Il ciclo di vita del SW

Ingegneria del Software

V. Ambriola, G.A. Cignoni,
C. Montangero, L. Semini

Aggiornamenti : T. Vardanega (UniPD)

Contenuti

- Il ciclo di vita del *software*
- Il modello a cascata
- Modelli iterativi
- Il modello a spirale
- Altri modelli
- Seminario: cicli di vita in ISO/IEC 12207

Il concetto di ciclo di vita – 1

- Concezione → sviluppo → utilizzo → ritiro
- Per identificare le attività necessarie
  - Ci basiamo su modelli generici, indipendenti dal progetto / prodotto
  - Decomponiamo le fasi in attività
  - Adottiamo una terminologia consistente
- Per organizzare le attività
  - Identifichiamo le dipendenze tra i loro ingressi e le loro uscite
  - Fissiamo il loro ordinamento nel tempo
  - Definiamo criteri di completamento e di avanzamento

Il concetto di ciclo di vita – 2

- Un ciclo di vita concreto attua processi di progetto
  - Istanziati a partire dal modello dei processi definiti
- Pone vincoli su pianificazione e gestione del progetto
- È indipendente da metodi e strumenti di sviluppo
  - Precede e non segue la loro selezione
- Richiede un sistema di qualità$_{|g|}$ per garantire e misurare conformità$_{|g|}$ e maturità$_{|g|}$
  - Temi che approfondiremo in una prossima lezione

Il ciclo di vita del *software*

Evoluzione dei modelli

- ***"Code-'n-Fix"* : un non-modello**
  - Attività eseguite senza organizzazione preordinata
  - Risulta in progetti caotici non gestiti né gestibili
- **Modelli organizzati (alcuni …)**
  - ***Cascata*** rigide fasi sequenziali
  - ***Incrementale*** realizzazione in più passi
  - ***Evolutivo*** con ripetute iterazioni interne
  - ***A spirale*** contesto allargato e modello astratto
  - **Agile** dinamico, a cicli iterativi e incrementali

Il ciclo di vita del *software*

Modello sequenziale (a cascata) – 1

- **Definito nel 1970 da Winston W. Royce**
  - **"*Managing the development of large software systems: concepts and techniques*"**
- **Successione di fasi rigidamente sequenziali**
  - **Non ammette ritorno a fasi precedenti**
  - **Eventi eccezionali fanno ripartire dall'inizio**
- **Prodotti**
  - **Principalmente "documenti", fino a includere il SW**
    - Emissione e approvazione di documenti come condizione necessaria per l'avvio della fase successiva (modello «*document driven*»)

