Università degli Studi di Napoli "Federico II"

Ingegneria del Software

a.a. 2012/13

Il processo software –
Il Modello di Processo a Cascata

Obiettivi della lezione

- Comprendere il concetto di Processo Software
- Comprendere il concetto di Modello di Processo Software
- Comprendere il Modello di Processo a Cascata

Processo

- "*Un processo è un particolare metodo per fare qualcosa costituito da una sequenza di passi che coinvolgono attività, vincoli e risorse*" (Pfleeger)
- "*Processo: una particolare metodologia operativa che nella tecnica definisce le singole operazioni fondamentali per ottenere un prodotto industriale*" (Zingarelli)
- "*Processo software: un metodo per sviluppare del software*" (Sommerville)

Processo/Ciclo di vita del software

- Insieme organizzato di attività che sovrintendono alla costruzione del software da parte del team di sviluppo utilizzando metodi, tecniche, metodologie e strumenti.
- È suddiviso in fasi, che vanno dalla nascita fino alla dismissione (o morte) del software.
  - Per un parallelo con la biologia, è noto come ***ciclo di vita del software***

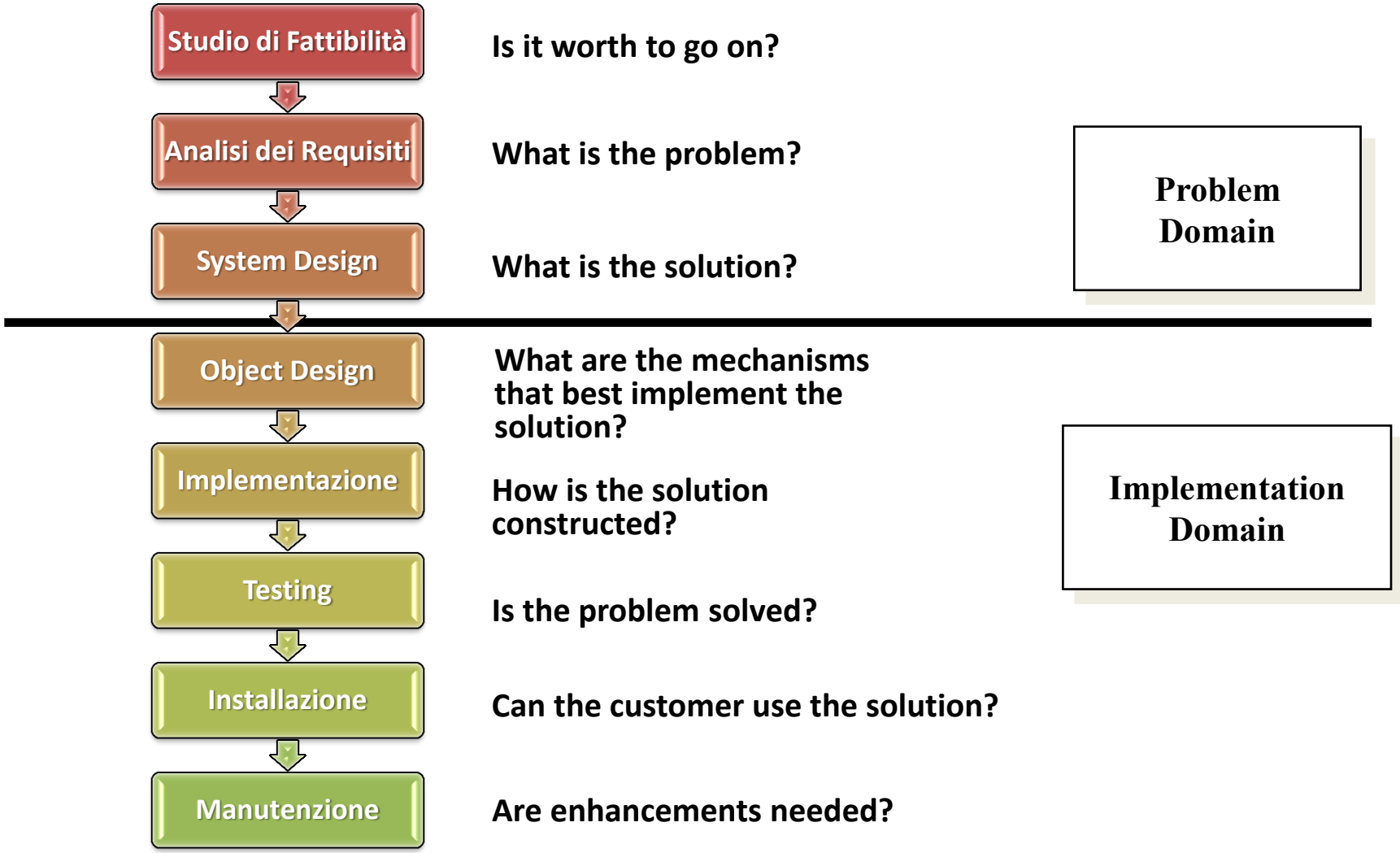
– Molti autori usano i termini ***processo*** *[di sviluppo del] software* e ***ciclo di vita*** *del software* come sinonimi.

