Modelli di processo per lo sviluppo del software: modelli pianificati

Prof. Paolo Ciancarini
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 di processo di sviluppo lineari
- Modelli iterativi

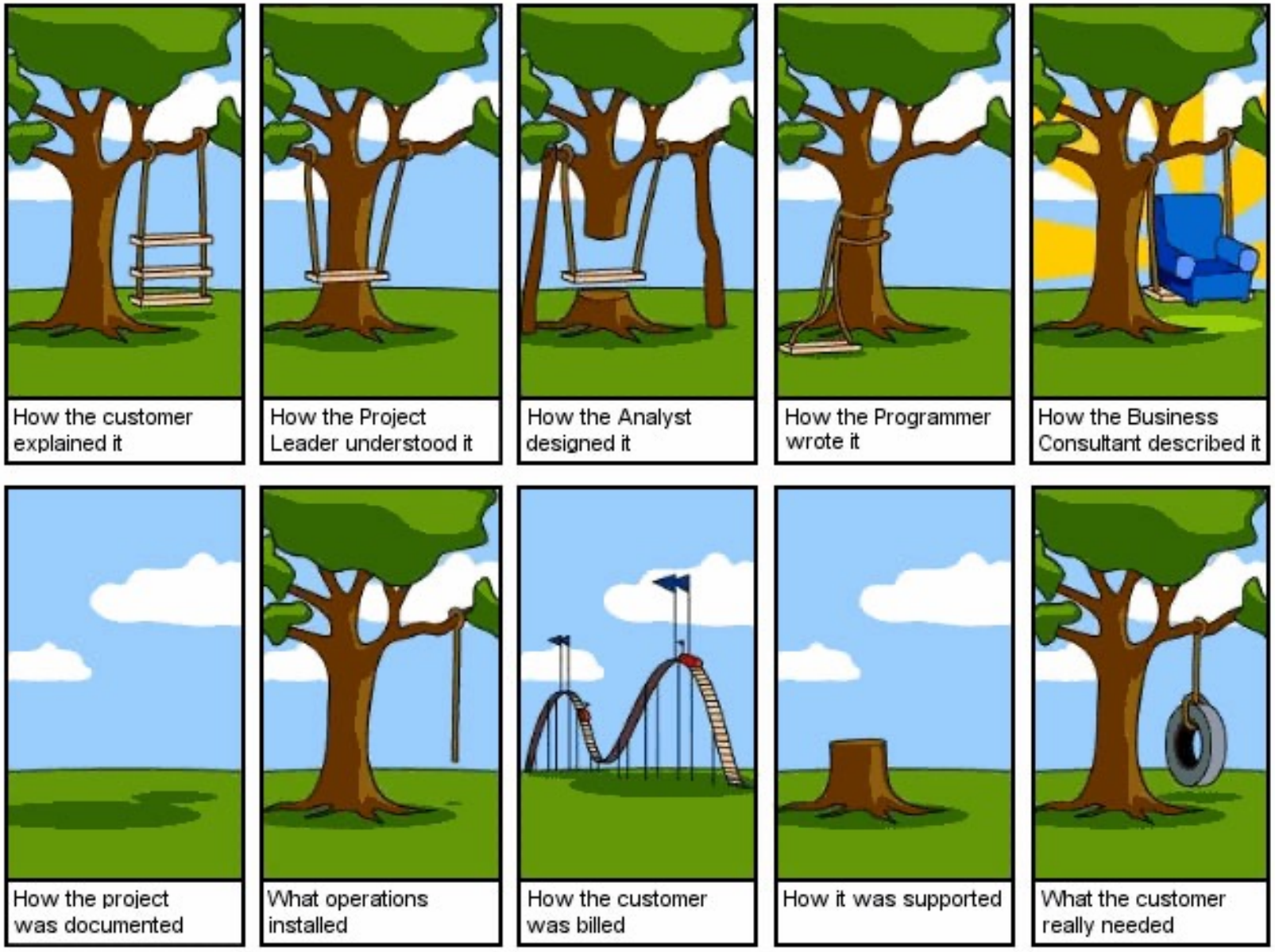
Nella prossima:

- Modelli agili

Successivamente:

- Modelli di processo orientati alla qualità
- Modelli open source

Costruire software



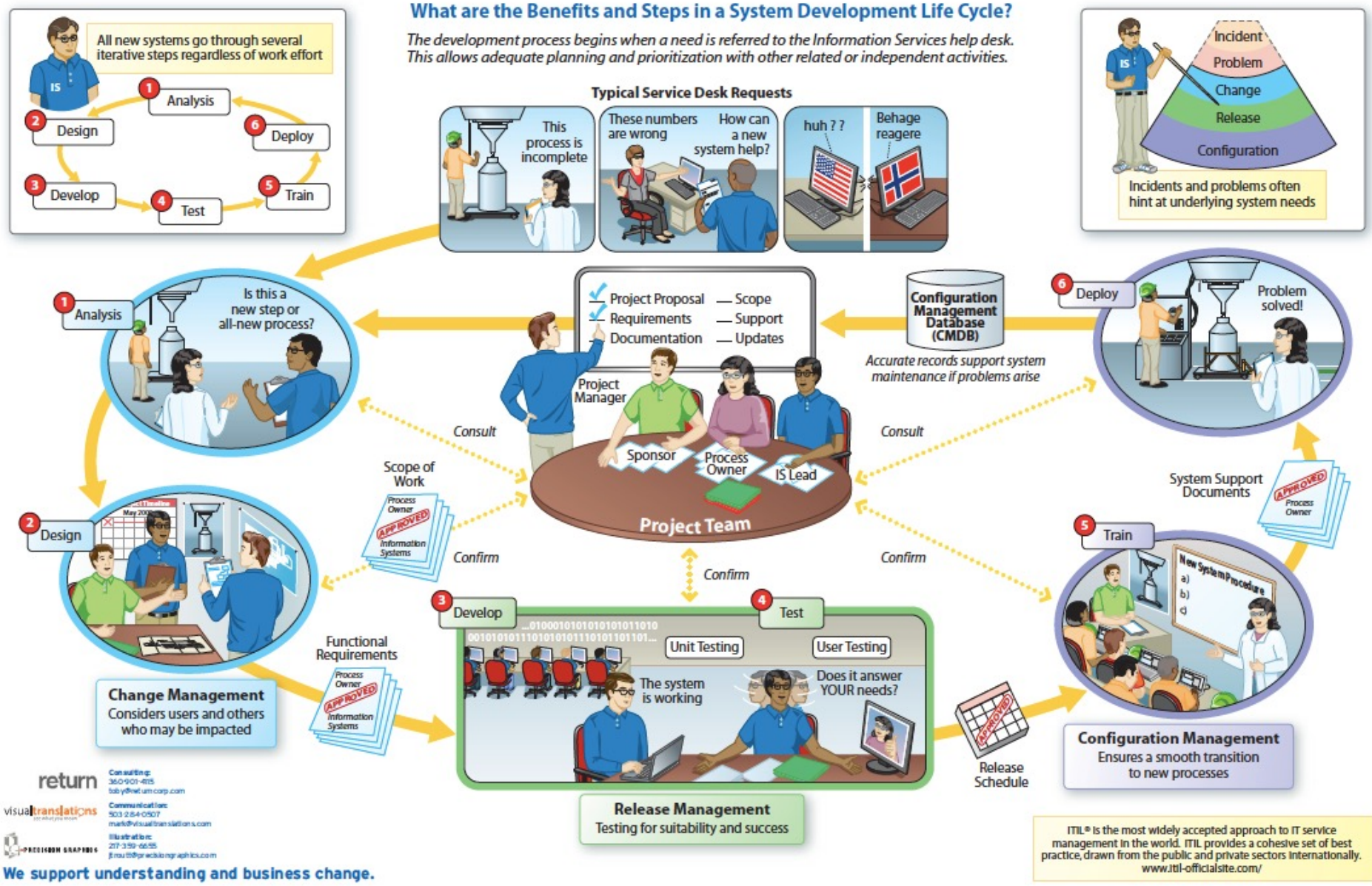
Le fasi “naturali” dello sviluppo software

- 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

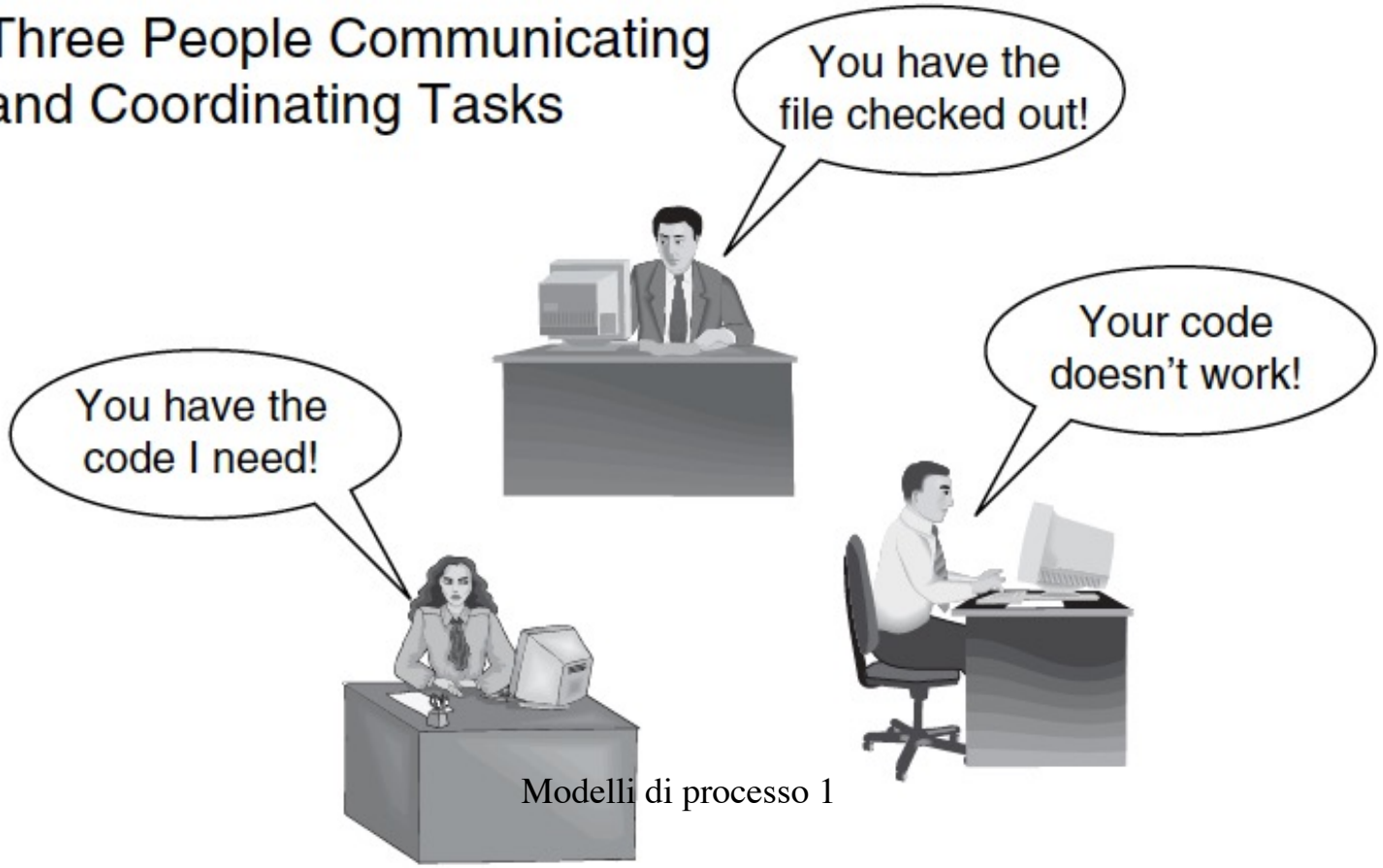
- Il **processo di sviluppo** è una parte del ciclo di vita del software
- Il **ciclo di vita del software** (*software lifecycle*), designa le varie fasi della vita di un software, dalla sua concezione al suo ritiro
- Di solito il processo di sviluppo inizia dalla concezione e finisce col rilascio finale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



Modelli di processo 1

One Person
Very Little Coordination
and Communication
Overhead

Three People Communicating
and Coordinating Tasks



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)

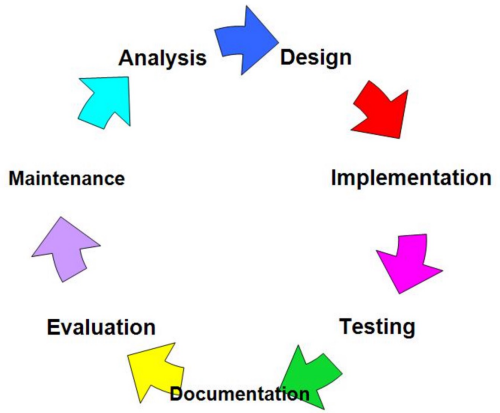
Discussione

- 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

Requisiti **nuovi** o **modificati**



Sistema e sua **documentazione**

Processo di sviluppo di un sistema

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

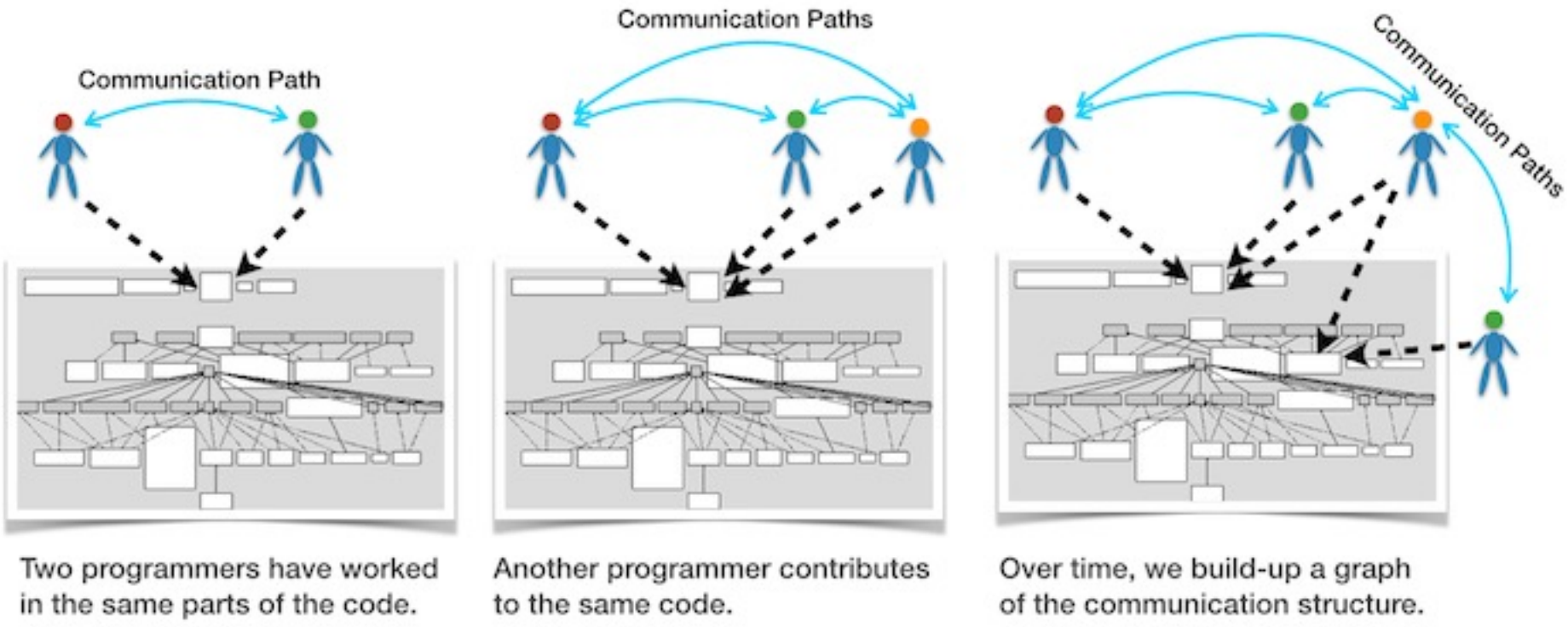
Le organizzazioni che progettano sistemi ne progettano la struttura riproducendo le proprie strutture comunicative (es. l'organigramma)

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

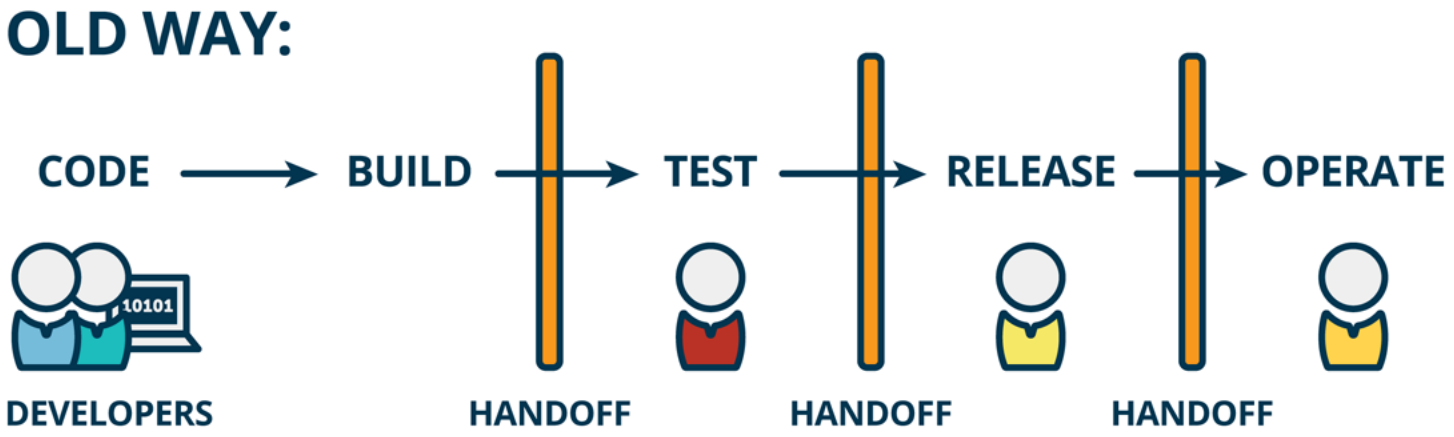
La legge di Conway ha per conseguenza che gli sviluppatori che lavorano sugli stessi componenti devono poter comunicare senza ostacoli.
Ovvero, devono essere vicini da un punto di vista organizzativo



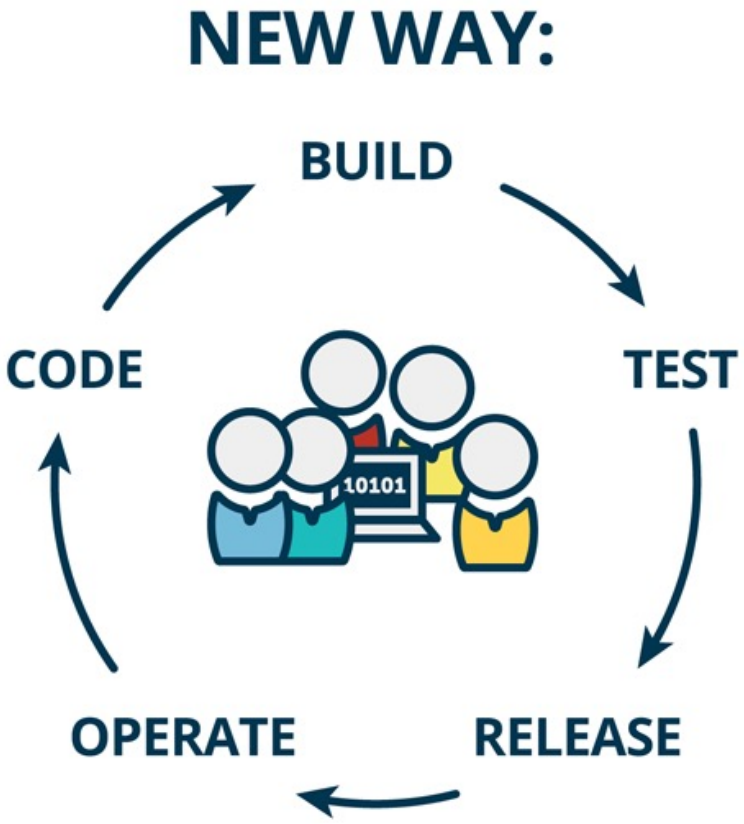
Conseguenze della legge di Conway

- 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



Conseguenze della legge di Conway



Il software è un costrutto sociale

- Il software è il prodotto di un processo sociale, che coinvolge molte persone con ruoli e interessi diversi
- Quando viene messo in opera modifica i rapporti sociali tra le persone
- Molte qualità del software si possono giudicare solo in termini «sociali»

MacCormack, Rusnak, Baldwin: Exploring the Duality between Product and Organizational Architectures: a Test of the Mirroring Hypothesis, *Research Policy*, 41:8(1309-1324), 2012

Processo software

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

Esempi 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

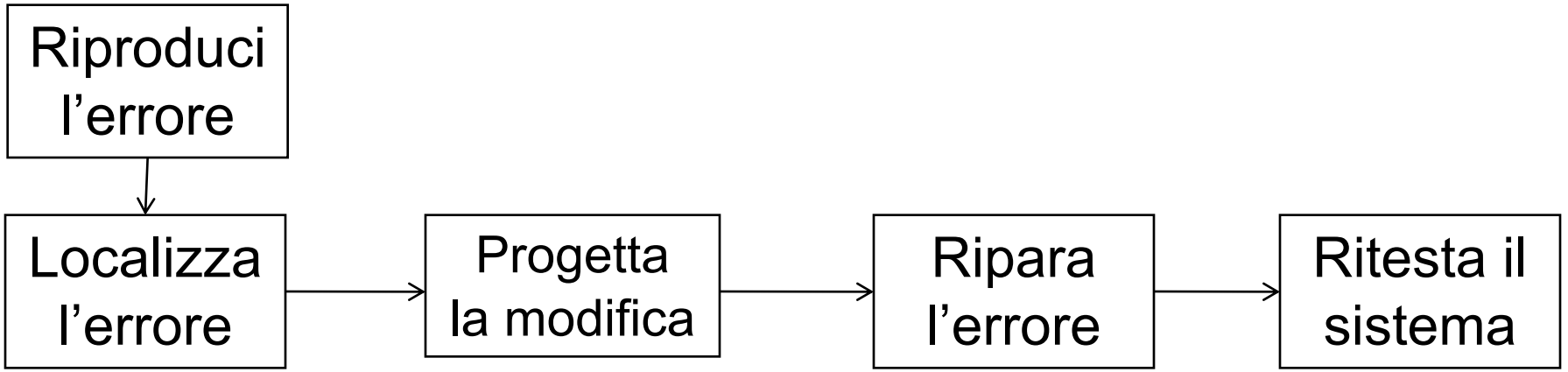
- **Programmare è un attività personale**
  di solito non si definisce una sequenza generica di passi “creativi” della programmazione

- **Il debugging è una forma di verifica**
  per scoprire e rimuovere errori nel programma