Il ciclo di vita del *software*

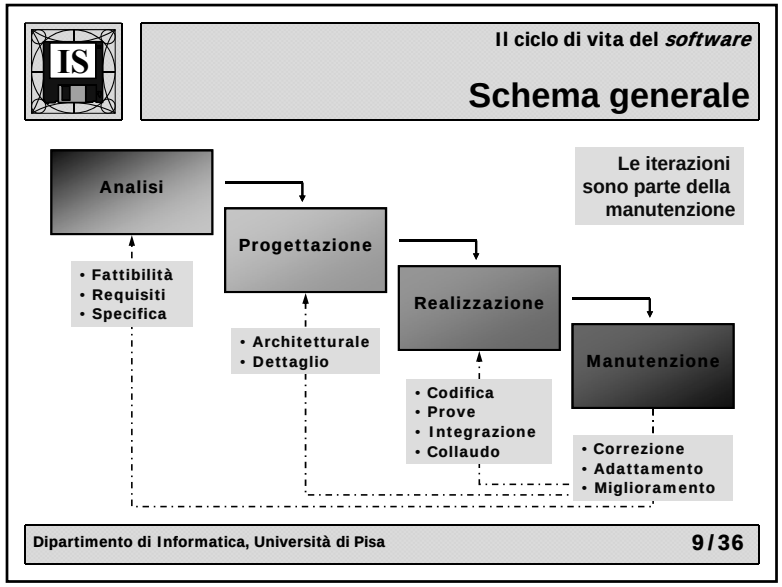
Modello sequenziale (a cascata) – 2

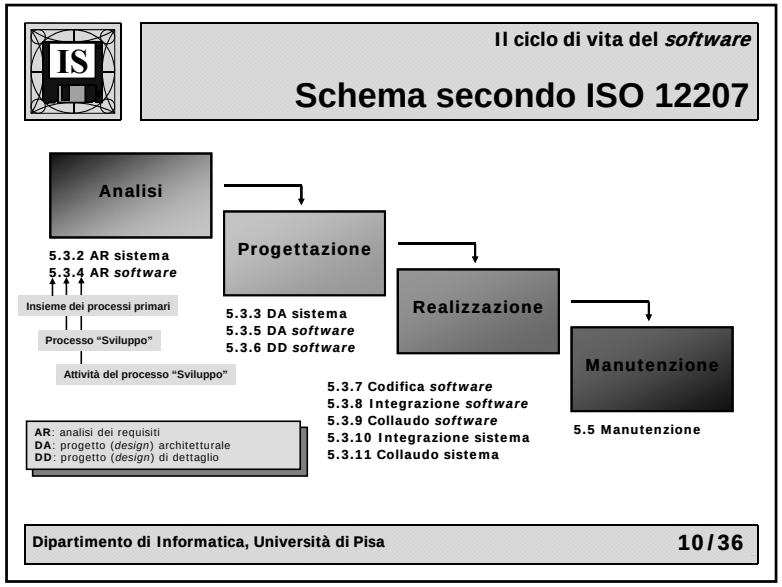
- **Ogni stato di vita (fase) è caratterizzato da pre-condizioni di ingresso e post-condizioni di uscita**
  - **Il loro soddisfacimento è dimostrato da prodotti costituiti prima da documentazione e poi da SW**
- **Le fasi sono distinte e non si sovrappongono**
- **Adatto allo sviluppo di sistemi complessi sul piano organizzativo**
  - **Le iterazioni costano troppo per essere un buon mezzo di mitigazione dei rischi tramite approssimazioni successive**

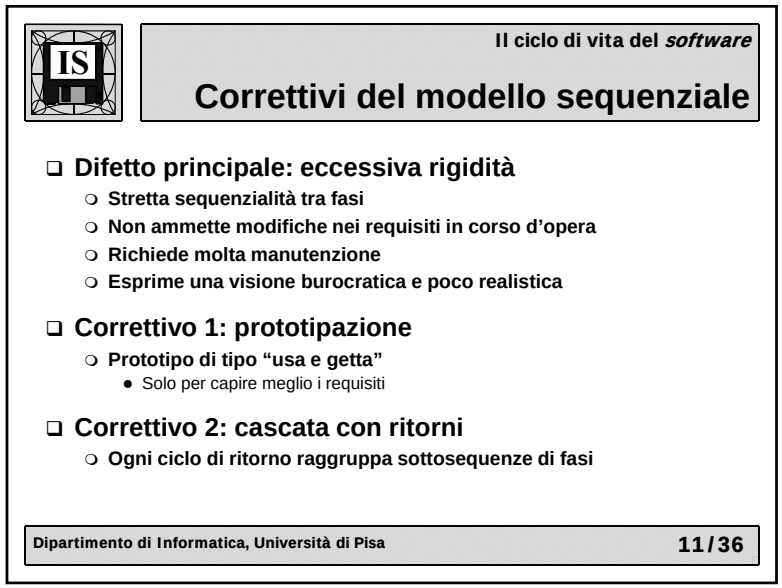
Il ciclo di vita del *software*

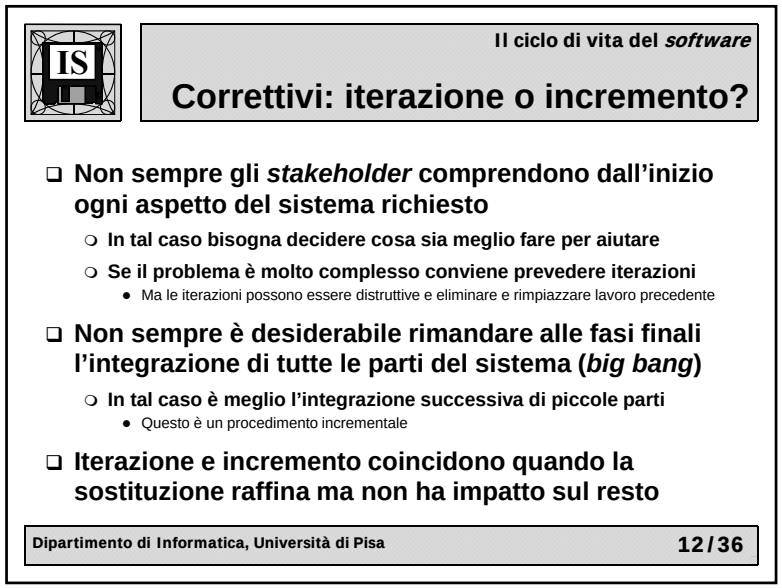
Modello sequenziale (a cascata) – 3

- **Ogni fase viene definita in termini di**
  - **Attività previste e prodotti attesi in ingresso e in uscita**
  - **Contenuti e struttura dei documenti**
  - **Responsabilità e ruoli coinvolti**
  - **Scadenze di consegna dei documenti**
- **Presentano dipendenze causali e temporali**
- **Ciascuna fase comporta azioni specifiche**
  - **Realizzate come attività erogate dai processi coinvolti**


