2. Modelli del ciclo di vita del software

Paola Barra
a.a. 2022/2023

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

Modellare significa suddividere in attività che si occupino di :

- cosa
- quali prodotti
- quando

I processi software

Un processo software è un insieme di attività che porta alla creazione di un prodotto software.

Tutti i processi software hanno in comune quattro importanti attività:

1. **Specifica del software**: vengono definite le funzionalità del software fissati i vincoli operativi
2. **Sviluppo del software**: deve essere prodotto il software che realizza i requisiti definiti nella specifica
3. **Convalida del software**: il software deve essere convalidato per garantire ciò che il cliente richiede
4. **Evoluzione del software**: il software deve evolversi per soddisfare le necessità e i cambiamenti dei requisiti dell'utente

Oltre i processi

- **prodotti e consegne** sono il risultato di un'attività di processo

- bisogna definire i **ruoli** che riflettono le responsabilità delle persone all'interno del progetto

- **precondizioni** e **postcondizioni** da rispettare per portare a termine il progetto con successo (ad esempio approvazione dei requisiti)

Modello di ciclo di vita:

Organizzazione delle attività:

- ordinamento delle attività
- criteri per terminare e passare alla successiva

Esempio: modello di sviluppo generico per la preparazione di un dolce

- fare la spesa, mentre il forno si scalda: impastare, mettere nella teglia e infornare

Sono i passi da seguire per la preparazione del dolce, non è la ricetta.

I modelli di ciclo di vita

Ogni modello che presenteremo è importante perchè affronta un nuovo e differente aspetto:

- gli aspetti affrontati rappresentano le caratteristiche peculiari che ci interessano e guidano lo sviluppatore
- un ciclo di vita di un progetto reale può mescolare idee e modelli.

Problemi connessi allo sviluppo del software

I requisiti cambiano costantemente

- Il cliente può non essere consapevole di tutti i requisiti in anticipo

Le continue modifiche sono difficili da gestire

- Identificare checkpoint per la pianificazione e la stima dei costi è difficile

C’è più di un sistema software

- Il nuovo sistema spesso deve essere compatibile con sistemi già esistenti (legacy system)

Ciclo di vita del software

Ciclo di vita del software (anche processo software)

- Insieme di attività e loro relazioni reciproche per supportare lo sviluppo di un sistema software

Modello di ciclo di vita (anche modello del processo software)

- Un’astrazione che rappresenta un ciclo di vita del software con lo scopo di capire, monitorare, o controllare lo sviluppo di un sistema software

Ciclo di vita del sw: lo standard IEEE 610.12-1990

Processo di sviluppo del software:

- il processo per cui le necessità dell'utente sono tradotte in un prodotto software

Il processo prevede la traduzione

- delle necessità **dell'utente** nelle
- **richieste del software**, trasformare le richieste del software nel
- **progetto**, implementare il progetto in forma di
- **codice**, **testing del codice** e talvolta,
- **installare** e **controllare** il software per un uso operativo
- Nota: queste attività possono sovrapporsi o essere eseguite iterativamente

Ciclo di vita del software

- In pratica …

Ciclo di vita del software: introduzioe

Non esiste un ciclo di vita del software ideale

- Esistono molteplici processi software differenti
  - Le grandi organizzazioni hanno sviluppato propri approcci in base alle proprie necessità
- I processi si sono evoluti anche in base alla specifiche caratteristiche dei sistemi da sviluppare
  - Ad esempio, per i sistemi critici (sistemi per cui un fallimento del sw può portare a gravi perdite economiche, danni fisici o minacce per la vita) un processo di sviluppo strutturato è più adeguato
  - Per i sistemi aziendali in cui i requisiti cambiano velocemente, potrebbe essere più efficace un processo agile e flessibile

Identificazione delle attività dello sviluppo software

Domande nello sviluppo del software

- Qual è il problema?
- Come possiamo dividere il problema?
- Qual è la soluzione?
- Qual è il miglior meccanismo per implementare la soluzione?
- Come è costruita la soluzione?
- Il problema è risolto?
- Il cliente è in grado di usare la soluzione?
- Sono necessarie estensioni?

Attività di sviluppo software - Esempio 1

Analisi requisiti	Qual è il problema?
Progettazione sistema	Qual è la soluzione?
Progettazione dettagliata	Quali sono i migliori meccanismi per implementare la soluzione?
Implementazione	Come è costruita la soluzione?
Testing	Problema risolto?
Consegna	Il cliente in grado di usare la soluzione?
Manutenzione	Sono necessarie estensioni?

Dominio dell' Applicazione

Dominio della Soluzione

Attività di sviluppo software - Esempio 2

Analisi requisiti — **Qual è il problema?**

Progettazione sistema — **Qual è la soluzione?**

Progettazione oggetti — Quali sono i migliori meccanismi Per implementare la soluzione?

Implementazione — **Come è costruita la soluzione?**

Dominio dell' Applicazione

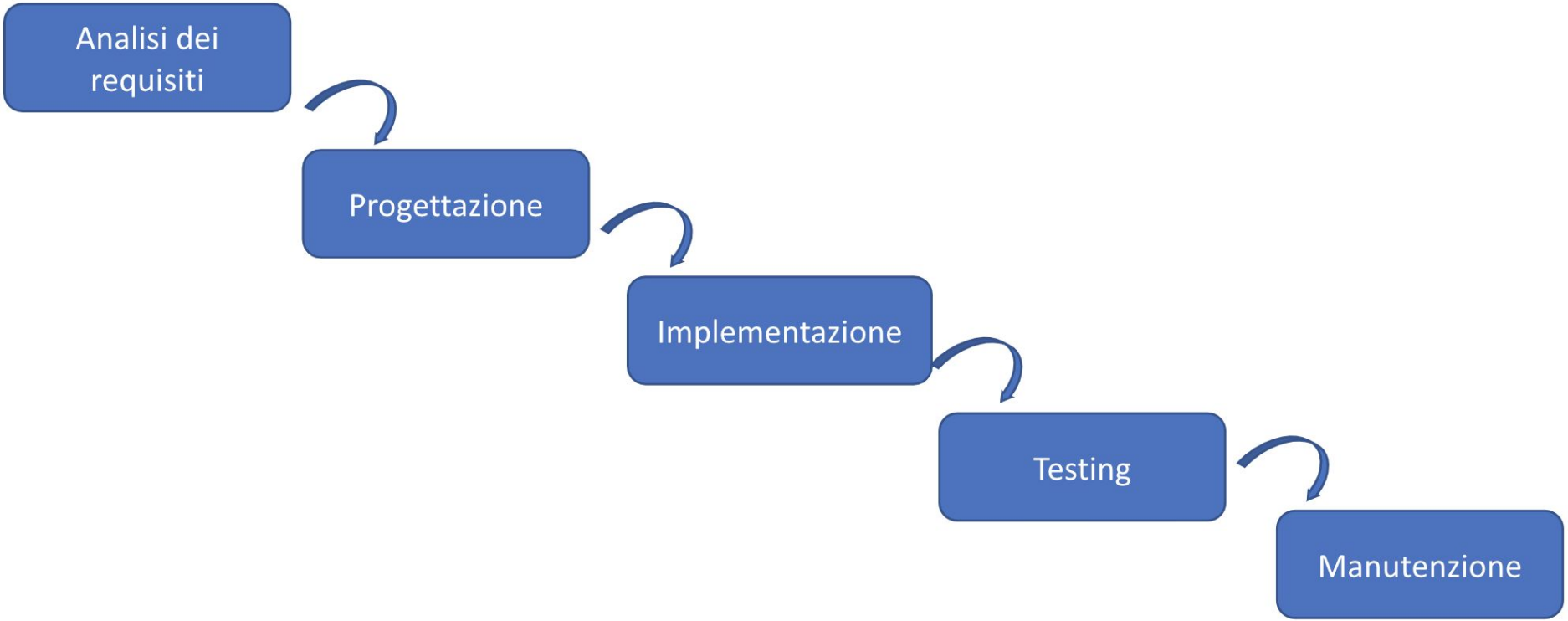
Dominio della Soluzione

Attività del processo software

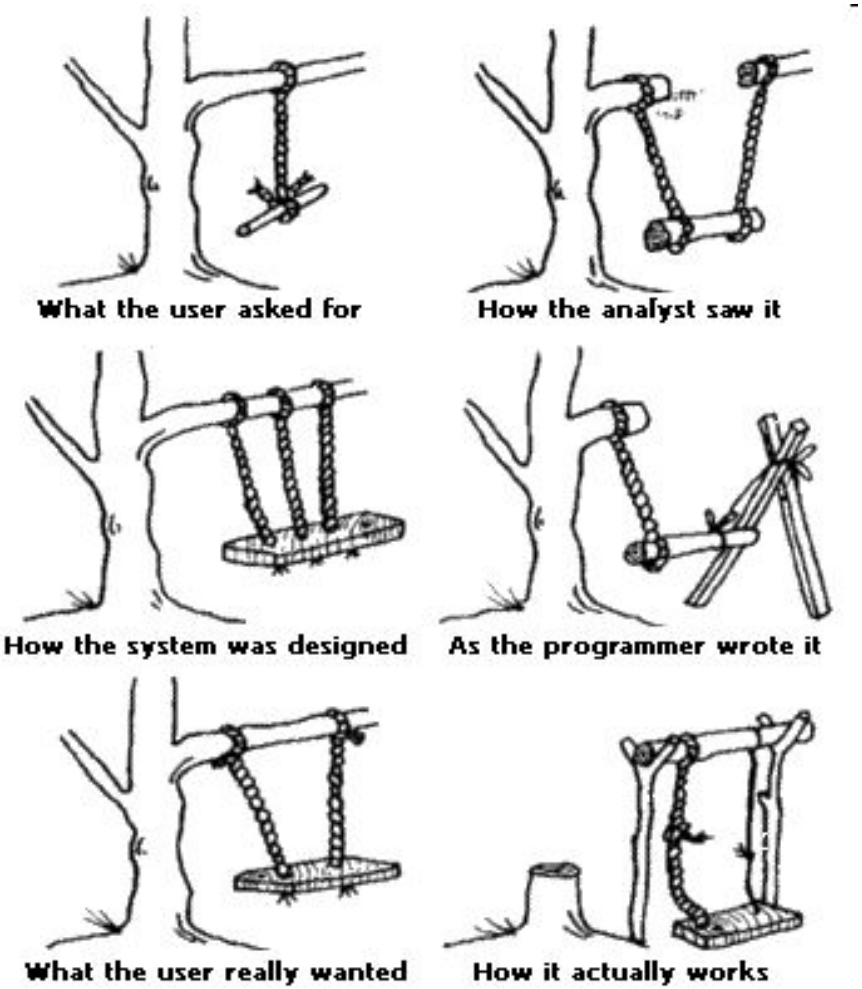
Le attività del processo software fondamentali, comuni a tutti i processi sono:

- **Specifiche del software**: si definiscono le funzionalità del **sw** e sono fissati i vincoli operativi
- **Progettazione e implementazione**: si sviluppa il **sw** in moda da realizzare i requisiti individuati
- **Convalida**: si verifica che il **sw** garantisca le funzionalità richieste dal cliente
- **Evoluzione**: il **sw** deve poter essere modificato in modo da soddisfare i cambiamenti dei requisiti dell'utente

Ciclo di vita del software: una possibile soluzione



I modelli di processo



Modelli di ciclo di vita del software o di processo di sviluppo del software

Un modello di processo è un'astrazione del processo software

- Rappresenta il processo software da una particolare prospettiva

"**Un modello del ciclo di vita del software è una caratterizzazione descrittiva o prescrittiva di come un sistema software è o dovrebbe essere sviluppato**"

W. Scacchi - Encyclopedia of Software Engineering Vol II pag 860

I modelli di processo software sono descrizioni precise e formalizzate delle attività, degli oggetti, delle trasformazioni e degli eventi per realizzare e/o ottenere l'evoluzione del software

N.B.: molti autori usano i termini *processo di sviluppo del software e ciclo di vita del software come* sinonimi

Modelli di processo: caratteristiche (1)

Una strutturazione dell'organizzazione del lavoro nelle fabbriche del software

- … fasi della produzione,
- … tipi di attività,
- … collegamento ed interfacciamento,
- … controllo e misura,

ma anche linee guida per: organizzare, pianificare, dimensionare personale, assegnare budget, schedulare e gestire, …

Modelli di processo: caratteristiche (2)

- definire e prescrivere prodotti e documenti da rilasciare al committente (**deliverables)**
- determinare e classificare metodi e strumenti più adatti a supportare le attività previste
- framework per analizzare, stimare, migliorare ...

Processi guidati da piani VS processi agili

- I processi guidati da un piano sono processi in cui **tutte** le attività del processo sono pianificate in anticipo ed il progresso è misurato rispetto al piano
- Nei processi agili, la pianificazione è incrementale ed è più facile cambiare il processo per riflettere la modifica dei requisiti del cliente
- In pratica, molti processi concreti includono elementi di ambo gli approcci
  - Non ci sono processi software corretti o sbagliati

I modelli

Build and fix model

Modelli prescrittivi :

- modello a cascata
- rapid prototyping
- modello incrementale

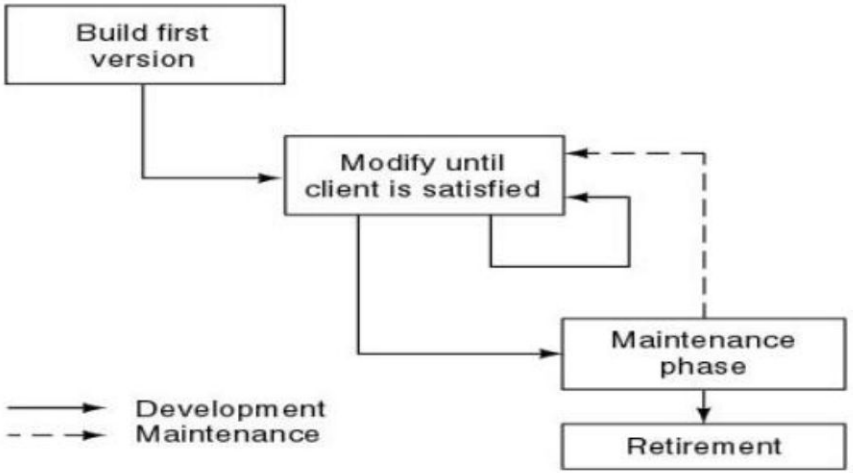
Modelli agili :

- Extreme programming
- SCRUM

Build and fix model

Build and fix model

- Le attività non sono identificate
- il prodotto è sviluppato senza specifica e senza un tentativo di progettazione
- lo sviluppatore scrive un programma
- che poi è modificato più volte finchè non soddisfa il committente

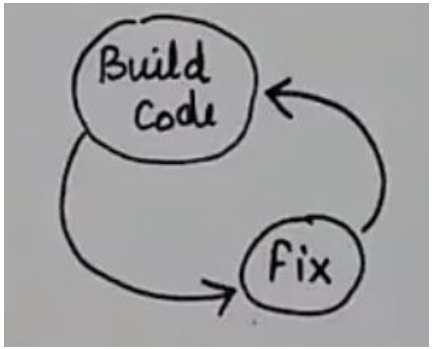


Il più semplice (e peggiore) modello di processo

- Molto veloce, feedback rapido
- Molti strumenti disponibili
- Specializzato per codifica
- Non incoraggia la documentazione
- Ingestibile durante manutenzione

