

# Modelli di processo per lo sviluppo del software: i modelli pianificati (lineari o iterativi)



*Corso di Ingegneria del Software*  
*CdL Informatica Università di Bologna*

# Obiettivi di questa lezione

- Cos'è un **processo di sviluppo del software**
- Cos'è un **modello** di processo software
- Modelli **lineari**
- Modelli **iterativi**

Nella prossima:

- Modelli **agili**

Successivamente:

- Modelli di processo orientati alla **qualità**
- Modelli **open source**

# Costruire software



How the customer explained it



How the Project Leader understood it



How the Analyst designed it



How the Programmer wrote it



How the Business Consultant described it



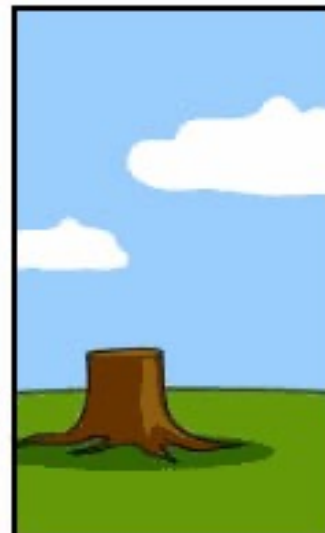
How the project was documented



What operations installed



How the customer was billed



How it was supported



What the customer really needed

# Le fasi “naturali” dello sviluppo software

- Entusiasmo
- Disillusione
- Panico
- Ricerca del colpevole
- Punizione dell'innocente
- Lodi e onori a chi non si è fatto coinvolgere

Non tutti gli sviluppi si svolgono così: in quelli di successo la differenza la fanno sempre **le persone** e le regole (il **processo di sviluppo**) che seguono

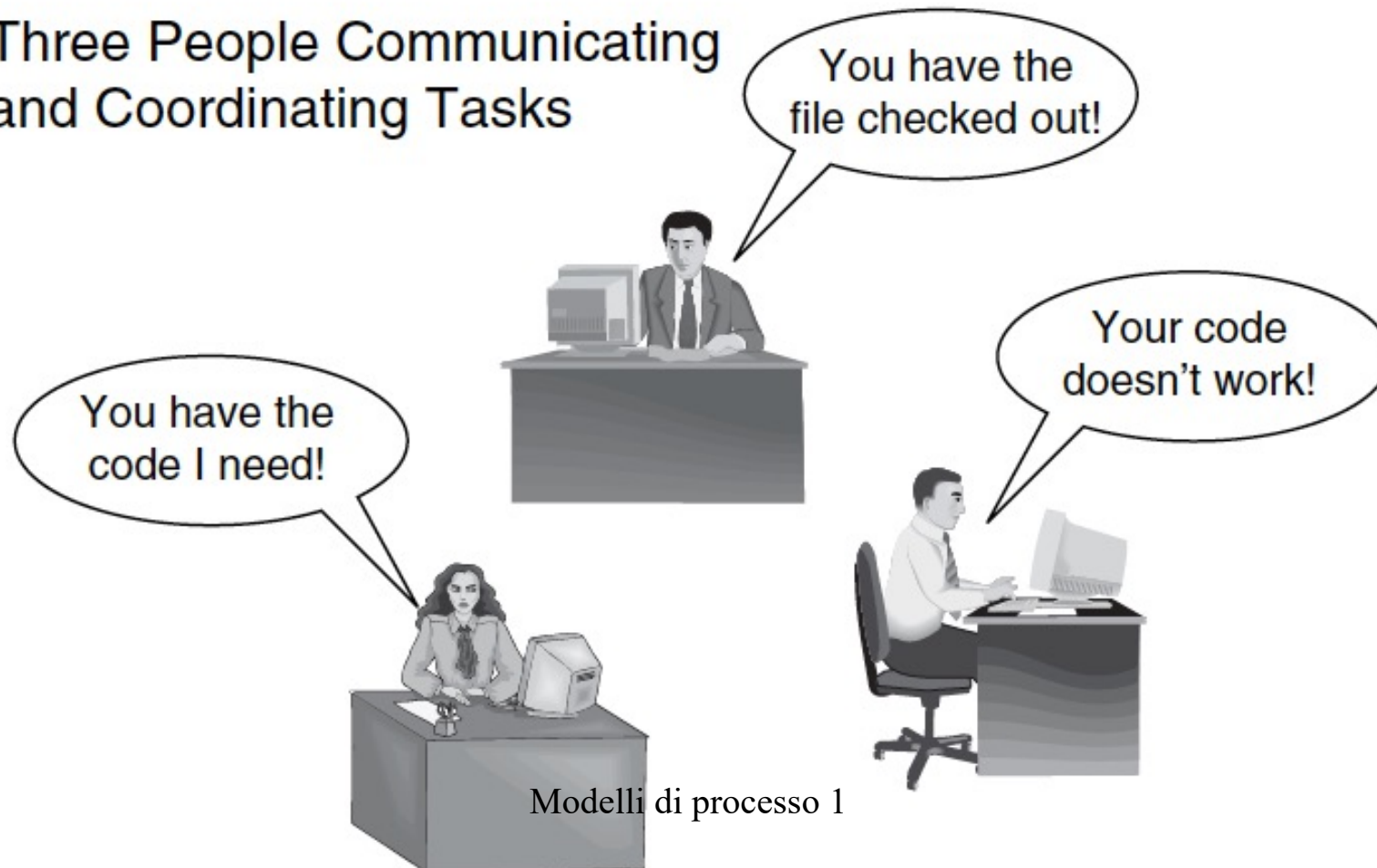
# Ciclo di vita e processo di sviluppo

- Il **processo di sviluppo** è una parte del ciclo di vita del software
- Il **ciclo di vita del software** (*software lifecycle*), designa le varie fasi della vita di un software, dalla sua concezione al suo ritiro
- Di solito il processo di sviluppo inizia dalla concezione e finisce col rilascio finale e il successivo deployment

One Person  
Very Little Coordination  
and Communication  
Overhead



Three People Communicating  
and Coordinating Tasks



Modelli di processo 1

# Il team include più ruoli

- Lo sviluppo in team è molto diverso dallo sviluppo “personale”
- Nel team ci sono persone con esperienze diverse, che ricoprono diversi **ruoli che hanno diverse abilità**:
  - Come progettare il prodotto software (**architetti**)
  - Come costruire il prodotto sw (**programmatori**)
  - A cosa serve il prodotto sw (**esperti di dominio**)
  - Come va fatta l'interfaccia utente (**progettisti di interfaccia**)
  - Come va controllata la qualità del prodotto sw (**testatori**)
  - Come usare le risorse di progetto (**project manager**)
  - Come riusare il software esistente (**gestori delle configurazioni**)

# Discussione

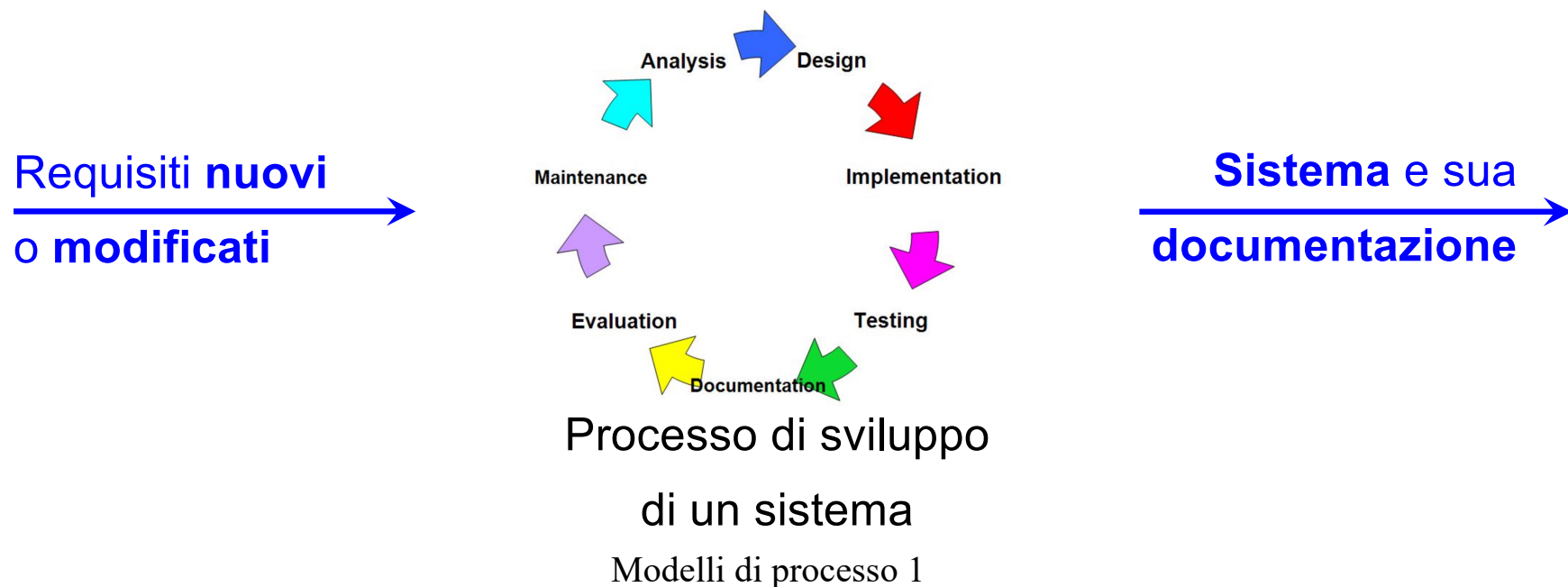
- Se dovete creare un sistema di 100 KLOC,
  - Quante *persone* occorrono? Per quanto tempo?
  - Cosa debbono fare?
  - Come le organizzate?
  - Quali documenti debbono produrre? Quando?





