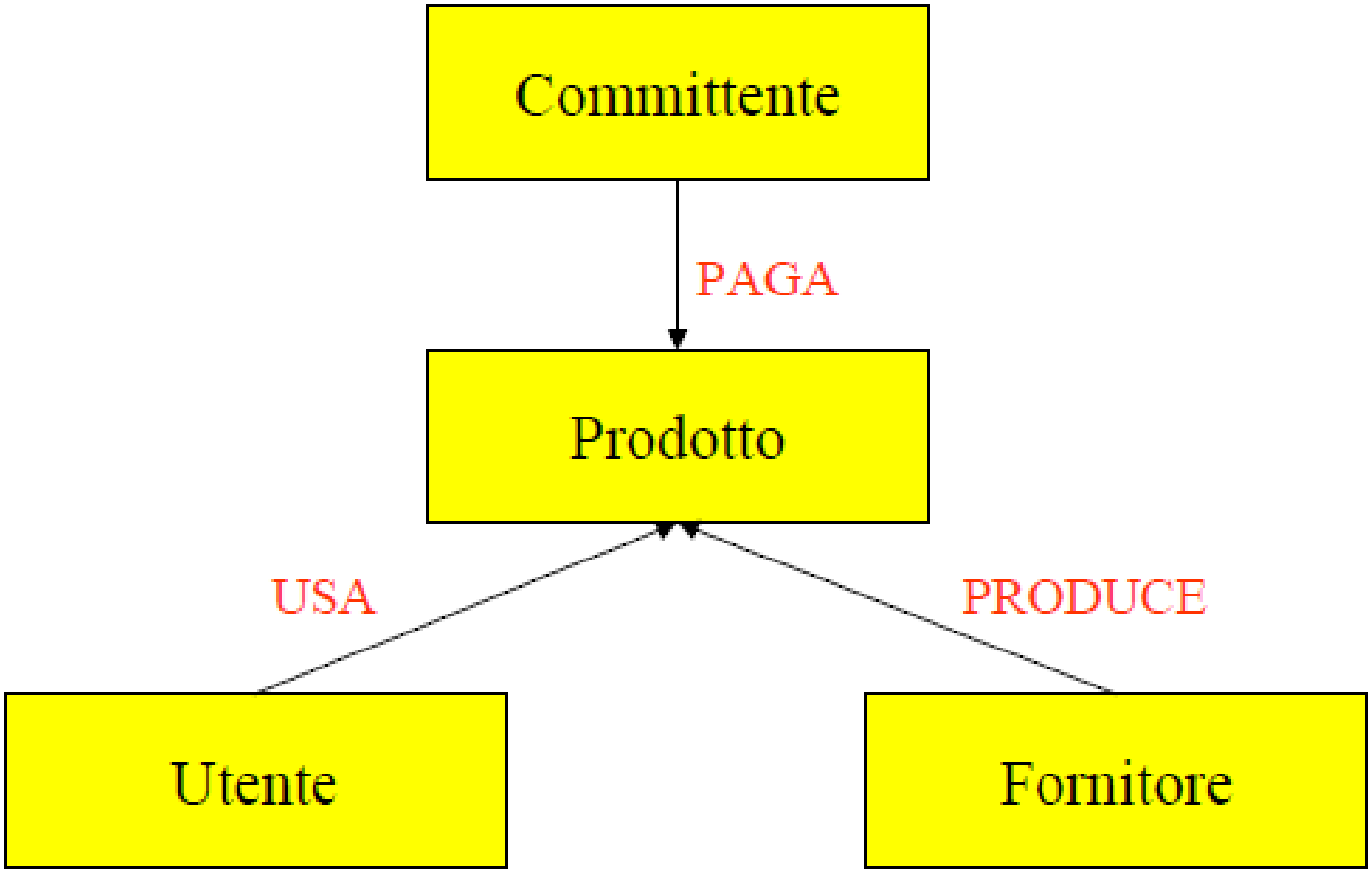
Il processo software
modelli di ciclo di vita

Vincenzo Gervasi, Laura Semini
Dipartimento di Informatica
Università di Pisa

Riassunto lezione precedente
Outline della lezione

- Lezione precedente:
  - Sistemi complessi
  - Necessità di tecniche e strumenti per realizzarli
  - Definizione di Ingegneria del Software
- Questa lezione
  - Organizzazione del lavoro per realizzare un sistema sw
    - Processo sw
    - Modelli di ciclo di vita del sw

Ruoli



Processo

- Un insieme di attività correlate che trasformano ingressi in uscite *(ISO 9000)*
- Modellare il processo significa (Strutturarlo):
  - Suddividerlo in *attività*
  - di ogni *attività,* dire:
    - Cosa
    - Quali prodotti
    - Quando
- Un esempio: ISO 12207

