La gestione del progetto software

Il Processo di sviluppo del SW

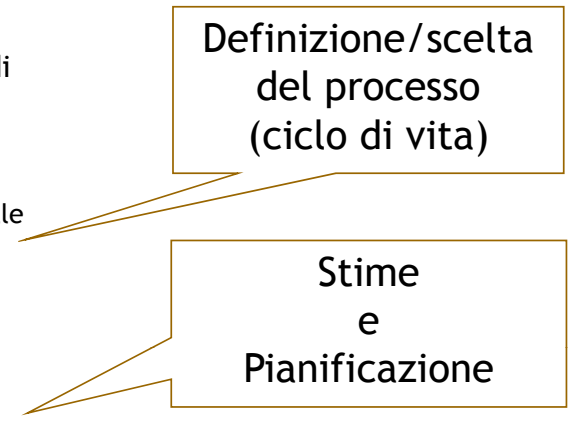
- Il Software è il componente la cui complessità e criticità determina in modo decisivo i costi complessivi di realizzazione e gestione di una “applicazione informatica”
- Per questo motivo è fondamentale (come in molti altri contesti) la definizione delle caratteristiche del processo di sviluppo del software, delle sue peculiarità e delle metodologie per affrontarlo in modo “ingegneristico”

Pianificazione e controllo

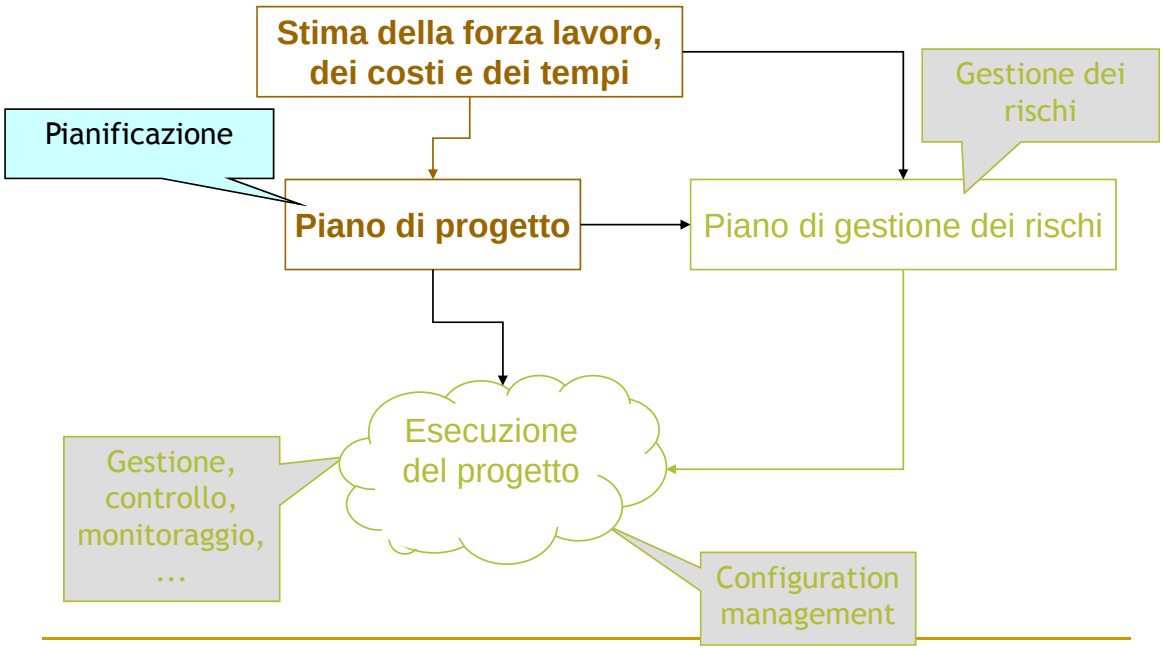
- Per affrontare queste sfide è necessario svolgere attività di:
  - **Pianificazione del progetto**: definire gli obiettivi, le strategie e i piani secondo i quali il progetto verrà condotto e sviluppato
  - **Organizzazione del progetto**: definire la struttura organizzativa del gruppo di lavoro a cui verrà affidato lo svolgimento del progetto e l'assegnazione delle diverse responsabilità funzionali
  - **Staffing**: identificare, reperire, formare e valutare le risorse umane necessarie allo svolgimento del progetto
  - **Direzione** comprende tutte le attività legate alla conduzione del progetto e, in particolare, il coordinamento delle attività, il decision-making e la supervisione del personale
  - **Controllo del progetto**: verificare che il progetto si svolga secondo i piani formulati durante la fase di pianificazione

Pianificazione del progetto

- Le principali attività della fase di pianificazione sono le seguenti:
  - Definizione degli **obiettivi** del progetto
  - Definizione delle **strategie** e delle **politiche** di gestione
  - Definizione delle **procedure** e delle **regole** standard
  - Scelte operative
  - Stesura del **piano operativo**
  - Definizione del piano di **gestione dei rischi**
  - Definizione del **budget**

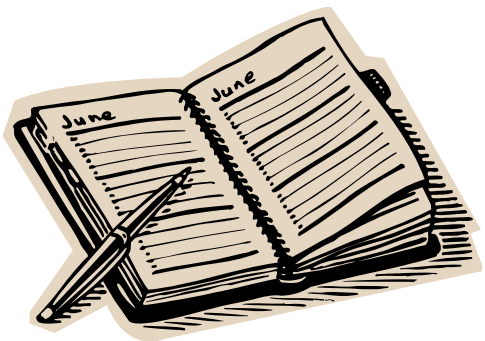


Elementi del progetto software



Summary Slide

- Cos'è un progetto
- Cos'è il Project Management
- Processi di PM e processi di prodotto
- Il progetto software



Il progetto 1/2

- Le organizzazioni eseguono lavoro
- Il lavoro implica attività che:
  - Sono eseguite da persone
  - Sono vincolate da risorse limitate
  - Sono pianificate, eseguite… controllate
- Tutte le attività sono progetti? Cos'è un PROGETTO?

Il progetto 2/2

- **PROGETTO: uno sforzo TEMPORANEO intrapreso per creare un prodotto (o un servizio) UNICO**
- Temporaneo
  - Indica che ha un ben definito inizio e una ben definita fine
    - Finisce quando si sono raggiunti gli obiettivi oppure è chiaro che NON si possono raggiungere gli obiettivi
  - NON indica necessariamente una temporaneità del prodotto realizzato durante il progetto
- Unico
  - Indica che il prodotto (o servizio) è diverso per qualche aspetto distintivo dagli altri prodotti (o servizi)

Note sui progetti

- Questa definizione non limita i progetti che possono
  - Essere intrapresi a qualunque livello dell'organizzazione
  - Coinvolgere 1 persona o diverse migliaia
  - Durare poche settimane o qualche anno
  - Coinvolgere una singola unità di una organizzazione o diverse organizzazioni sparse nel mondo
- I progetti sono un elemento importante nella strategia aziendale perché sono un mezzo tramite il quale si implementa la strategia
- Esempi di progetti:
  - Sviluppare un nuovo prodotto o servizio
  - Effettuare un cambiamento nella struttura di una organizzazione
  - **Sviluppare un prodotto SW**

Elaborazione progressiva 1/2

- Poiché il prodotto è unico, le caratteristiche che distinguono il prodotto devono essere **elaborate progressivamente**
  - Progressivamente indica **procedere per passi**, per incrementi
  - Elaborato indica **definite con cura, dettaglio e in modo completo**
- Queste caratteristiche devono essere principalmente definite all'inizio del progetto e devono essere sempre più esplicitate e dettagliate
- Attenzione: **Il lavoro che deve essere fatto (project scope) deve rimanere costante anche se le caratteristiche sono elaborate progressivamente**

Elaborazione progressiva 2/2

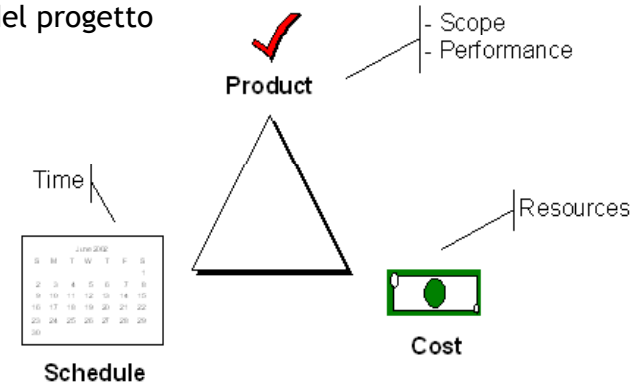
- Esempio
  - Il prodotto di un progetto software può essere definito come “Sviluppare il sito Intranet del LIUC”
  - Nelle prime fasi verrà ulteriormente caratterizzato definendo quali servizi devono essere forniti: documenti per gli studenti, informazioni per i docenti,…
  - Per ogni servizio verranno definite le caratteristiche dettagliate, progettate e implementate,…

Il Project Management

- Il **Project Management è l’applicazione di conoscenze, abilità, strumenti e tecniche alle attività del progetto per raggiungerne gli obiettivi**
- Il lavoro può essere riassunto nelle seguenti attività
  - Definizione di obiettivi, tempi, costi, rischi e qualità
  - Gestire tutte le persone e organizzazioni coinvolte nel progetto (stakeholder) con le loro specifiche necessità e aspettative
  - Gestire il progetto in tutti i suoi processi: inizio, pianificazione, esecuzione, controllo e chiusura

Gli obiettivi di un progetto

- Obiettivi strategici
  - Motivazioni strategiche che portano ad attivare un progetto
  - Risultati attesi come eredità del progetto
  - Nota: rispetto a questi obiettivi si valuta l'investimento giustificando l'assegnazione delle risorse e l'assunzione di eventuali rischi
- Obiettivi specifici
  - Risultati attesi al termine del progetto
    - Qualità
    - Tempo
    - Budget



Il contesto del Project Management

Il Ciclo di vita del progetto 1/3

- L'unicità dei progetti è anche causa di incertezza
- Per avere maggior controllo i progetti sono normalmente divisi in fasi
- Queste fasi sono definite **CICLO DI VITA DEL PROGETTO** che ha le seguenti caratteristiche
  - Ogni fase prevede il completamento di uno o più *deliverable*
    - E' un prodotto tangibile e verificabile (documento, prototipo,…)
  - Ogni fase termina con una verifica dei deliverable e delle performance di progetto per:
    - Decidere se il progetto può passare alla fase successiva
    - Individuare e correggere errori (e le loro conseguenze su costi e piani)

Il contesto del Project Management

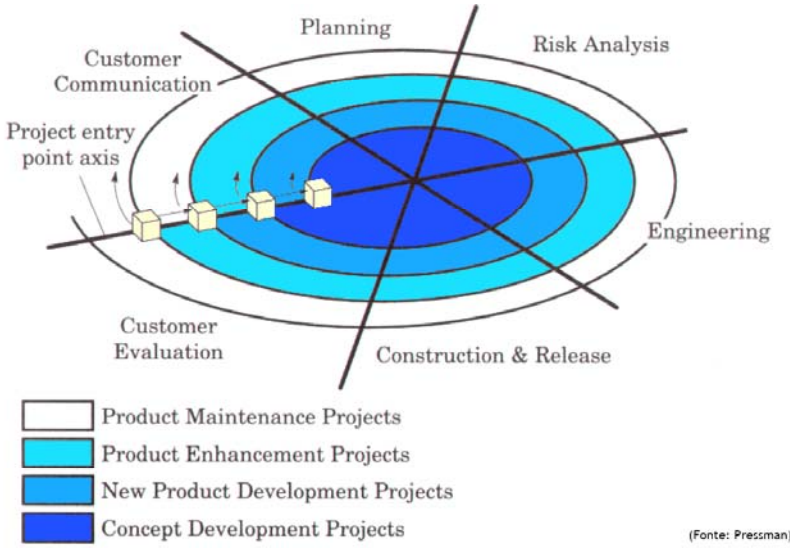
Il Ciclo di vita del progetto 2/3

- Un generico ciclo di vita definisce
  - Che lavoro deve essere fatto in ogni fase
  - Chi deve essere coinvolto in ogni fase
- I diversi cicli di vita hanno alcune caratteristiche in comune
  - I costi e le risorse umane coinvolte sono bassi all'inizio, crescono con il passare del tempo e crollano verso la fine del progetto
  - La probabilità di successo del progetto è bassa all'inizio e cresce al proseguire del progetto
  - La possibilità da parte degli stakeholder di influire sulle caratteristiche del prodotto è alta all'inizio e decresce nel tempo (se così non fosse avrebbe impatti pesanti sul progetto)

Il contesto del Project Management

Il Ciclo di vita del progetto 3/3

- Sebbene diversi cicli di vita hanno nomi simili, ognuno ha le sue peculiarità e caratteristiche specifiche e condiziona l'attività del progetto
- Un esempio molto particolare è il ciclo di vita di progetti software:



(Fonte: Pressman)

