CORSO I.F.T.S.

"TECNICHE PER LA PROGETTAZIONE E LA GESTIONE DI DATABASE"

Matricola 2014LA0033

DISPENSE DIDATTICHE
MODULO DI "PROGETTAZIONE SOFTWARE"

Dott. Imad Zaza

Lezione del 07/07/2014

Prologo

Obbiettivi Generali

- Elementi di ingegneria del software
- Programmazione ad oggetti
  - Ancora su classi, eriditarietà, polimorfismo ...
  - UML 2.0
  - Design Pattern

Obbiettivi specifici

- Ciclo di vita di un software
  - Waterfall, Agile, V & V ...
- Tecniche e strumenti disponibili nelle varie attività del processo di sviluppo software: specifica, progettazione, verifica, manutenzione,misura, gestione

Argomenti

- Introduzione: visione d'insieme, qualità del software, principi dell'ingegneria del software
- Progettazione: modularizzazione, progettazione orientata agli oggetti, architetture, design pattern e framework

Strumenti di sviluppo

- Eclipse
  - Modellazione Uml:
    - Papyrus project
  - Documentazione requisiti:
    - ProR
- In aziende "ricche" si userebbe Ibm rhapsody e Doors

Target

- Alla fine del corso si dovrebbe essere in grado di:
  - Capire come strutturare e guidare un grande progetto software
  - Capire come raccogliere i requisiti
  - Capire come progettare software ad oggetti
  - Capire come produrre buon codice e per impostare un processo per aiutare un team a produrre buon codice
  - Comprendere le difficoltà intrinsiche nello sviluppo di software e soprattutto come affrontarle

Target (cont.)

- Alla fine del corso si dovrebbe essere in grado di:
  - Comprendere e utilizzare i principali modelli di sviluppo del software
  - Comprendere e utilizzare strumenti di sviluppo e gestione software.
  - Capire che è necessario testare e integrare, e che la sperimentazione e l'integrazione non è qualcosa che si fa in un giorno

Ingengneria del software

Elementi

Software

Programmi, le procedure, e l'eventuale documentazione associata e i dati relativi all'operatività di un sistema di elaborazione.

Definizione formale di Ingegneria del software

IEEE (1990): Strategie sistematiche, a partire da richieste formulate del committente, per lo **sviluppo**, **esercizio** e **manutenzione** del software

Definizione informale

- L'ingegneria del software tratta della realizzazione di sistemi software di dimensioni e complessità tali da richiedere uno o più team di persone.
- L'ingegneria del software è la disciplina tecnologica e manageriale che riguarda la *produzione sistematica* e la *manutenzione* dei prodotti software che vengono sviluppati e modificati entro tempi e costi preventivati.
- L'ingegneria del software è un insieme di teorie, metodi e strumenti, sia di tipo tecnologico che organizzativo, che consentono di produrre applicazioni con le desiderate caratteristiche di *qualità*.

Dove viene applicata ?

- Costruzione di sistemi software
  - grandi e complessi
  - prodotti da squadre
  - esistenti in diverse versioni
  - in funzione per anni o decenni
  - che subiscono evoluzioni

Storia

- Decade Anni 50-60
  - Il software doveva risolvere problemi relativamente semplici e ben compresi
  - L'utente e il programmatore erano la stessa persona
  - Ingegneria del software = programmazione

Storia

- Decade 60-70
  - Calo dei costi dell'hardware, maggiore diffusione dei computer
  - Software applicato a problemi vari, più complessi, non sempre ben compresi
  - Utente != programmatore

Primi problemi

- Difficoltà di comunicazione fra utenti e programmatori, requisiti del software non chiari
- Tecniche di programmazione non adatte a grossi sistemi
- Parti dei sistemi software fortemente accoppiate:
  - necessità di coordinazione continua fra i programmatori
  - difficoltà nella sostituzione di chi abbandonava i progetti
  - la modifica di una parte influenzava l'intero sistema

The major cause of the software crisis is that the machines have become several orders of magnitude more powerful! To put it quite bluntly: as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem »