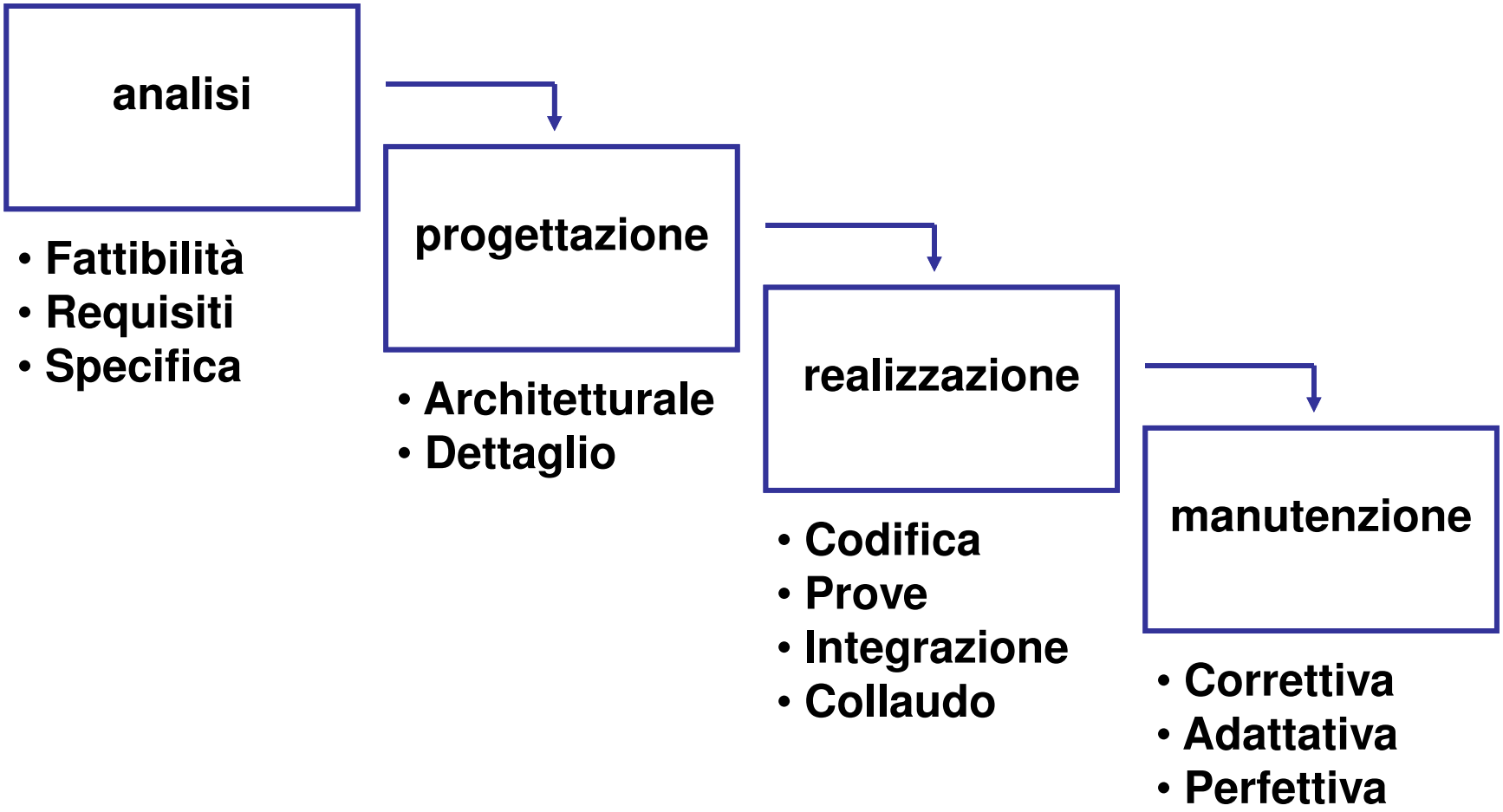
Modello di ciclo di vita

- Organizzazione delle attività
  - Ordinamento delle attività
  - Criteri per terminare e passare alla successiva
- Esempio
  - Preparazione di un dolce
    - Modello di sviluppo generico
    - Non è la ricetta del dolce!
      - Indipendente dal dolce….

Evoluzione dei modelli di ciclo di vita

- Code&Fix: un non-modello
  - Attività non identificate né organizzate
  - Progetti non gestiti
- Modelli prescrittivi
  - Cascata
  - Iterativi
  - Modello a V
  - Spirale
- Unified Process
- Modelli agili
  - Extreme Programming
  - Scrum

Il modello a cascata



Il modello a cascata

- Definito nel 1970 da Royce
- Successione di fasi sequenziali
  - Impossibilità di ritornare a fasi precedenti
  - In caso di eventi eccezionali il processo riparte
- Documentazione
  - Modello "document driven"
  - Ogni fase produce "documenti" che concretizzano la fase
  - I documenti sono necessari per la fase successiva

Caratteristiche delle fasi

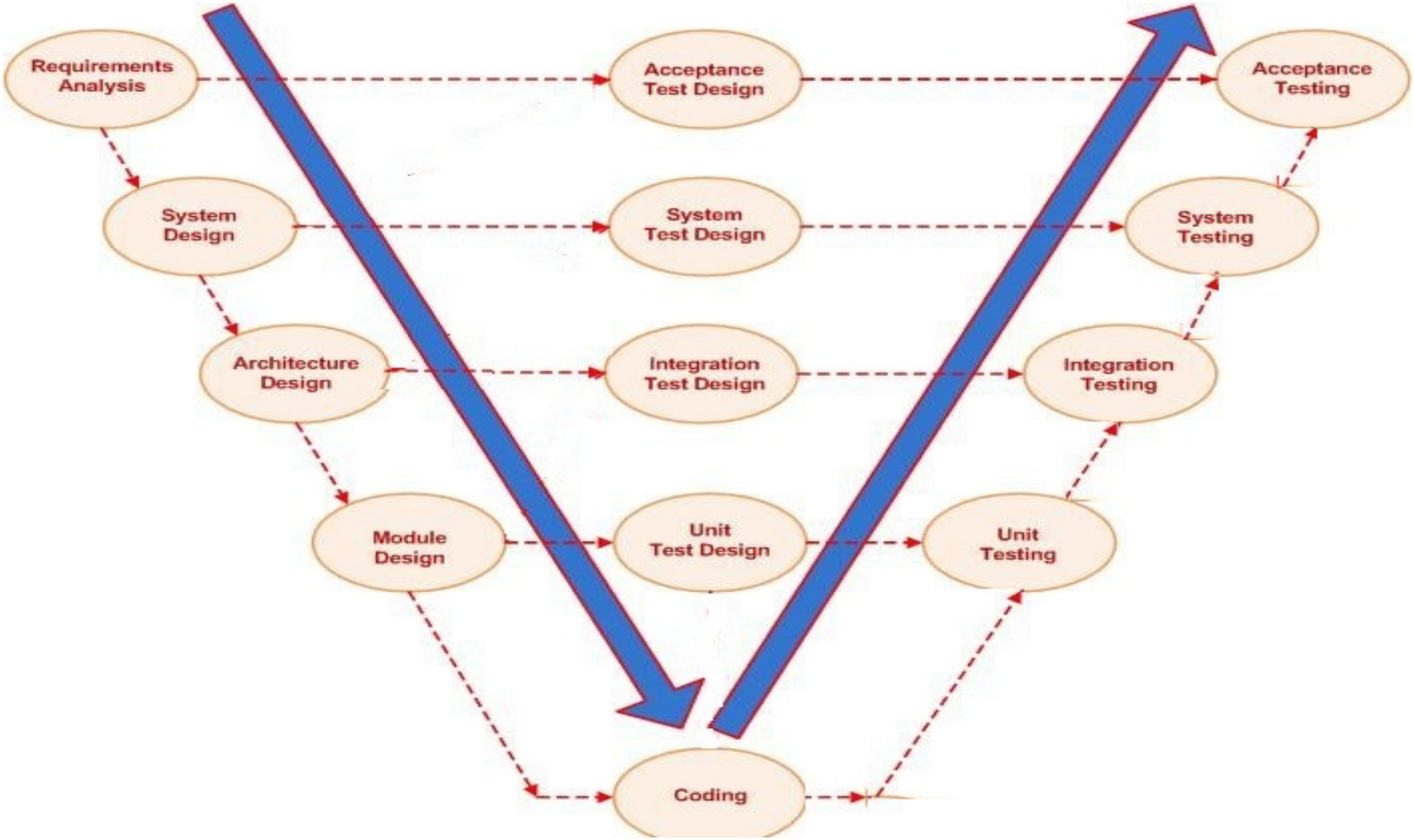
- Le fasi sono descritte in termini di:
  - Attività e prodotti intermedi
  - Contenuti e struttura dei documenti
  - Responsabilità e ruoli coinvolti
  - Scadenze di consegna dei documenti
- Dipendenze causali e temporali
- Riferimento per l'identificazione delle attività

Variazioni al modello a cascata

- Mancanza di flessibilità
  - Rigorosa sequenzialità delle fasi
  - Non prevede cambiamenti nei requisiti
  - Genera molta manutenzione
  - Burocratico e poco realistico
- Variazioni proposte:
  - Cascata con prototipazione
  - Cascata con ritorni

Fasi del modello a cascata vs Fuggetta

Modello a V
Hughes Aircraft ~1982 (sequenziale)