Processo software: Standard IEEE 610.12-1990

- *“Sofware development process: The process by which user needs are translated into a software product. The process involves translating user needs into software requirements, transforming the software requirements into design, implementing the design in code, testing the code, and sometimes, installing and checking out the software for operational use.*
  - *Note: These activities may overlap or be performed iteratively.*

Macropassi fondamentali del processo software

Attività richieste nel processo di sviluppo software



Studio di fattibilità

- Valutazione preliminare di costi e benefici
- Varia a seconda della relazione committente/produttore
- Obiettivo
  - Stabilire se avviare il progetto, individuare le possibili opzioni e le scelte più adeguate, valutare le risorse umane e finanziarie necessarie
  - Si conclude con una *offerta* al Cliente
- Output: documento di fattibilità
  - definizione preliminare del problema
  - scenari - strategie alternative di soluzione
  - costi, tempi, modalità di sviluppo per ogni alternativa

Analisi dei requisiti

- Analisi completa dei bisogni dell'utente e dominio del problema
- Coinvolgimento di committente e ingegneri del SW
- Obiettivo
  - Descrivere le caratteristiche di qualità che l'applicazione deve soddisfare

- Output:
  - documento di specifica dei requisiti

Progettazione
(o System/Object Design)

- Definizione di una struttura opportuna per il SW
- Scomposizione del sistema in componenti e moduli
  - allocazione delle funzionalità ai vari moduli
  - definizione delle relazioni fra i moduli
- Distinzione fra:
  - System Design: struttura modulare complessiva (componenti)
  - Object Design: dettagli interni a ciascuna componente
- Obiettivo

- Output: documento di specifica di progetto
  - possibile l'uso di linguaggi per la progettazione (UML)

Fasi basse del processo

- **Implementazione**: ogni modulo viene codificato nel linguaggio scelto e testato in isolamento
- **Testing**
  - Composizione dei moduli nel sistema globale
  - Verifica del corretto funzionamento del sistema
  - Validazione che il sistema faccia ciò che vuole il cliente
- **Installazione**: distribuzione e gestione del software presso l'utenza
- **Manutenzione**: evoluzione del SW. Segue le esigenze dell'utenza. Comporta ulteriore sviluppo per cui racchiude in sé nuove iterazioni di tutte le precedenti fasi

Problemi nel processo di sviluppo del software

- E’ qualcosa di altamente intellettuale e creativo, basato su giudizi delle persone
  - Non è possibile (almeno con i sistemi attuali) automatizzarlo
- I requisiti sono complessi e ambigui
  - Il cliente non sa bene, dall’inizio, cosa deve fare il software
- I requisiti sono variabili
  - Cambiamenti tecnologici, organizzativi, etc…
- Modifiche frequenti sono difficili da gestire

