Processo – parte III

Leggere Sez. 7.4 Ghezzi et al.

università di ferrara
DA SEICENTO ANNI GUARDIAMO AVANTI.

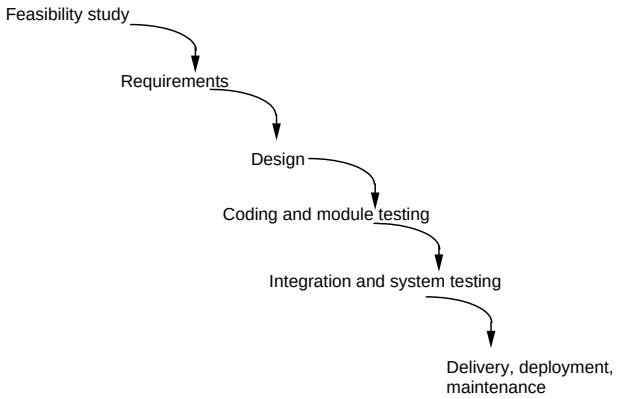
Modello Code and fix

- Modello iniziale
- Iterazione di due passi
  - scrittura del codice
  - correzione degli errori
- Problemi:
  - dopo una serie di cambiamenti, la struttura del codice diventa disorganizzata
  - processo non formulato precisamente né controllato attentamente
  - adatto solo ad applicazioni semplici, in cui l'utente e il programmatore coincidono

Modello a cascata (waterfall)

- Inventato negli anni Cinquanta per sistemi di difesa aerea
- Diffuso a partire dagli anni Settanta
- Ancora oggi modello di riferimento per molti testi e in molte organizzazioni
- Organizza le attività in modo sequenziale
- Varianti

Modello a cascata



Modello a cascata

- Ogni attività deve essere completata prima che si possa passare alla successiva.
- L'output di una attività è l'input della successiva
- Le attività si possono suddividere in sotto-attività che possono essere eseguite parallelamente

Studio di fattibilità

- Le modalità e gli scopi dipendono dal tipo di relazione fra sviluppatore e cliente:
  - Software house che sviluppa un'applicazione su commissione
  - Gruppo di sviluppo software che sviluppa un'applicazione per l'organizzazione cui appartiene
  - Software house che sviluppa un'applicazione da immettere sul mercato.

Modelli a cascata

- Spesso, le organizzazioni che li adottano stabiliscono standard per gli output delle varie fasi (deliverable).
- Per ogni fase possono essere prescritti dei metodi di realizzazione.
- I metodi compongono una metodologia aziendale.

Esempio: MIL-STD-2167

- Standard per lo sviluppo di software in sistemi di difesa (sostituito nel 1994)
- Parte di un più generale ciclo di vita dei sistemi consistente in:
  1. Esplorazione dei concetti
  2. Dimostrazione e convalida
  3. Sviluppo
  4. Produzione, distribuzione e installazione

MIL-STD-2167

- Il ciclo di sviluppo software è suddiviso in
  1. Analisi dei requisiti
  2. Progettazione preliminare
  3. Progettazione dettagliata
  4. Produzione del codice e test dei moduli
  5. Integrazione dei componenti e test
  6. Test del componente di configurazione CSCI (computer software configuration item)

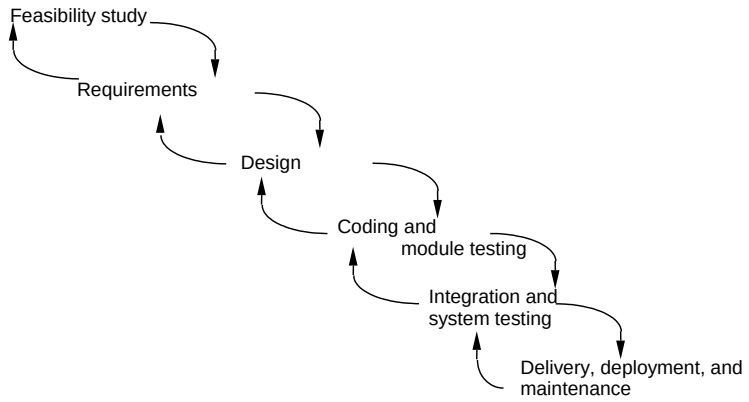
MIL-STD-2167

- Per ogni fase, lo standard definisce dettagliatamente
  - Le attività da svolgere
  - I prodotti da rilasciare
  - Le analisi necessarie per il monitoraggio
  - La struttura del software e della documentazione associata