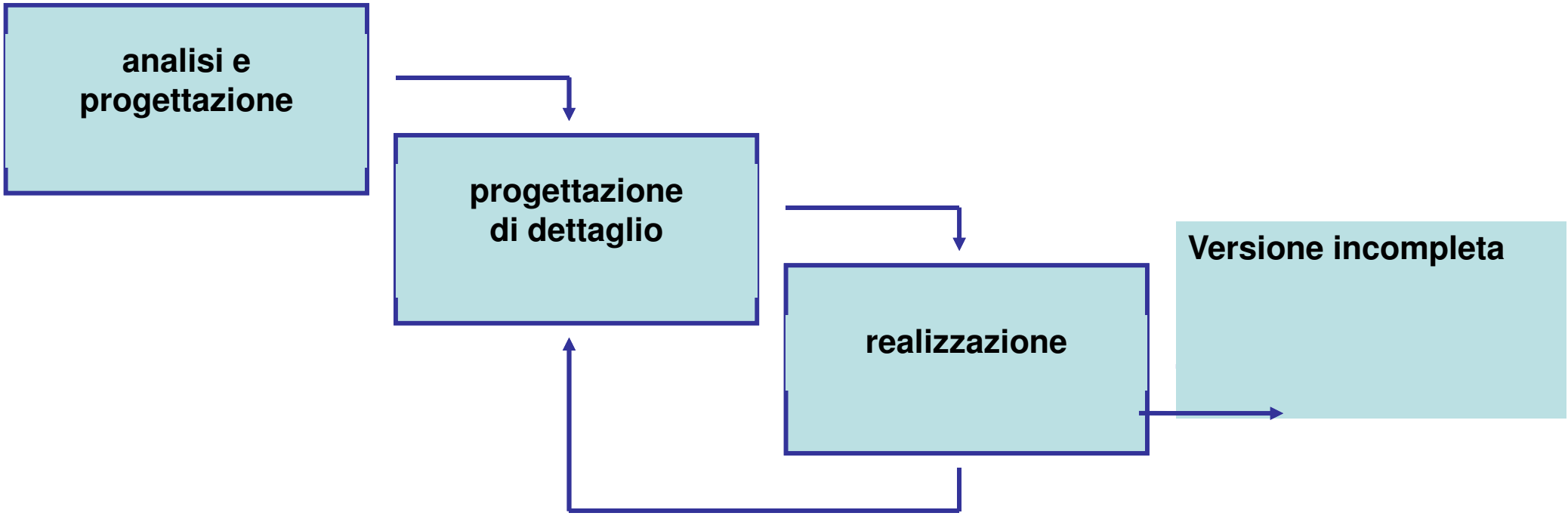


Modelli iterativi

- Necessità di modelli adattabili ai cambiamenti
  - delle soluzioni e delle tecnologie
  - dei requisiti
- Soluzione generale
  - Ritardare la realizzazione delle componenti che dipendono criticamente da fattori esterni (tecnologie, hardware sperimentale, ecc)
- Le iterazioni sono pianificate

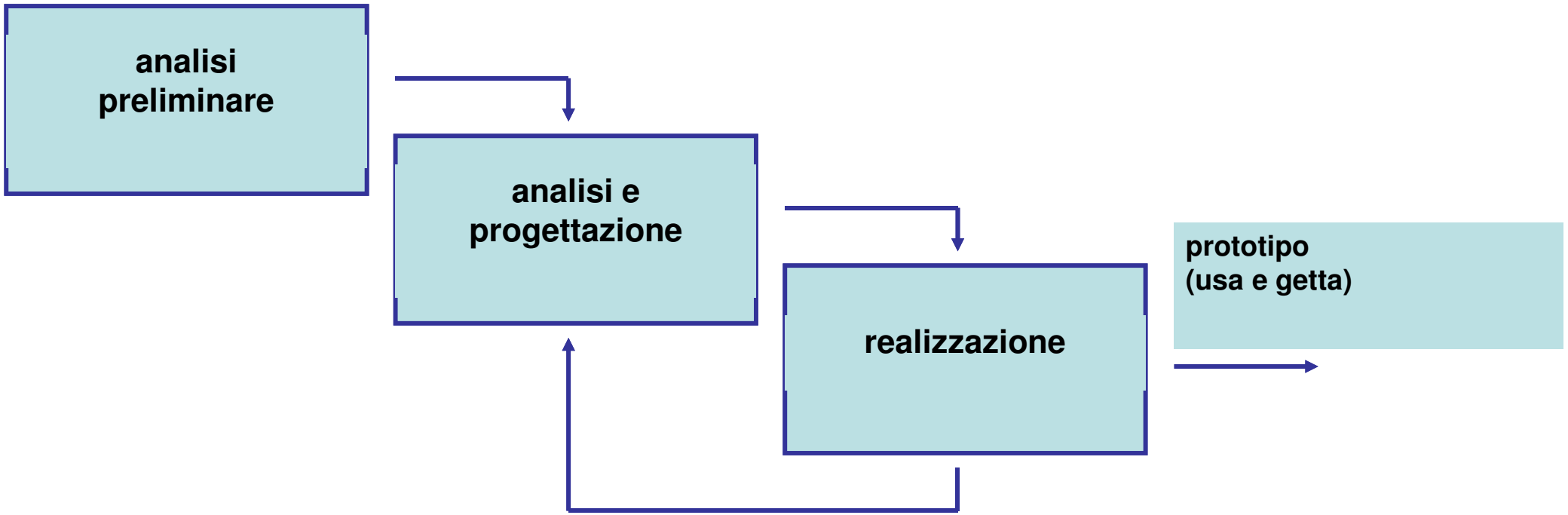
Modello incrementale

- Requisiti stabili
- Necessità di "uscire" con qualcosa velocemente



Modello evolutivo

- Requisiti instabili
- Necessità di "uscire" con un prototipo



Il modello a spirale

- Proposto da Boehm nel 1988
- È un modello astratto: va specializzato
- Ogni iterazione organizzata in 4 fasi

- Note:
  - attenzione alla parole "incrementale" nel Fuggetta → "incrementale o evolutivo" .
  - Errore nella legenda Figura 8.3 (Fuggetta): invertire i colori

Il modello a spirale

Il modello a spirale

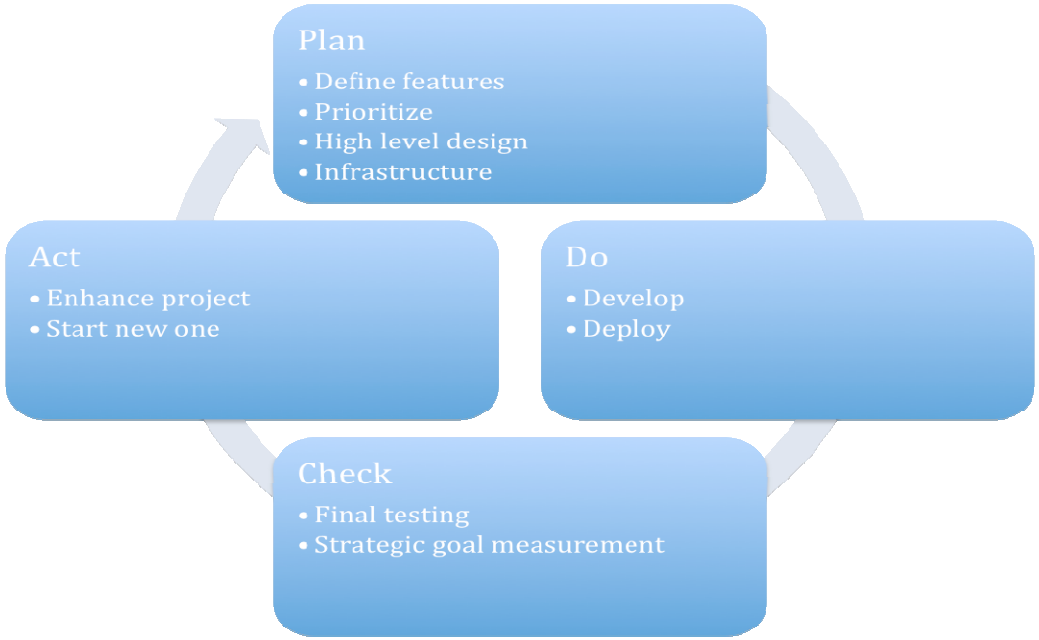
- Evidenzia gli aspetti gestionali
  - Pianificazione delle fasi
  - Analisi dei rischi (modello “risk driven”)

- Tipici Rischi: Dominio poco noto, Linguaggi, strumenti nuovi, Personale non addestrato

- Applicabile ai cicli tradizionali
- Maggior comunicazione e confronto con il committente
- Ispirato dal plan-do-check-act plan.

