Ingegneria del Software T

Introduzione

Argomenti

- Evoluzione dello sviluppo del software
- Prodotto software
- Processo di sviluppo del software
  - Attività
  - Modelli
  - Prototipi
  - Linguaggi di modellazione – UML
- Fattori di qualità del software

Evoluzione dello sviluppo del software

- Seconda metà degli anni '60
  l'avvento dei calcolatori di terza generazione (circuiti integrati), rende possibili applicazioni considerate in precedenza non realizzabili

- Si rende necessario
  - Progettare
  - Realizzare
  - Effettuare la manutenzione (*manutenere*)

  sistemi software di grandi dimensioni

Evoluzione dello sviluppo del software

- Nel settore dell'hardware, netto, progressivo:
  - aumento delle prestazioni
  - riduzione dei costi

Evoluzione dello sviluppo del software

- Nel settore del software:
  - Pochissimi i progetti terminati nei tempi stabiliti
  - Rari i progetti che non sfondano il budget
  - Sistemi generalmente pieni di difetti
  - Sistemi così rigidamente strutturati che è praticamente impossibile apportare significativi cambiamenti senza doverli riprogettare totalmente

- "*Crisi del software*"

Evoluzione dello sviluppo del software

Anni	Base tecnologica	Problema	Rivoluzione software
1950	primi calcolatori in serie	**produttività**	Assembler
1960	calcolatori mono-utente in batch	**produttività**	linguaggi di alto livello (Fortran, Cobol, Algol)
1970	tecnologia dei compilatori	**complessità**	programmazione strutturata (Pascal, C)
1980	calcolatori sempre più potenti	**complessità**	programmazione modulare ADT (Ada, Modula-2)
1990	workstation, reti, interfacce grafiche,...	**enorme aumento della complessità**	programmazione orientata agli oggetti (Smalltalk, C++, Eiffel, CLOS, Object Pascal, Java, C#, …)

Evoluzione dello sviluppo del software

- Da lavoro individuale e creativo (software come arte)
- A lavoro di piccoli gruppi specializzati (livello artigianale)
- Sino a lavoro di gruppi anche di grosse dimensioni in cui è necessaria un'opera di pianificazione e coordinamento (livello industriale)

Evoluzione dello sviluppo del software

- Lo sviluppo del software deve evolvere in una disciplina ingegneristica, con basi
  - Teoriche
  - Metodologiche

- Nel 1968, durante una conferenza sull'evoluzione del software, viene coniato il termine "***ingegneria del software***"

Ingegneria del Software

- Una disciplina
  - tecnologica
  - gestionale

  che permette di affrontare in modo
  - sistematico
  - quantificabile (in termini di tempi e costi)

  lo sviluppo e la manutenzione
  dei prodotti software

Prodotto Software

- Codice che, quando eseguito, svolge una funzione prestabilita con prestazioni prestabilite
- Strutture dati mediante le quali il codice tratta adeguatamente le informazioni
- Documenti che descrivono le operazioni e l'uso del prodotto software:
  - documentazione tecnica
  - manualistica di installazione
  - manualistica di utilizzo
  - …

Prodotto Software

- Può essere realizzato
  - per un particolare cliente
  - per il mercato in generale
- Può fare parte di un sistema di elaborazione che comprende
  - sia la parte software
  - sia la parte hardware

Prodotto Software

- Si sviluppa, non si fabbrica
  - Cliente
  - Sviluppatore/i
  - Processo di sviluppo
  - Linguaggio e strumenti di modellazione
- Non “si consuma”…

- Struttura
- Caratteristiche

Sviluppo del software

- Esistono molti esempi di
  - Progetti falliti
  - Tempi e costi non rispettati
  - Soluzioni non corrette
  - Sistemi non gestibili
- Il successo di un progetto
  - Non può essere garantito
  - È determinato
    - Principalmente da fattori umani
    - Solo secondariamente da scelte tecnologiche

Principali cause di fallimento

- Necessità del cliente comprese male o in modo insufficiente
- Requisiti del cliente troppo instabili
- Mancanza di cooperazione tra cliente e sviluppatori
- Attese non realistiche da parte del cliente
- Mancanza di benefici per il cliente
- Fornitura insufficiente di risorse da parte del cliente

Principali cause di fallimento

- Sviluppatori non all'altezza delle attività in cui sono coinvolti
- Capacità e conoscenza degli sviluppatori sono i fattori che maggiormente contribuiscono
  - alla qualità del software
  - alla produttività
- Buoni sviluppatori possono fornire una soluzione, ma ottimi sviluppatori possono fornire
  - soluzioni migliori
  - più velocemente
  - a minor costo

Obiettivo dell'ingegneria del software

- Mettere in grado lo sviluppatore di affrontare la **complessità** dei grossi progetti
  - Sviluppare prodotti con le caratteristiche di qualità desiderate
  - Gestire le risorse (specie quelle umane) in modo produttivo
  - Rispettare i vincoli economici e di tempo specificati

Ingegneria tradizionale

- Il problema dello sviluppo del software è un problema di gestione della complessità
- In altri settori, l'uomo ha imparato bene a progettare e costruire sistemi complessi, come un edificio, un'automobile, un calcolatore, un impianto industriale
- Quali sono i principi usati nell'ingegneria tradizionale per affrontare la complessità?

