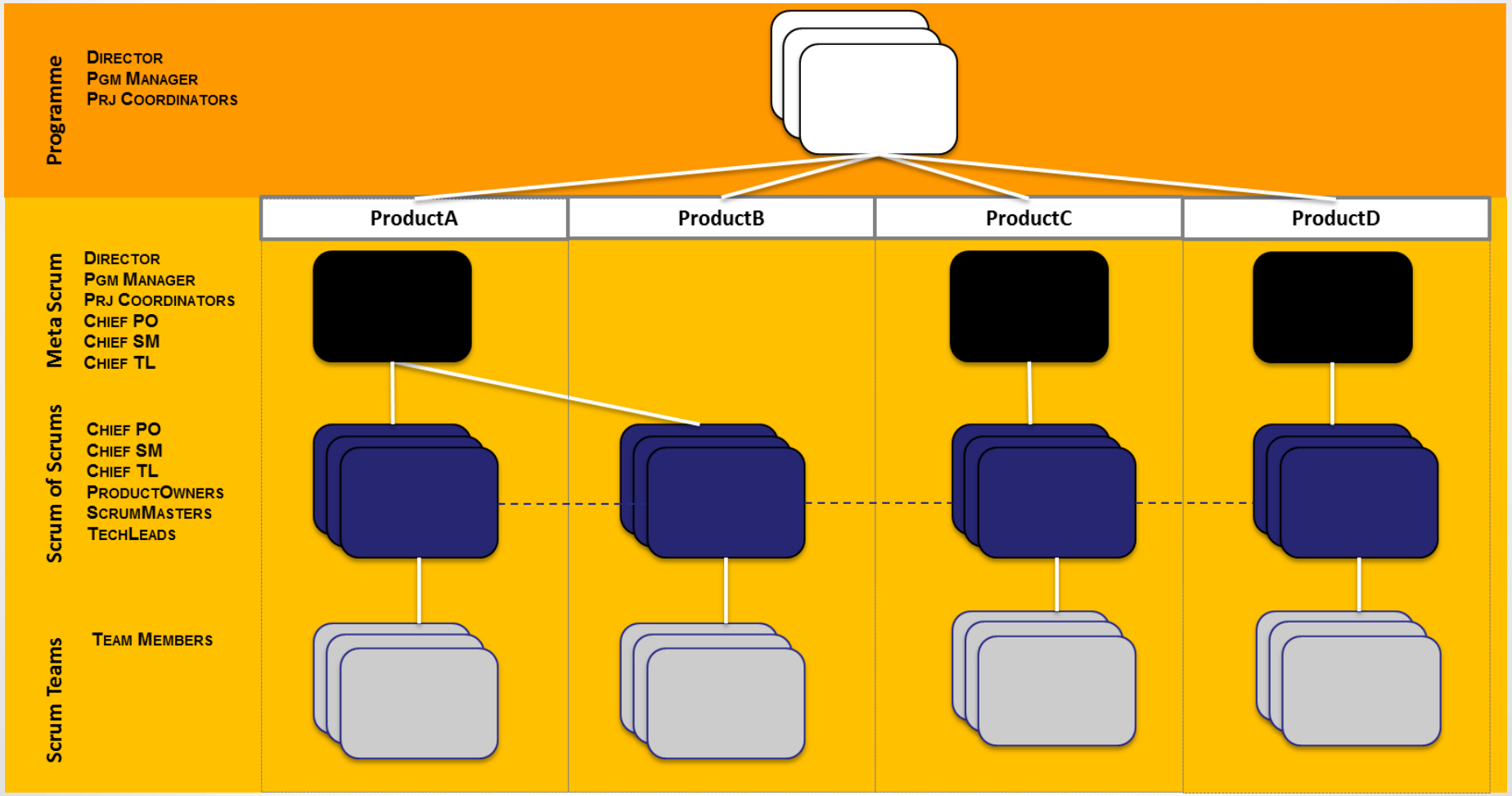
Scrum (Scalabilità)

- ✓ **Lato Team**: Scrum può essere utilizzato come framework agile in combinazione con alcune tecniche di sviluppo agile estratte dall'eXtreme Programming