IS
Il ciclo di vita del *software*
Schema generale
Analisi
Progettazione
Realizzazione
Manutenzione
Le iterazioni sono parte della manutenzione
• Fattibilità
• Requisiti
• Specifica
• Architetturale
• Dettaglio
• Codifica
• Prove
• Integrazione
• Collaudo
• Correzione
• Adattamento
• Miglioramento
Dipartimento di Informatica, Università di Pisa
9/36


IS
Il ciclo di vita del *software*
Schema secondo ISO 12207
Analisi
Progettazione
Realizzazione
Manutenzione
5.3.2 AR sistema
5.3.4 AR *software*
Insieme dei processi primari
Processo "Sviluppo"
Attività del processo "Sviluppo"
5.3.3 DA sistema
5.3.5 DA *software*
5.3.6 DD *software*
5.3.7 Codifica *software*
5.3.8 Integrazione *software*
5.3.9 Collaudo *software*
5.3.10 Integrazione sistema
5.3.11 Collaudo sistema
5.5 Manutenzione
AR: analisi dei requisiti
DA: progetto (*design*) architetturale
DD: progetto (*design*) di dettaglio
Dipartimento di Informatica, Università di Pisa
10/36


IS
Il ciclo di vita del *software*
Correttivi del modello sequenziale
Difetto principale: eccessiva rigidità
Stretta sequenzialità tra fasi
Non ammette modifiche nei requisiti in corso d'opera
Richiede molta manutenzione
Esprime una visione burocratica e poco realistica
Correttivo 1: prototipazione
Prototipo di tipo "usa e getta"
Solo per capire meglio i requisiti
Correttivo 2: cascata con ritorni
Ogni ciclo di ritorno raggruppa sottosequenze di fasi
Dipartimento di Informatica, Università di Pisa
11/36


IS
Il ciclo di vita del *software*
Correttivi: iterazione o incremento?
Non sempre gli *stakeholder* comprendono dall'inizio ogni aspetto del sistema richiesto
In tal caso bisogna decidere cosa sia meglio fare per aiutare
Se il problema è molto complesso conviene prevedere iterazioni
Ma le iterazioni possono essere distruttive e eliminare e rimpiazzare lavoro precedente
Non sempre è desiderabile rimandare alle fasi finali l'integrazione di tutte le parti del sistema (*big bang*)
In tal caso è meglio l'integrazione successiva di piccole parti
Questo è un procedimento incrementale
Iterazione e incremento coincidono quando la sostituzione raffina ma non ha impatto sul resto
Dipartimento di Informatica, Università di Pisa
12/36

Vantaggi dei modelli incrementali

- **Possono produrre "valore" a ogni incremento**
  - **Un insieme non vuoto di funzionalità diventa presto disponibile**
  - **I primi incrementi possono corrispondere a fasi di prototipazione**
    - Che aiutano a fissare meglio i requisiti per gli incrementi successivi
- **Ogni incremento riduce il rischio di fallimento**
  - **Senza però azzerarlo a causa dei costi aggiuntivi derivanti dalle eventuali iterazioni**
- **Le funzionalità essenziali sono sviluppate nei primi incrementi**
  - **Attraversano più fasi di verifica**
    - E quindi diventano più stabili con ciascuna iterazione

