Ingegneria del Software

- Termine coniato nel corso di una conferenza NATO organizzata a Garmisch (Germania) nell'Ottobre del 1968

- Evoluzione del software (in seguito a quella nota come ***crisi del software***) concepita gia` nella seconda meta` degli anni 60

- Esigenza che lo sviluppo del software diventasse sempre piu` una ***disciplina ingegneristica***, con basi teoriche e metodologiche

- Progettare, costruire e manutenere sistemi software di grandi dimensioni

- E` un settore dell'ingegneria in evoluzione

Una definizione

- Nel glossario dell'IEEE ("IEEE Standard Glossary of Software Engineering"), l'***ingegneria del software*** e` definita come:

  applicazione di un approccio sistematico, disciplinato e quantificabile allo sviluppo, all'operativita` e alla manutenzione del software.

- Il ***software*** e` definito come:

  i programmi, le procedure, e l'eventuale documentazione associata e i dati relativi all'operativita` di un sistema di elaborazione.

- L'ingegneria del software e` la disciplina tecnologica e gestionale che riguarda la produzione sistematica e la manutenzione dei prodotti software che vengono sviluppati e modificati ***entro i tempi e i costi preventivati***

La dimensione economica

- L'economia di tutti i paesi industrializzati dipende dal software
- Sempre piu` sistemi sono sotto il controllo di sistemi software
- Inizi degli anni '80, circa 1 milione di persone addette al software negli USA (circa un terzo del totale mondiale) e costo del software circa il 2% del Prodotto Interno Lordo (PIL)
- Costi incrementati nel decennio di circa 12% l'anno, con una crescita del personale di circa il 7% l'anno
- Le spese per lo sviluppo del software sono una frazione significativa del PIL in tutti i paesi industrializzati

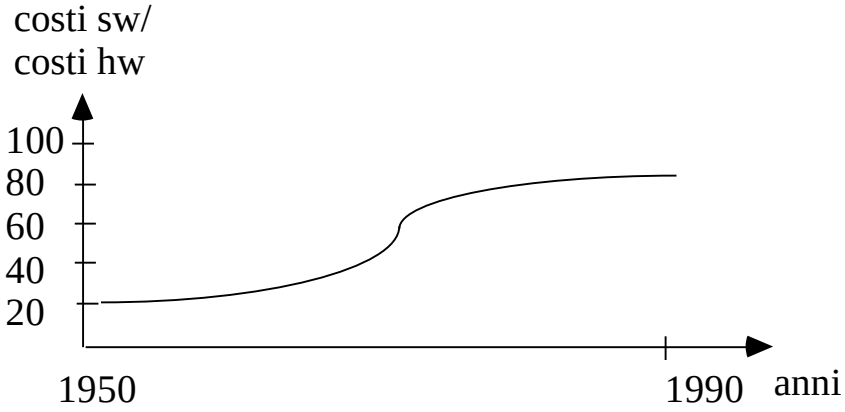
Prodotti software

- Un prodotto software e` un sistema consegnato ad un cliente con la documentazione che descrive come installare ed utilizzare il sistema

- Molto spesso, i prodotti software sono parti di sistemi dove sia la parte hardware che software sono venduti al cliente

- L'hardware ha avuto un guadagno di 6 ordini di grandezza nel rapporto prestazioni/costi nell'arco di 30 anni

- Il costo del software e` dominante sugli altri costi nello sviluppo di un sistema informatico



- Nella vita di un prodotto software la manutenzione costa molto di piu` dello sviluppo

- L'ingegneria del software si occupa dello sviluppo di sistemi di ***qualità*** con ***costi proporzionati***

Tipologie dei prodotti

- **Prodotti generici**

Sistemi “stand-alone” prodotti da un’azienda che sviluppa software ed immessi sul libero mercato

- **Prodotti personalizzati (“customized”)**

Sistemi commissionati da un particolare cliente. Il software viene quindi sviluppato con quel cliente come committente

- Fino agli anni ’80 la maggioranza dei sistemi sviluppati ricadeva nella seconda categoria

- Oggi maggiore spesa per i prodotti generici, ma i prodotti personalizzati sono quelli che richiedono il maggiore sforzo nello sviluppo

- L'attivita` di progettazione del software risulta prevalentemente “brain intensive” e dunque e` difficilmente meccanizzabile

- La complessita` di capire, descrivere, progettare un frammento di software e` di gran lunga superiore a quella corrispondente per qualunque altro prodotto

Attributi dei prodotti software

Caratteristiche presentate dal prodotto quando e` installato ed in uso

- **Manutenibilita`**

Deve essere possibile modificare il software in modo da soddisfare nuovi requisiti