Il contesto del Project Management

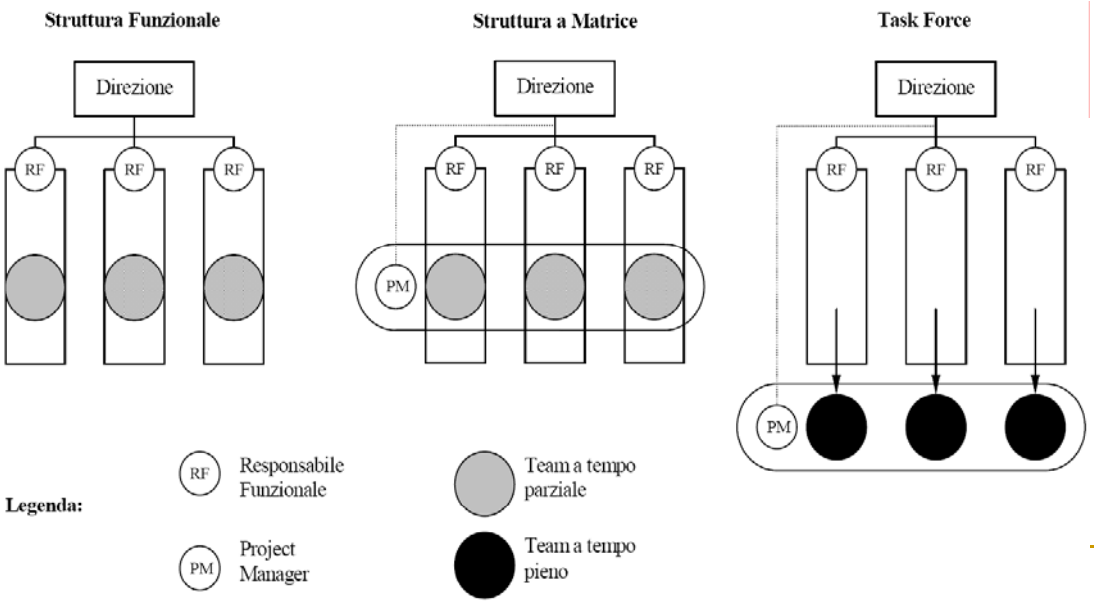
Gli stakeholder

- Sono gli individui e le organizzazioni attivamente coinvolti nel progetto o i cui interessi possono essere influenzati positivamente o negativamente dall'esecuzione/successo del progetto
  - Project Manager – chi gestisce il progetto
  - Cliente – Individuo o organizzazione che userà il prodotto del progetto
  - Esecutore – Unità organizzativa che svolge le attività del progetto
  - Altri: interni, esterni, sponsor, fornitori, sotto-contraenti, agenzie governative, …
- Il problema è gestire le diverse esigenze e aspettative (a volte conflittuali fra loro)
  - Normalmente si devono risolvere a favore del cliente

Il contesto del Project Management

L'organizzazione complessiva

- I progetti coinvolgono una parte di una realtà più complessa la cui organizzazione influisce sull'organizzazione stessa del progetto



Il contesto del Project Management

L'autorevolezza del Project Manager

- Di diritto
  - Poteri delegati
  - Livello organizzativo
  - Grado di influenza
- Di merito
  - Competenze tecniche
  - Competenze gestionali
  - Stile di leadership
- Come fare se non si hanno tutte?

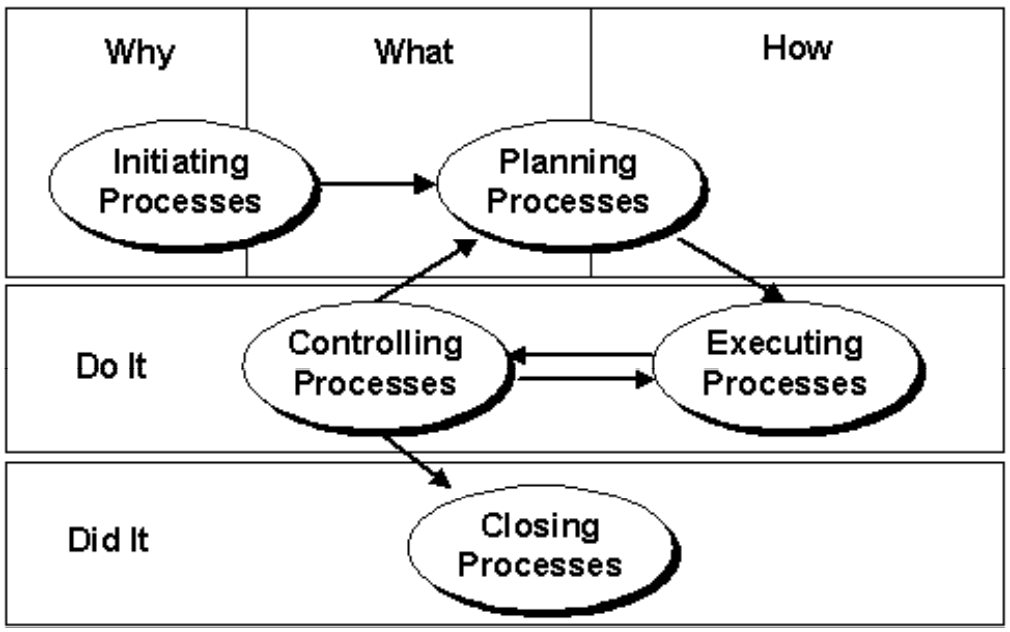
I processi di un progetto

- Un progetto è composto da processi
  - Processo: serie di azioni che portano ad un risultato
- Due categorie principali
  - **Processi project management**: sono relativi alla descrizione e organizzazione del lavoro e sono comuni alle diverse tipologie di progetto
  - **Processi orientati al prodotto**: sono specifici per il tipo di prodotto e costituiscono il **ciclo di vita del progetto**
- Le due tipologie di processi si sovrappongono e interagiscono nel corso del progetto
  - Per esempio non posso definire gli obiettivi e l'ambito del progetto senza avere qualche conoscenza di base su come creare il prodotto

I processi di Project Management 1/4

- I processi di Project Management possono essere organizzati in 5 gruppi (http://www.pmi.org)
  - **Processi di Avvio** – autorizzano il progetto o la fase
  - **Processi di Pianificazione** – definiscono gli obiettivi e il percorso di attività migliore per raggiungerli (piano)
  - **Processi di Esecuzione** – Coordinano le persone e le altre risorse per attuare il piano
  - **Processi di controllo** – assicurano il raggiungimento degli obiettivi di progetto monitorando e misurando i progressi e le variazioni dal piano (per poter attuare azioni correttive)
  - **Processi di chiusura** – formalizzano l'accettazione del progetto o della fase e portano alla sua chiusura

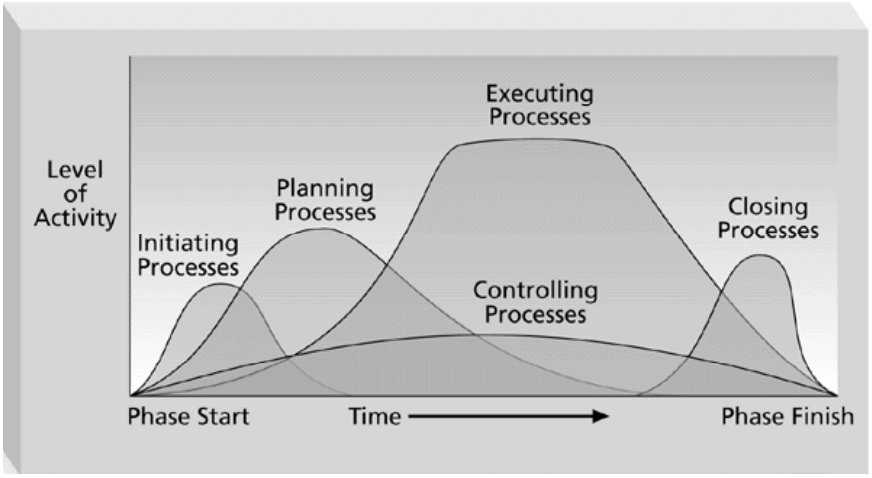
I processi di Project Management 2/4



Futrell, Shafer, "Quality Software Project Management"

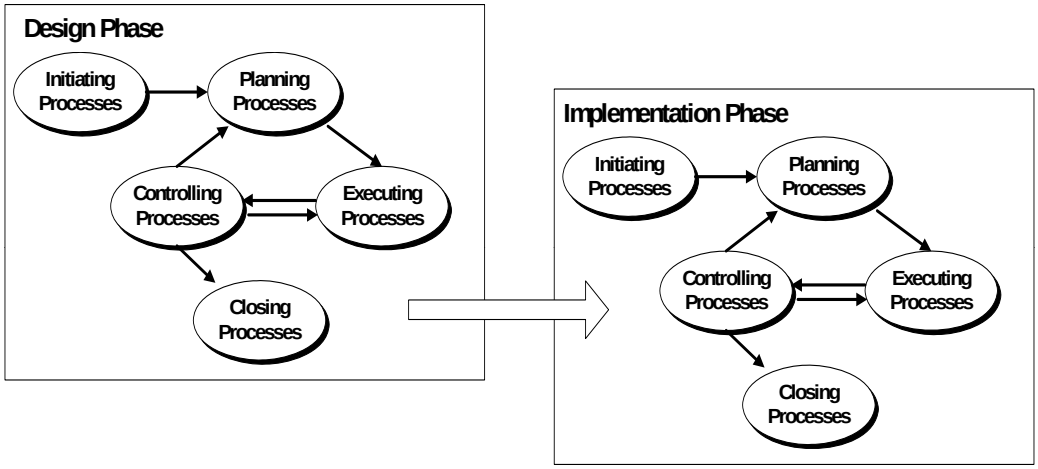
I processi di Project Management 3/4

- I gruppi processi di PM non sono rigidamente sequenziali, ma si sovrappongono nel tempo



I processi di Project Management 4/4

- I processi di PM intervengono in ogni fase del ciclo di vita del prodotto



I Processi orientati al prodotto

- Nel nostro corso ci concentriamo sulla gestione di una particolare tipologia di progetto: **i progetti SW**
- Il contesto è particolarmente interessante perché
  - Il SW è un prodotto con caratteristiche molto particolari
  - I processi che ne gestiscono la produzione sono particolarmente complessi
  - I ragionamenti sono facilmente estensibili anche ad altri contesti specifici
- Esistono alcune peculiarità che distinguono il SW da altri tipi di beni

Peculiarità del SW 1/2

- **Complessità**
  - Il SW è caratterizzato dall'avere un numero di stati elevato
    - Difficile identificare tutti i comportamenti un programma
    - Complesso trasmettere ad altre persone informazioni sul programma
- **Labilità**
  - Uno stesso problema può essere risolto da più programmi
    - Il programmatore non ha reali vincoli su come deve essere usato un linguaggio di programmazione
    - Esistono diversi progetti e diverse architetture in grado di soddisfare uno stesso problema
  - Ne consegue che dato un problema (anche ben espresso e ben compreso) è molto difficile "vedere" (e prevedere) la soluzione corrispondente

Peculiarità del SW 2/2

- **Modificabilità**
  - Il software è dal punto di vista operativo assai facilmente modificabile e riproducibile
  - Il software è continuamente modificato per
    - Correggere errori e, soprattutto
    - Fornire nuove funzionalità
  - Una fetta non trascurabile di effort finisce nella gestione delle diverse versioni dei vari componenti software (version and configuration management)
- **Invisibilità**
  - A differenza di altri manufatti, il software non dispone ancora di tecniche efficaci per descriverne la struttura, le funzionalità, le prestazioni
  - La difficoltà di formalizzare la descrizione del sistema contribuisce a complicare la possibilità di stimarne le proprietà

Il Processo di sviluppo del SW 1/4

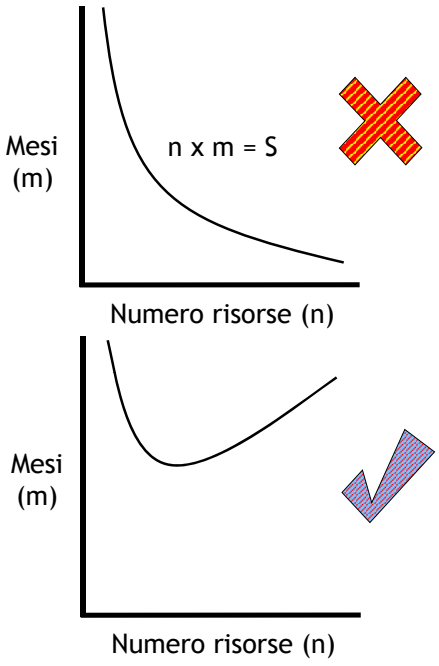
- Abbiamo visto che il Software è un componente la cui complessità e criticità è molto elevata
- E' fondamentale (come in molti altri contesti) la definizione delle caratteristiche del processo di sviluppo del software, delle sue peculiarità e delle metodologie per affrontarlo in modo "ingegneristico"
- Brooks [*The mithical Man-Month*] evidenzia alcuni miti nel processo di produzione del software...