I modelli di processo

Cosa sono i Modelli di Processo

- Come organizziamo la sequenza di passi del processo software, per massimizzare alcune carattistiche?
- Il processo di sviluppo deve essere **modellato** esplicitamente, per poter essere gestito e monitorato
- I modelli di processo software sono descrizioni precise e formalizzate delle attività, delle trasformazioni, dei deliverables e degli eventi per realizzare e/o ottenere l'evoluzione del software
- Obiettivo:
  - introdurre stabilitá, controllo e organizzazione in una attivitá tendenzialmente caotica, massimizzando alcune delle caratteristiche del processo

Modelli di processo: caratteristiche (1)

- Una strutturazione dell'organizzazione del lavoro nelle fabbriche del software in:
  - fasi della produzione,
  - tipi di attività,
  - collegamento ed interfacciamento,
  - controllo e misura,
- ma anche linee guida per: organizzare, pianificare, dimensionare personale, assegnare budget, schedulare e gestire, …

Modelli di processo: caratteristiche (2)

- definire e prescrivere prodotti e documenti da rilasciare al committente (**deliverables**)
- determinare e classificare metodi e strumenti più adatti a supportare le attività previste
- framework per analizzare, stimare, migliorare ...

Diverse tipologie di Modelli di processo

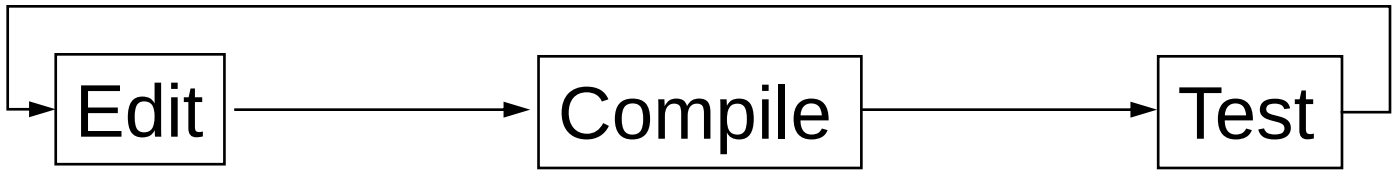
- Esistono vari Modelli di processo, nati (scoperti?) negli ultimi 30 anni
  - Waterfall (cascata)
  - Evolutivo
  - Trasformazionale
  - Basato sul riuso
  - Spiral model
  - eXtreme Programming

- Molti aspetti influenzano la definizione del modello
  - specificità dell'organizzazione produttrice
  - know-how
  - area applicativa e particolare progetto
  - strumenti di supporto
  - diversi ruoli produttore/ committente

Come valutare un modello di processo?

- Esattamente come esistono dei parametri di qualità per un software, esistono dei parametri per valutare la "bontà" di un modello con cui sviluppare il software:
  - **Comprensibilità**: Quanto è facile apprendere il processo?
  - **Visibilità** : Quanto facilmente il processo fa capire a che punto dello sviluppo si è giunti?
  - **Supportabilità** (CASE tools): Esistono tool che supportano il processo? Quali e quante fasi sono supportate? A che livello?
  - **Accettabilità**: Le persone coinvolte nel processo manifestano il loro consenso?
  - **Affidabilità**: Il processo facilità l'individuazione di errori? Il software prodotto è di alta qualità?
  - **Robustezza**: Quanto è robusto il processo nel gestire cambiamenti in corso d'opera?
  - **Mantenibilità**: Il processo è in grado di adattarsi ai cambiamenti nell'organizzazione che lo usa?