# Cos' è un processo di sviluppo

Un processo di sviluppo definisce **Chi** fa **Cosa**, **Quando**, e **Come**, allo scopo di conseguire un certo risultato



# Perché studiare il processo di sviluppo del sw?

- I sistemi software che costruiamo devono risultare affidabili e sicuri: il processo di sviluppo del software influenza tali **qualità**
- Esistono parecchi modelli di processi software, adatti a prodotti, organizzazioni e mercati **diversi**
- Alcuni **strumenti** sw di sviluppo sono efficaci solo nell'ambito di processi specifici
- Il processo di sviluppo del software impatta l'organizzazione che lo sviluppa
- L'organizzazione che esegue lo sviluppo impatta la struttura del prodotto (**legge di Conway**)

# Legge di Conway

*Le organizzazioni che progettano sistemi ne progettano la struttura riproducendo le proprie strutture comunicative (es. l'organigramma)*

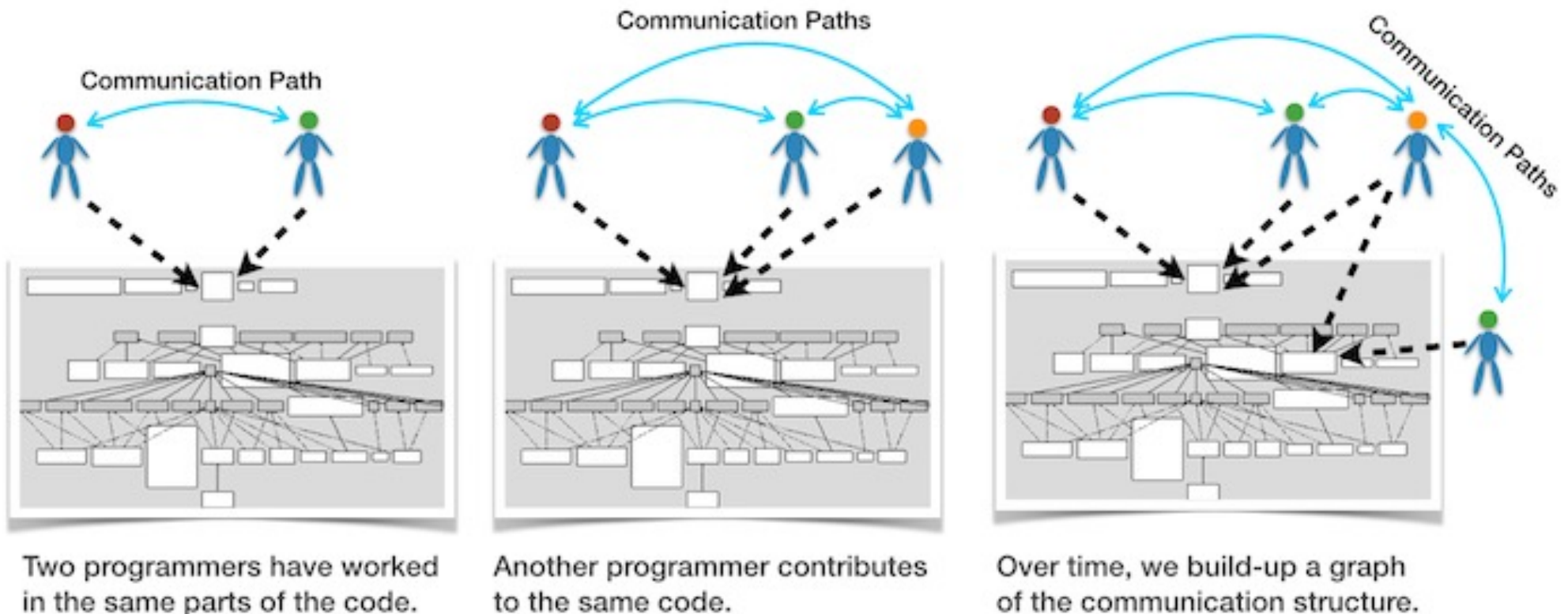
**Esempio:** se 4 team collaborano a costruire un compilatore, la struttura finale sarà su 4 processi in pipeline

**Principio:** alcune importanti proprietà di un sistema dipendono dal suo processo di costruzione

# Conseguenze della legge di Conway

La legge di Conway ha per conseguenza che gli sviluppatori che lavorano sugli stessi componenti devono poter comunicare senza ostacoli.

Ovvero, devono essere vicini da un punto di vista organizzativo

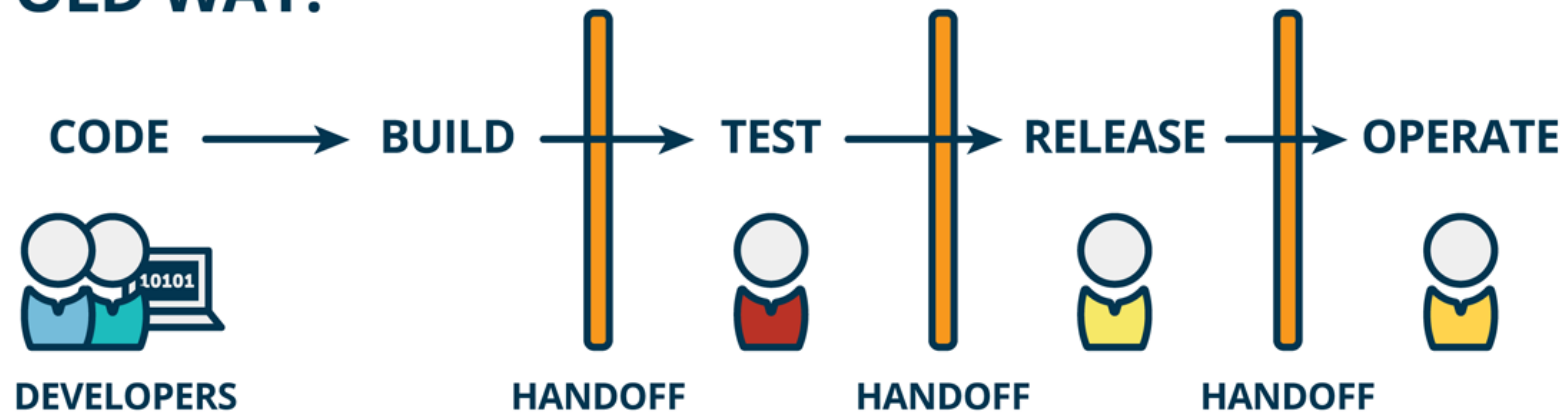


# Conseguenze della legge di Conway

- Se le parti di un'organizzazione (team, dipartimenti, divisioni) non riflettono le parti essenziali del prodotto/servizio, o se le loro relazioni non si riflettono nelle relazioni tra le parti del prodotto/servizio, allora il progetto avrà problemi
- Occorre assicurarsi che l'organizzazione di sviluppo e poi di produzione sia compatibile con l'architettura del prodotto
- Esempio: un'organizzazione che produce un portale o sito la cui struttura e contenuto rispecchiano gli interessi interni dell'organizzazione piuttosto che i bisogni degli utenti
- Morale: per innovare mediante i servizi ICT (interni o esterni) cambiate prima la vostra organizzazione, altrimenti è una battaglia persa!

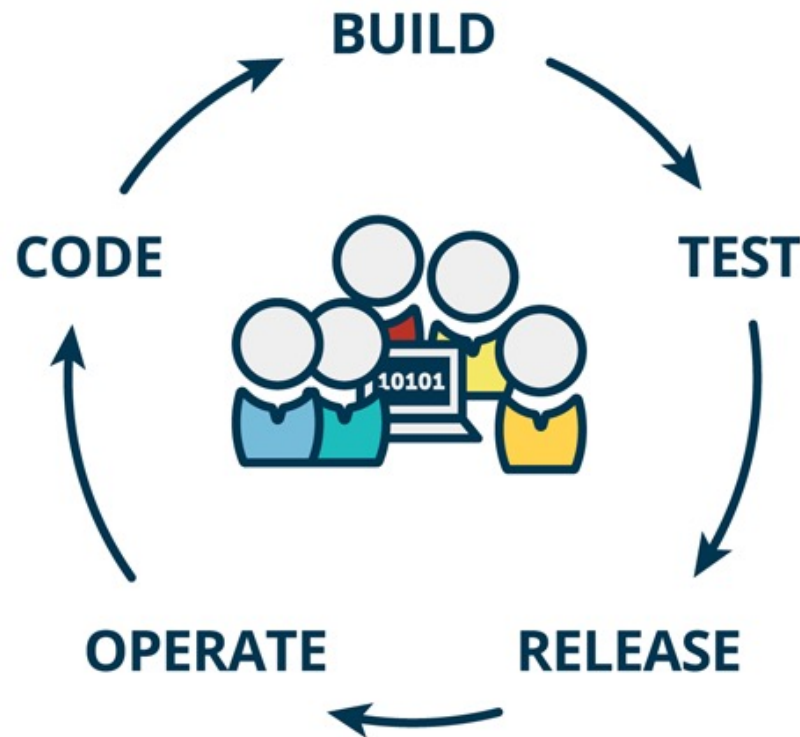
# Conseguenze della legge di Conway

## OLD WAY:



# Conseguenze della legge di Conway

**NEW WAY:**



# Il software è un costrutto sociale

- **Il software è il prodotto di un processo sociale**, che coinvolge molte persone con ruoli e interessi diversi
- Inoltre, spesso, quando un software viene messo in opera modifica i rapporti sociali tra le persone – ad esempio in una banca, o in una pubblica istituzione
- Molte qualità del software si possono giudicare solo in termini «sociali»



# Processo software

- Un processo di sviluppo del software (o “*processo software*”) è un **insieme di attività di persone** che costruiscono un **prodotto software**
- Il prodotto può essere costruito **da zero** o mediante **riuso di software esistente** (*asset*) che viene modificato o integrato

# Categorie di processi

- **Programming in-the-small:**  
uno sviluppatore, un modulo (es. edit-compile-debug)
- **Programming in-the-large:**  
uno sviluppatore, più moduli, più versioni, più configurazioni  
(es. Personal software process)
- **Programming in-the-many:**  
Più sviluppatori, coordinati mediante un **processo di sviluppo**  
(es. Scrum)

# Processo minimale: “programma e debugga”

Scrivere un programma e rimuovere gli errori

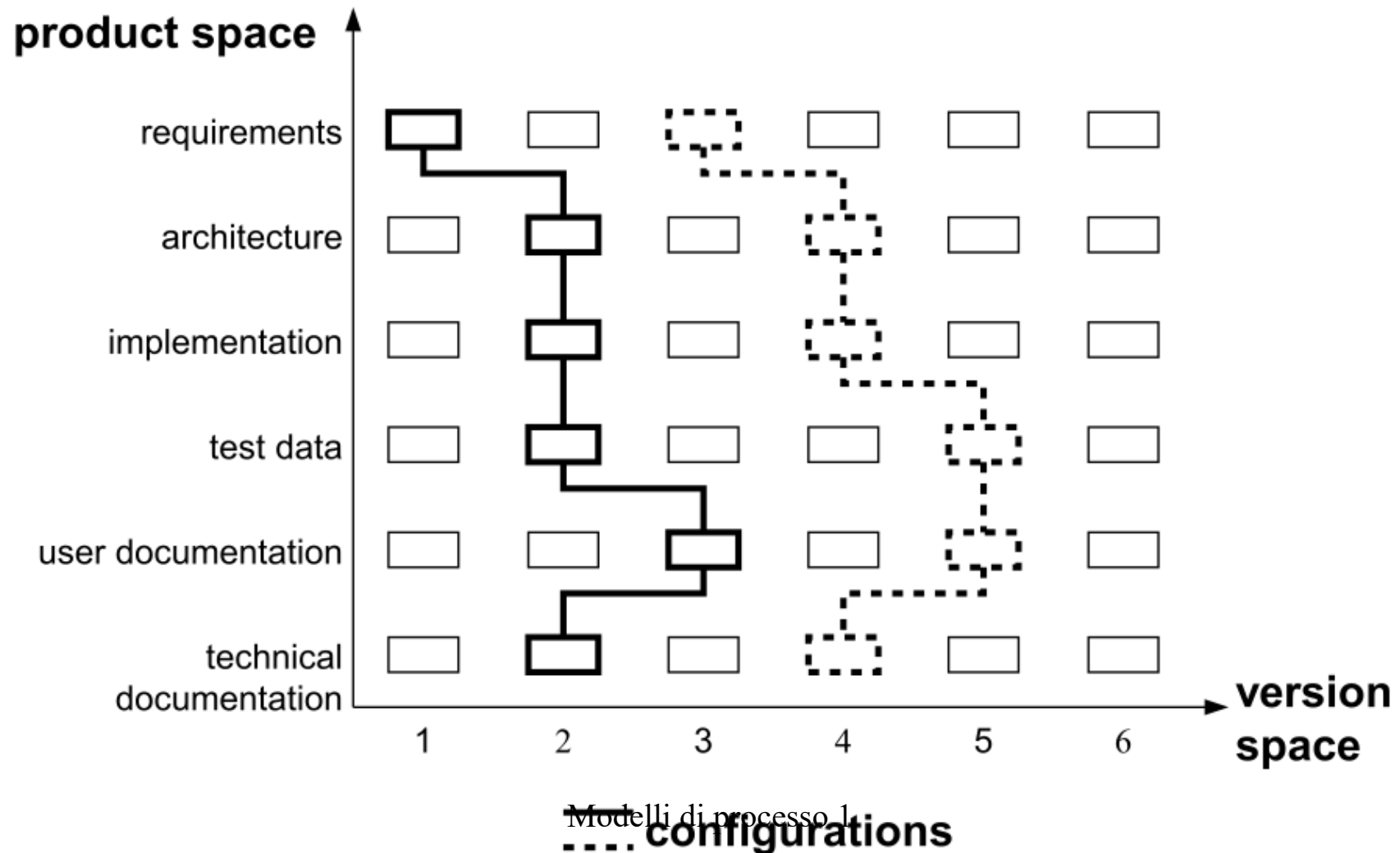
- **Programmare è un attività personale**  
di solito non si definisce una sequenza generica di passi “creativi” della programmazione
- **Il debugging è una forma di verifica**  
per scoprire e rimuovere errori nel programma

