Il processo software
modelli di ciclo di vita

Laura Semini
Dipartimento di Informatica
Università di Pisa

Il processo software

Con processo software si indica il percorso da svolgere per sviluppare un prodotto o sistema software.

- Inizia con l'esplorazione dell'idea e finisce con la dismissione del software
- Durante la sua vita il prodotto attraversa varie fasi
- Il processo software include anche gli strumenti e le tecniche per svilupparlo e i professionisti del software coinvolti

Processo

- 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
  - Possiamo dare un modello di sviluppo generico:
    - Fare la spesa, mentre il forno si scalda: impastare, mettere nella teglia, infine infornare.
  - Non è la ricetta del dolce: è indipendente dal dolce

Evoluzione dei modelli di ciclo di vita

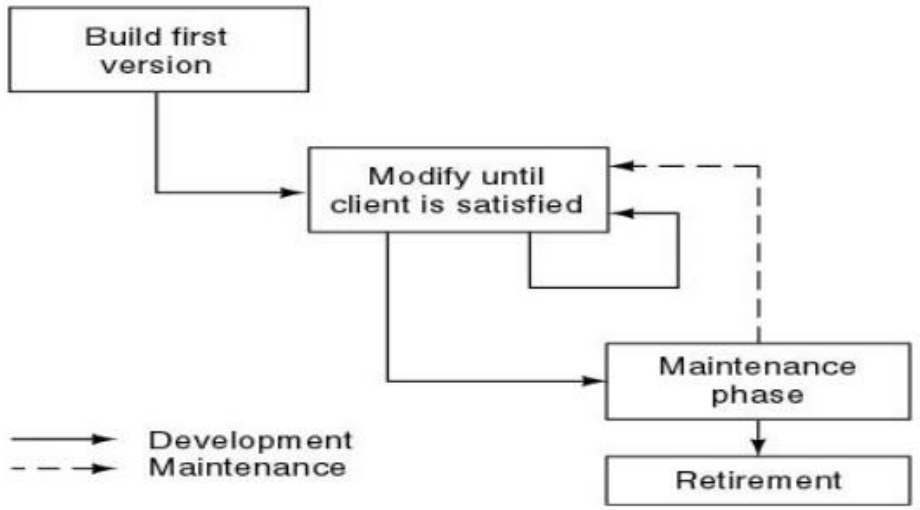
- Build-and-Fix: un non-modello
  - Attività non identificate né organizzate
  - Progetti non gestiti
- Modelli prescrittivi
  - Cascata
  - Modello a V
  - Rapid Prototyping
  - Modello incrementale
  - Modello a spirale
- Unified Process
- Modelli agili
  - Extreme Programming
  - Scrum

Così tanti???

- Li vedremo tutti perché:
  - Sono i più noti
  - Ma soprattutto perché ognuno di essi ha affrontato un nuovo aspetto
    - Sono queste caratteristiche peculiari che ci interessano e che guidano lo sviluppatore nella scelta di un ciclo di vita per ogni progetto
    - Un ciclo di vita di un progetto reale può mescolare idee dei modelli che vedremo

Build-and-Fix Model

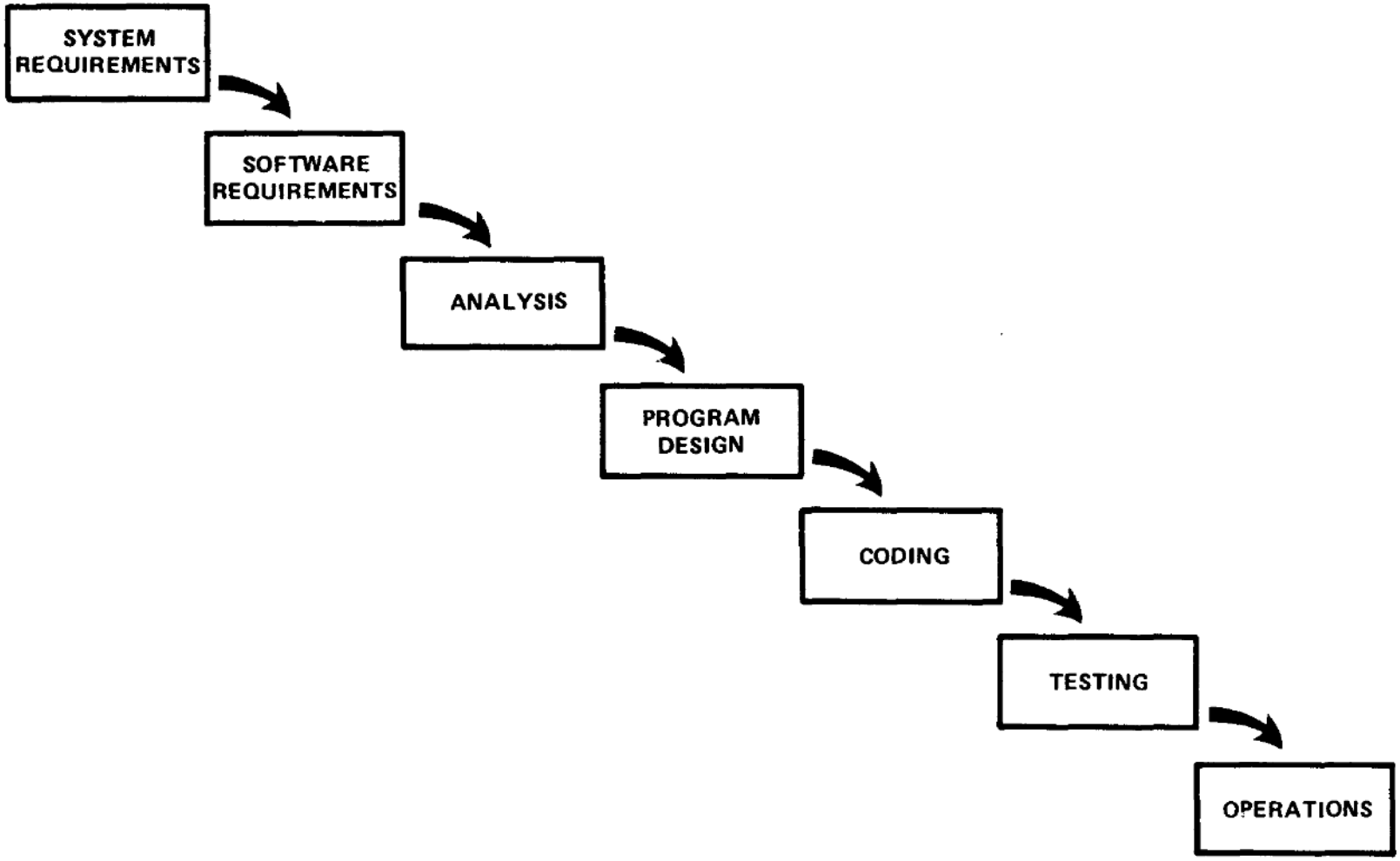
- Il prodotto è sviluppato senza specifica e senza un tentativo di progettazione
- lo sviluppatore scrive un programma
- che poi è modificato più volte finché non soddisfa il committente



Build-and-fix Model

- Forse adeguato a un progetto di 100 linee di codice
- Diventa totalmente improponibile per prodotti di ragionevole grandezza
- Inoltre la manutenzione di un prodotto senza specifica o documentazione che ne spieghi la progettazione è estremamente difficile
- Prima di iniziare lo sviluppo di un progetto dobbiamo scegliere un "vero" modello del ciclo di vita

Il modello a cascata [Royce,1970]

