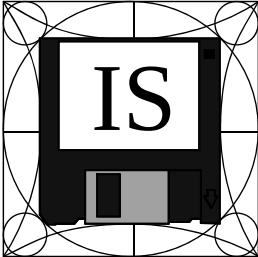




## Il ciclo di vita del SW

Ingegneria del Software  
V. Ambriola, G.A. Cignoni,  
C. Montanero, L. Semini  
Aggiornamenti : T. Vardanega (UniPD)



Dipartimento di Informatica, Università di Pisa1 / 36



Il ciclo di vita del *software*

## 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

Dipartimento di Informatica, Università di Pisa2 / 36



Il ciclo di vita del *software*

## 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

Dipartimento di Informatica, Università di Pisa3 / 36



Il ciclo di vita del *software*

## Il concetto di ciclo di vita – 2

- ❑ Un ciclo di vita concreto attua processi di progetto
  - Istanziati a partire dal modello dei processi definiti
- ❑ È fonte di 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
  - Come quello di cui abbiamo parlato

Dipartimento di Informatica, Università di Pisa4 / 36



Il ciclo di vita del *software*

## Evoluzione dei modelli

- ❑ “Code-’n-Fix” : un modello in negativo
  - 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

Dipartimento di Informatica, Università di Pisa5 / 36



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 consente 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»)

Dipartimento di Informatica, Università di Pisa6 / 36



Il ciclo di vita del *software*

**Modello sequenziale (a cascata) – 2**

- ❑ Ogni fase è caratterizzata da pre-condizioni di ingresso e post-condizioni di uscita
  - Il loro raggiungimento 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
  - Dove le iterazioni costano troppo come mezzo di attenuazione dei rischi

Dipartimento di Informatica, Università di Pisa

7 / 36



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

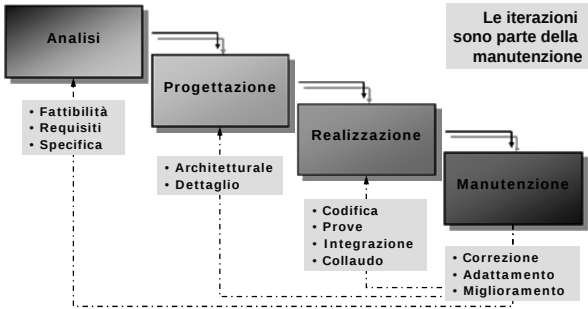
Dipartimento di Informatica, Università di Pisa

8 / 36



Il ciclo di vita del *software*

**Schema generale**



Dipartimento di Informatica, Università di Pisa

9 / 36



Il ciclo di vita del *software*

**Schema secondo ISO 12207**



Dipartimento di Informatica, Università di Pisa

10 / 36



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 un sottosequenza di fasi

Dipartimento di Informatica, Università di Pisa

11 / 36



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 tale 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



Il ciclo di vita del *software*

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

Dipartimento di Informatica, Università di Pisa

13 / 36



Il ciclo di vita del *software*

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
- ❑ Il numero di iterazioni deve stare entro un massimo predeterminato

Dipartimento di Informatica, Università di Pisa

14 / 36



Il ciclo di vita del *software*

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

Dipartimento di Informatica, Università di Pisa

15 / 36



Il ciclo di vita del *software*

Modello incrementale

- ❑ Analisi e progettazione architetturale non ripetute
  - Requisiti principali sono identificati e fissati completamente
  - Architettura del sistema è identificata e fissata definitivamente
  - I cicli di incremento sono pianificati
    - Controllo dei tempi e dei costi
- ❑ La realizzazione è incrementale
  - Attività di progettazione di dettaglio, codifica e prove
  - Prima i requisiti essenziali poi quelli desiderabili
  - Integrazione, collaudo, eventuale rilascio

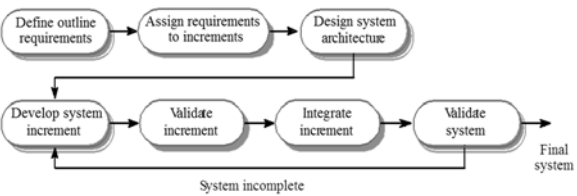
Dipartimento di Informatica, Università di Pisa

16 / 36



Il ciclo di vita del *software*

Schema generale



Le iterazioni sono parte dello sviluppo

Tratto da: Ian Sommerville, *Software Engineering*, 8<sup>a</sup> ed.