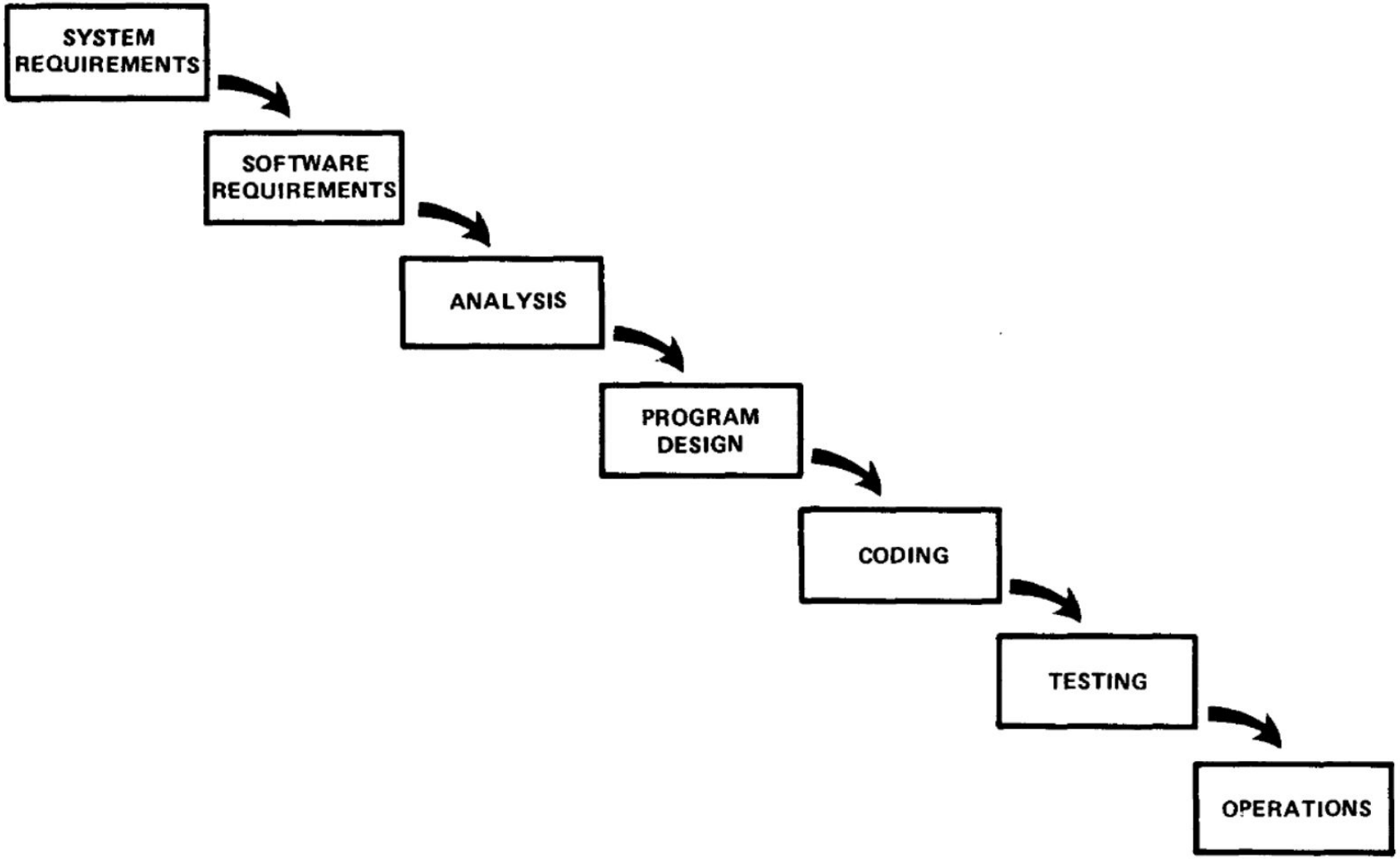
Build and fix model

- forse adeguato a un progetto di 100 linee di codice
- Diventa improponibile per prodotti di ragionevole grandezza
- la manutenzione di un prodotto senza specifica o documentazione che ne spieghi la progettazione è estremamente difficile

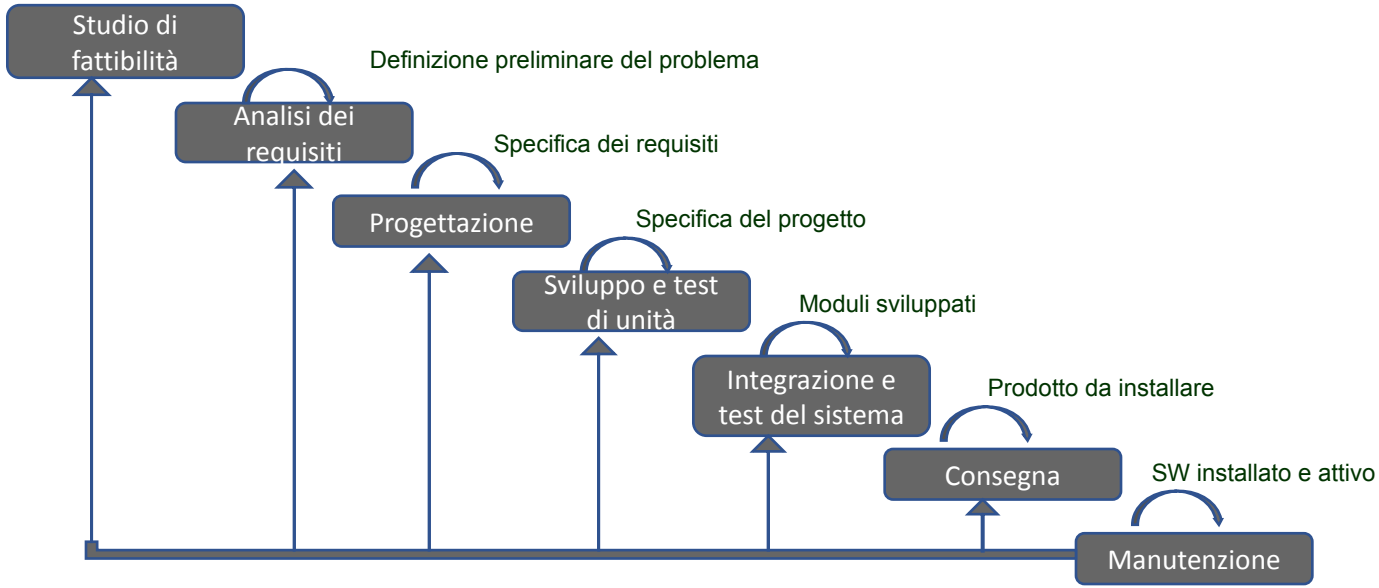


Prima di iniziare lo sviluppo di un progetto dobbiamo scegliere un “vero” modello del ciclo di vita

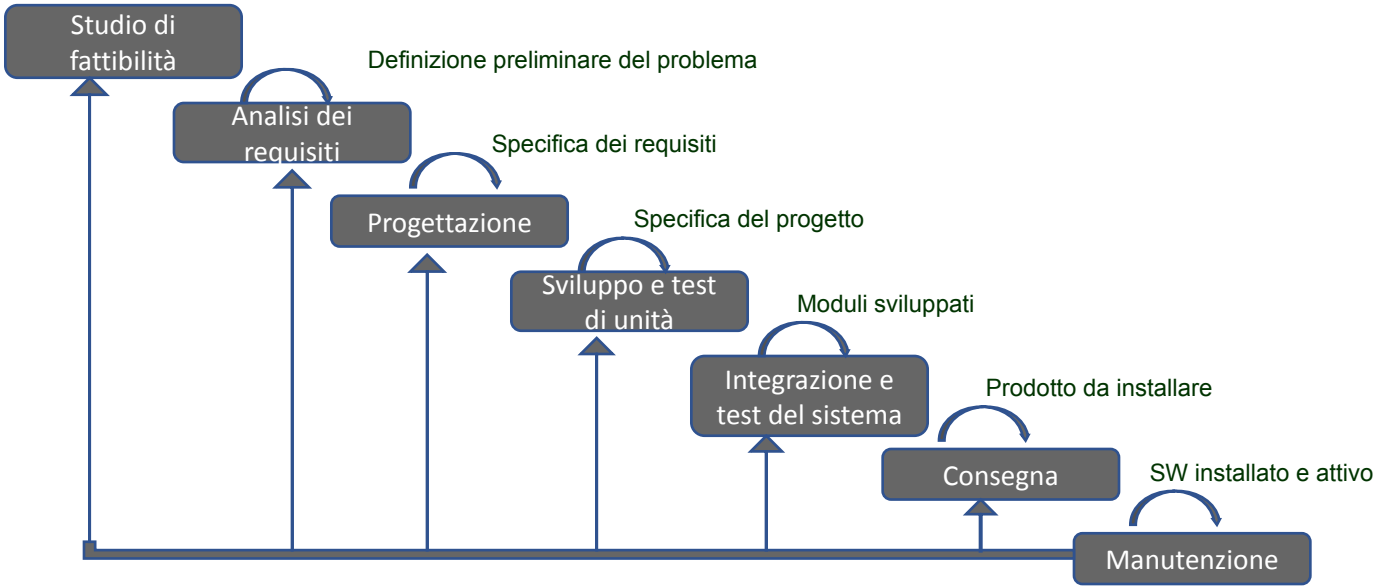
Il modello a cascata [Royce, 1970]


SYSTEM REQUIREMENTS
SOFTWARE REQUIREMENTS
ANALYSIS
PROGRAM DESIGN
CODING
TESTING
OPERATIONS

Le fasi di un modello a cascata

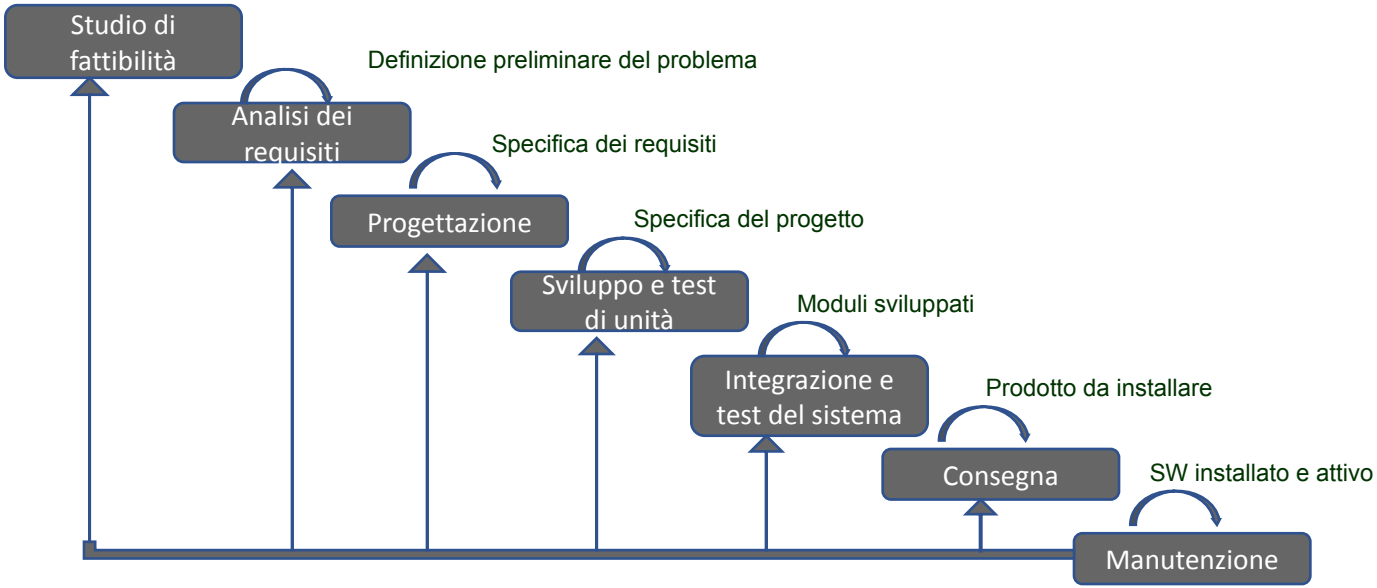


Analisi e definizione dei requisiti



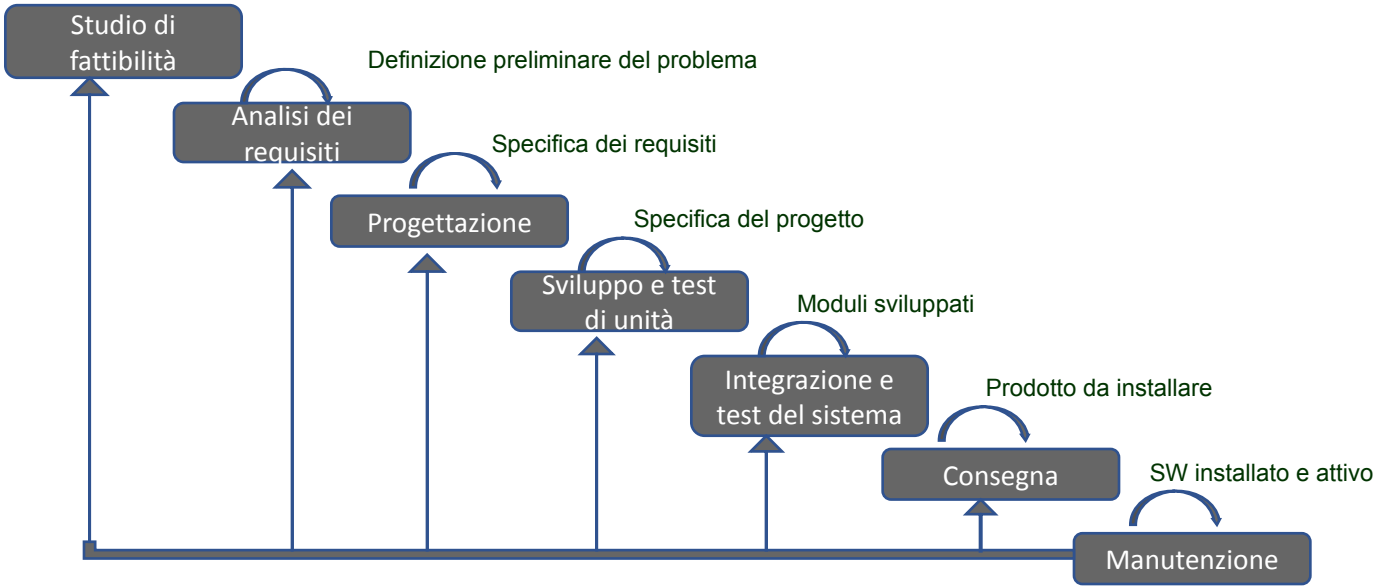
Si determinano i servizi del sistema, i suoi vincoli e obiettivi attraverso incontri con gli utenti del sistema. I requisiti vengono definiti in dettaglio e servono come specifica del sistema.

Progettazione del sistema e del software



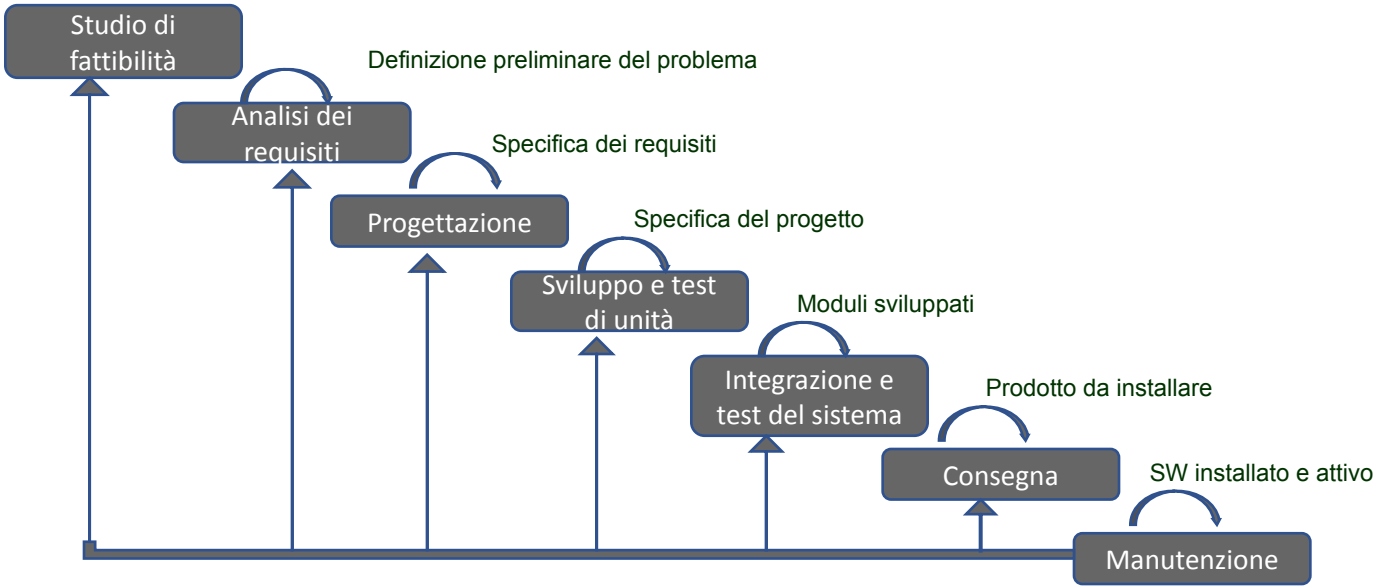
Il processo di progettazione del sistema suddivide i requisiti sul sistema hardware o sul software e stabilisce l'architettura generale del sistema. La progettazione coinvolge l'identificazione e la descrizione delle astrazioni fondamentali del sistema e le reciproche relazioni.