Vantaggi dei modelli iterativi

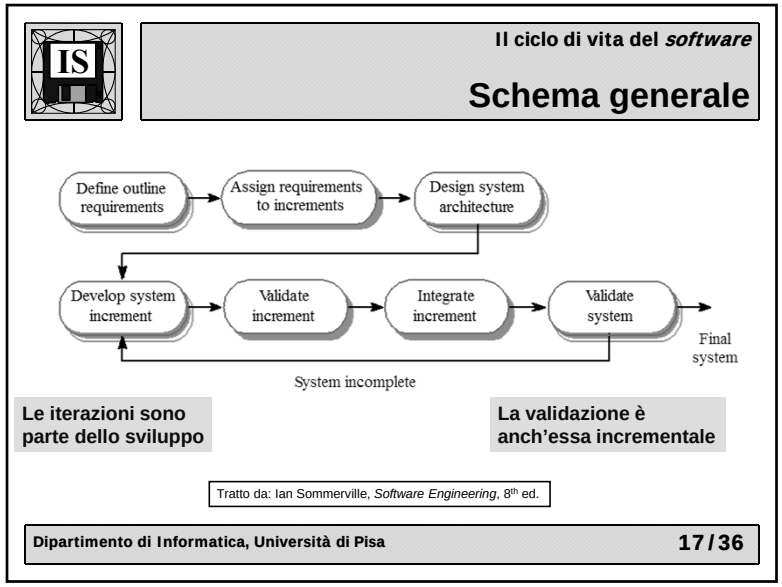
- **Sono applicabili a qualunque modello di ciclo di vita**
- **Consentono maggior capacità di adattamento**
  - **Evoluzione di problemi, soluzioni possibili e tecnologie utilizzabili**
  - **Diversificazione dei requisiti del committente**
  - **I requisiti restano stabili solo in casi molto rari**
- **Soluzione generale**
  - **Decomporre la realizzazione del sistema**
  - **Identificare e trattare prima le componenti più critiche**
    - Quelle più complesse
    - Oppure quelle i cui requisiti non sono sufficientemente chiari
- **Ma il numero di iterazioni va limitato superiormente**

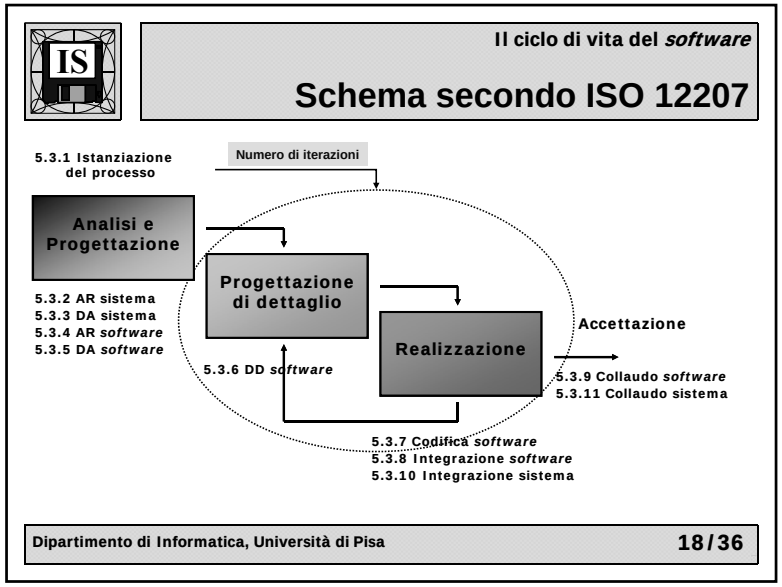
Modello incrementale – 1

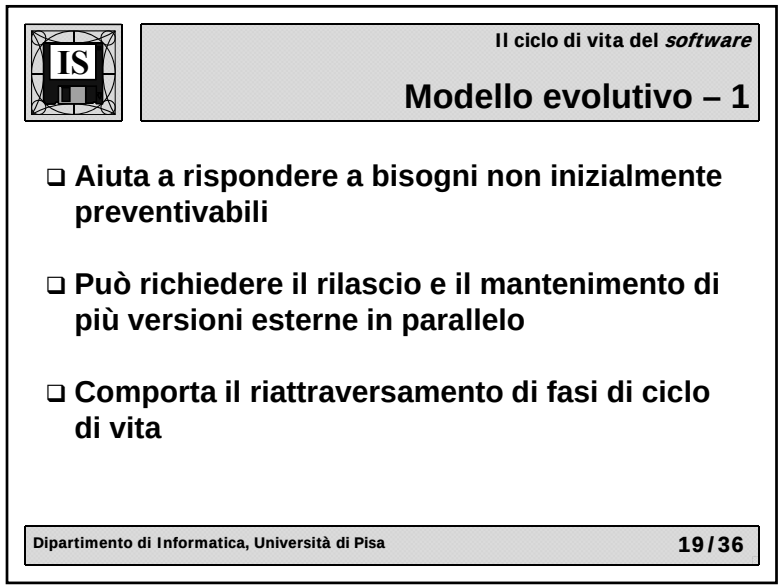
- **Prevede rilasci multipli e successivi**
  - **Ciascuno realizza un incremento di funzionalità**
- **I requisiti utente sono classificati e trattati in base alla loro importanza strategica**
  - **I primi rilasci puntano a soddisfare i requisiti più importanti**
  - **I requisiti importanti sono stabili dall'inizio**
    - Quelli meno importanti possono stabilizzarsi in corso di sviluppo