Dijkstra

Conseguenze

- sistemi poco soddisfacenti, poco affidabili, che non rispettavano i tempi e i costi previsti
- Si comprende la necessità di un approccio sistematico alla produzione del software
- L'espressione "ingegneria del software" nasce in questo periodo

Nascita dell'ingegneria del software

- 1968 il termine divenne populare in una conferenza NATO a Garmisch Partenkirchen
- Decade 60 - 70:
  - Linguaggi e strumenti
  - Approcci rigorosi a:
    - specifica
    - verifica
  - Standard

Metodologie pratiche, organizzative e gestionali

1987 Brooks

- Non esiste un " proiettile d’argento” cioè una soluzione magica ai problemi del software"
- La progettazione e lo sviluppo del software richiedono sforzo intellettuale, creatività e tempo

Problemi

- Sempre più sistemi sono sotto controllo software
- Spese per il software decuplicate
- Necessità di prodotto sempre più affidabili
- Contenimento dei costi

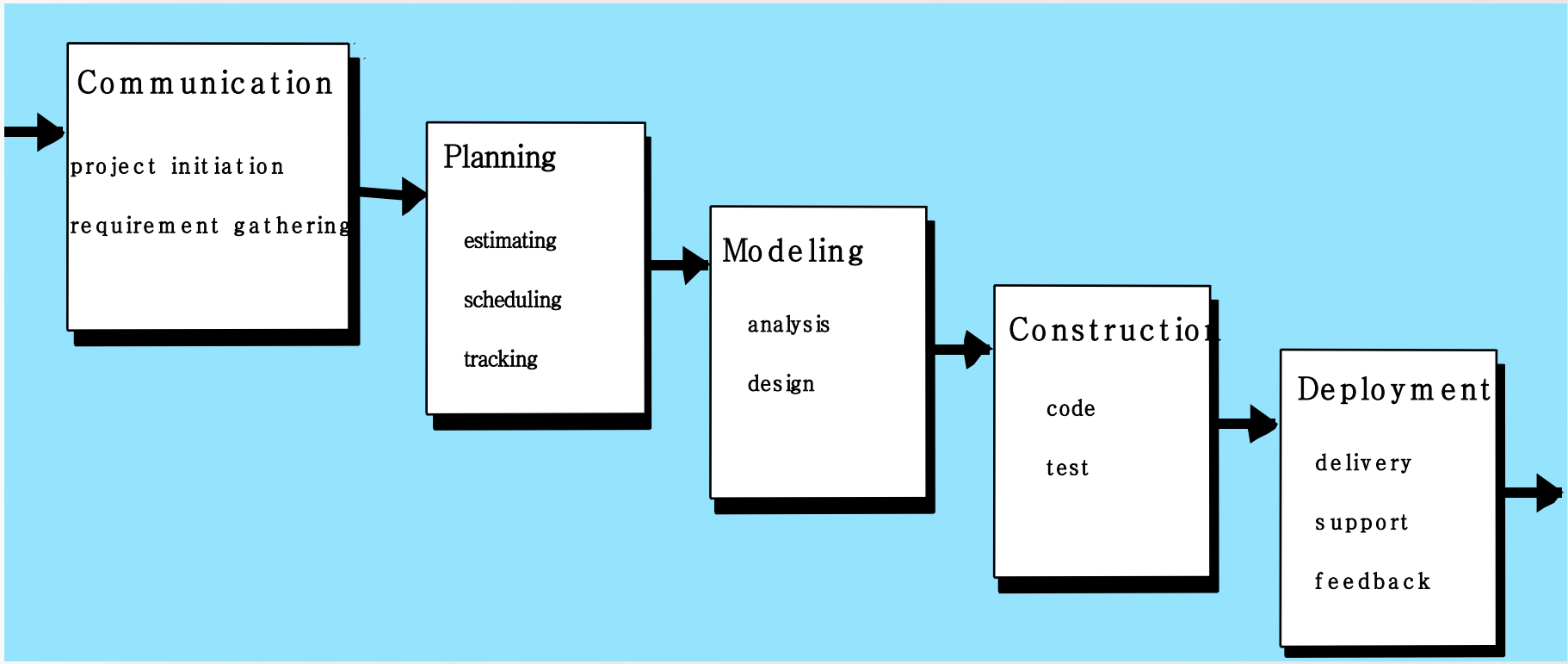
Non solo programmare

- Comprendere i requisiti e tradurli in specifiche precise
- Progettare
- Operare a diversi livelli di astrazione
- Comunicare con la squadra, gli utenti, i clienti
- Rispettare tempi e costi