Modello a spirale ispirato da PDCA

- Plan
  - Identifying and analyzing the problem
- Do
  - Developing&testing a potential solution
- Check
  - Measuring how effective the test solution was, and analyzing whether it could be improved in any way
- Act
  - Implementing the improved solution



- [William Edwards Deming 1950]

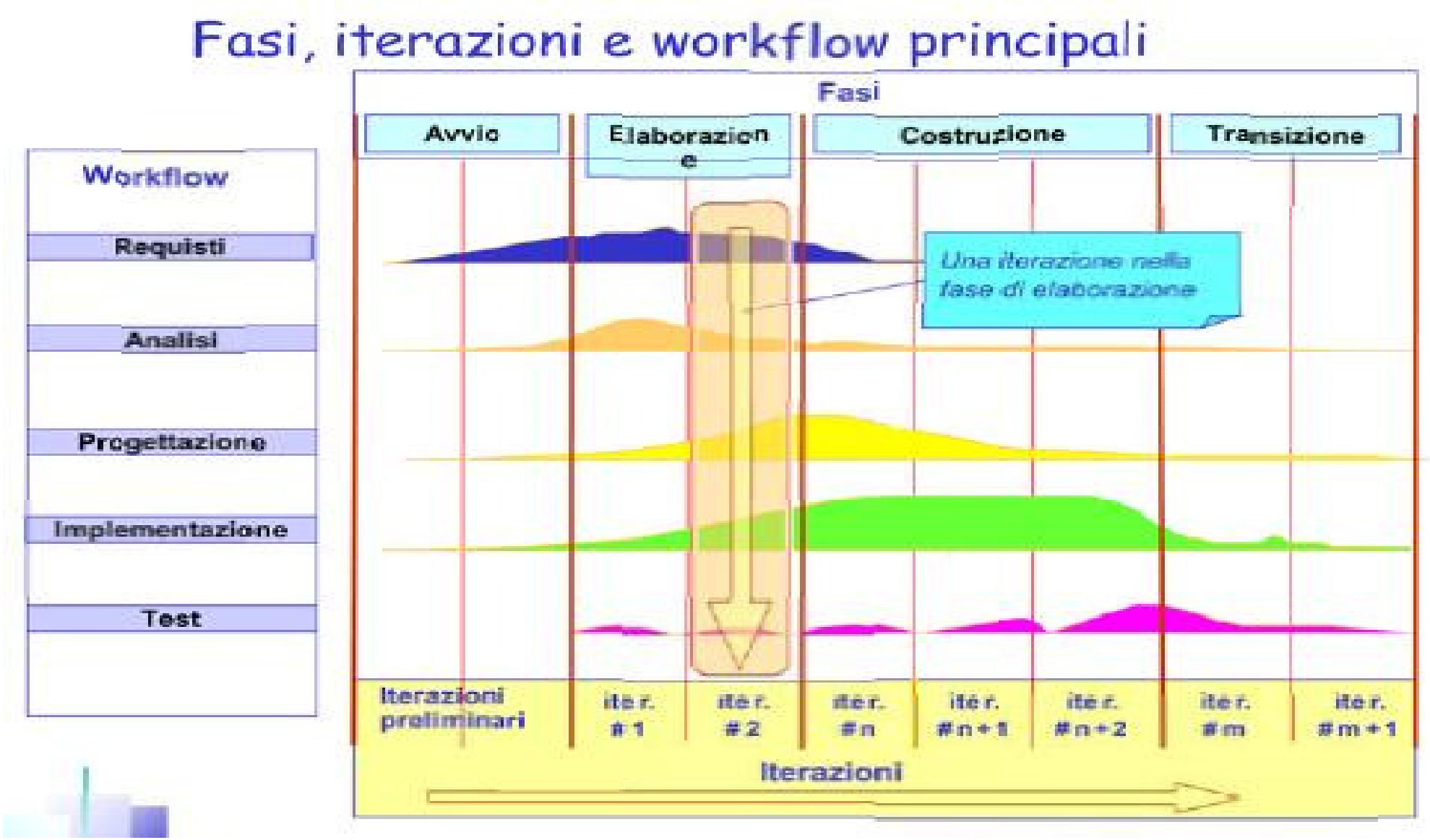
Unified Process

- Proposto nel 1999 da
  - Grady Booch, Ivar Jacobson, James Roumbaugh
- Caratteristiche
  - Guidato dai casi d'uso e dall'analisi dei rischi
    - Raccolta dei requisiti e passi successivi guidati dallo studio degli use case
  - Incentrato sull'architettura
    - il processo assegna alla descrizione dell'architettura del sistema un ruolo molto importante. L'approccio è infatti quello di concentrarsi, soprattutto nelle prime fasi, sull'architettura di massima, lasciando i dettagli alle fasi successive. In tal modo è possibile avere da subito una visione generale del sistema facilmente adattabile al cambiamento dei requisiti
- Iterativo incrementale