Sviluppo e test di unità



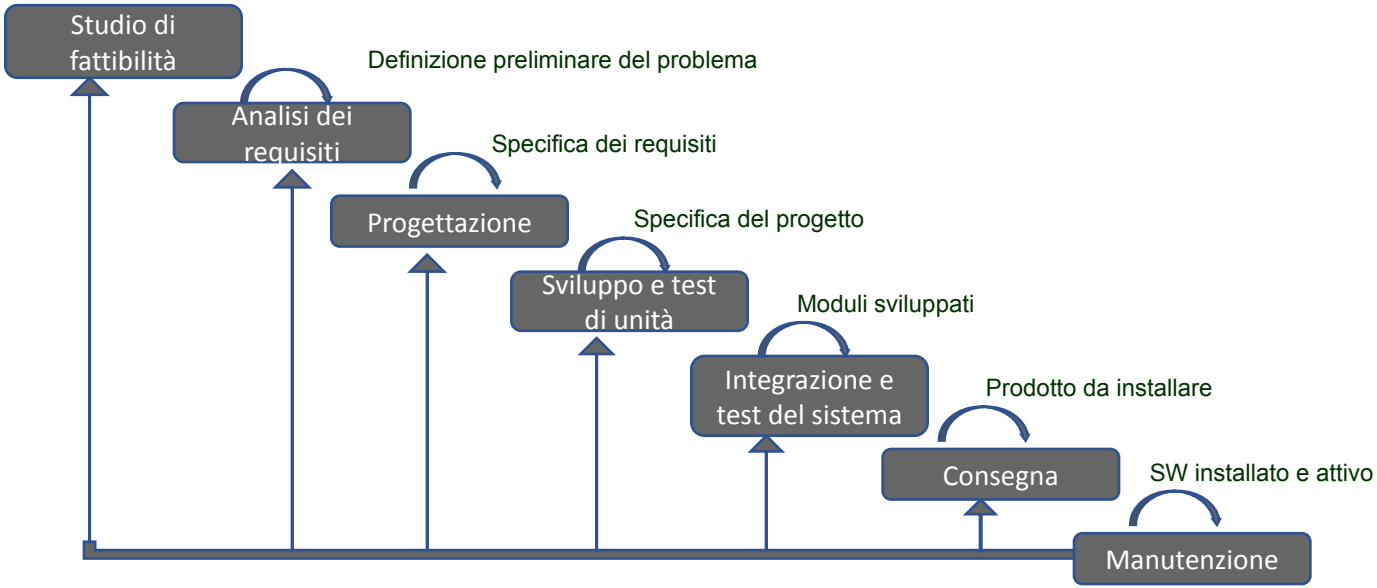
In questa fase il progetto del software è realizzato come un insieme di programmi o di unità del programma. Il test delle unità verifica che ognuna soddisfi le proprie specifiche.

Integrazione e test del sistema



Le singole unità di programma o i programmi sono integrati e testati come un sistema completo per accertare che i requisiti del software siano soddisfatti. Dopo il test finale, il sistema è consegnato al cliente.

Consegna e manutenzione



normalmente (anche se non è obbligatoria), questa è la fase più lunga dell'intero ciclo di vita.
Il sistema viene installato e messo in opera, si correggono gli errori non scoperti nei primi stadi del ciclo di vita, si migliora l'implementazione delle unità di sistema quando vengono evidenziati nuovi requisiti.

Modello a cascata

- Popolare negli anni ’70
  - reazione al “code and fix” originario
  - ispirazione dall’industria manifatturiera
- Modello sequenziale lineare
  - progressione sequenziale (in cascata) di fasi, senza cicli, al fine di meglio controllare tempi e costi
  - definisce e separa le varie fasi e attività del processo
    - nullo (o minimo) overlap fra le fasi
  - uscite intermedie: semilavorati del processo (documentazione di tipo cartaceo, programmi)
    - formalizzati in struttura e contenuti
  - consente un controllo dell’evoluzione del processo
    - attività trasversali alle diverse fasi

Il modello a cascata

Questo modello permette di distinguere e definire le fasi di un processo software evidenziando l'importanza delle fasi di analisi e progettazione prima di passare alla codifica

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 documento "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 alla fine del processo se ci sono discrepanze tutto il processo deve essere ripetuto

Un ulteriore punto critico è l'eccessiva produzione di documenti:

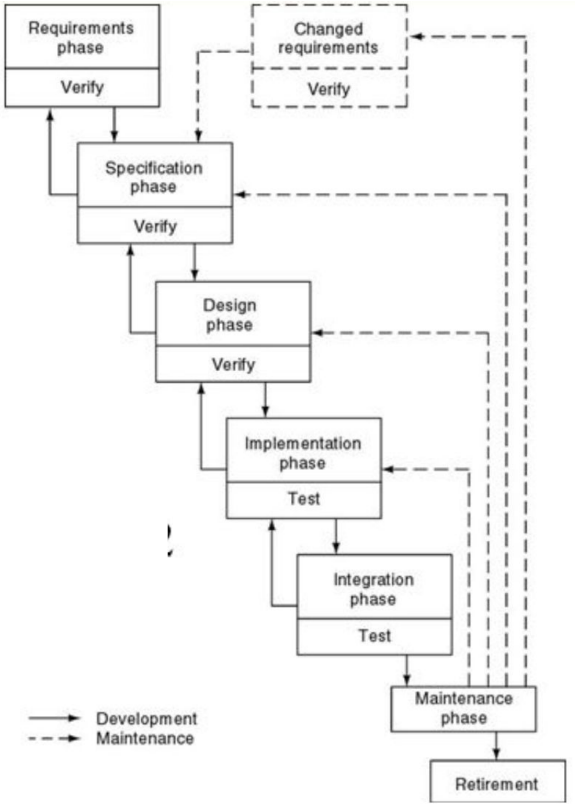
- nessuna fase è completa finchè anche il documento per quella fase non è stato terminato e approvato dal gruppo di valutatori (Software Quality Assurance)

Il modello a cascata ha avuto storicamente il valore di aver definito e strutturato le fasi di un processo software.

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 altri modelli in alternativa.



Proprietà dei modelli basati su modelli a cascata

- I responsabili amano il modello a cascata
  - Milestone diretti
  - Nessuna necessità di cicli all'indietro (Sistema lineare)
  - Sempre un'attività alla volta
  - Facile controllare il progresso durante lo sviluppo
- Tuttavia, lo sviluppo del software non è lineare
  - Mentre si sta sviluppando una progettazione, sono identificati problemi nei requisiti
  - Mentre si scrive codice, sono trovati problemi di progettazione e di requisiti
  - Mentre è testato un programma, emergono errori di codifica, di progettazione e nei requisiti

Modello a cascata: vantaggi e svantaggi

Vantaggi

- ha definito molti concetti utili
- ha rappresentato un punto di partenza importante per lo studio dei processi SW
- facilmente comprensibile e applicabile
- Ogni fase del processo produce la relativa documentazione

Svantaggi

- interazione con il committente solo all'inizio e alla fine
- requisiti congelati alla fine della fase di analisi
- requisiti utente spesso imprecisi: "l'utente sa quello che vuole solo quando lo vede"
- Errori nei requisiti scoperti solo alla fine del processo
- il nuovo sistema software diventa installabile solo quando è totalmente finito
- né l'utente né il management possono giudicare prima della fine la distanza del sistema dalle proprie aspettative

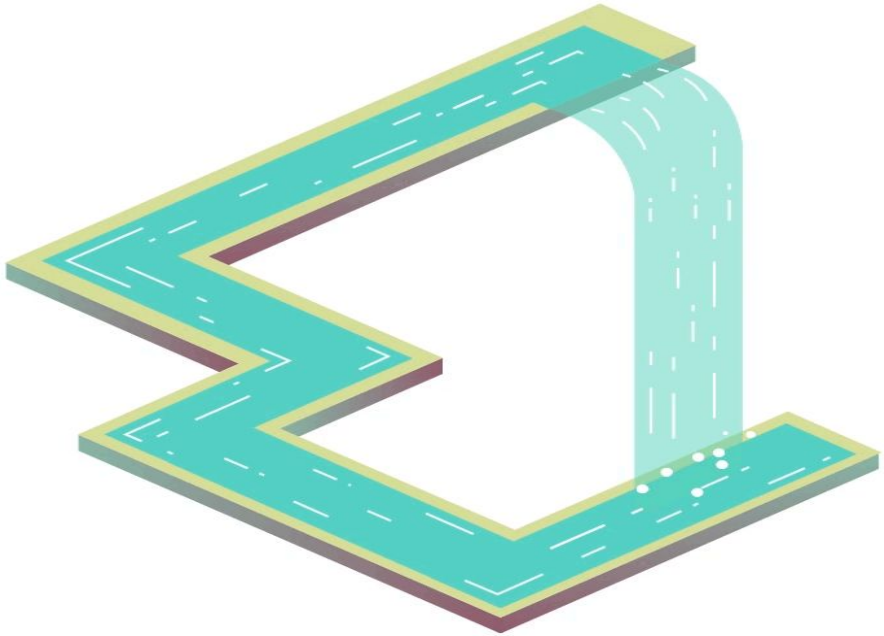
Modello a cascata, in pratica...

Il modello è adeguato solo quando i requisiti sono ben compresi e le modifiche sono piuttosto limitate durante la progettazione

E’ maggiormente usato in grossi progetti in cui un sistema è sviluppato in diversi siti

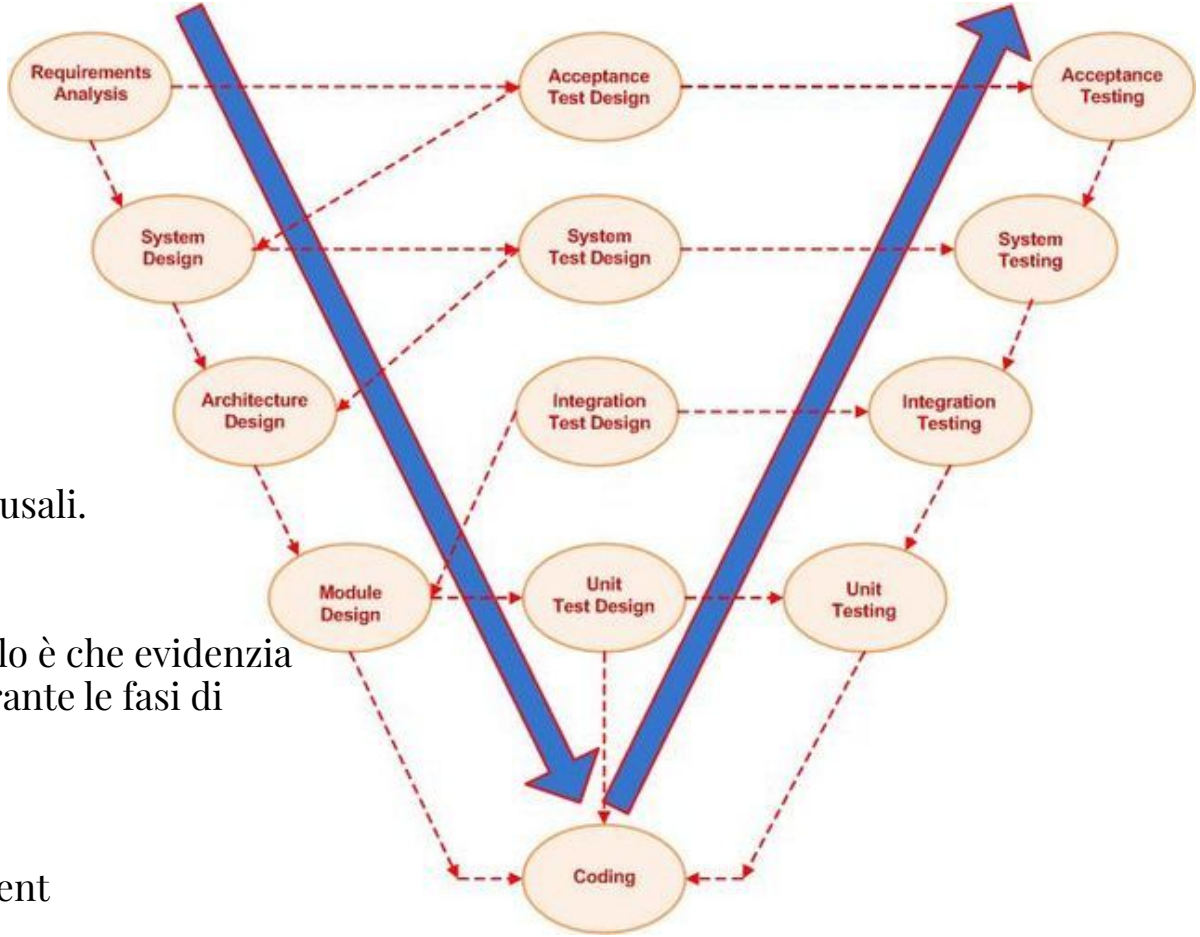
- In tali circostanze, la natura guidata da un piano aiuta a coordinare il lavoro

Iterazione del modello a cascata



Il modello a V
Hughes Aircraft - 1982 (sequenziale)

Il modello a V



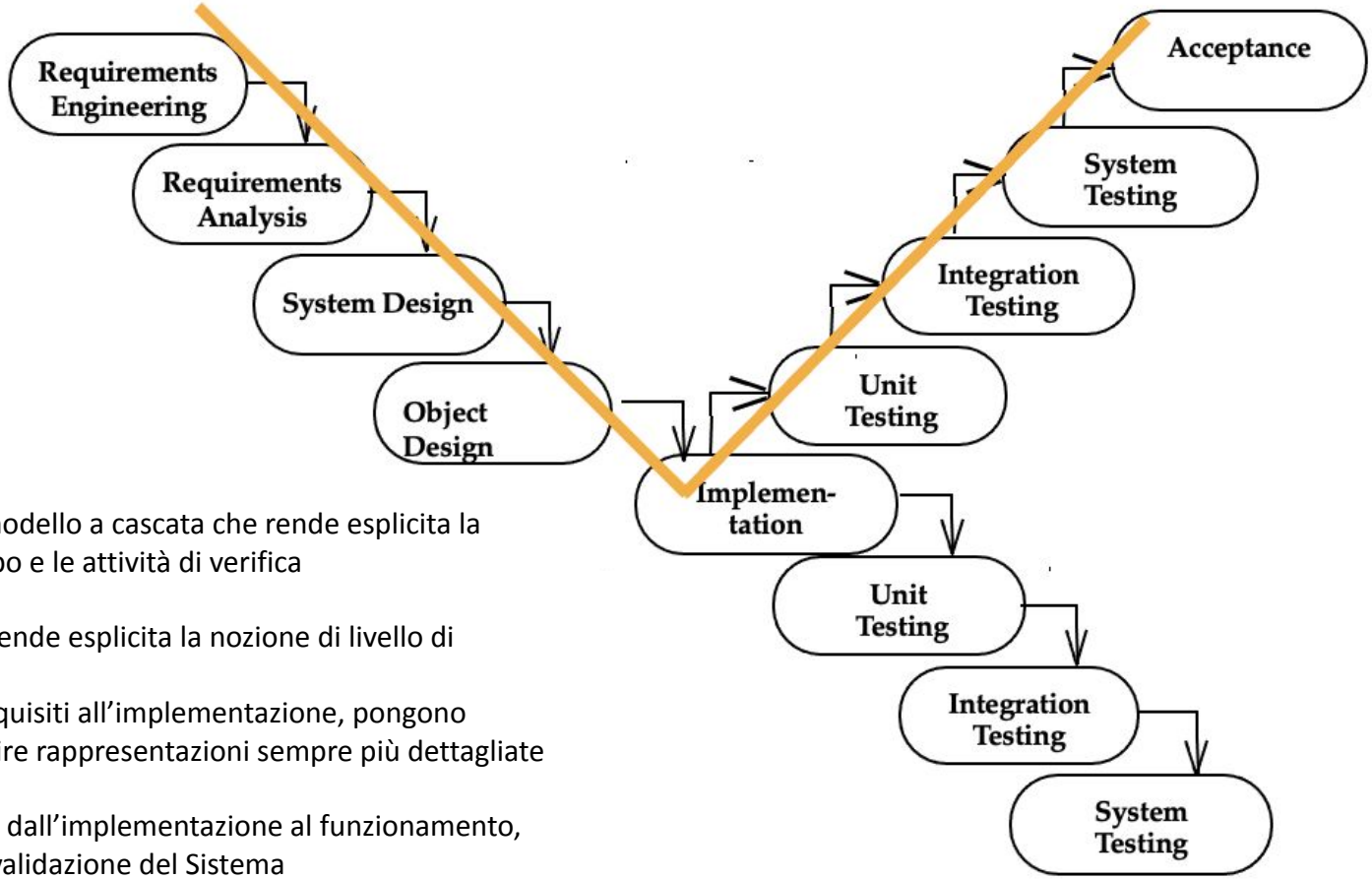
Le frecce blu rappresentano il tempo.

