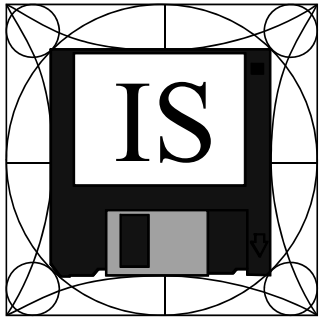
Il ciclo di vita del SW



Ingegneria del Software

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

Aggiornamenti : T. Vardanega (UniPD)

Il concetto di ciclo di vita – 1

- Concezione → sviluppo → utilizzo → ritiro
  - Gli stati (principali) assunti da un prodotto SW
  - Noi ci concentriamo sullo sviluppo e per questo parliamo modello/ciclo di sviluppo
- La transizione tra stati avviene tramite l'esecuzione di attività di processi di ciclo di vita
  - Processi istanziati da modelli generali (p.es. ISO/IEC 12207)
  - Le attività di processo prevedono specifici ruoli e responsabilità
- Per organizzare le attività dei processi implicati
  - Identifichiamo le dipendenze tra gli ingressi dell'uno e le uscite dell'altro
  - Fissiamo l'ordinamento nel tempo e i criteri di attivazione (pre-condizioni) e di completamento (post-condizioni)

Il concetto di ciclo di vita – 2

- Lo stazionamento in uno stato di ciclo di vita o in una transizione tra stati viene detta fase
  - "Fase" designa un segmento temporale con specifiche caratteristiche
- Per un progetto, aderire a un modello di ciclo di vita determina vincoli sulla sua pianificazione e gestione
  - La scelta del modello influenza la selezione dei metodi di lavoro e degli strumenti di supporto
- Associare un sistema di qualità al modello adottato (quindi ai processi corrispondenti) aiuta a perseguire conformità nel progetto e maturità nei processi
  - Conformità agli obiettivi del modello (nel rispetto dei vincoli)
  - Maturità = qualità misurata dei processi che lo attuano

Evoluzione dei modelli di sviluppo

- Dal "*Code-'n-Fix*" che è un non-modello
  - Insieme di attività senza organizzazione preordinata
  - Fonte di progetti caotici difficilmente gestibili
- Ai modelli organizzati
  - Cascata — rigide fasi sequenziali
  - Incrementale — realizzazione in più passi
  - Evolutivo — ripetute iterazioni interne
  - A componenti — orientato al riuso
  - Agile — altamente dinamico, fatto di brevi cicli iterativi e incrementali

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 poi a includere il SW
  - L'emissione e l'approvazione di documenti sono condizione necessaria per l'avvio della fase successiva

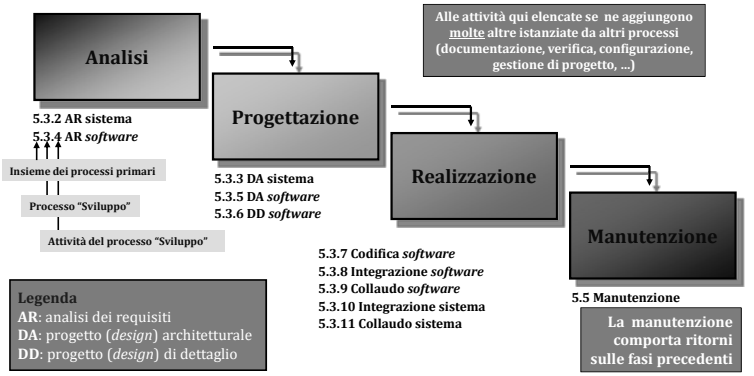
Modello sequenziale (a cascata) – 2

- Ogni stato (fase) è caratterizzato da pre-condizioni di ingresso e post-condizioni di uscita
  - Il loro soddisfacimento è dimostrato primariamente tramite prodotti documentali e infine tramite esecuzione del SW
- Le fasi sono distinte e non sovrapposte nel tempo
- Modello 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

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 prodotti
- Le fasi sono durate temporali con dipendenze causali tra loro
  - Entrare, uscire, stazionare in una fase comporta lo svolgimento di azioni specifiche
    - Realizzate come attività erogate da corrispondenti processi

Schema secondo ISO 12207:1995



Critica del modello sequenziale

- Difetto principale: eccessiva rigidità
  - Stretta sequenzialità tra fasi (nessun parallelismo e nessun ritorno)
  - Non ammette modifiche nei requisiti in corso d'opera
  - Esprime una visione burocratica e poco realistica del progetto
- Correttivo 1: con prototipazione
  - Prototipi di tipo "usa e getta"
    - Per capire meglio i requisiti o le soluzioni
    - Strettamente all'interno di singole fasi
- Correttivo 2: cascata con ritorni
  - Ogni ciclo di ritorno raggruppa sotto-sequenze di fasi

Ritorni: iterazione o incremento?