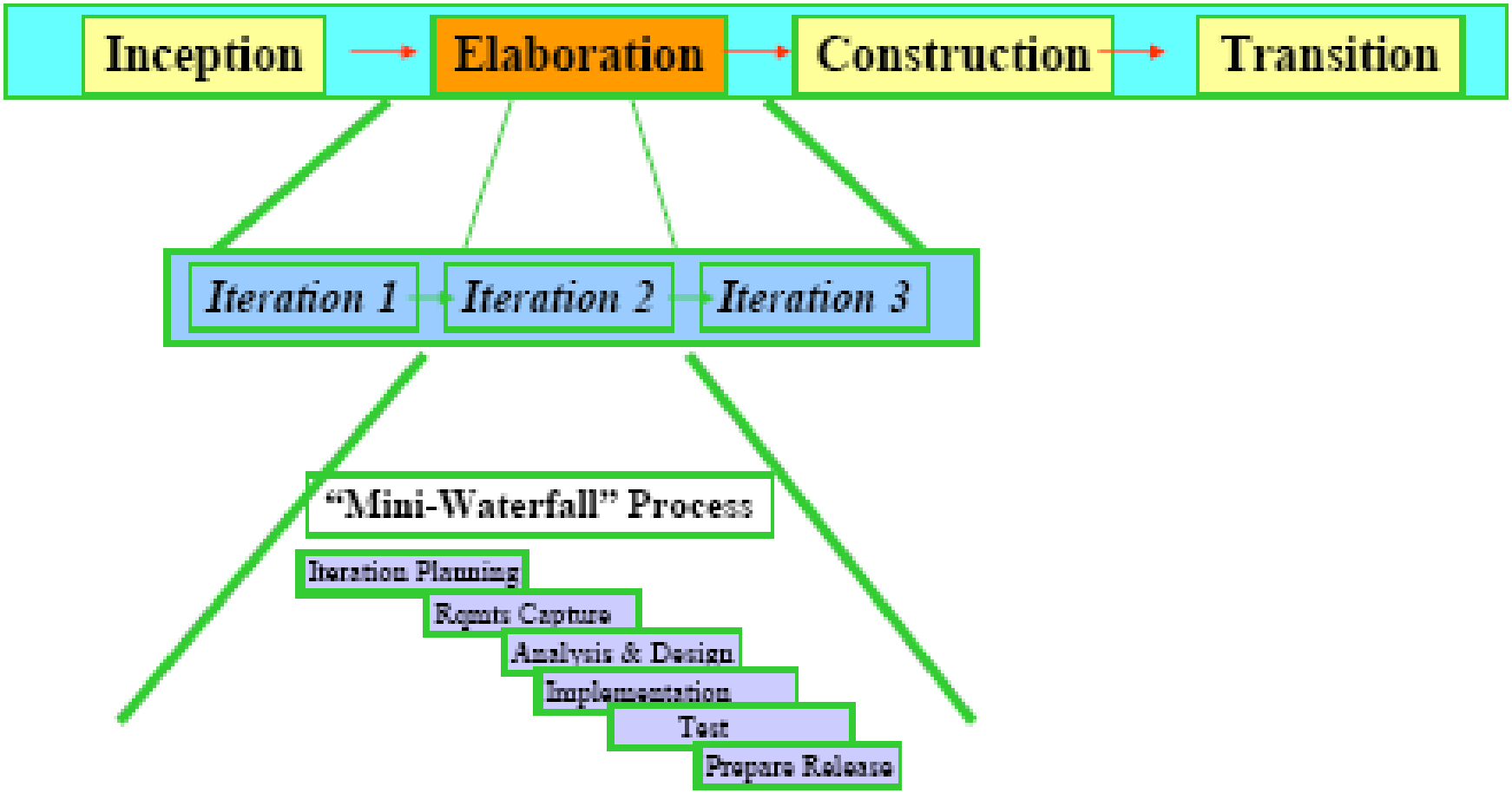
Fasi (temporali) di UP

- Avvio
  - Fattibilità; Analisi dei rischi; Requisiti essenziali per definire il contesto del sistema; Eventuale prototipo
- Elaborazione
  - Analisi dei requisiti; Analisi dei rischi; Sviluppo di un'architettura base; Piano per la fase di costruzione
- Costruzione
  - analisi, disegno, implementazione, testing
- Transizione
  - Beta testing, aggiustamento delle prestazioni, creazione di documentazione aggiuntiva, attività di formazione, guide utenti, creazione di un kit per la vendita

Schema di UP


Fasi, iterazioni e workflow principali
Fasi
Avvio
Elaborazione
Costruzione
Transizione
Workflow
Requisti
Analisi
Progettazione
Implementazione
Test
Una iterazione nella fase di elaborazione
Iterazioni preliminari
iter. #1
iter. #2
iter. #n
iter. #n+1
iter. #n+2
iter. #m
iter. #m+1
Iterazioni

Fasi e iterazioni



Processi agili

- Per **metodologia agile (o leggera)** o **metodo agile** si intende un particolare metodo per lo sviluppo del software che coinvolge quanto più possibile il committente.
- Adatti a progetti con meno di 50 persone
- Una metodologia agile si basa sui **principi** del **Manifesto di Snowbird, feb 2001**

Manifesto di Snowbird

- **Comunicazione**:
  - le persone e le interazioni sono più importanti dei processi e degli strumenti
  - (tutti possono parlare con tutti, persino l'ultimo dei programmatori con il cliente);
  - (ossia le relazioni e la comunicazione tra gli attori di un progetto software sono la miglior risorsa del progetto);
  - bisogna collaborare con i clienti al di là del contratto
    - la collaborazione diretta offre risultati migliori dei rapporti contrattuali;
- **Semplicità**: analisti mantengano la descrizione formale il più semplice e chiara possibile
  - è più importante avere software funzionante che documentazione
  - bisogna mantenere il codice semplice e avanzato tecnicamente, riducendo la documentazione al minimo indispensabile;
- **Feedback**
  - bisogna rilasciare nuove versioni del software ad intervalli frequenti
  - sin dal primo giorno si testa il codice
- **Coraggio**
  - si dà in uso il sistema il prima possibile e si implementano i cambiamenti richiesti man mano
  - bisogna essere pronti a rispondere ai cambiamenti più che aderire al progetto

eXtreme Programming
Esempio di processo *agile*

- Si basa su un insieme di prassi:

- Pianificazione flessibile
  - basata su scenari proposti dagli utenti
  - coinvolge i programmatori
- Rilasci frequenti
  - due-quattro settimane
  - inizio di una nuova pianificazione

eXtreme Programming
Esempio di processo *agile*

- Metafora condivisa
  - per descrivere una funzionalità del sistema software nel gruppo si usa una metafora.
  - Ad esempio, per descrivere **un gruppo di agenti software** in azione "**lo sciame di piccole api volano sui fiori e rientrano all'alveare solo una volta terminato il loro compito**".
  - Aiuta a far sì che il team usi un sistema comune di nomi di entità, tale che sia immediato trovare, per uno sviluppatore, una certo modulo in base al nome, o sia chiaro dove inserire le nuove funzionalità appena sviluppate.

eXtreme Programming
Esempio di processo *agile*

- Progetti semplici
  - comprensibili a tutti
- Verifica (testing)
  - di unità e di sistema (basati sugli scenari)
  - supporto automatico
- Test Driven Development
  - casi di test = specifica
- Cliente sempre a disposizione (circa ogni settimana)

eXtreme Programming
Esempio di processo *agile*

- Programmazione a coppie
  - un solo terminale, il *driver* scrive il codice mentre il *navigatore* controlla il lavoro del suo compagno in maniera attiva.
- No lavoro straordinario
- Collettivizzazione del codice
  - accesso libero
  - integrazione continua
  - standard di codifica

eXtreme Programming
Esempio di processo *agile*

- Code Refactoring
  - modifying it without changing its behavior,
  - Uno dei motti del XP è "se un metodo necessita di un commento, riscrivilo!" (codice *auto-esplicativo*).
- Daily Stand Up Meeting

SCRUM

- SCRUM è un processo "agile" il cui nome deriva dalla terminologia del gioco del Rugby. Si basa sulla teoria della complessità (www.controlchaos.com)
- E' un processo
  - Che può essere adottato per gestire e controllare lo sviluppo del software
  - E' iterativo, incrementale, per lo sviluppo e gestione di ogni tipologia di prodotto
  - Fornisce alla fine di ogni iterazione un set di funzionalità potenzialmente rilasciabili