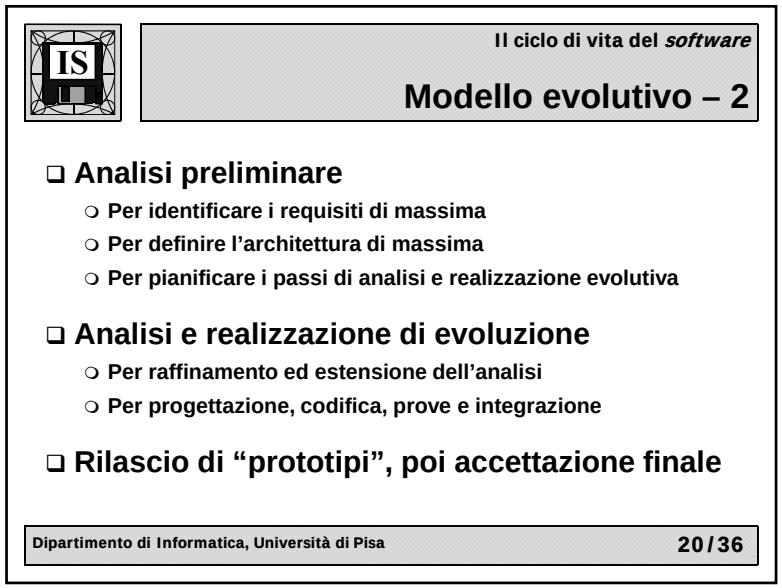
Modello incrementale

- **Analisi e progettazione architetturale non ripetute**
  - **Requisiti principali sono identificati e fissati completamente**
  - **Architettura del sistema è identificata e fissata definitivamente**
  - **Essenziale per pianificare i cicli di incremento**
- **La realizzazione è incrementale**
  - **Attività di progettazione di dettaglio, codifica e prove**
  - **Prima i requisiti essenziali poi quelli desiderabili**
  - **Integrazione, collaudo, eventuale rilascio**


Il ciclo di vita del *software*
Schema generale
Define outline requirements
Assign requirements to increments
Design system architecture
Develop system increment
Validate increment
Integrate increment
Validate system
Final system
System incomplete
Le iterazioni sono parte dello sviluppo
La validazione è anch'essa incrementale
Tratto da: Ian Sommerville, *Software Engineering*, 8th ed.
Dipartimento di Informatica, Università di Pisa
17/36

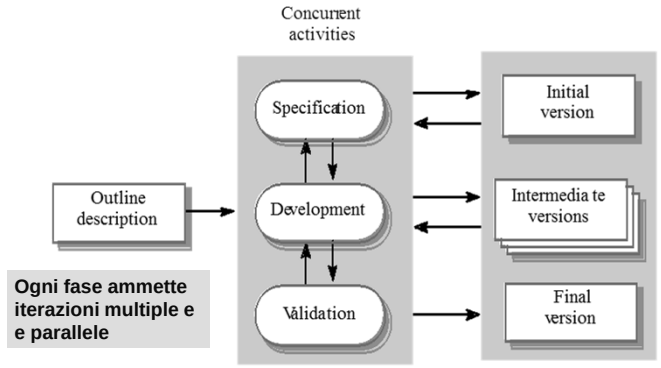

Il ciclo di vita del *software*
Schema secondo ISO 12207
5.3.1 Istanziazione del processo
Numero di iterazioni
Analisi e Progettazione
5.3.2 AR sistema
5.3.3 DA sistema
5.3.4 AR *software*
5.3.5 DA *software*
Progettazione di dettaglio
5.3.6 DD *software*
Realizzazione
Accettazione
5.3.9 Collaudo *software*
5.3.11 Collaudo sistema
5.3.7 Codifica *software*
5.3.8 Integrazione *software*
5.3.10 Integrazione sistema
Dipartimento di Informatica, Università di Pisa
18/36


Il ciclo di vita del *software*
Modello evolutivo – 1
Aiuta a rispondere a bisogni non inizialmente preventivabili
Può richiedere il rilascio e il mantenimento di più versioni esterne in parallelo
Comporta il riattraversamento di fasi di ciclo di vita
Dipartimento di Informatica, Università di Pisa
19/36