Il più semplice (e peggiore) modello di processo



- Molto veloce, feedback rapido
- Molti strumenti disponibili
- Specializzato per codifica
- Non incoraggia la documentazione
- Non scala (in the large, in the many)
- Ingestibile durante manutenzione

Valutiamo l’Edit-Compile-Test

- **Comprensibilità**:
- **Visibilità** :
- **Supportabilità**:
- **Accettabilità**:
- **Affidabilità**:
- **Robustezza**:
- **Mantenibilità**:
- **Rapidità**:

Il modello a cascata

L’approccio Tayloristico

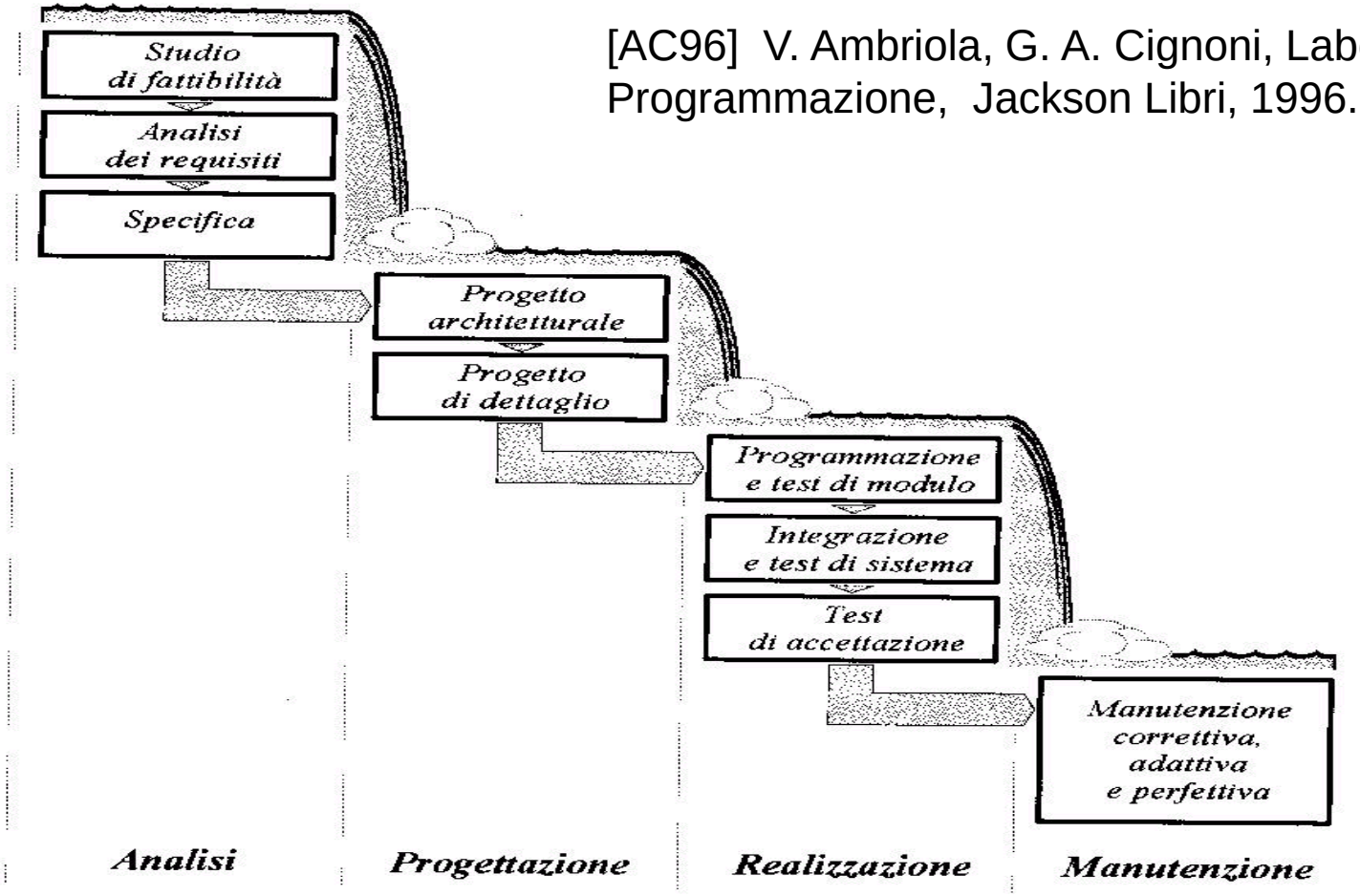
- Frederick Taylor – “The Principles of Scientific Management” (1911)
  - Gestione gerarchica
  - Passi fissi, non fluidi
  - Separare nettamente le postazioni di lavoro
  - Il lavoro, una volta suddiviso, diventa specializzato in un singolo task
  - Processo orientato al prodotto, non al cliente
    - “Any customer can have a car painted any color that he wants so long as it is black” - *Henry Ford*
  - Processo ripetibile