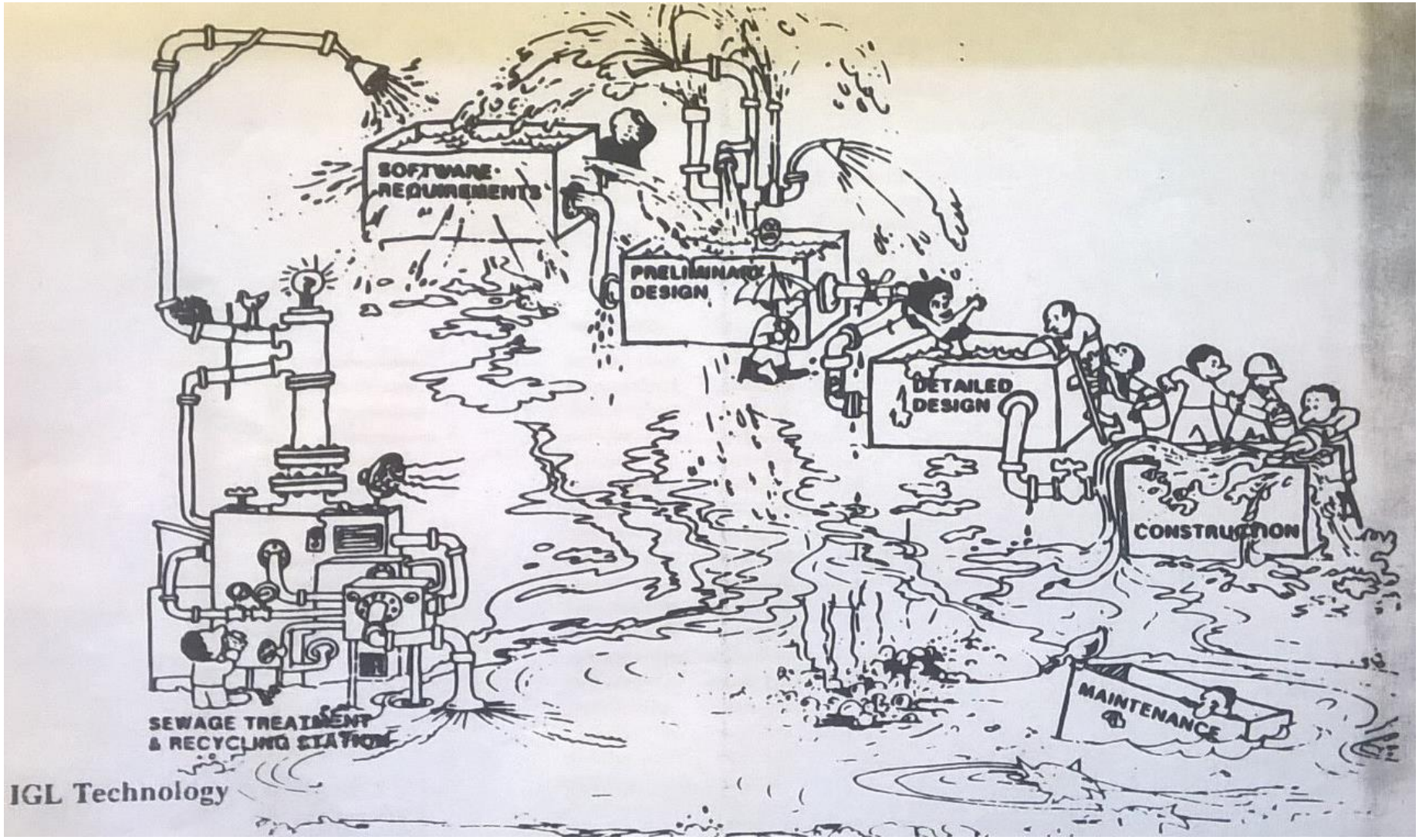


Il modello a cascata [Royce,1970]

- Il valore di questo modello è stato quello di distinguere e definire le fasi di un processo software.
  - Evidenziando l'importanza delle fasi di analisi e di progettazione prima di passare alla codifica.

- Inoltre il modello richiede che il passaggio a una nuova fase sia possibile solo dopo il completamento della precedente:
  - Ogni fase produce un documento che deve essere approvato da un gruppo di valutatori prima di passare alla fase successiva
  - Si parla anche di modello "document driven"

Critiche al modello a cascata....