- **Affidabilita`**

Nel caso di guasto, il software non deve produrre danni fisici od economici

- **Efficienza**

Il software non deve fare un uso indiscriminato di memoria e tempo di calcolo

- **Facilita` di utilizzo**

Il software deve essere corredato di una interfaccia utente e della documentazione appropriate

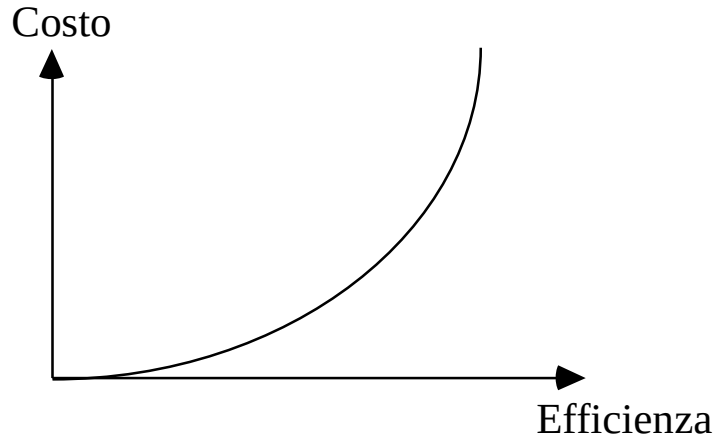
Attributi dei prodotti software

- Alcune delle caratteristiche ne includono altre

  Ad es., affidabilita` ☞ regolarita` di funzionamento
  protezione o sicurezza
  innocuita`

- L'importanza dell'una o dell'altra di tali caratteristiche varia (puo` variare) a seconda del settore applicativo

- In ogni caso, il costo del software aumenta esponenzialmente se e` richiesto un livello molto alto di una qualunque di tali caratteristiche



- L'ingegneria del software e` un corpus di teorie, metodi e strumenti – sia di tipo tecnologico che organizzativo – che consentano di produrre applicazioni con le caratteristiche desiderate

Affidabilita`, correttezza, robustezza

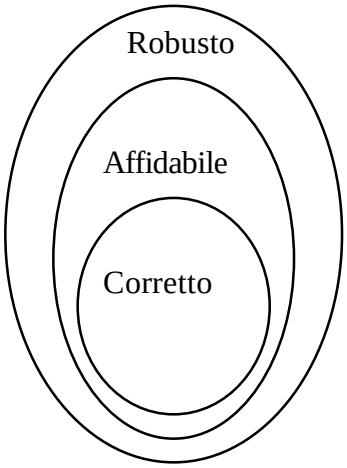
- Software **affidabile**, se i risultati calcolati, le elaborazioni effettuate, le azioni eventualmente eseguite producono gli effetti voluti o comunque con scostamenti tollerabili

- Software **corretto**, se data una definizione dei requisiti, il software li soddisfa

{Pre-condizioni} P *{Post-condizioni}*

- Software **robusto**, se si comporta in maniera accettabile anche in corrispondenza di situazioni non specificate nei requisiti

- Relazione tra i concetti di software corretto, affidabile e robusto



Protezione (sicurezza) e innocuita`

- ***Sistemi sicuri*** se proteggono l'accesso a informazioni, impedendo accessi non autorizzati, sia di natura involontaria che volontaria

- ***Sistemi innocui*** ("safe"), specie in connessione con sistemi che possono essere critici e pericolosi anche per la vita dell'uomo, sono sistemi che non entrano mai in uno stato in cui il livello di pericolo puo` essere intollerabile

Prestazioni

- Teoria della *complessita` del calcolo*, permette di valutare le prestazioni nel caso medio e nel caso peggiore, in termini asintotici rispetto ad alcune grandezze tipiche del programma in esame

- Esistono evidentemente altri modi per valutare le prestazioni di un sistema

- Ad esempio, con *software monitors* si misura il tempo di esecuzione o l'occupazione di memoria durante esecuzioni-campione del programma

- Oppure, si effettuano le misure non mediante esecuzioni reali, ma in *ambienti simulati*

- Un altro modo consiste nel costruire un modello del sistema sotto esame e nel dedurre alcune proprieta` prestazionali operando in maniera analitica sul modello

Fattori di qualita`

- Le *qualita` esterne* sono quelle percepibili da un osservatore esterno che esamina il processo o il prodotto come se fosse una scatola nera (*black-box*)

- Le *qualita` interne* sono quelle che possono essere osservate esaminando la struttura interna del processo o prodotto, come se questo fosse una scatola trasparente (*white-box*)