Fasi definite

- Prima dell'implementazione
  - Studio di fattibilità
  - Specifica del problema
  - Progettazione del programma
  - Selezione o sviluppo degli algoritmi, analisi
- Implementazione
  - Codifica
  - Debugging
- Dopo l'implementazione
  - Test, verifica e benchmarking
  - Documentazione
  - Manutenzione

Modello a cascata



Studio di Fattibilità

- l sistema può essere integrato con gli altri sistemi esistenti?
  - Il sistema contribuisce agli obiettivi aziendali?
  - Il sistema è fattibile con il budget previsto e con le tecnologie attuali?
  - Il sistema può essere integrato con gli altri sistemi esistenti?

Studio di Fattibilità

- Input:
  - Obiettivi aziendali
  - Descrizione sommaria del sistema
  - Come il sistema dovrebbe supportare gli obiettivi aziendali
- Output:
  - Documento che dà risposte motivate alle tre domande della diapositiva precedente

Studio di fattibilità

- Deve individuare le domande pertinenti.
  - Cosa succederebbe se il sistema non fosse costruito?
  - Quali sono i problemi che il sistema potrebbe risolvere o attenuare?
  - Quali sarebbero i contributi diretti del sistema agli obiettivi aziendali?
  - Il sistema può interfacciarsi con altri sistemi aziendali e non?
  - Il sistema richiede tecnologia attualmente non utilizzata dall'azienda?

Studio di Fattibilità

- Fonti di informazioni:
  - Manager dei dipartimenti interessati
  - Ingegneri del software
  - Esperti della tecnologia
  - Utenti finali
- Se il report non è completamente positivo, può suggerire modifiche a
  - Budget
  - Tempi previsti
  - Tipologia o caratteristiche del sistema

Studio di fattibilità

- Valutazione delle alternative
  - costruire il prodotto ex-novo
  - acquistarlo da terzi
- Per ciascuna di queste, valutazione di soluzioni
  - costi, risorse necessarie e possibili date di consegna
  - valutazione di costi e benefici

Studio di fattibilità

- Le precedenti valutazioni sono importanti
  - portano ad un’offerta al cliente e, successivamente, al contratto con il cliente

Analisi e specifica dei requisiti (COSA)

- Dopo studio di fattibilità
- Obiettivo: identificare e documentare i requisiti che il sistema software dovrà rispettare
- In caso di sistemi innovativi, richiede ampia interazione fra cliente e ingegnere
- Il documento dei requisiti dovrebbe:
  - essere comprensibile per l'utente finale
  - poter portare alla creazione del manuale utente e di
- Casi di test

Progettazione e specifica di sistema(COME)

- Obiettivo: progetto di un particolare sistema che soddisfi i requisiti
- Divisa in due fasi:
  - Progetto architetturale: organizzazione globale del sistema in componenti di alto livello e loro interazioni
  - Progetto dettagliato: scomposizione a livelli di dettaglio sempre maggiori, fino a una specifica diretta della codifica

Codifica e verifica di modulo

- Produzione del codice sorgente secondo il progetto dettagliato del sistema

- Verifica che i singoli moduli prodotti rispettino la loro specifica

Integrazione e verifica di sistema

I singoli moduli prodotti vengono

- assemblati
- integrati
- testati come sistema unico

Consegna e manutenzione

- Quando il sistema ha superato tutti i test viene consegnato all'utente
  - Successivamente, si rende necessaria la manutenzione del sistema:
  - per correggere i difetti che si manifestano
  - per adattare il sistema a cambiamenti dei requisiti (evoluzione)

Considerazioni

- Presume che:
  - si possa definire esattamente quali sono i requisiti
  - i requisiti non mutino
  - E' legato ai processi industriali basati su catena di montaggio