- E’ stato per decenni il fondamento dell’industria mondiale

Modelli a Cascata (Waterfall) - ’70

- Applicazione dell’approccio Tayloristico all’informatica
- Popolare negli anni ’70
  - reazione al “code and fix” originario
- Modello sequenziale lineare
  - progressione sequenziale (in cascata) di fasi, senza ricicli, al fine di meglio controllare tempi e costi
  - definisce e separa le varie fasi e attività del processo
    - nullo (o minimo) overlap fra le fasi
  - uscite intermedie: semilavorati del processo (documentazione di tipo cartaceo, programmi)
    - formalizzati in struttura e contenuti
  - consente un controllo dell’evoluzione del processo
    - attività trasversali alle diverse fasi

Modello Waterfall [AC96]



[AC96] V. Ambriola, G. A. Cignoni, Laboratorio di Programmazione, Jackson Libri, 1996.

fasi alte del processo ***fasi basse del processo***

Modelli a Cascata: organizzazione sequenziale delle fasi

- Ogni fase raccoglie un insieme di attività omogenee per metodi, tecnologie, skill del personale, etc.
- Ogni fase è caratterizzata da:
  - Attività (tasks),
  - Prodotti di tali attività (deliverables),
  - Controlli di qualità/stato (quality control measures)
- La fine di ogni fase è un punto rilevante del processo (**milestone**)
- I semilavorati output di una fase sono input alla fase successiva
- I prodotti di una fase vengono “congelati”, ovvero non sono più modificabili se non innescando un processo formale e sistematico di modifica

I documenti prodotti con il modello Waterfall

Activity	**Output documents**
Requirements analysis	Feasibility study, Outline requirements
Requirements definition	Requirements document
System specification	Functional specification, Acceptance test plan Draft user manual
Architectural design	Architectural specification, System test plan
Interface design	Interface specification, Integration test plan
Detailed design	Design specification, Unit test plan
Coding	Program code
Unit testing	Unit test report
Module testing	Module test report
Integration testing	Integration test report, Final user manual
System testing	System test report
Acceptance testing	Final system plus documentation

Modello a cascata: vantaggi e svantaggi

- Pro
  - ha definito molti concetti utili (semilavorati, fasi ecc.)
  - ha rappresentato un punto di partenza importante per lo studio dei processi SW
  - facilmente comprensibile e applicabile
  - E' un modello molto rigoroso: si applica bene in qualunque contesto in cui i requisiti siano stabili e chiari
- Contro
  - interazione con il committente solo all'inizio e alla fine
    - requisiti congelati alla fine della fase di analisi
    - requisiti utente spesso imprecisi: "l'utente sa quello che vuole solo quando lo vede"
    - Errori nei requisiti scoperti solo alla fine del processo
  - il nuovo sistema software diventa installabile solo quando è totalmente finito
    - né l'utente né il management possono giudicare prima della fine dell'adesione del sistema alle proprie aspettative

Valutiamo il modello a cascata

- **Comprensibilità**:
- **Visibilità** :
- **Supportabilità**:
- **Accettabilità**:
- **Affidabilità**:
- **Robustezza**:
- **Mantenibilità**:
- **Rapidità**:

Nella realtà …

- l'applicazione evolve durante tutte le fasi
  - overlap e loop sono inevitabili!
  - in alcuni casi è auspicabile sviluppare prima una parte del sistema e poi completarlo (utente finale= mercato)
  - … la manutenzione non può essere considerata marginale