# Il processo edit-compile-debug



- **Ciclo molto veloce, feedback rapido**
- **Disponibilità di molti strumenti**
- **Specializzato per la codifica**
- **Non incoraggia la documentazione**
- **Il processo non scala: in-the-large, in-the-many**
- **Ingestibile durante la manutenzione**
- **Il debugging ha bisogno di un processo specifico**

# Programming in-the-large: moduli, versioni, configurazioni



# Programming in the many

- In una grande azienda lo sviluppo sw è spesso suddiviso tra più team
- Occorre coordinare i team, e di solito si usa una struttura sociale gerarchica
- In ciascun team il Product Owner è la figura di riferimento che definisce gli obiettivi di sviluppo del team



©Project Management Institute. All rights reserved.

<https://www.pmi.org/disciplined-agile/process/program-management/team-structure-of-a-large-program>

# Modelli di processo

- Un **processo sw** è un insieme strutturato di attività necessarie per sviluppare un sistema sw:
  - i **ruoli**, le **attività** e i **documenti** da produrre
- Un **modello di processo sw** è una **rappresentazione** di una **famiglia di processi**
  - Fornisce una descrizione da prospettive particolari
    - per catturare caratteristiche importanti dei processi sw
      - utili a diversi scopi, ad esempio per valutarli, criticarli o estenderli

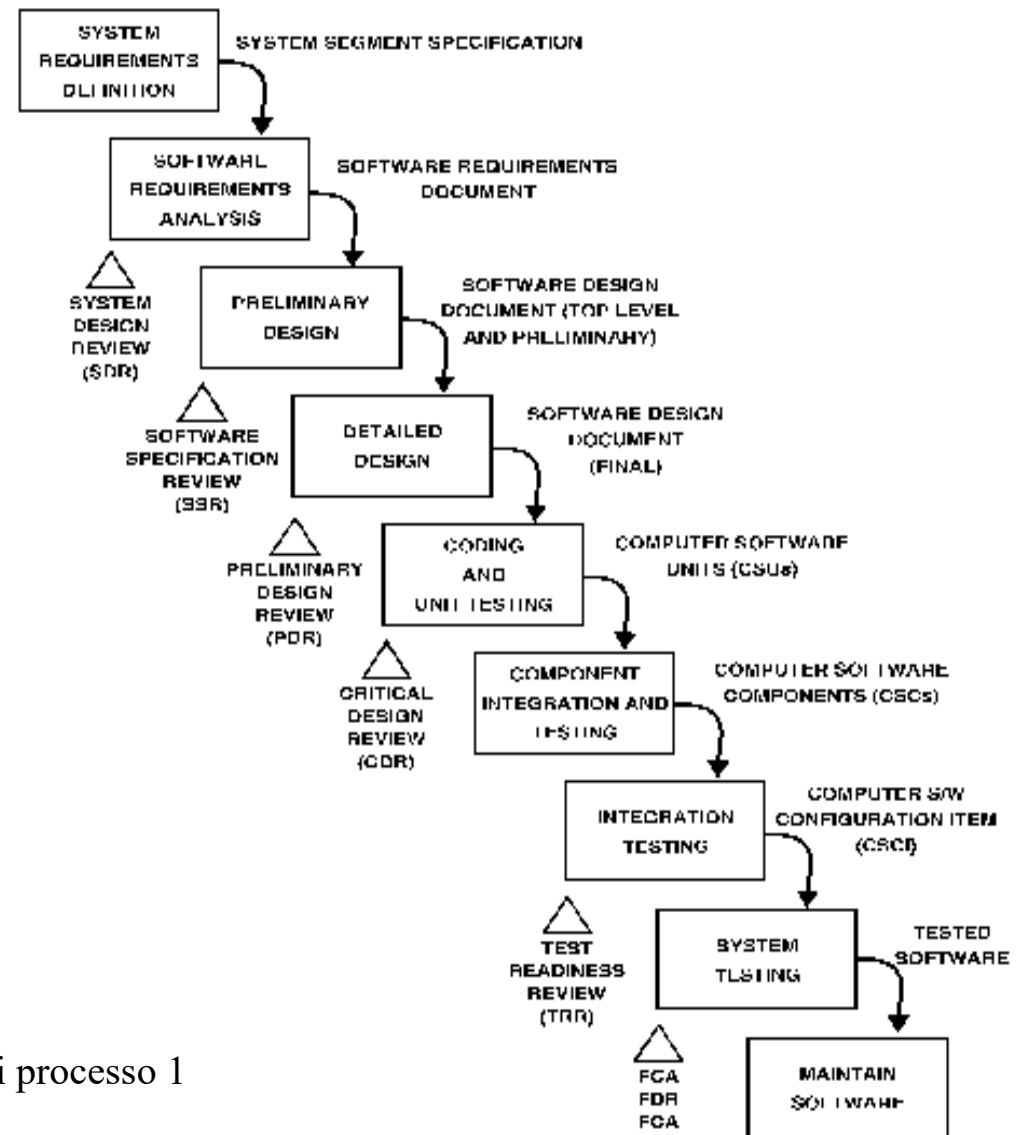
# Descrivere un processo

- Occorre descrivere/assegnare i **ruoli** (es. stakeholders)
- Occorre descrivere/monitorare le **attività**
- Occorre descrivere/assemblare gli **strumenti**
- Occorre descrivere/controllare i gli **eventi**
- Occorre descrivere/validare i **documenti** (es. requisiti)
- Occorre descrivere/verificare i **criteri di qualità**

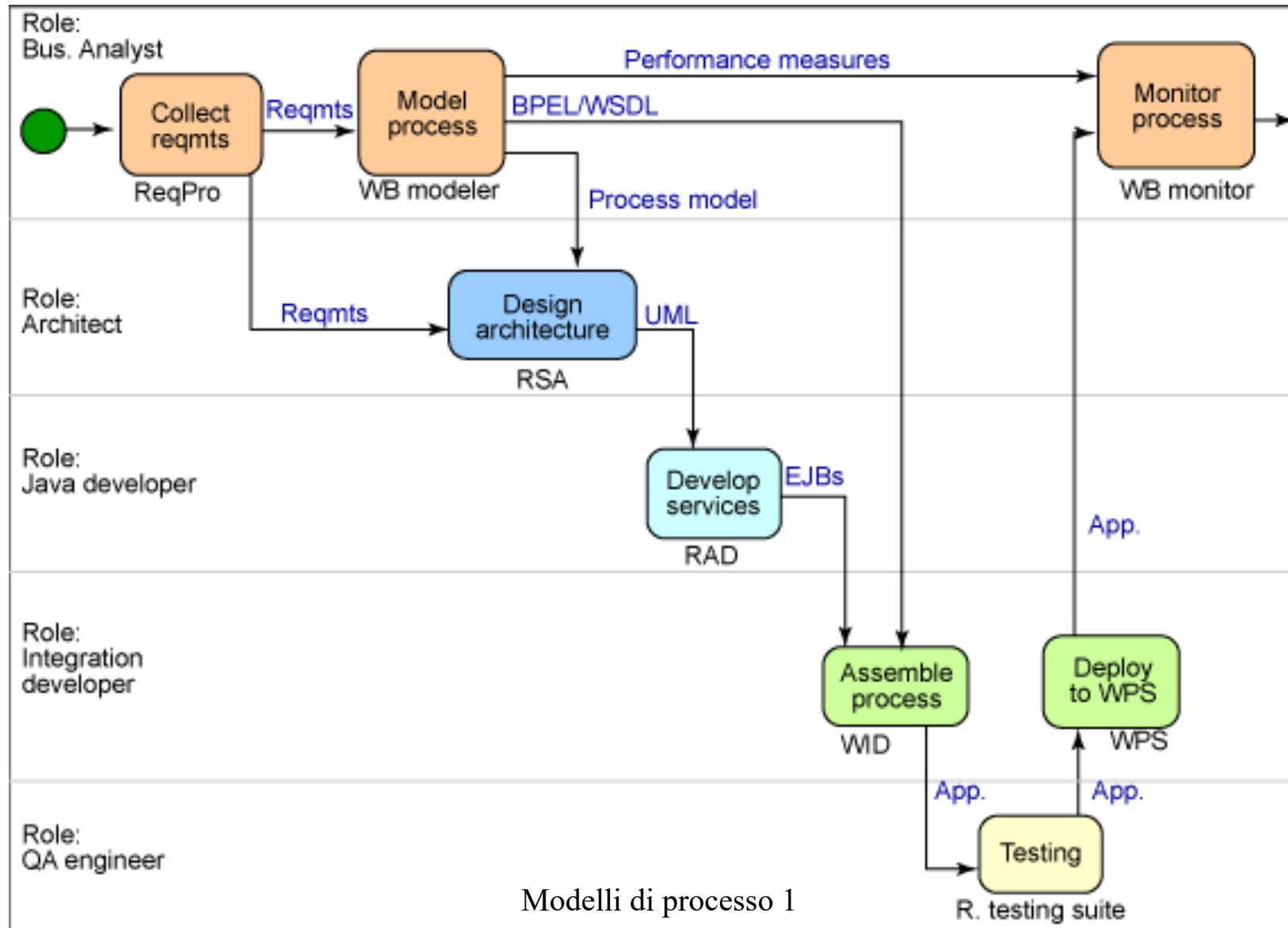


# Esempio

- **Modello di processo:** **waterfall**, cioè pianificato lineare
- **Processo:** istanza di un modello waterfall che viene “eseguita” per creare un sistema
  - crea una serie di artefatti come prescritto dal modello