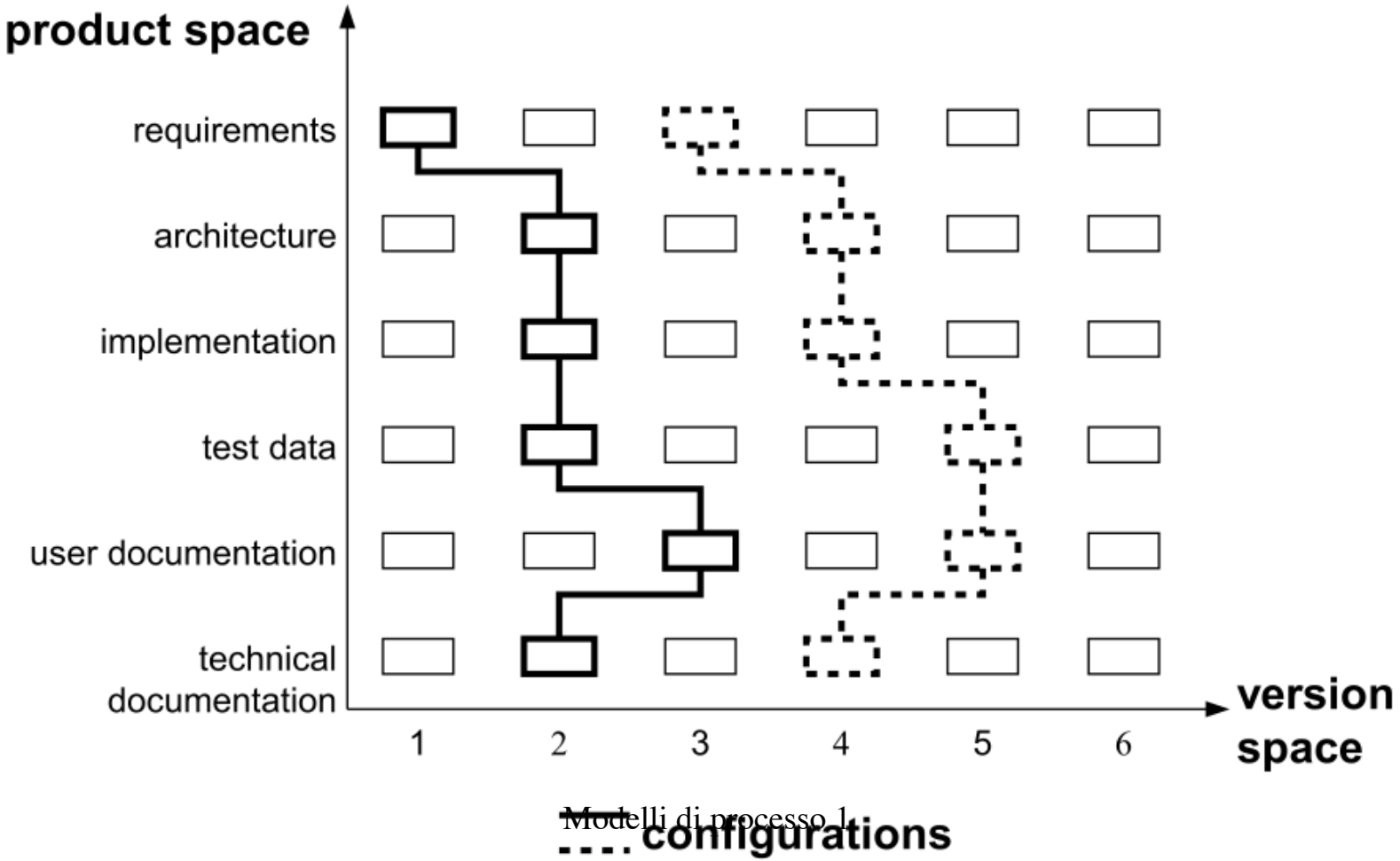
Il processo edit-compile-debug

- **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**

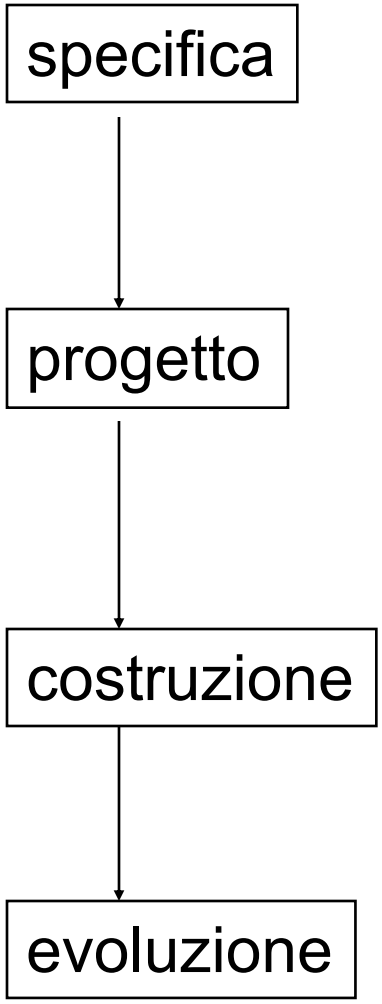
Il processo del debugging



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



Segmentare il ciclo di vita



È la fase di stesura dei **requisiti** e di descrizione degli scenari d'uso

Il progetto determina un'**architettura software** capace di **soddisfare** i **requisiti** specificati

La costruzione, o ***codifica***, è una fase complessa che include il ***testing*** e termina con il ***deployment*** del sistema

Manutenzione **perfettiva**
Manutenzione **correttiva**
Manutenzione **adattiva**

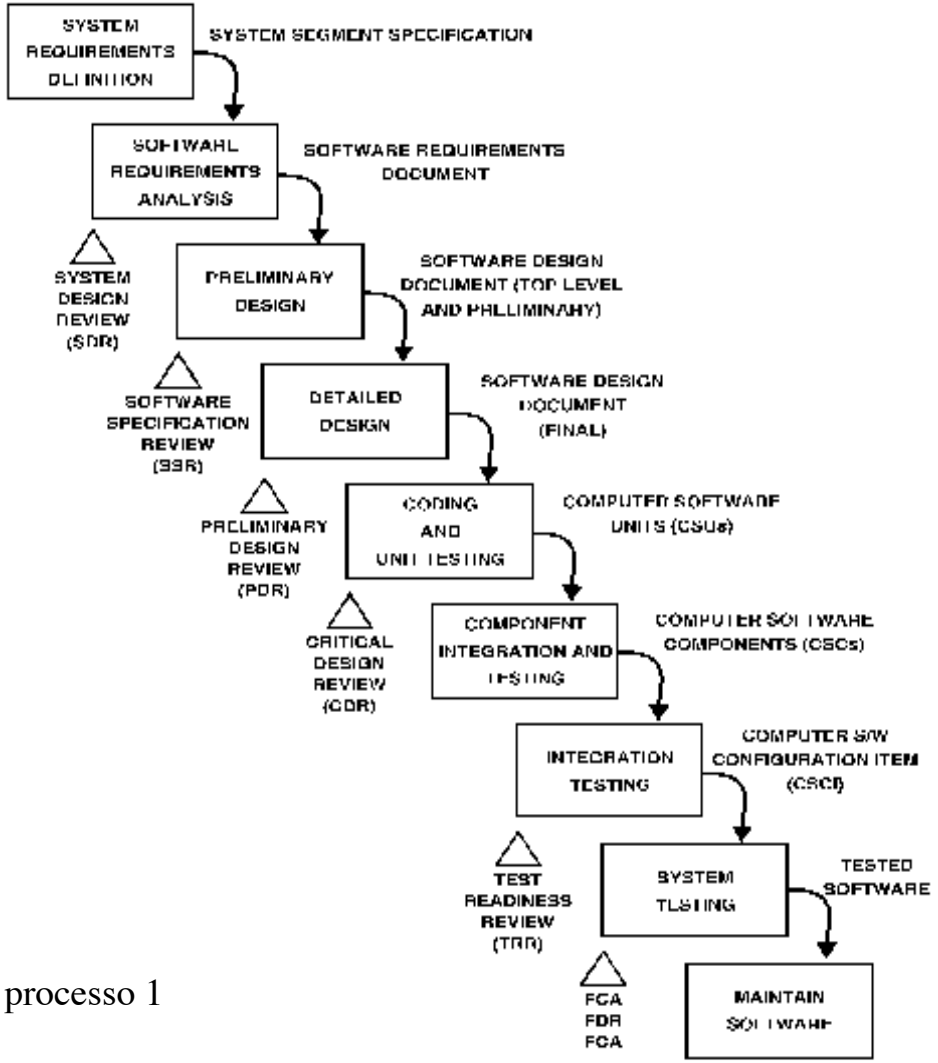
Modelli di processo

- 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 importante dei processi sw
      - utili a diversi scopi, ad esempio per valutarli, criticarli o estenderli

Esempio

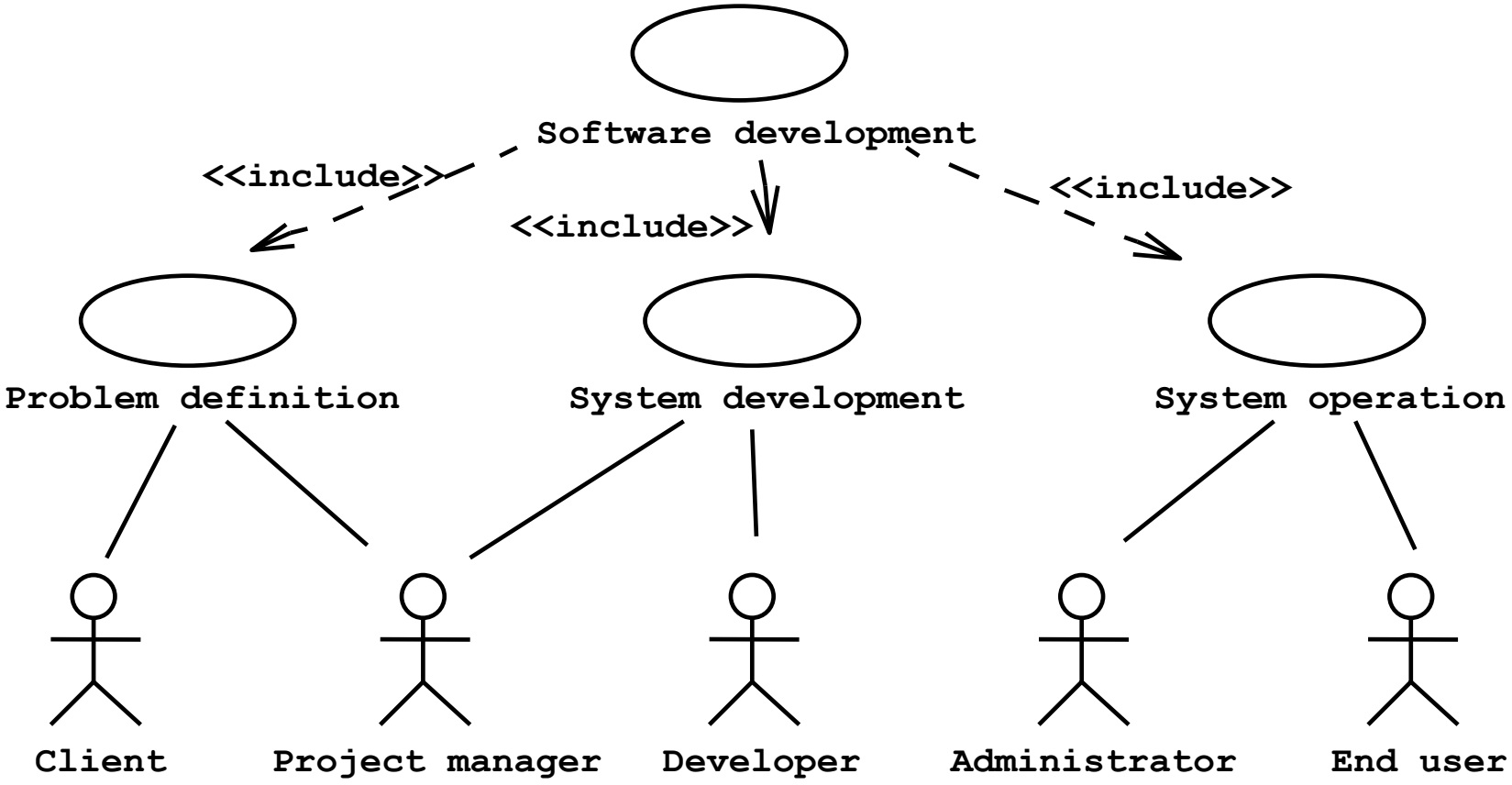
- **Modello di processo**: **waterfall**, cioè pianificato lineare
- **Processo**: istanza di un modello waterfall che viene “eseguita” per creare un sistema
  - crea una serie di artefatti come prescritto dal modello



Descrivere un processo

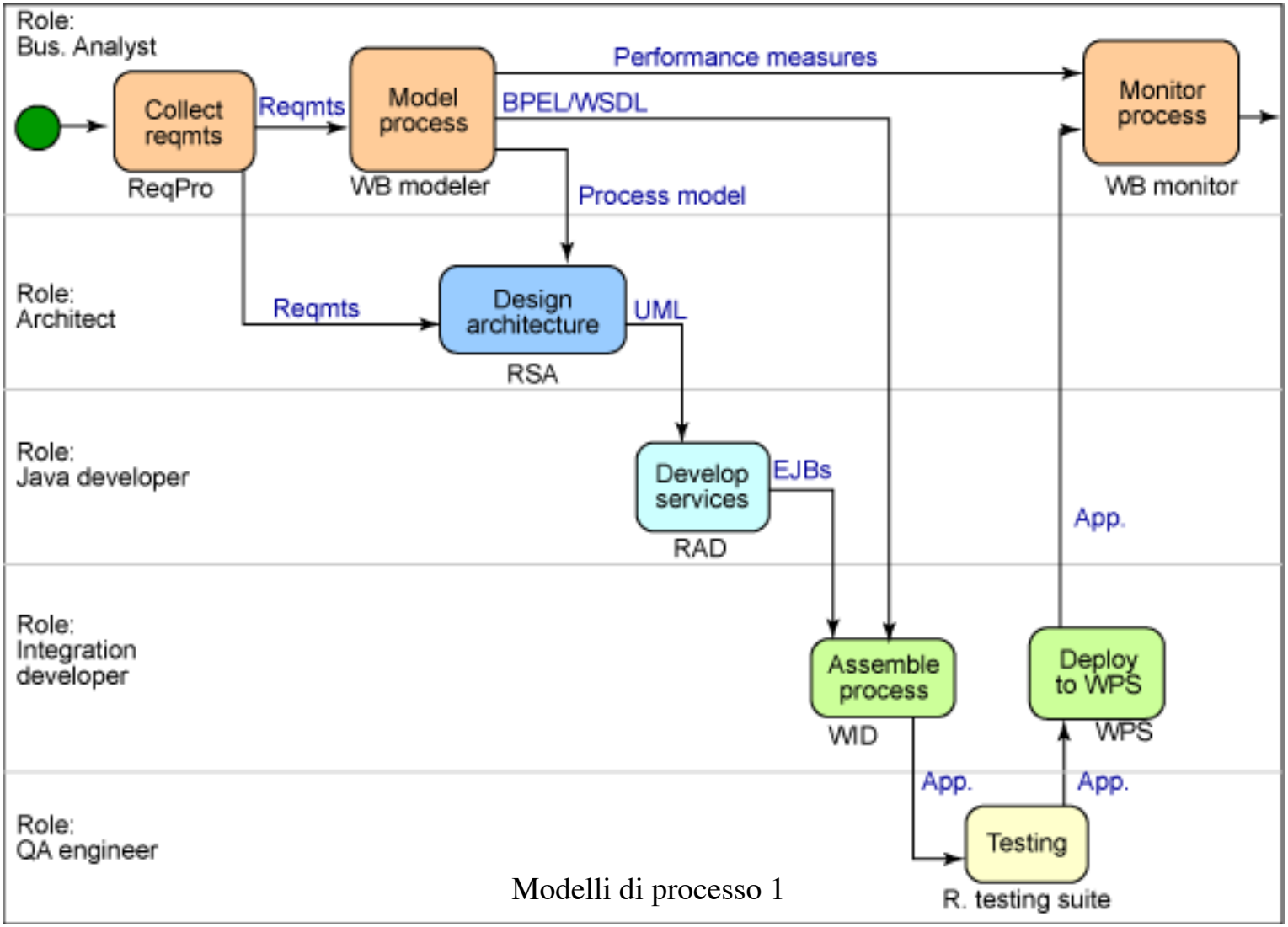
- Occorre descrivere/assegnare i **ruoli** (es. stakeolders)
- Occorre descrivere/monitorare le **attività**
- Occorre descrivere/assemblare gli **strumenti**
- Occorre descrivere/controllare i gli **eventi**
- Occorre descrivere/validare i **documenti** (es. requisiti)
- Occorre descrivere/verificare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

Modello di un ciclo di vita sw



Questo modello è descritto usando un diagramma dei casi d'uso (UML)

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



Perché descrivere un processo sw

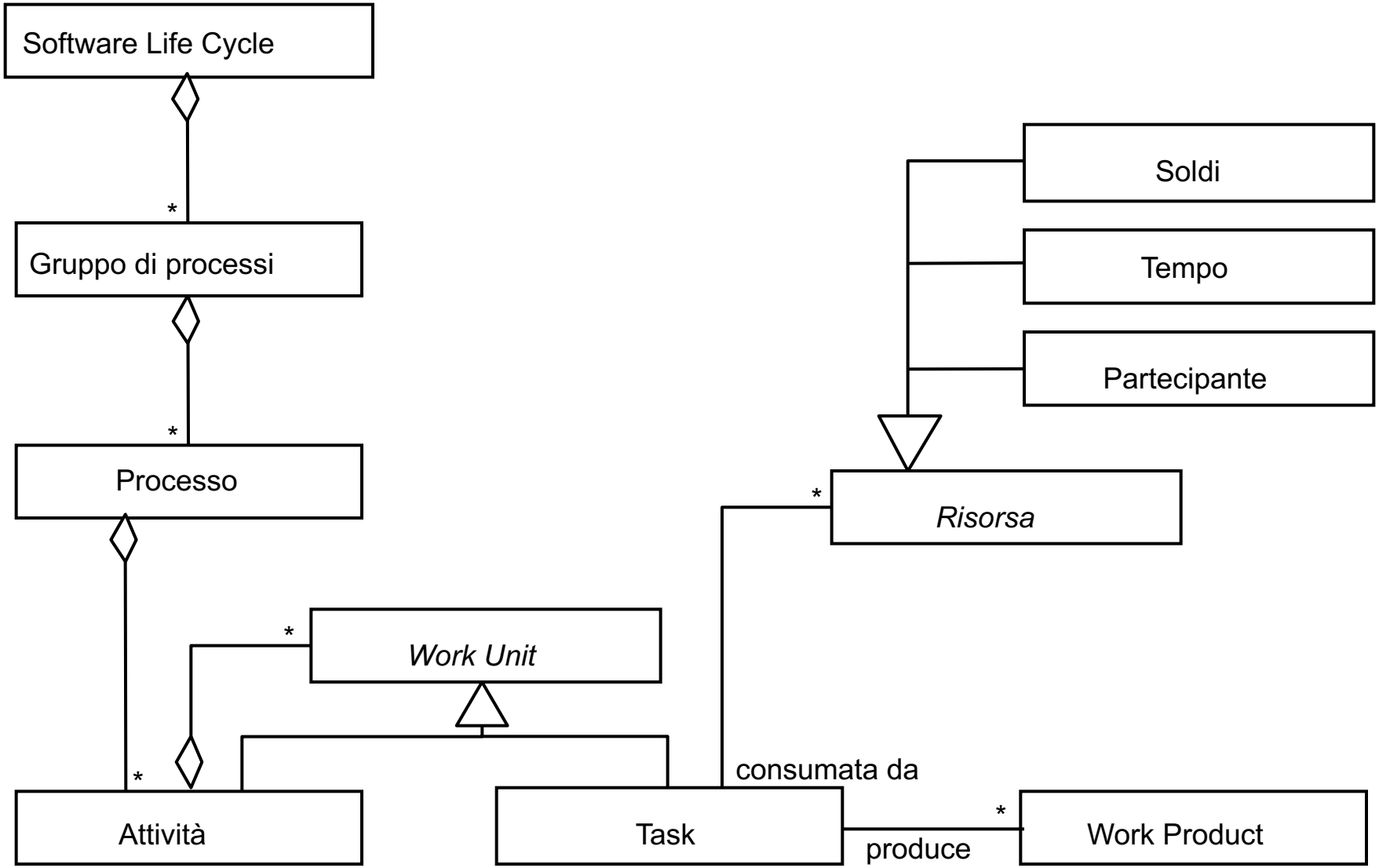
Processo software:

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

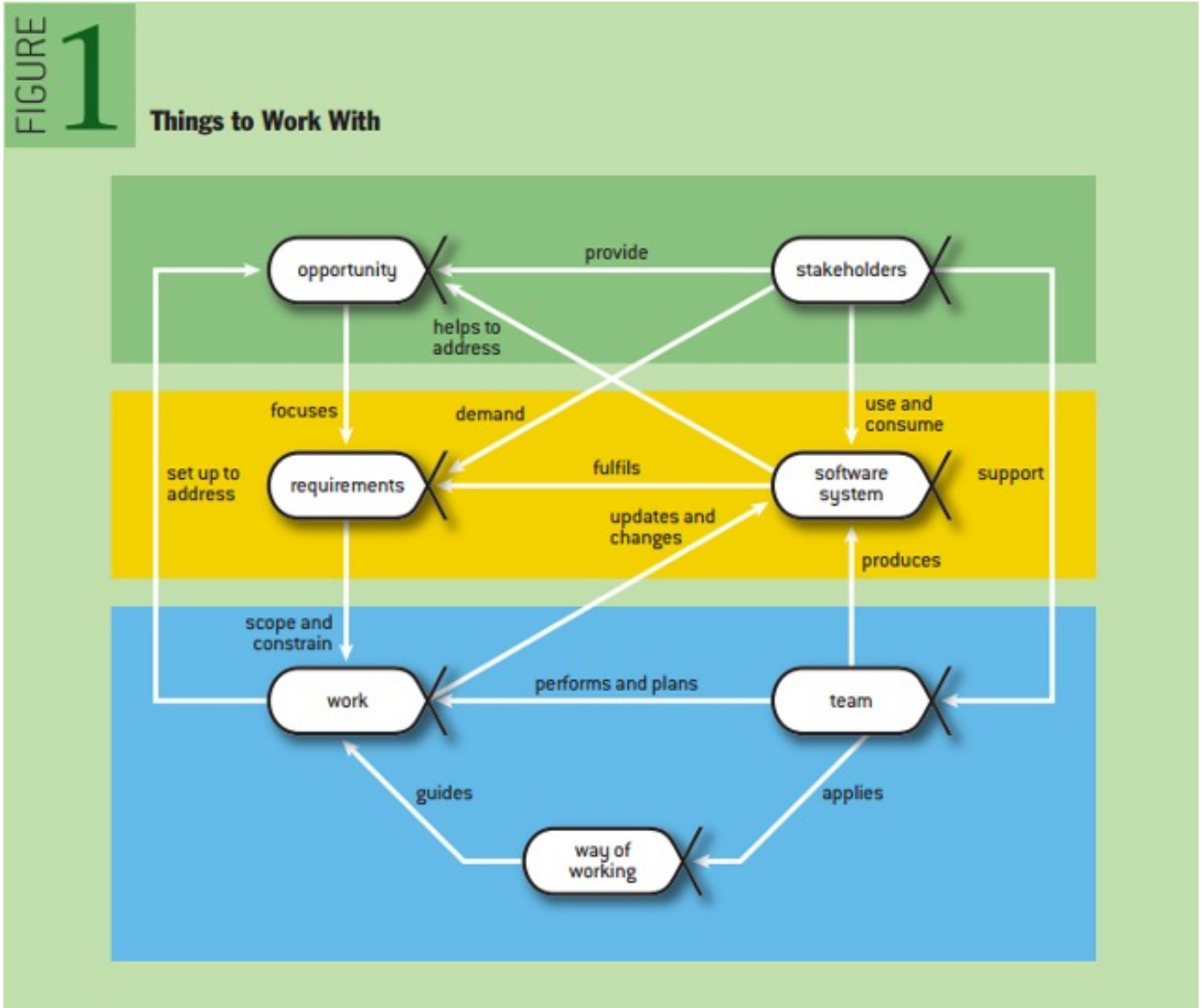
Benefici della modellazione dei processi sw:

- 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"

IEEE 1074 Standard: developing software life cycle process



Essence: un metodo di descrizione di processi sw (semat.org)