Problemi

- Tende a spostare in avanti le verifiche
  - più avanti è l'errore maggiore è il problema
- Forte disallineamento tra quello pensato e quello realizzato
  - il prodotto è developer-centered

Model Based

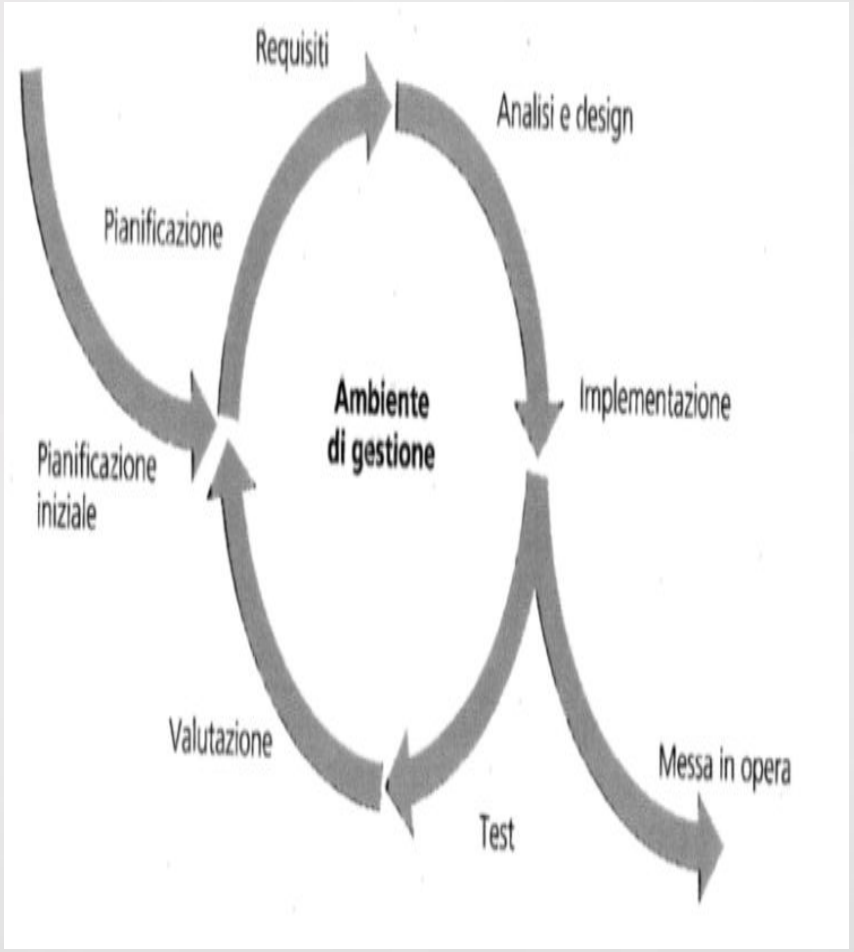
- Prototipo: realizzazione parziale di sistema, costituita allo scopo di permettere ai committenti, utenti e sviluppatori una miglior conoscenza del problem e delle sue soluzioni
  - Usa e getta
  - Prototipo evolutivo

Model Based (Cont.)

- L'uso di prototipi aiuta la stesura dei SRS
  - Il commitente interagisce con un oggetto e non con carta
  - Il prototipo può evidenziare aspetti non considerati
  - Alcuni requisiti possono essere generati direttamente dal prototipo
  - Il processo è ordinato, esiste una sequanza di rilasci di prototipi e srs fino a quello finale