Valutazione critica

- Lineare, rigido, monolitico
- Pregi
  - Impone disciplina, pianificazione, gestione
  - L'implementazione viene dopo una comprensione profonda dei requisiti
- Difetti
  - Non si ha feedback
  - Non permette parallelismo fra le fasi
  - Data di consegna unica
  - I risultati di una fase sono congelati prima di passare alla fase successiva

Modello a cascata con feedback



Problemi dei modelli a cascata

- Difficile stimare le risorse necessarie (tempo, budget) con informazioni limitate
- Il documento di specifica dei requisiti è un vincolo contrattuale, ma per l'utente è difficile da convalidare
- Spesso alcuni requisiti sono chiarificati solo dopo il rilascio (feedback dagli utenti)

Problemi dei modelli a cascata

- Comportano la produzione di documenti secondo standard rigidi (processo document driven)
- La rigidità del processo porta ad alti costi per la manutenzione
- L'evoluzione non è né anticipata né pianificata
- Spesso manutenzione senza aggiornare i documenti di specifica dei requisiti e del progetto

Do it twice

- Brooks 1995
- La prima versione di un prodotto è un prototipo che ha lo scopo di:
  - Valutare la fattibilità del prodotto
  - Convalidarne i requisiti
- Il prototipo va poi scartato
- Chiariti i requisiti, si sviluppa la seconda versione secondo il modello a cascata.

Modelli evolutivi

- Boehm 1988
- Modello le cui fasi consistono in versioni incrementali di un prodotto software funzionante, secondo una direzione evolutiva determinata dall'esperienza.
- Le versioni incrementali possono essere rilasciate al fine di raccogliere feedback (rilascio incrementale, incremental delivery).

Rilascio incrementale

- Rilascio al cliente di unità funzionali auto-contenute (cioè funzionanti, utili e documentate)
- Ciclo (Gilb, 1988)
  1. Rilascio
  2. Misura del valore aggiunto per il cliente secondo i criteri di valutazione
  3. Aggiustamento di requisiti e progetto

Implementazione incrementale

- Modello a cascata fino alla progettazione
- Poi una serie incrementale di implementazione, test, integrazione e rilascio
- Le varie versioni differiscono solo per l'implementazione.
- Potrebbero però essere necessari correttivi anche ai requisiti e al progetto

Sviluppo e rilascio incrementale

- Anche l'analisi dei requisiti e la progettazione sono svolte in passi incrementali.
- Sequenza di applicazioni del modello a cascata, dove ogni iterazione si basa sul feedback ricevuto per la precedente.
- Generalizzazione del "do it twice"
- Il prodotto è in continua evoluzione
- Rilasciati prototipi che possono contenere stub per realizzare alcune funzioni
- Strumenti (linguaggi) di prototipazione rapida

Vantaggi

- Consegna più rapida
  - Ogni passo incrementale fornisce al cliente le funzionalità a priorità maggiore.
- Maggior coinvolgimento del cliente
  - Il cliente è coinvolto nel processo
  - Convalida dei requisiti più efficace
  - Legame fra il cliente e il sistema

Svantaggi

- Gestione
  - Il modello tende a trascurare la documentazione del processo: difficile valutare i progressi
- Problemi contrattuali
  - La specifica iniziale è soggetta a modifiche, quindi non può avere valore contrattuale
- Manutenzione
  - Le continue modifiche tendono a corrompere il software e a rendere più costose le modifiche successive

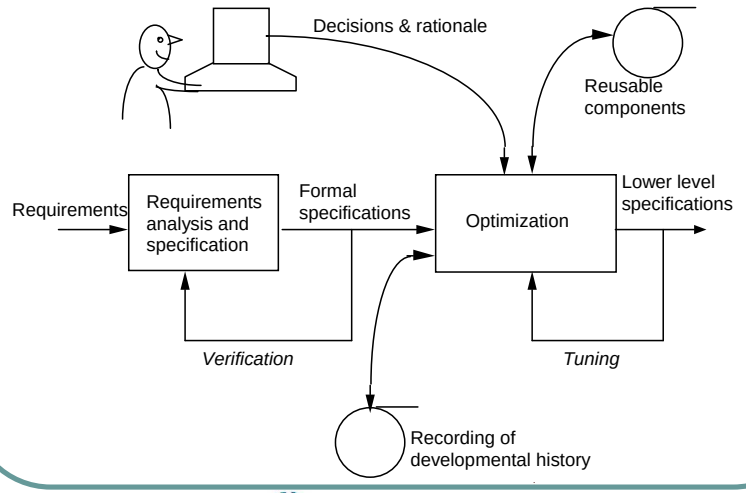
Modello trasformazionale

- Basato su specifiche formali
- Sviluppo come sequenza di passi che gradualmente trasforma una specifica in un'implementazione.
- Parte formalizzando i requisiti
- Ogni passo successivo è meno astratto e più dettagliato
- Se una specifica (cioè un passo del processo) è eseguibile, può essere utilizzata come prototipo