# Esempio: ruoli e strumenti in un processo a cascata (IBM WebSphere)



# Perché descrivere un processo sw

## **Processo software:**

*L'insieme strutturato di attività, eventi, documenti e procedure necessari per la costruzione di un sistema software*

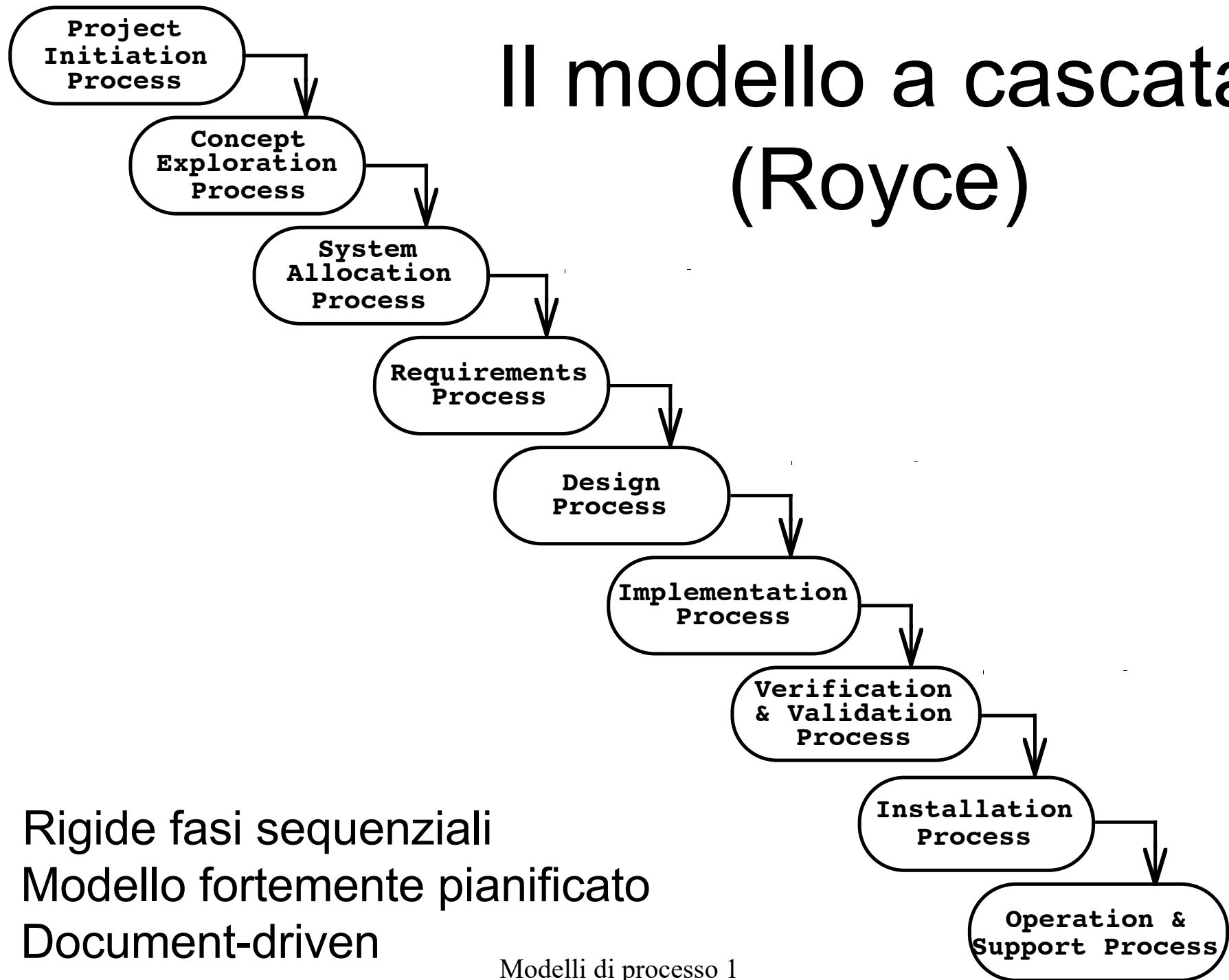
## **Benefici della modellazione dei processi sw:**

- Ciascuno sa cosa deve fare, giorno dopo giorno
- Miglior coordinamento del team
- Accumulazione di esperienza
- Aderenza agli standard internazionali
- “**Migliora il processo per migliorare il prodotto**”

# Modelli generici di processo sw

- **Modello a cascata (esempio: Waterfall)**
  - Specifica e sviluppo sono separati e distinti
- **Modello iterativo (esempio: UP)**
  - Specifica e sviluppo sono ciclici e sovrapposti
- **Modello agile**
  - Non pianificato, guidato dall'utente e dai test
- **Sviluppo formale (esempio: B method)**
  - Il modello matematico di un sistema viene trasformato in un'implementazione

# Il modello a cascata (Royce)



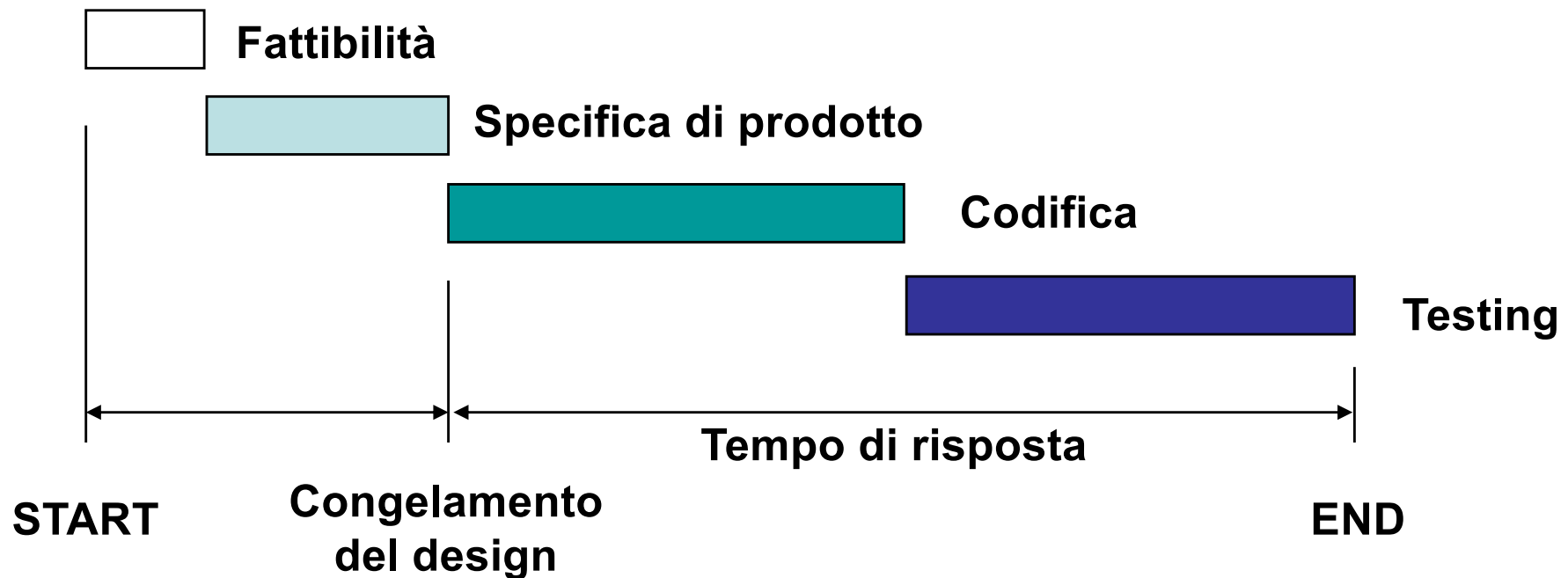
- Rigide fasi sequenziali
- Modello fortemente pianificato
- Document-driven

# Modello a cascata: aspetti positivi e negativi?

- Si adatta bene a progetti con requisiti stabili e ben definiti
- Problemi:
  - Il cliente deve sapere definire i requisiti
  - Versione funzionante del software solo alla fine
  - Difficili modifiche “in corsa”
  - Fasi fortemente collegate tra loro e bloccanti

# Problema dei processi lineari

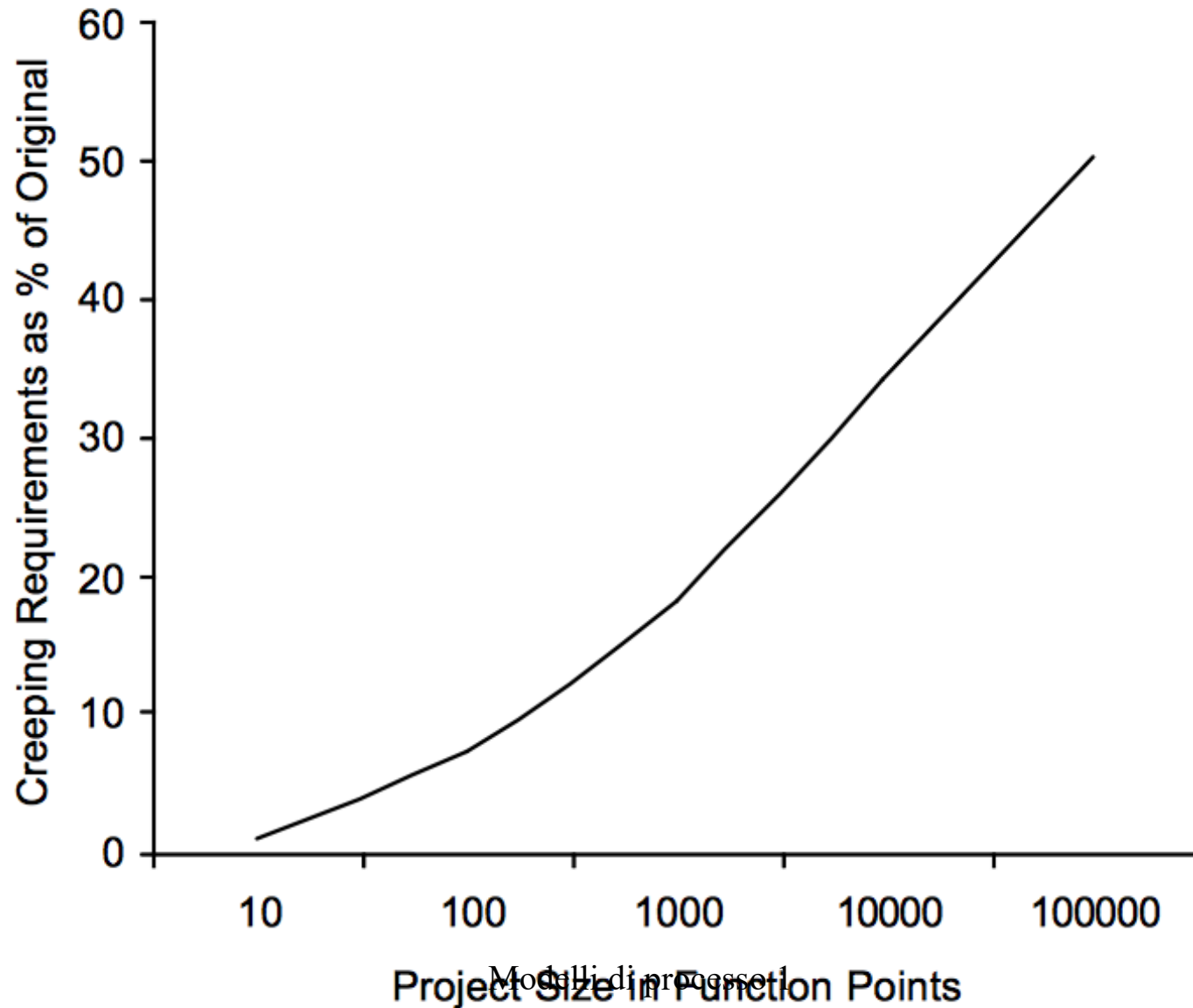
Modello di sviluppo rigidamente sequenziale