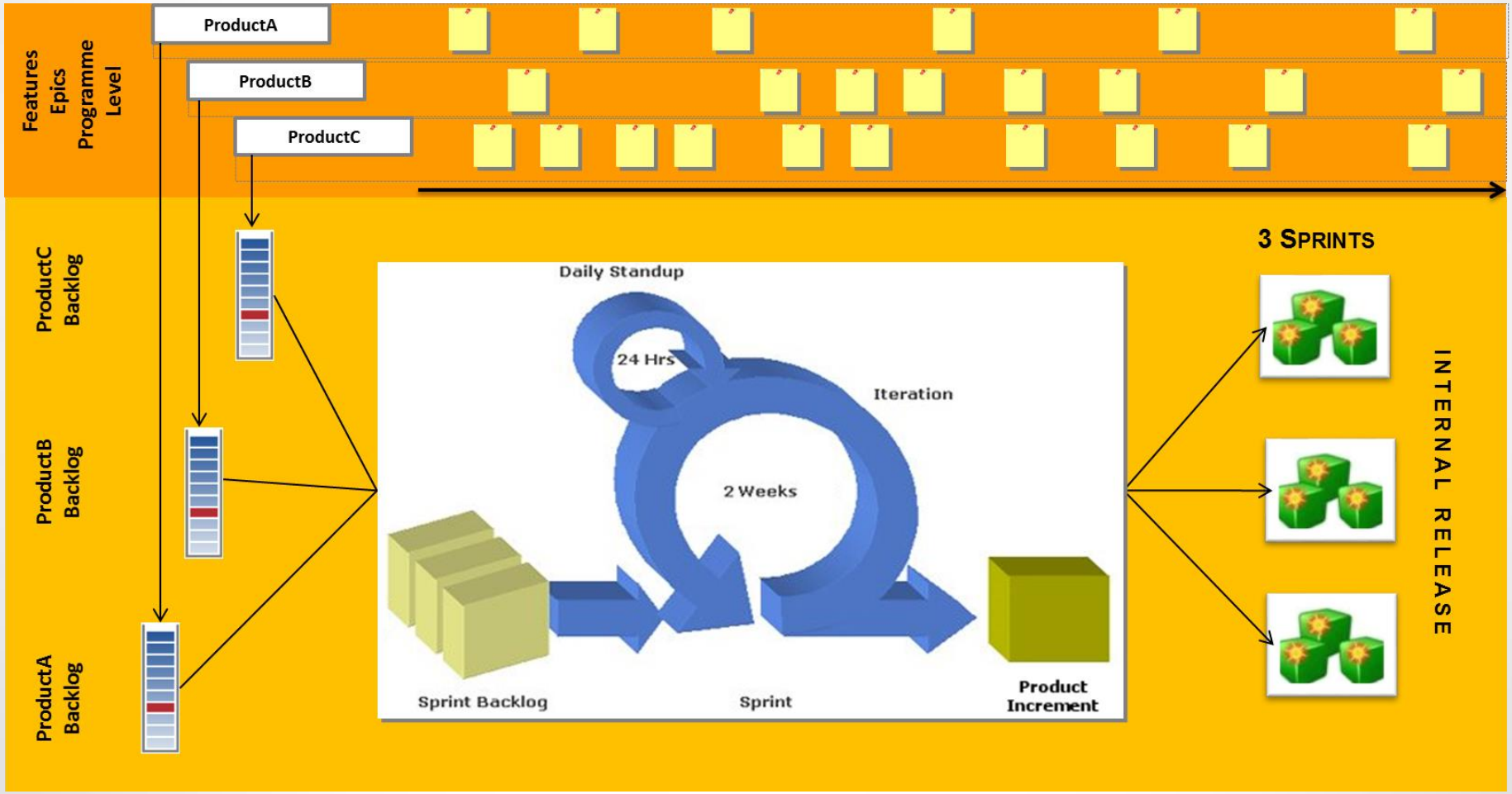
- ✓ **Lato Prodotto**: Nel caso di diversi team che lavorano cntemporaneamente sullo stesso prodotto, è possibile adottare l'approcciodefinito «Scrum of Scrums» per mantenere le squadre in sincronia e per assicurare un regolare ritmo di sviluppo del prodotto.