- Le seconde influenzano le prime, che sono quelle che ci interessa garantire

- Ad esempio, la caratteristica esterna di "affidabilita`" risulta influenzata dalla caratteristica di qualita` interna di "possibilita` di scoperta di errori da parte del compilatore"

- *Le qualita` interne sono un modo per realizzare le qualita` esterne*

- Molti dei fattori di qualita` del software interessanti sono definibili soltanto in maniera intuitiva e poco rigorosa, per cui, ad esempio, l'obiettivo di fornire meccanismi precisi di ***misura*** non e` realisticamente raggiungibile (ad esempio, per l'affidabilita`)

Il processo di sviluppo del software

- E` l'insieme di attivita` per sviluppare un sistema software

- Specifica

Si definiscono le funzionalita` richieste ed i vincoli

- Progetto (sviluppo)

Viene prodotto il software che soddisfi le specifiche

- Validazione

Per assicurare che il software soddisfi le richieste del cliente

- Modifica (evoluzione)

Il software evolve per soddisfare nuove necessita` del cliente (il costo di tale fase e` oltre il 60% del totale)

- Tali attivita` sono svolte da ingegneri del software, in alcuni casi con l'ausilio di strumenti CASE (Computer-Aided Software Engineering)

- La struttura di tale processo produttivo puo` avere una forte influenza sulla qualita` del prodotto finale e sul costo

Caratteristiche del processo di sviluppo del software

- ***Facilita` di comprensione*** (definizione esplicita del processo)
- ***Visibilita`*** (produzione di documenti ad intervalli regolari)
- ***Supportabilita`*** (utilizzabilita` di strumenti CASE)
- ***Accettabilita`***
- ***Affidabilita`*** (il processo di sviluppo e` definito in modo tale da evitare errori)
- ***Robustezza*** (il processo prosegue anche al verificarsi di problemi imprevisti)
- ***Capacita` di modifica*** (il processo evolve nel caso si verifichino modifiche a livello organizzativo)
- ***Rapidita`*** (velocita` con cui si consegna il prodotto date le specifiche)

  Non e` possibile ottimizzare simultaneamente tutte queste caratteristiche (ad esempio, rapidita` *vs* visibilita`)

Modelli del processo di sviluppo del software

- Sviluppo a cascata

Le fasi (specifica, progetto, realizzazione, testing, etc.) sono in cascata tra loro con retroazione

- Sviluppo evolutivo

Le fasi di specifica, sviluppo e validazione sono frammischiate tra loro

Partendo da una specifica astratta si sviluppa un primo prototipo che puo` poi essere raffinato

- Trasformazioni formali

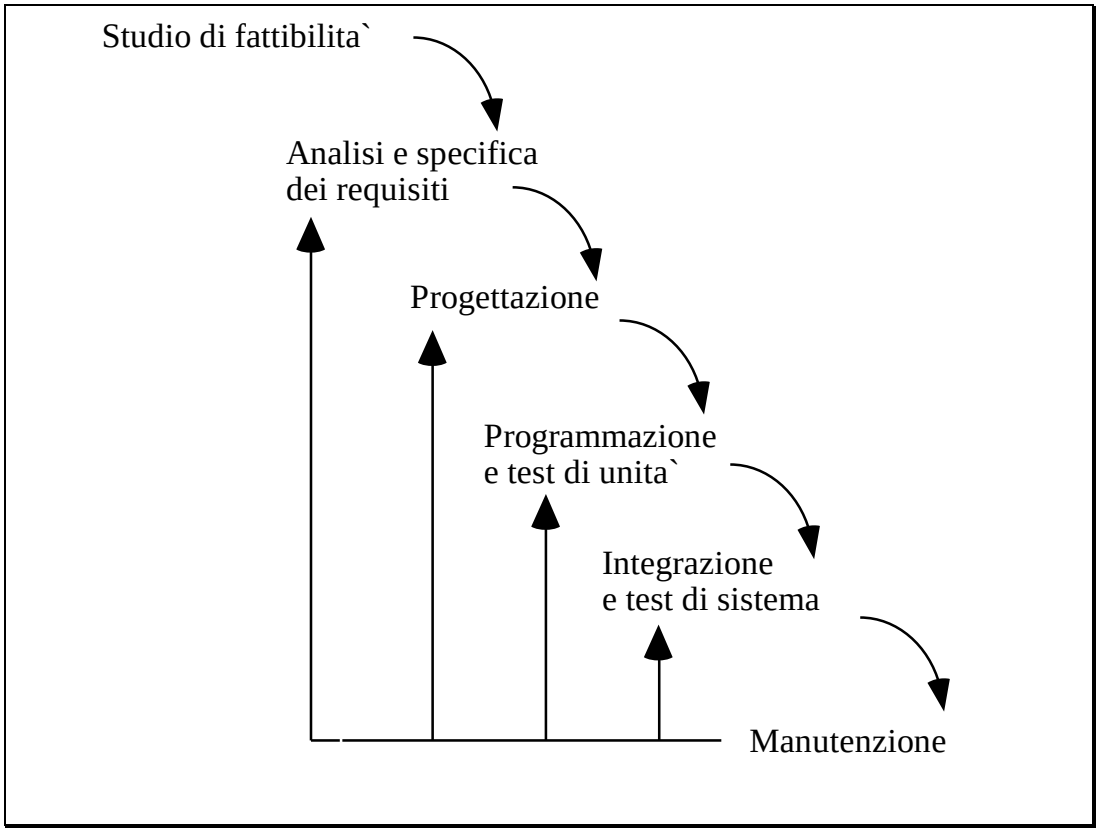
Si trasforma una specifica formale, con metodi matematici, in un programma