Principi usati nell'ingegneria tradizionale per affrontare la complessità

- Suddivisione del progetto in sottoprogetti (moduli), e così via sino ad avere moduli facilmente progettabili
- Utilizzo di parti e sottosistemi già pronti
  - Sviluppati in casa
  - Comprati da fornitori esterni
- Standardizzazione dei componenti in modo che possano essere facilmente collegabili e intercambiabili

Principi usati nell'ingegneria tradizionale per affrontare la complessità

- Utilizzo del calcolatore come ausilio per
  - La progettazione
  - L'esecuzione di compiti ripetitivi
  - I calcoli

- Lo sviluppo di un progetto non è lasciato al caso o all'estro di pochi progettisti, ma è un'attività in larga misura sistematica

Processo di sviluppo del software

- Insieme strutturato di attività che regolano lo sviluppo di un prodotto software
- La struttura del processo di sviluppo può avere una forte influenza
  - sulla qualità
  - sul costo
  - sui tempi di realizzazione

  del prodotto

Processo di sviluppo del software

- Stabilisce un ordine di esecuzione delle attività
- Specifica quali elaborati devono essere forniti e quando
- Assegna le attività agli sviluppatori
- Fornisce criteri per
  - monitorare il progresso dello sviluppo
  - misurarne i risultati
  - pianificare i progetti futuri
- Copre l'intero ciclo di vita del software

Fasi del processo di sviluppo del software

1. Studio di fattibilità
2. Analisi dei requisiti
3. Specifica dei requisiti
4. Progettazione
5. Realizzazione e collaudo dei moduli
6. Integrazione e collaudo del sistema
7. Installazione e *training*
8. Utilizzo e manutenzione

Studio di fattibilità

- Valutazione preliminare dei costi e dei benefici per stabilire se si debba avviare lo sviluppo dell'applicazione
  - Alternative possibili
  - Scelte più ragionevoli
  - 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

Studio di fattibilità

- Il risultato della fase di studio di fattibilità è un documento che dovrebbe contenere le seguenti informazioni:
  - definizione preliminare del problema
  - possibili scenari che illustrino eventuali diverse strategie di soluzione
  - costi, tempi e modalità di sviluppo per le diverse alternative
- Il committente può allocare risorse finanziarie perché venga condotto uno studio di fattibilità completo

Processo di sviluppo del software

Analisi dei requisiti

- Un requisito è la descrizione
  - di un comportamento del sistema (**requisito funzionale**)
  - di un vincolo (**requisito non funzionale**)
    - sul comportamento del sistema
    - sullo sviluppo del sistema

Analisi dei requisiti

- Applicazione per la gestione di uno sportello tipo bancomat:
  - Requisiti funzionali:
    - Il sistema dovrà validare la tessera inserita dal cliente
    - Il sistema dovrà validare il PIN inserito dal cliente
    - Il sistema dovrà erogare non più di 500 € alla stessa tessera nell'arco delle 24 ore
    - …
  - Requisiti non funzionali:
    - Il sistema dovrà essere scritto in C++
    - Il sistema dovrà validare la tessera entro 3 secondi
    - Il sistema dovrà validare il PIN entro 3 secondi
    - …

Processo di sviluppo del software

Analisi dei requisiti

- Si stabiliscono
  - funzionalità (requisiti funzionali)
  - vincoli (requisiti non funzionali)
  - obiettivi

  consultando gli utenti (analisi congiunta)

  "*Knowledge of what users really want often is the single most important factor in the failure or success of a software project. It's also one of the most neglected factors*" - Johnson, Skoglund and Wisniewsky

Processo di sviluppo del software

Specifica dei requisiti

- Formalizzazione dei requisiti
  - Validazione delle specifiche
  - Evoluzione delle specifiche

- Il risultato principale è il Documento di Specifica dei Requisiti (DSR)
  - elenca tutti i requisiti che il prodotto software deve soddisfare
  - è un contratto tra sviluppatori e cliente per la fornitura del prodotto software