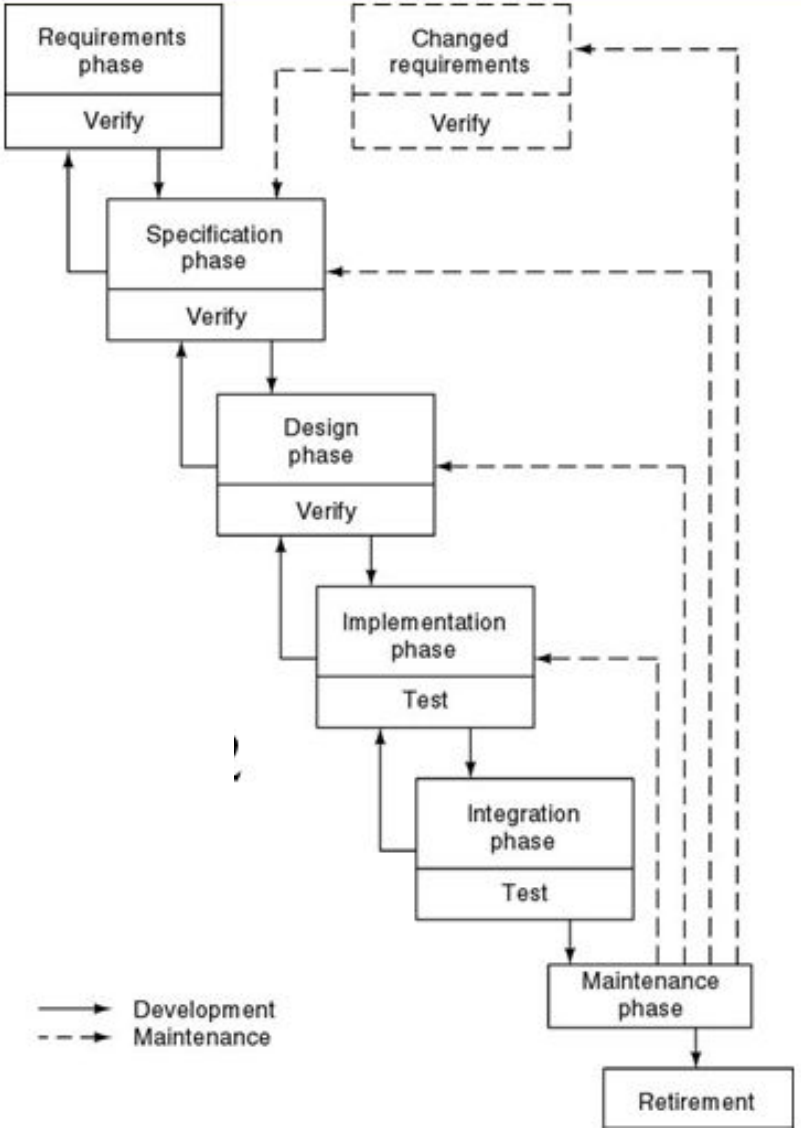


Analisi del modello a cascata

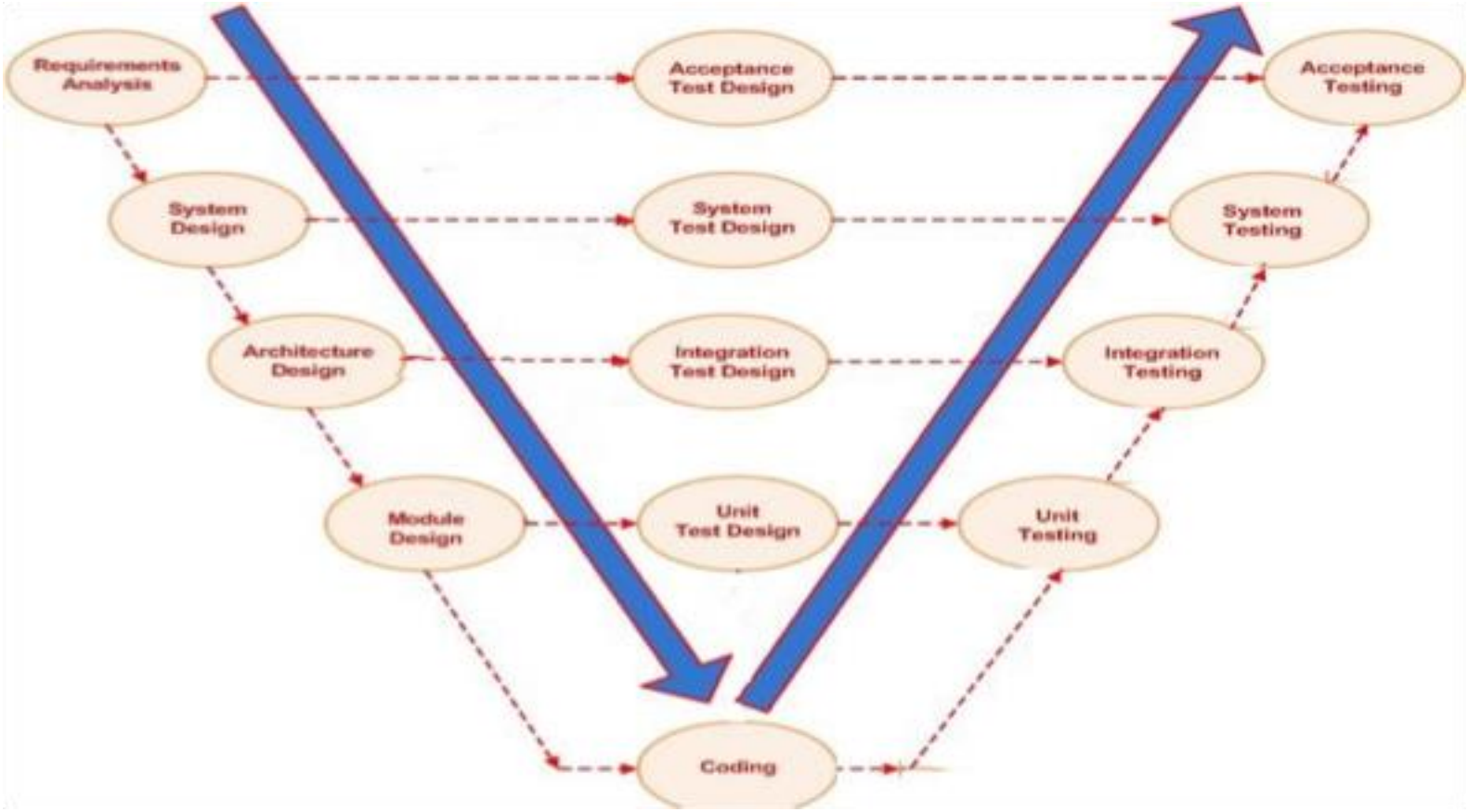
- Il vero punto debole è che manca l'interazione col cliente che vede solo il prodotto finito, alla fine del processo
  - Spesso c'è una sostanziale differenza tra come il cliente immagina un prodotto e come il prodotto viene realizzato
  - A quel punto, se ci sono discrepanze (e ci sono!) tutto il processo deve essere ripetuto.
- Un ulteriore punto critico è l'eccessiva produzione di documenti:
  - Nessuna fase è completa finché anche il documento per quella fase non è stato terminato e approvato dal gruppo di valutatori (Software Quality Assurance)
- Ricordiamo quindi il modello a cascata per il valore che ha avuto storicamente:
  - Definire e strutturare le fasi di un processo sw

Diamo a Royce quel che è di Royce

- Il modello a cascata è a tutti noto come quello visto, rigidamente "in caduta" (tutta la letteratura riporta quel modello)
- Royce in realtà già nell'articolo del 1970, evidenzia le limitazioni della cascata e propone come un modello con feedback loops da un fase alla precedente.



Modello a V
Hughes Aircraft ~1982 (sequenziale)



Le frecce blu rappresentano il tempo, quelle tratteggiate le dipendenze causali

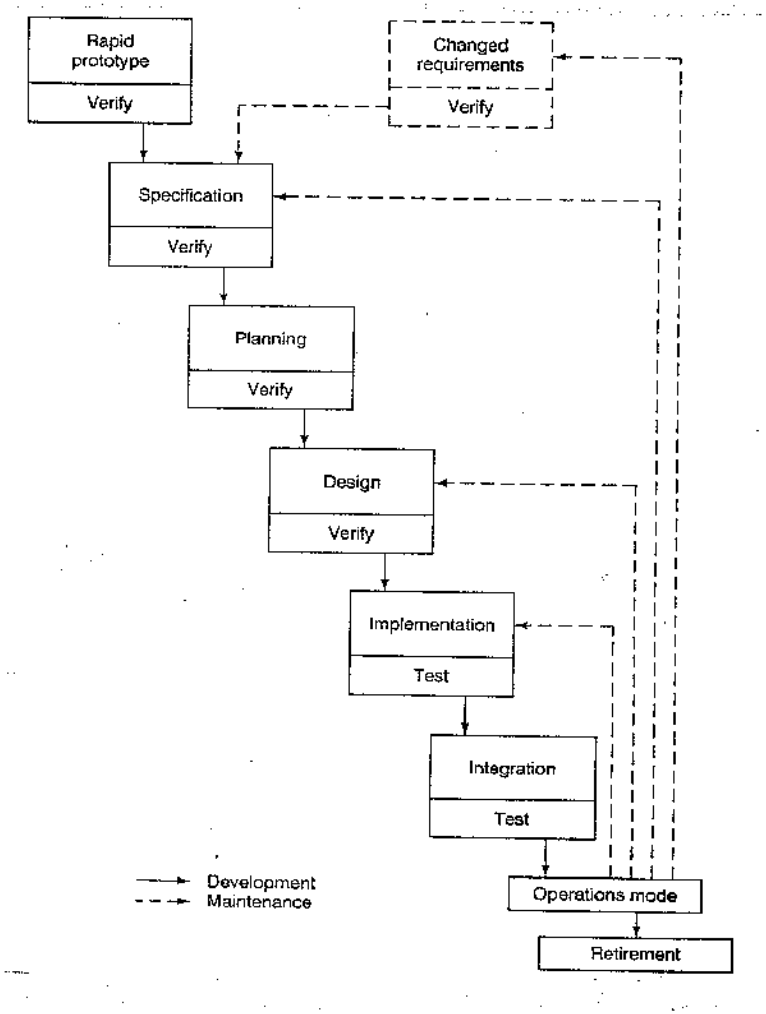
Modello a V

- L'aspetto interessante di questo modello è che evidenzia come sia possibile progettare i test durante le fasi di sviluppo (prima della codifica)

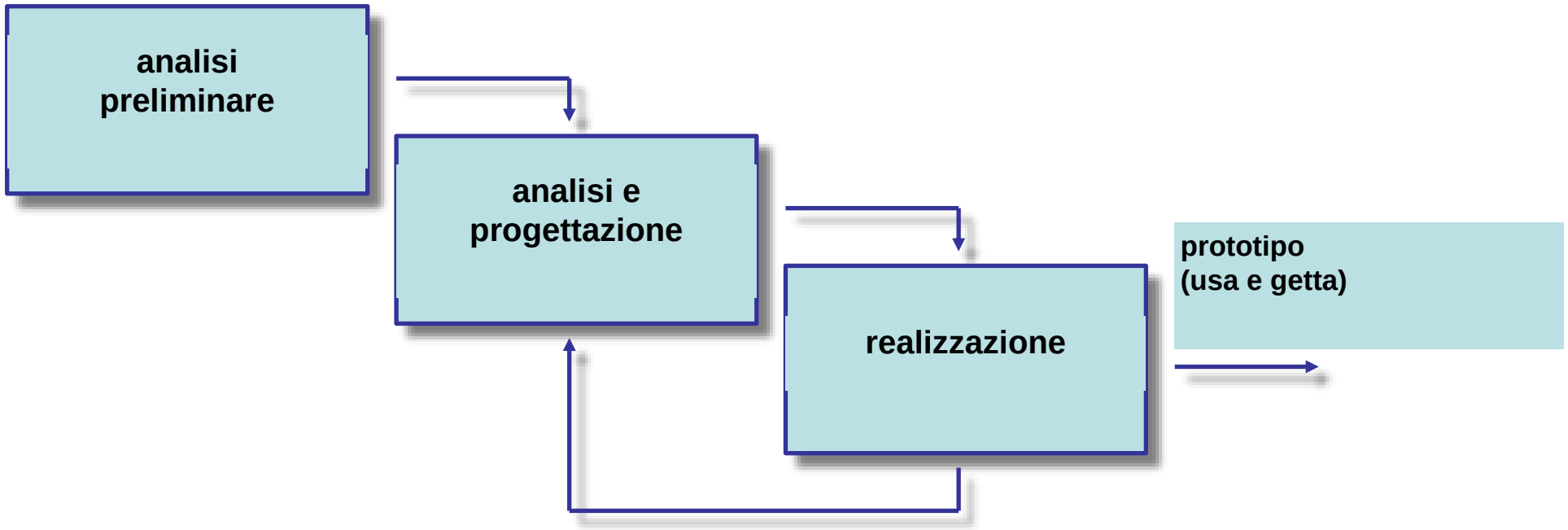
- Idea ripresa nel Test Driven Development