- Non sempre gli *stakeholder* hanno ben chiaro dall'inizio ogni aspetto del sistema richiesto
  - In tal caso bisogna decidere cosa sia meglio fare per aiutarli
  - Per problemi particolarmente complessi conviene prevedere iterazioni
  - Le iterazioni possono essere distruttive e annullare lavoro precedente
- Spesso non conviene posticipare troppo l'integrazione di tutte le parti del sistema (*big-bang integration*)
  - Meglio l'integrazione continua di piccole parti, che è un tipico procedimento incrementale
- Iterazione e incremento coincidono quando la sostituzione raffina ma non ha impatto sul resto

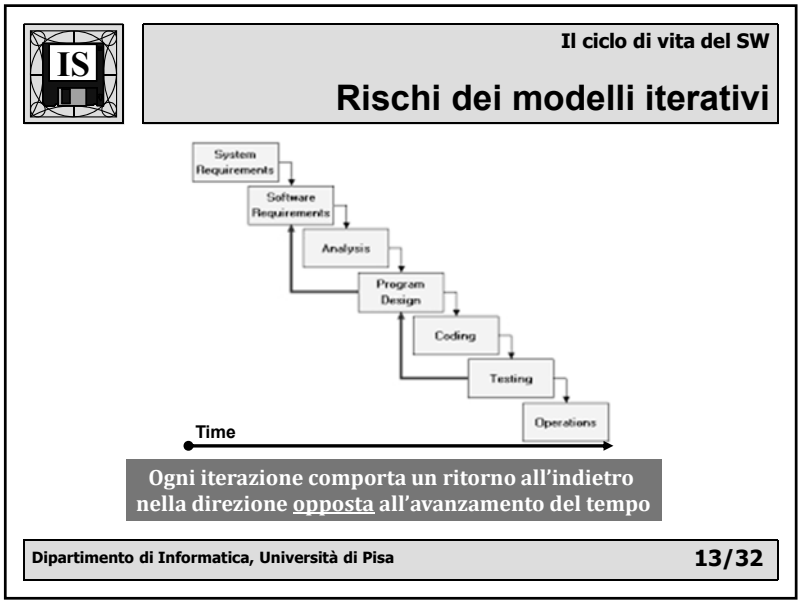
Vantaggi dei modelli incrementali

- Possono produrre valore a ogni incremento
  - Un insieme di funzionalità diventa presto disponibile
  - I primi incrementi possono essere frutto di prototipazione, aiutando 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 dalla caduta nell'iterazione
- Le funzionalità principali sono sviluppate nei primi incrementi
  - Quindi sono più frequentemente verificate e per questo diventano via via più stabili

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 parti più critiche
    - Quelle più complesse oppure quelle i cui requisiti vanno maggiormente chiariti
  - Limitando superiormente il numero di iterazioni



