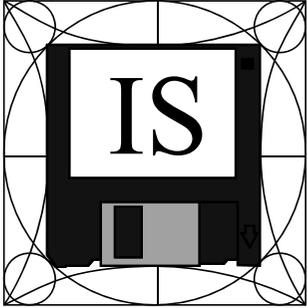




Il ciclo di vita del *software*

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 Pisa1/38



Il ciclo di vita del *software*

Il concetto di ciclo di vita – 2

- ❑ La durata temporale entro uno stato di ciclo di vita o in una transizione tra essi viene detta «fase»
- ❑ Il modello di ciclo di vita adottato pone vincoli su pianificazione e gestione del progetto
 - È indipendente da metodi e strumenti di sviluppo
 - Precede e non segue la loro selezione
- ❑ L'adozione di un modello richiede un sistema di qualità_[g] per garantire e misurare conformità_[g] e maturità_[g]

Dipartimento di Informatica, Università di Pisa3/38



Il ciclo di vita del *software*

Il concetto di ciclo di vita – 1

- ❑ **Concezione** → sviluppo → utilizzo → ritiro
 - Gli stati assunti da un prodotto SW
- ❑ **La transizione tra stati avviene tramite l'attuazione di attività specificate in processi di ciclo di vita**
 - Processi istanziati da modelli generali (p.es. ISO/IEC 12207)
 - Tramite attività con ruoli e responsabilità fissate
- ❑ **Per organizzare le attività di processo implicate**
 - Identifichiamo le dipendenze tra i loro ingressi e le loro uscite
 - Fissiamo il loro ordinamento nel tempo e i criteri di completamento e avanzamento

Dipartimento di Informatica, Università di Pisa2/38



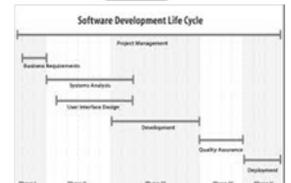
Il ciclo di vita del *software*

Modelli e fasi

Questi sono modelli ...



Queste sono fasi ...



Dipartimento di Informatica, Università di Pisa4/38



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

Dipartimento di Informatica, Università di Pisa

5/38



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

- ❑ **Fasi distinte e non sovrapposte nel tempo**

- ❑ **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

Dipartimento di Informatica, Università di Pisa

7/38



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”*
 - Centrato sull’idea di processi ripetibili

- ❑ **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*»)

Dipartimento di Informatica, Università di Pisa

6/38



Il ciclo di vita del *software*

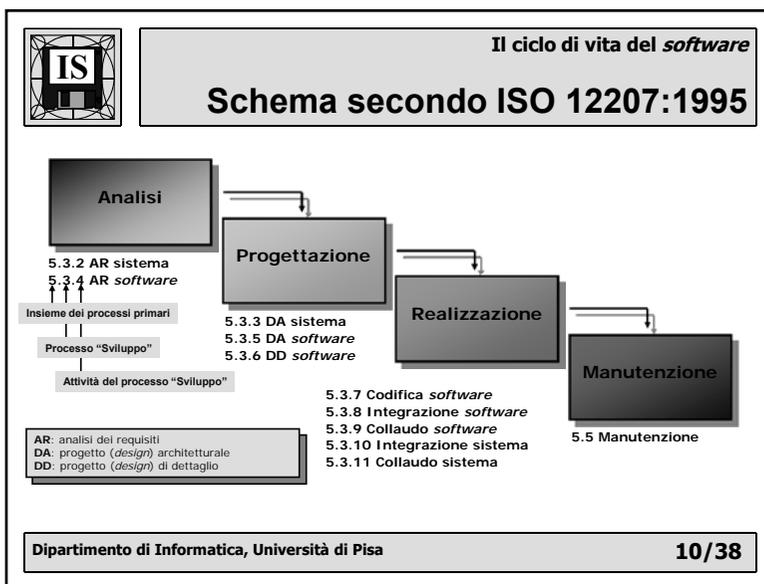
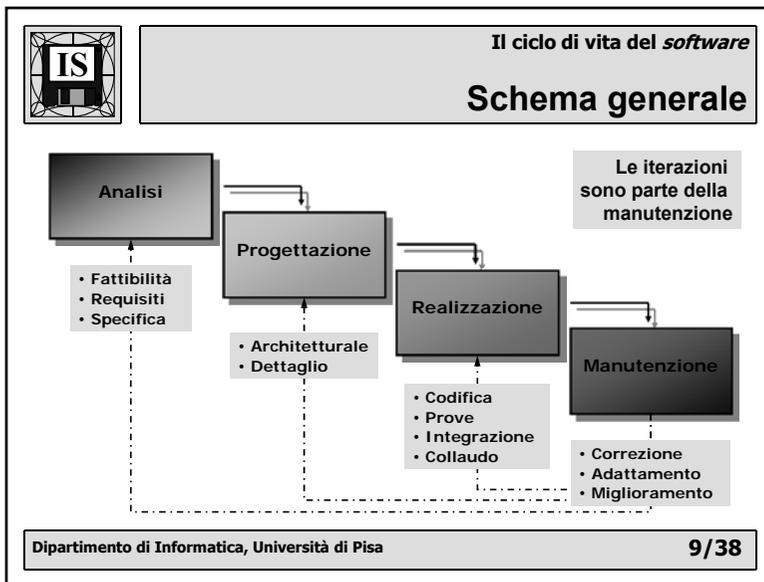
Modello sequenziale (a cascata) – 3