Il ciclo di vita del *software*
Modello evolutivo – 2
Analisi preliminare
Per identificare i requisiti di massima
Per definire l'architettura di massima
Per pianificare i passi di analisi e realizzazione evolutiva
Analisi e realizzazione di evoluzione
Per raffinamento ed estensione dell'analisi
Per progettazione, codifica, prove e integrazione
Rilascio di "prototipi", poi accettazione finale
Dipartimento di Informatica, Università di Pisa
20/36

Il ciclo di vita del *software*

Schema generale

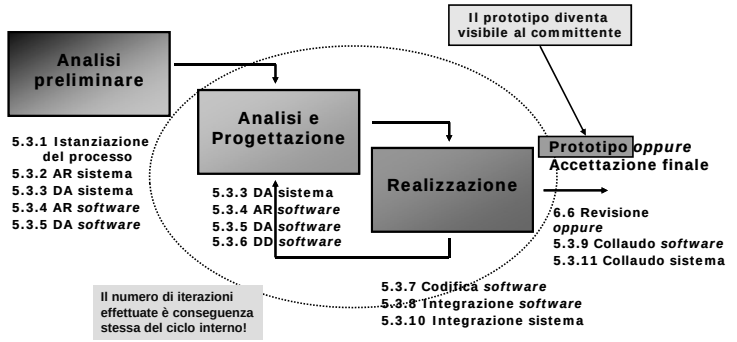


Ogni fase ammette iterazioni multiple e e parallele

Tratto da: Ian Sommerville, *Software Engineering*, 8th ed.

Il ciclo di vita del *software*

Schema secondo ISO 12207



Il ciclo di vita del *software*

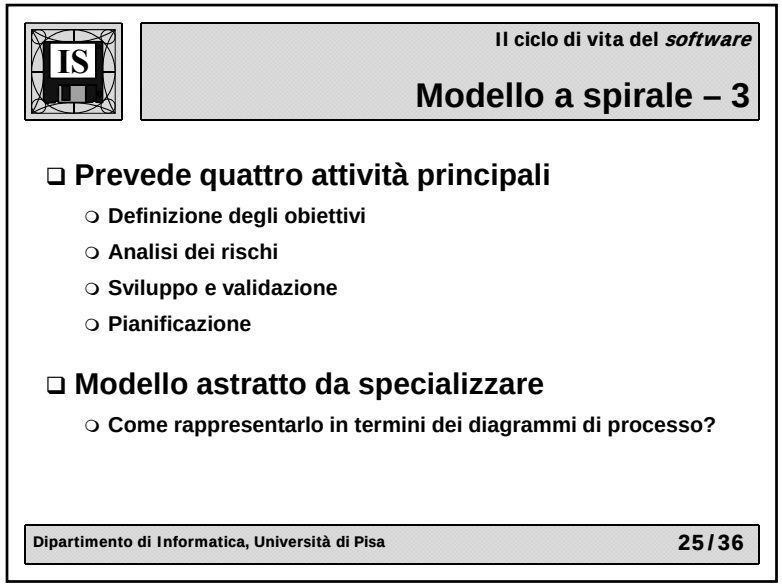
Modello a spirale – 1

- Proposto da Barry W Boehm
  - IEEE Computer, maggio 1988
- Per miglior controllo dei rischi di progetto
- Cicli interni rapidi e ripetuti
  - Dedicati ad analisi e sviluppi prototipali
- Cicli esterni che aderiscono a un qualsiasi altro modello standard di ciclo di vita

Il ciclo di vita del *software*

Modello a spirale – 2

- Pone grande attenzione sugli aspetti gestionali
  - Pianificazione delle fasi
  - Analisi dei rischi (modello «risk driven»)
- Richiede forte interazione tra committente e fornitore
  - Committente: definizione degli obiettivi
definizione dei vincoli sulla pianificazione
  - Fornitore: sviluppo e validazione
  - Entrambi: analisi dei rischi



