



UNIVERSITÀ DEGLI STUDI
DI MILANO

Ingegneria del Software

Lezione 2: Modelli di ciclo di vita del software

Ma prima qualche informazione logistica

Laboratorio di domani

- ho già detto che
 - non serve prenotarsi... vedremo al momento come siamo messi
 - in prima istanza la divisione in turni verrà fatta secondo parità della matricola
 - turno A: matricole pari, turno B: matricole dispari
- chi vuole usare il proprio computer deve avere installato:
 - git (tool di *versioning* v.2.37+)
 - gradle (tool di *automation building* v.7.4+)
 - IntelliJ (IDE di sviluppo v.2022.2+)
 - JDK (la 17)

Ma prima qualche informazione logistica (2)

Laboratori valutati (compitini)

- ho già detto che
 - bisogna essere frequentanti
- aggiungo che
 - ci sarà tolleranza per max **una** assenza a compitino
 - non può partecipare chi non ha sostenuto con successo esame di Programmazione (1)
 - chi non ha sostenuto con successo esame di programmazione 2... è sconsigliato... ma nel caso verranno messi in coppia tra di loro
- Non valgono più le prove sostenute quando tenevo il corso di ingegneria del software qualche anno fa...

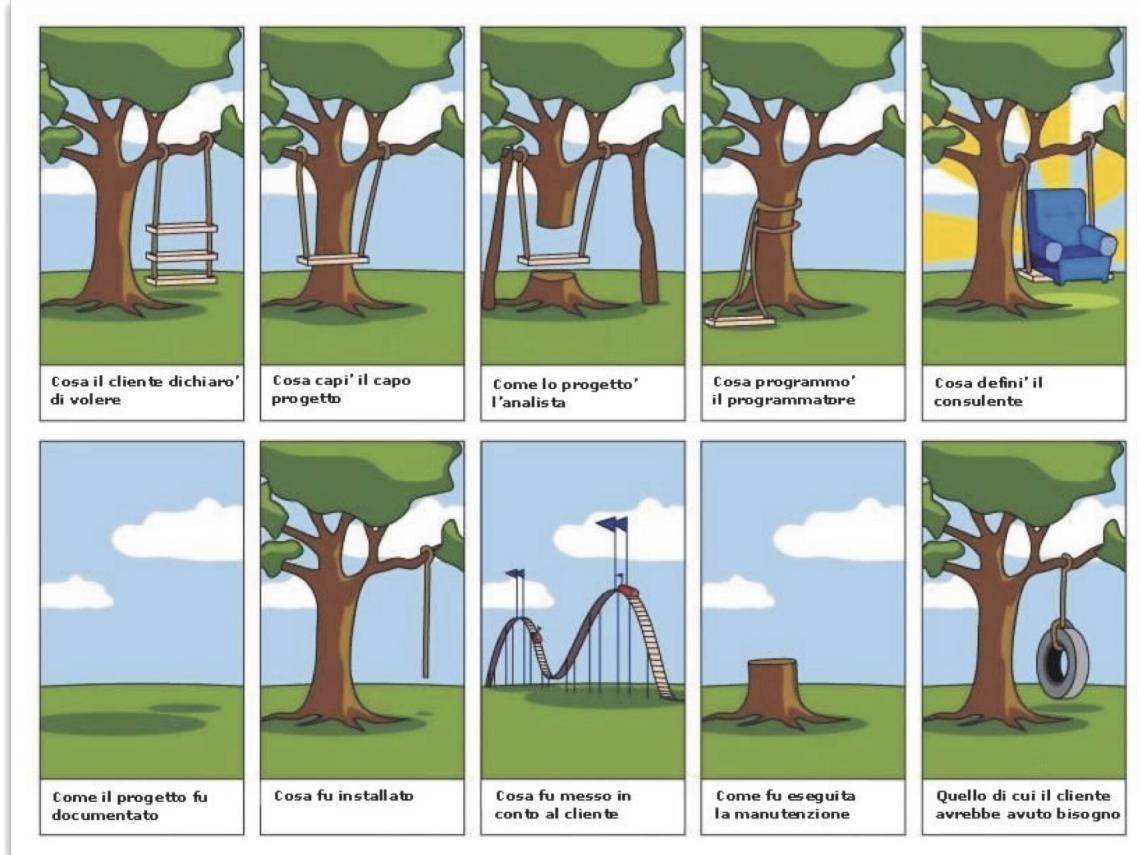
Modello a cascata

- Fine degli anni '50
- ... ma diventato "famoso" negli anni '70
- una famiglia di processi