- Sviluppo per assemblaggio di componenti riutilizzabili

Integrazioni di componenti software gia` esistenti

Il modello a cascata

- Detto anche *ciclo di vita del software*



- Le uscite intermedie che una fase produce come ingresso per la fase successiva sono i cosiddetti *semilavorati* del processo:
  - Documentazione di tipo cartaceo, in linguaggio naturale
  - Codice dei singoli moduli e il sistema complessivo

Il modello a cascata (cont.)

Nel modello di ciclo di vita a cascata, oltre a definire le fasi, ci si preoccupa anche di definire

- **semilavorati**

  fondamentale al fine di garantire che ci possa poi essere un'attivita` di controllo della qualita` dei semilavorati

- **date** alle quali questi devono essere prodotti i semilavorati

  fondamentale al fine di certificare l'avanzamento del processo secondo il piano stabilito

Perche` questo procedimento funzioni correttamente e` evidentemente necessario che il modello sia davvero "a cascata", e cioe` che non esistano retroazioni

L'intero modello a cascata suggerisce una ***distinzione*** arbitraria tra ***fasi***, il cui contorno non puo` certo essere definito in maniera assoluta e rigorosa

Altri modelli (ad esempio, modello a spirale)

Fasi del modello a cascata

- Studio di fattibilita`

Valutazione preliminare dei costi e dei benefici di un'applicazione, per stabilire se si debba avviarne lo sviluppo, quali siano le alternative possibili, quali le scelte piu` ragionevoli, e quali le risorse finanziarie e umane necessarie per l'attuazione del progetto

Il contenuto di questa fase risulta largamente variabile da caso a caso, in funzione del tipo di prodotto e dei ruoli del committente e del produttore dell'applicazione

Il risultato della fase di studio di fattibilita` e` un documento che dovrebbe contenere le seguenti informazioni:

- Definizione preliminare del problema
- Possibili scenari che illustrino eventuali diverse strategie di soluzione
- Costi, tempi e modalita` di sviluppo per le diverse alternative tra cui scegliere

Il committente puo` allocare risorse finanziarie perche` venga condotto uno studio di fattibilita` completo

- Analisi e specifica dei requisiti

Si stabiliscono funzionalita` (*requisiti funzionali*), vincoli e obiettivi consultando gli utenti

Il risultato principale della fase di analisi e specifica dei requisiti e` il *Documento di Specifica dei Requisiti* (DSR) che deve essere approvato dal committente

La definizione deve essere comprensibile sia agli utenti che agli sviluppatori

Il documento costituisce un riferimento per l'attivita` di sviluppo del prodotto

Un modo possibile di descrivere i requisiti funzionali consiste nel fornire una versione iniziale del *Manuale Utente* (MU)

In questa fase viene anche definito il *Piano di Test di Sistema* (PTS) che descrive le modalita` con cui, al termine dello sviluppo, nella fase di integrazione, si possa verificare il sistema sviluppato rispetto ai requisiti fissati

Anche questo documento andrebbe sottoscritto dal committente

- Progettazione

Si definisce l'***architettura generale*** (hardware e software) del sistema

Si descrivono le funzioni che il sistema deve svolgere, ciascuna delle quali verra` trasformata in uno o piu` programmi eseguibili

L'*architettura software* puo` essere composta da moduli, evidenziando quali siano le funzionalita` offerte dai diversi moduli e le relazioni tra i moduli

Il risultato dell'attivita` di progettazione e` il *Documento di Specifiche di Progetto* (DSP) nel quale la definizione dell'architettura software puo` anche essere data in maniera rigorosa, o addirittura formale, usando opportuni *linguaggi di specifica di progetto*

- Realizzazione e test delle unita`

Il progetto viene realizzato come insieme di programmi o unita` di programmi (moduli) nel linguaggio di programmazione scelto