Specifica dei requisiti

- Il DSR costituisce un riferimento per l'attività di sviluppo del prodotto, pertanto deve essere preciso, completo e consistente – inadeguatezza del linguaggio naturale
- Il DSR definisce
  - quali funzionalità deve fornire il sistema
  - NON come sono realizzate le funzionalità

Specifica dei requisiti

- Un ulteriore modo di descrivere i requisiti funzionali consiste nel fornire al committente una versione iniziale del Manuale Utente
- Viene anche definito il Piano di Test di Sistema che descrive le modalità con cui, al termine dello sviluppo, nella fase di integrazione, si può verificare il rispetto dei requisiti da parte del sistema sviluppato
- Tutti i documenti dovrebbero essere approvati e sottoscritti dal committente
- Stiamo realizzando il prodotto desiderato?

Processo di sviluppo del software

Progettazione

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

- Si decompone l'architettura software in uno o più programmi eseguibili
- Per ogni programma, si descrivono:
  - le funzioni che svolge
  - le relazioni con gli altri programmi

- Si decompone ogni programma eseguibile in più moduli
- Per ogni modulo, si descrivono:
  - le funzioni che svolge
  - le relazioni con gli altri moduli

Processo di sviluppo del software

Progettazione

- Progettazione *object-based*
  - Tipi di dati astratti
  - Incapsulamento
- Progettazione *object-oriented*
  - Ereditarietà
  - Polimorfismo
- Progettazione generica rispetto ai tipi
  - *Template* in C++
  - Generici in Java e C#
- Gestione delle eccezioni
- Gestione degli eventi
- *Design pattern*

Processo di sviluppo del software

Progettazione

- Architettura del sistema
  - Modello *repository*
  - Modello *client-server*
  - Modello *abstract-machine*
- Controllo del sistema
  - Controllo centralizzato
  - Sistemi *event-driven*
- Concorrenza
- Progetto di interfacce utente
- Progetto di basi di dati

Progettazione

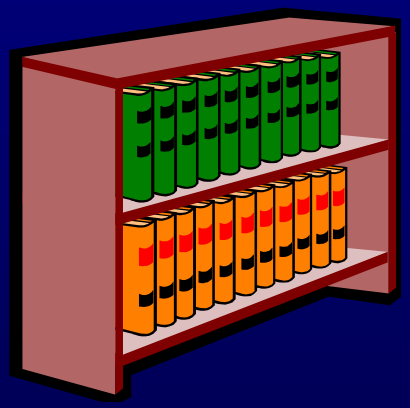
- Risultato: documento di specifiche di progetto nel quale la definizione dell'architettura software può anche essere data in maniera rigorosa, o addirittura formale, usando opportuni linguaggi di specifica di progetto

- Stiamo realizzando correttamente il prodotto?

Realizzazione e collaudo dei moduli

- Il progetto viene realizzato come insieme di programmi e/o moduli
  nel linguaggio di programmazione scelto
  (o nei linguaggi di programmazione scelti)

"*The fastest line of code to develop is line of code you don't have to write*" - Jeff Tash