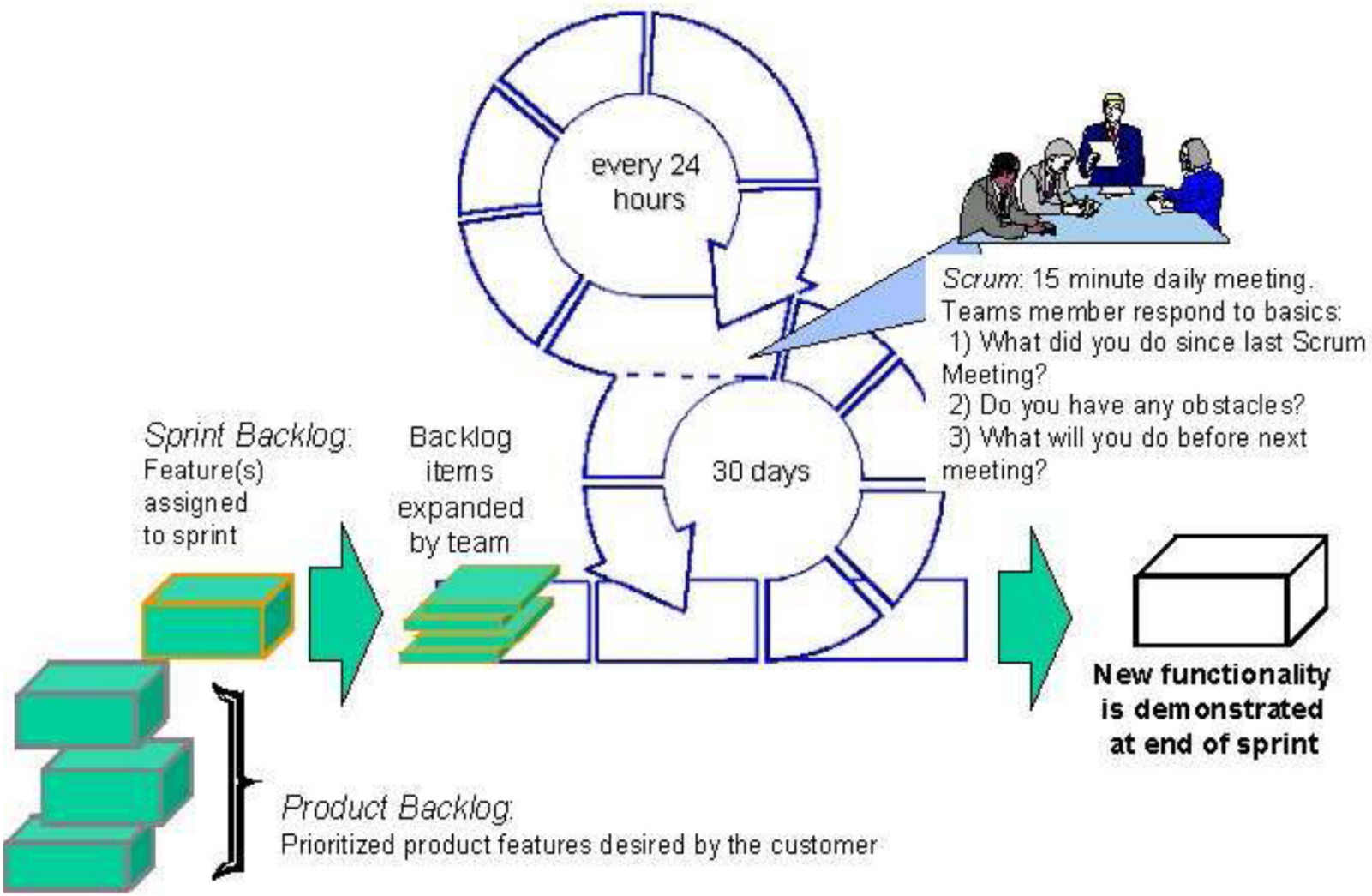
SCRUM, tre fasi: fase 1

- **Pre-game phase**
  - **Planning sub-phase**
    - Include la definizione del sistema che deve essere sviluppato. Viene creata una Product Backlog List, che contiene tutti i requisiti attualmente conosciuti
  - **Architecture sub-phase**
    - Viene pianificato un design di alto livello del sistema, inclusa l'architettura, in base agli elementi contenuti nel Product Backlog

SCRUM, tre fasi: fase 2

- **Development (Game) phase (parte "agile" dell'approccio Scrum)**
  - **Nella Development Phase, il sistema viene sviluppato attraverso una serie di Sprint**
    - Cicli iterativi nei quali vengono sviluppate o migliorate una serie di funzionalità
    - Ciascuno Sprint include le tradizionali fasi di sviluppo del software
    - L'architettura ed il design del sistema evolve durante lo sviluppo negli Sprint
    - Uno Sprint si svolge in un intervallo di tempo che va da una settimana ad un mese

SCRUM: sprint e schedule quotidiano



SCRUM, tre fasi: fase 3

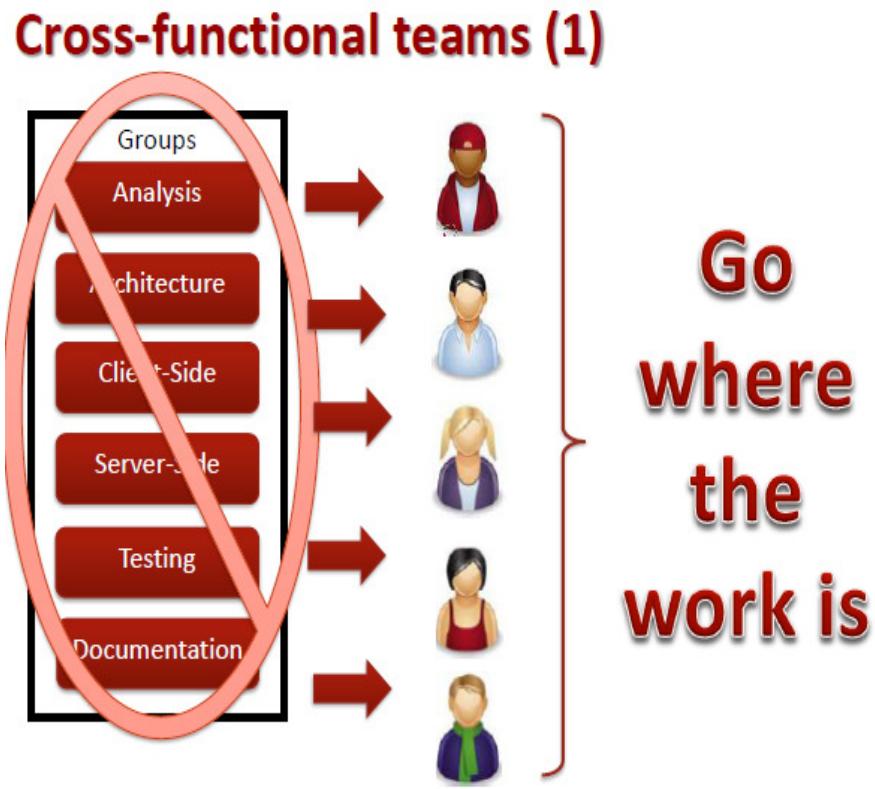
- **Post-game phase (contiene la chiusura definitiva della release)**

Tre ruoli
Product Owner

- Responsibilities
  - Product features identification
  - Return of Investment
  - Feature list priority (continuous)
- A single person
  - The customer or the voice of the customer
  - Product Manager or Product Marketing Manager
- Powers:
  - Drives directly the development from a business perspective
  - Accepts or rejects work results
  - Stops a Sprint if necessary