- ✓ **Scalabilità agile all'interno dell'organizzazione**: Come ottenere una scalabilità agile all'interno di un'organizzazione dipende dall'organizzazione stessa, anche se di regola è sufficiente seguire le direttive del framework Agile

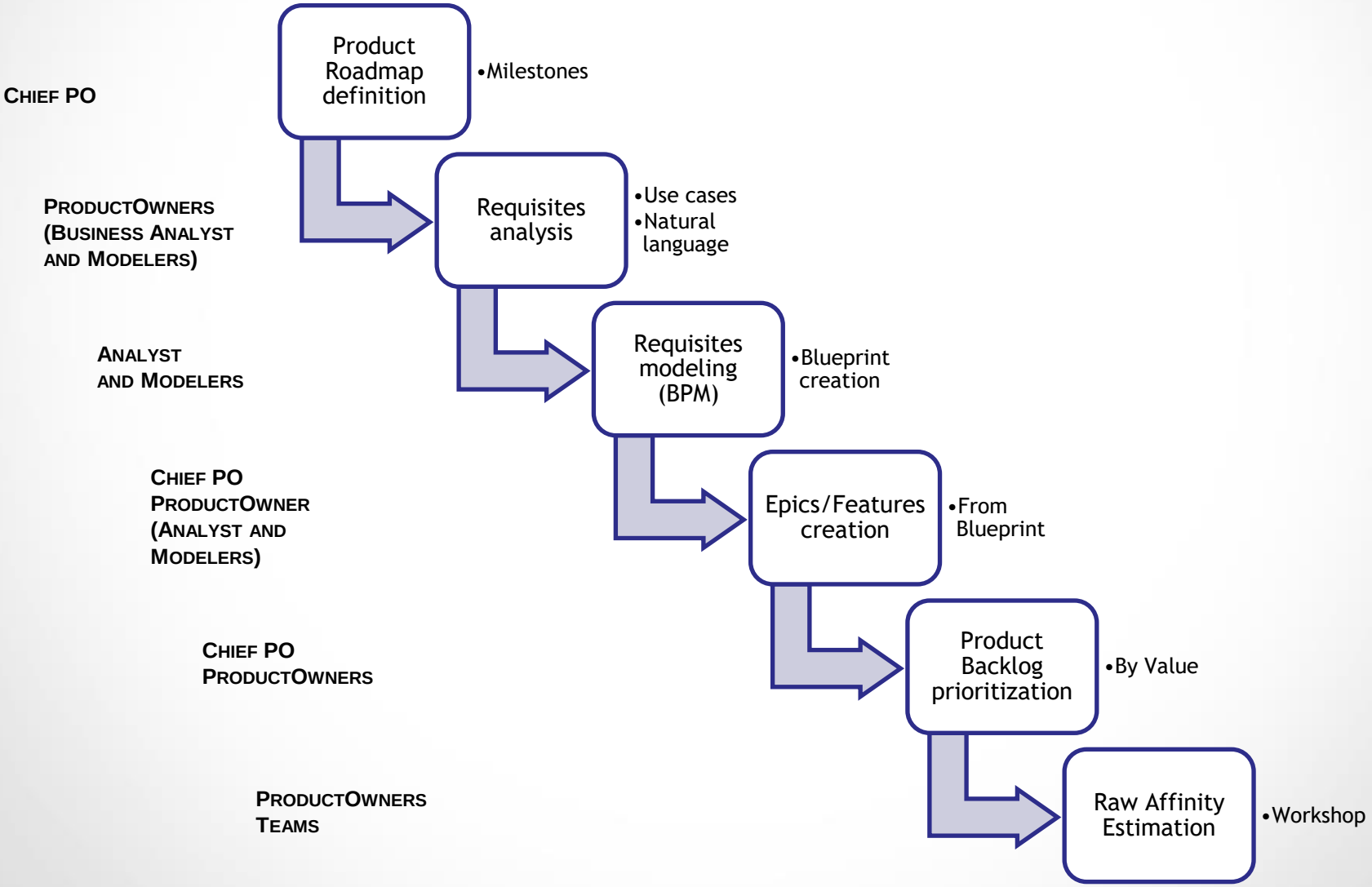
Scrum (Scalabilità nell'organizzazione)