Il Processo di sviluppo del SW 2/4

- *Tutti i programmatori sono ottimisti*
  - Chi sviluppa un'applicazione è tendenzialmente portato a considerare in modo ottimistico l'evolversi del proprio lavoro
    - Tende a minimizzare i problemi
    - Ritiene di dominare in pieno il problema applicativo e gli strumenti di lavoro
  - In realtà
    - La tipologia del lavoro è intrinsecamente legata alla sua creatività
    - Ne deriva un elevato livello di incertezza e discrezionalità
- Impatto: Scelta oculata di chi deve stimare e pianificare

Il Processo di sviluppo del SW 3/4

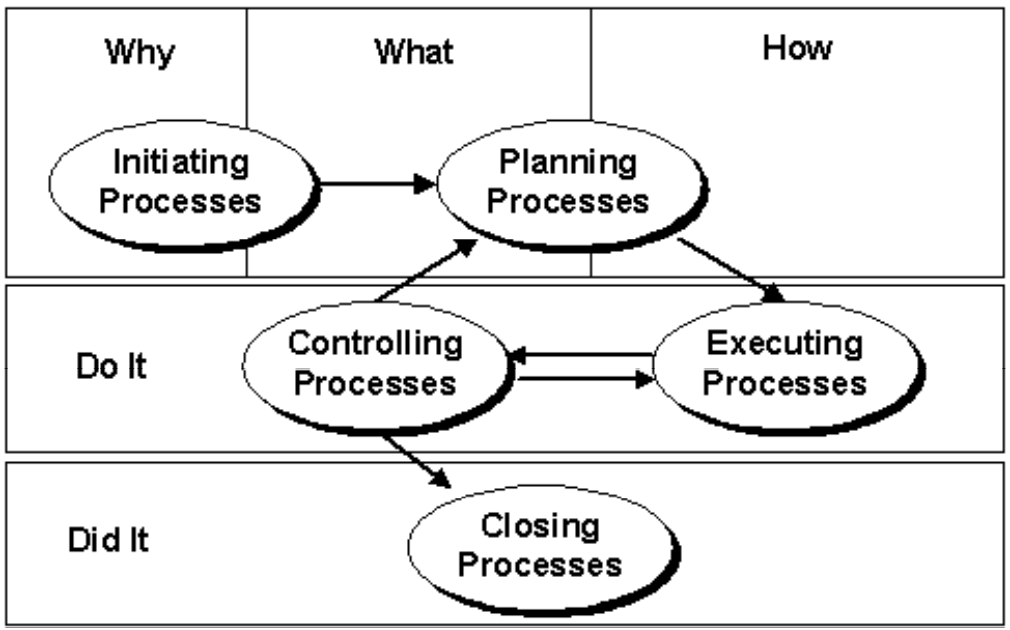
- *Il mese-uomo come unità di misura è un mito pericoloso e illusorio*
  - Per misurare la quantità di lavoro è consuetudine utilizzare il cosiddetto mese-uomo (MM - lavoro svolto da un uomo in un mese)
  - Il MM viene usato per indicare l'interscambiabilità tra numero delle risorse impiegate e tempo di sviluppo
  - ATTENZIONE: nello sviluppo SW la comunicazione ha un ruolo essenziale → → → ...
- **Impatto: NON è vero che posso sempre ridurre i tempi aumentando le risorse in gioco**



Il Processo di sviluppo del SW 4/4

- *Aggiungere nuove risorse ad un progetto che è in ritardo può ritardarne ulteriormente la conclusione*
  - Deriva da quanto detto prima
  - E' giustificato dalla necessità di "distrarre" il personale già attivo sul progetto per
    - Formare le nuove risorse
    - Coinvolgerle nel progetto
- Impatto: NON è semplice gestire e controllare un progetto SW

Come procediamo…



Futrell, Shafer, "Quality Software Project Management"

Come procediamo…

Pianificazione del progetto

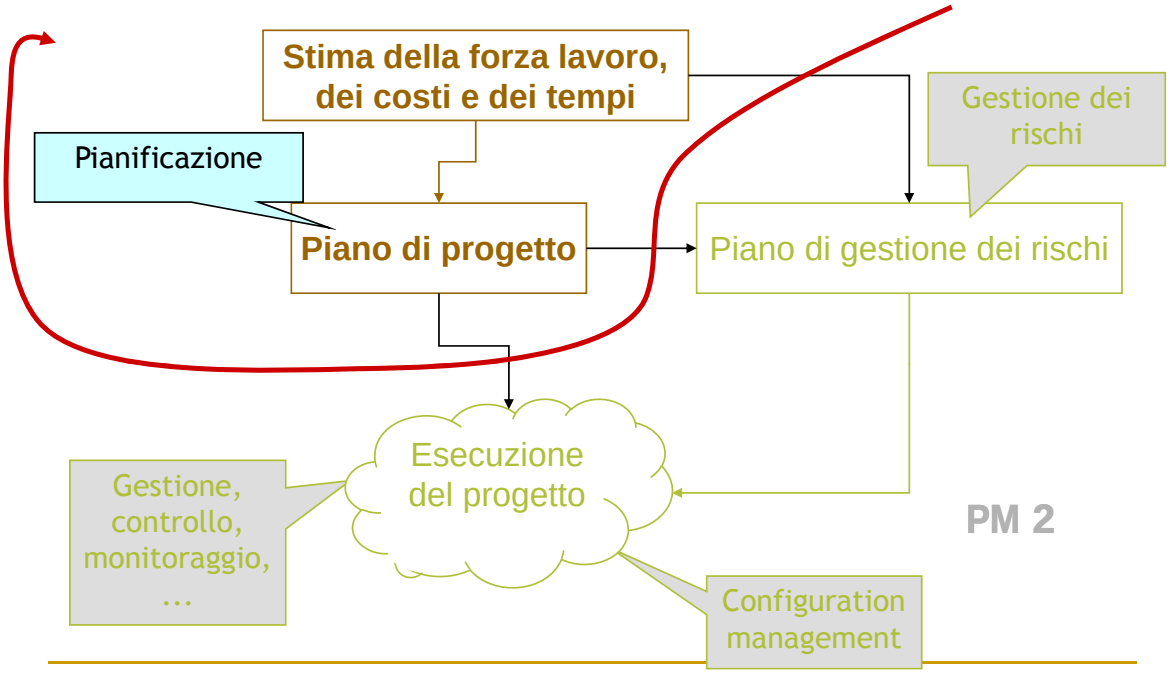
- Le principali attività della fase di pianificazione sono le seguenti:
  - Definizione degli obiettivi del progetto
  - Definizione delle strategie e delle politiche di gestione
  - Definizione delle procedure e delle regole standard
  - Scelte operative
  - Stesura del piano operativo
  - Definizione del piano di gestione dei rischi
  - Definizione del budget

Definizione/scelta del processo (ciclo di vita)

Stime e Pianificazione

Come procediamo…

Argomenti



Il contesto aziendale tipico

Ruoli

Responsabile Tecnico	E' colui che ha la qualifica ed il grado aziendale opportuni per impegnare le risorse dell'Azienda nella realizzazione della fornitura
Capo Progetto	È la figura professionale cui compete la responsabilità delle attività connesse alla esecuzione di un Contratto
Cliente	Ente esterno con il quale esiste od e' esistito almeno un contratto di fornitura. A volte, in modo un po' improprio, si utilizza il termine "Cliente" anche quando, più correttamente, si dovrebbe parlare di Prospect
Responsabile Commerciale	Ha il compito di supportare il Capo Progetto e la Struttura Tecnica in generale, nelle attività di gestione (conduzione, varianti, ecc.) del Contratto

Il contesto aziendale tipico

Tipologie di progetto

- **Chiavi in mano**
  - Progetti di sviluppo software o di integrazione di prodotti (System Integration), che possono anche prevedere la fornitura di materiale hardware
- **Tempo e spese**
  - Attività professionali, contrattualmente pattuite, riconducibili a singole persone/figure professionali, a fronte delle quali è stabilita una tariffa unitaria di fatturazione al Cliente (tariffa giornaliera, tariffa oraria, ecc.)
- **Prodotti**
  - Pacchetti software di proprietà dell'Azienda o di Terze Parti, che potranno eventualmente essere customizzati sulla base delle esigenze del Cliente
- **Servizi**
  - Prestazioni non riconducibili a realizzazione di specifici moduli/prodotti software (chiaramente identificati nell'oggetto del contratto), né a specifiche prestazioni di persone/figure professionali tariffate al Cliente unitariamente
    - Es. manutenzione di applicazioni con livelli di servizio predeterminati

Bibliografia 1/3

- Testo di riferimento
  - Ghezzi, Fuggetta, Ingegneria del SW – Progettazione, sviluppo e verifica, Mondadori informatica
  - WebBook of Software Engineering
    - http://www.cefriel.it/~alfonso/SEBook/index.htm
  - Project Management Institute
    - http://www.pmi.org
    - http://www.pmi.org/prod/groups/public/documents/info/pp_pmbok2k_conf.asp
  - Columbia University - Q7503 Principles of Software Project Management
    - http://www.columbia.edu/~jm2217/

Bibliografia 2/3

- Stime
  - PARAMETRIC ESTIMATING HANDBOOK, Second Edition (Spring 1999), DoD USA, http://www.ispa-cost.org
  - COCOMO http://sunset.usc.edu/research/COCOMOII/index.html
  - D. J. Reifer: Estimating web development costs: there are differences (WEBMO) www.reifer.com/documents/webcosts.pdf

Bibliografia 3/3

- Processi evolutivi
  - RUP
    - Software Project Management, A Unified Framework
      W. Royce, Addison Wesley
    - http://www-136.ibm.com/developerworks/rational/products/rup
    - http://www-106.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf
  - Metodologie Agili
    - http://www.martinfowler.com/articles/newMethodology.html
    - http://www.stsc.hill.af.mil/crosstalk/2002/10/highsmith.html
    - http://www.extremeprogramming.org
  - Confronto metodologie agili e tradizionali
    - http://www.cutter.com/articles.html

Processi di Sviluppo Tradizionali

Un mondo perfetto… forse!

Definizioni

- **Prodotto software**
  - **Insieme di tutti gli artefatti che permettono l'utilizzo di un programma da parte di un utente**: codice, documentazione, prodotti intermedi quali casi di test, manuali tecnici, ecc.
- **Processo software**
  - **Insieme organizzato di pratiche** (attività) che sovrintendono alla costruzione del prodotto da parte del team di sviluppo utilizzando metodi, tecniche, metodologie e strumenti
  - É suddiviso in varie fasi secondo uno schema di riferimento (il **ciclo di vita** del software)
  - Descritto da un **modello**: informale, semi-formale o formale (maturità del processo)

Importanza dell'adozione di un Processo di Sviluppo

- Una gestione sistematica del progetto
  - Aiuta a **pianificare** e a rispettare i tempi prestabiliti per la consegna del prodotto ("Time-to-Market")
  - Aiuta a **stimare** i costi di un progetto
  - Aiuta a **tenere sotto controllo** i costi di produzione
  - Consente di affrontare più semplicemente le problematiche di
    - **Gestione sistematica dei requisiti**
    - **Gestione dei cambiamenti**
      - Gestione dell'evoluzione dell'applicazione nel tempo
      - E' impossibile congelare i requisiti utente!!

Modello di processo 1/2

- Descrive la **strategia** secondo cui si organizzano le attività e si utilizzano metodi e strumenti nel processo di sviluppo
  - Dinamica del processo (attività, organizzazione, regole e ruoli)
  - Metodologie (tecniche + metodi)
  - Infrastrutture e strumenti
  - Dati manipolati dal processo (*artifacts* del prodotto)
    - Loro struttura e relazioni
    - Regole generali di accesso e manipolazione ai dati
    - *Configuration Management*

Modello di processo 2/2

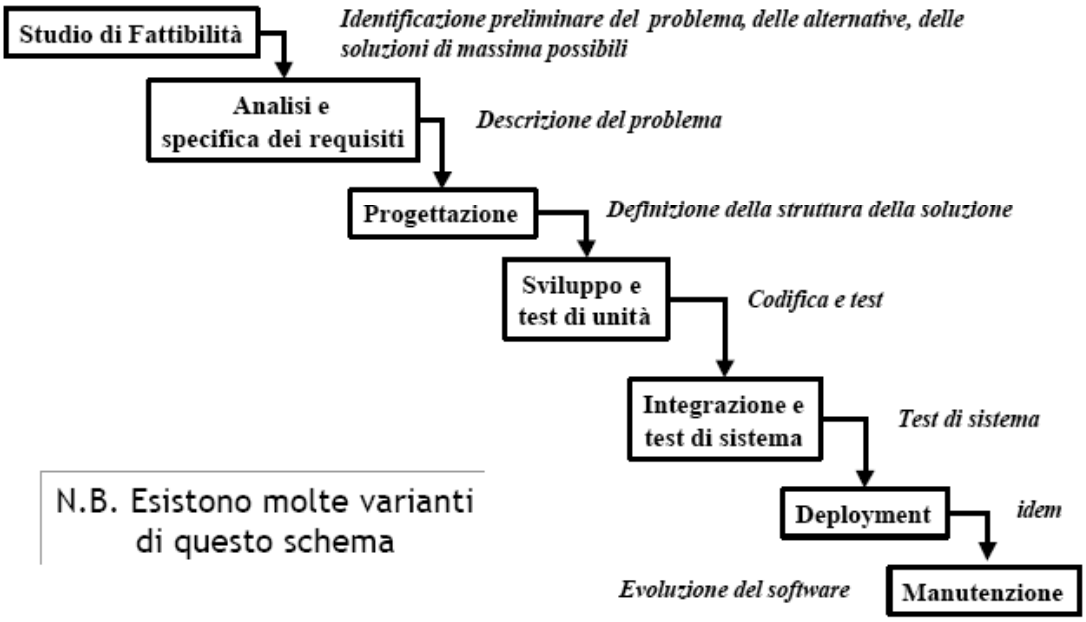
- Tipologie di modelli del processo software:
  - Approccio tradizionale
    - Sequenziale/lineare
    - Prototipale
    - RAD (Rapid Application Development)
  - Approccio evolutivo
    - Incrementale
    - Spirale
    - Assemblaggio di componenti
  - Framework di processi
    - RUP
  - Metodologie leggere