Le frecce tratteggiate le dipendenze causali.

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

Dal modello a cascata al modello a V



Il modello a V è una variante del modello a cascata che rende esplicita la dipendenza tra le attività di sviluppo e le attività di verifica

La differenza è che il modello a V rende esplicita la nozione di livello di astrazione

- Tutte le attività, dai requisiti all'implementazione, pongono l'attenzione nel costruire rappresentazioni sempre più dettagliate del sistema
- invece tutte le attività, dall'implementazione al funzionamento, sono focalizzate sulla validazione del Sistema

Rapid prototyping

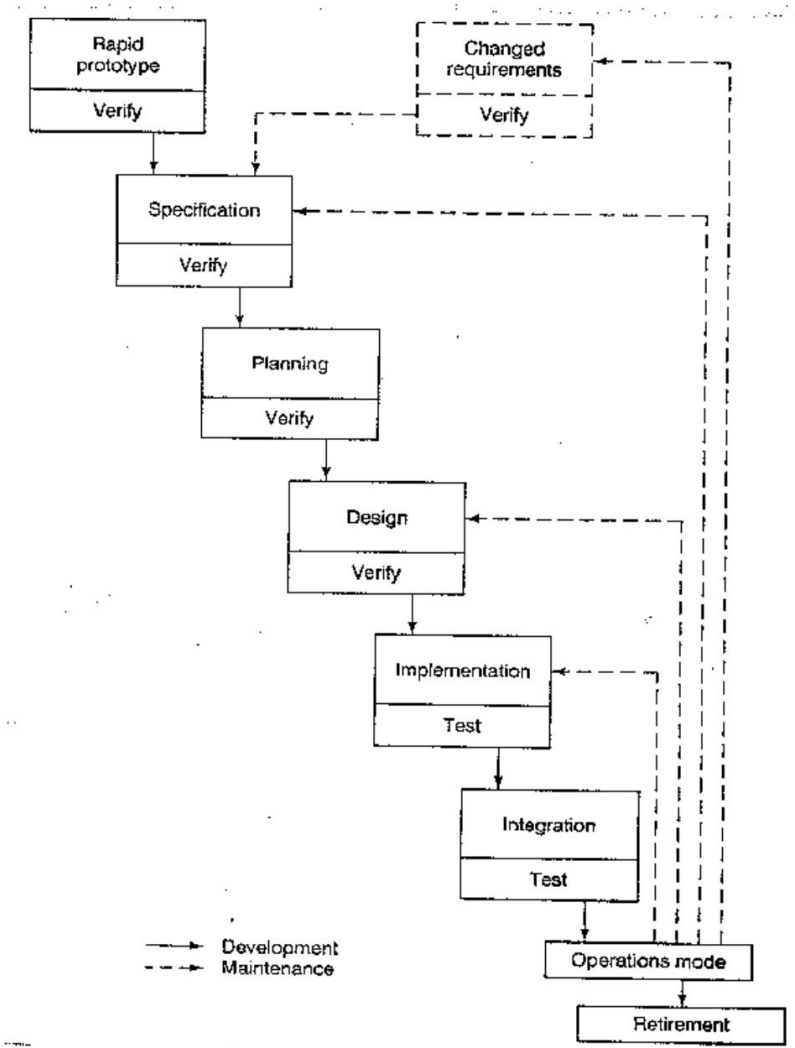
Rapid prototyping

Il primo passo è quello di costruire un prototipo velocemente per permettere al committente di sperimentalo.

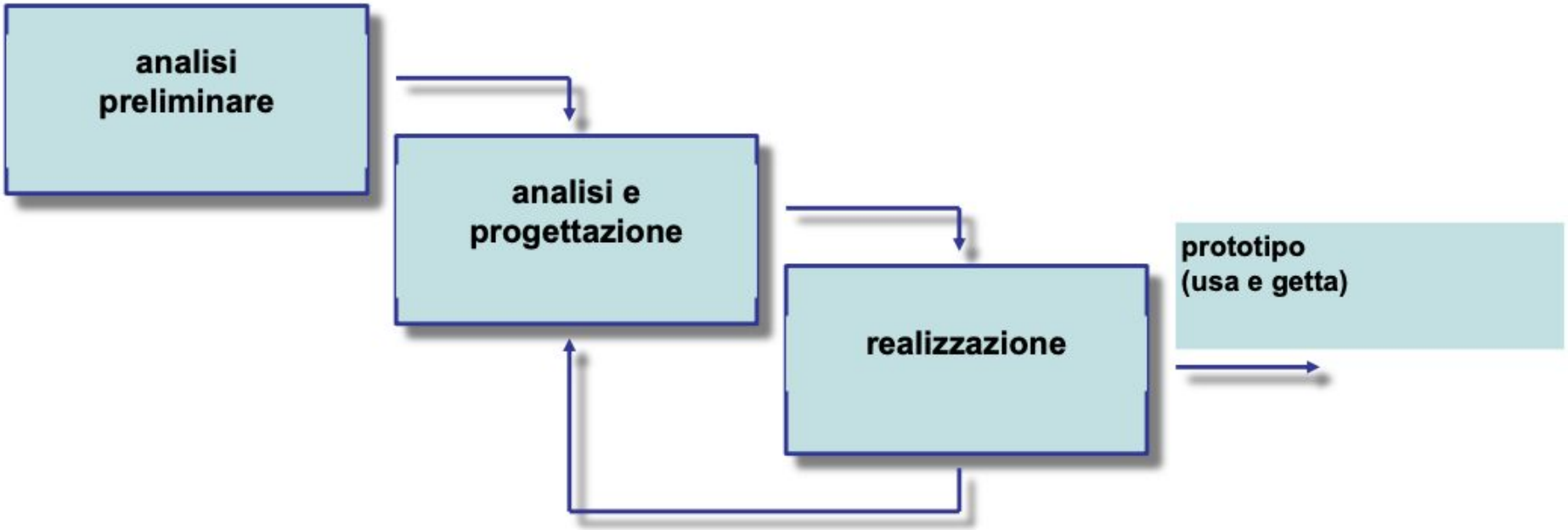
Utile quando i requisiti non sono chiari

Il prototipo aiuta il cliente a meglio descrivere i requisiti:
- si passa a questo punto alla fase di specifica

Anche noto come modello evolutivo



Rapid prototyping



Modello a spirale

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

E' un modello astratto:
- va specificato per dire cosa fare in concreto in ogni iterazione e in ogni sua fase

Il modello a spirale

Evidenzia gli aspetti gestionali:

- pianificazione delle fasi
- centrato sull'analisi dei rischi (modello risk driven)

Tipici rischi:

dominio poco noto, linguaggio o strumenti nuovi, personale non addestrato

applicabile ai cicli tradizionali

prevede maggior comunicazione e confronto con il committente

Modello a Spirale

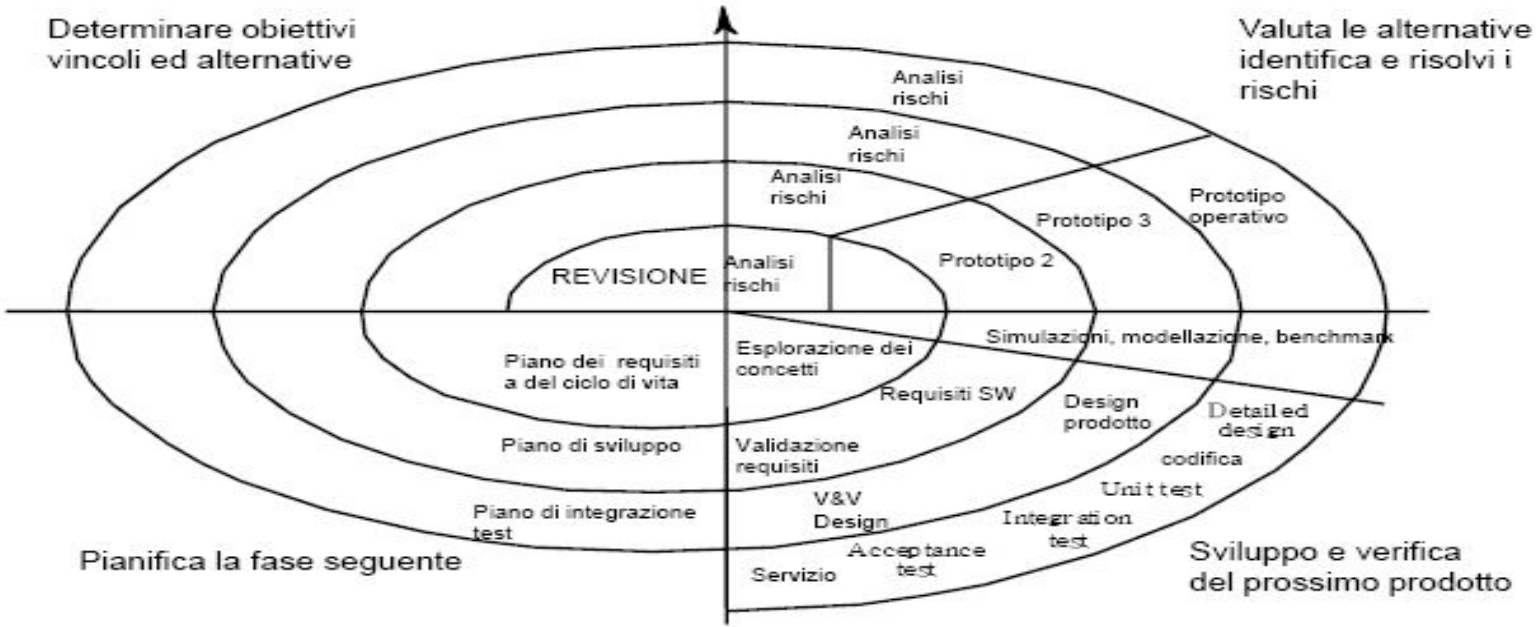
- Il modello a spirale proposto da Boehm prevede il seguente insieme di attività
  - Determinare obiettivi e vincoli
  - Valutare le alternative
  - Identificare i rischi
  - Risolvere i rischi assegnando delle priorità ai rischi
  - Sviluppare una serie di prototipi per i rischi identificati partendo dal rischio maggiore
  - Usare un modello a cascata per lo sviluppo di ciascun prototipo
  - Se un rischio è stato risolto con successo, valutare i risultati del ciclo e pianificare il prossimo ciclo
  - Se un certo rischio non può essere risolto, terminare il progetto immediatamente

Round nel modello a spirale

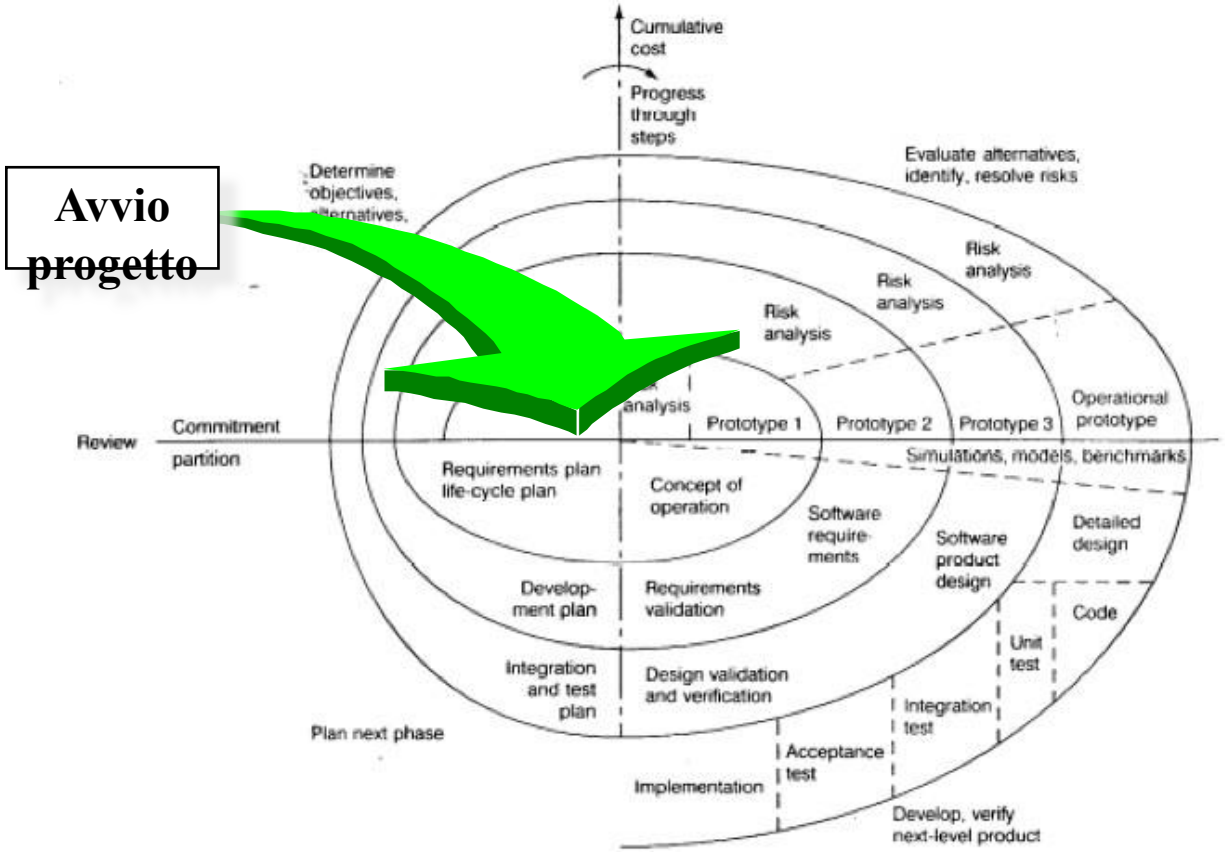
- Concetto operazioni
- Requisiti software
- Progettazione prodotto software
- Progettazione dettagliata
- Codice
- Test unità
- Integrazione e test
- Test accettazione
- Implementazione