Il ciclo di vita del SW

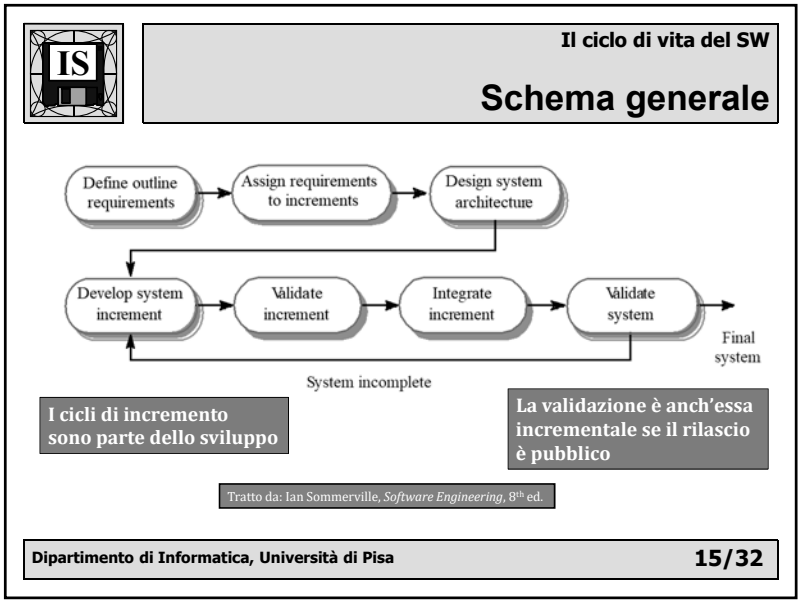
Rischi dei modelli iterativi

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

Il ciclo di vita del SW

Modello incrementale – 1

- Prevede rilasci multipli e successivi
  - Ciascuno realizza un incremento di funzionalità
- I requisiti sono classificati e trattati in base alla loro importanza strategica
  - I primi incrementi puntano a soddisfare i requisiti più importanti sul piano strategico
  - Così i requisiti importanti diventano presto chiari e stabili, quindi più facilmente soddisfacibili
  - Quelli meno importanti hanno invece più tempo per stabilizzarsi e armonizzarsi con lo stato del sistema



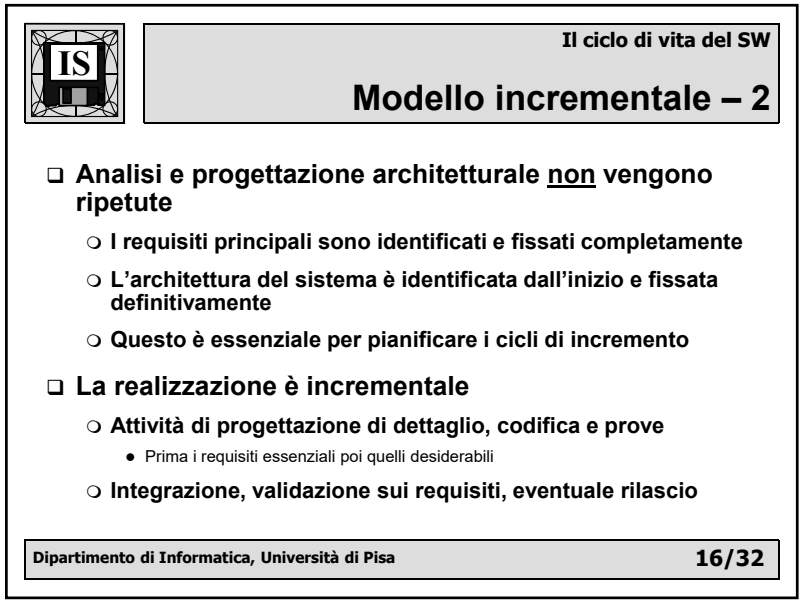
Il ciclo di vita del SW

Schema generale

I cicli di incremento sono parte dello sviluppo

La validazione è anch'essa incrementale se il rilascio è pubblico

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

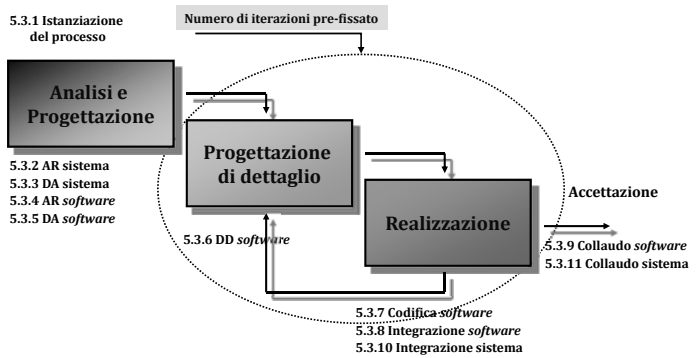


Il ciclo di vita del SW

Modello incrementale – 2

- Analisi e progettazione architetturale non vengono ripetute
  - I requisiti principali sono identificati e fissati completamente
  - L'architettura del sistema è identificata dall'inizio e fissata definitivamente
  - Questo è 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, validazione sui requisiti, eventuale rilascio