Tre ruoli
Team member

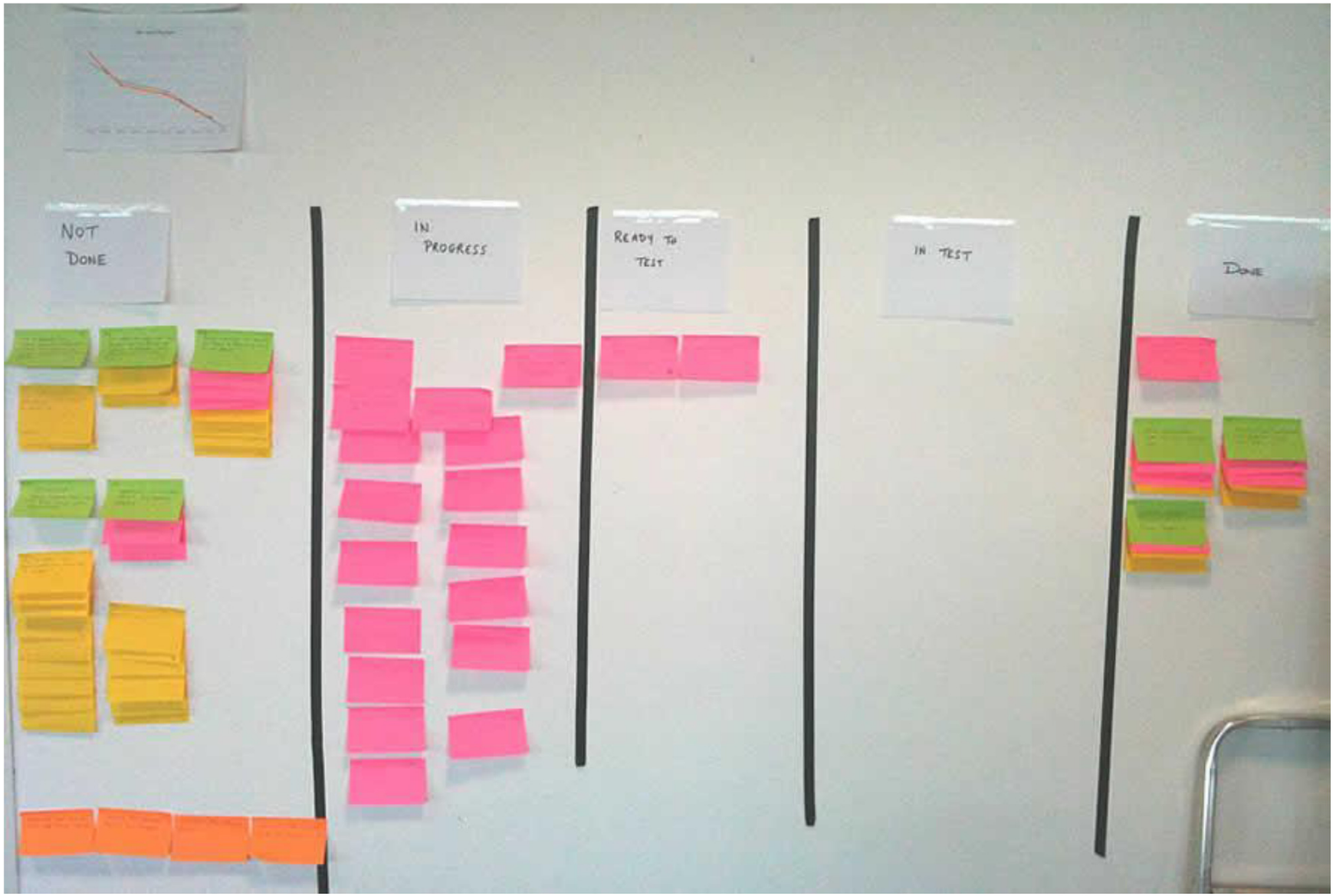
- Responsibilities
  - Build the product
  - Commit what they can do in a Sprint
- Characteristics
  - Cross-functional
    - Do not wait for "someone who can do it faster than us"
    - Reduce waste
    - Handoff
    - People waiting while other are overwhelmed
    - Increase learning
    - No role other than team member inside a team
- Self-organizing
- No project (or team) manager
- Avoid multitasking
- Feature teams
  - 7 + 2 persons
  - Ideally co-located



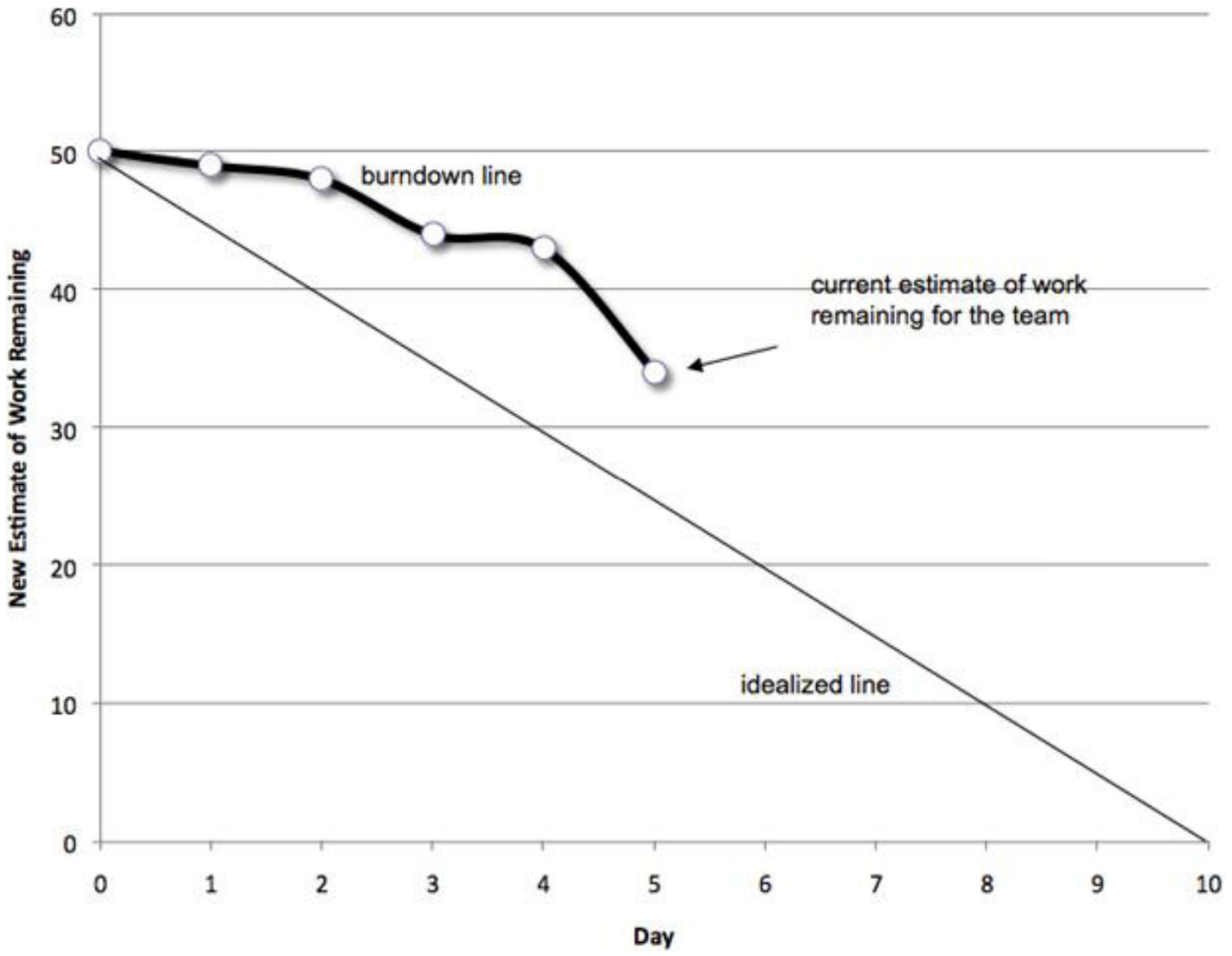
Tre ruoli
Scrum Master

- Responsibilities
  - Teaches Scrum to all roles
  - Coach for the team
  - Removes barriers for productivity
  - Shields the team from external interferences
- The Scrum Master is not
  - A team manager
  - A project manager
- Does not have authority over the team
- Typical behaviour: “I observe [thing], what should we do?”