Modelli di processo 1

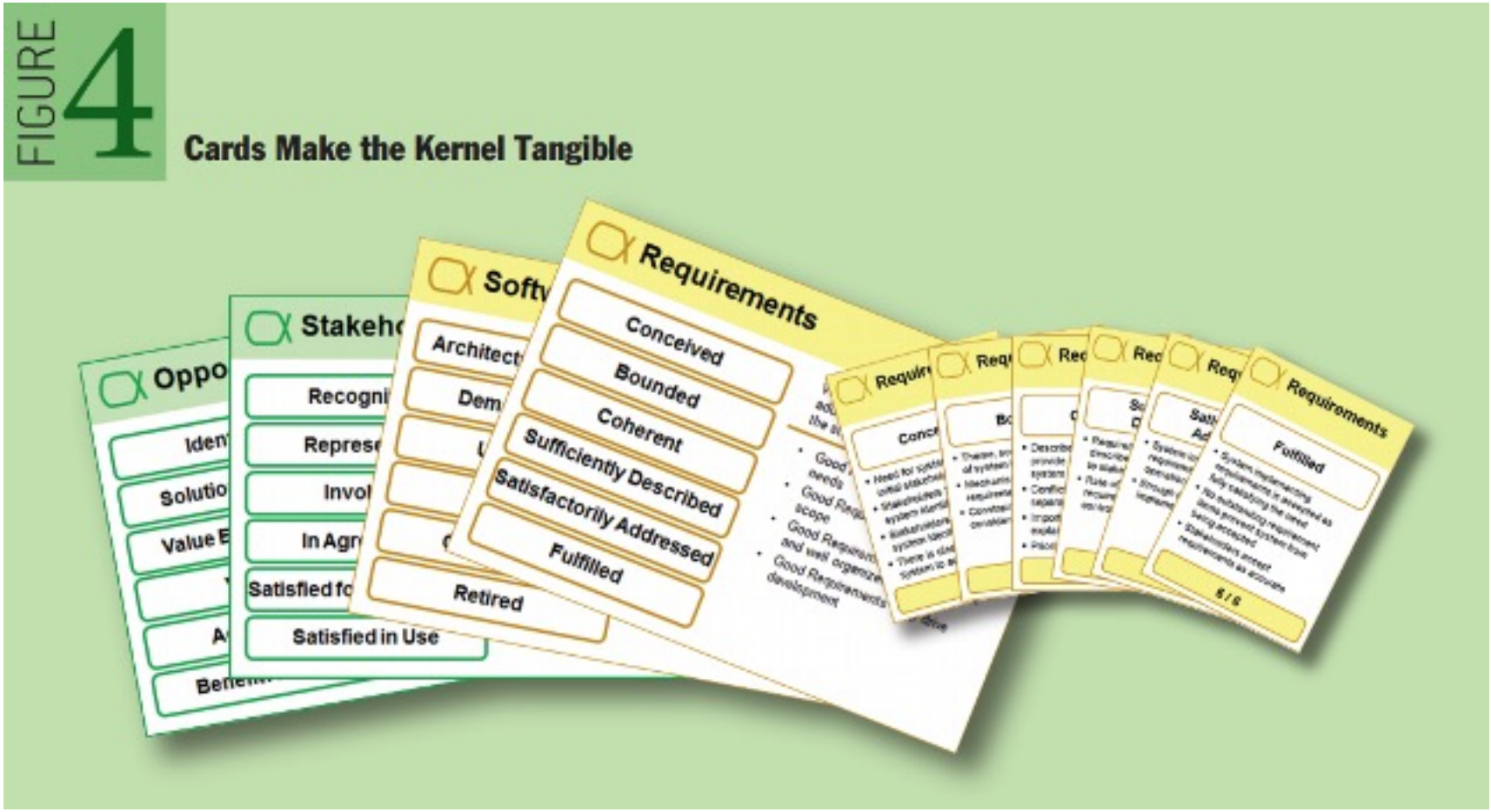


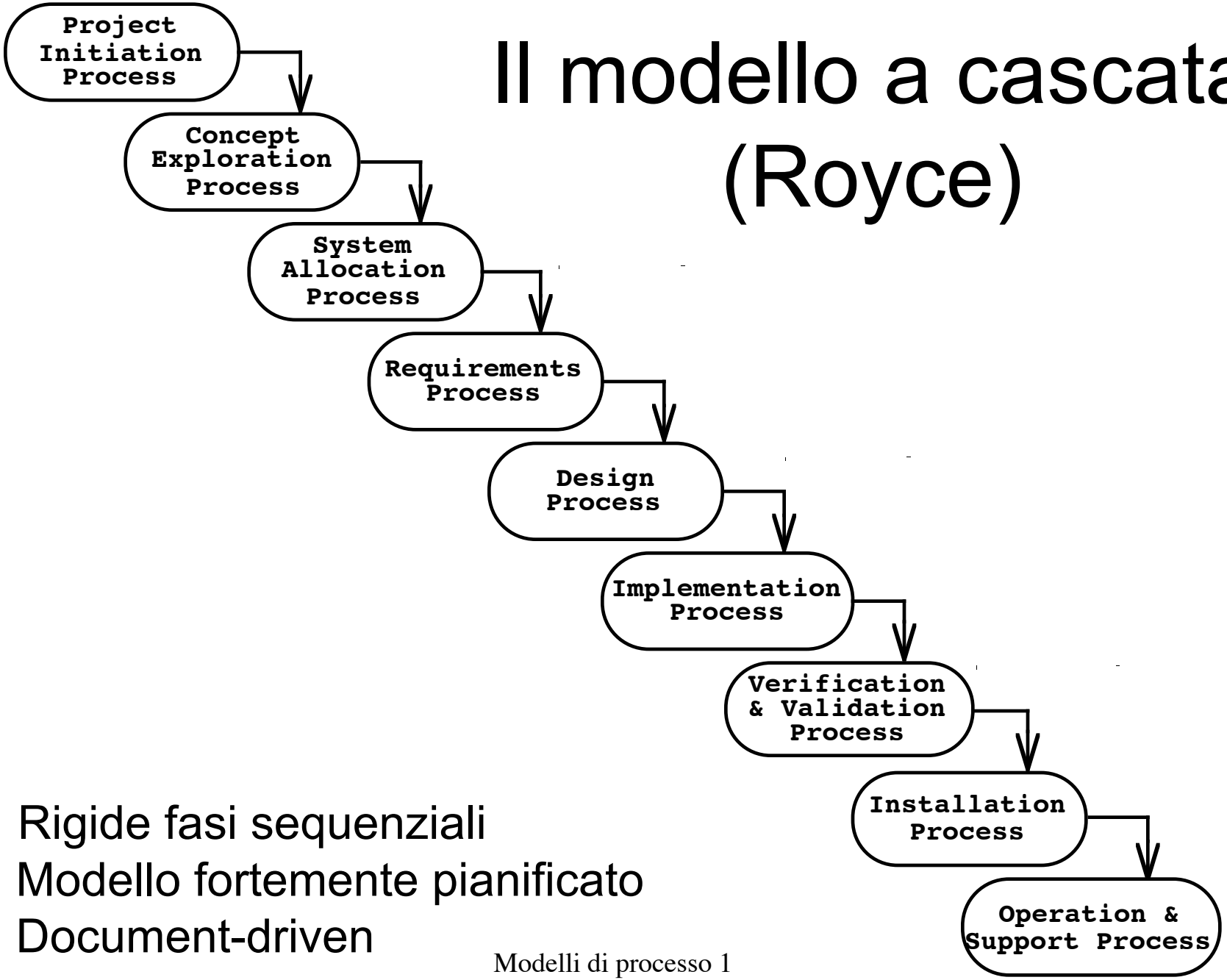
FIGURE 4
Cards Make the Kernel Tangible
Requirements
Conceived
Bounded
Coherent
Sufficiently Described
Satisfactorily Addressed
Fulfilled
Retired
Satisfied in Use
Requirements
Fulfilled

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

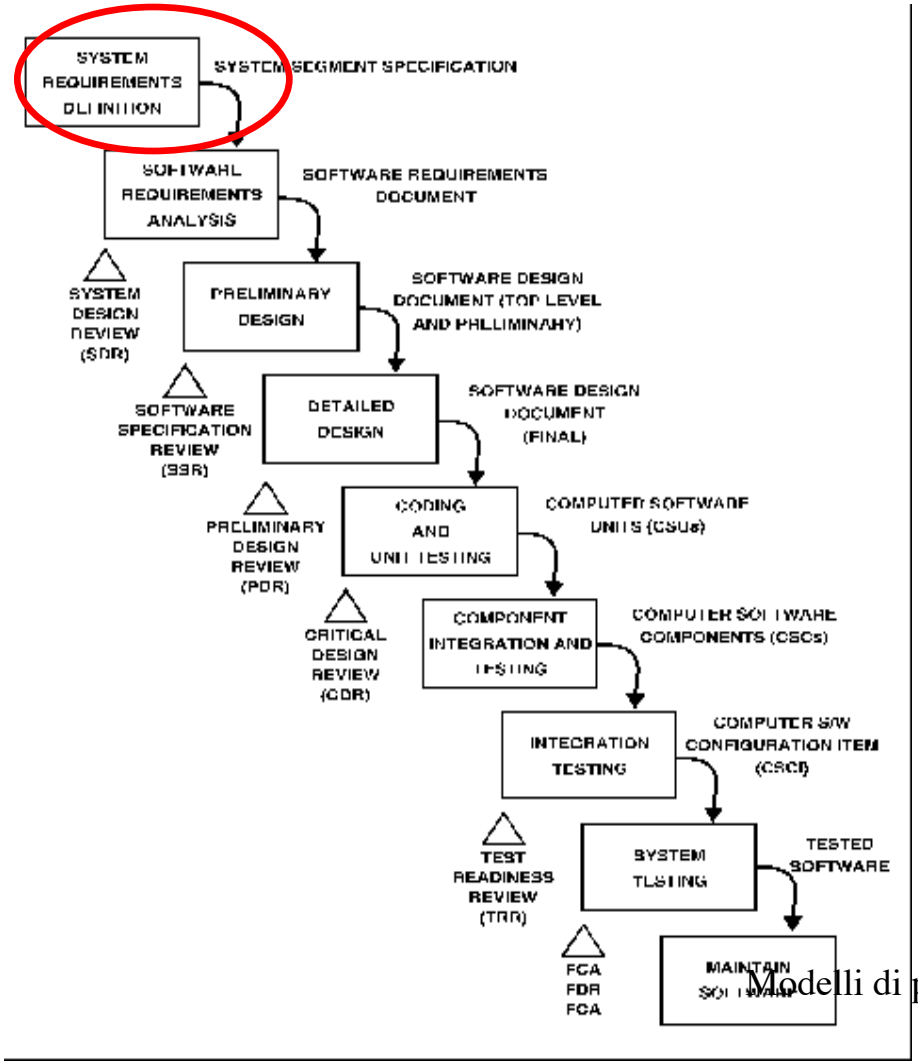
- **Sviluppo formale (esempio: B method)**
  - Il modello matematico di un sistema viene trasformato in un'implementazione

Il modello a cascata (Royce)



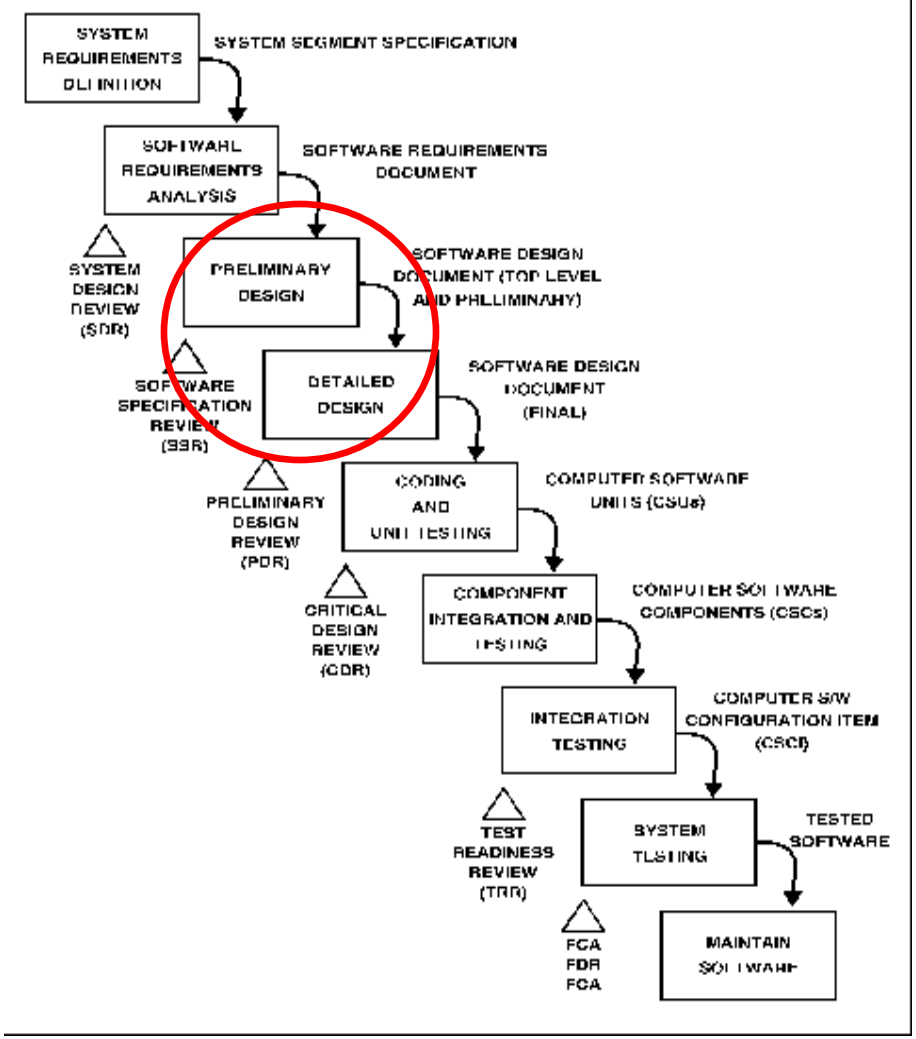
- Rigide fasi sequenziali
- Modello fortemente pianificato
- Document-driven

Analisi dei requisiti



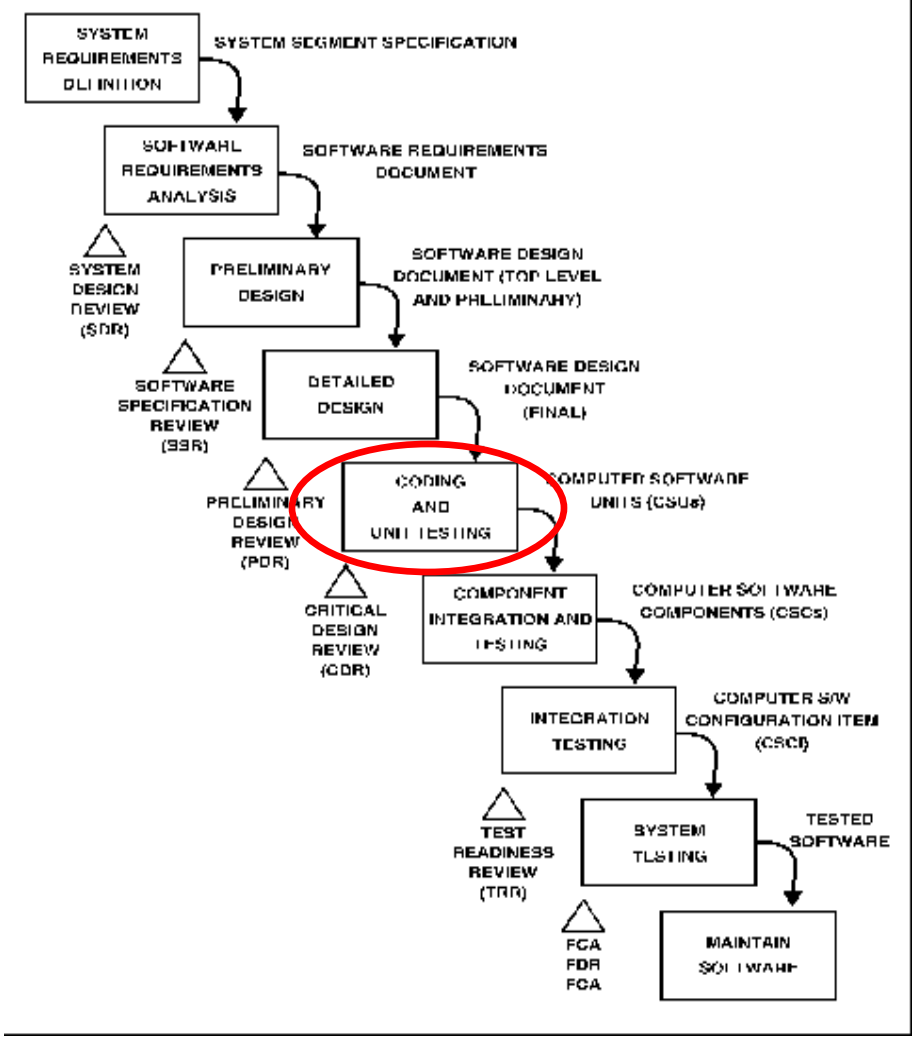
- Utenti e sviluppatori sw
- Successo se:
  - il sw soddisfa i requisiti
  - i requisiti soddisfano i bisogni come percepiti dall'utente
  - I bisogni percepiti dall'utente riflettono i bisogni reali
- Prodotto di questa fase (*deliverable*):
  - documento su cosa il sistema deve fare (non come)

Progetto



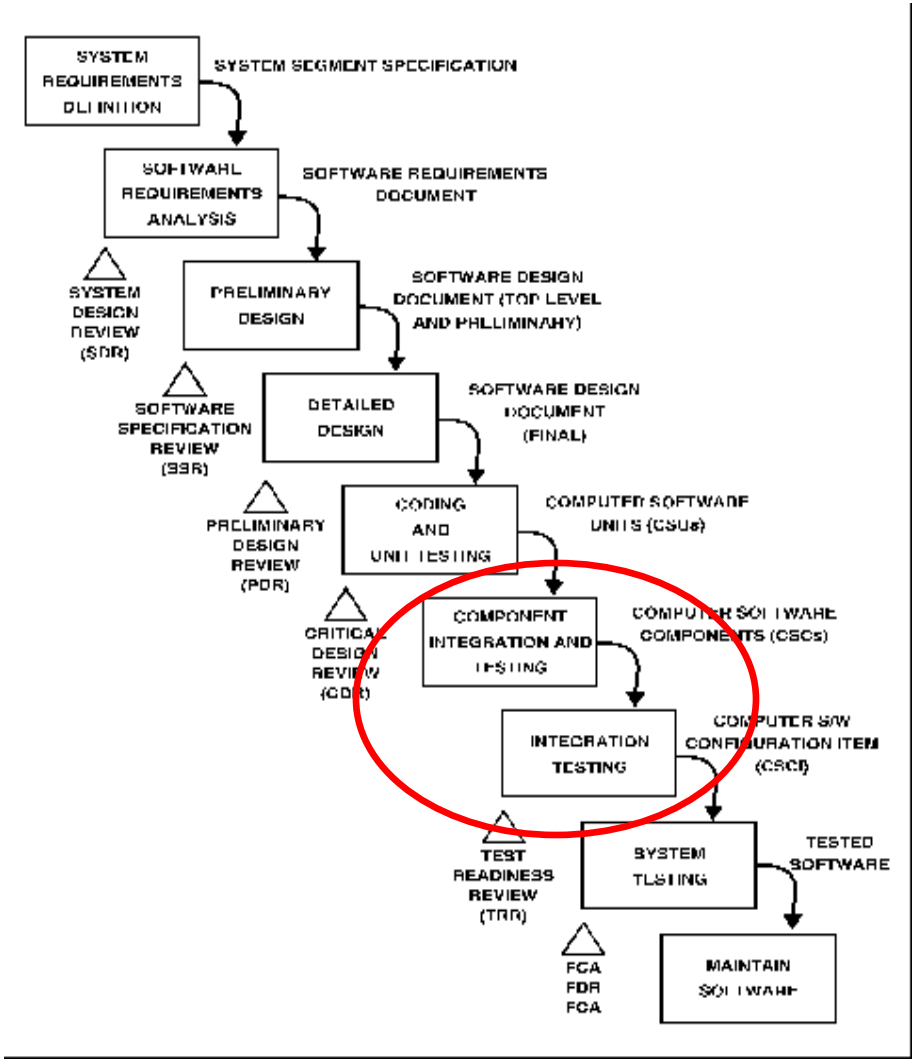
- I progettisti sw usano i requisiti per determinare l'architettura del sistema
- Diversi approcci e metodologie
  - Strumenti
  - Linguaggi di programmazione
  - "librerie"
- Risultato di questa fase:
  - documento di progetto del sistema che identifica
    - tutti i moduli
    - le loro interfacce

Codifica



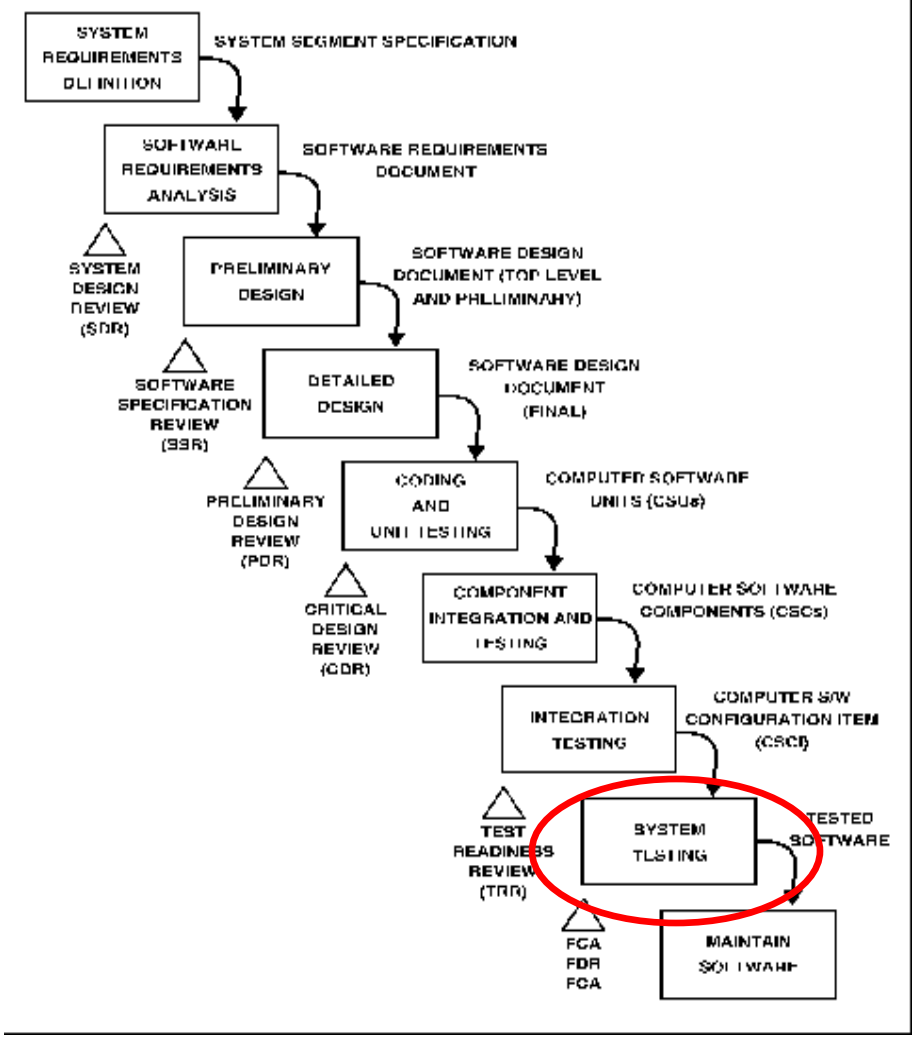
- I singoli moduli sono *implementati* secondo le loro specifiche
- Si usa qualche *linguaggio di programmazione*
- Il singolo modulo è *testato* rispetto alla propria specifica
- Risultato di questa fase:
  - Codice dei moduli
  - Test dei singoli moduli

Integrazione



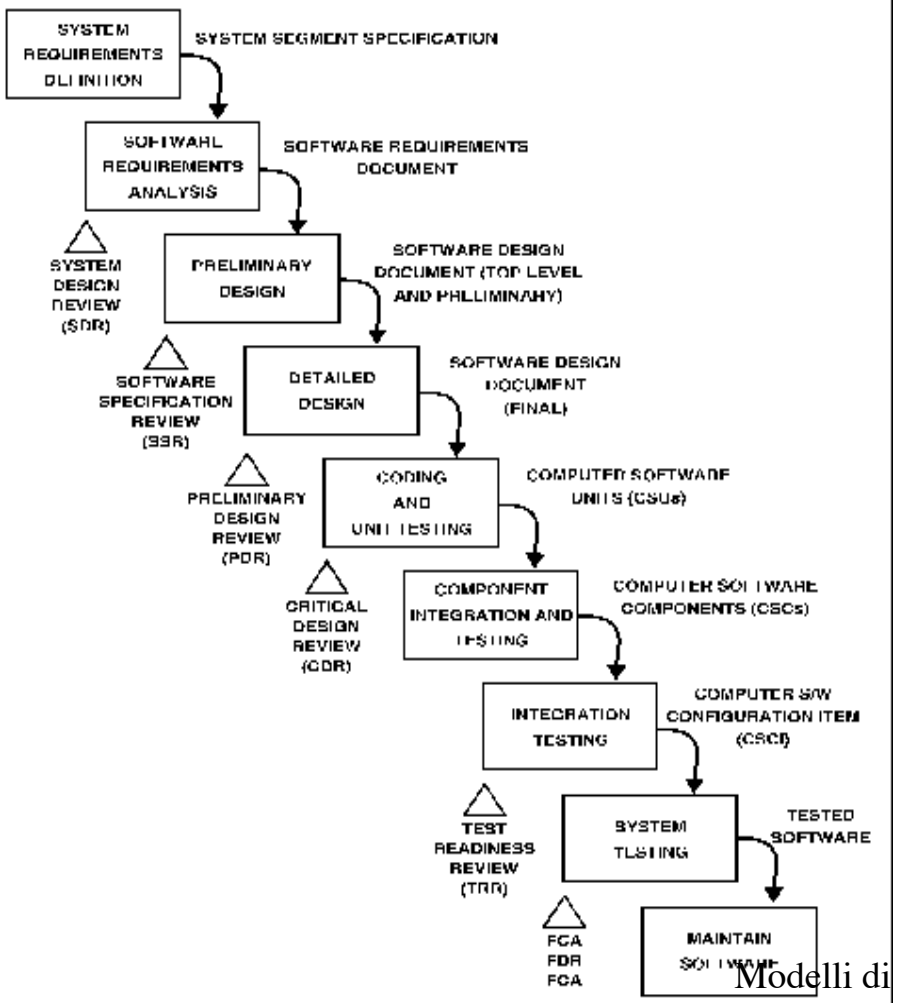
- I moduli sono integrati tra loro
- Testing delle interazioni inter-modulo
- Testing preliminare di sistema
- Risultato di questa fase:
  - il sistema completamente implementato
  - Documento dei test di integrazione

Test del sistema



- Il sistema è testato nella sua globalità e nel suo ambiente d'uso (con gli utenti)
- Accettazione (validazione)
- Risultato di questa fase:
  - un sistema completamente testato, pronto per essere *deployed*

Modello a cascata: aspetti positivi e negativi?