Schema secondo ISO 12207:1995



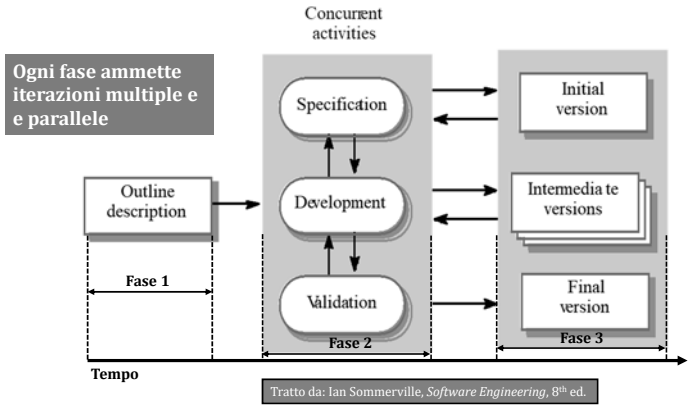
Modello evolutivo – 1

- Aiuta a rispondere a bisogni non inizialmente preventivabili
- Può richiedere il rilascio e il mantenimento di più versioni esterne attive in parallelo
- Comporta il riattraversamento di più stati di ciclo di vita

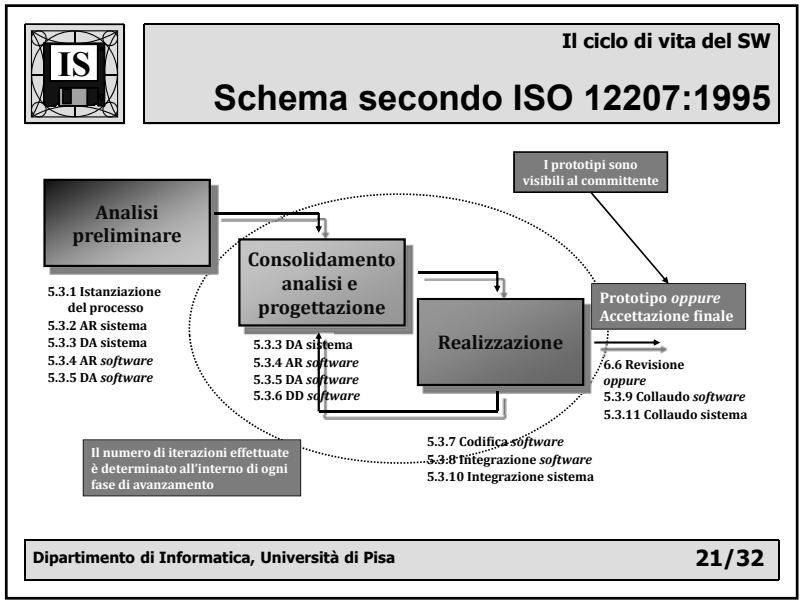
Modello evolutivo – 2

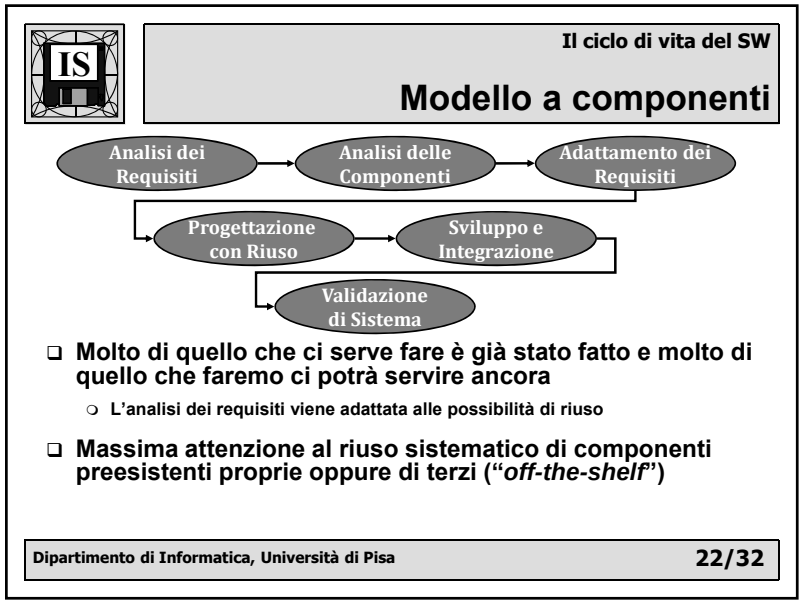
- Analisi preliminare
  - Identificare i requisiti di massima
  - Definire l'architettura di massima
  - Pianificare i passi di analisi e realizzazione evolutiva
- Analisi e realizzazione di una singola evoluzione
  - Per raffinamento ed estensione dell'analisi iniziale
  - Per progettazione, codifica, prove, integrazione
- Rilascio di versioni sempre più complete
  - Fintanto che necessario