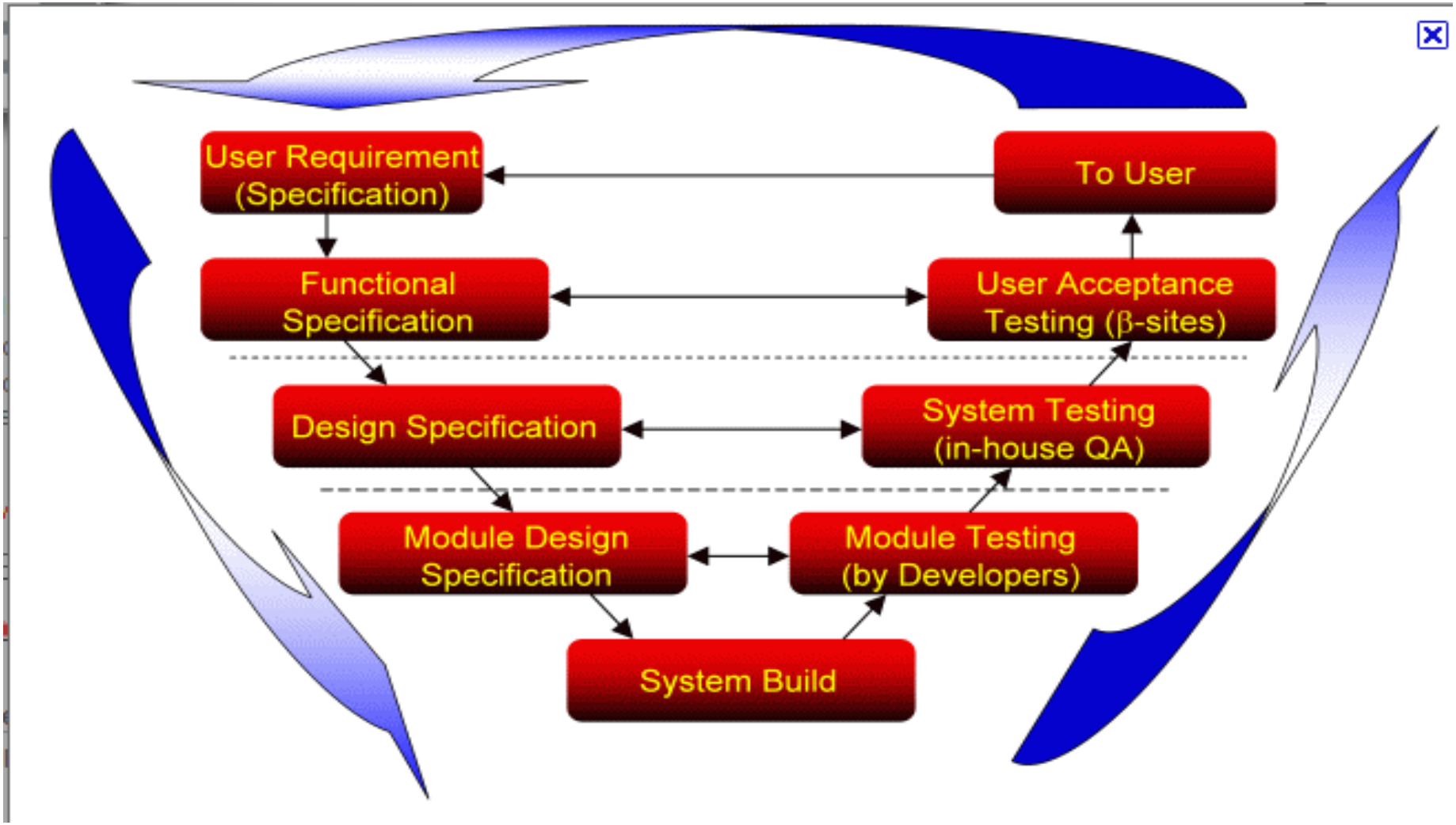
SCRUM: backlog



SCRUM: backlog



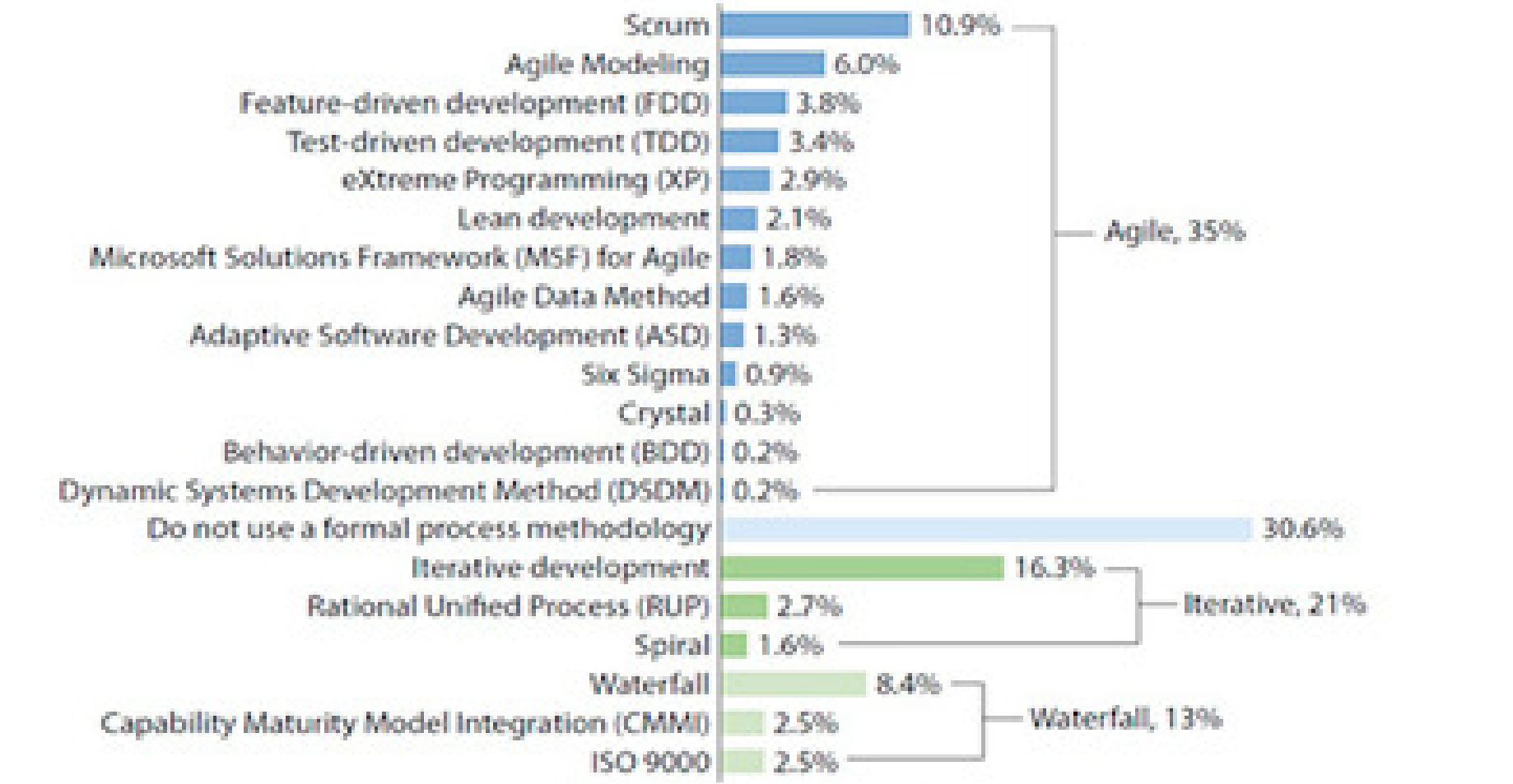
Versione agile del V-modell



Modello a componenti

- Fasi
  - Identificazione dei componenti necessari
  - Ricerca
  - Realizzazione di componenti (eventuale)
  - Integrazione e testing del componente
- Favorisce il riuso
  - Non solo di cose sviluppate in casa, ma anche di software comprato o free / open source o no.
- Nessuno sviluppa un DBMS, oggi!

Una rassegna



Bibliografia

- Cap 8 Fuggetta
- Cap 2 Arlow

- Altre letture che potrebbero interessarvi
  - Il manifesto di Snowbird
  - Ken Schwaber e Jeff Sutherland, *La Guida a Scrum. La Guida Definitiva a Scrum: Le Regole del Gioco* (**PDF**), Scrum.Org and ScrumInc, 2014.