Problema: il processo non risponde ai cambiamenti di mercato che siano più rapidi della sua esecuzione

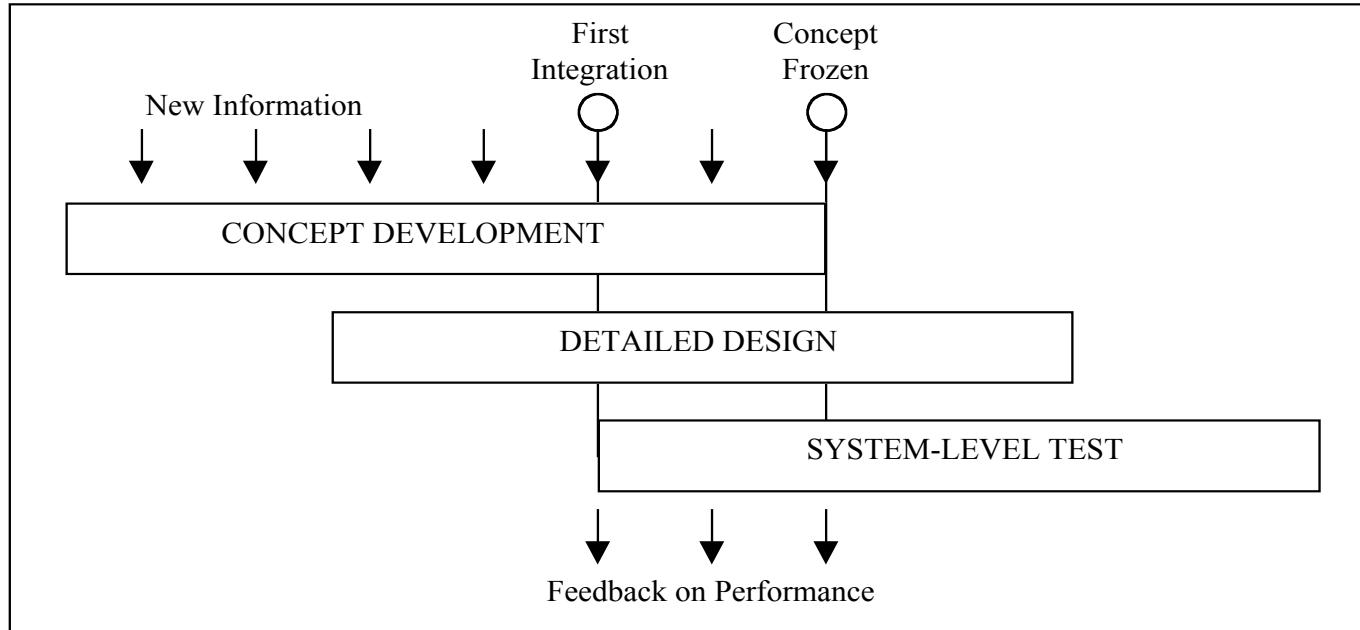
# I requisiti cambiano durante lo sviluppo

(più il sistema è grosso più cambiano!)





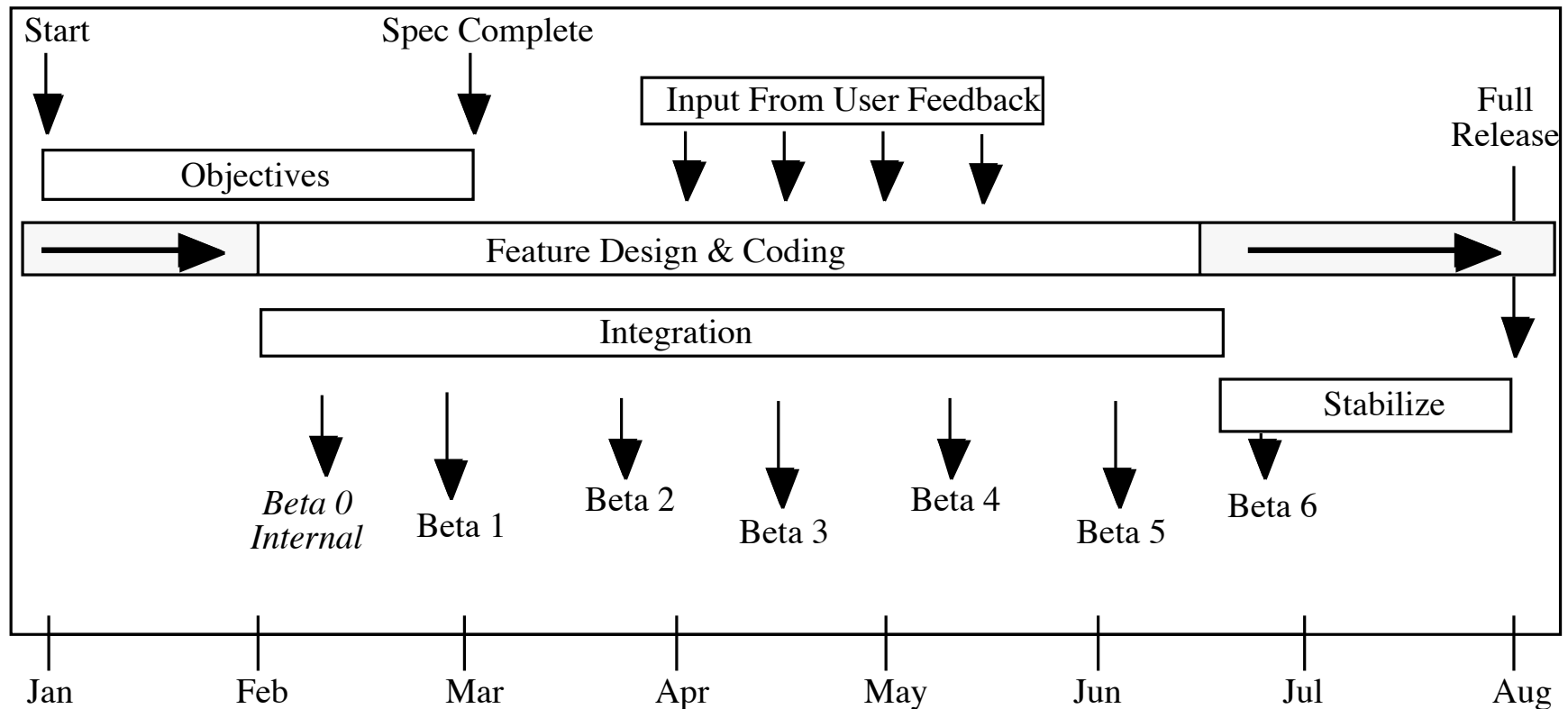
# Un modello più flessibile: fasi sovrapposte



- **Caratteristiche**
  - Basato su **apprendimento e adattamento** vs *pianificazione ed esecuzione*
  - Processo iterativo