Il testing delle unita` serve per verificare che ciascuna soddisfi le specifiche richieste

La distinzione tra progettazione e programmazione tende a diventare sempre piu` sfumata

- Integrazione e test del sistema

Si integrano le singole unita` e/o i programmi tra loro e si esegue il test del sistema completo per assicurarsi che le specifiche siano soddisfatte

- *alfa test* quando il sistema e` rilasciato per l'uso, ma all'interno dell'organizzazione del produttore
- *beta test* quando si ha un rilascio controllato a pochi e selezionati utenti del prodotto

Il sistema viene consegnato al cliente

- Utilizzo e manutenzione

E` la fase piu` lunga del ciclo di vita di un prodotto software (oltre il 50% dei costi complessivi del ciclo di vita)

Il sistema viene utilizzato

La manutenzione comporta la correzione degli errori che non erano stati scoperti nelle fasi precedenti, migliorando la realizzazione delle unita` del sistema ed aumentando i servizi forniti man mano che si richiedono nuovi requisiti

Il costo della manutenzione e` ripartito in:

- 20% manutenzione correttiva
- 20% manutenzione adattativa
- oltre 50 % manutenzione perfettiva

Spesso il software non e` stato progettato per essere modificato facilmente

Si apportano modifiche intervenendo direttamente sui programmi, senza modificare, se e` il caso, la documentazione di progetto e di test, la specifica dei requisiti, etc.

Un problema molto sentito dai produttori di software e` quello di “recuperare” le applicazioni esistenti

La *ri-ingegnerizzazione* del software (*reverse engineering*) consiste nella possibilita` di riportare software ormai poco strutturato e non documentato in uno stato in cui sia possibile poi ripartire in modo sistematico nella manutenzione

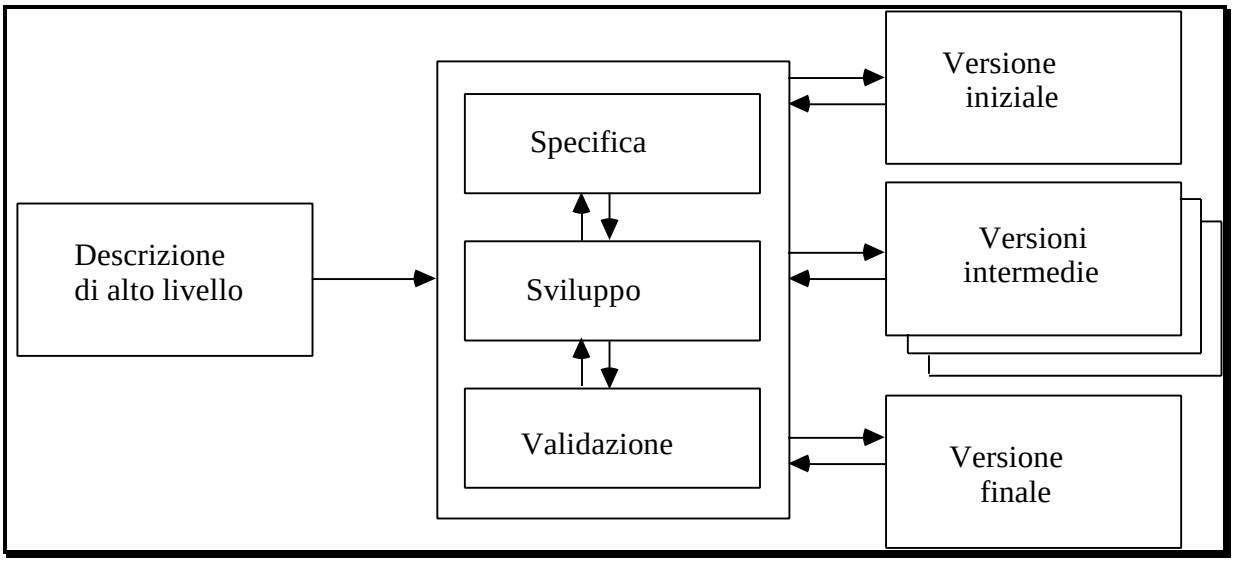
Ad esempio, produzione di diagrammi E-R da programmi COBOL

Attivita` ortogonali nel modello a cascata

- Attivita` che risultano sostanzialmente ortogonali all'intero ciclo di vita
  - *documentazione* (gran parte dei semilavorati fluiscono da una fase all'altra)
  - *verifica* (o *controllo della qualita`*), accompagna l'intero ciclo di vita
  - *gestione,* che riguarda sia il processo che i suoi prodotti
- Gestione del processo, definizione delle attivita` e delle fasi attraverso cui deve evolvere il modello a cascata, dimensionamento delle risorse necessarie per lo sviluppo e la manutenzione e gestione delle persone che vengono a far parte del progetto
- Gestione dei prodotti, mantenere l'insieme dei semilavorati del processo in uno stato consistente