Scrum (Managing Plans & Requirements)

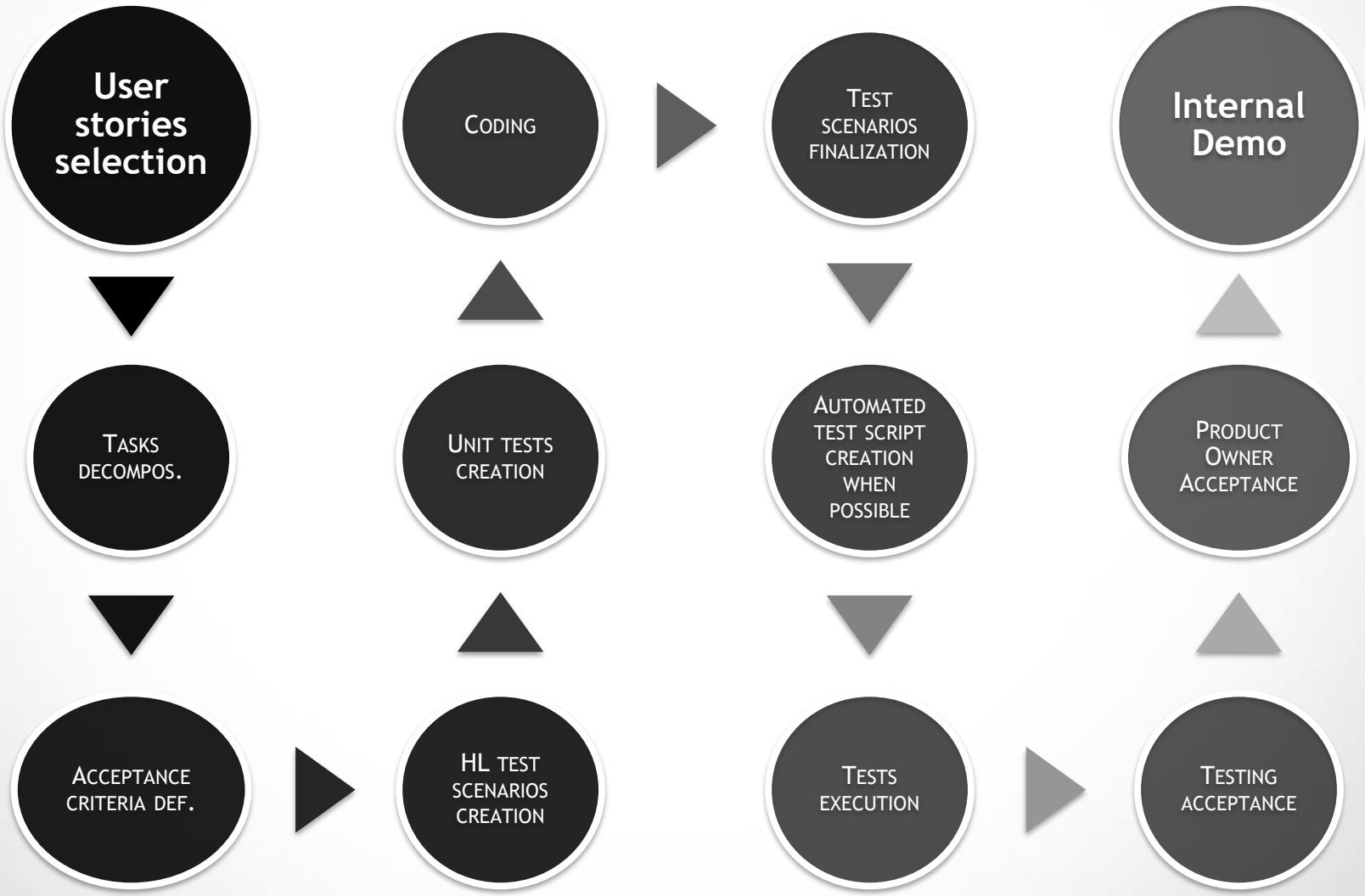


Scrum (Product Backlog Initial Creation)