Modello Evolutivo

- Successivi ampliamenti attraverso rifiniture
- Il sistema cresce incrementale

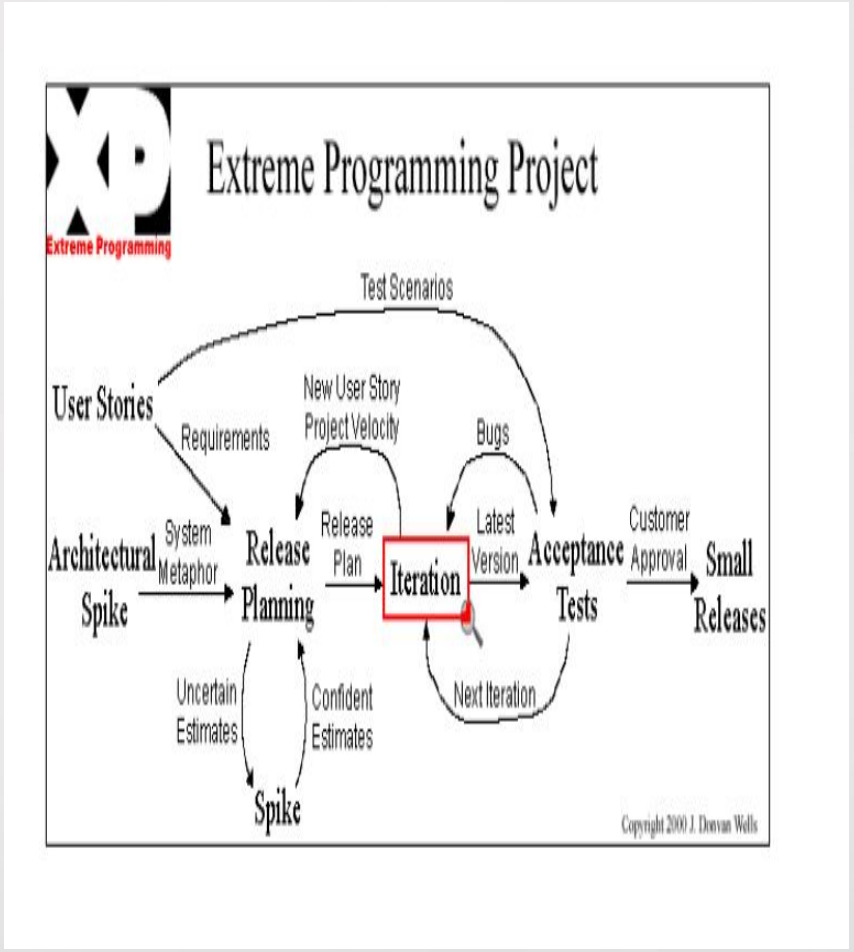


UP

- Proposto da UML
- Ogni iterazione è fatta di
  - Requisiti, analisi, progettazione, implementazione, test
  - E' guidato dai casi d'uso

XP

- Xp is a lightweight methodology for small to medium size teams developing software in the face of vague or rapidly changing of requiriments (Kent Beck)



Che cosè ?

- E' una disciplina di sviluppo software
- Obblighi:
  - Scrivere test prima del codice
  - Programmare a coppie
  - Integrare frequentemente
  - Comunicare col commitente giornalmente
  - Seguire le direttive del commitente
  - Il codice deve essere pulito giornalmente
  - Il processo e le metodologie devono essere adattate all'ambiente di sviluppo

Vantaggi

- Evita produzione di similavorati diversi da quelli necessari all'applicazione
- Fornisc meccanismi che assicurano agli sviluppatori di non discostarsi dal prodotto desiderato dal committente

Attività

- Si ripetono durante lo sviluppo del codice
  - Scrittura codice (Coding)
  - Verifica funzionalità (Testing)
  - Interazione col commitente (Listening)
  - Progetto dell'applicazione (Design)

Criteri

- I programmatori sono responsabili sia del codice che del testing
  - Prima di scrivere codice pensare alla verifica
  - Nessun segmento di codice è parte dell' applicazione finchè non è stato verificato

Sunto

- Painificazione attività
- Brevi iterazioni, rilasci frequenti
- Progetti semplici
- Integrazione continua
- Ristrutturazione del codice
- Verifica ogni funzionalità
- Programmazione a coppie
- Proprietà colletiva del codice
- Partecipazione committente
- Uso di standart di codifica
- Rinuncia al lavoro straordinario (40h/Settimana)

Vantaggi (Cont)

- Coinvolgimento committente
- Attività più gratificante per il programmatore
- Riduzione costi
- Maggior addattamento ai cambiamenti in corso d'opera