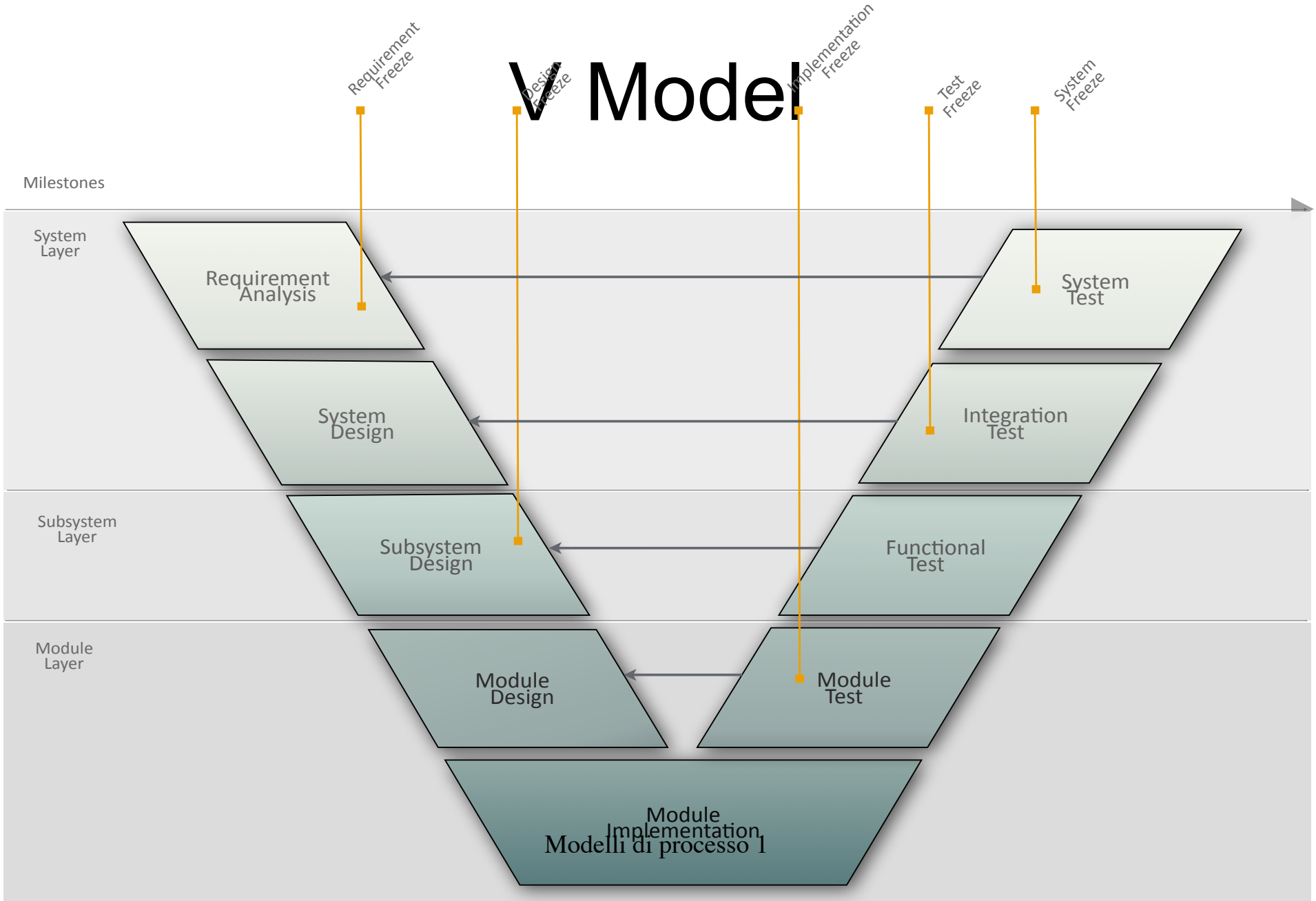


- Pianificato
- Molto dettagliato
- Molto rigido
- Orientato alla documentazione
- Orientato agli standard
- Adatto solo per organizzazioni gerarchizzate (burocrazie)
- Coinvolge il cliente solo alla fine del processo

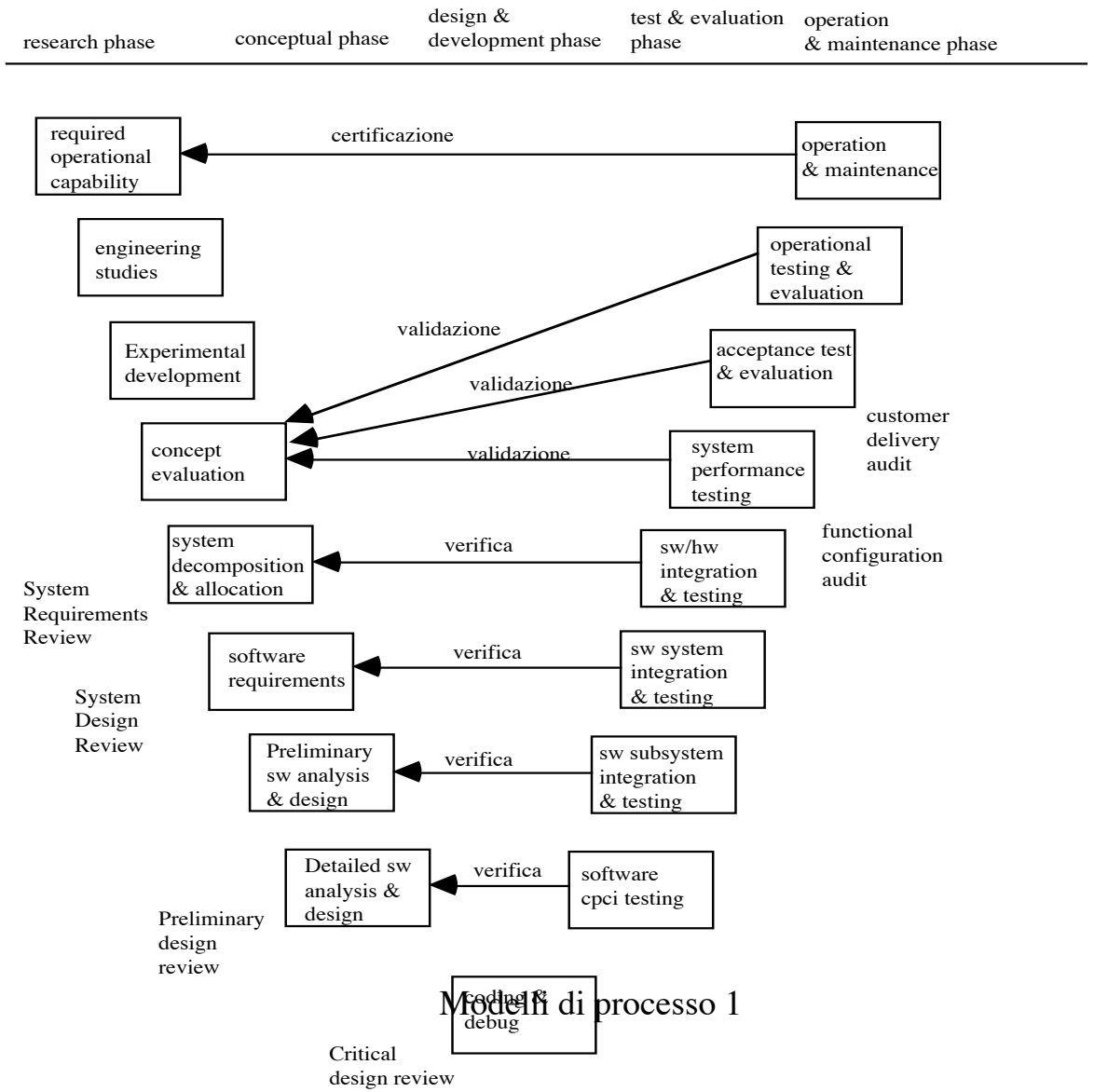
Modello a cascata: aspetti positivi e negativi?

- 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

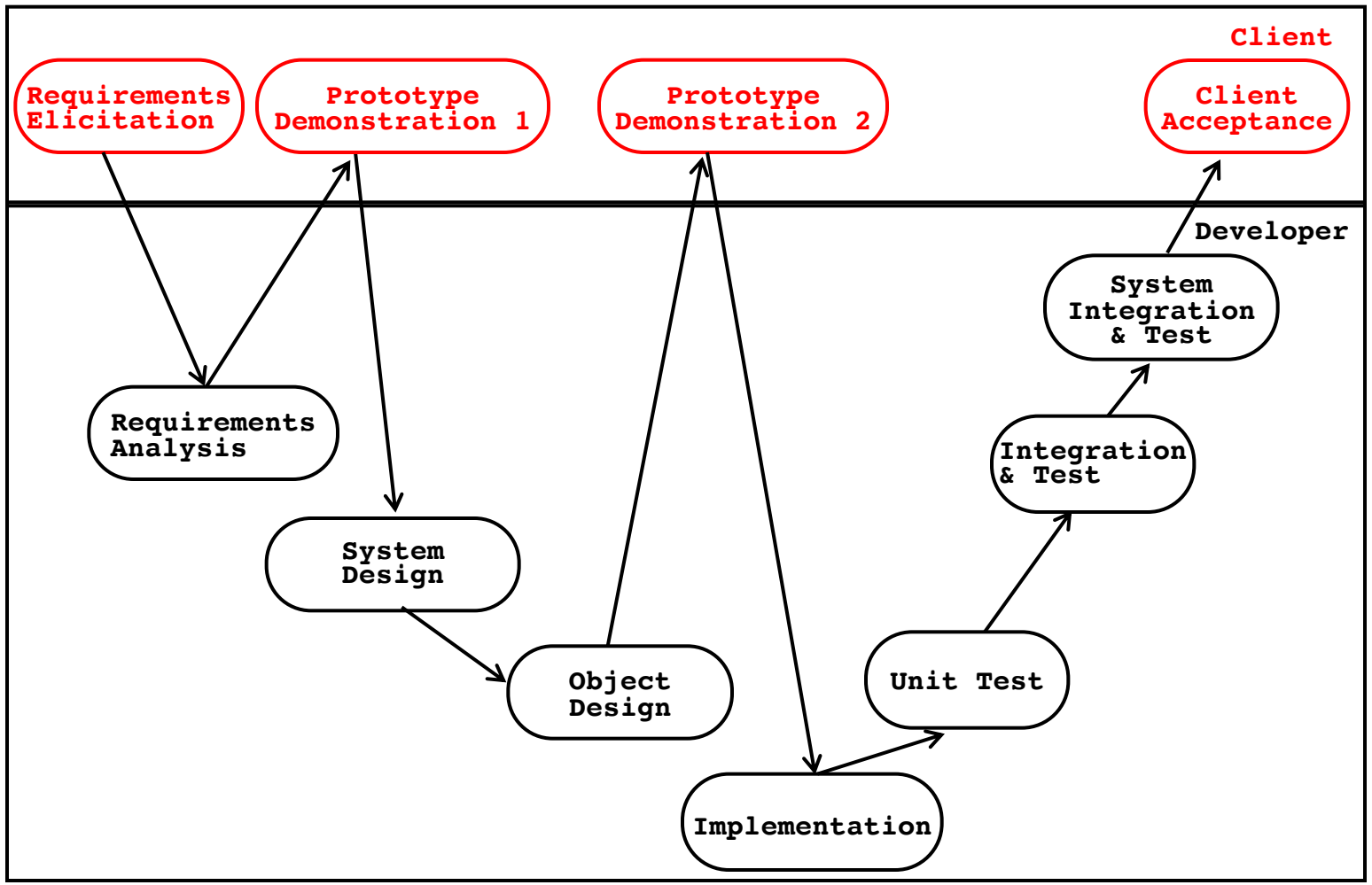
V Model


Requirement Freeze
Design Freeze
Implementation Freeze
Test Freeze
System Freeze
Milestones
System Layer
Requirement Analysis
System Design
Subsystem Layer
Subsystem Design
Module Layer
Module Design
Module Implementation
Module Test
Functional Test
Integration Test
System Test

Modello a cascata, versione a V

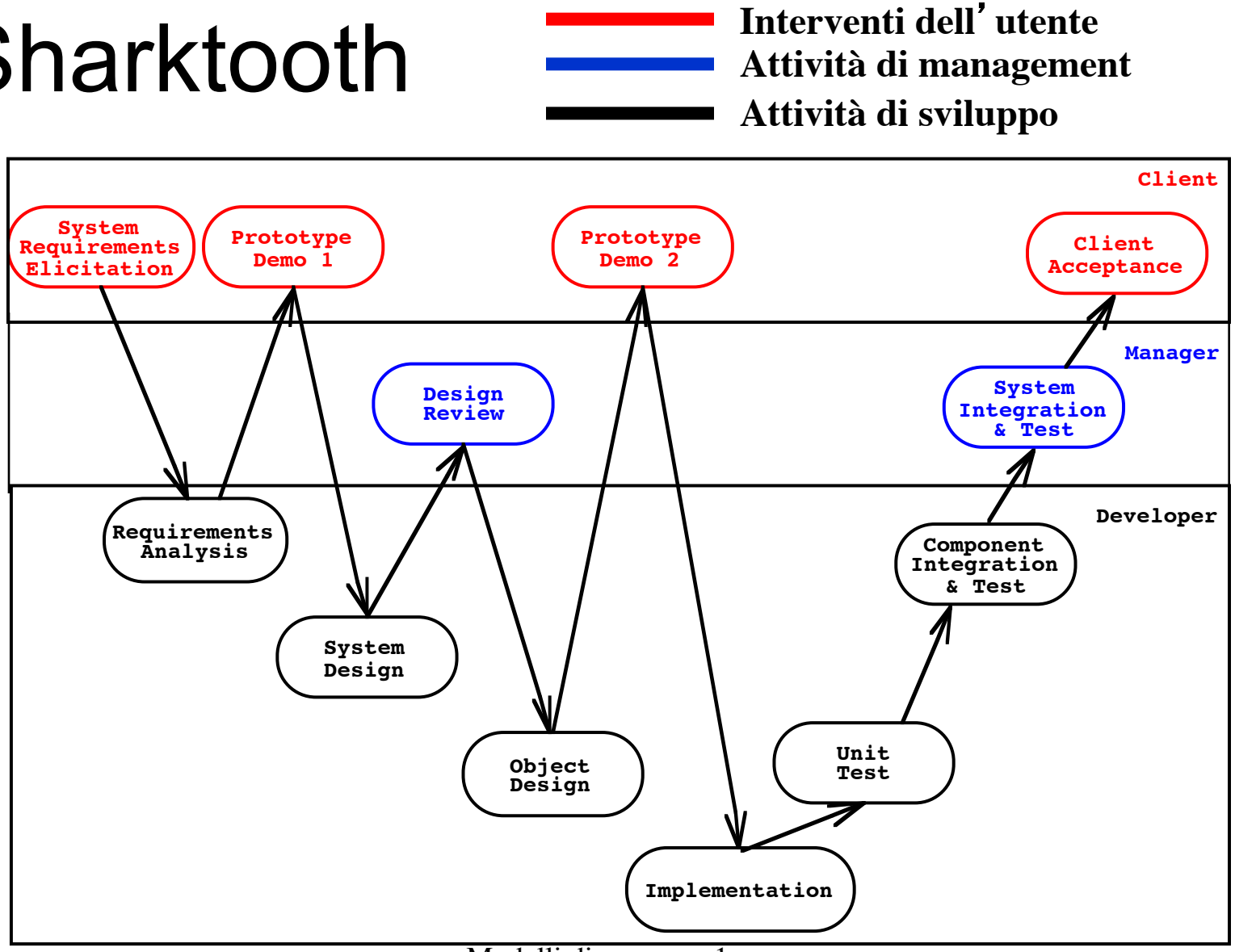


Modello Sawtooth



Modello waterfall con prototipazione (revolutionary and evolutionary)

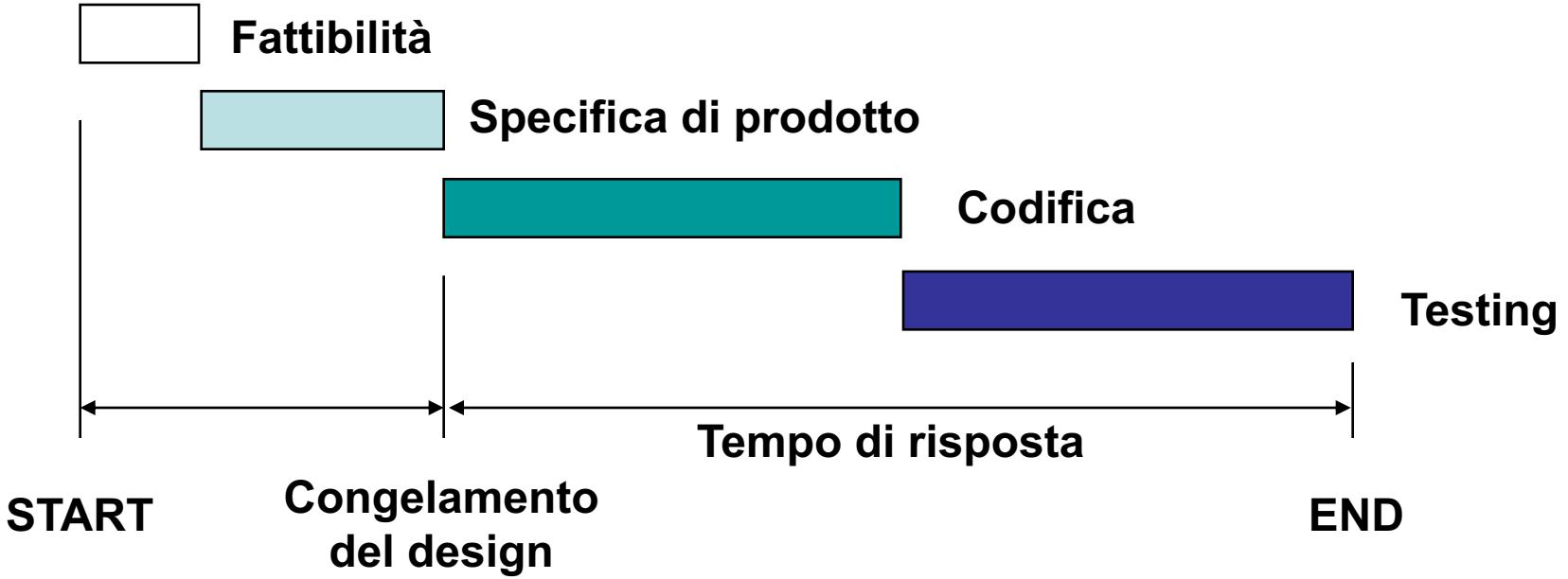
Modello Sharktooth



Waterfall con prototipazione, usabile per outsourcing

Problema dei processi lineari

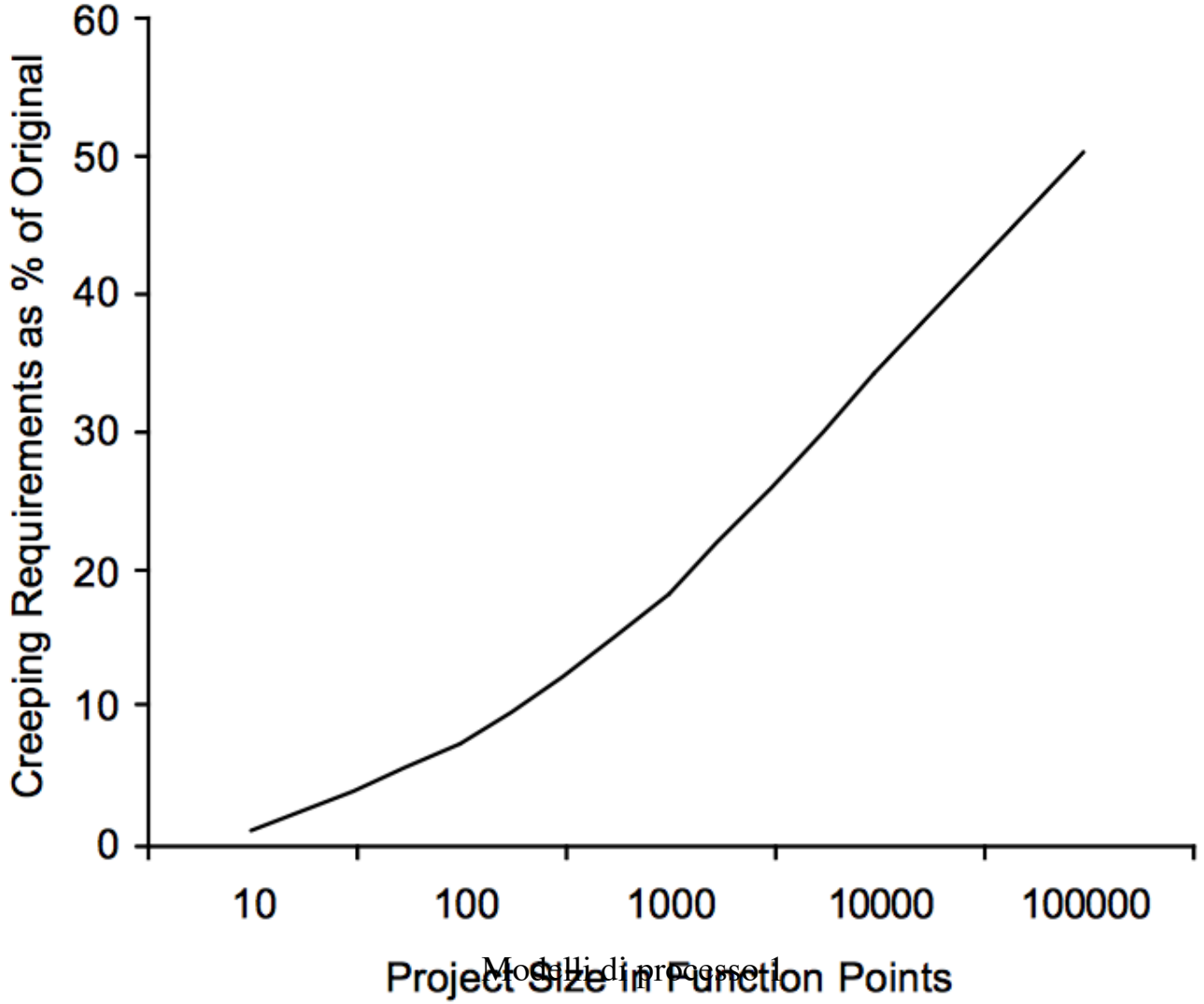
Modello di sviluppo rigidamente sequenziale



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

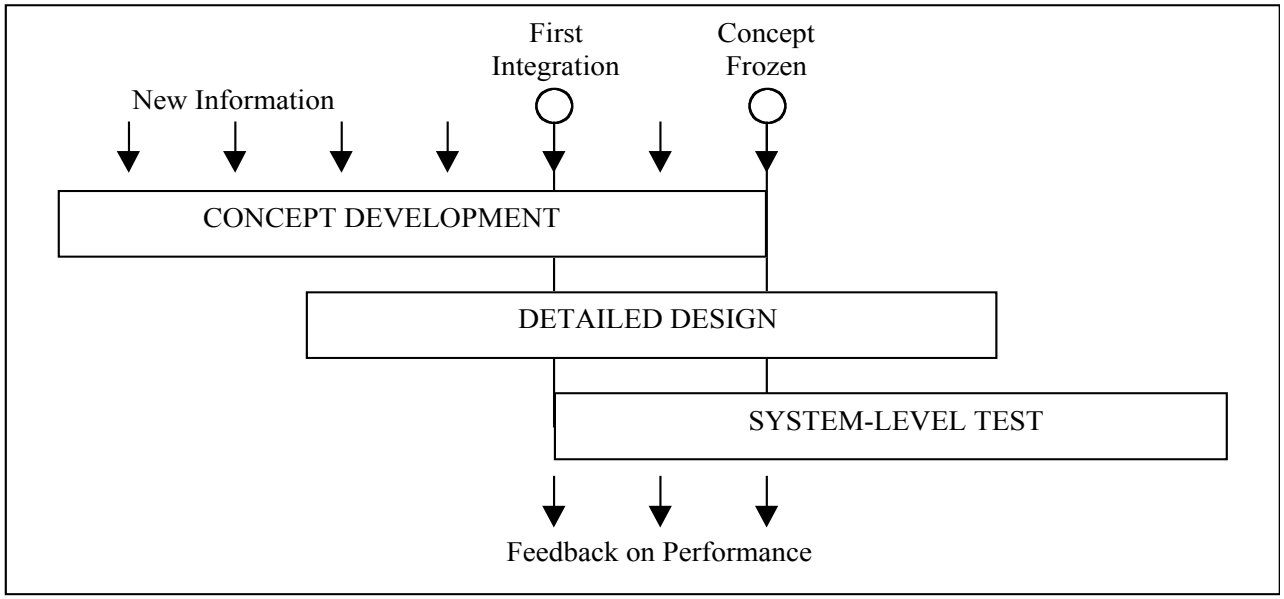
I requisiti cambiano durante lo sviluppo

(più il sistema è grosso più cambiano!)