# Esempio

## Progetto Netscape's Navigator 3.0

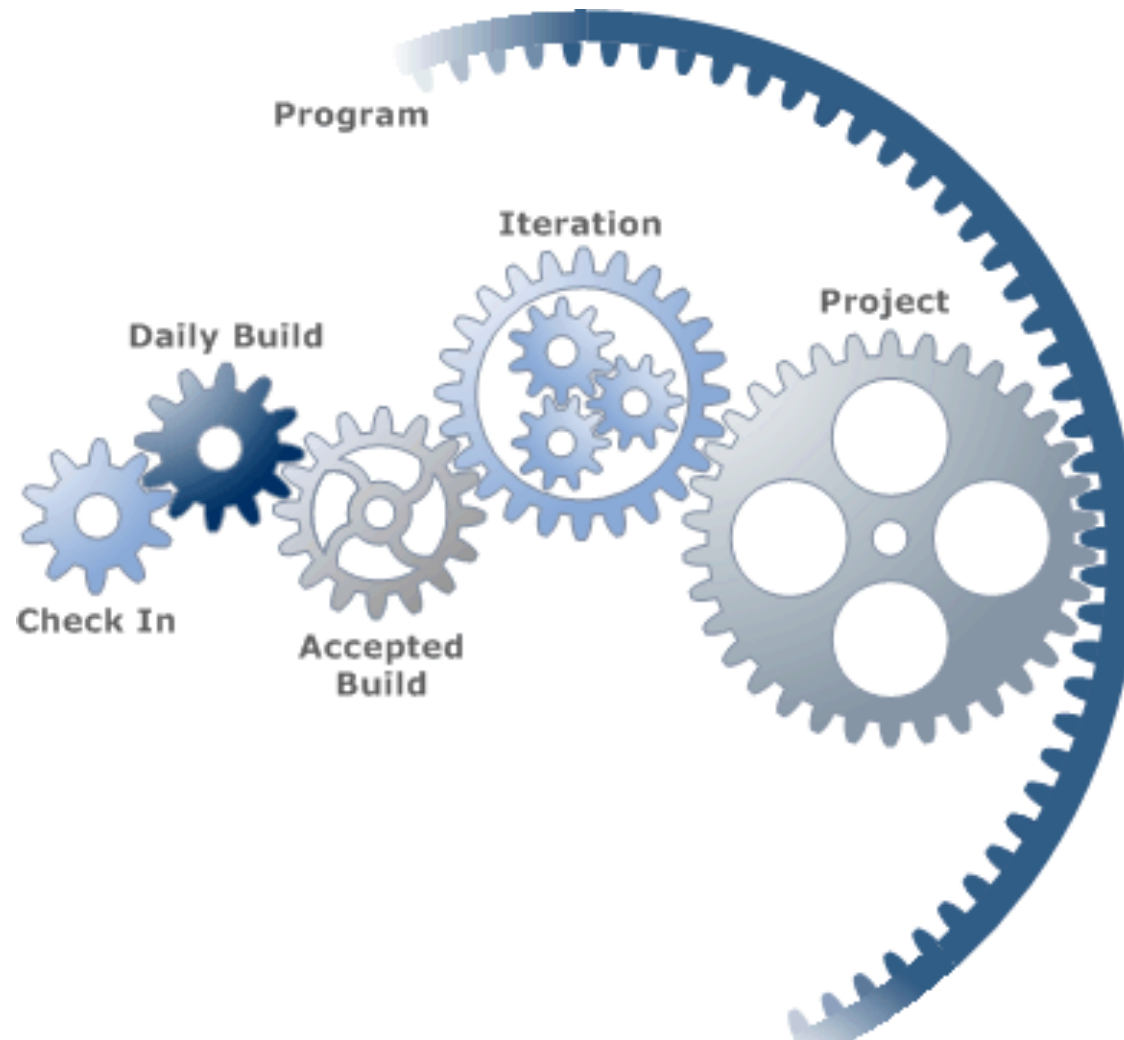


**>50% di nuovo codice sviluppato dopo il primo rilascio beta!**

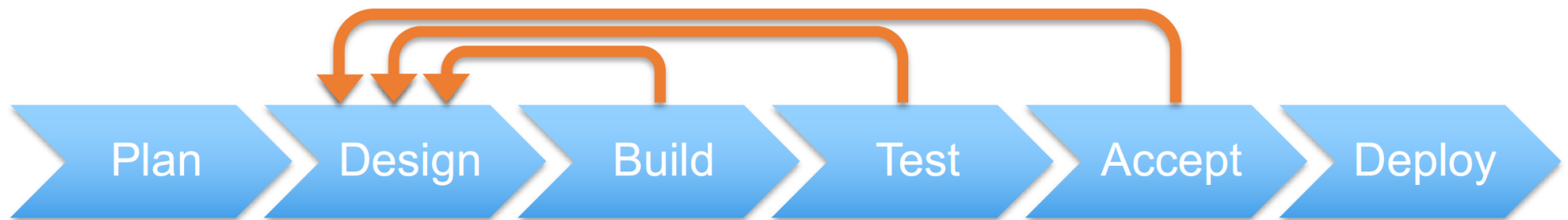
# Dimensioni del tempo

- Tempo lineare
  - Collegato ad una nozione intuitiva di progresso
  - “Oggi più di ieri e meno di domani”
- Tempo ciclico
  - Ore, giorni, settimane, mesi, stagioni...
  - Il tempo ciclico in apparenza si ripete senza progredire, ha bisogno di indicazioni di progresso
  - *“Sentinella, a che punto è la notte?” La sentinella risponde: «Viene la mattina, e viene anche la notte. Se volete interrogare, interrogate pure; tornate un'altra volta» Isaia, 21*

# Il software si costruisce in cicli (e cicli dentro cicli)

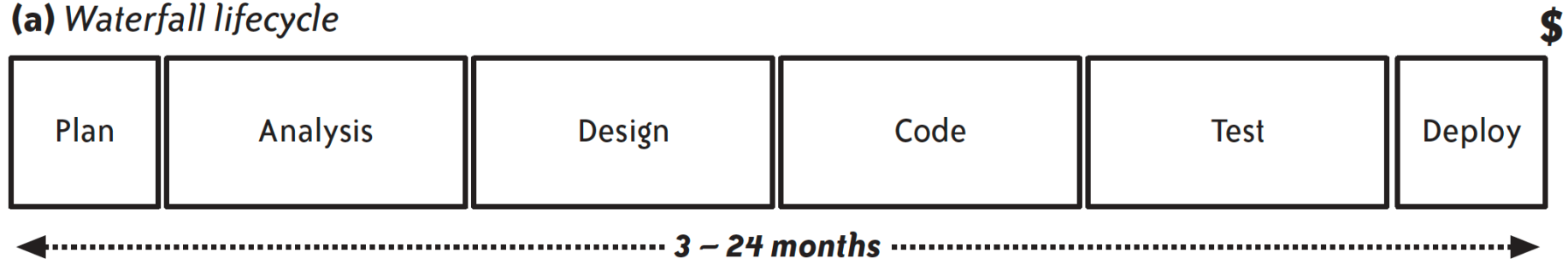


# Cicli dentro cicli

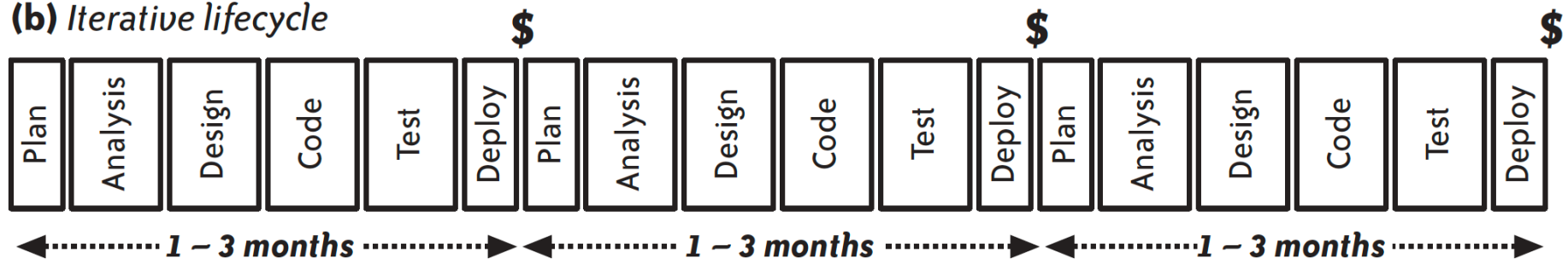


# Waterfall vs iterativo

**(a)** *Waterfall lifecycle*



**(b)** *Iterative lifecycle*



\$ = Potential release

# Modello a spirale (Boehm)



# I settori del modello a spirale

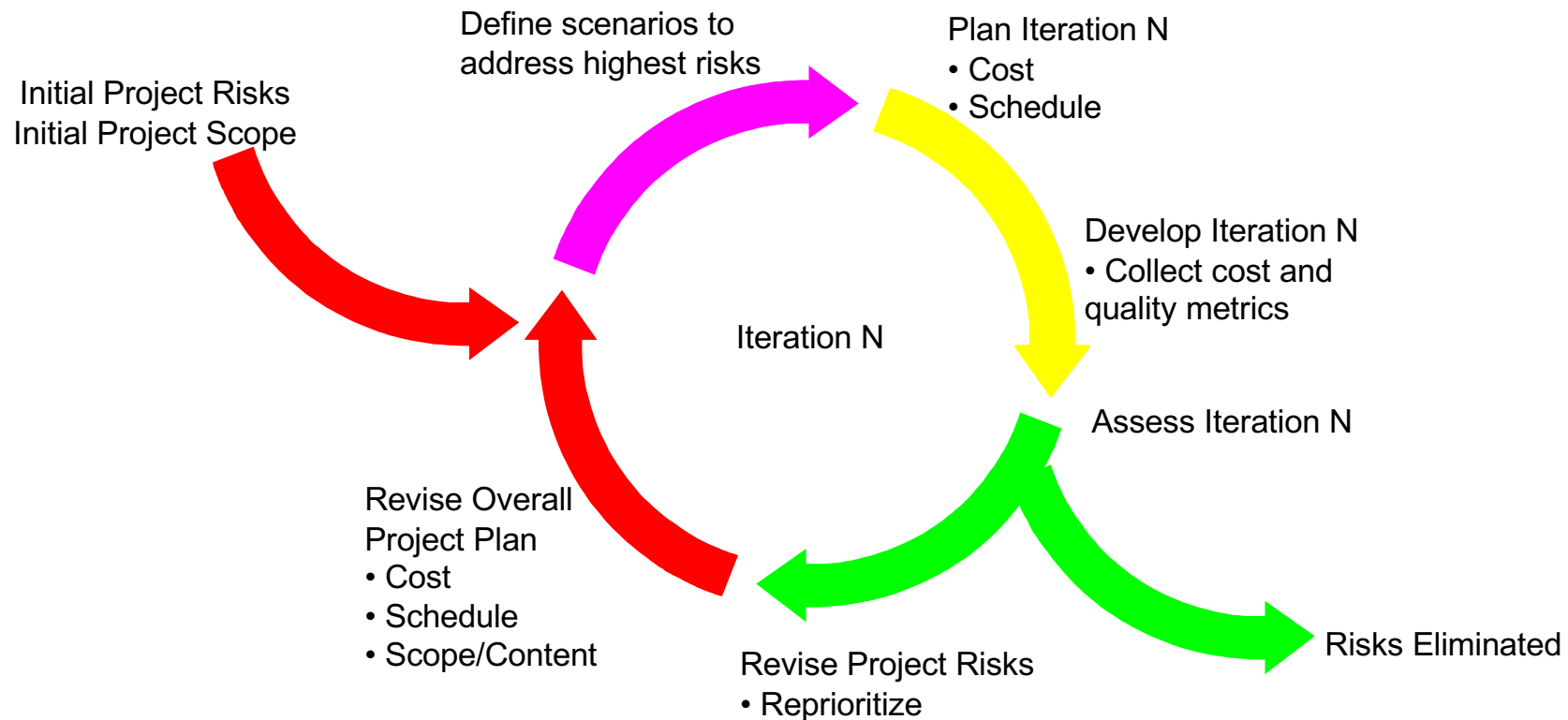
- I. Definizione dell'obiettivo
  - Ogni round identifica i propri obiettivi
- II. Valutazione e riduzione dei rischi
  - Messa in priorità dei rischi
  - Ogni rischio deve essere affrontato
- III. Sviluppo e validazione
  - Il modello di sviluppo può essere generico
  - Ogni round include sviluppo e validazione
- IV. Pianificazione
  - Revisione del progetto e pianificazione del suo futuro