Modelli evolutivi

- Basati sull'idea di sviluppare un primo prototipo da sottoporre al committente e da raffinare successivamente



- Alternativa interessante in tutti i casi in cui lo sviluppo dell'applicazione parte inizialmente con requisiti non perfettamente noti o instabili

Prototipo

Modello approssimato dell'applicazione, il cui obiettivo e` di essere mostrato al committente – o usato da questi – al fine di ottenere un'indicazione su quanto il prototipo colga i reali fabbisogni

Deve essere sviluppabile in tempi brevi ed economicamente vantaggiosi

Modelli evolutivi (cont.)

- Due tipi di sviluppo evolutivo:

- **Prototipazione "throw-away"**

Il prototipo che si realizza e` del tipo *usa e getta*

L'obiettivo e` comprendere le richieste del cliente e quindi sviluppare una migliore definizione dei requisiti del sistema

Il prototipo si concentra sulle parti che sono mal comprese con l'obiettivo di contribuire a chiarire i requisiti

Modelli evolutivi (cont.)

- **Programmazione esplorativa**

Il prototipo si puo` trasformare progressivamente nel prodotto

L'obiettivo del processo di sviluppo e` lavorare in stretto contatto con il cliente per indagarne i requisiti e giungere ad un *prodotto finale*

Si sviluppano le parti del sistema che sono ben chiare (requisiti ben compresi)

Successivamente si aggiungono nuove parti/funzionalita` come proposto dal cliente

Modelli evolutivi: problemi

- Il processo di sviluppo non e` visibile (ad es., documentazione non disponibile)
- Il sistema sviluppato e` poco strutturato (modifiche frequenti)
- E` richiesta una particolare abilita` nella programmazione (team ristretto)

Per questi motivi, questi modelli di sviluppo sono spesso applicati solo in fase prototipale, per giungere in tempi brevi ad un primo prodotto e validare le specifiche

Modelli evolutivi: appicabilita`

- Sistemi piccoli o di dimensioni medie
- Sistemi che avranno breve durata
- Parti di sistemi piu` grandi (ad es., interfacce utente)

Il modello trasformazionale

- Sviluppo di software come progressione di passi in cui una *descrizione formale* (di solito dei requisiti iniziali) viene trasformata in una descrizione formale meno astratta ed eseguibile in modo piu` efficiente

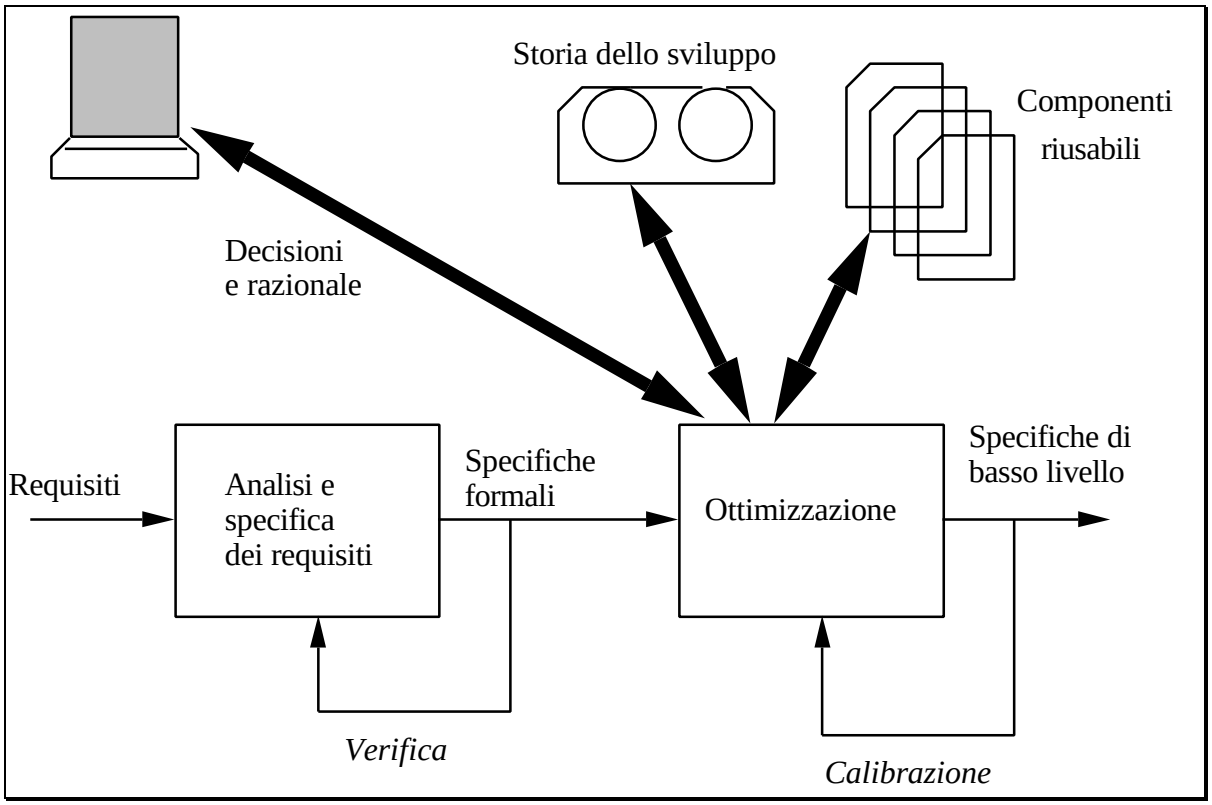
- Trasformazioni applicate fino ad una descrizione in un linguaggio di programmazione