C.Jones. Applied Software Measurement, McGraw-Hill

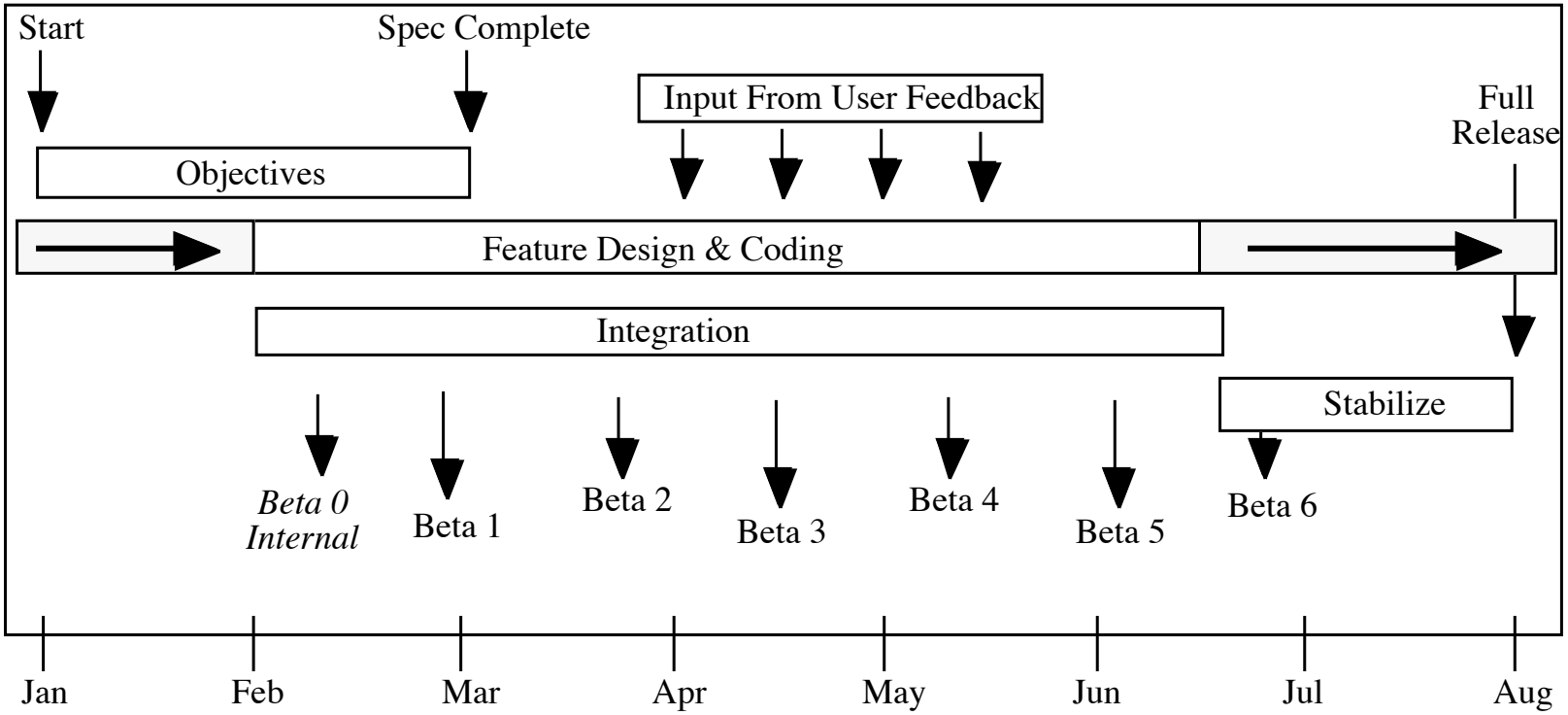
Un modello più flessibile: fasi sovrapposte



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

Esempio

Progetto Netscape's Navigator 3.0

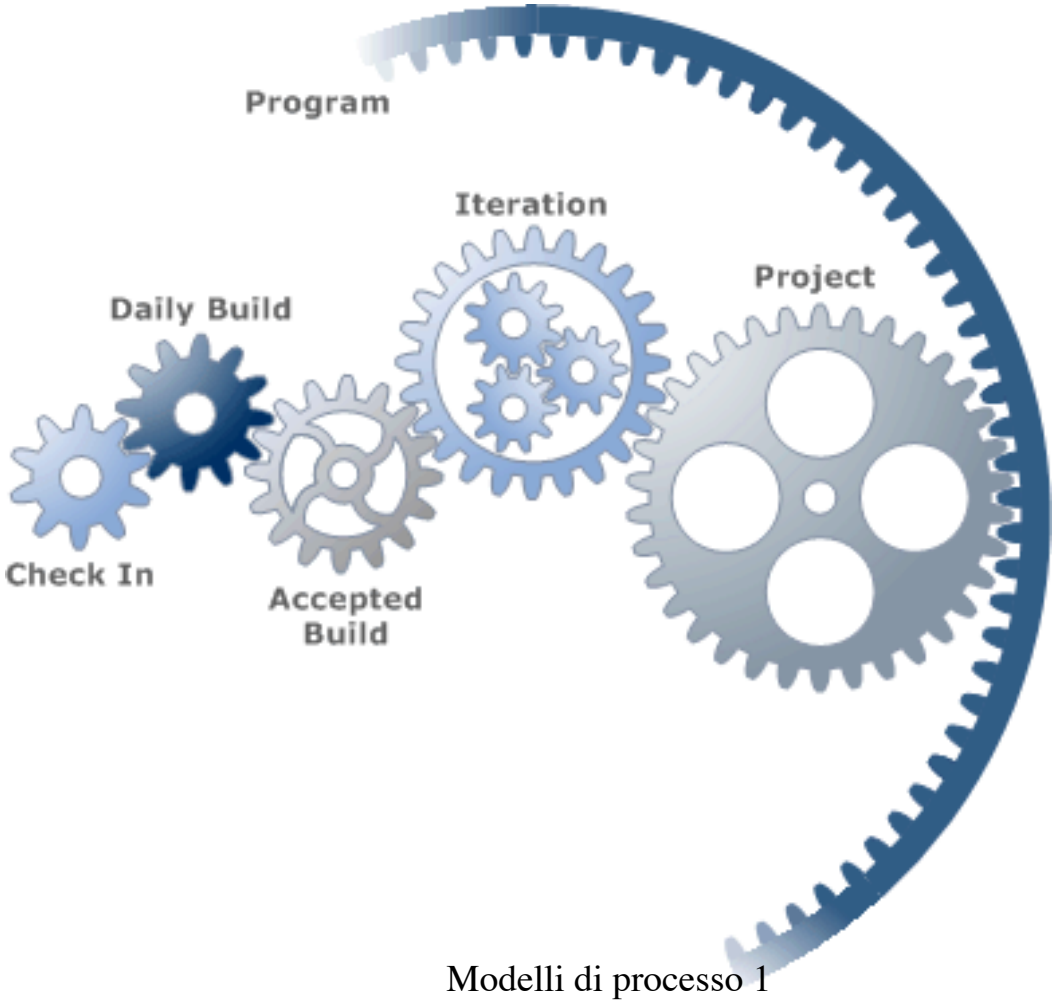


>50% di nuovo codice sviluppato dopo il primo rilascio beta!

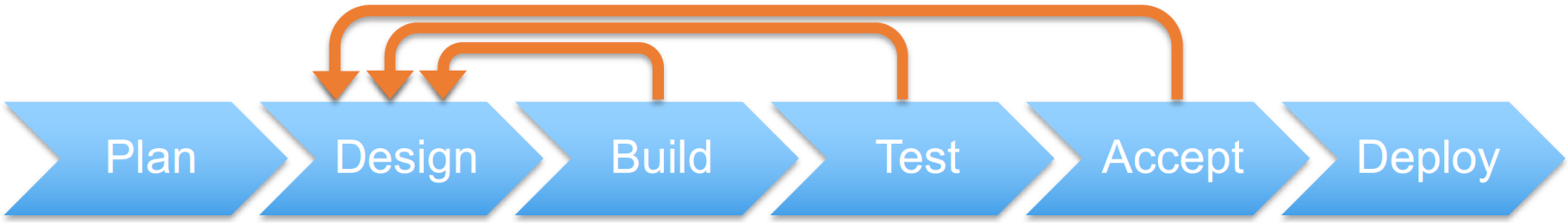
Dimensioni del tempo

- Tempo lineare
  - "Oggi più di ieri e meno di domani"
  - La nozione di progresso
- Tempo ciclico
  - Ore, giorni, settimane, mesi, stagioni…
  - "Sentinella, a che punto è la notte?" Isaia, 21

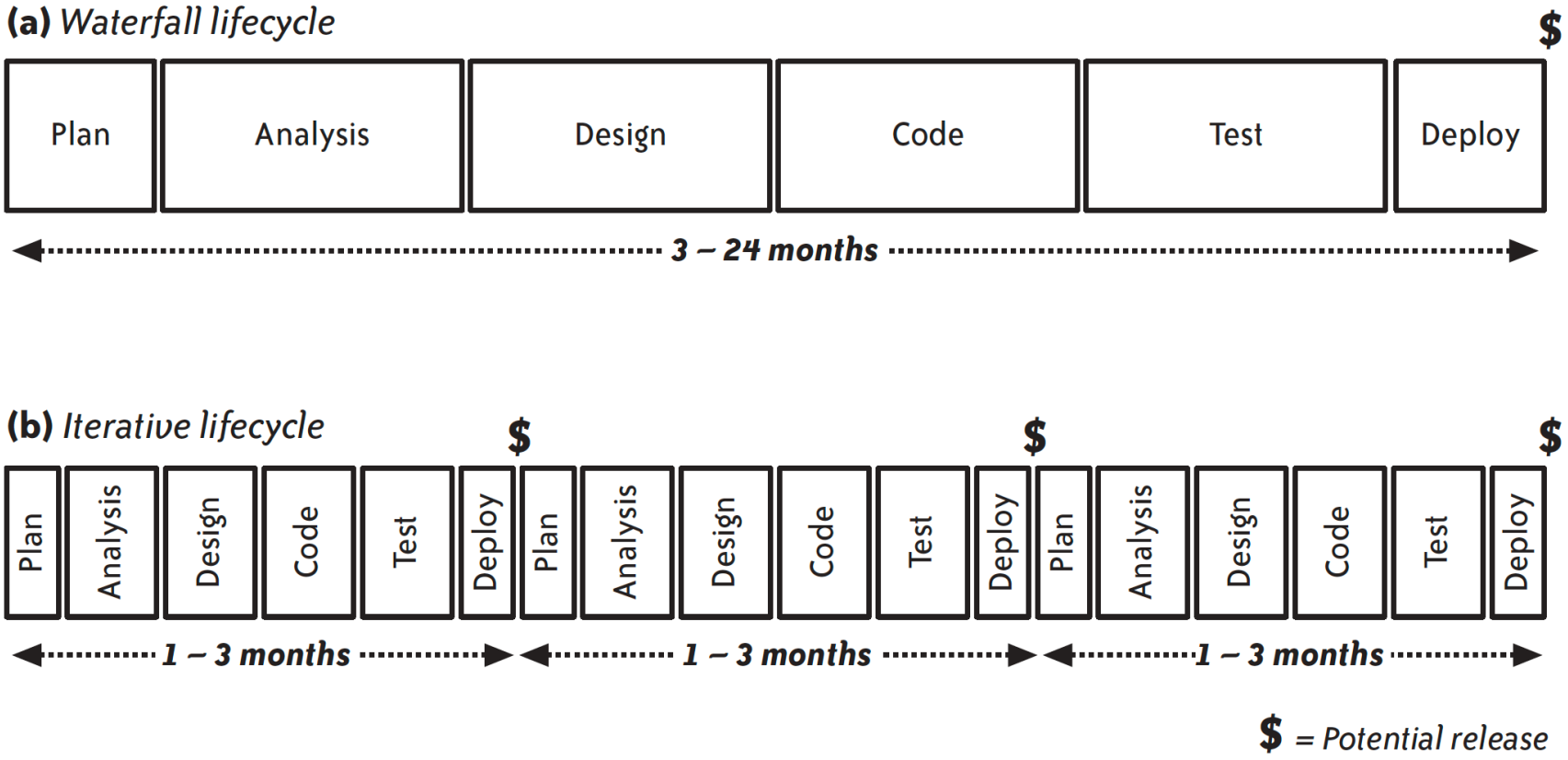
Il software si costruisce in cicli (e cicli dentro cicli)



Cicli dentro cicli



Waterfall vs iterativo

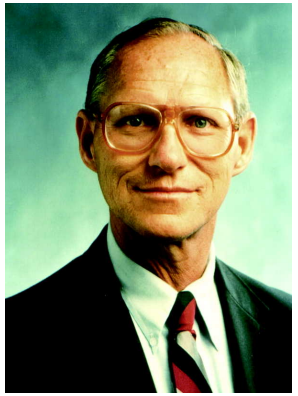


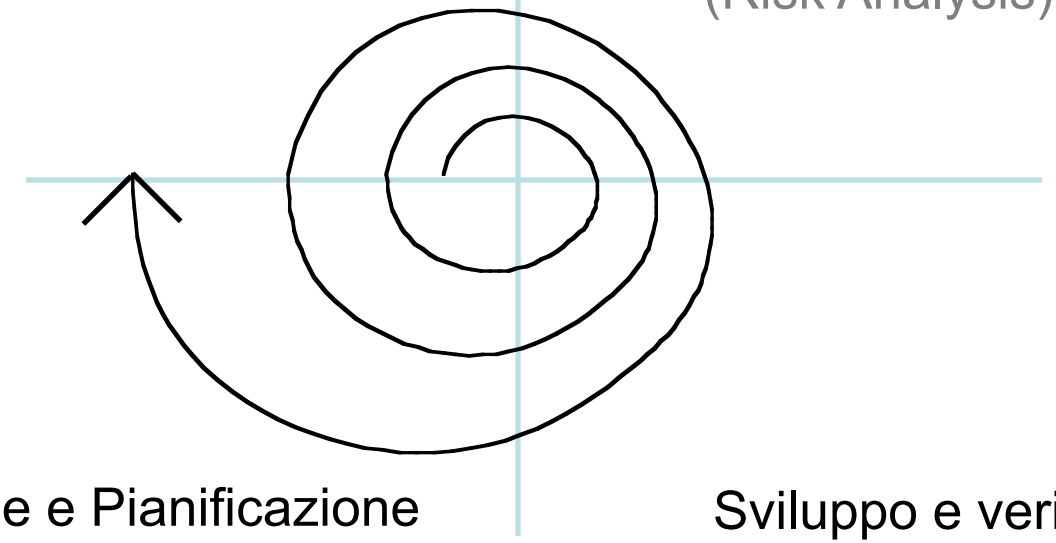
Modello a spirale

- Modello **iterativo** in cui **i rischi vengono valutati continuamente ed esplicitamente**

- Il processo viene modellato da una spirale (e non da una sequenza di attività)
- Ogni spira della spirale è una **iterazione** del processo e prevede diverse attività, divise in quattro fasi:
  - Planning, Risk Analysis, Engineering, Evaluation

Modello a spirale (Boehm)