Modello a cascata o sequenziale

- Waterfall, linear sequential model
- Definito ed adottato all'inizio degli anni '70
- **E' un riferimento per tutti gli altri processi**
  - Le fasi indicate da questo modello sono il punto di partenza per tutti gli altri
- Caratteristiche
  - Le fasi si succedono in modo **sequenziale**
    - La fase successiva non inizia se non è terminata la precedente
  - Ogni fase produce un **risultato finito**
  - Ogni fase è soggetta ad **attività di controllo**
  - I **requisiti** devono essere ben **definiti** e **stabili**
  - L'utente vede il **prodotto solo alla fine** del processo
  - Varianti del modello dipendenti dal rapporto committente/produttore

Modello a cascata

Fasi del ciclo a cascata



N.B. Esistono molte varianti di questo schema

Modello a cascata

Studio di fattibilità

- Identificazione preliminare delle alternative
- Valutazione preliminare di costi e benefici
- **Obiettivi**
  - Individuare le possibili opzioni e le scelte più adeguate
  - Valutare (in termini di massima) le risorse umane e finanziarie necessarie
  - Decidere se avviare o meno il progetto
- **Output**: studio di fattibilità (documento)
  - Descrizione preliminare del problema
  - Scenari, strategie alternative di soluzione
  - Costi, tempi, modalità di sviluppo per ogni alternativa (stime di massima)

Modello a cascata

System/Information engineering

- N.B.: non mostrato nello schema precedente, precede l'analisi e specifica dei requisiti
- **Obiettivi**
  - Identificare i requisiti generali dell'intero sistema (hardware, software, fonti di informazioni, persone,...)
  - Identificare il (sotto)insieme dei requisiti che dovranno essere realizzati nel prodotto software

Modello a cascata

Analisi e specifica dei requisiti 1/2

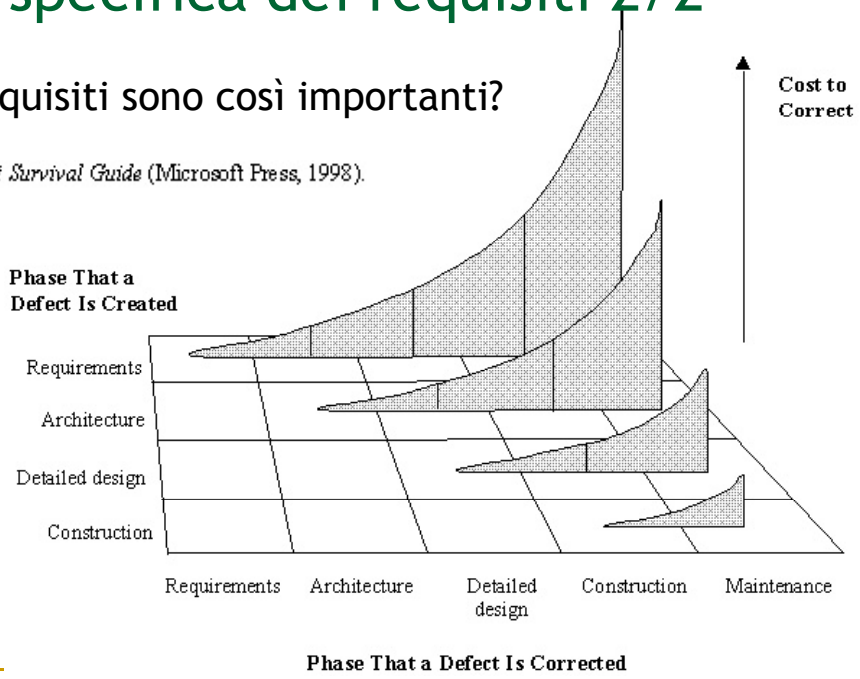
- Deve portare alla comprensione del dominio del problema e dei bisogni dell'utente (in termini di funzionalità richieste, performance, interfacce, ...)
- Coinvolgimento di committente/utente e ingegnere del software (analista)
- **Obiettivo**
  - Descrivere le caratteristiche che dovrà avere l'applicazione
- **Output**
  - Documento dei requisiti
  - Documento di specifica delle interfacce
  - Manuale d'uso
  - Piano dei test di accettazione del sistema

Modello a cascata

Analisi e specifica dei requisiti 2/2

- Perché i requisiti sono così importanti?

from *Software Project Survival Guide* (Microsoft Press, 1998).



Modello a cascata
Progettazione (program design)

- **Obiettivo**
  - Ideazione della struttura che dovrà avere l'applicazione in termini di:
    - Sotto-sistemi, componenti, moduli, classi, ...
    - Interfacce e relazioni fra sotto-sistemi, ecc., protocolli, allocazione delle funzionalità
  - In altri termini: ideazione e modularizzazione della soluzione
- In genere, specie per applicazioni distribuite, si distinguono almeno due livelli di dettaglio:
  - Architectural design
  - Detailed design
- **Output**
  - Documento di descrizione dell'architettura
  - Documento di descrizione del design di dettaglio

Modello a cascata
Codifica

- **Obiettivo**
  - Passare:
    - Dalla descrizione della soluzione in termini di architectural/detailed design
    - Alla descrizione della soluzione in formato eseguibile da un calcolatore
- **Output**
  - Codice sorgente di ciascun modulo, codificato nel linguaggio scelto e testato in isolamento

Modello a cascata
Testing

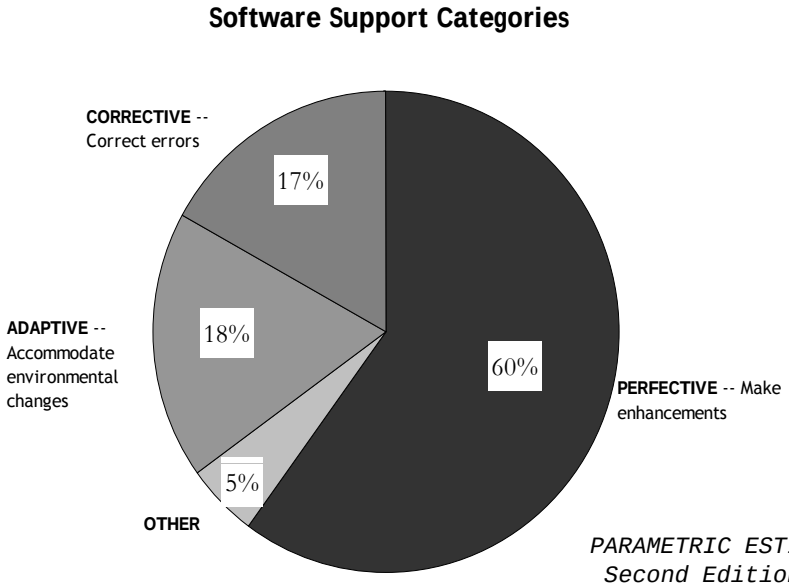
- **Obiettivi**
  - Individuare gli errori presenti
  - Assicurare che l'applicazione abbia il comportamento atteso
- **Output**
  - Applicazione rilasciabile ai clienti
  - Si possono identificare le seguenti sotto-attività:
    - Integrazione e test di sistema
      - Composizione dei moduli/sotto-sistemi
      - Verifica del corretto funzionamento delle interfacce
    - α-test: sistema "rilasciato" al gruppo di test interno
    - ß-test: sistema rilasciato a pochi e selezionati utenti

Modello a cascata
Deployment ed evoluzione 1/2

- **Deployment**
  - Distribuzione ed installazione del software presso l'utenza
- **Evoluzione del software**
  - E' necessario fare evolvere il software per seguire l'evolversi delle esigenze degli utenti
  - L'evoluzione consiste in realtà in iterazioni di (quasi) tutte le precedenti fasi

Modello a cascata

Deployment ed evoluzione 2/2



PARAMETRIC ESTIMATING HANDBOOK, Second Edition (Spring 1999), DoD USA, http://www.ispa-cost.org

Modello a cascata

Analisi del modello

Punti di forza

- Riflette il modello di progettazione tipico del ciclo ingegneristico
- E' facile da usare
- Ogni fase è sottoposta a controllo
- Permette di individuare delle chiare milestone nel progetto

Punti di debolezza

- I progetti nella realtà raramente seguono un approccio sequenziale
- Mal si adatta all'incertezza che esiste all'inizio di ogni progetto
- La prima versione del prodotto è disponibile solo alla fine del processo
- Un ritardo in una fase si ripercuote sull'intero processo

Modello a cascata

Quando usare il modello

Quando può essere usato

- In presenza di requisiti stabili
- Quando si hanno buone competenze sul dominio di business
- In presenza di contratti con costi e tempi ben definiti
- Contesto tecnologico maturo

Quando non può essere usato

- Software complesso e contesto innovativo
- Progetti di lunga durata (mal si adatta ai cambiamenti)
- In presenza di requisiti non ben definiti all'inizio
- In progetti che usano tecnologia di frontiera

Modello prototipale

- Sviluppo rapido di una serie di prodotti in scala ridotta (prototipi) per acquisire informazioni su requisiti non ben definiti
  - Livello di dettaglio dei requisiti insufficiente
  - Incertezza sull'applicabilità di una tecnologia
  - Necessità di verificare aspetti critici (es. interazione con altri sistemi)
- Il prototipo ha lo scopo di:
  - Chiarire i requisiti o identificare requisiti nascosti
  - Sperimentare architetture, tecnologie, ...
  - Valutare requisiti non funzionali (es. prestazioni)

Modello prototipale
Caratteristiche

- Raccolta dei requisiti necessari a capire il problema
- **Progettazione rapida** concentrata sugli aspetti comprensibili dall'utente
- Costruzione di un **prototipo** che viene sottoposto continuamente alla valutazione del cliente
- **Raffinamento** dei requisiti **mediante la valutazione** del prototipo
- **Ciclo iterativo**
- Sviluppo del prodotto finale sulla base del prototipo

Modello prototipale
Tipi di prototipo

- **Disposable prototype**
  - Di norma è realizzato per le interfacce utenti e poi si getta
- **Time box prototype**
  - Di norma realizza una versione in piccolo del prodotto finale e comunque si getta
- **Evolutionary prototype**
  - Evolve nel prodotto finale

Modello prototipale

Analisi del modello

Punti di forza

- Aiuta l'utente a definire cosa gli serve (requisiti)
- Prototipi funzionanti sono rilasciati in tempi brevi
- Le modifiche ai requisiti non costituiscono un problema
- Il prototipo finale rispecchia le aspettative del cliente

Punti di debolezza

- Il prototipo finale, risultando un aggregato progressivo di funzioni, non è ottimizzato
- Per avere un processo veloce la documentazione è penalizzata
- La manutenzione potrebbe essere di difficile gestione
- Il Cliente il più delle volte non capisce che il risultato finale è comunque un prototipo da rifare

Modello prototipale

Quando usare il modello

Quando può essere usato

- Il cliente non ha chiaro gli obiettivi e serve uno strumento per migliorare la comunicazione
- Nello sviluppo sistemi software innovativi
- Nello sviluppo di parti di un sistema dove è preponderante il punto di vista del cliente