Problemi

- Loop Code and Fix
- Il team deve essere composto di buoni elementi

I sistemi critici

- Sostanzialmente viene eseguito il criterio del metodo a cascata
  - Analisi dei requisiti, del rischio, tracciabilità dei requisiti, pianificazione, assicurazione di qualità, versionamento

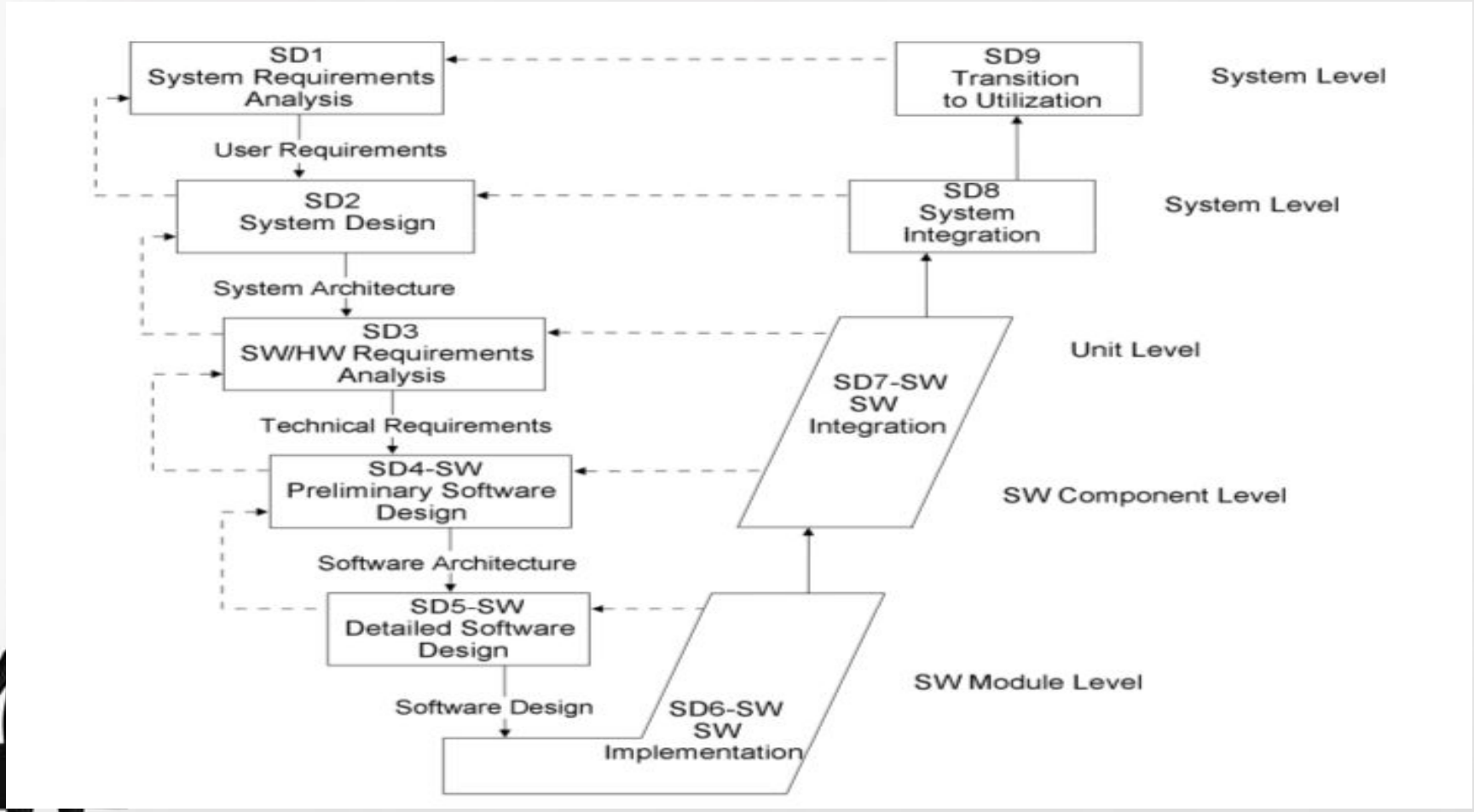
Sicurezza

- Sicurezza = Safety + Security
  - Safety= Condizione di essere sicuro, libero da pericoli, rischi o danni
  - Security= Qualcosa che garantisce la Safety
- Safety Integrety Level per indicare la misura della garanzia che un sistam dà di essere libero da determinati pericoli
  - (Livello crescente 0,1,2,3,4)