Modello a cascata

- Forza una **progressione lineare** da una fase alla successiva
- Le varie fasi comunicano attraverso semilavorati
 - permette quindi una **separazione dei compiti**
- È possibile fare una **pianificazione dei tempi** e monitoring dello stato di avanzamento
 - ma a senso unico

Problemi di comunicazione



Manutenzione

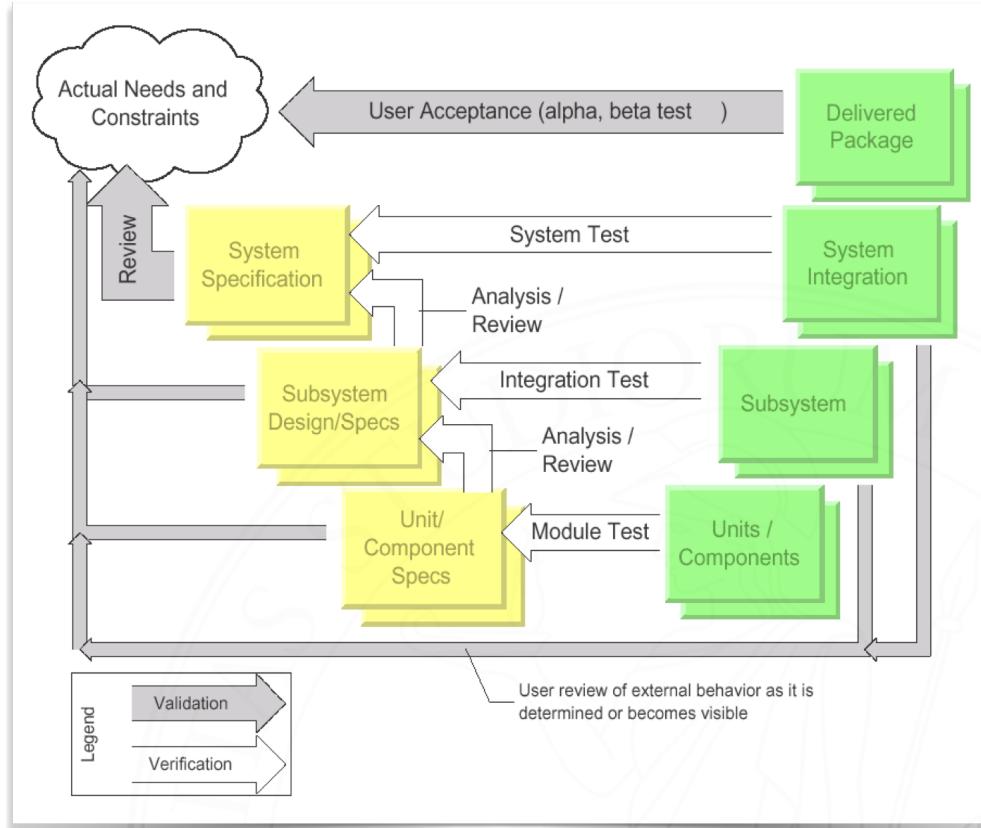
- È una fase prevista?
 - È considerata una eccezione... senz'altro non pianificata
- È possibile farla nel modello a cascata?
 - Non si può tornare indietro!!
 - Niente modifiche a specifiche, codice, documentazione...
- ... ma il 70% dei costi di sviluppo ricadono in questa fase