Quando non può essere usato

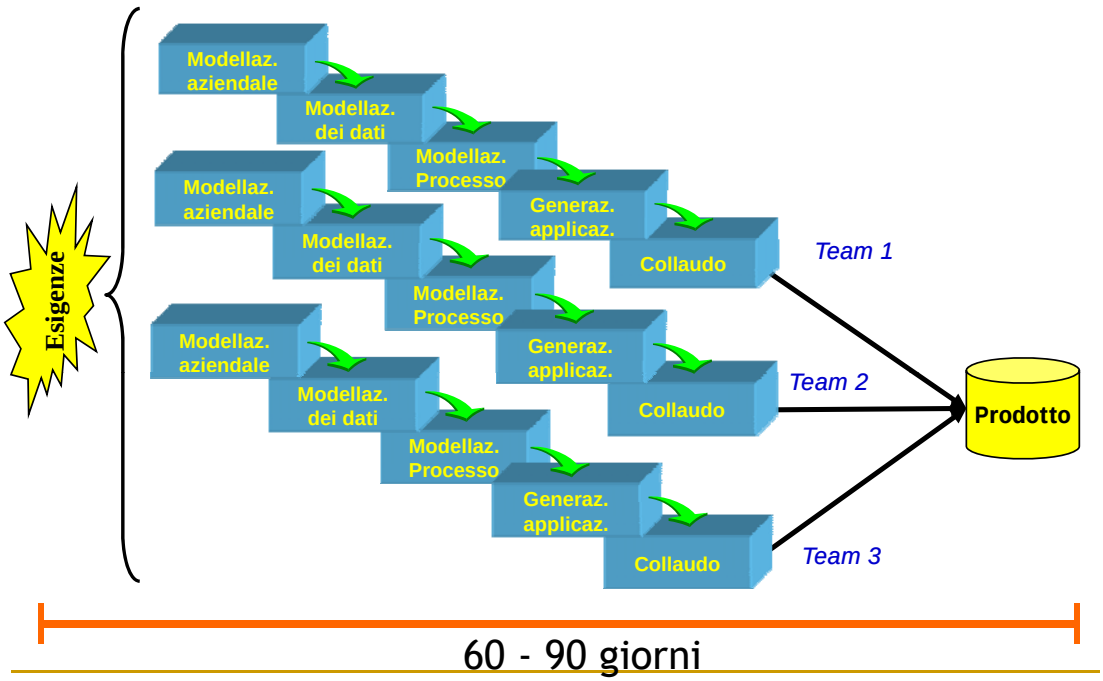
- Nel caso di sistemi software in cui sono chiaramente compresi gli obiettivi applicativi
- Nel caso di sistemi che utilizzano tecnologie mature
- Quando i costi e i tempi sono proibitivi
- Quando le prestazioni del sistema sono vincolanti

Rapid Application Development

- Adattamento del ciclo a cascata
  - **Ciclo di sviluppo breve** (60-90 gg)
- **Modularizzazione e parallelizzazione** dello sviluppo delle funzioni primarie
- **Negoziazione dei requisiti**
- Approccio "Timeboxing": **focus sui tempi e sui costi**
- Uso di **componenti** riusabili e linguaggi di IV generazione
- **Team di progetto di piccole dimensioni** (max 6 persone) con presenza full-time dell'utente

RAD

Caratteristiche



RAD

Ciclo di vita

- Passi principali
- **Business modeling**: analisi dei processi esistenti, del flusso di informazioni
- **Data modeling**: descrizione delle strutture dei dati che compongono le informazioni e loro relazioni
- **Process modeling**: descrizione delle trasformazione subite dai dati in base al flusso delle informazioni fra i passi del processo
- **Application generation**: sviluppo (meglio: generazione di applicazioni) basato sul riuso di componenti e linguaggi di quarta generazione
- **Testing and turnover**

RAD

Analisi del processo

Punti di forza

- Maggior controllo del progetto (focus sui tempi e sui costi)
- Costi ridotti (brevi cicli di sviluppo e codice riusabile)
- Buona identificazione dei requisiti (coinvolgimento dell'utente nel progetto)
- Visibilità e standardizzazione del prodotto

Punti di debolezza

- Ridotte funzionalità secondarie del prodotto (sacrificate dall'approccio "timeboxing")
- Minore efficienza del prodotto (utilizzo di componenti standard)
- Difficile gestione degli sviluppi paralleli

RAD
Quando usarlo

Quando può essere usato

- L'applicazione SW è di tipo "standalone"
- Le performance e l'affidabilità non sono critiche
- Il sistema è scomponibile in moduli indipendenti
- Sono disponibili librerie di componenti riusabili
- Il prodotto ha una ristretta distribuzione (in house o su mercati verticali)

Quando non può essere usato

- L'applicazione ha una forte interazione con altri sistemi
- Il prodotto è "mission-life critical"
- Il sistema non può essere modularizzato
- Non si ottengono vantaggi dall'uso di librerie di componenti riusabili o 4GL

Conclusioni

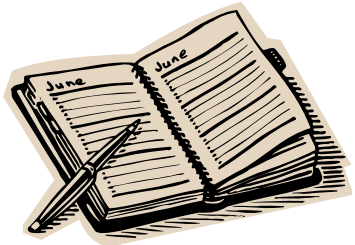
- **I modelli tradizionali sono importanti perché definiscono le attività alla base di qualunque altro processo**
  - Ciclo a cascata: rivolto ad uno sviluppo lineare, definisce le attività elementari
  - Modello prototipale: rivolto ad assistere l'utente (o lo sviluppatore) nella comprensione dei requisiti
  - Modello RAD è destinato a comprimere la durata del ciclo di sviluppo
- **Limite: NON considerano la "natura evolutiva del software"**
- I "modelli evolutivi" si propongono di superare questi limiti con un approccio "a cicli"

Le “dimensioni” del SW

Dimmi quanto pesi e ti dirò chi sei

Summary Slide

- Il costo del SW
  - Cosa stimare
  - Fattori che influiscono sul costo del SW
- Metriche Dimensionali
  - Definizione
  - Modalità di Stima
- Metriche Funzionali
  - Definizione
  - Modalità di Stima
  - Esempio di Calcolo
- Metriche Dimensionali vs Metriche Funzionali



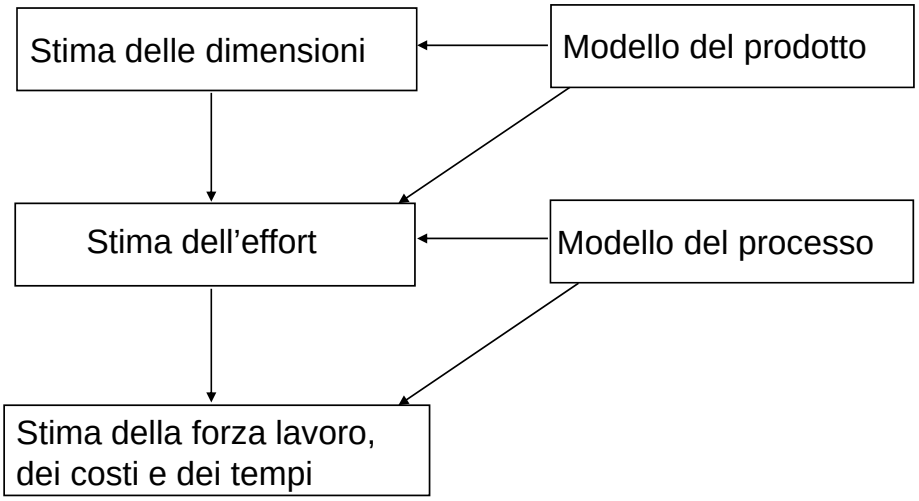
Le fonti di costo

- Il costo di una applicazione informatica è sostanzialmente caratterizzato dal costo delle risorse necessarie al suo sviluppo:
  - **Personale tecnico**: programmatori, team leaders, project manager
  - **Personale di supporto**: segreteria, supporto amministrativo, supporto tecnico
  - **Risorse informatiche**: costi di ammortamento e di manutenzione dell'hardware e degli strumenti di sviluppo
  - **Materiali di consumo**: carta, nastri per stampanti, ...
  - **Costi generali di struttura**: costo degli uffici, telefono, energia elettrica ...
- **Il costo oggi dipende principalmente dai costi delle risorse umane**

La stima dei costi

- Cosa occorre stimare, e perché
  - **Dimensioni del prodotto**
    - Determina la forza lavoro necessaria
  - **Forza lavoro da impiegare**
    - Perché devo gestire la forza lavoro
    - Per lo scheduling delle attività
    - Perché determina il costo
  - **Tempi di sviluppo**
    - Per la contrattazione
    - Per lo scheduling delle attività
  - **Costo**
    - Determina il prezzo del prodotto
- Nota
  - Tutto questo solo da un punto di vista gestionale (cioè prescindendo da considerazioni tecniche)
  - Per di più **molte di queste stime vanno fatte per fase di sviluppo**

Procedura normale di stima



Difficoltà di produrre stime accurate

- Prevedere con precisione il costo di un sistema software è molto difficile
- Il motivo sta nelle caratteristiche intrinseche del software
  - Complessità
  - Labilità
  - Modificabilità
  - Invisibilità

I fattori che influenzano i costi 1/4

- Sono molti i fattori che influenzano i costi di sviluppo del software
- Risultano particolarmente interessanti quelli proposti da Boehm nell'ambito del COCOMO (principali)
  - Numero di istruzioni da codificare
  - Il fattore umano
  - Complessità dei programmi
  - Stabilità dei requisiti
  - Atri fattori

I fattori che influenzano i costi 2/4

- Numero di istruzioni da codificare
  - Chiaramente il costo aumenta con il numero delle istruzioni da codificare
  - Questo fatto costituisce una delle ragioni di fondo di diverse attività di ricerca:
    - linguaggi ed ambienti della IV generazione
    - strumenti e tecniche di supporto per il riutilizzo del software
- Il Fattore Umano
  - La qualità del software prodotto e la produttività dipendono fortemente dalla inventiva e dalle capacità delle risorse umane impiegate

I fattori che influenzano i costi 3/4

- Complessità dei programmi
  - Il COCOMO definisce tre livelli crescenti di complessità:
    - Organic mode (es.: applicazioni scientifiche ed EDP)
    - Semi-detached mode (es.: compilatori, utilities di sistema, ...)
    - Embedded mode (es.: controllo di processo, avionica, telefonia, ...)
- Stabilità dei requisiti
  - È uno dei fattori più importanti e spesso meno considerati
  - Diviene sempre più critico con l'avanzare del progetto

I fattori che influenzano i costi 4/4

- Altri fattori
  - Prestazioni richieste all'applicazione
  - Ambiente di sviluppo
  - ...
- Si noti che la minore rilevanza dell'ambiente di sviluppo è una ulteriore conferma della ancora scarsa ingegnerizzazione del processo di sviluppo

Centralità della stima delle dimensioni

- **La stima delle dimensioni del prodotto rappresenta in un certo senso "cosa c'è da fare"**
  - Quindi è inevitabilmente alla base di ogni considerazione seguente
- Ci sono altri fattori da considerare
  - complessità del problema
  - esperienza dei progettisti
  - potenza dell'ambiente di sviluppo
  - ...
- **Diversi studi hanno tuttavia dimostrato che questi fattori hanno un'influenza relativamente ridotta sulle grandezze dipendenti (effort, ecc.)**
  - Normalmente sono usati per "tarare" le stime basate sulle dimensioni

Le dimensioni del software

- Come anticipato, la dimensione del software è uno dei fattori che ne influenzano maggiormente il costo
- Servono quindi metriche che consentano una efficace quantificazione di questo elemento
- Due tipologie principali
  - **Metriche dimensionali**: si basano sul numero di istruzioni del programma
  - **Metriche funzionali**: si basano sulle caratteristiche funzionali del programma

Metriche Dimensionali

Metriche dimensionali

- Si basano su una misura diretta delle linee di codice del programma
- La misura si ottiene sottraendo dal numero totale di linee (incluse le dichiarazioni) i seguenti valori:
  - numero di linee di commenti;
  - istruzioni riusate da altri programmi senza modifiche
- Si usano diverse sigle che tuttavia fanno sostanzialmente riferimento allo stesso concetto:
  - LOC (lines of code) o SLOC (source LOC)
  - DSLOC (delivered SLOC)
  - DSI (delivered source instructions) o KDSI (migliaia (Kilo) DSI)

Metriche derivate

- Produttività = LOC/mesi-uomo
- Qualità = Numero totale di errori/LOC
- Costo unitario = Costo totale/LOC
- Livello di documentazione = Pagine di documentazione/LOC

Metodi per la stima delle dimensioni

- Possiamo fare considerazioni analoghe a quelle per la stima diretta dei costi:
  - Per analogia
    - Confronta il progetto con progetti simili
  - Giudizio degli esperti
    - Indicazioni da uno o più esperti
  - Metodi Bottom-up
    - Assegnano i costi ai singoli componenti del prodotto o a fasi del processo e da questi derivano i costi complessivi del progetto
  - Metodi Parametrici
    - Stime basate su algoritmi matematici parametrici

Metriche funzionali

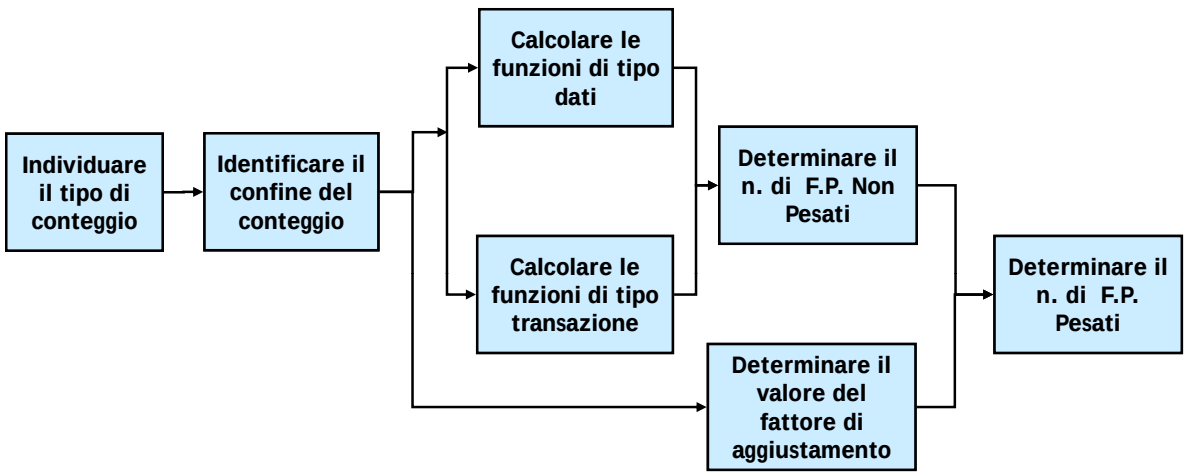
Metriche Funzionali 1/2

- I punti funzione vengono ideati nel 1975 da Allan Albrecht (IBM)
- **L'idea fondamentale è di rappresentare le dimensioni di un prodotto software caratterizzato in base alle sue funzionalità, anziché rispetto alle sue dimensioni fisiche**
- Obiettivi fondamentali
  - Indipendenza dalla tecnologia
    - Ci si basa sui documenti di specifica
    - realizzazioni diverse (per linguaggio di programmazione o per strumenti di sviluppo) danno la stessa misura
  - Misurare tutte le funzionalità fornite all'utente
  - Misurare solo le funzionalità fornite all'utente