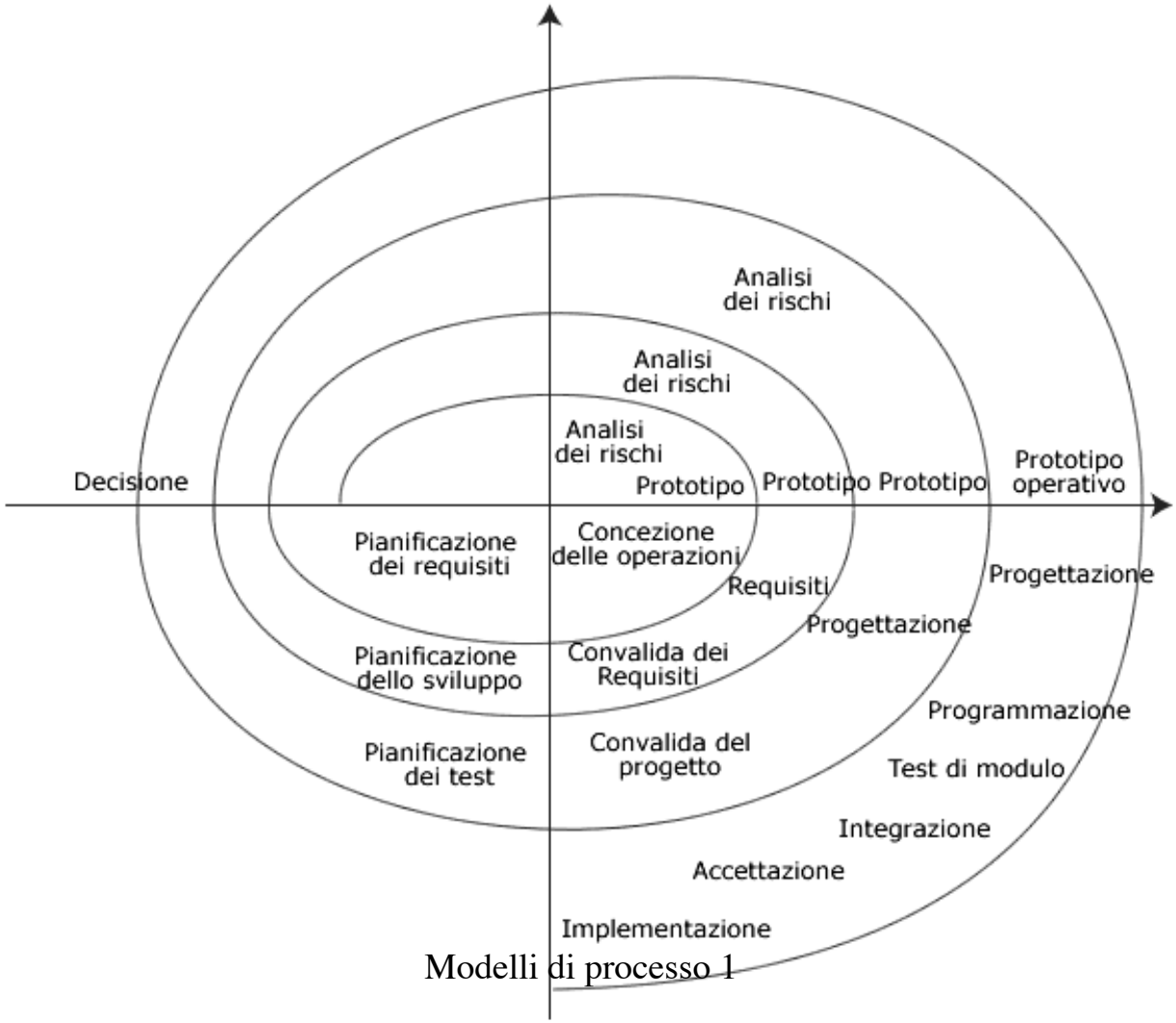




I settori del modello a spirale

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

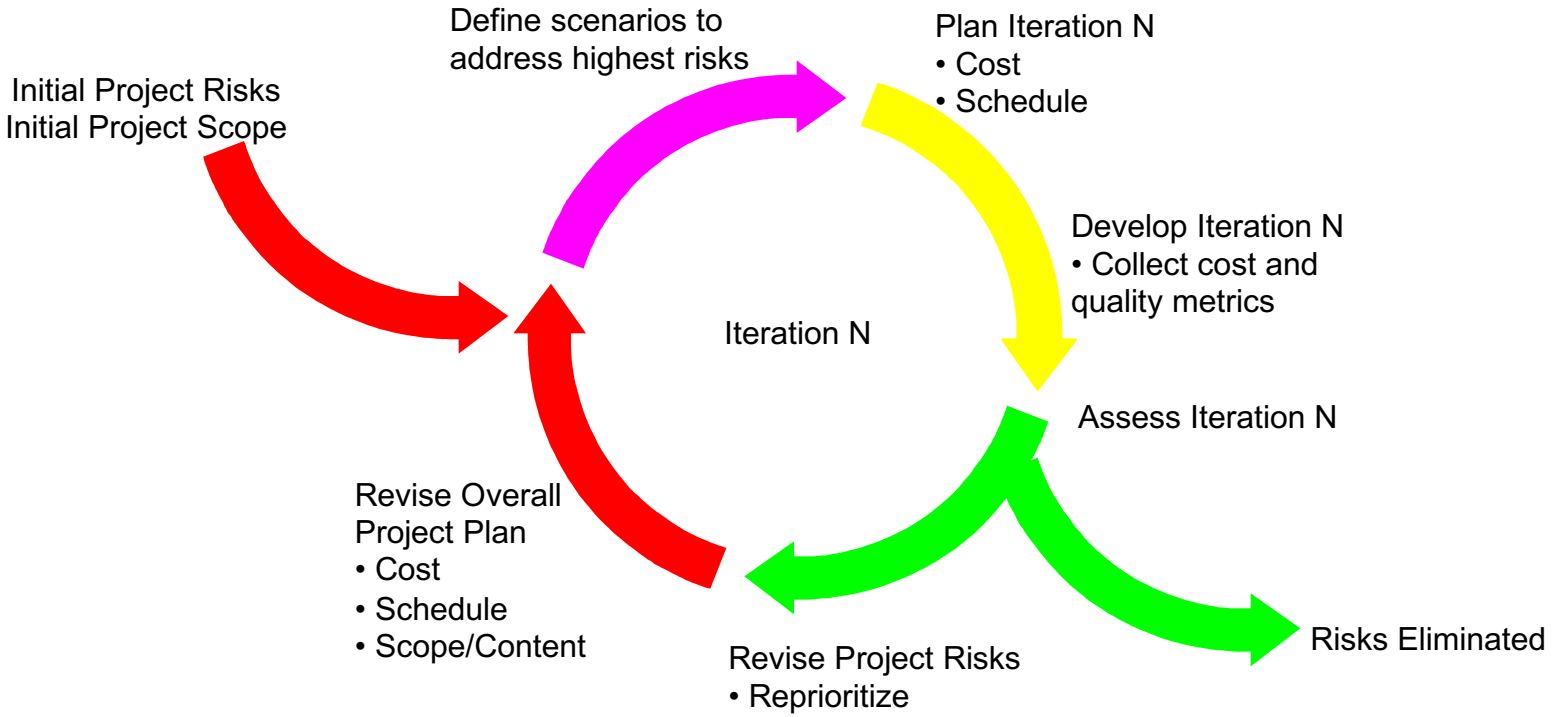
Modello a spirale



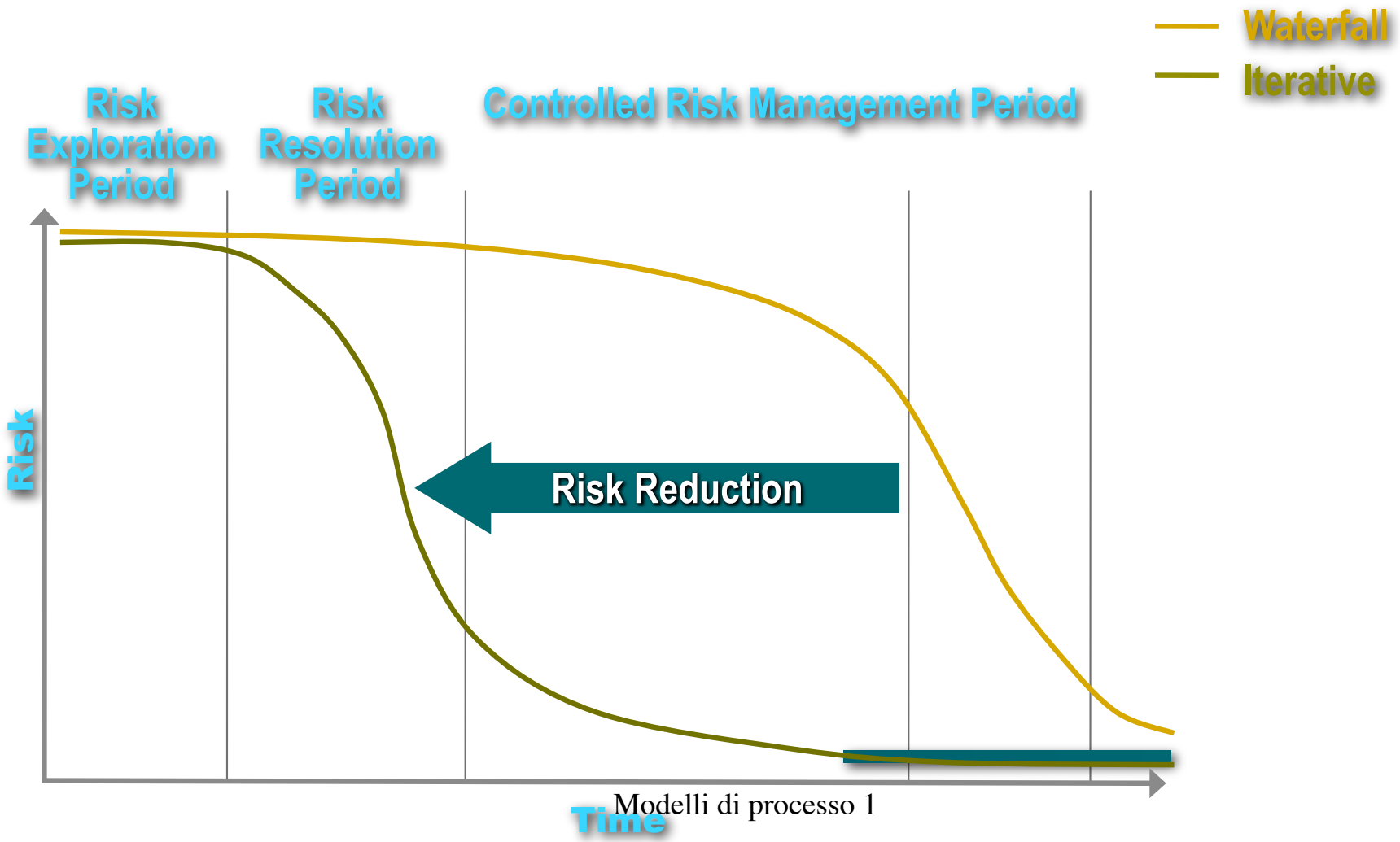
Modello a spirale

- 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)?

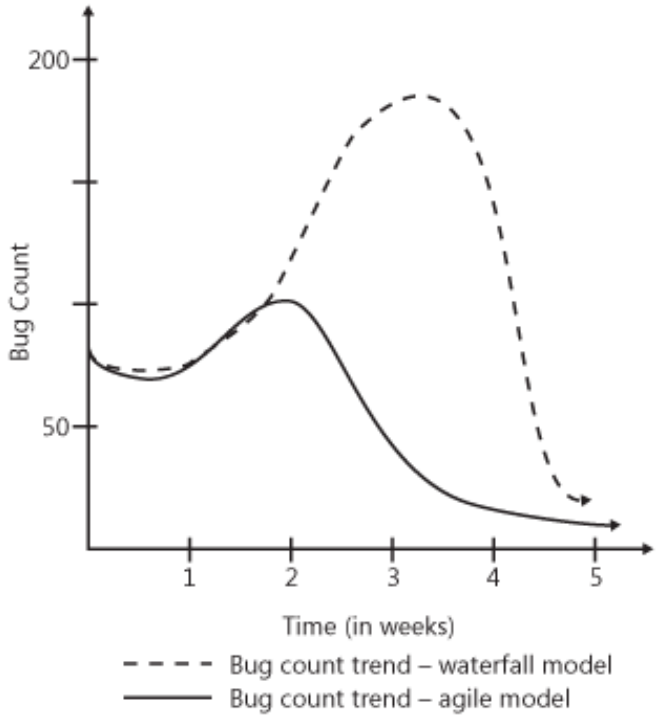
I processi iterativi diminuiscono i rischi



Cascata vs Iterativi

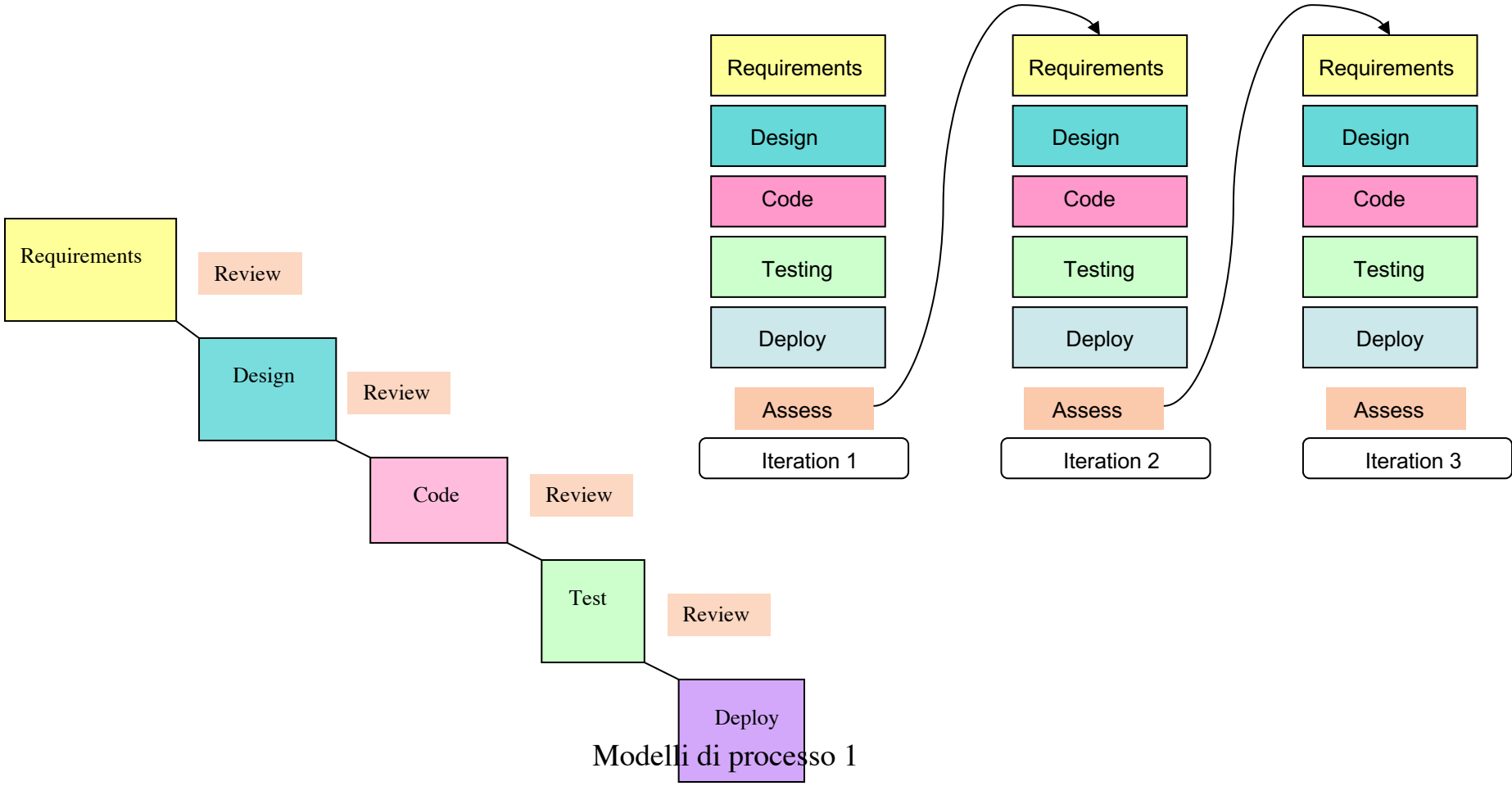


La diminuizione del rischio

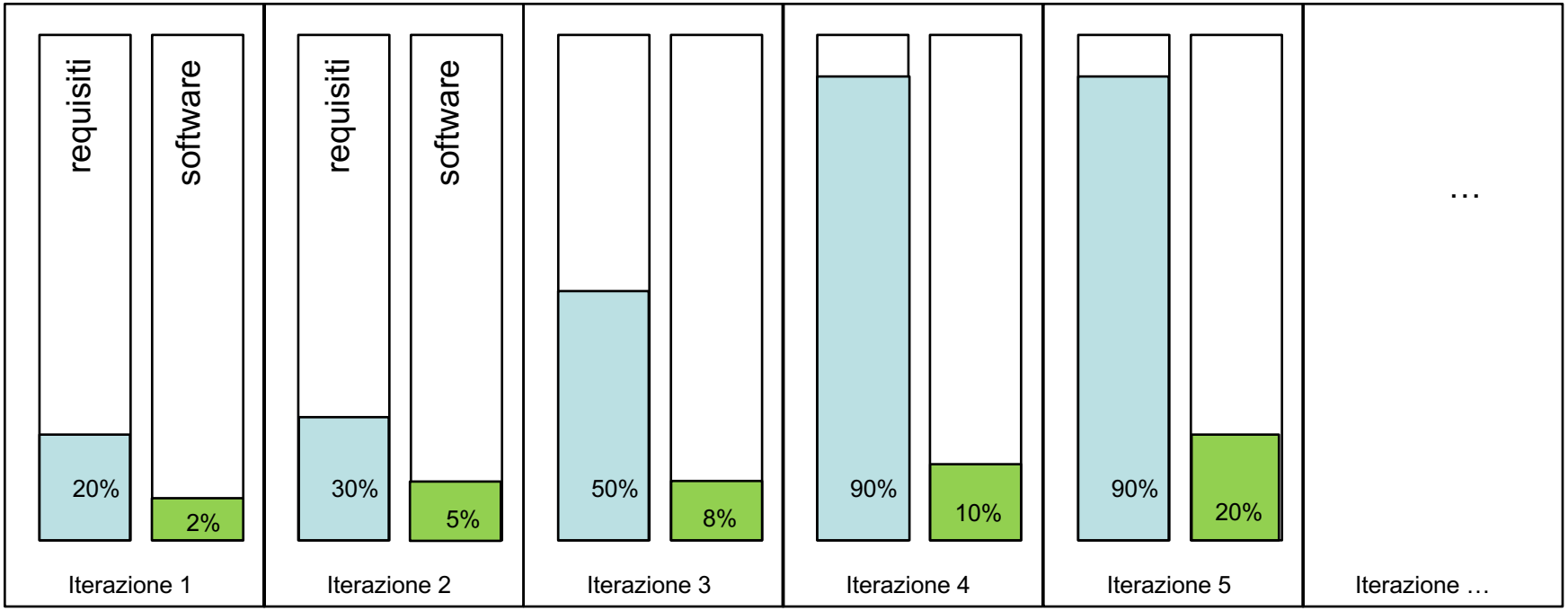


Windows Live Hotmail bug trend comparison, da Marshall, Solid Code, O'Reilly 2009

Cascata vs iterativi

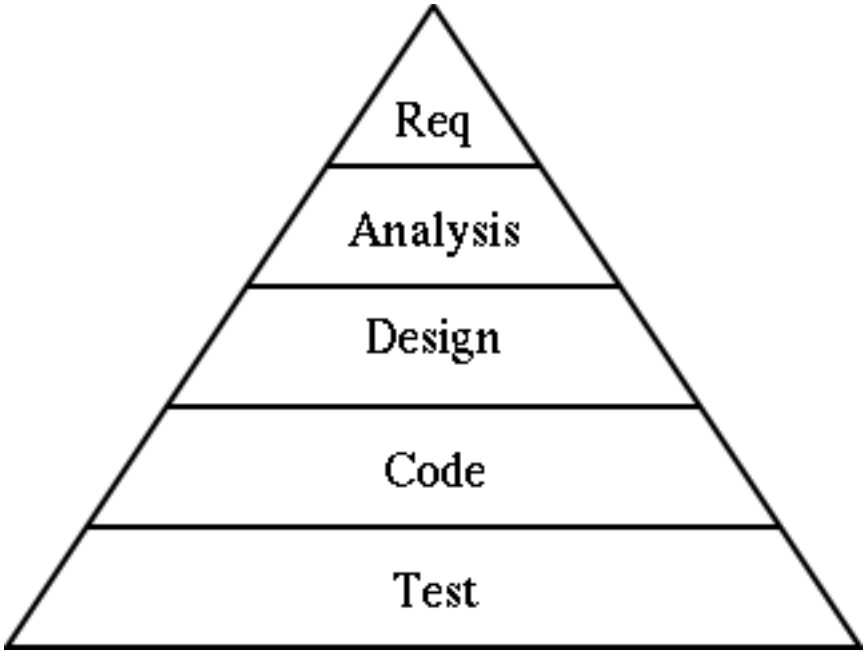


Iterazioni

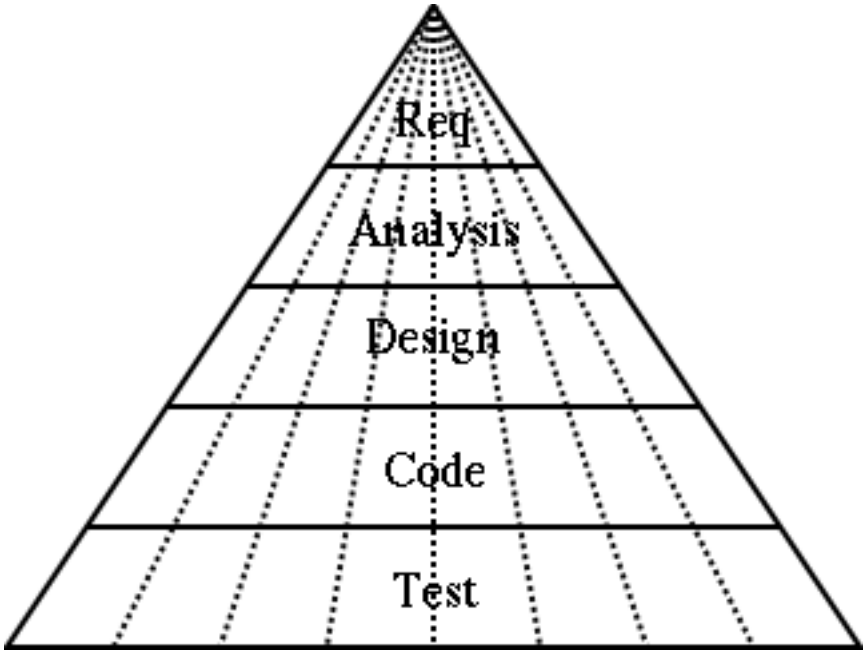


- Questa figura rappresenta un processo iterativo da oltre 5 iterazioni
- Alla fine della quarta iterazione il 90% dei requisiti è stabile ma solo il 10% del software è stato costruito

Cascata vs iterativi: sforzo

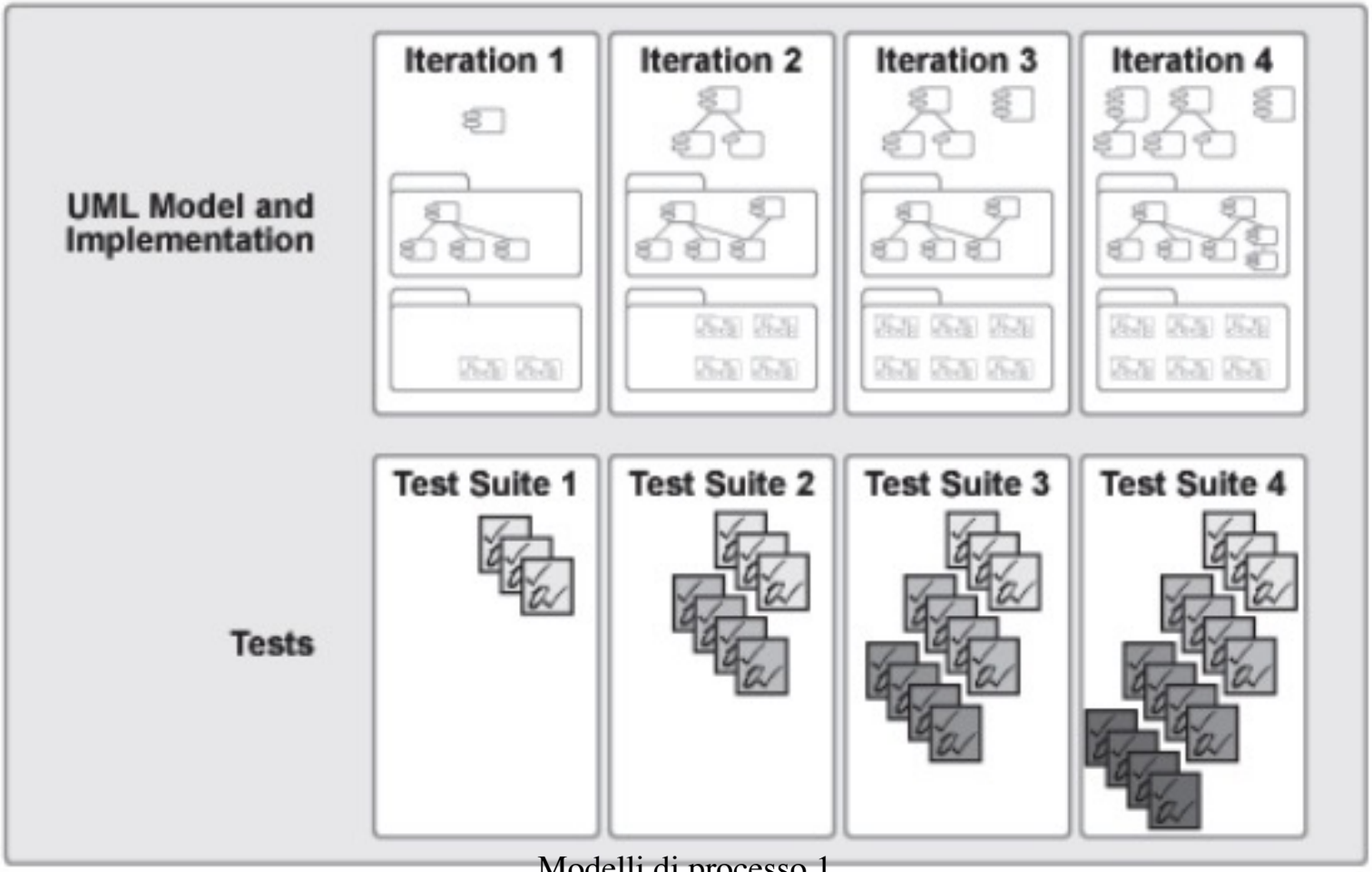


Distribuzione dello sforzo nelle fasi di un processo a cascata: il testing assorbe molto più sforzo delle altre fasi



Segmentazione dello sforzo nelle fasi di un processo iterativo

Iterativi: testing incrementale



Processi iterativi

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

RUP

- Modello di processo di tipo iterativo e incrementale, diviso in quattro fasi
  I. *Inception* (fattibilità)
  II. *Elaboration* (progettazione)
  III. *Construction* (codifica e test)
  IV. *Transition* (deployment)
- Articolato su diverse discipline (workflows)
- Supportato da strumenti proprietari IBM (esempio: Rational Rose)

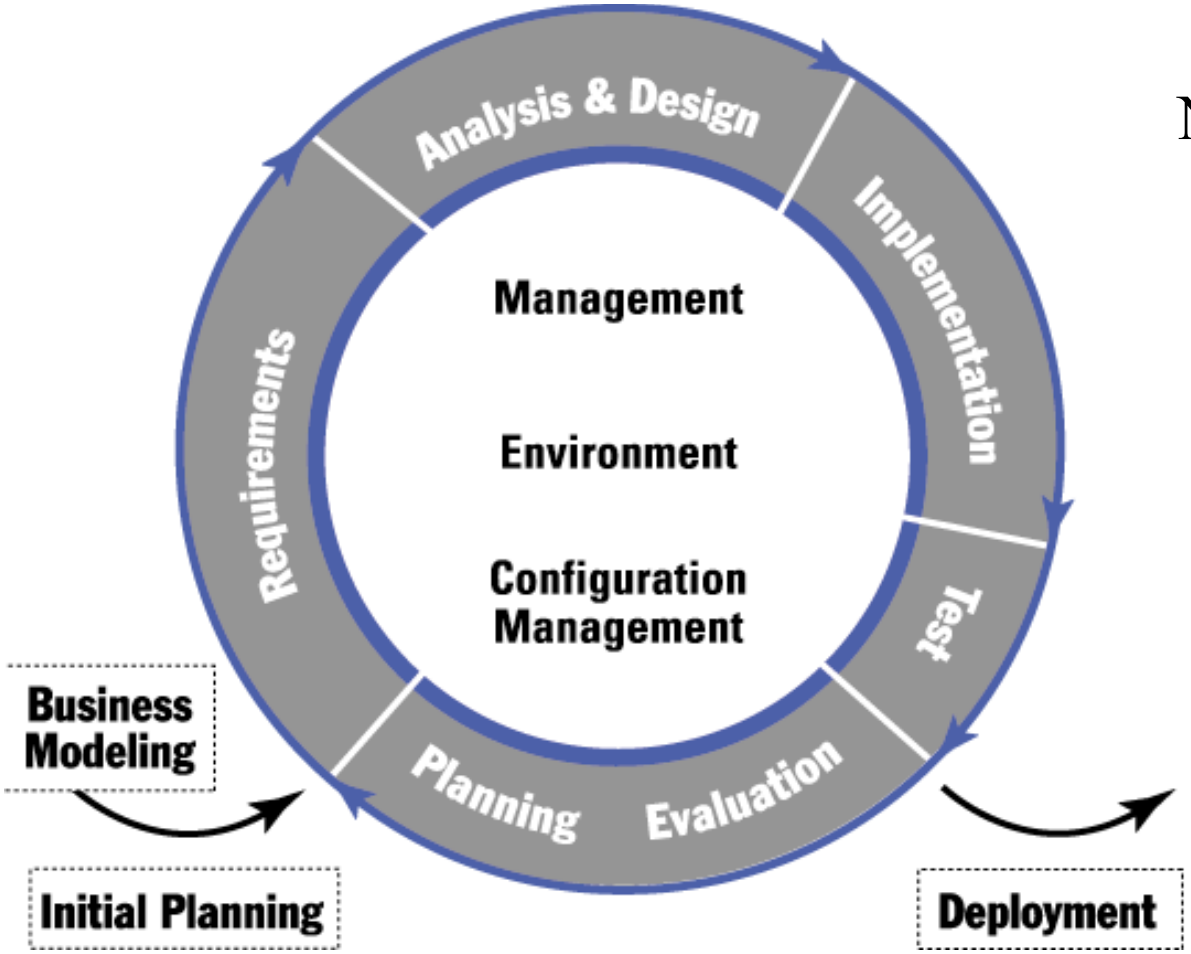
RUP: piccola storia

- Jacobson propose all'inizio degli anni '90 un metodo per progettazione ad oggetti chiamato Objectory
- Nel 1996 Objectory venne scelto da Rational come processo di riferimento col nome USDP Unified Sw Development Process
- In seguito Rational registrò questo processo col nome di RUP™ (TM = trade mark cioè marchio depositato)
- Nel 2001 IBM comprò Rational e continua a supportare RUP™
- I processi che si ispirano a RUP™ di solito si chiamano UP (per es.: Open UP oppure Enterprise UP)