Esempi

- Sistemi embedded, medicali, controllo (Aereo, Ferroviario, Centrali Nucleari ...)
- Il processo utilizzato è una derivazione di quello a cascata detto modello a V.
- Normative standart per i vari settori

Modello a V


SD1
System Requirements Analysis
User Requirements
SD2
System Design
System Architecture
SD3
SW/HW Requirements Analysis
Technical Requirements
SD4-SW
Preliminary Software Design
Software Architecture
SD5-SW
Detailed Software Design
Software Design
SD6-SW
SW
Implementation
SD7-SW
SW
Integration
SD8
System
Integration
SD9
Transition
to Utilization
System Level
System Level
Unit Level
SW Component Level
SW Module Level

Modello a V

- Originariamente sviluppato come standart per la German Federal Administration
- E' un modello di processo per pianificare e realizzare progetti.
  - Migliora la trasparenza
  - Gestione
  - Probabilità di successo definendo metodi concreti associati a risultati e responsabilità di ruoli
  - Pubblico
  - Adottato da molte industrie e specializzato a necessità

Considerazioni: relazione con altri campi

- Linguaggi di programmazione
- Sistemi operativi
- Basi di dati
- Intelligenza artificiale
- Metodi formali

Linguaggi di programmazione

- Forte influenza diretta dell'ingegneria del software sui linguaggi di programmazione:
  - Costrutti per programmazione modulare
  - Costrutti per la gestione delle eccezioni
  - Separazione tra interfaccia e implementazione

Influenza inversa

- Specifiche di progetto rigorose, adatte all'implementazione in un linguaggio di programmazione
- Formalizzazione del linguaggio con cui si scrive l'input di un software (da Job Control Language di OS 360 a shell di Unix)
- Formalizzazione per favorire l'automazione

Sistemi operativi

- Hanno avuto influenza notevole sull'ingegneria del software:
  - primi sistemi di grandi dimensioni
  - macchine virtuali, livelli di astrazione, separazione fra politiche e meccanismi

Influenza inversa

- modularità e portabilità dei sistemi operativi moderni:
  - separazione dell'interprete dei comandi
  - architetture a micro-kernel

Basi di dati

- Indipendenza dalla rappresentazione dei dati (astrazione, separazione degli interessi)

- uso di DBMS come componente risolve "gratis" i problemi legati all'accesso concorrente a grosse quantità di dati

Influenza inversa

- Necessità di nuovi modelli per usare DBMS come strumenti per l'ingegneria del software
  - memorizzazione di codice sorgente, documenti, file binari
  - memorizzazione di diverse versioni dello stesso oggetto

Intelligenza artificiale

- Tecniche per produrre sistemi software grossi e complessi con requisiti incerti (sviluppo esplorativo)
- Utilizzo della logica

Influenza inversa

- Sistemi di IA modulari (a regole o shell)

- Concezione di tecniche di intelligenza artificiale da
- applicare a ingegneria del software:

  - Assistenti di programmazione

  - Elaborazione del linguaggio naturale per le interfacce

Metodi formali

- Modelli formali utilizzati in ingegneria del software (specifiche e modelli):
  - automi a stati finiti
  - State charts
  - reti di Petri
  - logica matematica

Influenza inversa

- Sviluppo di nuovi metodi formali:
  - specifiche algebriche
  - tipi di dato astratti
  - logica temporale

Scienze organizzative

- Modelli gestionali applicati a produzione del software per effettuare stime, gestire le risorse umane, monitorare il processo:
  - Necessità di nuovi modelli
- Ingegneria dei sistemi (SoS):
  - Software come componente di un sistema più complesso