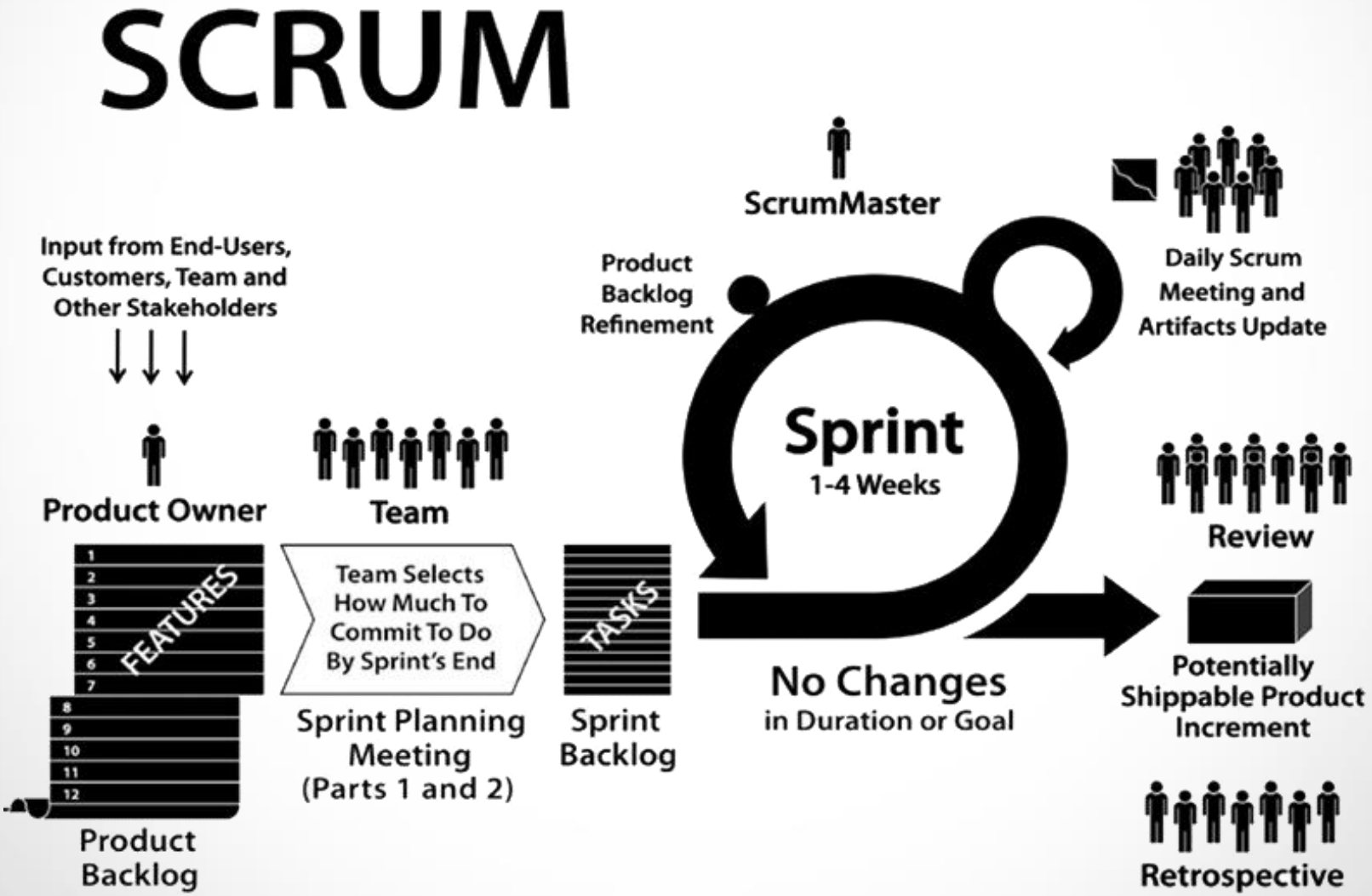


Scrum (Release Planning & Sprint Prepartion)