Dunque

Objectory = USDP = RUP™ = UP

Il RUP è un processo iterativo

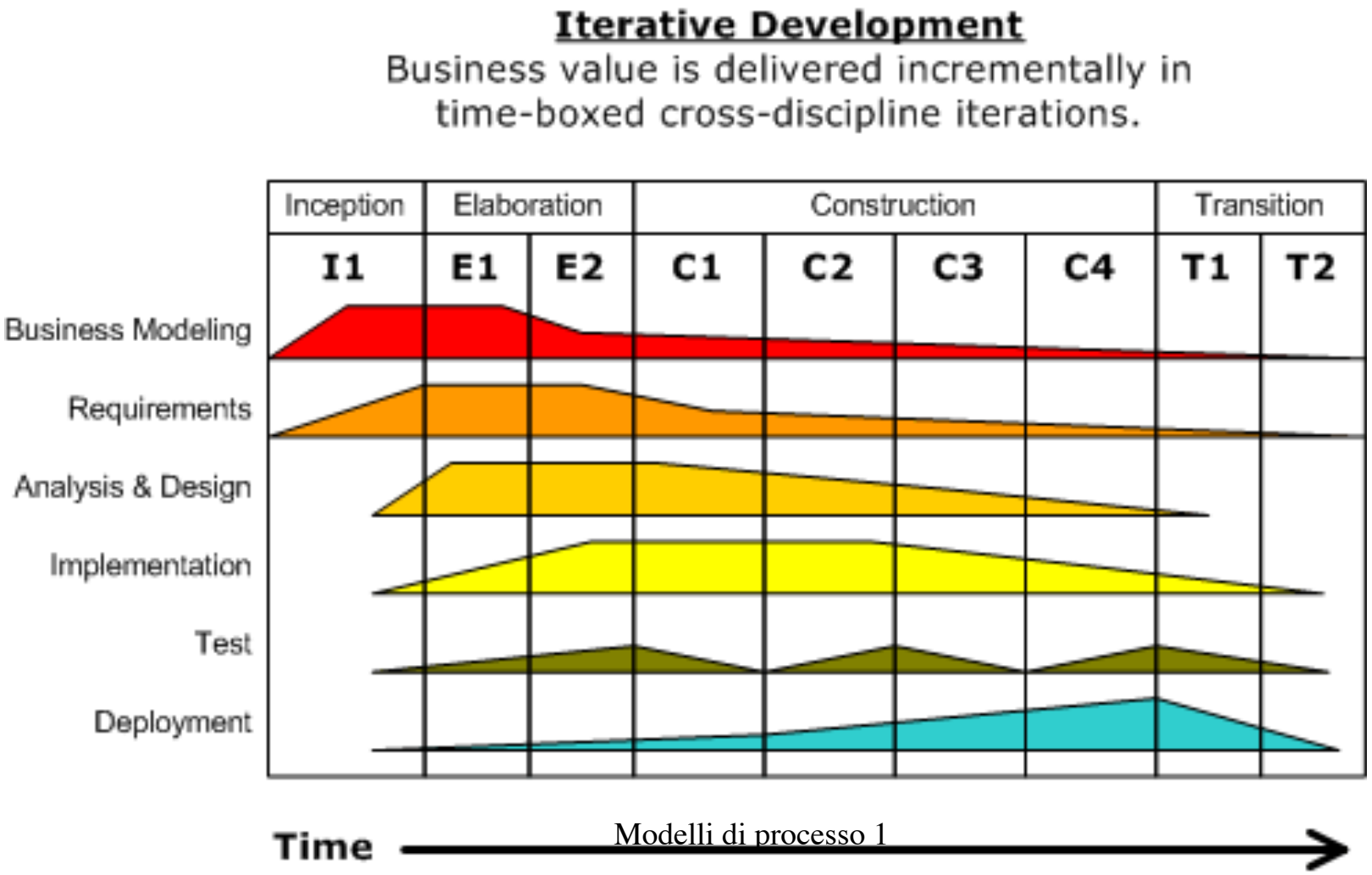


Nel tempo…

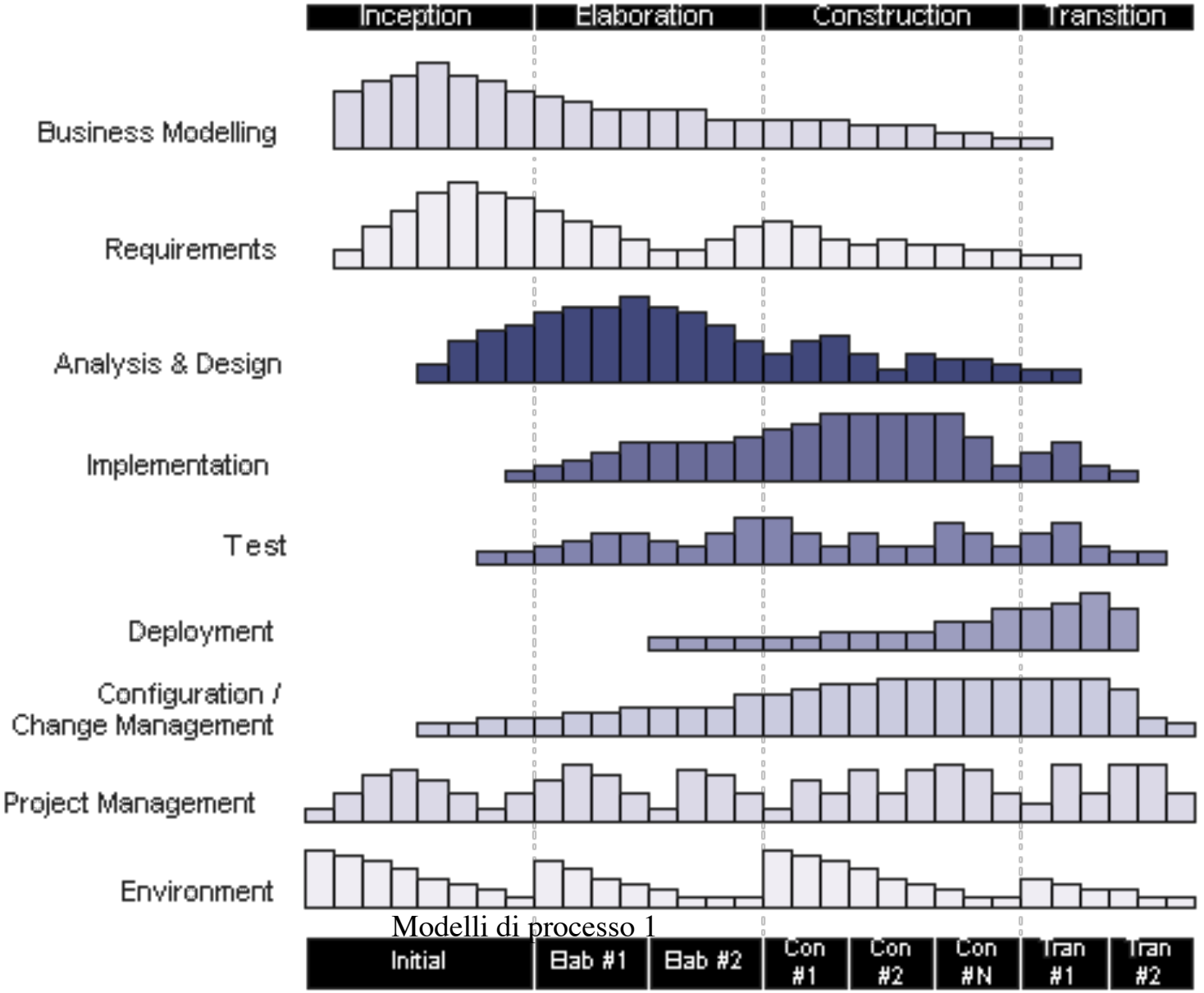
Iterazione: sequenza di *attività* con un *piano* prestabilito e dei *criteri* di valutazione, che termina con un *rilascio* eseguibile

http://www-306.ibm.com/software/rational/sw-library/

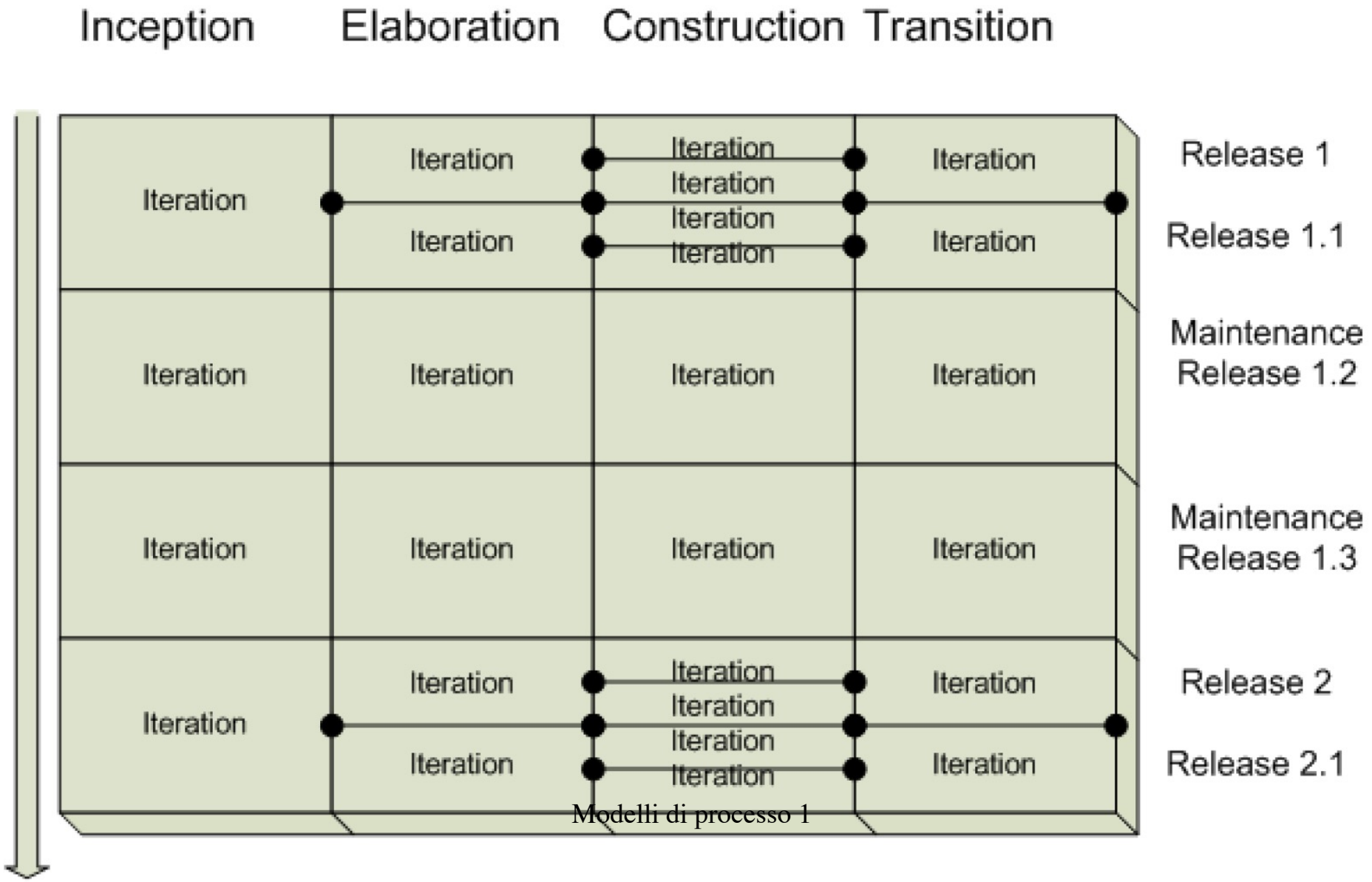
RUP: visione grafica



RUP: visione grafica 2



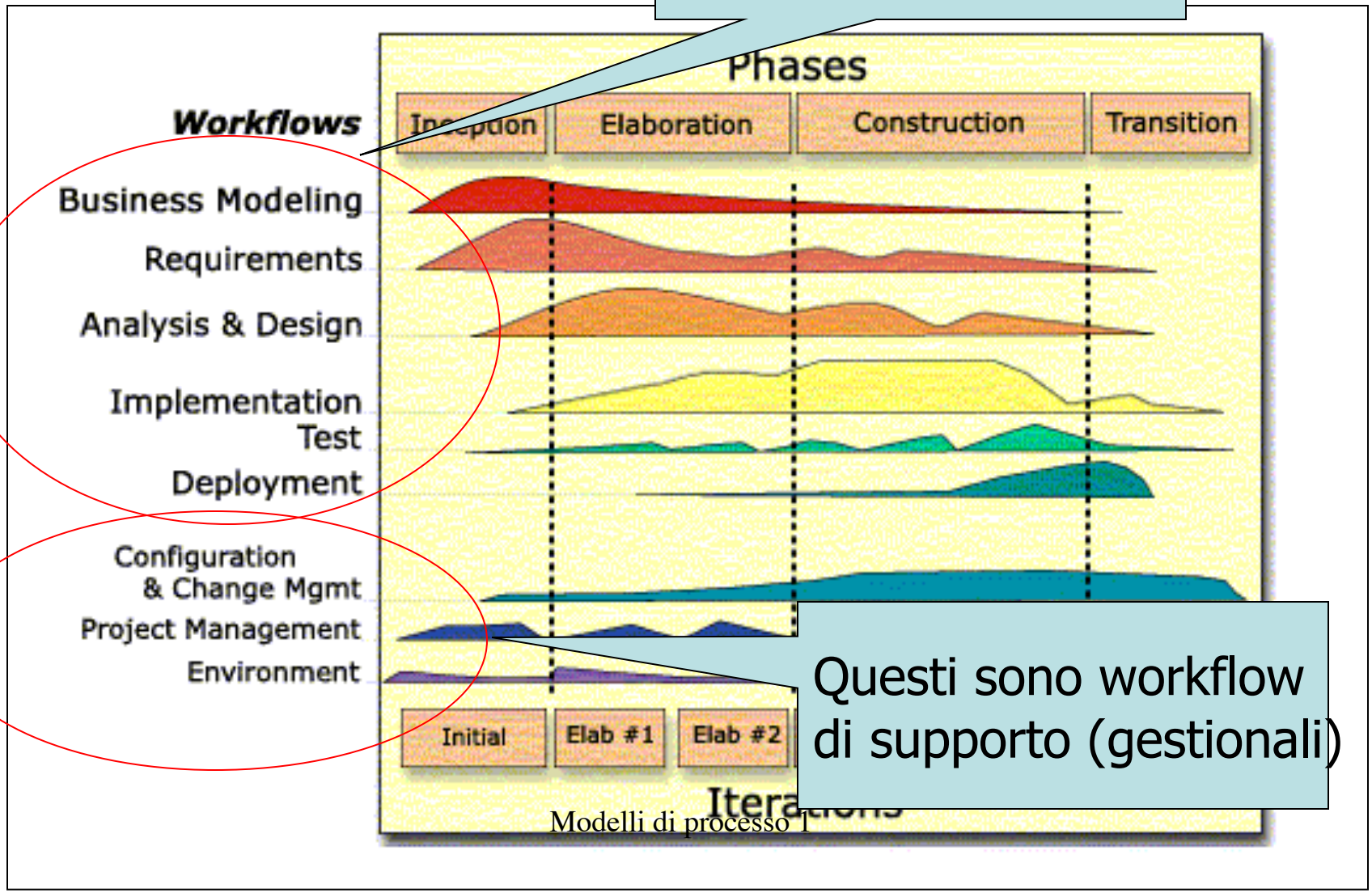
RUP: visione grafica 3



Il RUP ha due prospettive

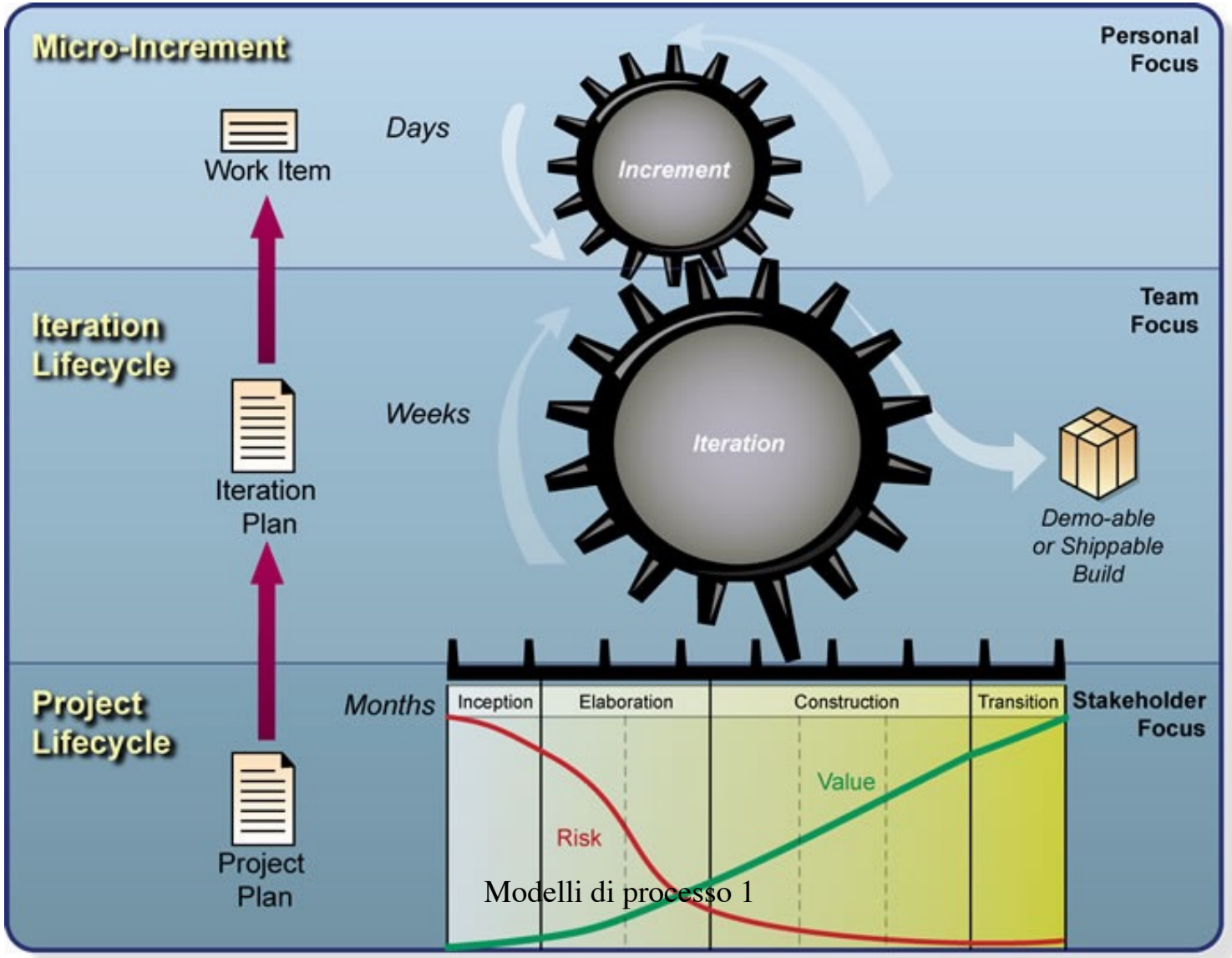
- Il processo RUP integra due diverse prospettive:
  - Una ***prospettiva tecnica***, che tratta gli aspetti qualitativi, ingegneristici e di metodo di progettazione
  - Una ***prospettiva gestionale***, che tratta gli aspetti finanziari, strategici, commerciali e umani
- Le due prospettive sono rispettivamente articolate su sei e tre "**core workflow**"

Struttura

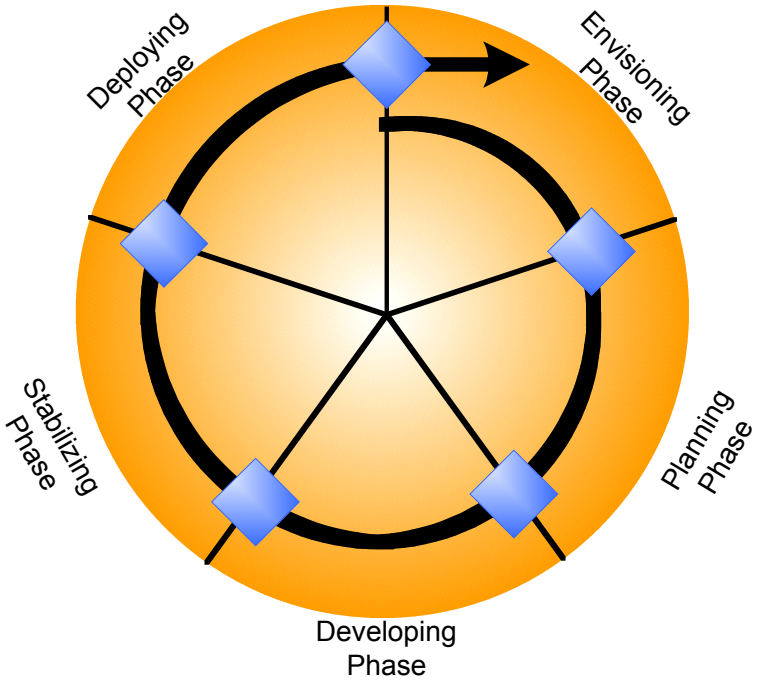

Questi sono workflow di processo (tecnici)
Phases
Workflows
Inception
Elaboration
Construction
Transition
Business Modeling
Requirements
Analysis & Design
Implementation
Test
Deployment
Configuration & Change Mgmt
Project Management
Environment
Initial
Elab #1
Elab #2
Iterations
Questi sono workflow di supporto (gestionali)