Rapid Prototyping

- Il primo passo è quello di costruire un prototipo velocemente per permettere al committente di sperimentarlo
- Utile quando i requisiti non sono chiari
- Il prototipo aiuta il cliente a meglio descrivere i requisiti
  - si passa a questo punto alle fase di specifica
- Anche noto come modello evolutivo



Rapid Prototyping: rappresentazione semplificata

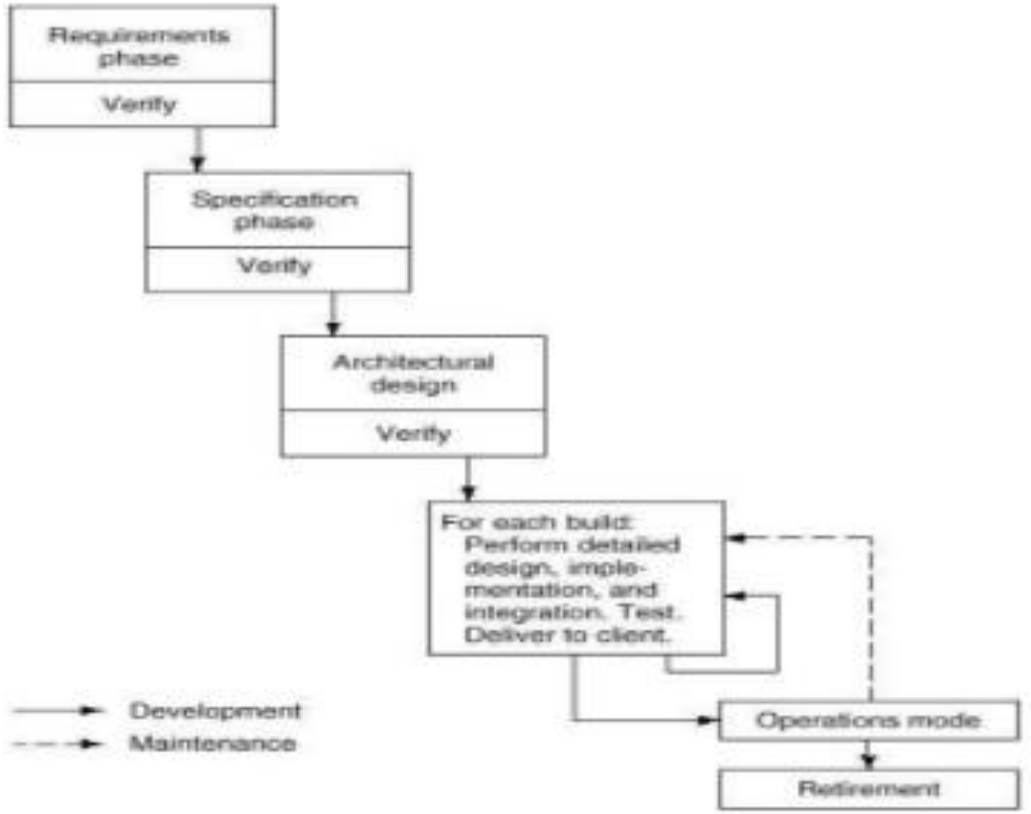


Sempre piu' iterativi

- Modello a Cascata
- Modello a V
- RAP
- Incrementale
- Modello a spirale

Modello incrementale

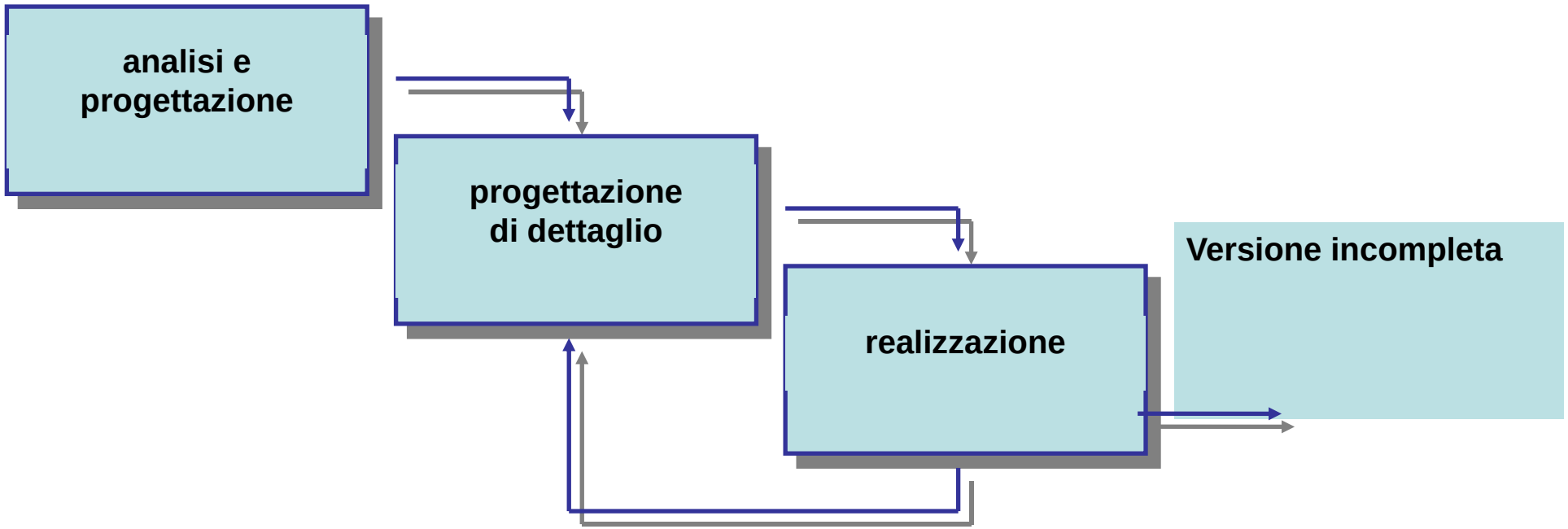
- Il sistema è costruito aggiungendo qualcosa a quello che già era stato fatto
  - I requisiti e il progetto sono definiti inizialmente, poi
  - Il sistema è implementato, integrato e testato con una serie di passaggi incrementali



Modello incrementale

- Si applica in caso di requisiti stabili, serve a
  - Ritardare la realizzazione delle componenti che dipendono criticamente da fattori esterni (tecnologie, hardware sperimentale, ecc)
  - "uscire" velocemente con qualcosa
- Se non progettato bene diventa un Build-and-Fix