- ❑ **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

- ❑ **Le fasi sono durate temporali con dipendenze causali tra loro**
 - Entrare, uscire, stazionare in una fase comporta azioni specifiche
 - Realizzate come attività erogate dai processi coinvolti

Dipartimento di Informatica, Università di Pisa

8/38





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

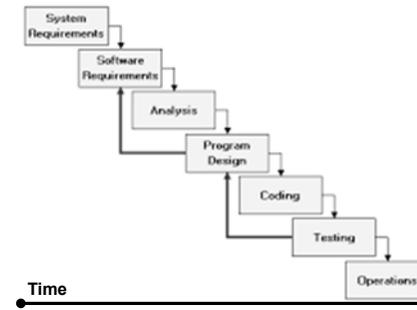
Dipartimento di Informatica, Università di Pisa

13/38



Il ciclo di vita del *software*

Rischi dei modelli iterativi



Ogni iterazione comporta un ritorno all'indietro nella direzione opposta all'avanzamento del tempo

Dipartimento di Informatica, Università di Pisa

15/38



Il ciclo di vita del *software*

Vantaggi dei modelli iterativi

- ❑ **Sono applicabili a qualunque modello di ciclo di vita**
 - Con opportuni vincoli
- ❑ **Consentono maggior capacità di adattamento**
 - Evoluzione di problemi, requisiti utente, soluzioni e tecnologie
- ❑ **Ma comportano il rischio di non convergenza**
- ❑ **Soluzione generale**
 - Decomporre la realizzazione del sistema
 - Identificare e trattare prima le componenti più critiche
 - Quelle più complesse oppure quelle i cui requisiti vanno maggiormente chiariti
 - Limitando superiormente il numero di iterazioni

Dipartimento di Informatica, Università di Pisa

14/38



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

16/38



Il ciclo di vita del *software*

Modello incrementale – 2

- ❑ **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

Dipartimento di Informatica, Università di Pisa

17/38



Il ciclo di vita del *software*

Schema secondo ISO 12207:1995

Dipartimento di Informatica, Università di Pisa

19/38



Il ciclo di vita del *software*

Schema generale

I cicli di incremento sono parte dello sviluppo

La validazione è anch'essa incrementale

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

Dipartimento di Informatica, Università di Pisa

18/38



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 più fasi di ciclo di vita**

Dipartimento di Informatica, Università di Pisa

20/38

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 una 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

21/38

Il ciclo di vita del *software*

Schema secondo ISO 12207:1995

Il prototipo diventa visibile al committente

Il numero di iterazioni effettuate è conseguenza stessa del ciclo interno!

Dipartimento di Informatica, Università di Pisa

23/38

Il ciclo di vita del *software*

Schema generale

Concurrent activities

Ogni fase ammette iterazioni multiple e e parallele

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

Dipartimento di Informatica, Università di Pisa

22/38

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**

Dipartimento di Informatica, Università di Pisa

24/38



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

Dipartimento di Informatica, Università di Pisa

25/38



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

Dipartimento di Informatica, Università di Pisa

27/38



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?