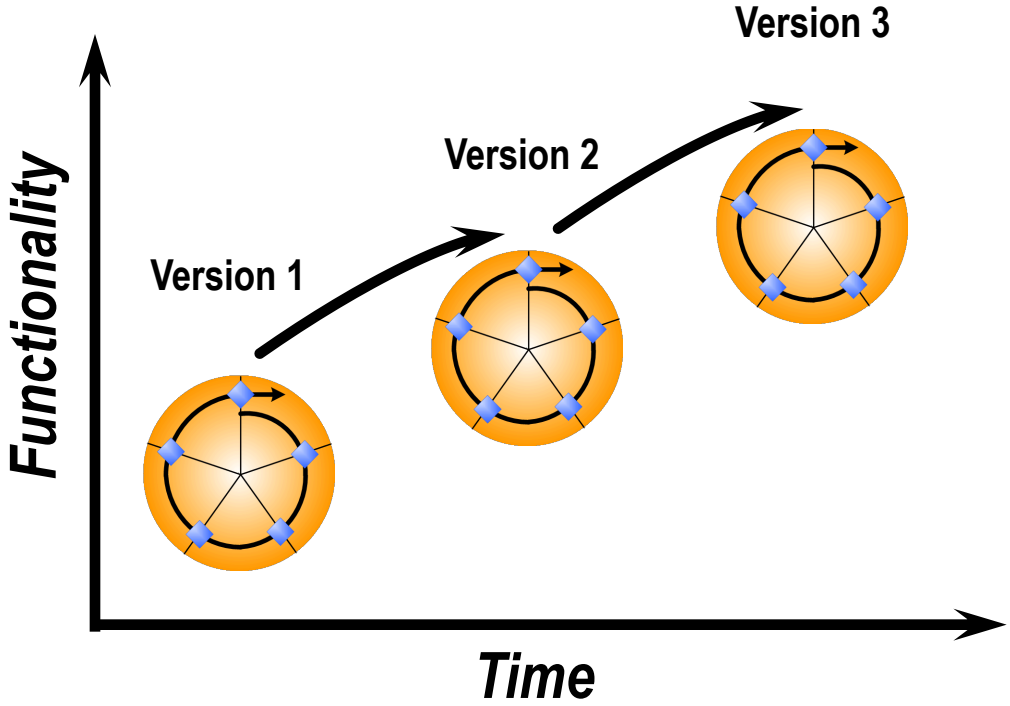
Open UP

http://epf.eclipse.org/wikis/openup/



Anche MSF è un processo iterativo e incrementale



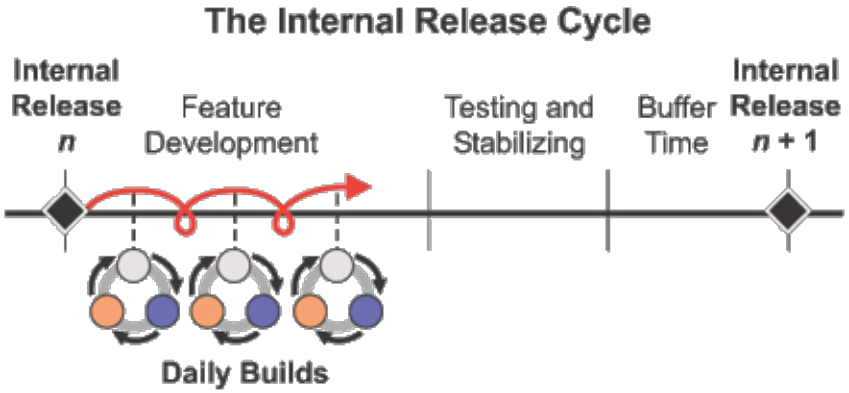


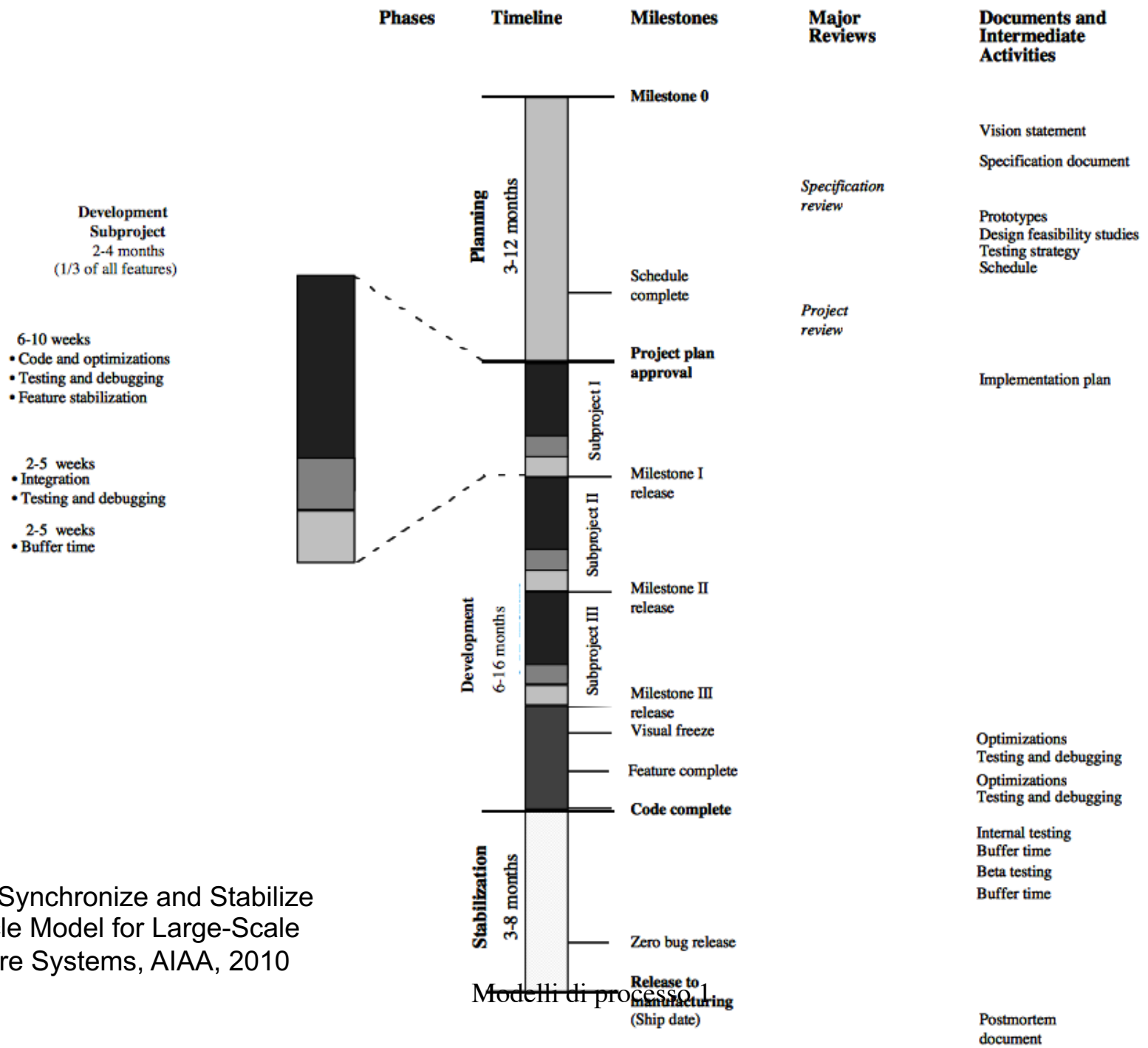
MSF: Microsoft Solutions Framework

https://msdn.microsoft.com/it-it/library/jj161047(v=vs.120).aspx

Il processo interno in Microsoft: Synch-and-Stabilize

- Sviluppi paralleli, sincronizzati di continuo
- Stabilizzare e incrementare periodicamente il prodotto (non: integrare tutto alla fine)
- Processo noto anche sotto i nomi:
  - milestone process
  - daily build process
  - nightly build process
  - zero-defect process



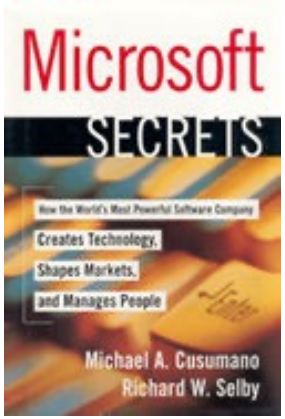


Selby, Synchronize and Stabilize Lifecycle Model for Large-Scale Software Systems, AIAA, 2010

Il processo interno in Microsoft

- Pianificazione
  - Documento programmatico
  - Specifica
  - Team management
- Sviluppo
  - 3-4 Sottoprogetti
- Stabilizzazione
  - Collaudo interno
  - Collaudo esterno
  - Golden master

Michael Cusumano



Pianificazione

- Documento di “visione” a cura del manager di prodotto
- Definisce gli obiettivi del nuovo prodotto o delle nuove funzioni (“*features*”)
- Definisce le priorità delle funzioni da implementare in base ai bisogni degli utenti
- Pianificare lo sviluppo con “buffer time”
- Documenti tipici:
  - **Specifica** di ciascuna “feature”
  - Pianificazione e definizione dei **team di progetto**
    - 1 program manager
    - 3-8 developers
    - 3-8 testers (1:1 ratio with developers)

Sviluppo

- Lo sviluppo delle ***features*** è suddiviso in 3-4 **sottoprogetti** (da 2-4 mesi ciascuno) usando specifiche funzionali opportunamente partizionate e messe in priorità
- Sottoprogetto: progetto, codifica, debugging
  - Si inizia con le funzioni prioritarie e col codice condiviso
  - L'insieme di "feature" può cambiare del 30% o più durante lo sviluppo

Sviluppo di un sottoprogetto

- Ciascun **sottoprogetto** esegue **in autonomia** il ciclo completo di sviluppo, integrazione di feature, testing e debugging
- I **testers** vengono accoppiati agli **sviluppatori**
- Le **squadre** si **sincronizzano** costruendo il prodotto e correggendo gli errori su **base quotidiana** e **settimanale**
- Il **prodotto** si **stabilizza** alla fine del sottoprogetto

Stabilizzazione

- Testing interno del prodotto completo
- Testing esterno: varie tipologie
  - Siti beta
  - ISVs (Independent Sw Vendors)
  - OEMs (Original Equipment Manifacturers)
  - Utenti finali
- Preparazione della release

Approccio alla qualità in Microsoft

Il principio base seguito da MS per perseguire la qualità dei propri prodotti è: "***Eseguire ogni attività in parallelo, con frequenti sincronizzazioni*** "

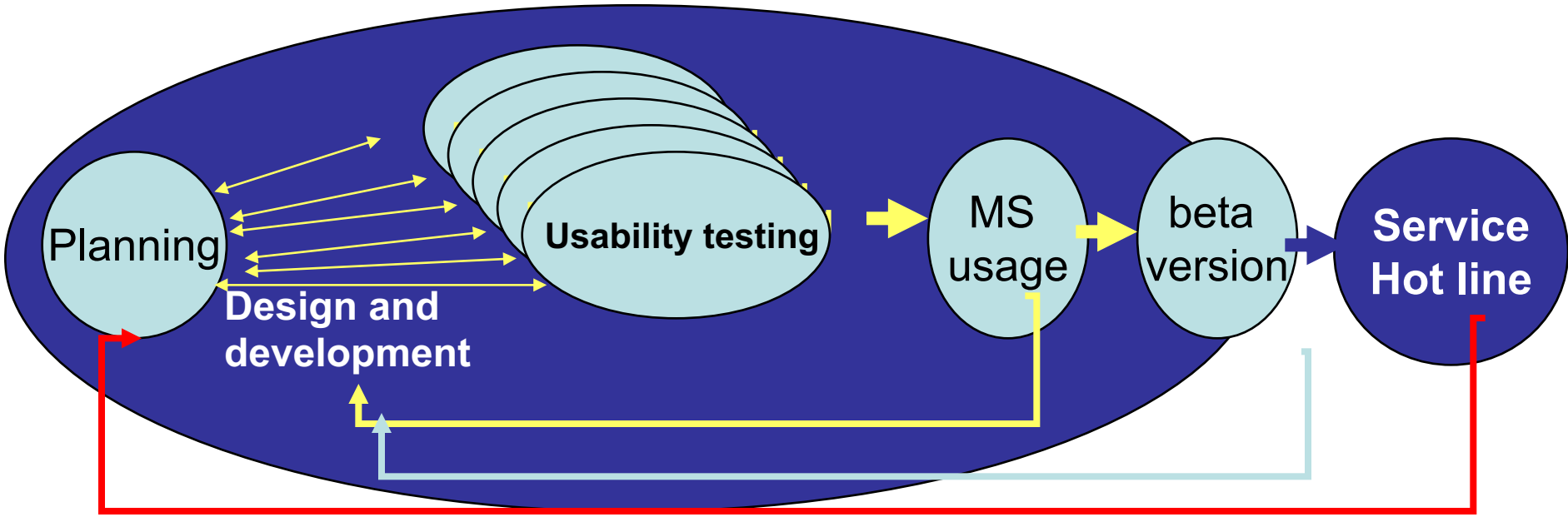
- Dividere i grandi progetti in più cicli di traguardi interni, con *buffer* e *nessuna manutenzione* separata del prodotto
- Utilizzare come guida un *documento programmatico* e un *documento di specifica* delle funzionalità del prodotto
- Fondare la scelta delle funzionalità e la scala delle priorità *sui dati e le attività dell'utente*
- Sviluppare un'architettura modulare e orizzontale, *in cui la struttura del prodotto sia rispecchiata nella struttura del progetto*
- Controllare un progetto impegnando i singoli in compiti di breve portata e "bloccando" le risorse del progetto

Come Microsoft coinvolge gli utenti

Gli utenti sono coinvolti durante lo sviluppo

- ‘pianificazione basata su attività’
- test di usabilità
- uso interno della nuova applicazione da parte di personale Microsoft
- linee di supporto alla clientela

Come Microsoft coinvolge gli utenti

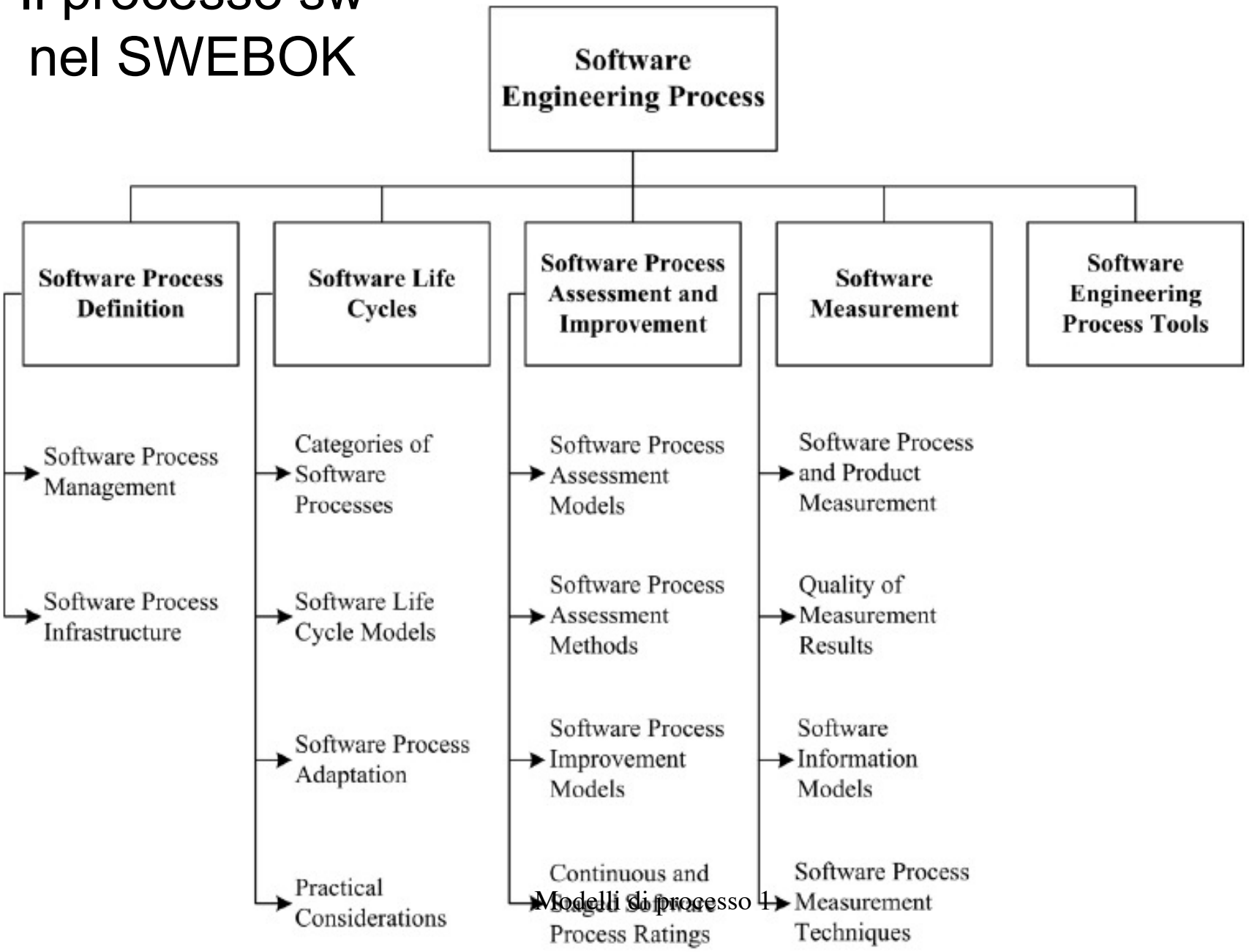


I principi chiave di Synch&Stab

(per definire i prodotti e modellare i processi)

1. Dividere i grandi progetti in più iterazioni con buffer time appropriato (20-50%)
2. Usare un "vision statement" e classificare le specifiche delle funzioni per guidare il progetto
3. Selezione delle caratteristiche principali e ordinamento di priorità basato su dati d'uso
4. Architettura modulare
5. Gestione basata su piccoli compiti individuali e risorse di progetto prestabilite

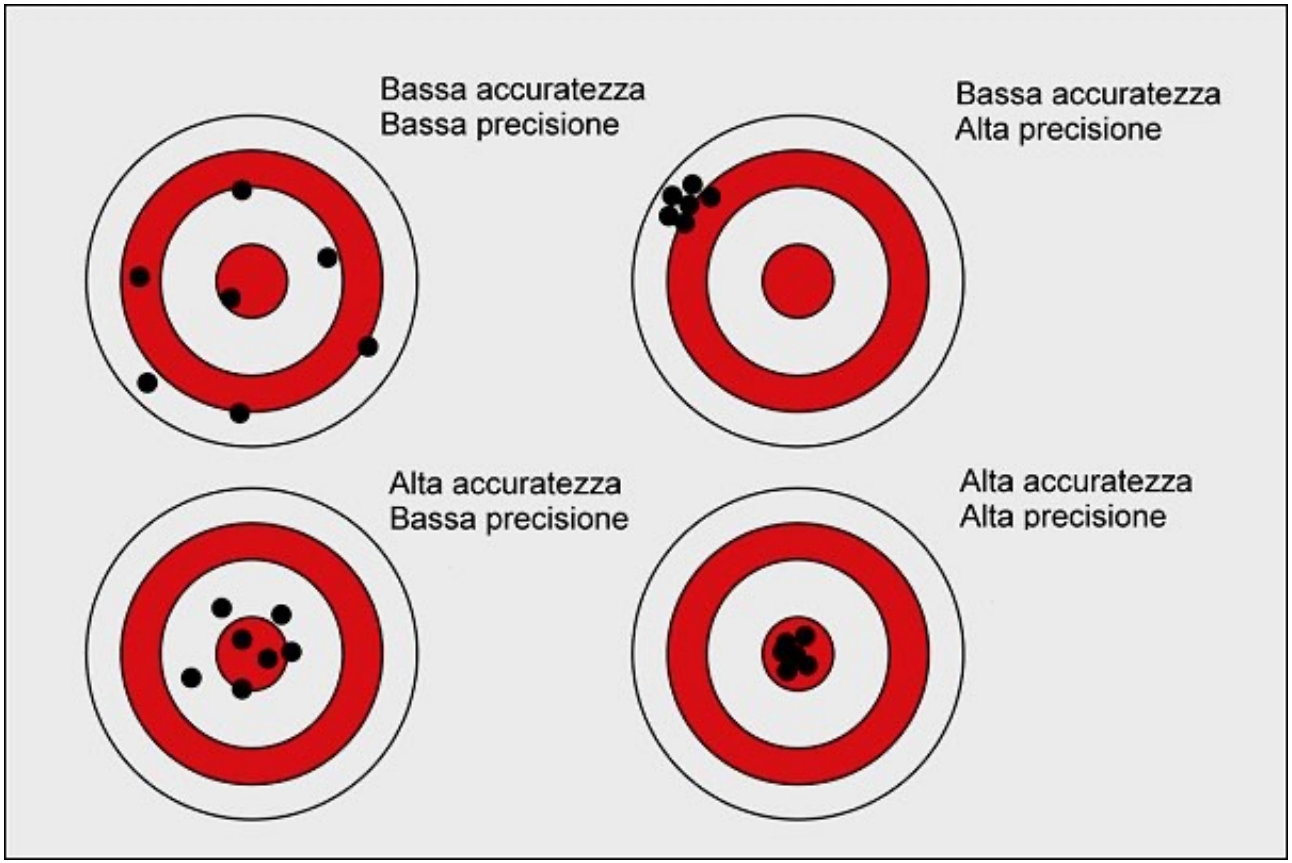
Il processo sw nel SWEBOK



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

Ripetibiilità: precisione e accuratezza



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?
- Quali sono i rischi principali che incontra chi sviluppa software?
- 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

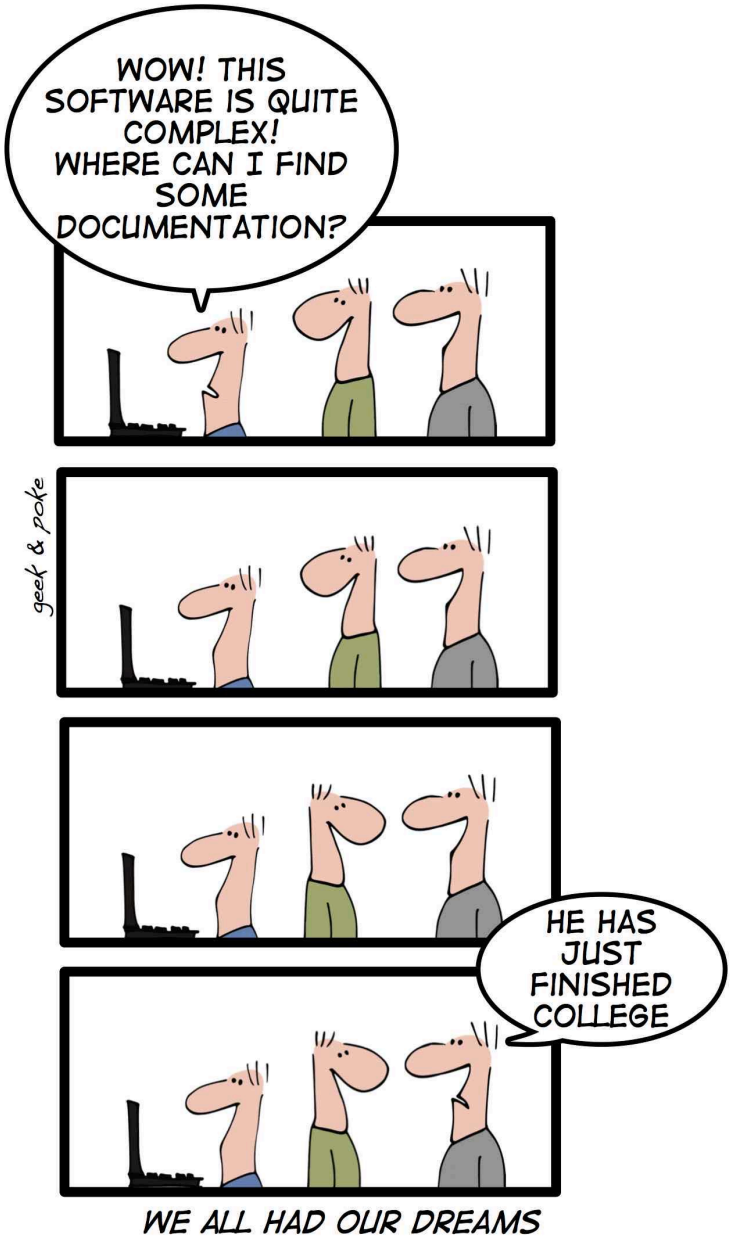
Siti

- ESSENCE `semat.org`
- RUP
  www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf
- Microsoft Solutions Framework
  `www.microsoft.com/technet/itsolutions/msf/default.mspx`
- Personal Software Process
  `resources.sei.cmu.edu/library/asset-view.cfm?assetid=5283`
- Adaptable Process Model `www.rspa.com/apm/`

Pubblicazioni di ricerca

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

Domande?


WOW! THIS SOFTWARE IS QUITE COMPLEX! WHERE CAN I FIND SOME DOCUMENTATION?
geek & poke
HE HAS JUST FINISHED COLLEGE
WE ALL HAD OUR DREAMS