Problemi

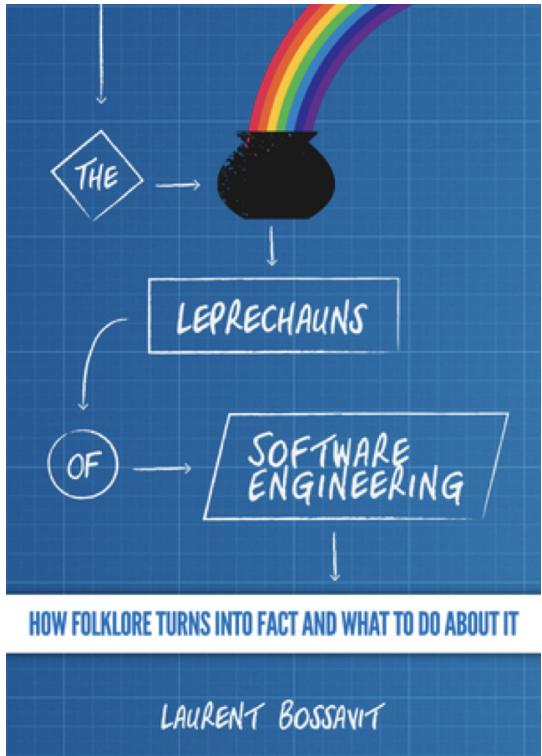
- Rigidità
- Congelamento sotto prodotti
 - specifiche e stime fatte solo nelle prime fasi
- Monoliticità
 - Tutta la pianificazione è orientata ad un singolo rilascio
 - manutenzione fatta solo sul codice

Riposizionamento modello a V

Espande
fase di
testing e
chiarisce
alcune
relazioni

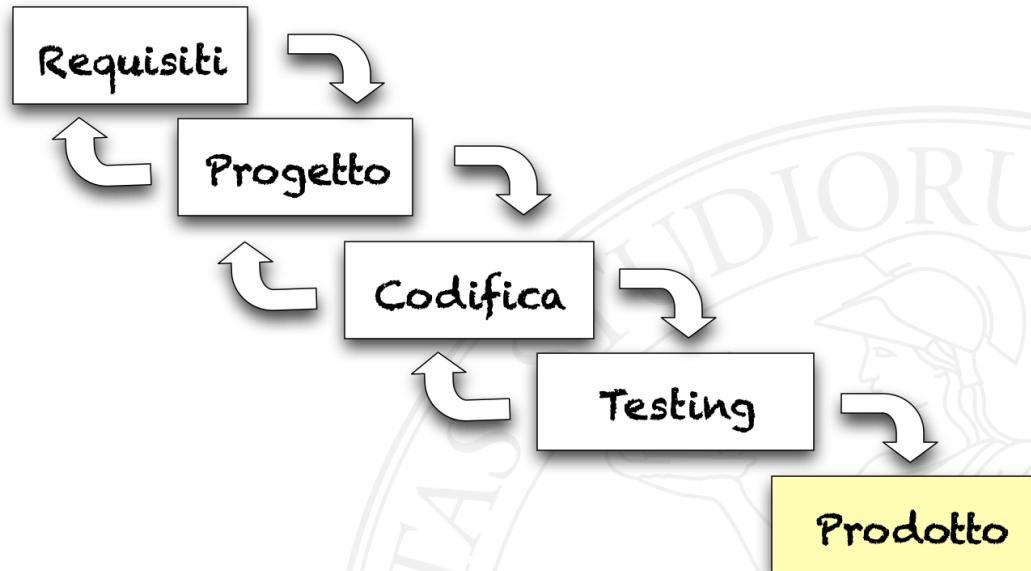


Who's Afraid of The Big Bad Waterfall?



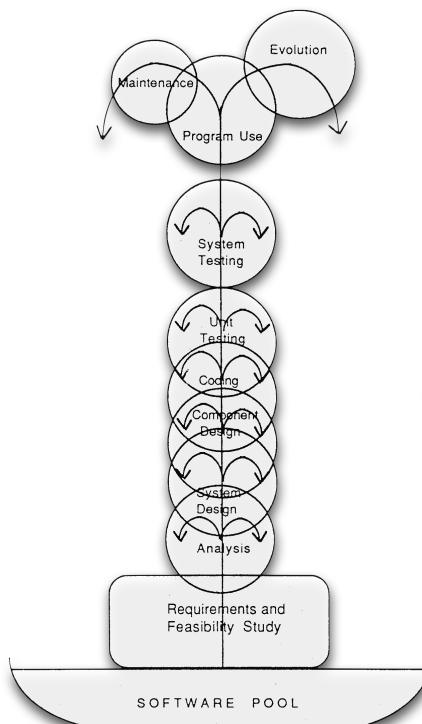
Varianti del processo a cascata

- Con singola retroazione
 - una fase può portare a modifiche nella fase precedente (**iterazione**)