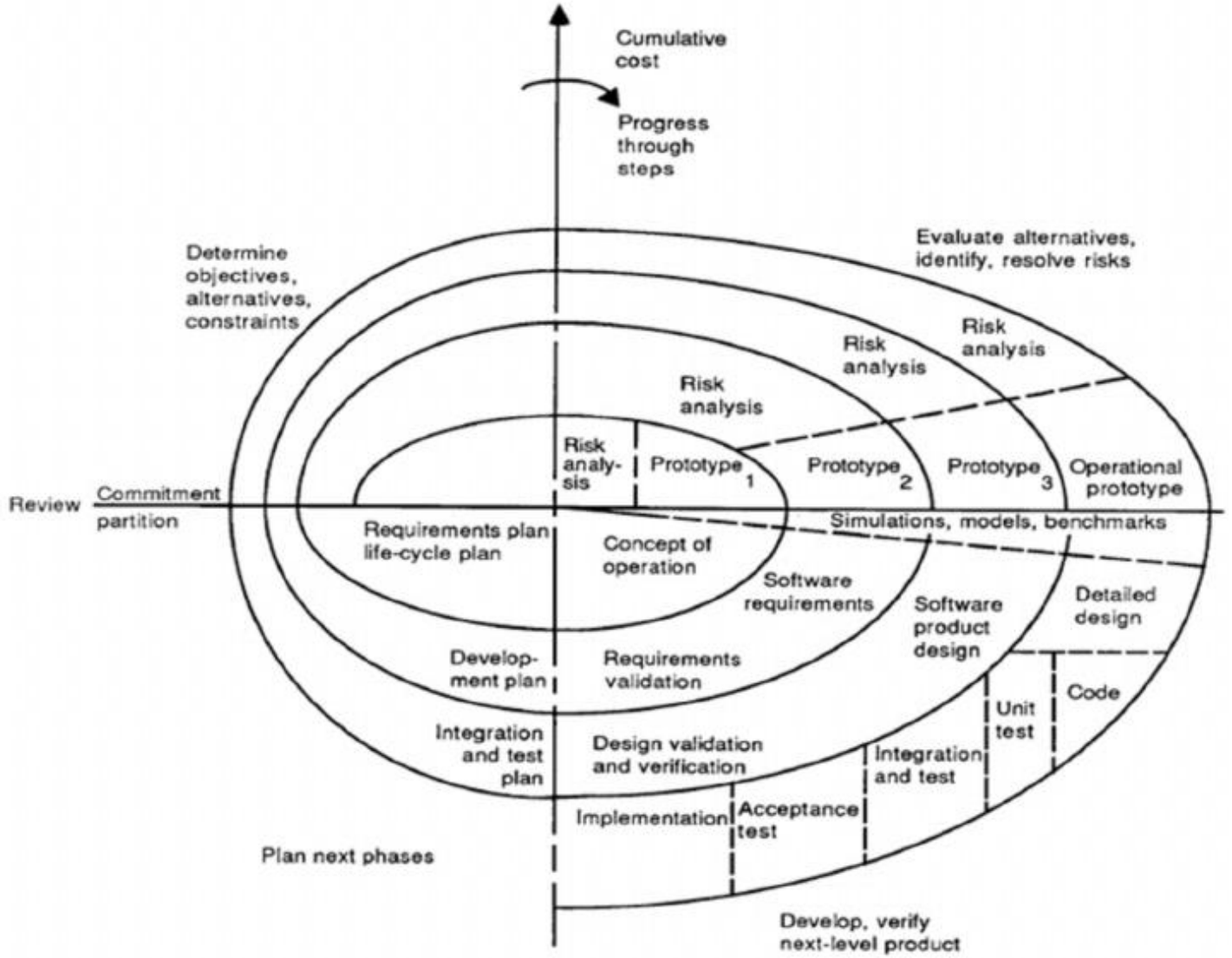
Modelo incrementale: rappresentazione semplificata



Il modello a spirale

- Proposto da Boehm nel 1988
- Iterativo, ogni iterazione è organizzata in 4 fasi:
  - Definizione degli obiettivi
  - Analisi dei rischi
  - Sviluppo e validazione
  - Pianificazione del nuovo ciclo
- È un modello astratto
  - Va specializzato per dire cosa fare in concreto in ogni iterazione e in ogni sua fase

Il modello a spirale: rappresentazione originale di Bohem



Il modello a spirale

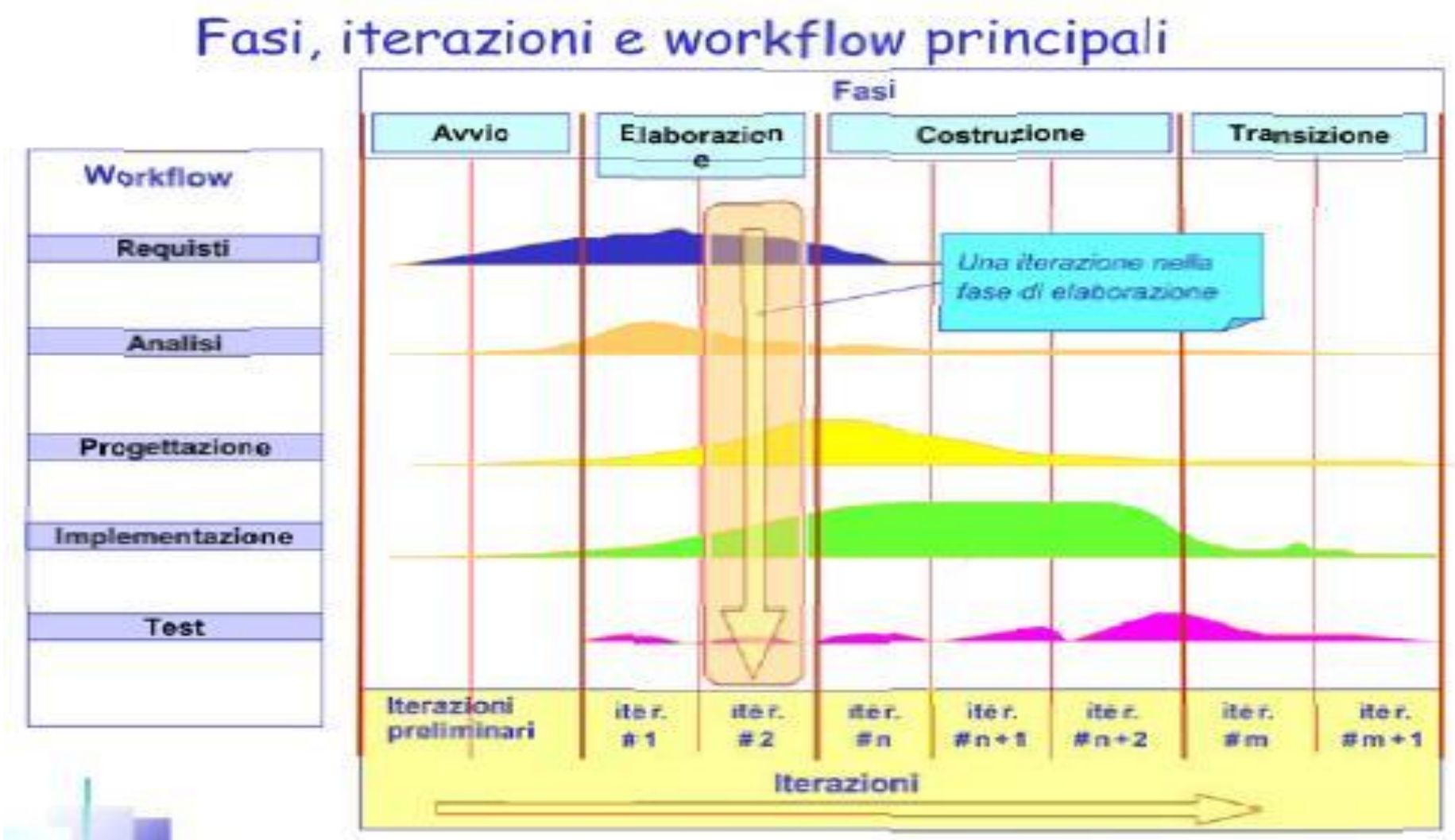
- Evidenzia gli aspetti gestionali
  - Pianificazione delle fasi
  - Centrato sull'analisi dei rischi (modello "risk driven")
  - Ispirato dal plan-do-check-act cycle [William Edwards Deming 1950]

- Tipici rischi:
  - Dominio poco noto, Linguaggi o strumenti nuovi, Personale non addestrato
- Applicabile ai cicli tradizionali
- Prevede maggior comunicazione e confronto con il committente