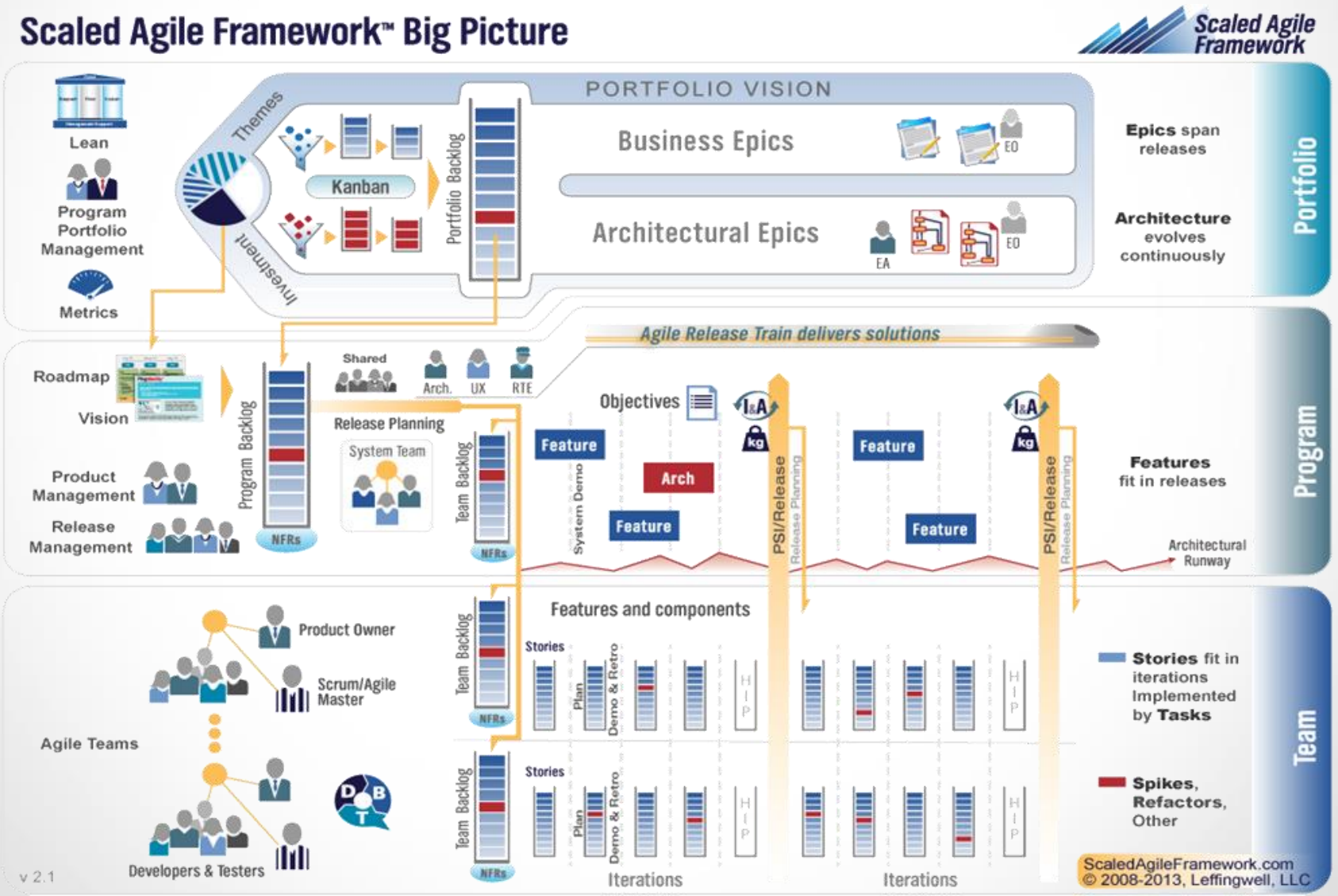
Scrum (User Stories Processing & Development)



Scrum (Concept)



Scrum (Scaled Agile Framework)



Scrum (SAFe ScrumXP)