- Basata su due concetti fondamentali:

  - **prototipazione** di tipo evolutivo
    deriva automaticamente
    dalla eseguibilita` delle
    specifiche formali

  - **formalizzazione** rende possibile l'esecuzione
    delle specifiche

Il modello trasformazionale



- ***Requisiti*** dell'applicazione descritti in una ***notazione formale***, che puo` essere verificata se le specifiche formali sono eseguibili

- Un processo di ottimizzazione (guidato dal progettista) genera progressivamente codice eseguibile di piu` basso livello

- Il processo di trasformazione e` anche guidato da un catalogo di componenti riusabili (moduli "prefabbricati", possibili passi di trasformazione)

- Richiede tecnologia avanzata ed esperti (***alto rischio***)

Analisi dei rischi

- Quale modello scegliere per il processo di sviluppo?
- La scelta deve dipendere da una valutazione dei rischi che le diverse alternative presentano
- Obiettivo, ***minimizzazione dei rischi***

Rischio inerente un'attivita`

E` una valutazione (misura) dell'incertezza che si ha sul risultato finale dell'attivita`

- La scelta di un modello o dell'altro e` dettata da un'analisi dei rischi che si incontreranno poi nel processo

Analisi dei rischi (cont.)

- **Modello a cascata**

Alto rischio per nuovi sistemi con requisiti non stabili

Basso rischio per sviluppo di sistemi ben specificati e con tecnologia assestata

- **Prototipazione**

Alto rischio per mancanza di visibilita`

Basso rischio per sviluppo di nuove applicazioni

- **Modello trasformazionale**

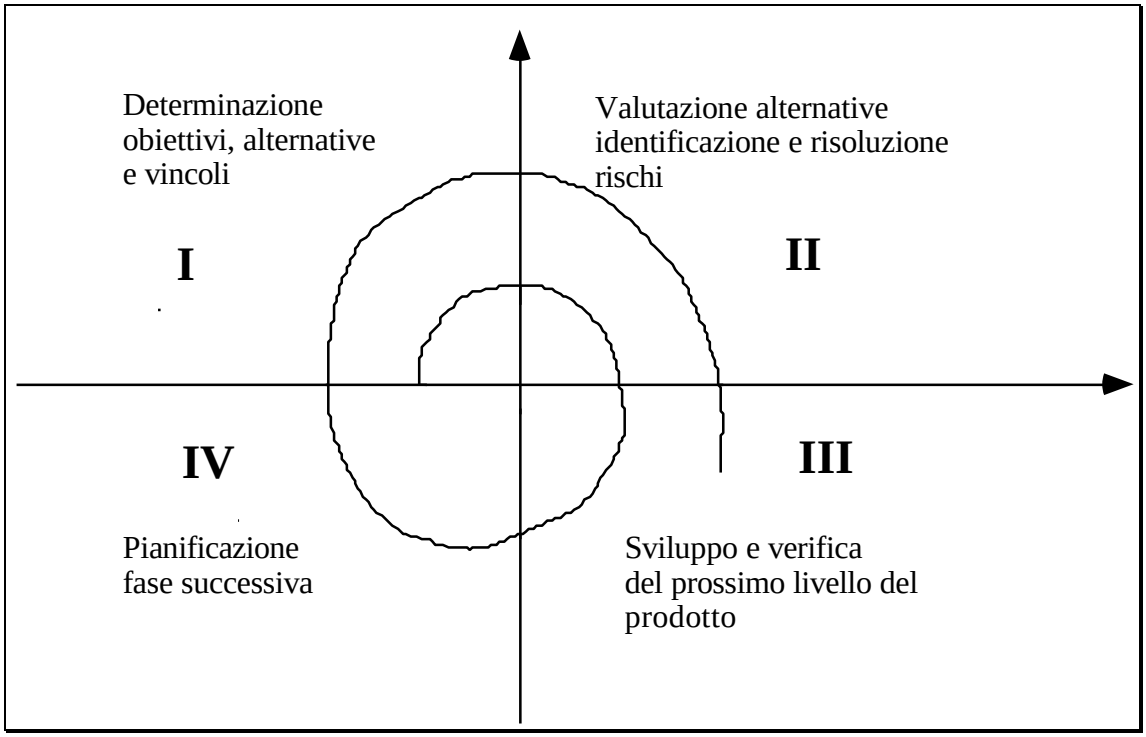
Alto rischio per necessita` di tecnologie avanzate ed alta competenza