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

Schema di un ciclo



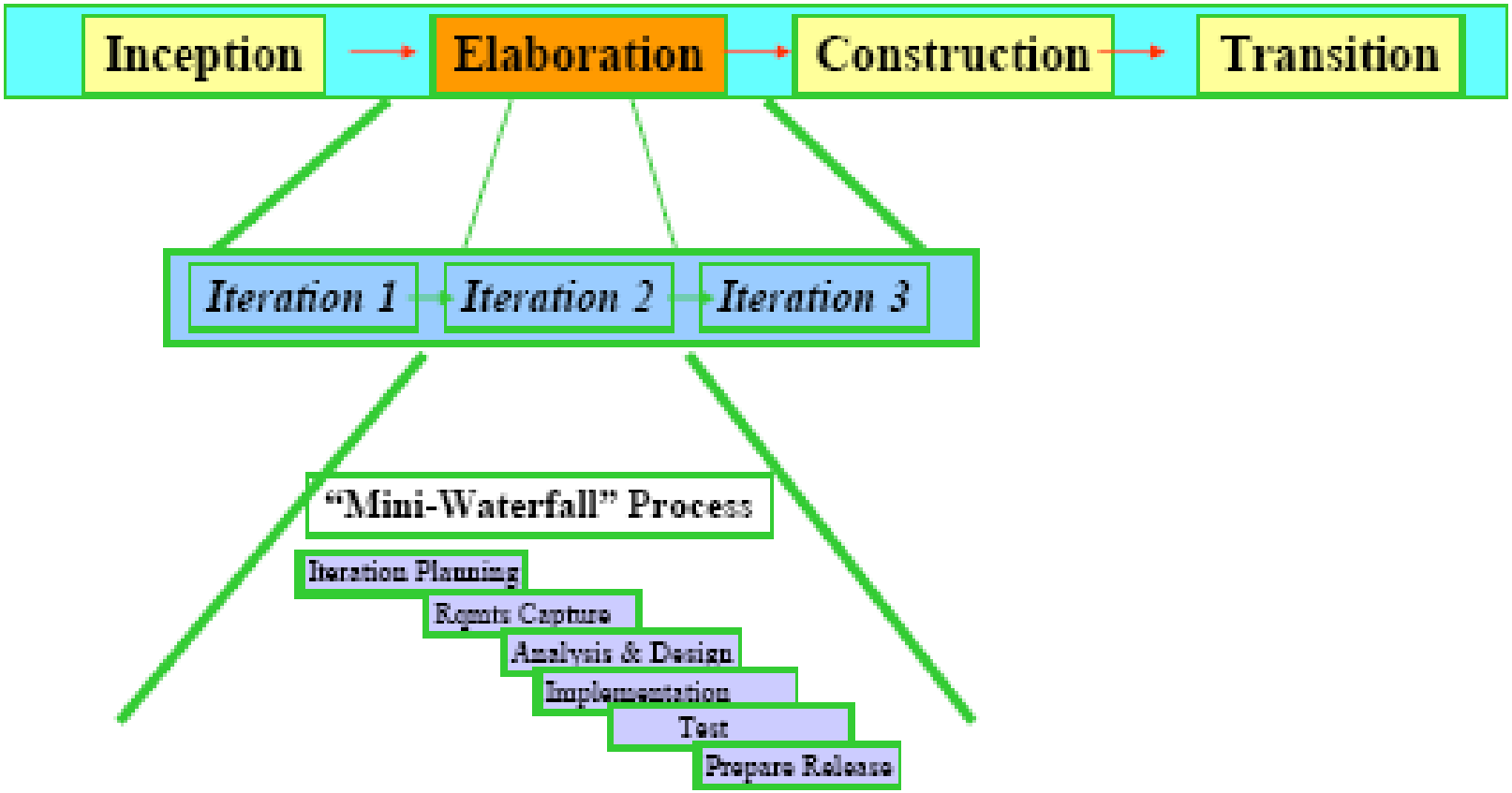
The Unified Process

- Consiste in una serie di cicli che hanno tutti la precedente forma
- Ogni ciclo si conclude con una release del sistema
- Ogni ciclo ha diverse iterazioni
- Le iterazioni sono raggruppate in 4 fasi
- Ogni fase finisce con la decisione del manager di continuare o terminare il progetto

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

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 svilupatori
- Una metodologia agile si basa sui **principi** del **Manifesto di Snowbird, feb 2001**

Agile manifesto (aka di Snowbird)

- **Comunicazione**:
  - le persone e le interazioni sono più importanti di processi e strumenti
  - tutti possono parlare con tutti, e.g. l'ultimo dei programmatori con il cliente;
  - la comunicazione tra gli attori di un progetto sw è la miglior risorsa del progetto;
  - 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**
  - rilasciare nuove versioni del software ad intervalli frequenti
  - sin dal primo giorno si testa il codice
- **Coraggio**
  - dare in uso il sistema il prima possibile e implementare i cambiamenti richiesti man mano
  - 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*

- Prassi di XP (continua)
  - Progetti semplici
    - comprensibili a tutti
  - Verifica (testing)
    - di unità e di sistema (basati sugli scenari)
    - supporto automatico
  - Test Driven Development
    - casi di test prima del codice
  - Cliente sempre a disposizione (circa ogni settimana)

eXtreme Programming
Esempio di processo *agile*

- Prassi di XP (continua)
  - 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*

- Prassi di XP (continua)
  - 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 (mischia).
- SCRUM: un processo in cui un insieme di persone si muove all'unisono per raggiungere un obiettivo predeterminato, tale obiettivo garantisce la soddisfazione delle ambizioni di squadra e delle ambizioni personali.
- 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

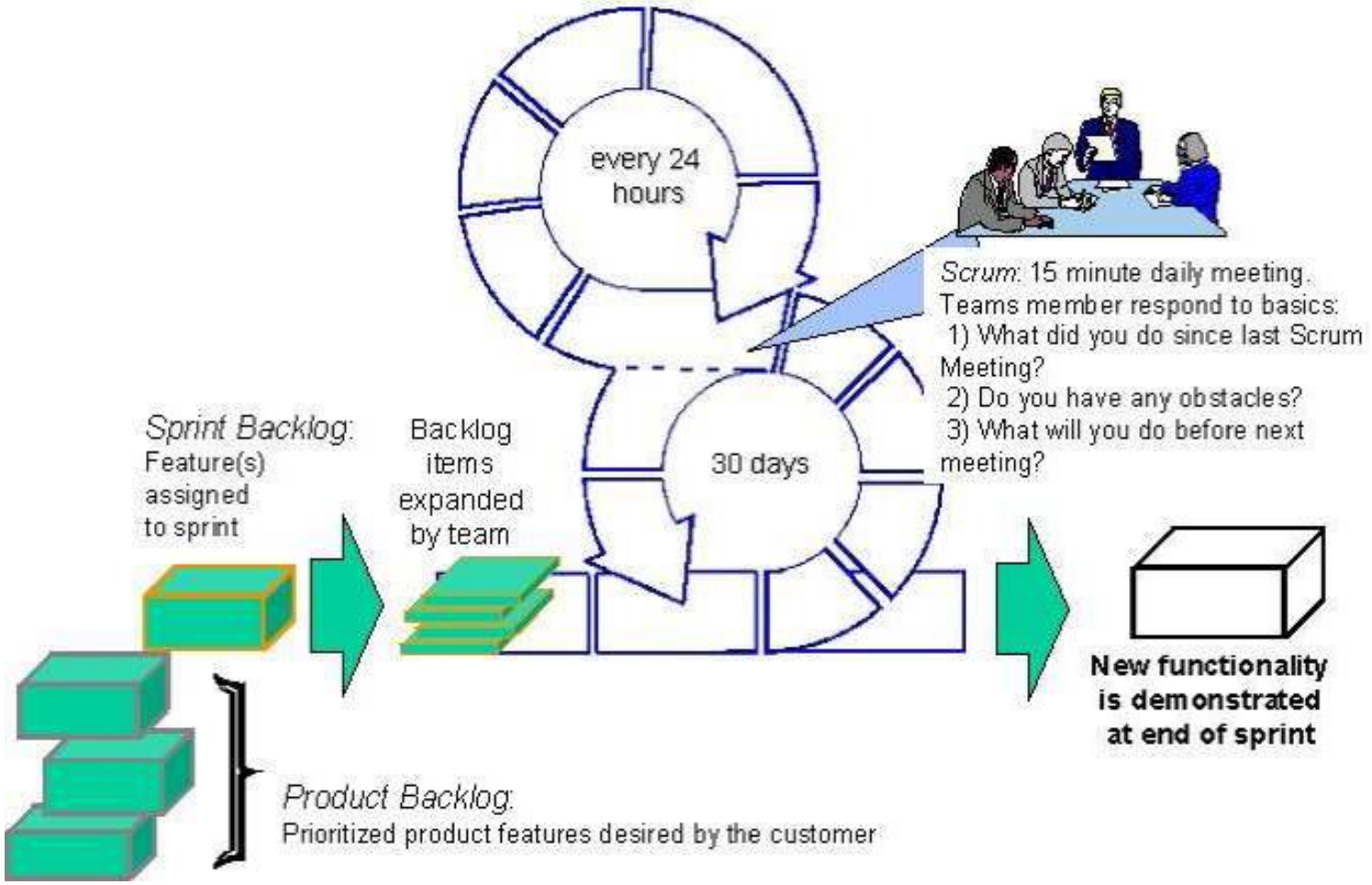
SCRUM, tre fasi: fase 1

- **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 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