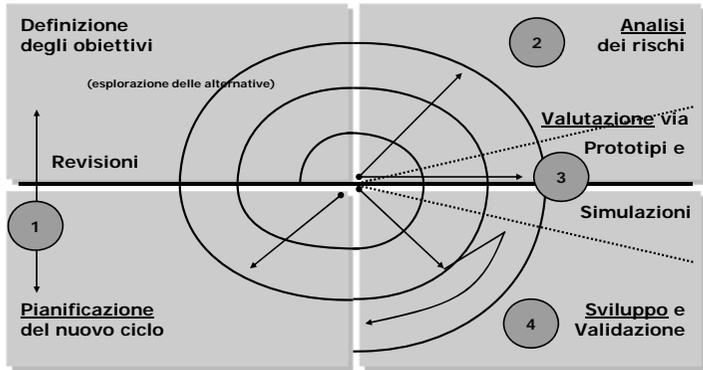
Dipartimento di Informatica, Università di Pisa

26/38



Il ciclo di vita del *software*

Schema generale



Dipartimento di Informatica, Università di Pisa

28/38



Il ciclo di vita del *software*

Modello a componenti



```
graph TD; A(Analisi dei Requisiti) --> B(Analisi delle Componenti); B --> C(Adattamento dei Requisiti); C --> D(Sviluppo e Integrazione); D --> E(Validazione di Sistema); E --> F(Progettazione con Riuso); F --> A;
```

- ❑ 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"

Dipartimento di Informatica, Università di Pisa29/38



Il ciclo di vita del *software*

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

Dipartimento di Informatica, Università di Pisa31/38



Il ciclo di vita del *software*

Metodi agili – 1

- ❑ Nascono alla fine del '90 come reazione alla eccessiva rigidità dei modelli allora in vigore
 - <http://agilemanifesto.org/>
- ❑ Si basano su 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 e non ingessata
 - 4) *Responding to change over following a plan*
 - La capacità di adattamento al cambiare delle situazioni è importante

Dipartimento di Informatica, Università di Pisa30/38



Il ciclo di vita del *software*

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

Dipartimento di Informatica, Università di Pisa32/38

Il ciclo di vita del software

Il ciclo di vita del *software*

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

Dipartimento di Informatica, Università di Pisa

33/38

Il ciclo di vita del *software*

Il ciclo di vita secondo SEMAT /1

Opportunity

Identified

Solution Needed

Value Established

Viable

Addressed

Benefit Accrued

Stakeholders

Recognized

Represented

Involved

In Agreement

Satisfied for Deployment

Satisfied in Use

www.ivarjacobson.com/semat

Sheet 1 of 2

Dipartimento di Informatica, Università di Pisa

35/38

Il ciclo di vita del *software*

Scrum – 2

- **Sprint Planning**
Pianificazione dello sprint
- **Sprint Review**
Controllo prodotti dello sprint

- **Daily Scrum**
Controllo giornaliero avanzamento
- **Sprint Retrospective**
Controllo qualità sullo sprint

Dipartimento di Informatica, Università di Pisa

34/38

Il ciclo di vita del *software*

Il ciclo di vita secondo SEMAT /2

Software System

Architecture Selected

Demonstrable

Useable

Ready

Operational

Retired

Team

Seeded

Formed

Collaborating

Performing

Adjourned

www.ivarjacobson.com/semat

Sheet 2 of 2

Dipartimento di Informatica, Università di Pisa

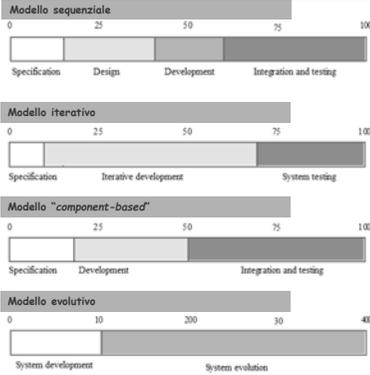
36/38



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



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

Dipartimento di Informatica, Università di Pisa

37/38



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
- ❑ ISO/IEC TR 15271:1998, Information Technology – Guide for ISO/IEC 12207
- ❑ Scrum: http://www.scrumalliance.org/learn_about_scrum

Dipartimento di Informatica, Università di Pisa

38/38