- Per ogni round esegue le attività seguenti
  - Definisce obiettivi, alternative, vincoli
  - Valuta alternative, identifica e risolve rischi
  - Sviluppa e verifica un prototipo
  - Pianifica il prossimo giro

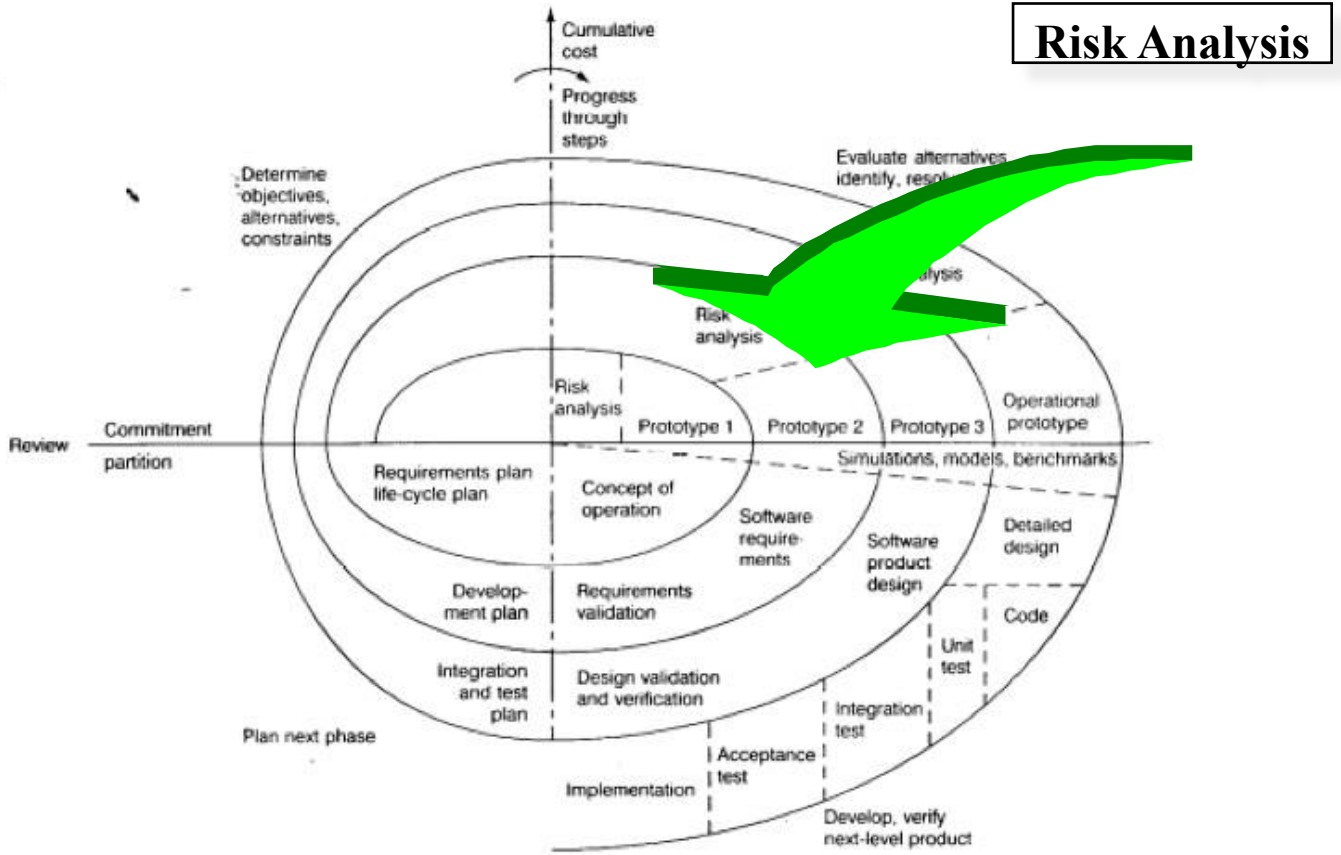
Diagramma del modello aspirale



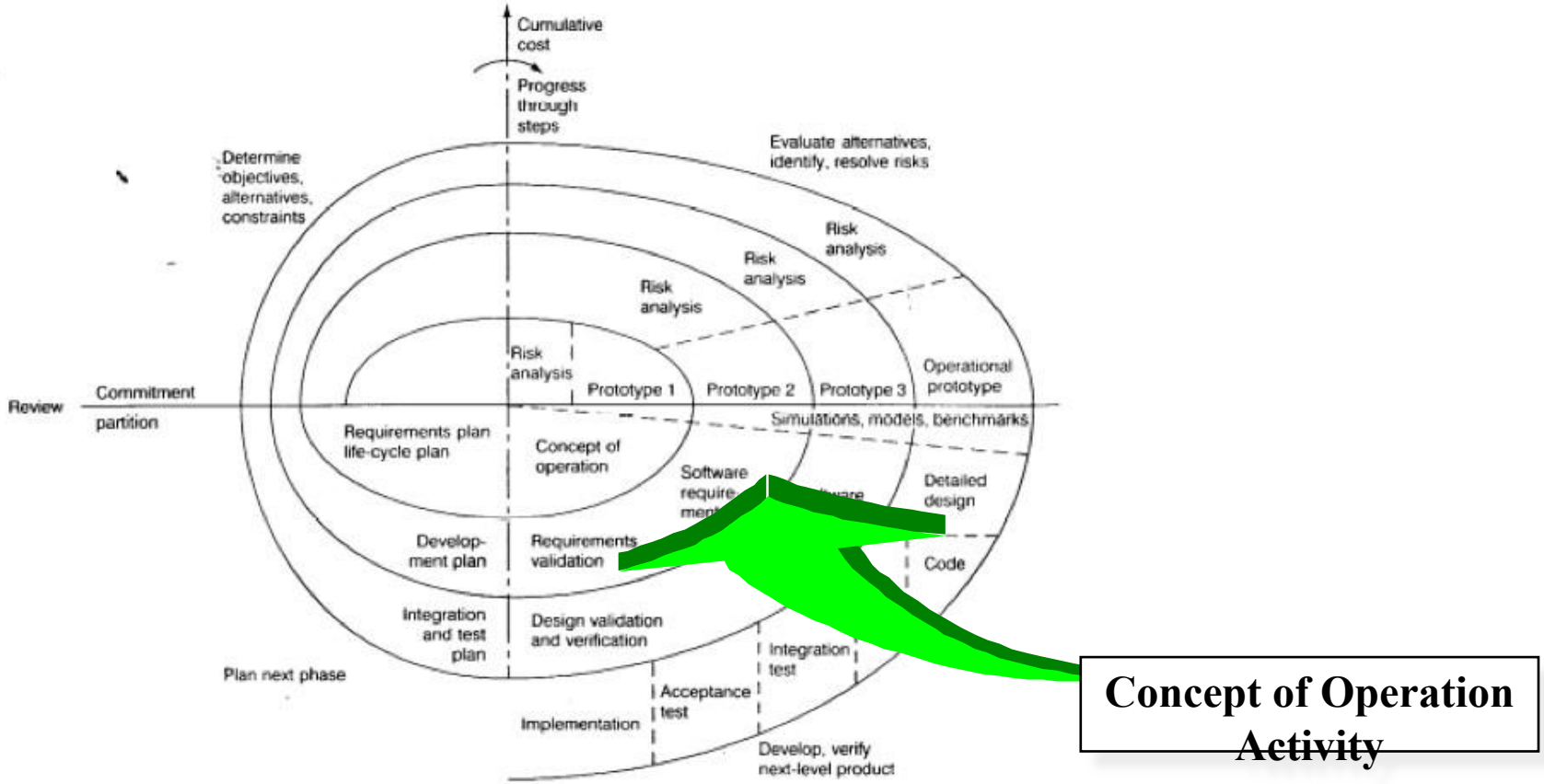
Round 1, Concept of Operations, Quadrant IV: Determine Objectives, Alternatives & Constraints



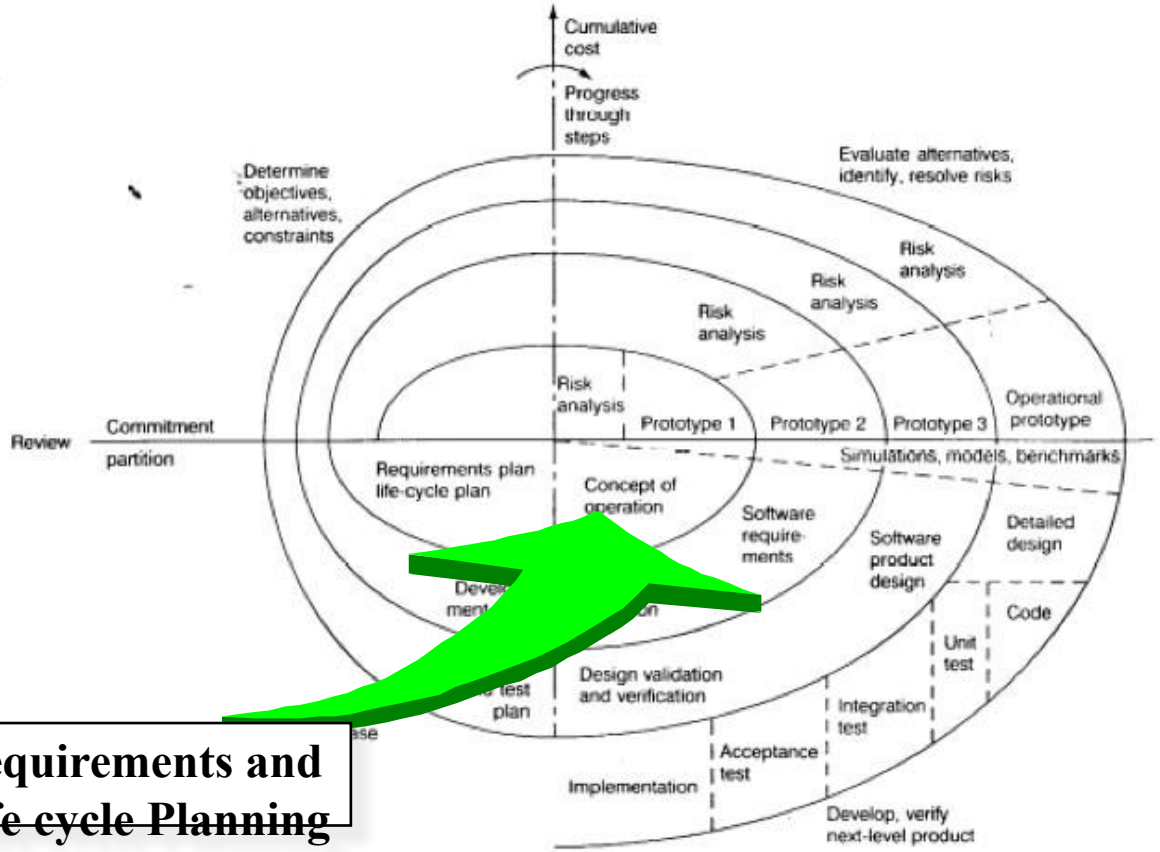
Round 1, Concept of Operations, Quadrant I: Evaluate Alternatives, identify & resolve Risks



Round 1, Concept of Operations, Quadrant II: Develop and Verify

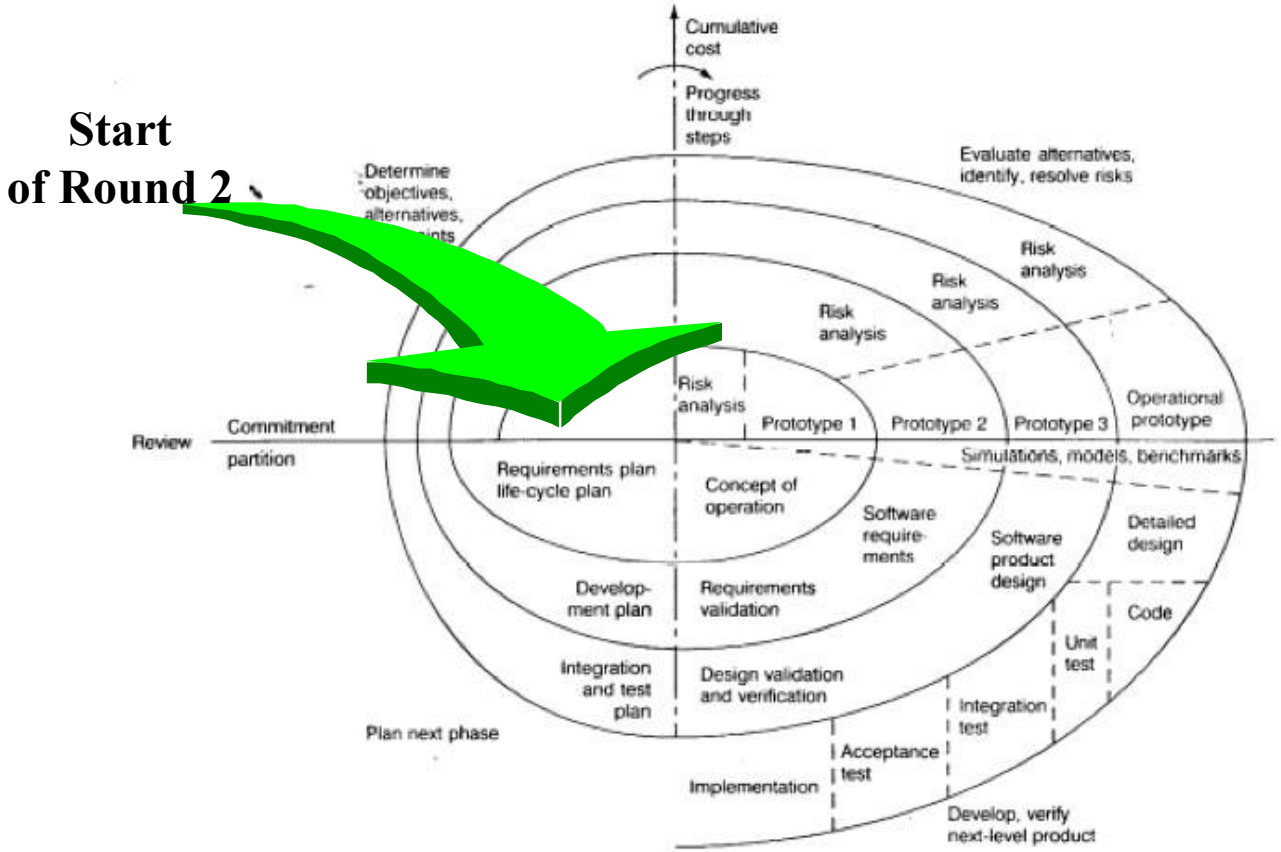


Round 1, Concept of Operations, Quadrant III: Prepare for Next Activity

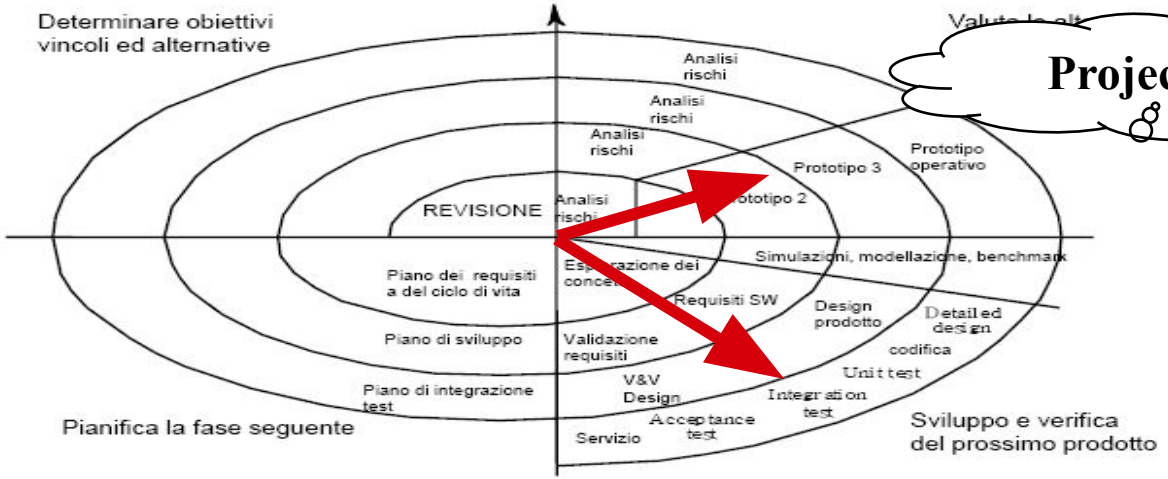


Requirements and Life cycle Planning

Round 2, Software Requirements, Quadrant IV: Determine Objectives,Alternatives & Constraints