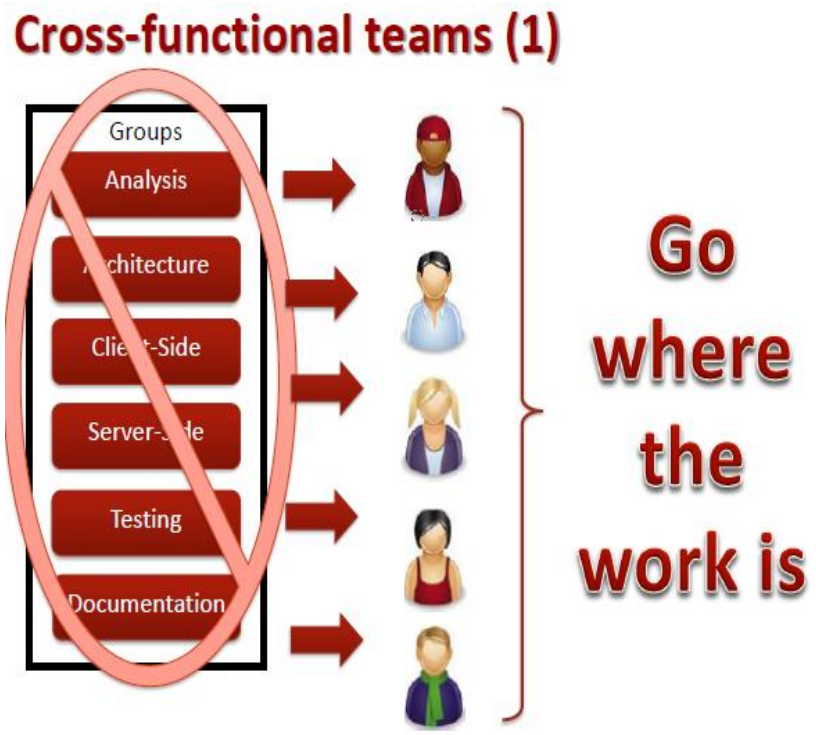
- **Post-game phase (contiene la chiusura definitiva della release)**

Tre ruoli:
Product Owner

- Tale ruolo è occupato da quella persona a cui fanno riferimento tutti i soggetti interessati al progetto, compreso il cliente finale.
- Figura di raccordo in grado di effettuare stime, aggiustare i processi che presentano difetti e di gestire l'intero procedimento secondo la pianificazione inizialmente fatta.
- Poteri :
  - Accettare o rigettare i risultati di un lavoro
  - Terminare uno sprint se necessario

Tre ruoli:
Membro del team

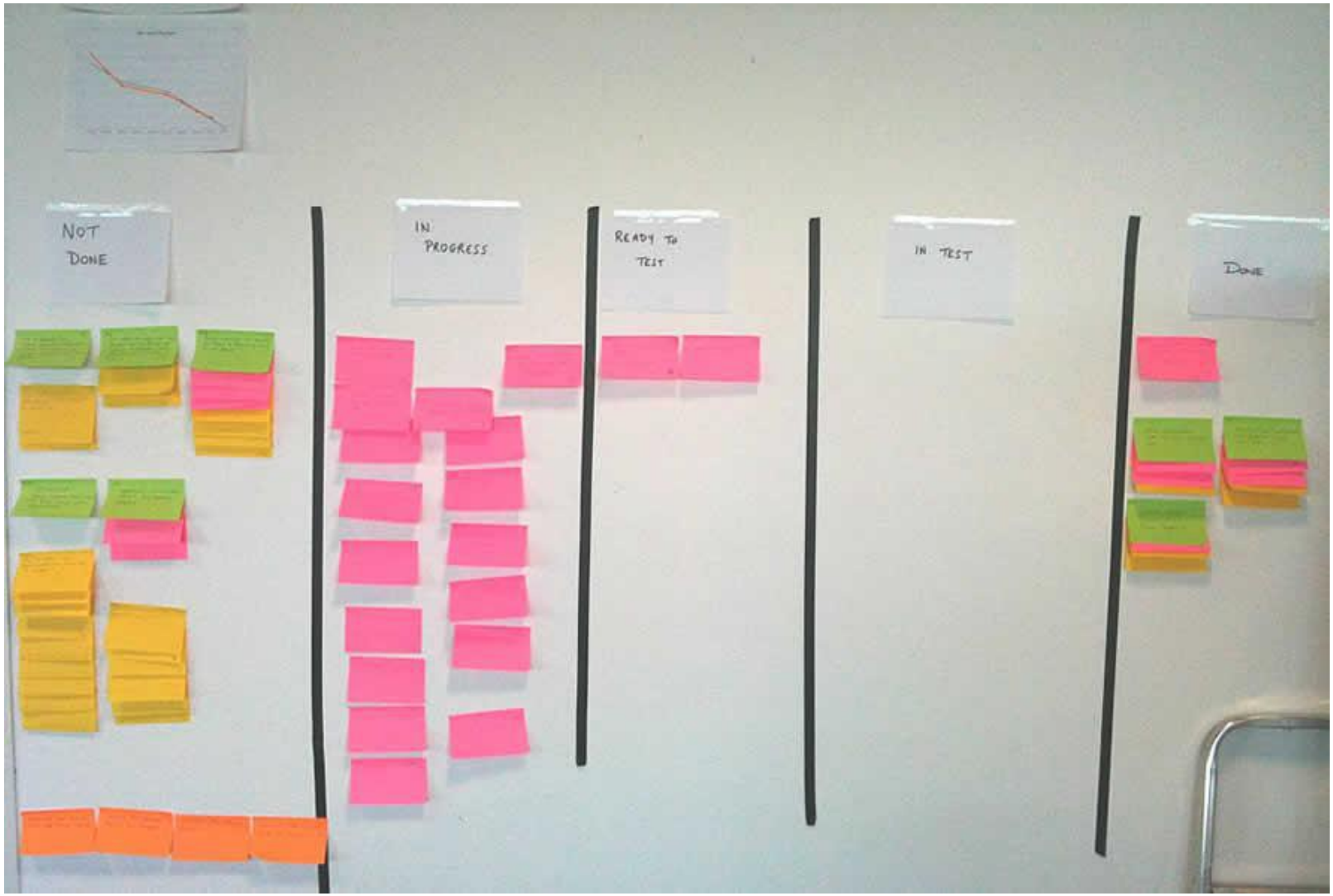
- Responsibilità:
  - Costruiscono il prodotto
  - Decidono cosa fare in ciascuno Sprint
- Caratteristiche:
  - Cross-functional (l'eccessiva specializzazione rischia di avere persone cariche di lavoro e altre che aspettano)
  - Team organizzati indipendentemente
  - Senza project (or team) manager
  - Ognuno realizza una cosa alla volta (no multi-tasking)
- Team:
  - 7 + 2 persone
  - Se è possibile nella stessa sede/ufficio