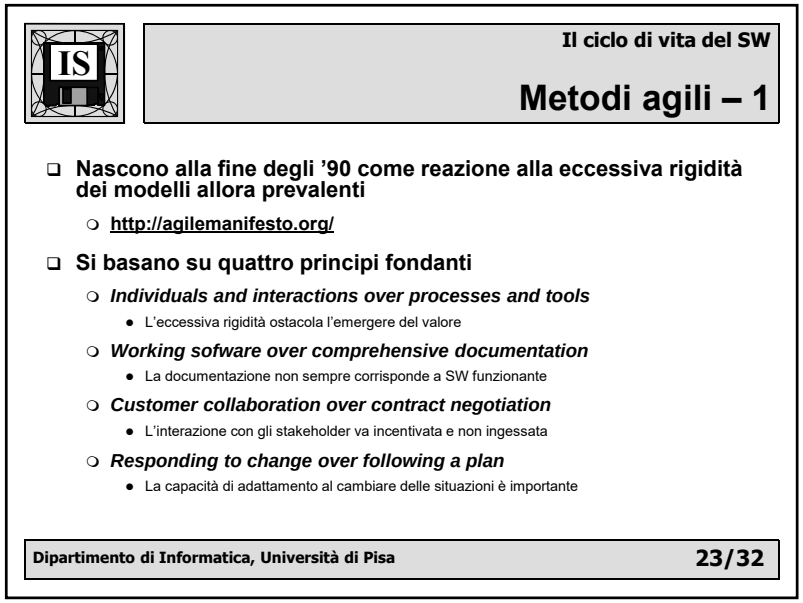
Schema generale

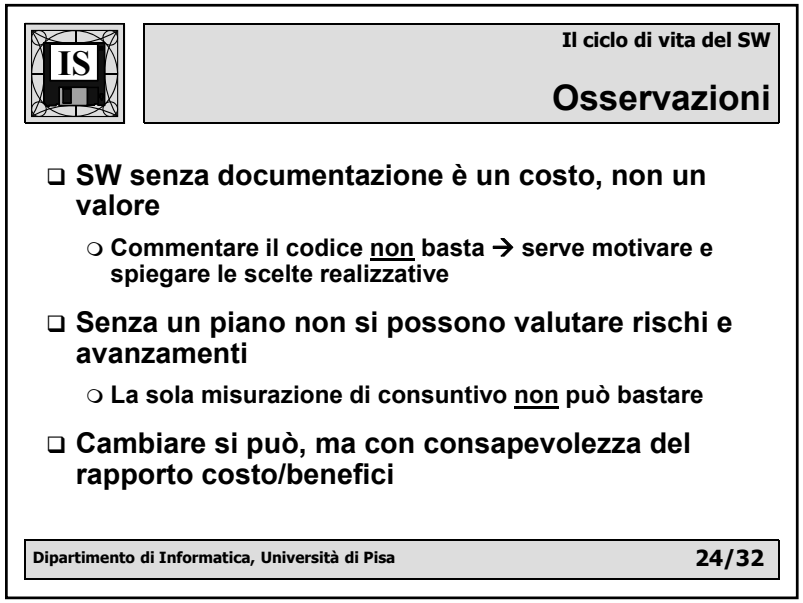


Tratto da: Ian Sommerville, *Software Engineering*, 8th ed.


IS
Il ciclo di vita del SW
Schema secondo ISO 12207:1995
I prototipi sono visibili al committente
Analisi preliminare
Consolidamento analisi e progettazione
Realizzazione
Prototipo *oppure* Accettazione finale
5.3.1 Istanziazione del processo
5.3.2 AR sistema
5.3.3 DA sistema
5.3.4 AR *software*
5.3.5 DA *software*
5.3.3 DA sistema
5.3.4 AR *software*
5.3.5 DA *software*
5.3.6 DD *software*
6.6 Revisione *oppure*
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
Il numero di iterazioni effettuate è determinato all'interno di ogni fase di avanzamento
Dipartimento di Informatica, Università di Pisa
21/32