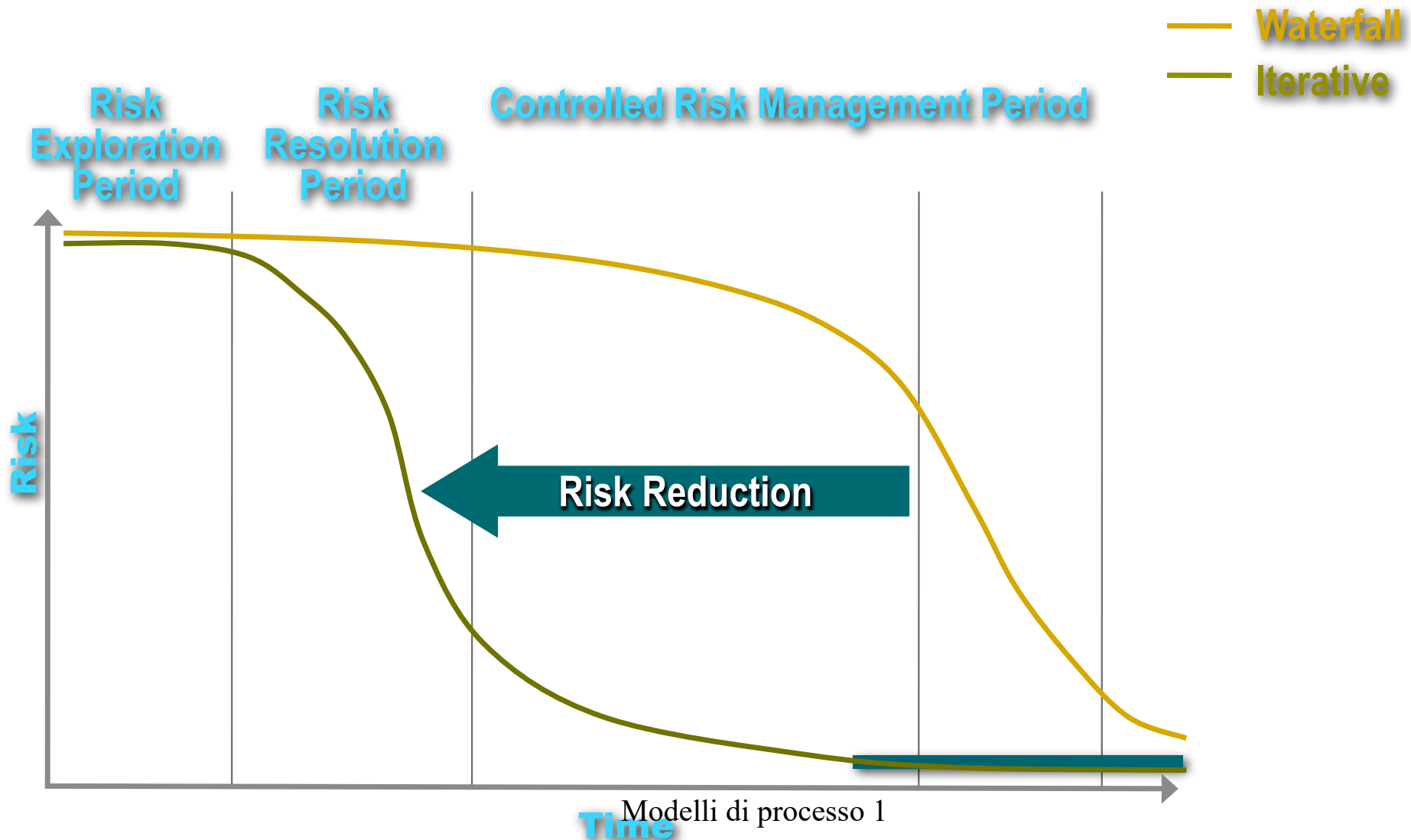
# Modello a spirale

- Adatto se requisiti instabili
- Non lineare ma **pianificato**
- Flessibile, si adatta alle esigenze utente
- Valuta il rischio per ogni iterazione
- Può supportare diversi modelli di processo
- Richiede il **coinvolgimento** del cliente
- **Difficile valutare i rischi**
- **Costoso: ROI (Return On Investment)?**

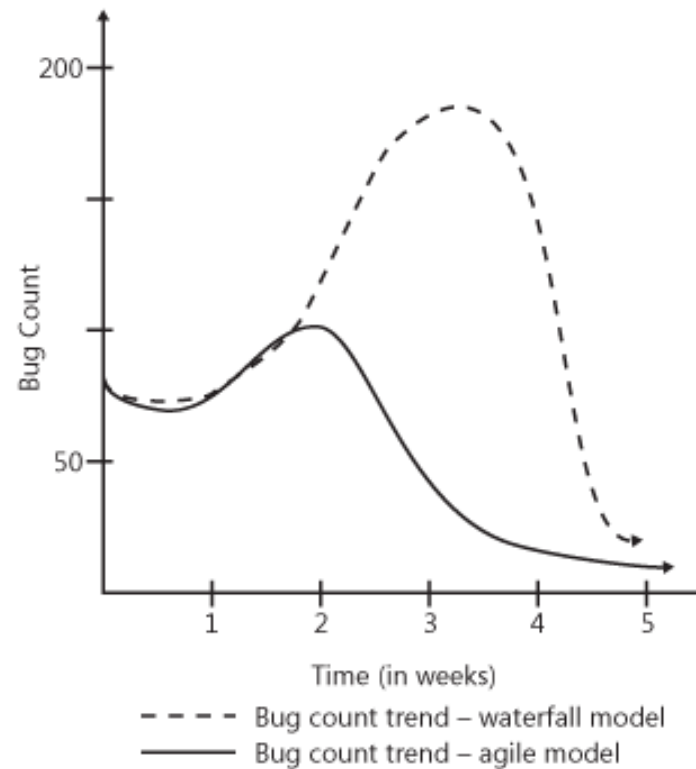
# I processi iterativi diminuiscono i rischi



# Cascata vs Iterativi

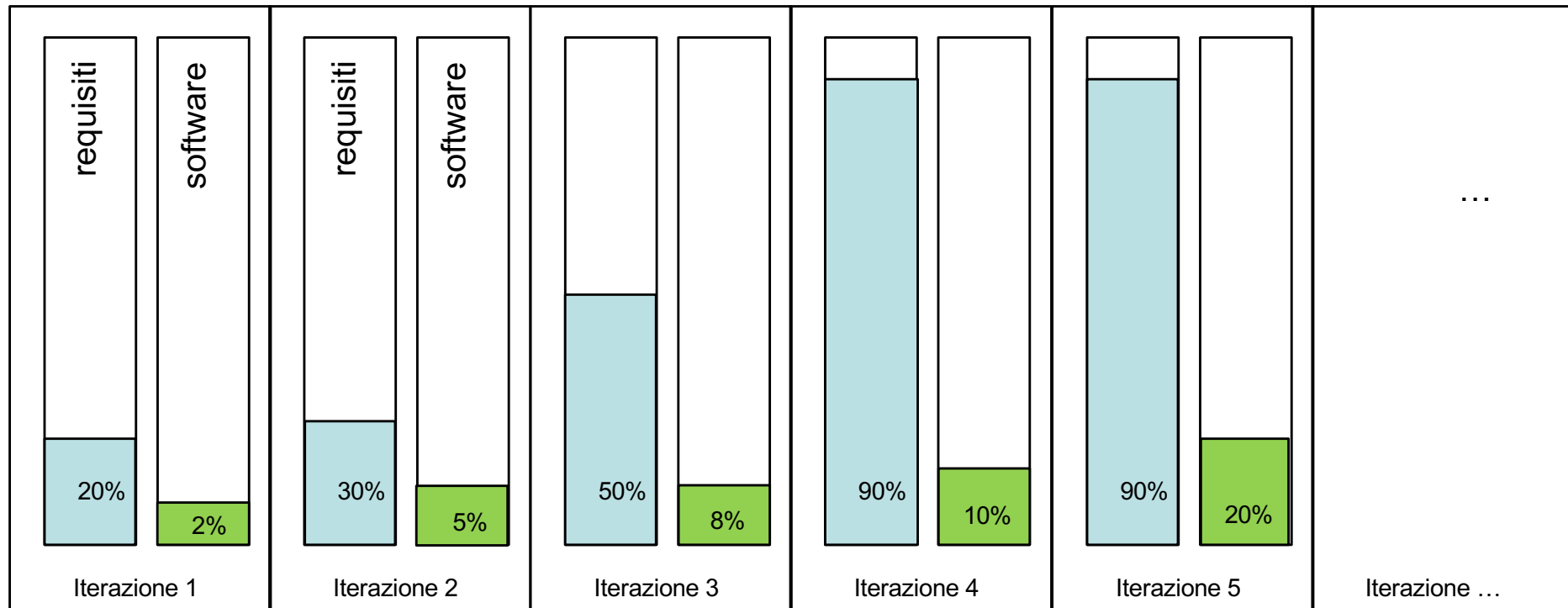


# La diminuzione del rischio



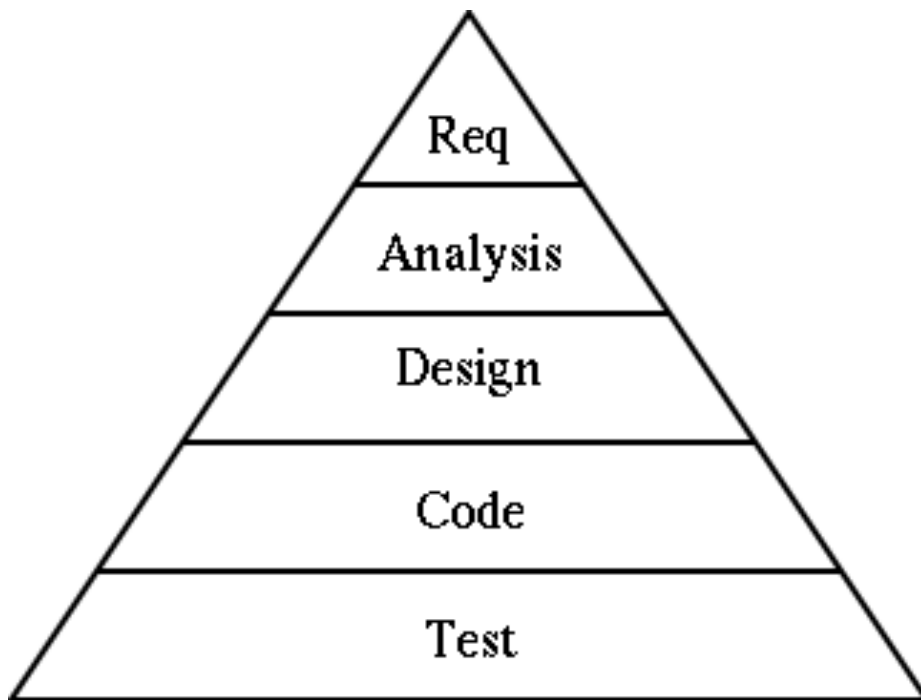
Windows Live Hotmail bug trend comparison, da Marshall, Solid Code, O'Reilly 2009

# Iterazioni

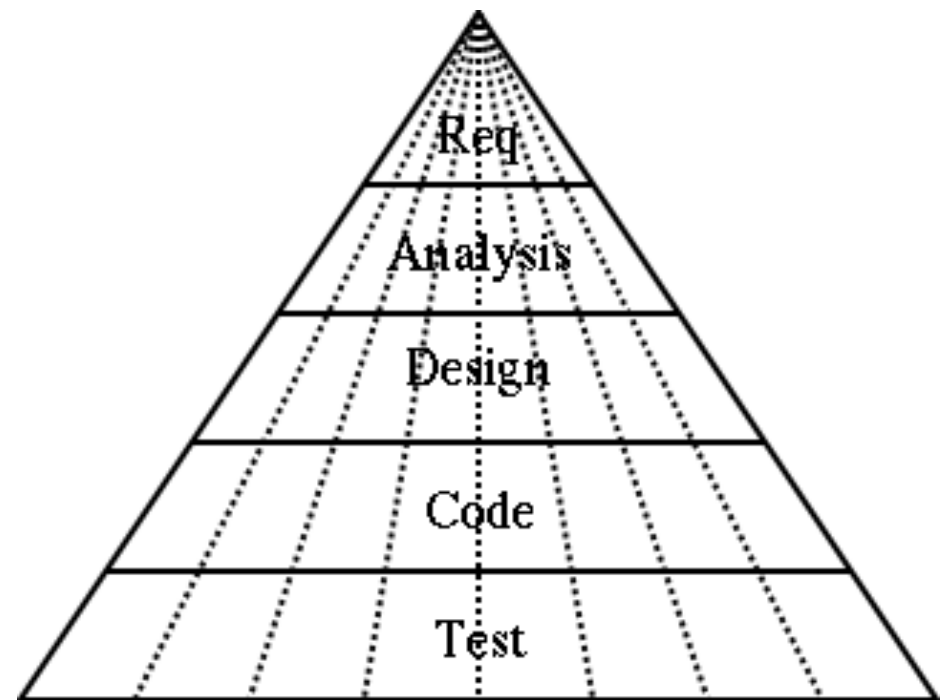


- Questa figura rappresenta un processo iterativo da oltre 5 iterazioni
- Alla fine della quarta iterazione il 90% dei requisiti è stabile ma solo il 10% del software è stato costruito