Modello trasformazionale

- Una volta che un prototipo è soddisfacente, è necessario ottimizzarlo per rispettare i requisiti non funzionali (in particolare quelli riguardanti le prestazioni).
- Supporto dell'ambiente di sviluppo:
  - Convalida dei requisiti
  - Gestione dei componenti
  - Ottimizzazione
  - Registrazione della storia dello sviluppo

Modello trasformazionale



Valutazione

- Supporta il cambiamento grazie alla tracciabilità dello sviluppo
- Le trasformazioni non sono, in generale, automatizzabili, ma operando su entità formali obbligano il progettista a procedere rigorosamente.
- Prove costruttive di correttezza: si genera il programma che rispetta la relazione fra precondizione e postcondizione
- Attualmente usato solo in ambiti di ricerca.

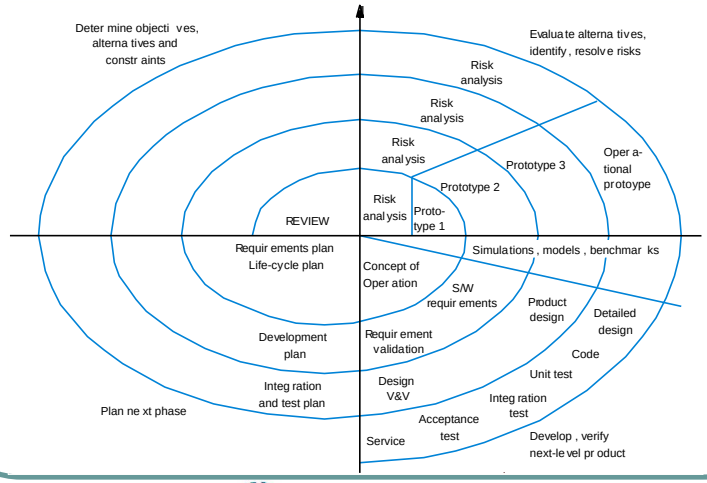
Modello a spirale

- Obiettivo: riduzione e gestione dei rischi in sistemi non ben compresi
- Processo iterativo composto da quattro fasi (una per settore) ripetute per ogni attività necessaria (fattibilità, analisi dei requisiti, etc.)
- Al termine di ogni iterazione, possibile rilascio di un prototipo
- Raggio della spirale: costo accumulato
- Angolo: avanzamento dell'iterazione.

Settori del modello a spirale

- Il modello non presuppone una organizzazione predefinita delle attività: la scelta è dettata dalla minimizzazione dei rischi (ad esempio tramite prototipazione o simulazione)
- I settori sono:
  - Determinazione degli obiettivi
  - Analisi e riduzione dei rischi
  - Sviluppo e convalida: secondo uno dei modelli generici
  - Pianificazione: revisione del progetto e pianificazione dell'iterazione successiva

Modello a spirale



Confronto fra modelli

- Code & fix: guidato dall'ispirazione del momento
- A cascata: guidato dalla documentazione
- Evolutivo: guidato dagli incrementi
- Trasformazionale: guidato da specifiche
- A spirale: guidato dai rischi

Confronti quantitativi

- Non sufficienti per una scelta a priori.
- Confronto fra approccio a cascata ed evolutivo (Boehm):
  - approccio a cascata permette un miglior controllo del prodotto e dei rischi
  - approccio evolutivo permette una gestione migliore delle interfacce utente
  - approccio evolutivo comporta una riduzione del 40% del numero di istruzioni e del tempo di sviluppo
  - approccio a cascata presenta meno problemi di debug e integrazione

Confronto fra modelli

- La flessibilità riduce i rischi
  - Incomprensioni sui requisiti
  - Time to market eccessivo
- Flessibilità non significa improvvisazione
- L'approccio a cascata può essere utilizzato a posteriori per la documentazione del processo (Parnas e Clements, 1986)