Confronto tra progetti



Project P1

Project P2

Limiti modelli a cascata e a spirale

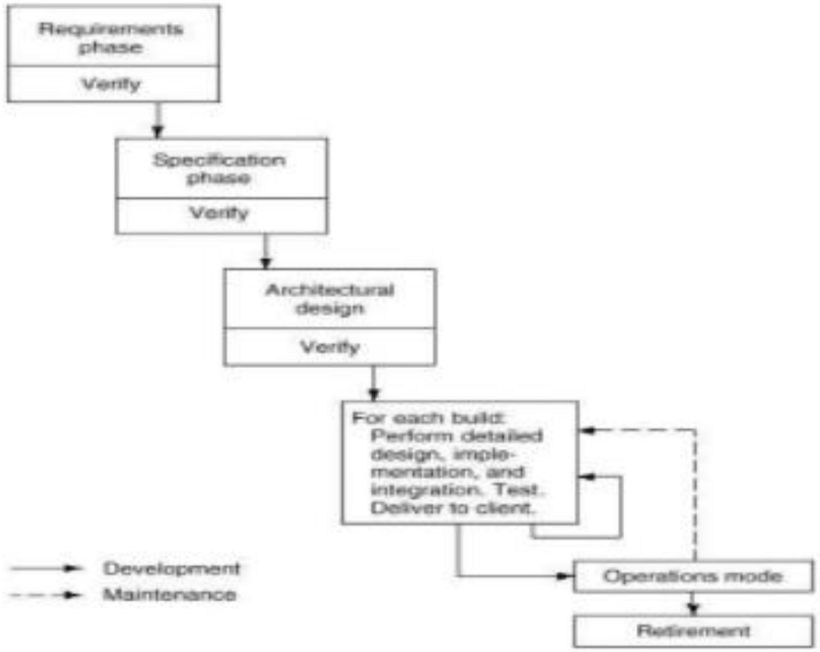
- Nessuno dei due gestisce i cambiamenti frequenti
  - Il modello a cascata assume che una volta terminata una fase, tutti i problemi relativi ad essa sono chiusi e non possono essere riaperti
  - Il modello a spirale può gestire i cambiamenti tra le fasi, ma non permette modifiche nella fase
- Che si fa se le modifiche si verificano più spesso?
  - “L’unica costante è il cambiamento”

Modello incrementale

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

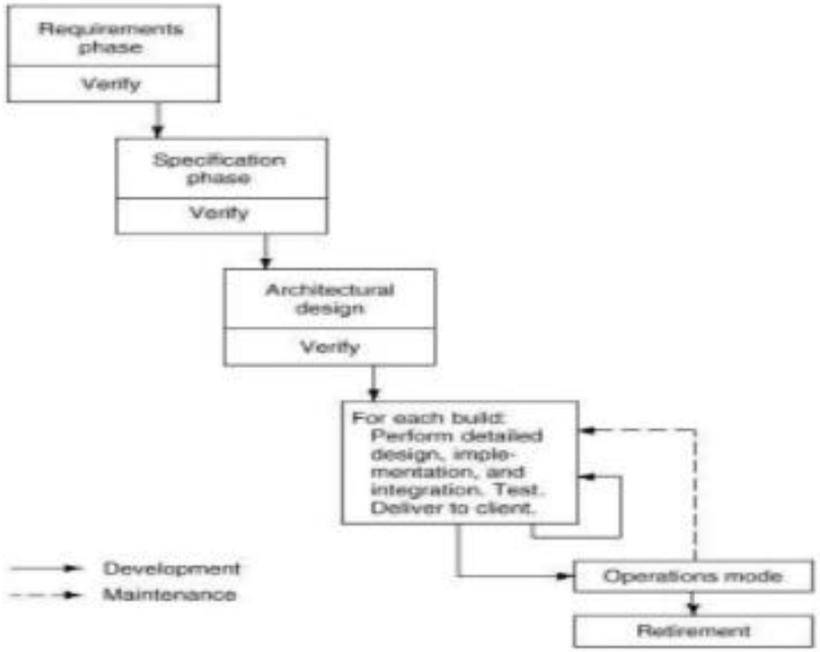


Modello incrementale

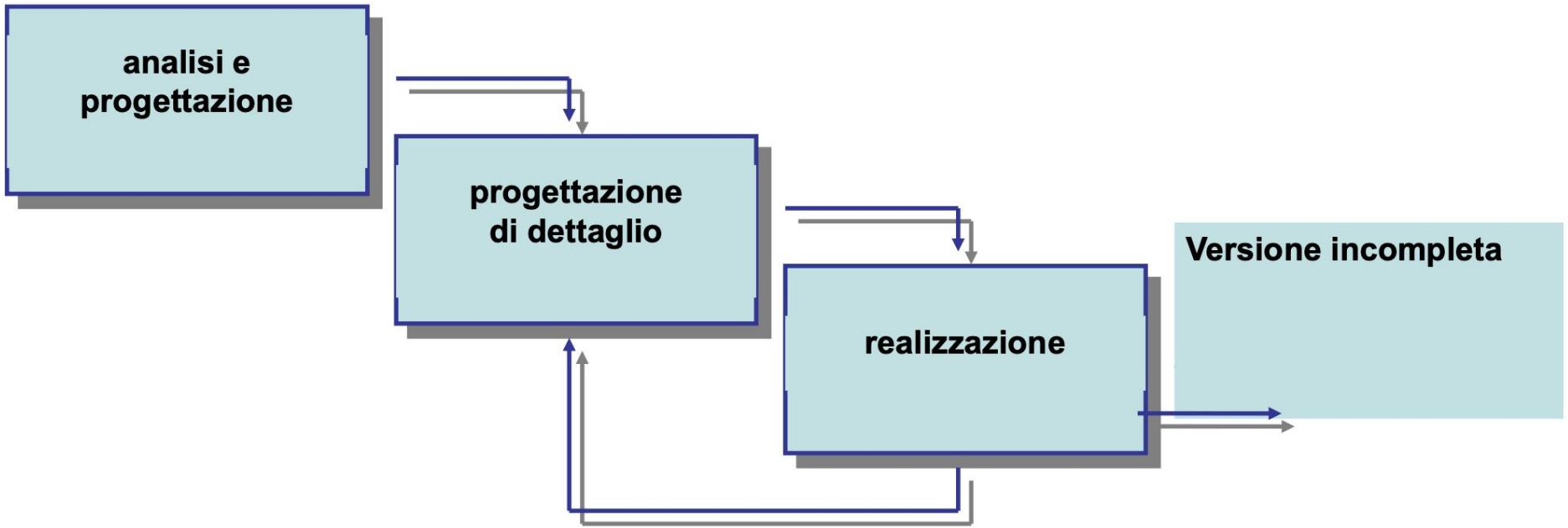
Si applica in caso di requisiti stabili, serve a :

- ritardare le realizzazioni delle componenti che dipendono criticamente da fattori esterni (tecnologie, hardware sperimentale, ecc)

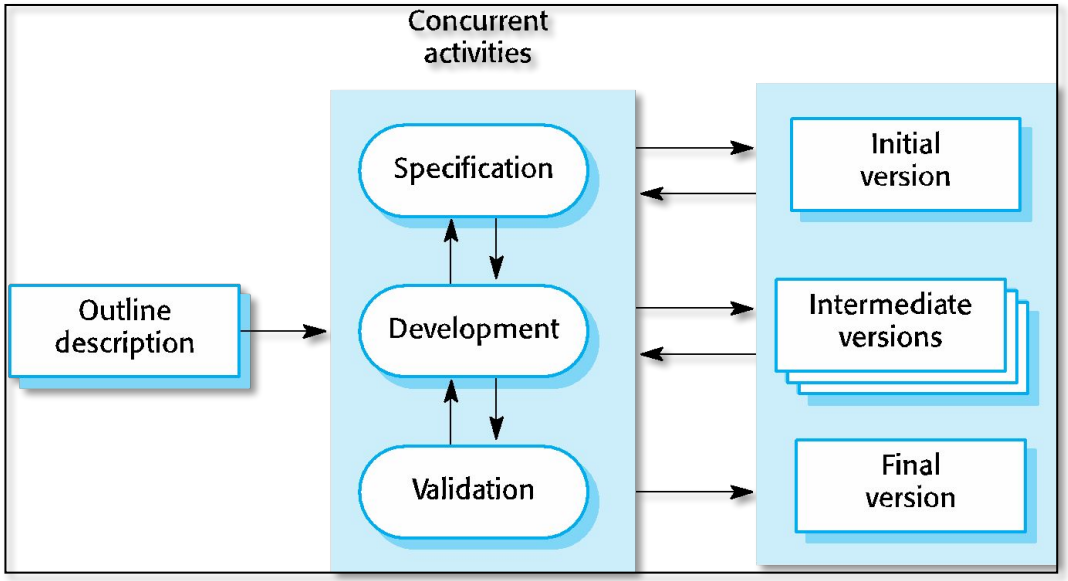
Se non è progettato bene diventa un Build and Fix.



Modello incrementale



Sviluppo incrementale



Si basa sull'idea di sviluppare un'implementazione iniziale, esporla agli utenti e perfezionarla attraverso molte versioni, finchè non si ottiene il sistema richiesto.

Può essere agile.

Gli incrementi del sistema sono stabiliti in anticipo.

Vengono definiti gli incrementi iniziali, ma lo sviluppo dei successivi incrementi dipende dall'avanzamento del lavoro e dalle priorità del cliente.

Benefici dello sviluppo incrementale

Il costo per soddisfare le richieste di modifiche dei requisiti del cliente è ridotto

- La quantità di analisi e di documentazione che deve essere rifatto è molto inferiore di quanto richiesto dal modello a cascata

E’ più facile considerare il feedback del cliente sul lavoro di sviluppo che è stato fatto

- I clienti possono commentare le dimostrazioni del software e vedere quanto è stato implementato

E’ possibile una consegna più rapida e la distribuzione di software utile al cliente

- I clienti sono in grado di usare e riceverne guadagno dal software molto prima di quanto possibile con un processo a cascata

Problemi dello sviluppo incrementale

Il processo non è visibile

- I responsabili hanno bisogno di deliverable consegnati con regolarità per misurare il progresso
- Se i sistemi sono sviluppati velocemente, non è efficiente con i costi di produzione di documenti associati ad ogni versione del sistema

La struttura del sistema tende a degradare quando sono aggiunti nuovi incrementi

- A meno che non sono investiti tempo e denaro di refactoring per migliorare il software, le modifiche regolari tendono a corromperne la struttura
- Incorporare ulteriori modifiche nel software diventa sempre più difficile e costoso

Sviluppo basato su componenti

Basato sul riuso del software dove i sistemi sono integrati da componenti o sistemi esistenti

Gli elementi riusati possono essere configurati per adattare il loro comportamento e funzionamento ai requisiti dell'utente

Il riuso è ora un approccio standard per costruire molti sistemi commerciali

Tipi di software riusabile

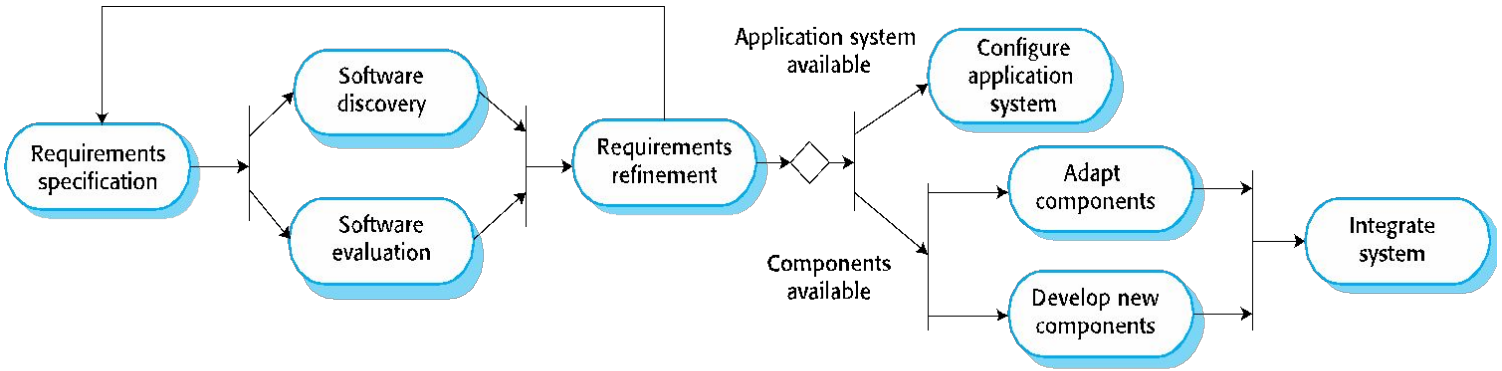
Sistemi applicativi stand-alone (talvolta chiamati COTS) che sono configurati per essere usati in particolari ambienti

Collezione di oggetti che sono sviluppati come un package da integrare con un framework di componenti come .NET or J2EE

Web service che sono sviluppati in accordo agli standard di servizio e disponibili per invocazioni da remoto

Ingegneria del software basata sul riuso

- Specifica dei requisiti
- Scoperta e valutazione del software
- Rifinitura dei requisiti
- Configurazione del Sistema applicativo
- Adattamento ed integrazione dei componenti



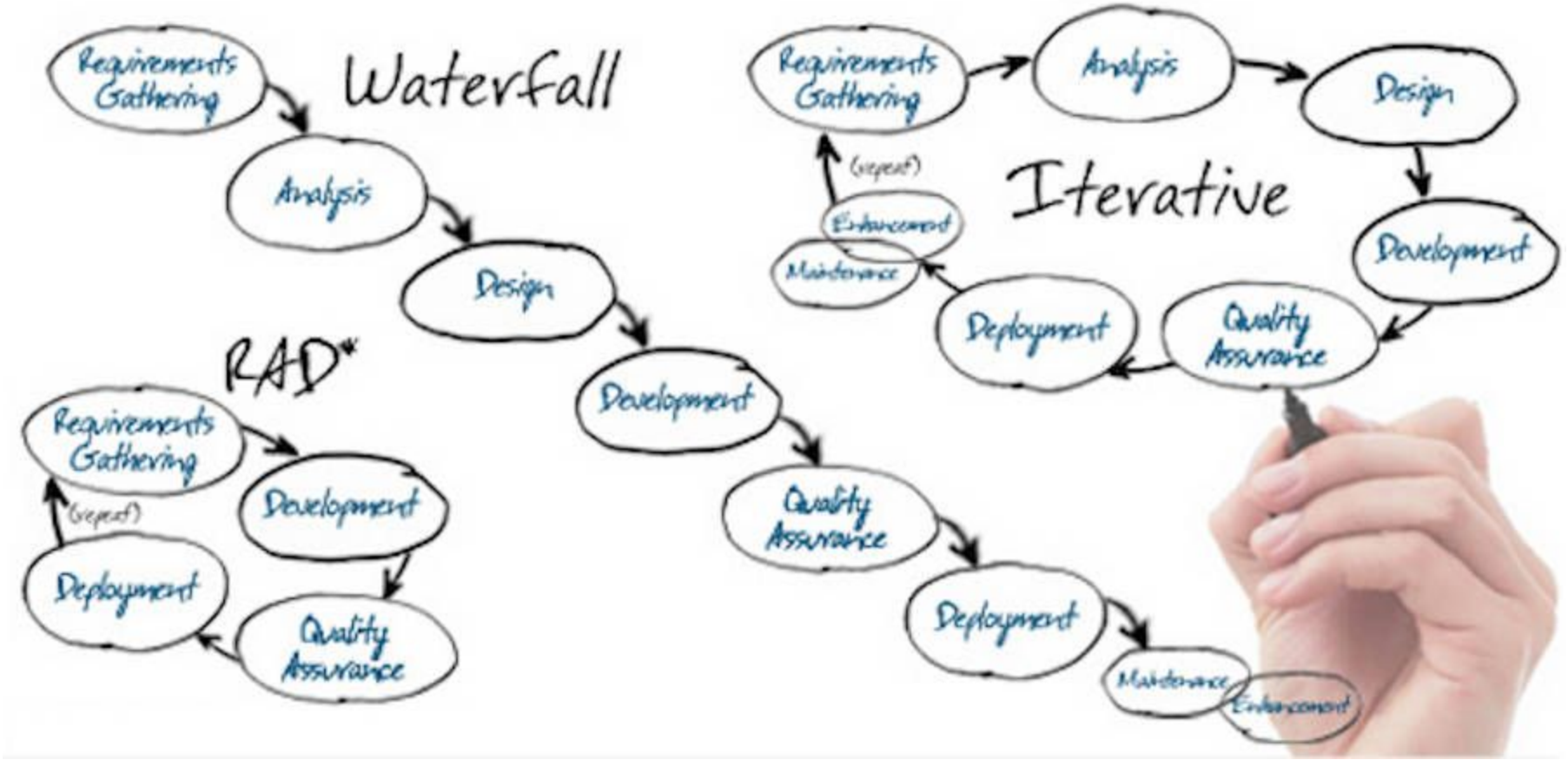
Vantaggi e svantaggi del riuso del software