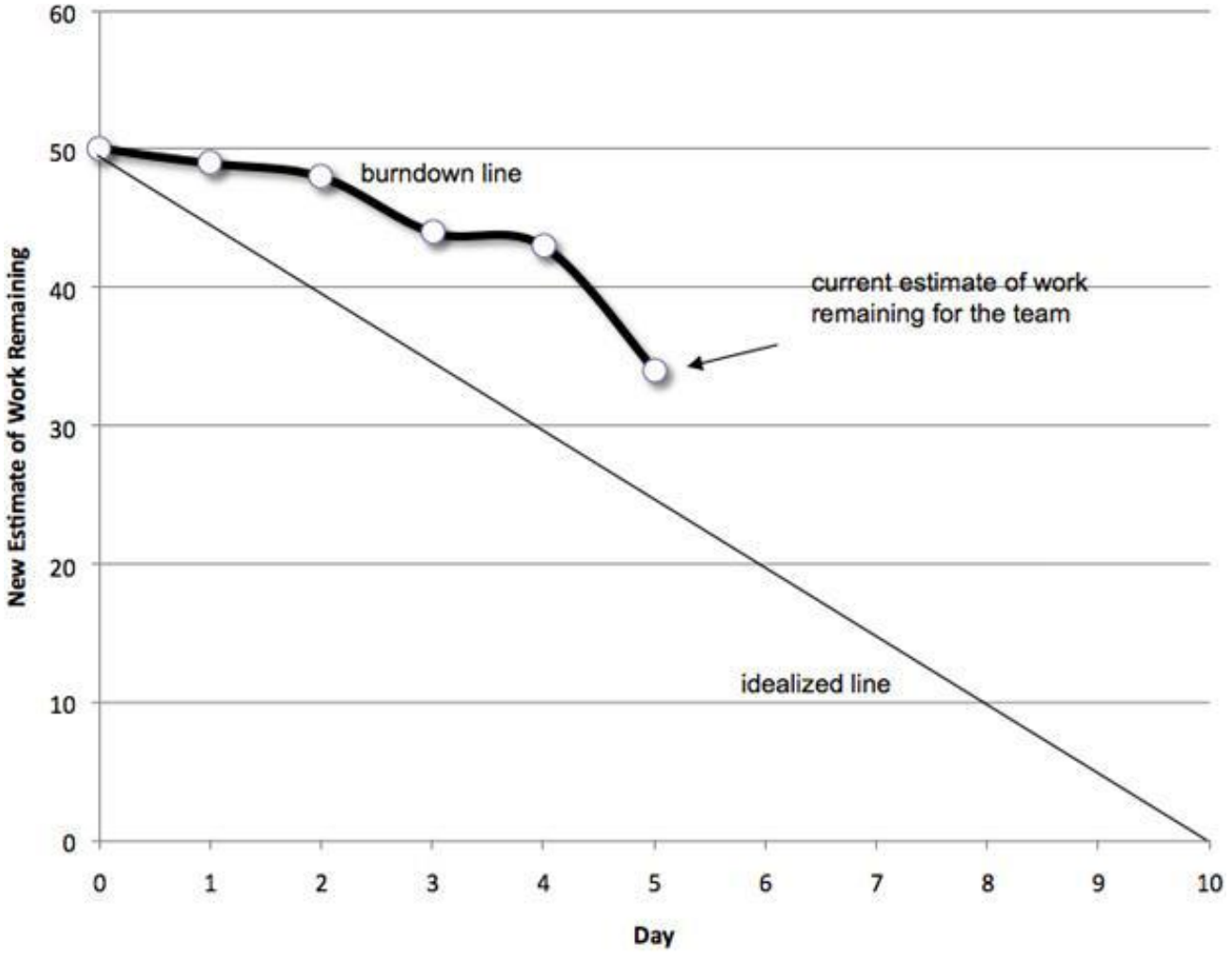
Tre ruoli:
Scrum Master

- Tale figura si occupa di supportare il team garantendo le condizioni ambientali e le motivazioni necessarie ad eseguire al meglio il lavoro commissionato.
- Non ha autorità sul team

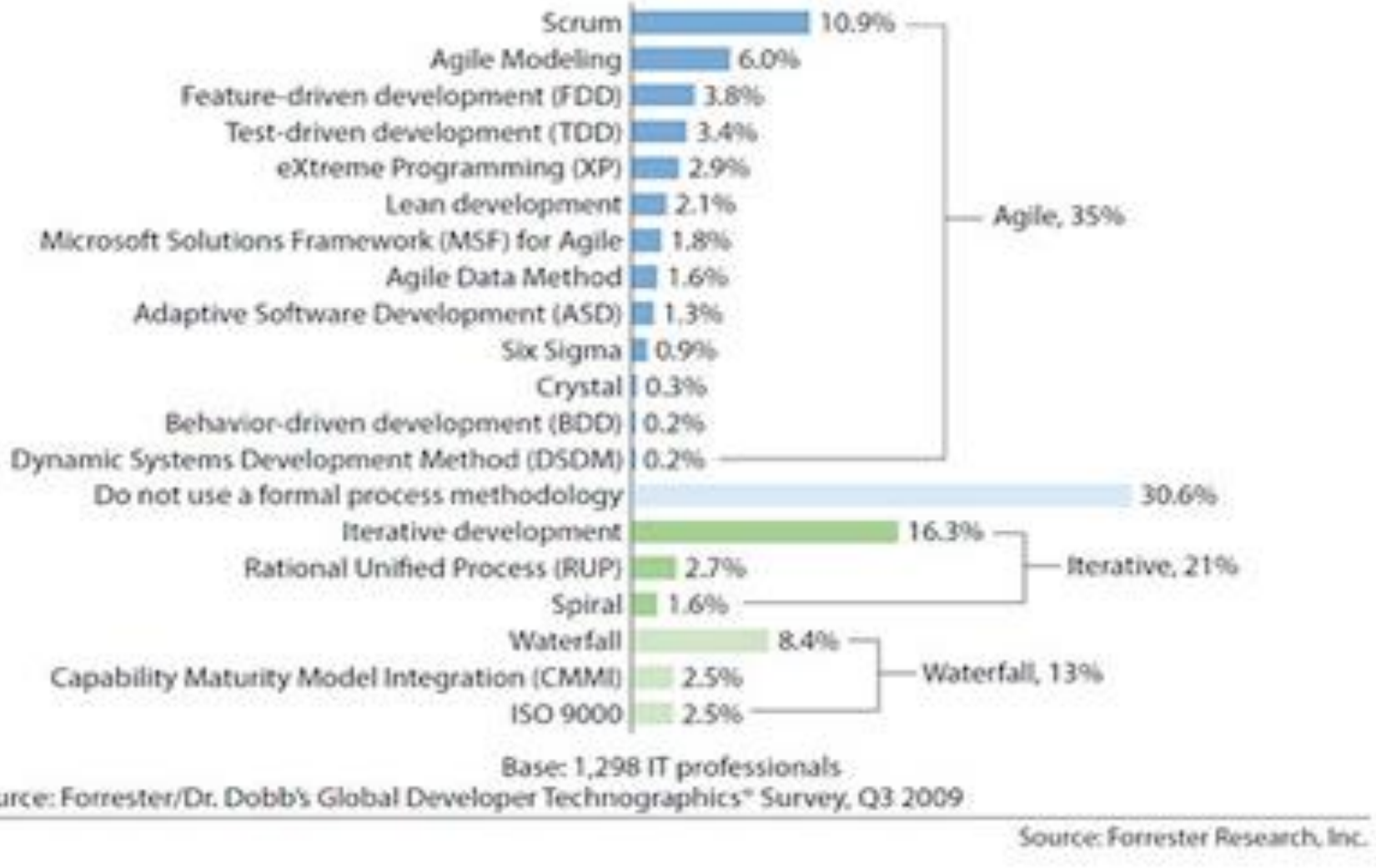
SCRUM: backlog



SCRUM: backlog



Una rassegna



Riassunto: nuove idee introdotte dai modelli di ciclo di vita

- Build-and-Fix: un non-modello → forget it
- Modelli prescrittivi
  - Cascata → definizione delle fasi, documentazione
  - Modello a V → prima idea di test driven development
  - Rapid Prototyping → iterativo con requisiti poco chiari
  - Modello incrementale → iterativo con requisiti chiari ma prodotto rilasciato "a puntate".
  - Modello a spirale → analisi dei rischi
- Unified Process → fasi vs iterazioni
- Modelli agili → "ribellione alle prescrizioni" + molte nuove idee
  - Extreme Programming
  - Scrum

Bibliografia

- **Object Oriented and Classical Software Engineering** , Stephen R.Schach, Fifth edition Cap 1,3 e 10
- **Object-Oriented Software Engineering**, David C. Kung Cap 2
- Altre letture che potrebbero interessarvi
  - Il manifesto Agile 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.