Metriche Funzionali

- Possiamo dire che sono modelli parametrici perché usano parametri per calcolare la dimensione del SW
  - Questi parametri NON si basano su algoritmi matematici/statistici
  - Albrecht ipotizza che si possono utilizzare alcune funzionalità dei programmi (input, output, file logici, ecc.) per stimare la stima delle dimensioni
- In generale: misurano un prodotto software quantificando le funzionalità contenute in esso e rilasciate all'utente
- Possono essere utilizzate per effettuare
  - La **stima** nei progetti di sviluppo software
  - Studi sulla produzione (**produttività**) del software
  - Studi sul patrimonio del software di un'azienda o per prendere decisioni **"make" or "buy"**
  - Studi sulla **qualità** del software (n. test x FP, difetti x FP)

Procedura di conteggio dei FP



Perché i FP sono così usati? 1/4

- Forte **esigenza di metodi di stima e valutazione** dei costi di sviluppo
- Essendo calcolati sulle specifiche, **risultano comprensibili anche all'utente finale**
  - Risultano inoltre ricalcolabili con relativa facilità a fronte di modifiche nelle specifiche
- **In teoria** il calcolo può essere fatto da un esperto di FP **senza bisogno di supporto da parte degli esperti** del dominio applicativo (quindi in modo non intrusivo)
  - Ma col pericolo di interpretare scorrettamente le specifiche

Perché i FP sono così usati? 2/4

- Il **calcolo è parzialmente automatizzabile** in congiunzione con l'utilizzo di strumenti CASE
- Sono - almeno **apparentemente - misure oggettive**
  - E anche semplici (un solo numero!)
- Perché supportano un'analogia con altri settori produttivi, dove la produzione è quantificabile facilmente
  - Ma il software è il prodotto di attività intellettuali e ingegneristiche, non manifatturiere!!

Perché i FP sono così usati? 3/4

- **Forniscono una soluzione "algoritmica"** a problemi manageriali che risulta gradita agli informatici
  - Ma **nascondono la complessità reale** del problema e la quantità di tecniche e conoscenze necessarie per una buona gestione del processo software
- Si possono usare (anche se non è corretto) in ragionamenti tradizionali
  - Ad esempio si può stimare il costo in base a considerazioni tecniche e poi, usando fattori di conversione FP/costo medi, determinare la presunta dimensione in FP

Perché i FP sono così usati? 4/4

- Molti sostengono che non esistono alternative ragionevolmente praticabili
- **La verità è che non esistono ricette semplici ed universali che risolvano il problema**
  - né potrebbero esistere, data la complessità del problema
- La soluzione è data un insieme di tecniche, metodologie e strumenti
- In ambito Pubblica Amministrazione è **richiesto** l'uso dei Function Point

Confronto tra le metriche
Metriche Dimensionali

- **Pro**
  - Sono facilmente definibili
  - E' facile misurare il valore di LOC per un qualsiasi programma
  - Sono basate su concetti molto diffusi tra gli addetti del settore
  - Sono utilizzate in molti metodi esistenti per la stima dei costi e per la misura della produttività
  - Quasi tutti i dati storici raccolti in passato sono basati su questo tipo di metriche

Confronto tra le metriche
Metriche Dimensionali

- **Contro**
  - Il numero di linee di codice sorgente risulta fortemente dipendente dalle attitudini e capacità del programmatore
  - Essendo un criterio indipendente dal contenuto del programma, non tiene conto della diversa complessità e potenza delle istruzioni di uno stesso linguaggio o di linguaggi diversi
  - Risulta assai difficile definire in modo preciso quale sia il criterio di conteggio
  - Poiché gli indici di qualità vengono a dipendere dal numero di linee rilasciate, un tentativo di migliorare la produttività potrebbe portare chi sviluppa a scrivere più codice, senza una particolare attenzione alla qualità di ciò che si produce

Confronto tra le metriche
Metriche Funzionali

- **Pro**
  - Presentano dei vantaggi per ciò che concerne i primi due aspetti negativi delle metriche dimensionali
- **Contro**
  - Si basano su una definizione di complessità molto semplificata (dipendente fortemente dal numero di input e di output)
  - Il conteggio è parzialmente soggettivo. Non esiste un criterio univoco per i conteggi, le assegnazioni dei pesi e la valutazione dei parametri correttivi
  - Persone diverse danno generalmente valutazioni diverse
    - IFPUG cerca di fornire un criterio omogeneo

Processi di sviluppo evolutivi

Prevenire o adattarsi… questo è il problema!

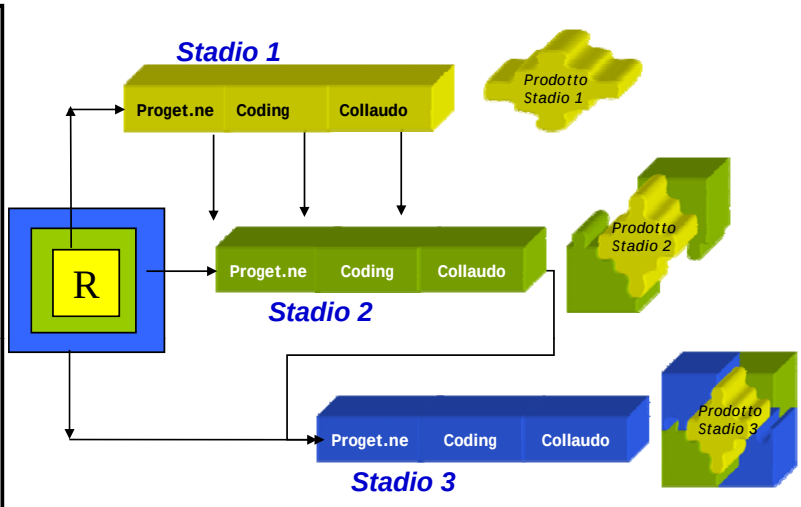
Modelli evolutivi
Caratteristiche generali

- Riconoscono
  - **L'inevitabilità delle evoluzioni dei requisiti, delle tecnologie, del mercato, …**
  - **L'impossibilità di arrivare a realizzare l'intera applicazione in un unico passo**
- Prevedono il rilascio di versioni successive, con funzionalità parziali, per far fronte a:
  - Necessità di seguire le inevitabili **evoluzioni**
  - Pressioni del mercato e dei concorrenti
- Vantaggi
  - Un "primo prodotto finito" è disponibile per l'utente in tempi "rapidi"
- Svantaggi
  - Il "prodotto finale" può richiedere un maggiore tempo ed effort di sviluppo rispetto all'approccio "a cascata" (affermazione vera in caso di requisiti stabili)

Modello incrementale

CARATTERISTICHE

- **Combina gli approcci Sequenziale e Prototipale**
- Consiste nell'applicazione di tanti **cicli sequenziali sfasati nel tempo**
- La caratteristica principale di tale modello è che **risulta applicabile nel momento in cui si è in grado di poter definire gli incrementi che compongono il sistema e di pianificarne lo sviluppo**



Modello incrementale
Caratteristiche

- Ogni incremento è del tutto operativo
  - Differente rispetto alla prototipazione
- Ogni sotto-progetto può differenziarsi rispetto a:
  - Risorse (staff)
  - Complessità - rischi tecnici
- Ogni rilascio stimola una reazione da parte dell'utente
  - Di questa reazione si tiene conto per i rilasci successivi
- I sottoinsiemi di requisiti vengono **ordinati per priorità**
  - La priorità esprime la necessità di ricevere feedback dall'utente
  - I requisiti più critici fanno parte dei primi rilasci
  - Primo incremento = parte "core" del prodotto
- Feedback dall'utenza a ogni deliverable consegnato
- Arricchimento progressivo rispetto a:
  - Requisiti utente aggiornati
  - Feedback dal campo

Modello incrementale
Analisi del modello

Punti di forza

- Il prodotto finale rispecchia le aspettative del cliente
- La produzione può iniziare dalle parti più critiche dove serve un feedback del cliente e si deve fare piu' testing
- Il prodotto è ingegnerizzato e ben documentato
- Le modifiche ai requisiti sono sotto controllo

Punti di debolezza

- All'inizio il prodotto non contiene tutte le funzionalità
- E' rischioso se richiesto un forte controllo del processo
- Non sempre si riesce a pianificare in anticipo il numero di stadi
- Alla lunga potrebbero rivelarsi problemi nell'allocazione delle risorse

Modello a spirale di Boehm

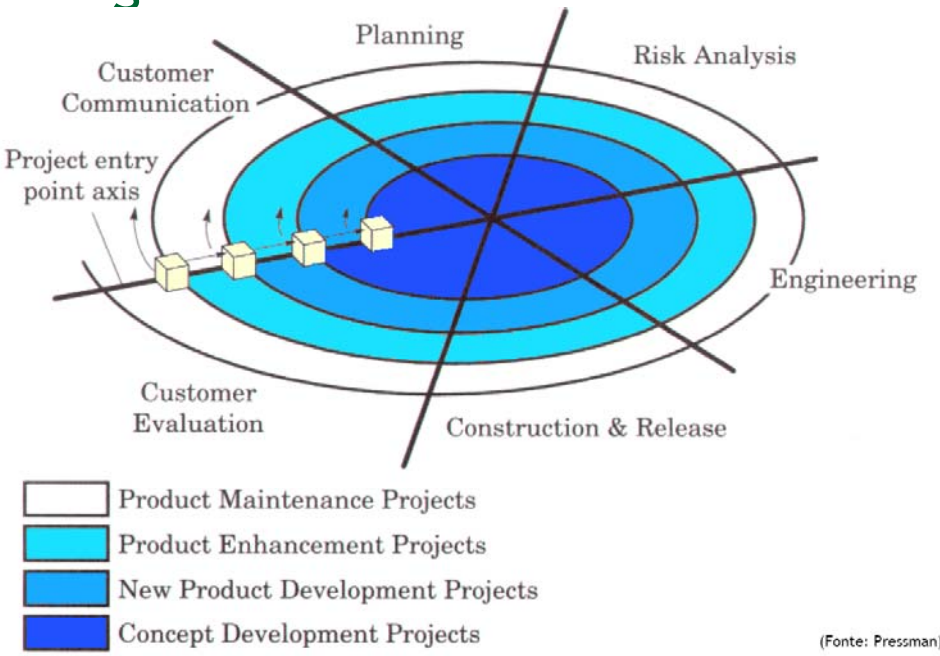
- Secondo Boehm (1998) la scelta del modello di processo deve dipendere da una valutazione dei rischi presentati dalle diverse alternative
- Per questo motivo propone un meta-modello (un modello per descrivere modelli) definito modello a spirale di Boehm di tipo CICLICO
- Suddiviso in 4 regioni
  - Definizione degli obiettivi
    - Si identificano gli obiettivi specifici della fase
  - Valutazione e riduzione dei rischi
    - Si valutano i rischi e si eseguono attività tese a ridurre i rischi più rilevanti
  - Sviluppo e validazione
  - Pianificazione
    - Si fa la revisione del progetto e si pianifica la fase successiva della spirale

Modello a spirale

- Il ciclo di vita a spirale (Bohem - 1998) riconcilia in sé i modelli visti prima
  - Sviluppo in una serie di incrementi
  - I primi incrementi anche "su carta" (es. 1°specifica iniziale)
- Ogni incremento è una ripetizione ciclica di task region:
  - **Customer communication** (comunicazioni con il cliente):
    - Attività volte a stabilire un efficace colloquio tra il cliente e il team di sviluppo
  - **Planning** (pianificazione):
    - Raccolta requisiti e definizione piano di progetto (risorse e scadenze)
  - **Risk analysis** (analisi dei rischi):
    - Stima e prevenzione dei rischi tecnici e di gestione
  - **Engineering** (Strutturazione)
    - Costruzione di rappresentazioni dell'applicazione da sviluppare
  - **Construction & release** (Costruzione e rilascio)
    - Realizzazione, collaudo e installazione
  - **Costumer evaluation** (Valutazione da parte del cleinte):
    - Rilevazione delle reazioni da parte del cliente