Modelli ibridi

- Sistemi composti di sotto-sistemi
- Lo stesso modello di sviluppo non e` adottabile per ogni sotto-sistema
- Prototipazione, per sotto-sistemi con specifiche ad alto rischio
- Modello a cascata, per sotto-sistemi con specifiche ben definite

Modello a spirale di Boehm

- Proposto per supportare l'***analisi dei rischi***
- Modello di tipo ciclico, dove il raggio della spirale rappresenta il costo accumulato durante lo svolgimento del progetto



- ***Ogni ciclo*** della spirale rappresenta ***una fase***
  - il piu` interno puo` essere lo studio di fattibilita`
  - il successivo la definizione dei requisiti
  - il successivo ancora la progettazione, etc.

- Non sono definite fasi a priori

- Ogni ciclo della spirale passa attraverso i quadranti del piano che rappresentano i seguenti passi logici:

- **I: determinazione di obiettivi, alternative e vincoli**

- **II: valutazione di alternative, identificazione e risoluzione di rischi**

ad es., sviluppo di un prototipo per validare i requisiti;

- **III: sviluppo e verifica del prossimo livello del prodotto**

modello evolutivo, se i rischi riguardanti l'interfaccia utente sono dominanti; a cascata se e` l'integrabilita` del sistema il rischio maggiore; trasforma-zionale, se la sicurezza e` piu` importante

- **IV: pianificazione della fase successiva**

si decide se continuare con un altro ciclo della spirale

Modello a spirale di Boehm (cont.)

- Costituisce un ***meta-modello*** dei processi software, cioe` un modello per descrivere modelli
- Non e` detto che per un ciclo della spirale si adotti un solo modello di sviluppo
- Puo` descrivere uno sviluppo incrementale, in cui ogni incremento corrisponde a un ciclo di spirale
- Puo` descrivere il modello a cascata (quadranti I e II corrispondenti alla fase di studio di fattibilita` e alla pianificazione del progetto; quadrante III corrisponde al ciclo produttivo)
- Differisce dagli altri modelli perche` considera esplicitamente il fattore ***rischio***

Modello a spirale di Boehm (cont.)

- Boehm suggerisce di considerare, per ciascun ciclo, le seguenti voci:
    - Obiettivi
    - Vincoli
    - Alternative
    - Rischi
    - Soluzioni per i rischi
    - Risultati
    - Piani
    - Decisioni

Esempio: aumento qualita` prodotti

Obiettivi

- Migliorare significativamente la qualita` dei prodotti sw

Vincoli

- Entro 3 anni
- Con investimenti contenuti
- Senza cambiare radicalmente gli standard della compagnia

Alternative

- Riutilizzare software esistente certificato
- Introdurre specifiche formali e tecniche di verifica formale
- Investire in strumenti per il testing e la validazione

Rischi

- Miglioramenti della qualita` impossibili senza costi significativi
- Introduzione di nuovi metodi potrebbe provocare le dimissioni del personale esistente

Soluzione dei rischi

- Panoramica della letteratura
- Progetto pilota
- Panoramica dei componenti sw riusabili
- Stati dell'arte degli strumenti disponibili
- Seminari di aggiornamento per il personale

Risultati

- Esperienza su metodi formali limitata - difficile quantificare i miglioramenti
- Strumenti di supporto disponibili limitati
- Componenti riusabili disponibili, ma poca esperienza e strumenti

Piani

- Esplorare le opzioni per riutilizzo in maggiore dettaglio
- Sviluppare prototipi di sistemi di aiuto al riutilizzo
- Esplorare schemi di certificazione di componenti

Decisione

- Finanziare una fase di studio di 18 mesi