Vantaggi

- Costi e rischi ridotti poiché è sviluppato meno software da zero
- Consegna e distribuzione del sistema più veloce

Svantaggi

- Compromessi sui requisiti inevitabili
  - il sistema potrebbe non soddisfare le reali esigenze degli utenti
- Perdita di controllo sull'evoluzione degli elementi del sistema riusati


Waterfall
Requirements Gathering
Analysis
Design
Development
Quality Assurance
Deployment
Maintenance
Enhancement
RAD
Requirements Gathering
Development
Deployment
Quality Assurance
(repeat)
Iterative
Requirements Gathering
Analysis
Design
Development
Quality Assurance
Deployment
Maintenance
Enhancement
(repeat)

Modelli sempre più iterativi

- Modello a cascata
- Modello a V
- Rapid prototyping
- Incrementale
- Modello a spirale

Processi agili

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 sviluppatori

Una metodologia agile si basa sui principi del Manifesto di Snowbird, feb 2021.

Agile manifesto (aka Snowbird)

Comunicazione:

- tutte 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; gli analisti mantengano la descrizione formale il più semplice e chiara possibile)

Semplicità:

- è più importante avere sw 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 al cambiamento più che aderire al progetto

Gestione dei cambiamenti

Affrontare le modifiche

- Le modifiche sono inevitabili in tutti i grandi progetti software
  - I cambiamenti aziendali portano a nuovi requisiti o modifiche ai requisiti del sistema
  - Le nuove tecnologie aprono nuove possibilità per il miglioramento dell'implementazione
  - Cambiare le piattaforme comporta modifiche all'applicazione
- Le modifiche comportano una rilavorazione e quindi i costi delle modifiche sommano la rilavorazione (es., rianalizzare i requisiti) e i costi di implementazione della nuova funzionalità

Ridurre i costi di rilavorazione

- E’ necessario **prevedere le modifiche**
  - un’attività del processo sw che prevede i possibili cambiamenti prima che sia necessaria una rilavorazione significativa
    - Ad esempio, potrebbe essere sviluppato un **prototipo** per mostrare ai clienti le caratteristiche chiave del sistema
- **Tolleranza alle modifiche**
  - il processo è progettato in modo che le modifiche possono essere accomodate a costi relativamente bassi
    - di solito, comporta qualche forma di sviluppo incrementale
      - le modifiche proposte possono essere implementate in incrementi che non sono stati ancora sviluppati

Gestire i cambiamenti dei requisiti

Prototipizzazione del sistema

- E’ sviluppata velocemente una versione del sistema o una sua parte per verificare i requisiti del cliente e la fattibilità delle decisioni
- Supporta l’anticipazione delle modifiche

Consegna incrementale

- Sono consegnati al cliente degli incrementi del sistema con i quali è possibile fare commenti o sperimentare
- Supporta ambo le possibilità di evitare i cambiamenti e la loro tolleranza

Prototipizzazione del sistema

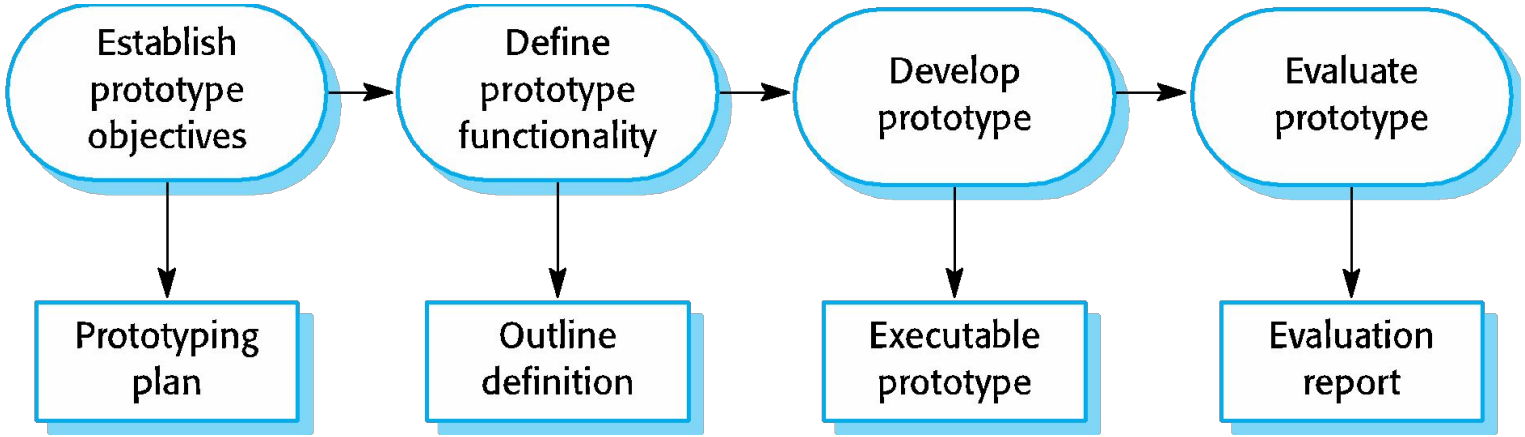
Prototipazione del software

- Un prototipo è una versione iniziale del sistema usato per dimostrare i concetti e determinare opzioni di progettazione
- Un prototipo può essere usato:
  - Nel processo di ingegneria dei requisiti per aiutare la scoperta e la validazione dei requisiti
  - Nei processi di progettazione per esplorare le opzioni e sviluppare un progetto di interfaccia utente
  - Nel processo di testing per eseguire test back-to-back

Benefici della prototipazione

- Migliore usabilità del sistema
- Maggiore corrispondenza con le necessità del cliente
- Migliore qualità di progettazione
- Migliore manutenibilità
- Sforzo di sviluppo inferiore

Processi di sviluppo di un prototipo



Sviluppo di un prototipo

- Può essere basato su linguaggi o strumenti di prototipazione rapida
- Può trascurare alcune funzionalità
  - Il prototipo dovrebbe focalizzarsi su parti del prodotto che non sono ben comprese
  - Il controllo di errori e il loro recupero può non essere incluso nel prototipo
  - Il focus è su requisiti funzionali e non su quelli non-funzionali quali attendibilità e sicurezza

Prototipi throw-away

- I prototipi dovrebbero essere scartati dopo lo sviluppo poiché non rappresentano una buona base per un sistema di produzione:
  - Può essere impossibile raffinare il sistema per soddisfare i requisiti non-funzionali
  - Normalmente, i prototipi non sono documentati
  - La struttura del prototipo è generalmente degradata dopo i rapidi cambiamenti
  - E’ molto probabile che il prototipo non soddisferà gli standard qualitativi organizzativi

Consegna incrementale

Consegna incrementale

- Piuttosto che consegnare il sistema in un unica volta, lo sviluppo e la consegna è suddivisa in incrementi, attraverso i quali, ciascuno, consegna parte della funzionalità richiesta
- I requisiti utente sono resi prioritari e sono inclusi nei primi incrementi i requisiti di priorità più elevata
- Una volta che lo sviluppo di un incremento è avviato, i requisiti sono congelati sebbene i requisiti per gli incrementi successivi possano continuare ad evolvere

Sviluppo incrementale VS consegna incrementale

- Sviluppo incrementale
  - Sviluppa il sistema con incrementi e valuta ogni incremento prima di procedere allo sviluppo dell'incremento successivo
  - Approccio tipico usato nei metodi agili
- Consegna incrementale
  - Distribuisce un incremento usato dagli utenti finali
  - Valutazione più realistica sull'uso pratico del software
  - Difficile da implementare per sostituire il sistema poiché gli incrementi hanno minori funzionalità del sistema da sostituire

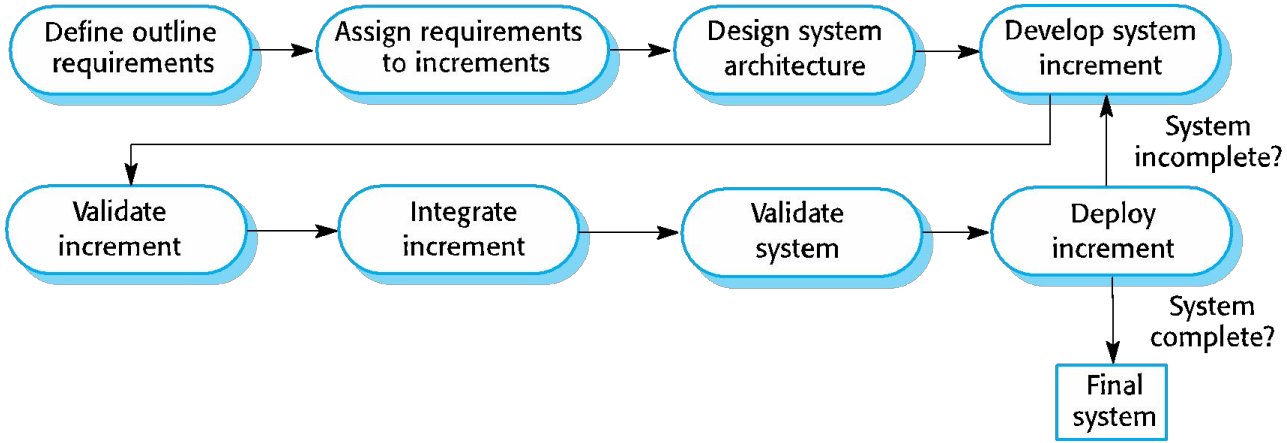
WATERFALL

AGILE

Iterative

Incremental

Consegna incrementale



Vantaggi della consegna incrementale

- I primi incrementi fungono come un prototipo per aiutare nella scoperta dei requisiti per gli incrementi successivi
- Minor rischio di fallimento dell'intero progetto
- I servizi di sistema a maggiore priorità tendono ad essere maggiormente testati

Problemi della consegna incrementale

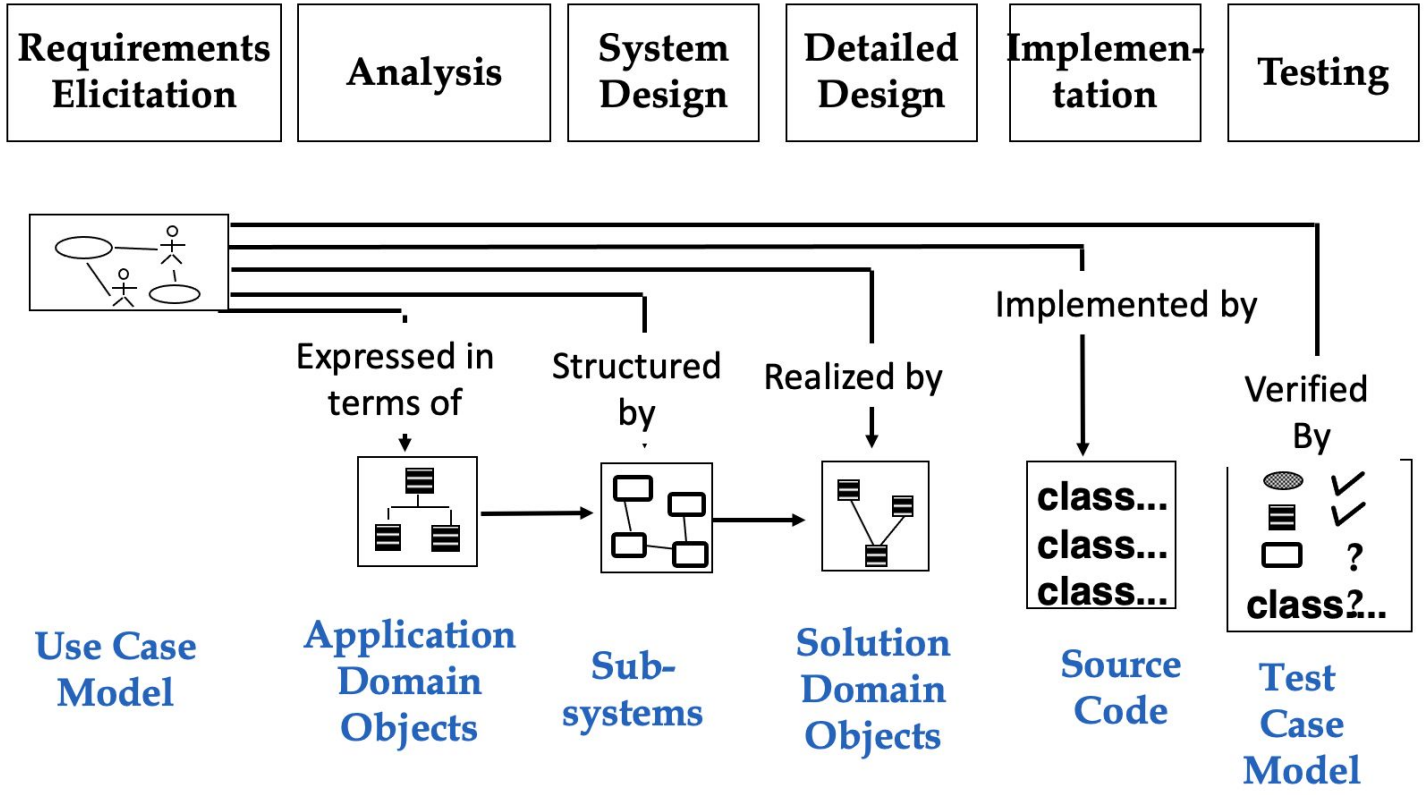
La maggior parte dei sistemi richiede una serie di servizi di base utilizzati da diverse parti del sistema

- Siccome i requisiti non sono definiti nel dettaglio fino a che è implementato un incremento, può essere difficile identificare i servizi comuni necessari a tutti gli incrementi

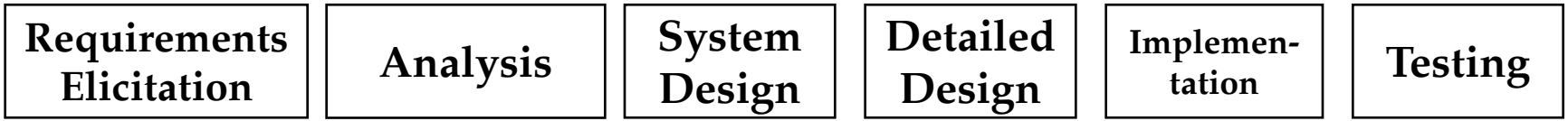
L'essenza dei processi iterativi è che la specifica è sviluppata congiuntamente al software