# Cascata vs iterativi: sforzo



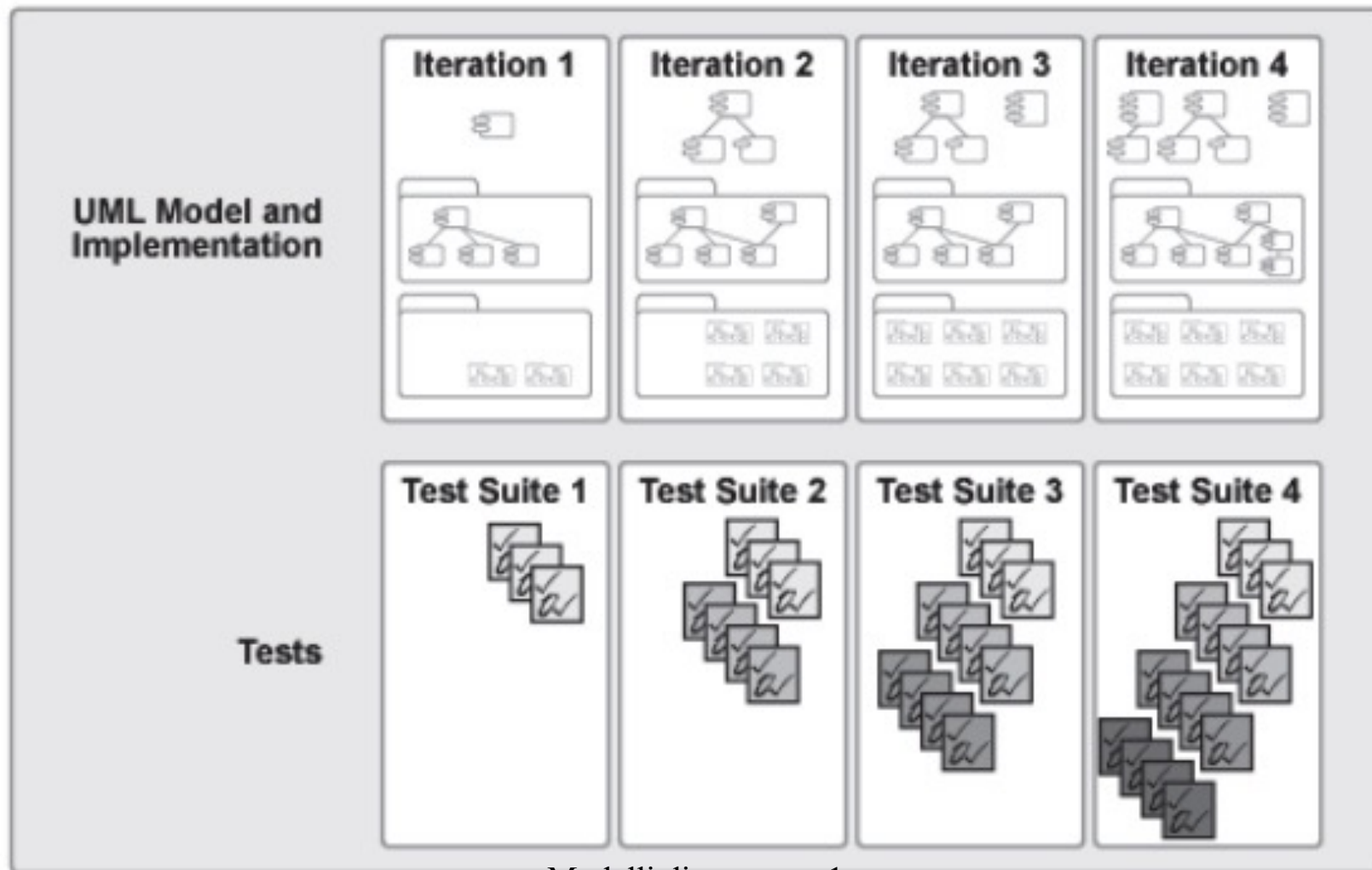
Distribuzione dello sforzo nelle fasi di un processo a cascata: il testing assorbe molto più sforzo delle altre fasi



Segmentazione dello sforzo nelle fasi di un processo iterativo

Modelli di processo 1

# Iterativi: testing incrementale

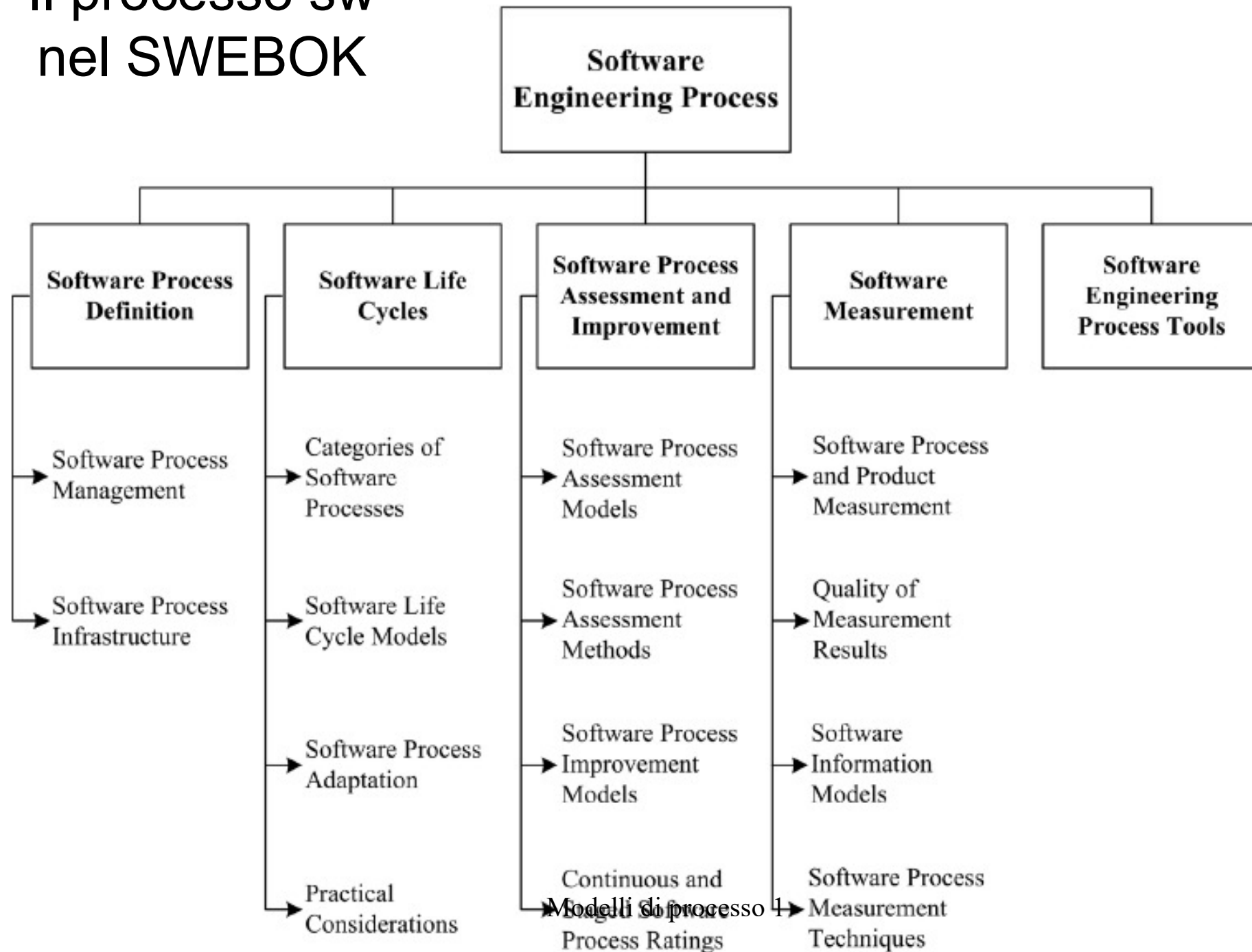


# Processi iterativi

- RUP
- Open UP
- Varianti RUP e MSF
- Synch and stabilize



# Il processo sw nel SWEBOK



# Qualità dei processi software

1. **Visibilità** – le regole e gli artefatti del processo sono noti a tutti gli stakeholder
2. Misurabilità/accuratezza - il risultato del processo può essere stimato e/o valutato (KPI: Key Performance Indicator). Esempi
  - I. Leadtime: tempo necessario dall'idea al software
  - II. Cycletime: tempo necessario per effettuare un cambiamento al software e renderlo operativo
  - III. Open/close rate: issues – cioè problemi - trovati nel sw e risolti in una unità di tempo
3. **Precisione** – il processo descrive ruoli task e artefatti in modo chiaro e comprensibile a chi deve eseguirlo
4. **Ripetibilità** – il processo può essere ripetuto anche da persone diverse, ottenendo lo stesso risultato

# Conclusioni

- Processi waterfall: pianificati, rigidi
- Processi iterativi: pianificati, flessibili
- Processi agili: non pianificati, adattivi
- Esistono molte varianti
- Ogni organizzazione definisce il modello che preferisce, eventualmente adattandolo per classi di prodotti o progetti software

# Autotest

- Cos'è un processo software?
- Quali sono le fasi tipiche del processo di sviluppo?
- Quali sono le principali differenze tra processi lineari e processi iterativi?
- Come si riconosce un processo waterfall?
- Quali sono i principali indicatori prestazionali di processo software?

# Riferimenti

- Capitolo 8 del SWEBOK: “Software engineering process”
- Boehm: A spiral model of software development and enhancement, *IEEE Computer* 21:5(61-72), May 1988
- Cusumano: How Microsoft builds software, *CACM* 1997
- Kneuper, *Software Processes and Life Cycle Models*, Springer 2018

# Siti

- **ESSENCE** [semat.org](http://semat.org)
- **RUP**  
[www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251\\_bestpractices\\_TP026B.pdf](http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf)
- **Microsoft Solutions Framework**  
[www.microsoft.com/technet/itsolutions/msf/default.msp](http://www.microsoft.com/technet/itsolutions/msf/default.msp)
- **Personal Software Process**  
[resources.sei.cmu.edu/library/asset-view.cfm?assetid=5283](http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=5283)
- **Adaptable Process Model** [www.rspa.com/apm/](http://www.rspa.com/apm/)

# Publicazioni di ricerca

- International Conference on Software and System Process
- Journal of Software Maintenance and Evolution: research and practice

# Domande?