Il ciclo di vita del *software*

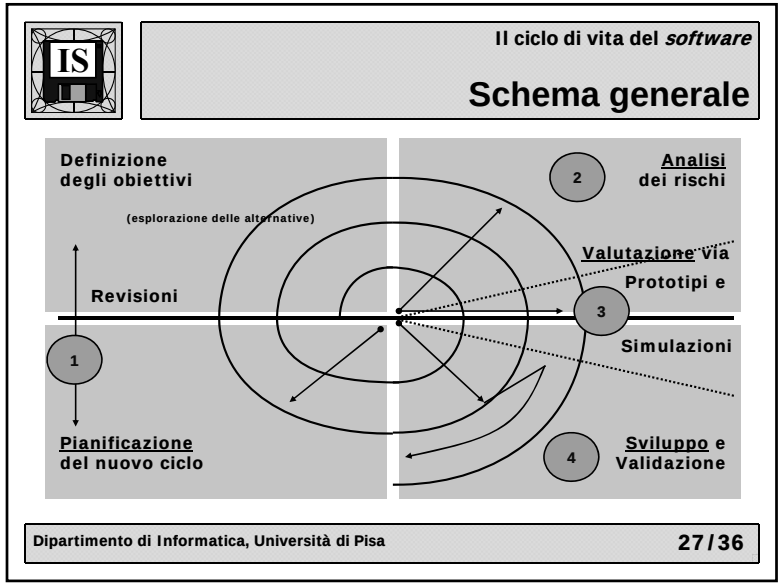
Modello a spirale – 3

- **Prevede quattro attività principali**
  - Definizione degli obiettivi
  - Analisi dei rischi
  - Sviluppo e validazione
  - Pianificazione
- **Modello astratto da specializzare**
  - Come rappresentarlo in termini dei diagrammi di processo?

Il ciclo di vita del *software*

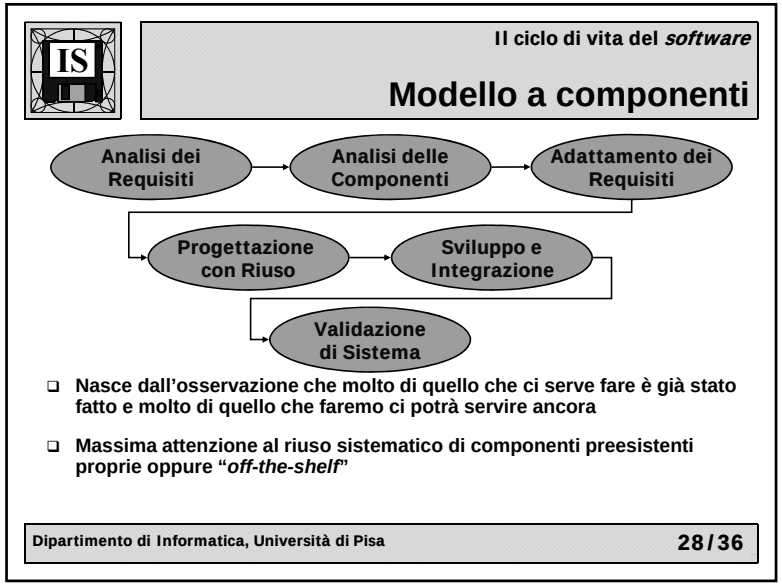
Fasi del modello a spirale

- **Definizione degli obiettivi**
  - Requisiti, rischi, strategia di gestione
- **Analisi dei rischi**
  - Studio delle conseguenze
  - Valutazione delle alternative con l'ausilio di prototipi e simulazioni
- **Sviluppo e validazione**
  - Realizzazione del prodotto
- **Pianificazione**
  - Decisione circa il proseguimento
  - Pianificazione del proseguimento



Il ciclo di vita del *software*

Schema generale



Il ciclo di vita del *software*

Modello a componenti

- **Nasce dall'osservazione che molto di quello che ci serve fare è già stato fatto e molto di quello che faremo ci potrà servire ancora**
- **Massima attenzione al riuso sistematico di componenti preesistenti proprie oppure "*off-the-shelf*"**

Metodi agili – 1

- **Nascono alla fine del ’90 come reazione alla eccessiva rigidità dei modelli allora in vigore**
  - **http://agilemanifesto.org/**
- **Quattro principi fondanti**
  1) ***Individuals and interactions over processes and tools***
     - L’eccessiva rigidità ostacola l’emergere del valore
  2) ***Working sofware over comprehensive documentation***
     - La documentazione non sempre corrisponde a SW funzionante
  3) ***Customer collaboration over contract negotiation***
     - L’interazione con gli stakeholder va incentivata
  4) ***Responding to change over following a plan***
     - La capacità di adattamento al cambiare delle situazioni è importante

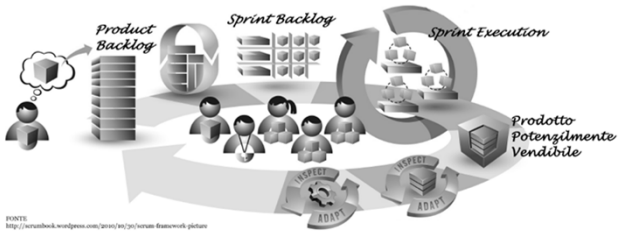
Metodi agili – 2

- **L’idea base è il concetto di “*user story*”**
  - **Un compito significativo che l’utente vuole svolgere con il SW richiesto**
- **Ogni “*user story*” è definita da**
  - **Un documento di descrizione**
  - **La minuta di conversazioni con il cliente (gli *stakeholder* in generale) per fissare la comprensione comune**
  - **La strategia da usare per confermare che il SW realizzato soddisfi gli obiettivi**