IS
Il ciclo di vita del SW
Modello a componenti
Analisi dei Requisiti
Analisi delle Componenti
Adattamento dei Requisiti
Progettazione con Riuso
Sviluppo e Integrazione
Validazione di Sistema
Molto di quello che ci serve fare è già stato fatto e molto di quello che faremo ci potrà servire ancora
L'analisi dei requisiti viene adattata alle possibilità di riuso
Massima attenzione al riuso sistematico di componenti preesistenti proprie oppure di terzi ("*off-the-shelf*")
Dipartimento di Informatica, Università di Pisa
22/32


IS
Il ciclo di vita del SW
Metodi agili – 1
Nascono alla fine degli '90 come reazione alla eccessiva rigidità dei modelli allora prevalenti
http://agilemanifesto.org/
Si basano su quattro principi fondanti
Individuals and interactions over processes and tools
L'eccessiva rigidità ostacola l'emergere del valore
Working sofware over comprehensive documentation
La documentazione non sempre corrisponde a SW funzionante
Customer collaboration over contract negotiation
L'interazione con gli stakeholder va incentivata e non ingessata
Responding to change over following a plan
La capacità di adattamento al cambiare delle situazioni è importante
Dipartimento di Informatica, Università di Pisa
23/32


IS
Il ciclo di vita del SW
Osservazioni
SW senza documentazione è un costo, non un valore
Commentare il codice non basta → serve motivare e spiegare le scelte realizzative
Senza un piano non si possono valutare rischi e avanzamenti
La sola misurazione di consuntivo non può bastare
Cambiare si può, ma con consapevolezza del rapporto costo/benefici
Dipartimento di Informatica, Università di Pisa
24/32

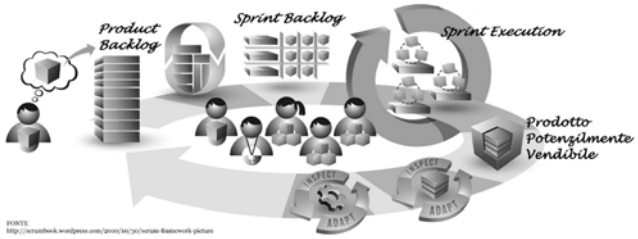
Metodi agili – 2

- L'idea base è il concetto di "*user story*"
  - Una funzionalità significativa che l'utente vuole realizzare con il SW richiesto
- Ogni "*user story*" è definita da
  - Un documento di descrizione del problema individuato
  - La minuta delle conversazioni con gli *stakeholder* effettuate per discutere il problema e comprenderlo insieme
  - La strategia da usare per confermare che il SW realizzato soddisfi gli obiettivi del problema

Metodi agili – 3

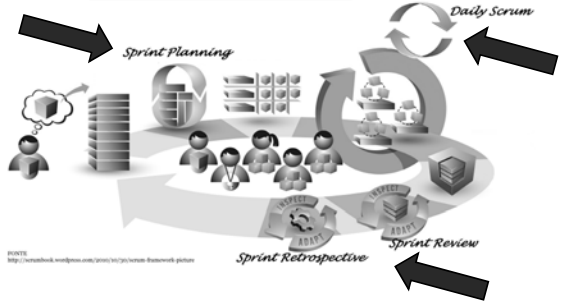
- Migliori assunti base
  - Suddividere il lavoro in piccoli incrementi a valore aggiunto che possono anche essere sviluppati indipendentemente
  - Sviluppare gli incrementi in una sequenza continua dall'analisi all'integrazione
- Obiettivi strategici
  - Poter costantemente dimostrare al cliente quanto è stato fatto
  - Verificare l'avanzamento tramite progresso reale
  - Dare agli sviluppatori la soddisfazione del risultato
  - Dimostrare che l'intero prodotto SW è ben integrato e verificato
- Esempi
  - Scrum (organizzazione dietro caos apparente), 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