Modello ciclo di vita a fontana

Henderson-Sellers 1993



- con "da capo"
- imprevedibile la profondità di ogni zampillo
- se guardiamo però in cima abbiamo che siamo arrivati a una versione **incrementale**

Modelli iterativi vs. incrementali

- Si parla di **incrementale** quando nelle **iterazioni** viene inclusa la consegna

Ad esempio si può parlare di:

- Implementazione **iterativa**
 - stressa la modularizzazione e l'identificazione di sottosistemi e si ripetono fasi di coding ed integrazione
- Sviluppo **incrementale**
 - viene esteso a tutte le fasi (specifiche comprese)
 - il “modello a cascata” viene eseguito per ogni incremento
 - la fase di manutenzione in senso stretto sparisce diventando semplice riciclo

Modelli prototipali

- Prototipo usa e getta (*throw away*)
- Pubblici o esterni
 - Per capire meglio i requisiti e/o far compiere scelte all'utente interfacce
- Privati o interni
 - Per esplorare nuovi strumenti, linguaggi, diverse scelte per problema difficili progetti pilota, testbed

Boehm's Law (L3)

Prototyping (significantly) reduces requirements and design errors,
especially for user interfaces

“Problemi” con modelli incrementali

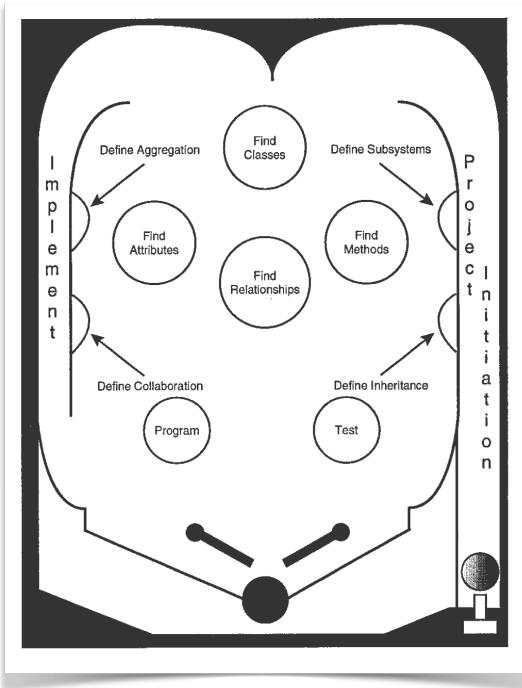
From Waterfall to Iterative Development – A Challenging Transition for Project Managers

- Viene complicato il lavoro di planning
 - Lo stato del processo è meno visibile
 - Bisogna pianificare tutte le iterazioni
- Si riconosce che bisogna rimettere mano a ciò che si è fatto
 - Il sistema può non convergere
- Cosa è una iterazione, quanto dura?
 - Rischio di voler mettere troppo nella prima iterazione
 - Rischio di overhead dovuto a troppe iterazioni
 - Rischio di avere un eccessivo overlapping tra le iterazioni
 - Non si ha tempo di avere feedback dell'utente