Dipartimento di Informatica, Università di Pisa

17 / 36



Il ciclo di vita del *software*

Schema secondo ISO 12207



Dipartimento di Informatica, Università di Pisa

18 / 36



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

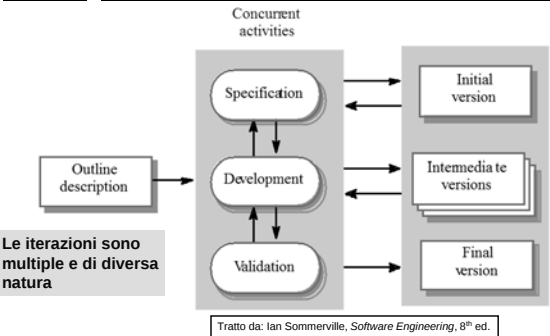


Modello evolutivo – 2

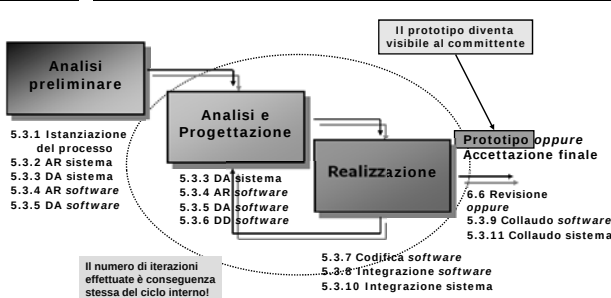
- ❑ **Analisi preliminare**
  - Per identificare i requisiti di massima
  - Per definire l'architettura di massima
  - Per pianificare i passi di analisi e realizzazione evolutiva
- ❑ **Analisi e realizzazione di evoluzione**
  - Per raffinamento ed estensione dell'analisi
  - Per progettazione, codifica, prove e integrazione
- ❑ **Rilascio di “prototipi”, poi accettazione finale**



Schema generale



Schema secondo ISO 12207



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**



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



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?

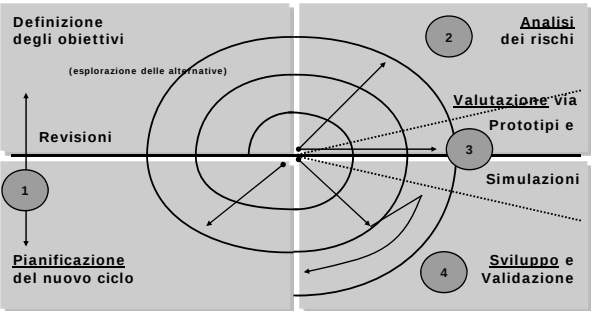


Fasi del modello a spirale

- ❑ 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



Schema generale



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 riutilizzo 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 software 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 soddisfa gli obiettivi



Il ciclo di vita del *software*

Modelli agili – 3

❑ Extreme Programming (XP)

<http://www.extremeprogramming.org>

Prima manifestazione concreta del movimento ‘agile’

Eccessiva enfasi sul SW come unico prodotto dello sviluppo

Coinvolge molto il committente ma in modo informale

- Non solo nell'analisi ma anche in progettazione e sviluppo (!)
- Processo incrementale e iterativo (a passo piccolo)

Richiede ambiente favorevole e grande esperienza delle figure chiave per essere attuato al meglio

Dipartimento di Informatica, Università di Pisa

31/36



Il ciclo di vita del *software*

Modelli agili – 4

❑ Scrum, Kanban, Scrumban

[http://www.scrumalliance.org/learn\\_about\\_scrum](http://www.scrumalliance.org/learn_about_scrum)

Principio del “caos organizzato”

A ciclo fortemente iterativo

Una lista (dinamica) di cose da fare

Product Backlog

Sprint Backlog

24 hours

2-4 weeks

Controllo rapido e frequente

Potentially Shippable Product Increment

Integrazione incrementale dei risultati e modifiche alla lista iniziale

Dipartimento di Informatica, Università di Pisa

32/36



Il ciclo di vita del *software*

Metodi agili – 5

❑ 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

Dipartimento di Informatica, Università di Pisa

33/36



Il ciclo di vita del *software*

Metodi agili – 6

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

Sviluppo

User Story

Task 1

Task ...

Task di Infrastruttura

Task Title

Stima tempo:

Responsabile:

Review svolta da:

Project Automation

Test Driven Development

Pair Programming

Processi definiti

QA Code Review

Continuous Integration and Deployment

Jenkins

Dipartimento di Informatica, Università di Pisa

34/36



Il ciclo di vita del *software*

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

Modello sequenziale

0 25 50 75 100

Specification Design Development Integration and testing

Modello iterativo

0 25 50 75 100

Specification Iterative development System testing

Modello “component-based”

0 25 50 75 100

Specification Development Integration and testing

Modello evolutivo

0 10 200 30 400

System development System evolution

Tratto da: Ian Sommerville, Software Engineering, 9<sup>th</sup> ed.

Dipartimento di Informatica, Università di Pisa

35/36



Il ciclo di vita del *software*

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](http://sunset.usc.edu/research/spiral_model)

❑ ISO/IEC TR 15271:1998, Information Technology – Guide for ISO/IEC 12207

Dipartimento di Informatica, Università di Pisa

36/36

IS 2011 - Ingegneria del Software

6