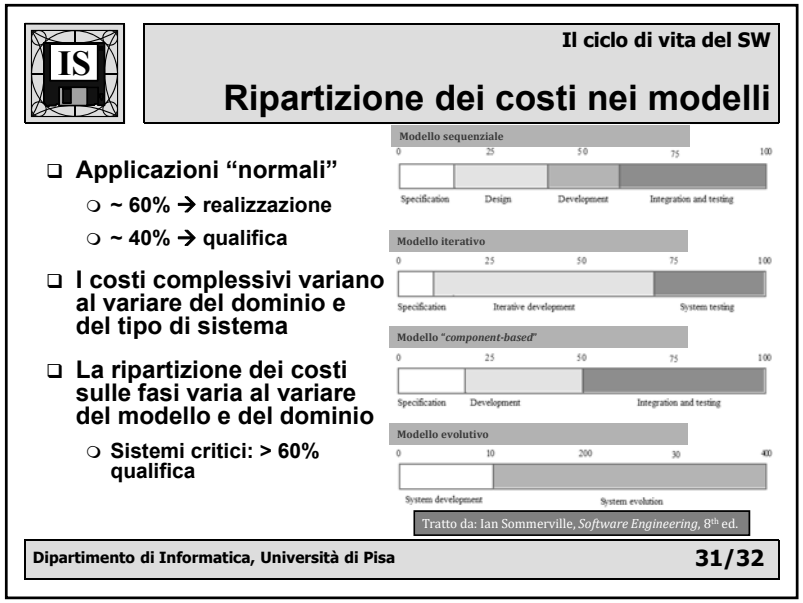
Scrum – 2

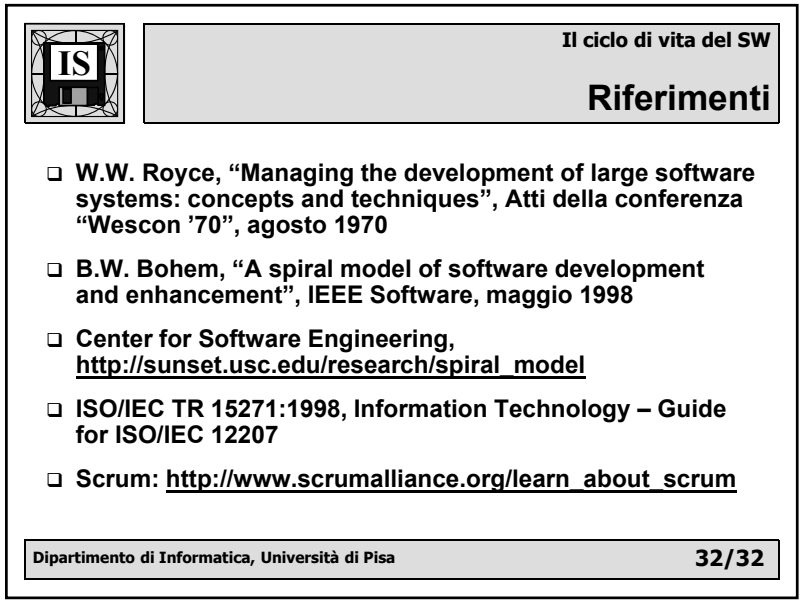


- **Sprint Planning**
  Pianificazione dello sprint
- **Sprint Review**
  Controllo prodotti dello sprint
- **Daily Scrum**
  Controllo giornaliero avanzamento
- **Sprint Retrospective**
  Controllo qualità sullo sprint


IS
Il ciclo di vita del SW
Il ciclo di vita secondo SEMAT /1
IVAR JACOBSON INTERNATIONAL
Software Engineering Kernel
Stakeholders
Recognized
Represented
Involved
In Agreement
Satisfied for Deployment
Satisfied in Use
Opportunity
Identified
Solution Needed
Value Established
Viable
Addressed
Benefit Accrued
Requirements
Conceived
Bounded
Coherent
Acceptable
Addressed
Fulfilled
www.ivarjacobson.com/semat
Sheet 1 of 2
Dipartimento di Informatica, Università di Pisa
29/32


IS
Il ciclo di vita del SW
Il ciclo di vita secondo SEMAT /2
IVAR JACOBSON INTERNATIONAL
Software System
Architecture Selected
Demonstrable
Useable
Ready
Operational
Retired
Team
Seeded
Formed
Collaborating
Performing
Adjourned
Work
Initiated
Prepared
Started
Under Control
Concluded
Closed
Way-of-Working
Principles Established
Foundation Established
In Use
In Place
Working Well
Retired
www.ivarjacobson.com/semat
Sheet 2 of 2
Dipartimento di Informatica, Università di Pisa
30/32


IS
Il ciclo di vita del SW
Ripartizione dei costi nei modelli
Applicazioni "normali"
~ 60% → realizzazione
~ 40% → 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, 8th ed.
Dipartimento di Informatica, Università di Pisa
31/32


IS
Il ciclo di vita del SW
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
32/32