Modello a spirale

Dettaglio del ciclo di vita 1/2



(Fonte: Pressman)

Modello a spirale

Dettaglio del ciclo di vita 2/2

- Ogni task region è ulteriormente scomposta in *task*
  - Numero di task e formalità nella loro definizione dipendenti dal tipo di progetto
- A ogni iterazione si ripetono i task relativi a ciascuna task region
  - es. numero di iterazioni, pianificazione, schedule, stime dei task successivi sono ricalcolati ad ogni passaggio dalla task region *planning*
- Attività trasversali: *configuration management*, *quality assurance*, ...
- I cicli della spirale possono corrispondere anche alle evoluzioni "post-rilascio"
- Adotta un approccio *risk-driven* al processo di sviluppo del software
- Fa uso della prototipazione per ridurre i rischi (in qualsiasi momento, non solo nelle prime iterazioni)
- Ciascuna iterazione adotta l'approccio sistematico del ciclo di vita a cascata

Modello a spirale
Analisi del modello

Punti di forza

- Possono essere utilizzati i vantaggi della prototipazione ad ogni ciclo dell'evoluzione
- Si presta alla gestione di progetti complessi e/o in contesti innovativi
- Focalizza l'attenzione sugli obiettivi
- I rischi sono presi in considerazione ad ogni stadio del progetto

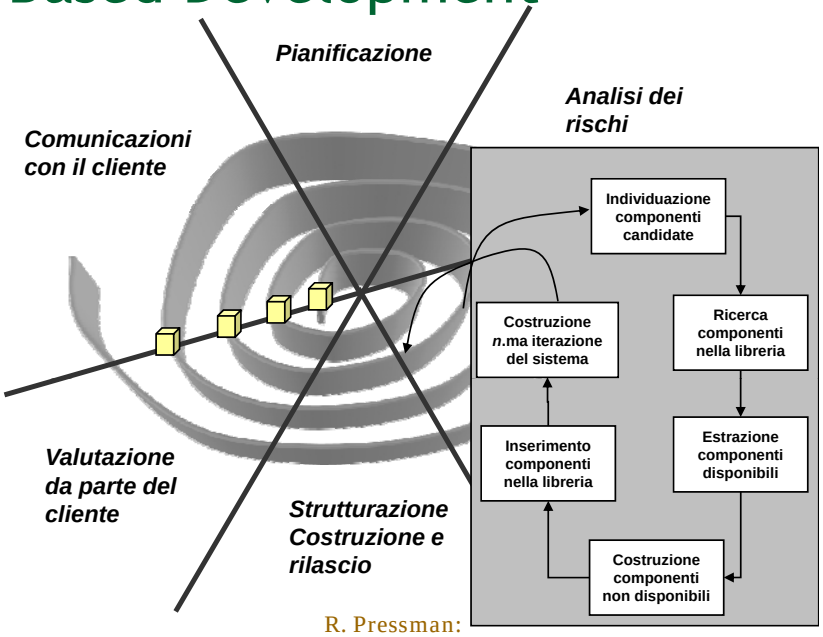
Punti di debolezza

- Il modello esige competenze ad alto livello per la stima dei rischi
- Il controllo della strategia evolutiva potrebbe risultare difficoltoso
- Mancanza di dati statistici sulla sua applicazione (ancora poco diffuso)

Altri modelli
Component Based Development

CARATTERISTICHE

- Si basa sul **modello a spirale**
  - Modifica la fase di Engineering
- Ha come quadro di riferimento la **tecnologia orientata agli oggetti**
- Le applicazioni sono realizzate partendo da **componenti** preconfezionati
- Stimola il **riuso** del software

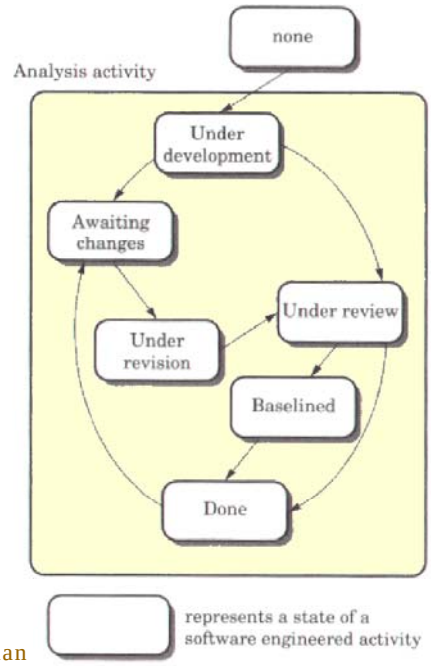


R. Pressman:
Principi di ingegneria del software

Altri modelli

Concurrent engineering

- Considerazione: quando un progetto si trova nominalmente in una fase (es. codifica) si svolgono contemporaneamente anche altre attività (es. design, testing, ...)
- Concurrent engineering:
  - Un progetto consiste di un insieme di attività
  - Ciascuna attività (es. analisi) può trovarsi in un insieme di stati possibili (vedi figura)
  - Le attività possono evolvere parallelamente
- Il modello di processo definisce condizioni ed eventi che fanno evolvere gli stati delle attività



Fonte: Pressman

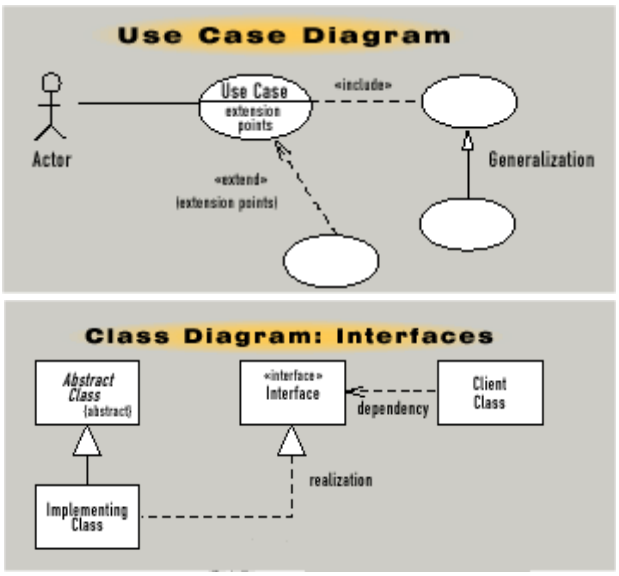
Unified Process - UP

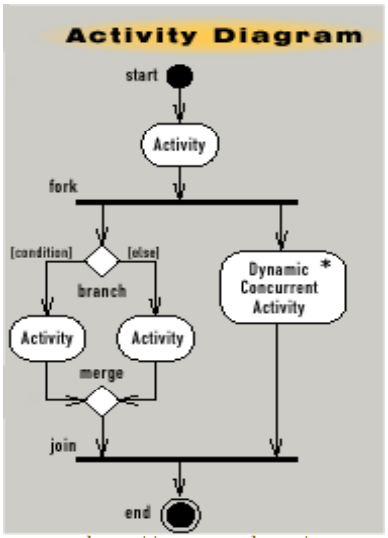
- E' un processo "configurabile":
  - Definisce un "framework" o "generic process" che può essere specializzato per una ampia classe di sistemi software, aree applicative, organizzazioni, livelli di competenza e dimensioni dei progetti
- ma…
  - E' anche un prodotto commerciale sviluppato e mantenuto da Rational Software (Rational Unified Process)
  - Rational fornisce un insieme di tool per automatizzare il più possibile le attività di creazione e manutenzione dei diversi artifatti del processo
- Utilizza UML come linguaggio per la specifica di molti artefatti del processo
- Basato su "Best Practices": approcci allo sviluppo di sistemi software largamente utilizzati con successo nell'industria

Unified Process - UP
Assiomi principali

- UP è un processo:
  - Guidato dagli **use case** e **dal rischio**
    - L'intero ciclo di sviluppo è condotto verso la realizzazione, l'implementazione ed il test degli *use case* definiti
    - UP suggerisce una serie di tecniche per *affrontare fino dalle fasi iniziali il rischio* e fornisce una serie di linee guida da adottare per gestire il rischio durante tutte le fasi di sviluppo
  - Incentrato sull'**architettura**
    - L'approccio è quello di realizzare ed implementare fino dalle *fasi iniziali del progetto un robusto studio architetturale*
    - La validità dell'architettura e l'aderenza del software all'architettura stessa vengono verificate durante tutte le fasi del ciclo di sviluppo

Unified Process - UP
Parentesi: UML





http://www.uml.org/

Unified Process - UP
Ciclo di vita

- Il ciclo di vita di riferimento è l'evolutivo a spirale di Boehm
  - **Iterativo e incrementale**
    - Per assecondare la complessità dello sviluppo
    - Per assecondare i cambiamenti nei requisiti
  - **Enfasi sui modelli** più che sui documenti in linguaggio corrente
    - Più leggibili, più modificabili, più maneggiabili in modo automatico
  - **Centrato sull'architettura**
    - L'architettura dell'applicazione emerge sin dalle prime fasi progettuali
    - Sviluppo in parallelo
    - Riuso e manutenibilità
  - Incoraggia controllo della **qualità e gestione del rischio**
    - E' un processo controllato in tutte le sue fasi
  - E' **supportato da tool CASE** consentono di automatizzare la creazione dei modelli utilizzati nel processo e la l'attività di produzione della documentazione

Unified Process - UP
Processo Iterativo e incrementale 1/2

- **Iterativo**
  - Raffinamenti successivi dell'architettura tramite ricicli delle fasi di analisi/design/implementazione
- **Incrementale**
  - Ciascun ciclo porta ad affinare gradualmente i prodotti (semilavorati) realizzati, che non vengono scartati o riprogettati, ma piuttosto corretti o estesi
- Fornisce una strategia per lo sviluppo di un prodotto software in passi "piccoli" e gestibili
  - "*You plan a little,*
    You specify, design and implement a little,
    You integrate, test and run each iteration a little"

Unified Process - UP
Processo Iterativo e incrementale 2/2

- Si procede con il passo successivo soltanto se il passo corrente ha prodotto risultati "soddisfacenti"
- **Fra due passi successivi è fondamentale avere un feedback** che consenta di "aggiustare il tiro" per il passo successivo
- Le prime iterazioni hanno come obiettivi principali
  - la comprensione dei rischi
  - la valutazione della fattibilità
  - la costruzione del nucleo dell'applicazione
- Le altre iterazioni, ed in particolare le ultime, aggiungono degli "incrementi" a quanto realizzato in precedenza
- Quando tutti i passi sono stati eseguiti il prodotto è pronto per il rilascio finale al committente

Unified Process - UP
Concetti chiave

- Fasi, Iterazioni

Come evolve il processo nel tempo?

- Process Workflows
  - Attività, passi,...

Quali sono le attività che caratterizzano il processo?

- Artefatti (semilavorati)
  - Modelli
  - Reports, documenti

Che cosa viene prodotto?

- Ruoli

Chi lo produce?

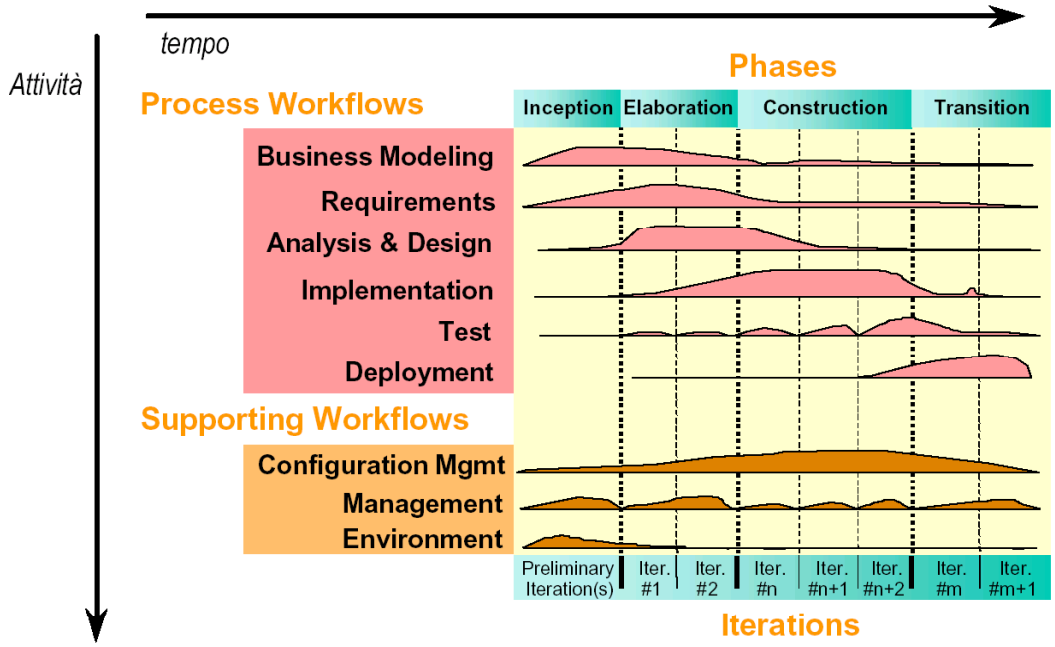
Unified Process - UP

Dimensioni del processo

- Il processo è strutturato secondo due dimensioni
  - Dimensione **temporale**: evoluzione dinamica del processo in termini di cicli, fasi, iterazioni, milestones
  - Dimensione delle **componenti del processo**: attività, artifatti, ruoli e risorse umane coinvolte
- Il processo può essere rappresentato graficamente con una matrice bidimensionale che abbia su un lato la fasi-iterazioni (dimensione temporale) sull'altro le componenti di processo