Pinball Life-Cycle

Ambler, 1994

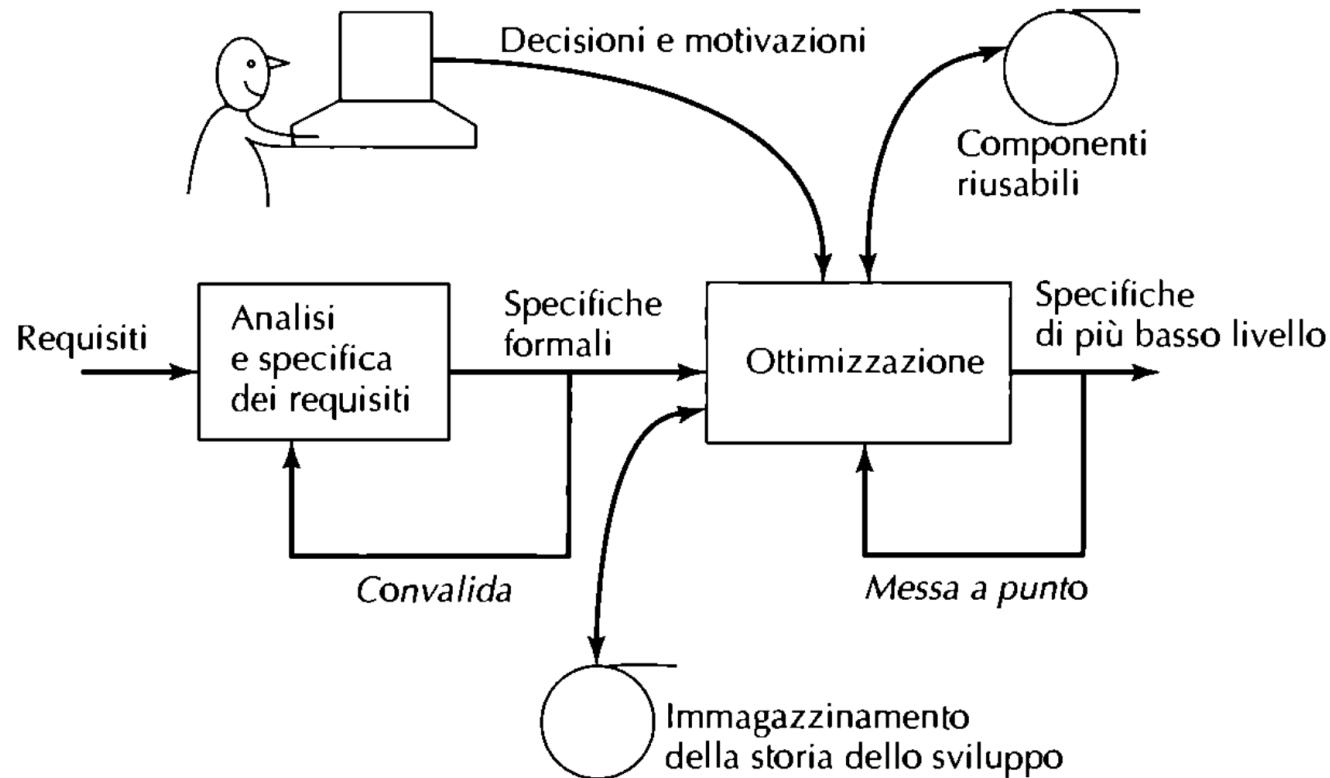


- Indefinito
- Qualunque passo è possibile
- Non vengono imposti vincoli temporali
- non misurabile

Modelli trasformazionali

- Raffinamenti di rappresentazioni formali del problema
 - ad ogni passo vengono specializzate, ottimizzate, rese più concrete attraverso trasformazioni automatiche o comunque controllate
- A partire da specifiche formali si ottiene un prototipo
 - differisce dal prodotto finale per efficienza e completezza
 - passi di trasformazioni (formalmente dimostrabili come corretti) portano ad avere la versione finale