- Tuttavia, ciò confligge con il modello di acquisto delle organizzazioni, nel quale la specifica dell'intero sistema è parte del contratto dello per lo sviluppo del sistema

Attività del ciclo di vita del software



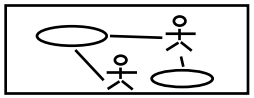
Attività del ciclo di vita del software



Attività del ciclo di vita del software

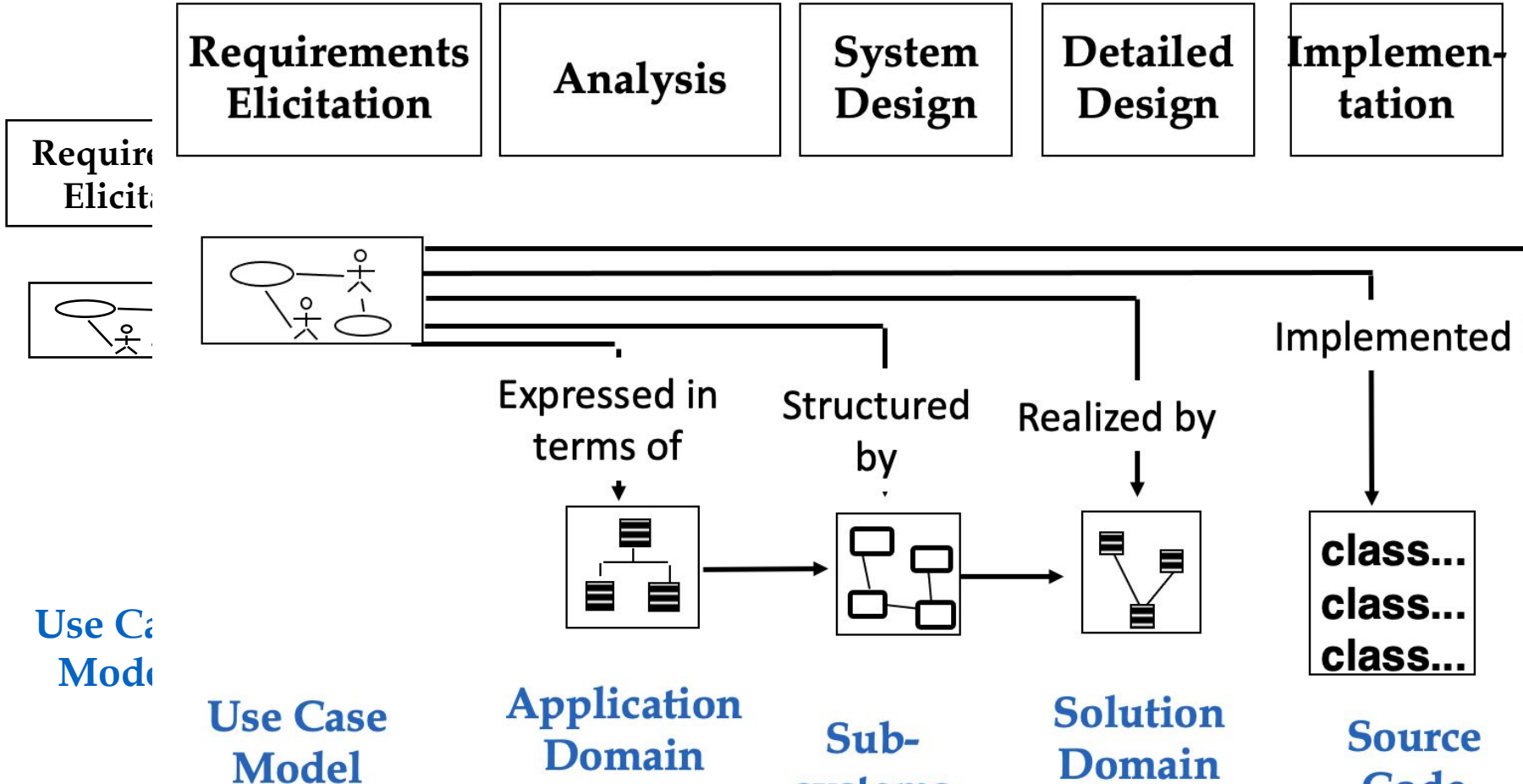
...e i loro modelli

Requirements Elicitation	Analysis	System Design	Detailed Design	Implemen-tation	Testing



Use Case Model

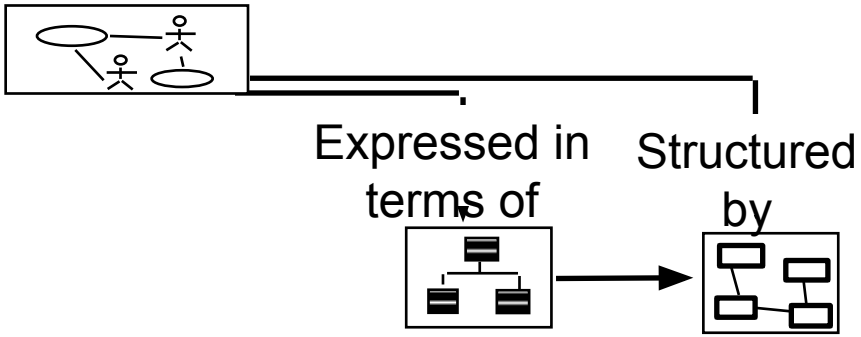
Attività del ciclo di vita del software



Attività del ciclo di vita del software

...e i loro modelli

Requirements Elicitation	Analysis	System Design	Detailed Design	Implemen-tation	Testing



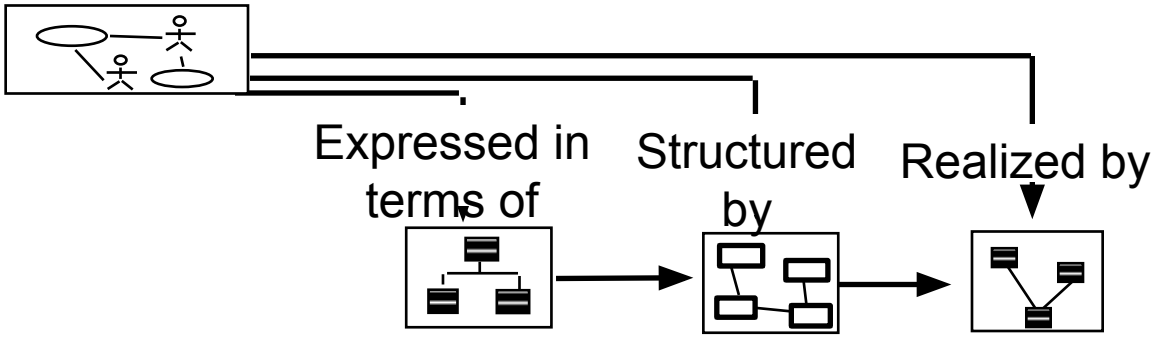
Use Case Model

Application Domain Objects

Sub-systems

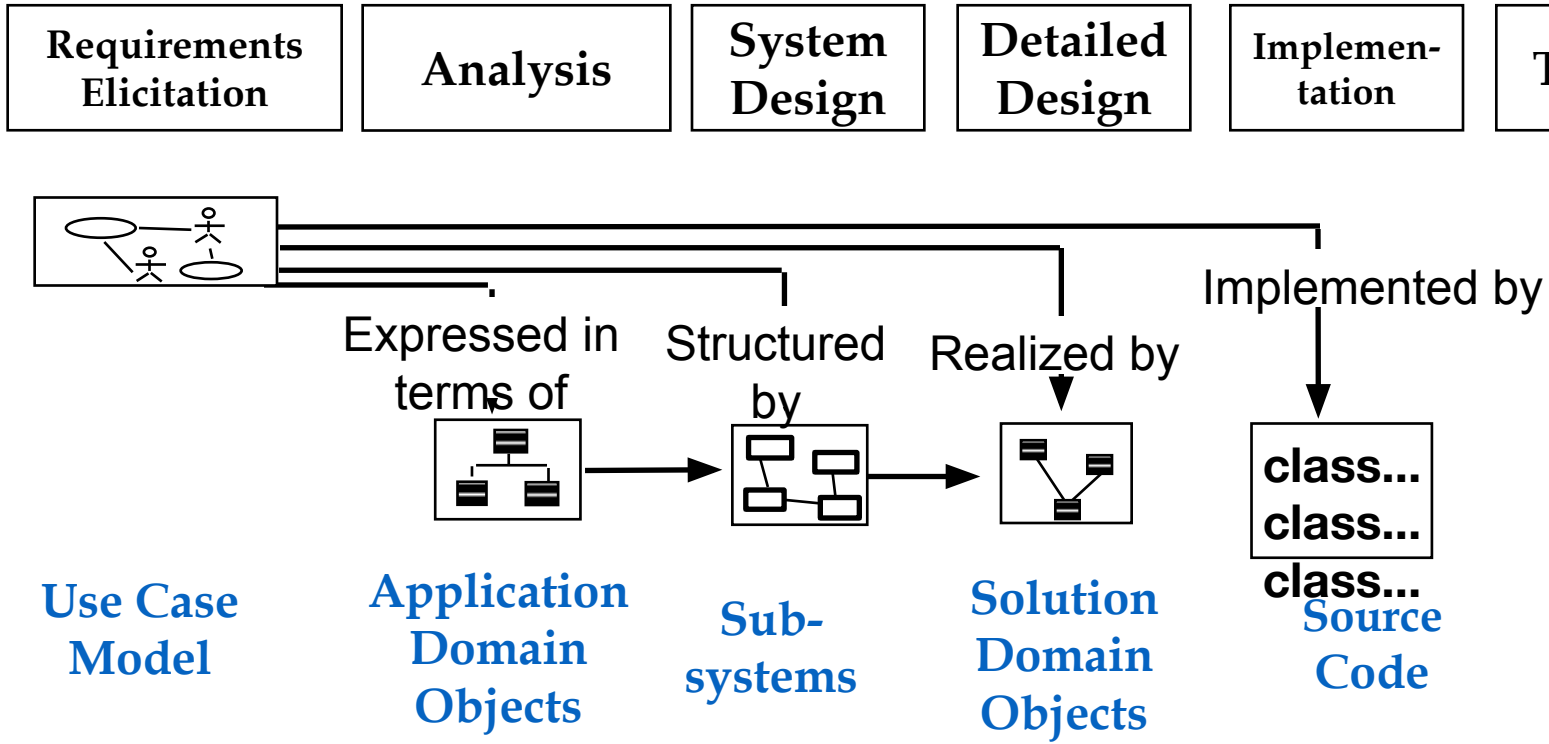
Attività del ciclo di vita del software
...e i loro modelli

Requirements Elicitation	Analysis	System Design	Detailed Design	Implemen-tation	Testing

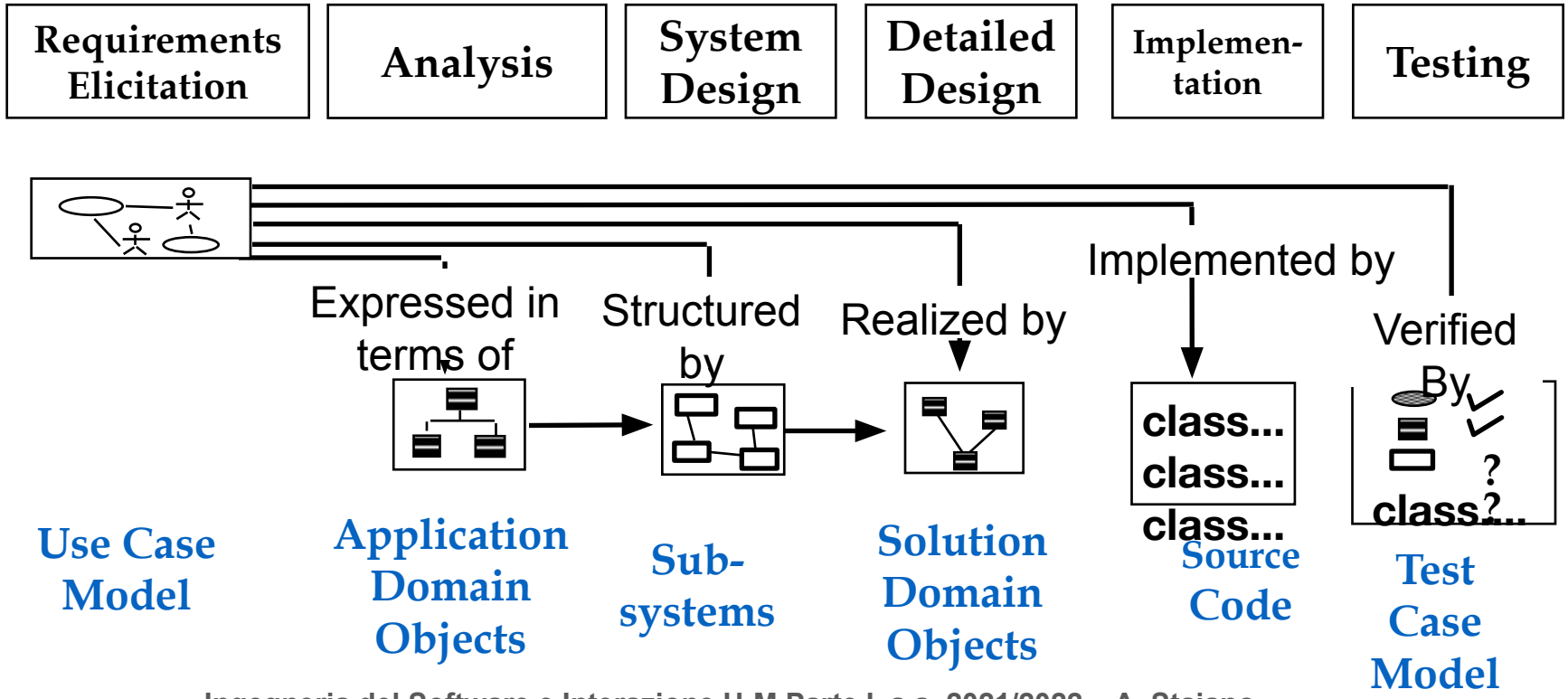


Use Case Model | **Application Domain Objects** | **Sub-systems** | **Solution Domain Objects**

Attività del ciclo di vita del software
...e i loro modelli



Attività del ciclo di vita del software
...e i loro modelli



CICLO DI VITA E INGEGNERIA DEL SOFTWARE

- Design pattern
  - Cosa sono
  - Vantaggi
  - Esempio, come si doocumentano

- Ingegneria del software
  - Perchè serve?
  - Cosa è?

- Qualità del software
  - ISO\IEC 25010
  - Compatibilità
  - Funzionabilità
  - Affidabilità
  - Usabilità
  - Efficienza
  - Manutenibilità
  - Portabilità
  - Sicurezza

- Ciclo di vita del software
  - Modello a cascata
  - Modello a spirale

- Linguaggio UML
  - Diagramma dei casi d'uso
  - Diagramma delle classi
  - Diagramma degli oggetti
  - Diagramma degli stati
  - Diagramma delle attività
  - Diagramma di sequenza

- Metodologie agili
  - Aspetti principali
  - Esempio


CICLO DI VITA E INGEGNERIA DEL SOFTWARE
Design pattern
Cosa sono
Vantaggi
Esempio, come si doocumentano
Qualità del software
ISO\IEC 25010
Compatibilità
Funzionabilità
Affidabilità
Usabiilità
Efficienza
Manutenibilità
Portabilità
Sicurezza
Linguaggio UML
Diagramma dei casi d'uso
Diagramma delle classi
Diagramma degli oggetti
Diagramma degli stati
Diagramma delle attività
Diagramma di sequenza
Diagramma dei componenti
Ingegneria del software
Perchè serve?
Cosa è?
Ciclo di vita del software
Modello a cascata
Modello a spirale
Metodologie agili
Aspetti principali
Esempio