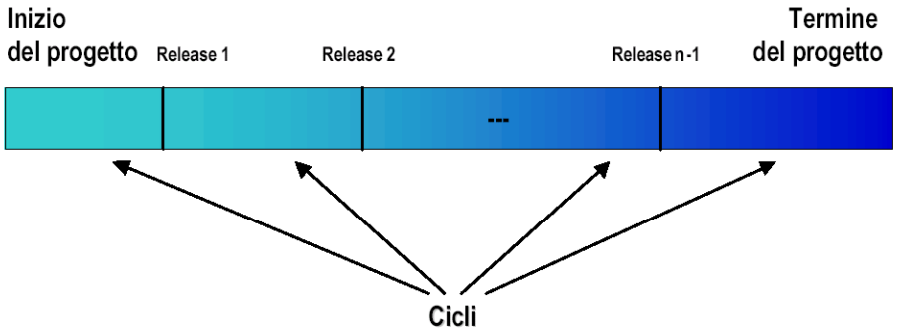
Unified Process - UP

Ciclo di vita



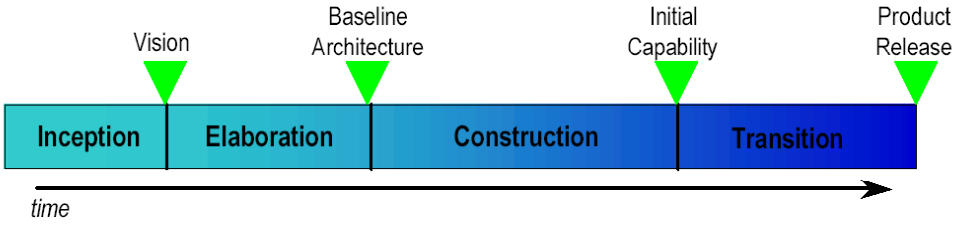
Unified Process - UP
La dimensione temporale

- Il processo viene suddiviso in cicli di sviluppo, ciascuno dei quali opera su una nuova generazione del sistema e si conclude con una release del prodotto



Unified Process - UP
Fasi e Milestone

- **Inception**
  - Identificazione del "Business Case" e definizione dell'ambito e dell'obiettivo del progetto
- **Elaboration**
  - Pianificazione del progetto, specifica delle features e definizione dell'architettura
- **Construction**
  - costruzione del prodotto ed evoluzione dell'architettura e dei piani sino al rilascio
- **Transition**
  - correzione dei difetti
  - consegna e formazione
  - supporto e manutenzione



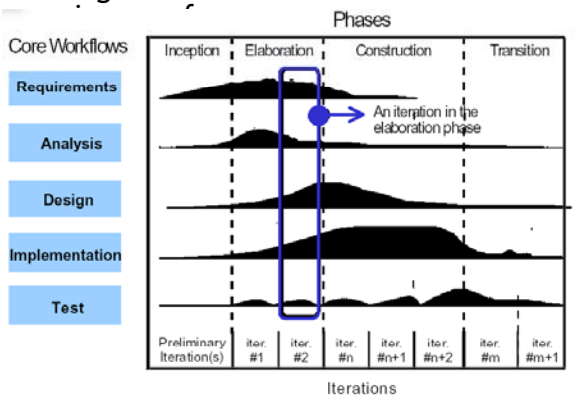
Unified Process - UP
Workflow di processo

- **Modellazione del business**
  - comprensione organizzazione
  - definizione requisiti di sistema
- **Requisiti**
  - raffinamento e formalizzazione dei requisiti
  - modello dei casi d'uso
- **Analisi e progettazione**
  - formalizzazione tecnica dei requisiti
  - formalizzazione architettura tecnologica
- **Implementazione**
  - costruzione del software
- **Test**
  - verifica del livello di soddisfacimento dei requisiti
- **Rilascio agli utenti**
  - Packaging e distribuzione

Unified Process - UP
Workflow di supporto

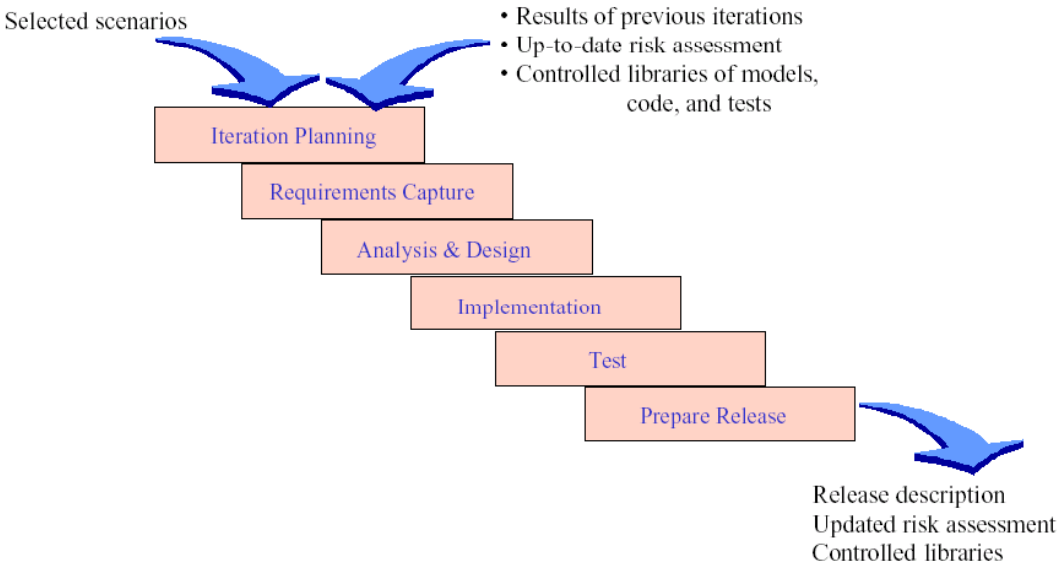
- **Gestione di progetto**
  - gestione dei rischi
  - pianificazione del progetto e della singola iterazione
  - controllo evoluzione del progetto iterativo
- **Gestione modifiche e configurazione**
  - gestione delle richieste di modifica
  - gestione configurazione
  - stato e metriche per il controllo del progetto
- **Gestione avviamento**
  - installazione del sw
  - Formazione e supporto utenti

- I singoli workflow hanno peso diverso nelle diverse fasi e nelle singole iterazioni all'interno di



Unified Process - UP

Il ciclo di vita di una iterazione



Unified Process - UP

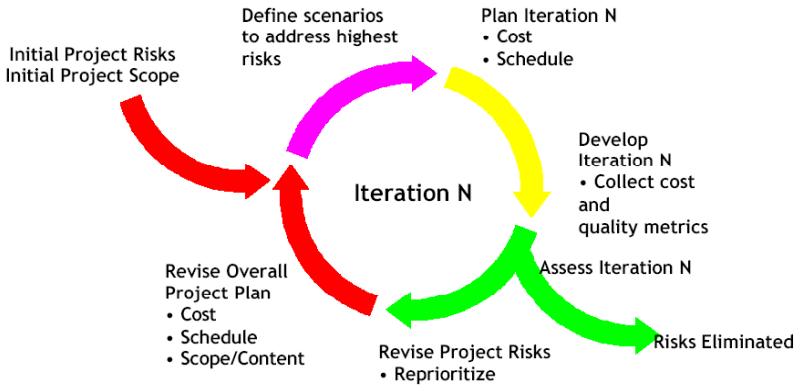
La prima interazione

- **E' solitamente la più difficile**
- Richiede che siano stabiliti l'ambiente di sviluppo e il team degli sviluppatori
- Problemi di integrazione di tool, di staffing, ecc.
- Team nuovi all'approccio iterativo sono solitamentetroppo ottimisti
- **Obiettivi modesti alla prima iterazione, altrimenti**
  - Probabili ritardi
  - Numero totale di iterazioni sarà ridotto
  - I benefici dell'approccio iterativo ridotti

Unified Process - UP

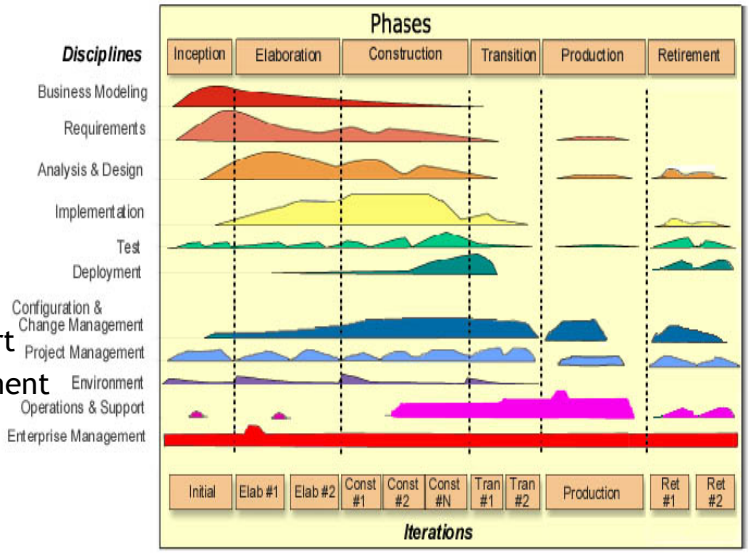
Riduzione del rischio

- Rischi = fattori variabili che influiscono negativamente sulla riuscita di un progetto
  - Di tipo tecnico: tecnologie di implementazione utilizzate, architettura del sistema, requisiti di performance
  - Di tipo non tecnico: disponibilità di risorse umane, competenze, date di rilascio del prodotto, budget
- In Unified Process (e in generale in un processo di sviluppo iterativo) ogni fase è guidata dall'obiettivo di ridurre i rischi più significativi



EUP – Enterprise Unified Process

- E' un'estensione di Unified Process e prevede in aggiunta
- Due nuove fasi
  - Produzione
  - Manutenzione
- Due nuove discipline
  - Operations & support
  - Enterprise management



Metodologie Agili – Agile Modeling - AM

- Nascono in alternativa alle metodologie tradizionali e si propongono come metodologie "leggere" caratterizzate da due aspetti principali:
  - **Adattivi più che predittivi**
    - Non cercano di di programmare lo sviluppo nel dettaglio e in modo da soddisfare tutte le specifiche, ma progettano programmi pensati per cambiare nel tempo
  - **People-oriented invece che process-oriented**
    - L'approccio prevede di adattare il processo di sviluppo alla natura dell'uomo... lo sviluppo software deve diventare un'attività piacevole per chi lo opera.
- Le metodologie agili fanno largo uso di **best practices**
  - Consuetudini affermate nello sviluppo del software che, anche se con caratteristiche diverse e intensità diverse, vengono adottate in tutti i processi di sviluppo
- Per alcuni è una "filosofia"

CMM - Capability Maturity Model

- Il SW-CMM (SW - Capability Maturity Model) definisce un modello dei processi di sviluppo del software e un insieme di regole per il loro miglioramento
  - http://www.sei.cmu.edu/cmm
- Con CMM si intende la **capacità di definire, controllare e misurare un processo software**
- E' stato sviluppato nel 1993 dal SEI, Software Engineering Institute della Carnegie Mellon University, Pittsburgh, USA, per conto del DOD (Dept. Of Defense, USA)

CMM
Concetti fondamentali

- Il concetto di **Capability**
  - Il modello CMM permette di valutare le organizzazioni che operano nel settore IT in funzione del loro livello di "maturità" e della "capability" dei processi che attuano
    - Per capability si intende, in termini di produzione industriale, la qualità attesa dei risultati di un processo
    - Dipende, sostanzialmente, dal livello di definizione, gestibilità, misurabilità, controllabilità ed effettiva implementazione raggiunto da uno specifico processo
- Il concetto di **Maturità**
  - La maturità di una organizzazione è la capacità di fornire prodotti di qualità attraverso la capacità dei processi che attua
    - In una organizzazione è quindi la capacità di controllare / prevedere / garantire la qualità dei prodotti in uscita dai processi
    - In funzione di questa capacità le organizzazioni possono essere posizionate su una scala ordinale da 0 a 5

CMM
I livelli

Level 1 / Initial	The software process is characterized as ad-hoc, and occasionally chaotic. Project schedules, budgets, functionality, and quality are generally unpredictable.
Level 2 / Repeatable	Basic project management processes are established to track cost, schedule, functionality, and quality.
Level 3 / Defined	The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the organization. All projects use an approved, tailored version of the organization's standard software process for developing and maintaining software.
Level 4 / Managed	Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled.
Level 5 / Optimizing	Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies.

PARAMETRIC ESTIMATING HANDBOOK, Second Edition (Spring 1999), DoD USA, http://www.ispa-cost.org