Modelli trasformazionali

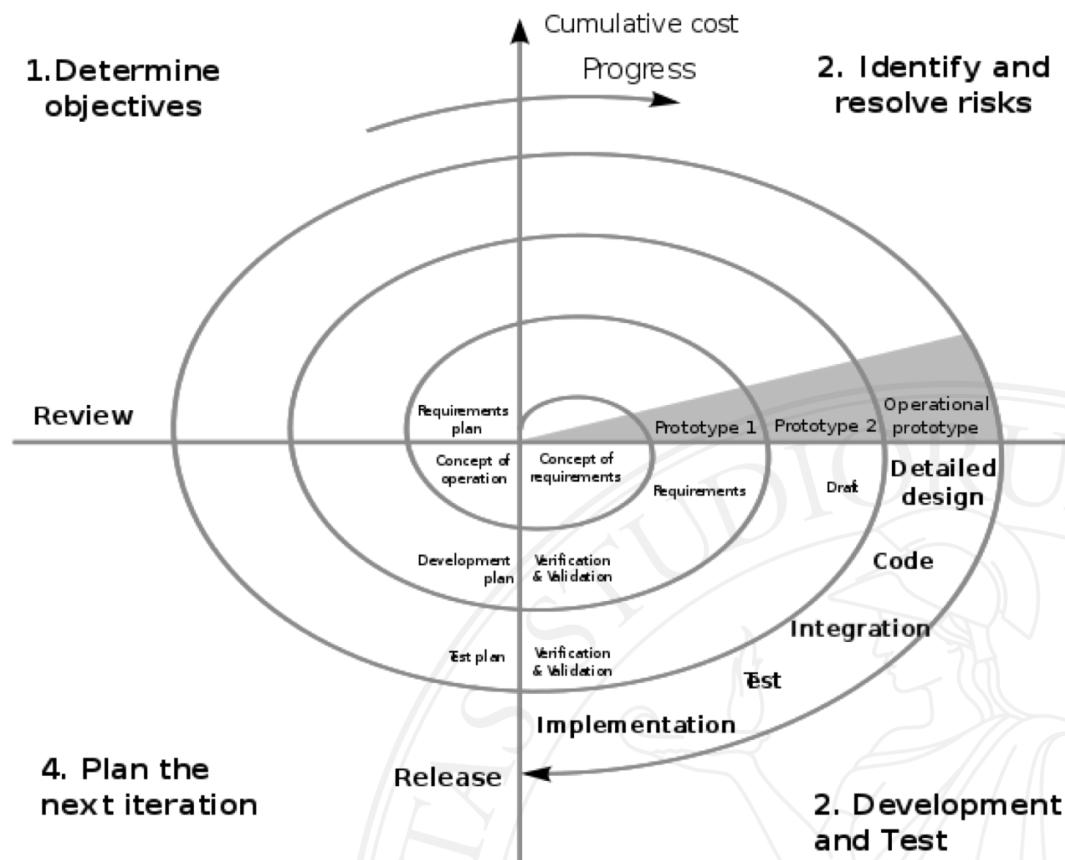


MetaModello a spirale

Boehm 1988

- Non è un modello particolare, ma un framework in cui si possono inquadrare altri modelli
- Guidato dall'analisi dei **rischi**
- Fasi:
 - Determinazioni obiettivi, alternative e vincoli
 - Valutazione alternative, identificazione rischi
 - Sviluppo e verifica
 - Pianificazione fase successiva

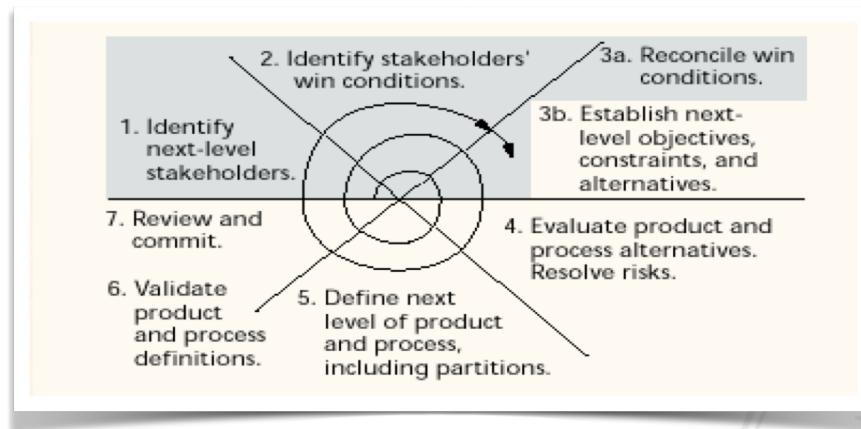




Modello Spirale Win-Win

Boem 1998

- Evidenzia le comunicazioni con i clienti e che non sono di tipo “pacifico” ma necessitano contrattazioni e negoziazione.