Metodi agili – 3

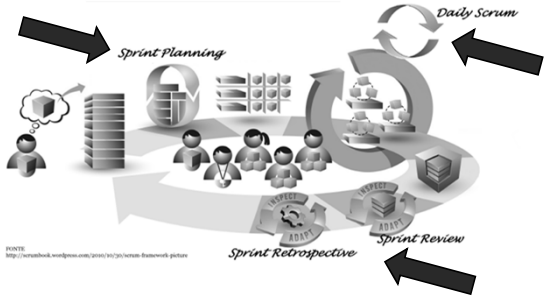
- **Migliori assunti base**
  - **È possibile suddividere il lavoro da fare in piccoli incrementi a valore aggiunto che possono essere sviluppati indipendentemente**
  - **È possibile sviluppare questi incrementi in una sequenza continua dall’analisi dei requisiti all’integrazione**
- **Obiettivi strategici**
  - **Poter costantemente dimostrare al cliente quanto è stato fatto**
  - **Verificare l’avanzamento tramite progresso reale**
  - **Dare agli sviluppatori la soddisfazione del risultato**
  - **Assicurare che l’intero prodotto SW è ben integrato e verificato**
- **Esempi**
  - **Scrum (caos organizzato), Kanban (*just-in-time*), Scrumban**

Scrum – 1



- **Product Backlog**
  Requisiti e funzionalità del prodotto
- **Sprint Backlog**
  Insieme di storie del prossimo sprint
- **Sprint**
  Fase operativa di sviluppo
  Durata media 2 - 4 settimane
  Prodotto potenzialmente vendibile

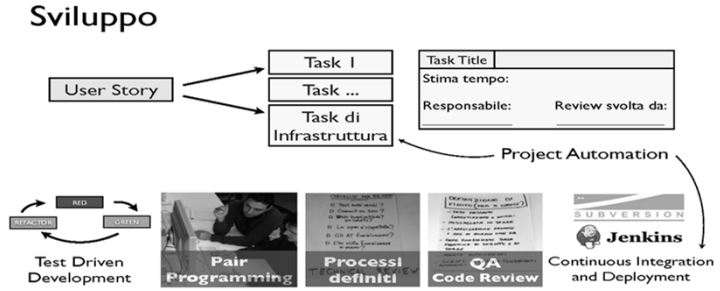
Scrum – 2



- **Sprint Planning**
  Pianificazione dello sprint
- **Sprint Review**
  Controllo prodotti dello sprint
- **Daily Scrum**
  Controllo giornaliero avanzamento
- **Sprint Retrospective**
  Controllo qualità sullo sprint

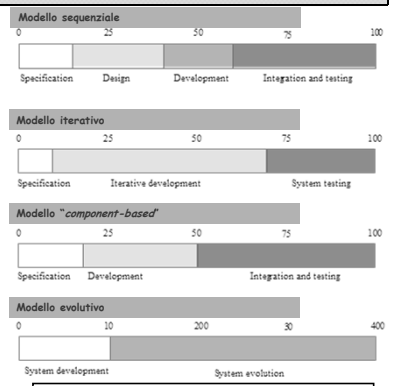
Dal dire al fare ...

Un assaggio di metodo agile in stage (D. Giorgini, 2011)



Ripartizione dei costi sui modelli

- **Applicazioni serie "normali"**
  - **~ 60% dei costi allo sviluppo**
  - **~ 40% alla qualifica**
- **I costi complessivi variano al variare del dominio e del tipo di sistema**
- **La ripartizione dei costi sulle fasi varia al variare del modello e del dominio**
  - **Sistemi critici: > 60% qualifica**



Tratto da: Ian Sommerville, *Software Engineering*, 8th ed.

Riferimenti

- **W.W. Royce, "Managing the development of large software systems: concepts and techniques", Atti della conferenza "Wescon '70", agosto 1970**
- **B.W. Bohem, "A spiral model of software development and enhancement", IEEE Software, maggio 1998**
- **Center for Software Engineering, http://sunset.usc.edu/research/spiral_model**
- **ISO/IEC TR 15271:1998, Information Technology – Guide for ISO/IEC 12207**
- **Scrum: http://www.scrumalliance.org/learn_about_scrum**