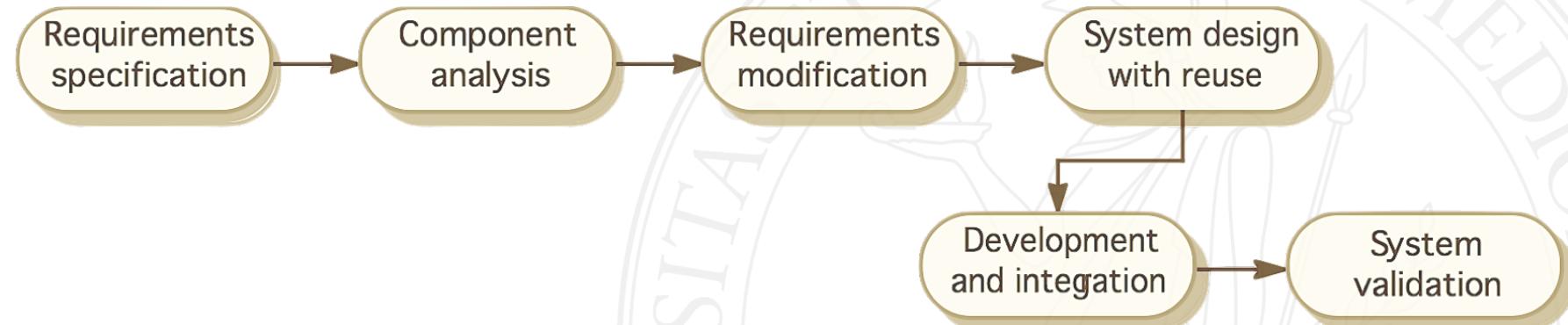


Win Win vuol dire che entrambi le parti vincono (o hanno l'illusione di vincere)

Modello COTS

Component Off The Shelf

- Partire dalla disponibilità interna o sul mercato di moduli preesistenti e creare il sistema a partire da quelli mediante integrazione
- Apre nuovi problemi (tematiche) quali:
 - classificazione dei moduli
 - retrieval dei moduli





UNIVERSITÀ DEGLI STUDI
DI MILANO

Metodologie Agili

eXtreme Programming

Il manifesto dei metodi agili

We are uncovering better ways of developing software by doing it and helping others do it.

Through this work we have come to value:

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Kent Beck

Mike Beedle

Arie van Bennekum

Alistair Cockburn

Ward Cunningham

Martin Fowler

James Grenning

Jim Highsmith

Andrew Hunt

Ron Jeffries

Jon Kern

Brian Marick

Robert C. Martin

Steve Mellor

Ken Schwaber

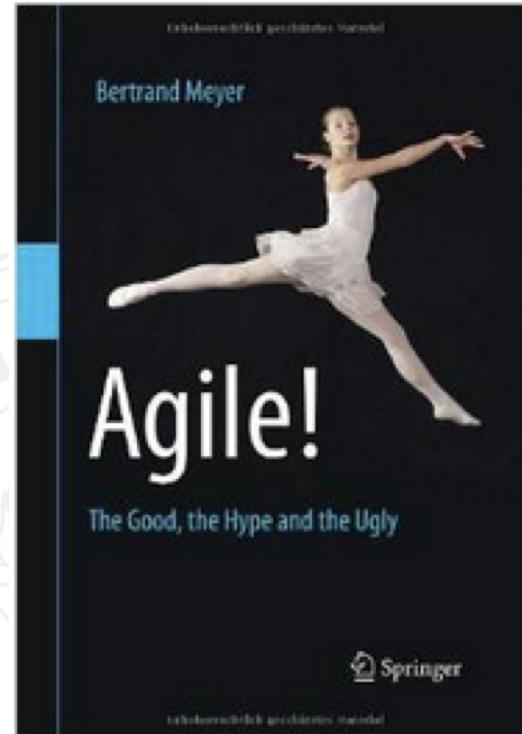
Jeff Sutherland

Dave Thomas

Agile!

Bertrand Meyer

- In realtà non è solo un libro su Agile
- analizza in maniera critica molti concetti metodologici su come si fa a parlare di un processo



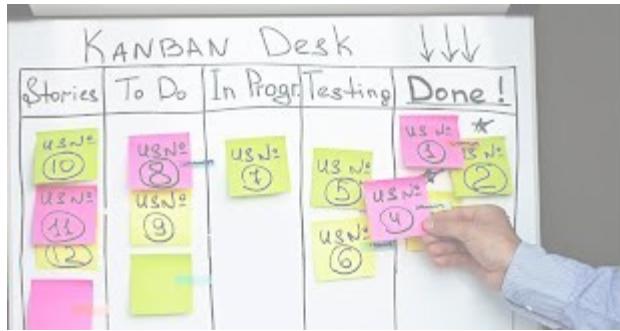
Lean Software

- Nasce da Lean Manufacturing della Toyota

Reduce waste



Kanban



Minimize Work In Progress

Freeze Requirements during short Iterations

Osmotic Communication



eXtreme Programming

Increment then simplify