- Gestione delle versioni del software

Processo di sviluppo del software

Realizzazione e collaudo dei moduli

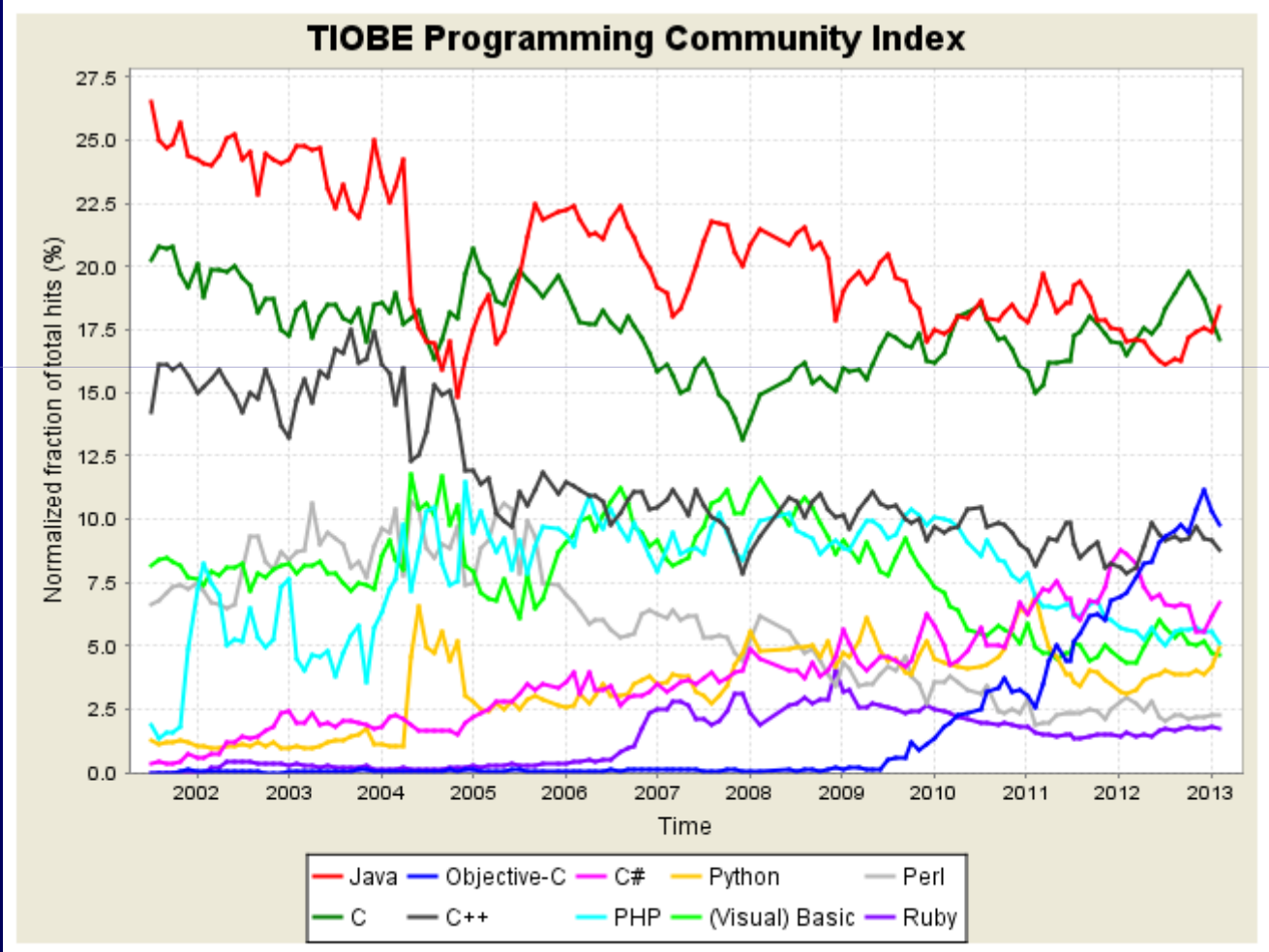
- Verifica del software
  - Analisi statica
  - Analisi dinamica
  - *Debugging*

- Collaudo dei moduli (unit testing)
  verifica che un modulo soddisfi le specifiche di progetto

Processo di sviluppo del software

Tecnologie e Linguaggi

- C / C++
- Java
- C#



Processo di sviluppo del software

Tecnologie e Linguaggi

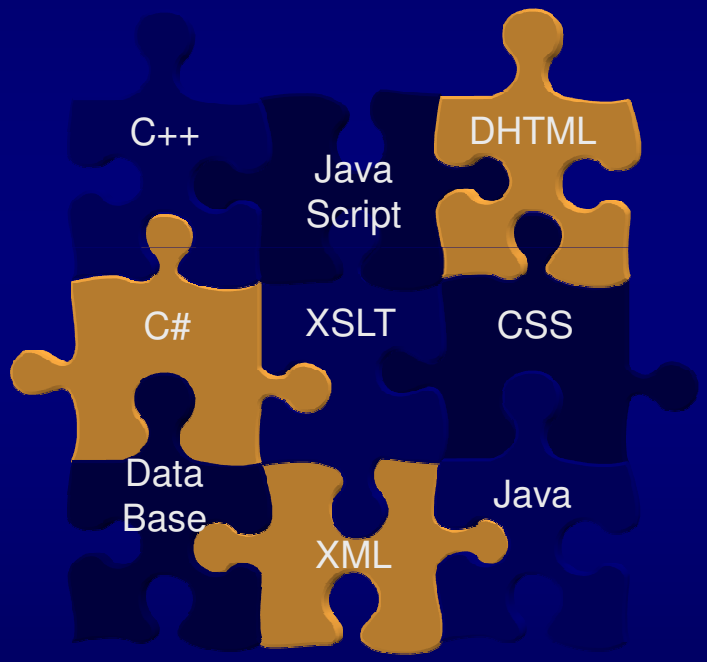
- HTML – *Hypertext Markup Language*
- DHTML – *Dynamic* HTML
- CSS – *Cascading Style Sheets*
- Javascript

- XML – *eXtensible Markup Language*
- DTD – *Document Type Definition*
- XSD – XML *Schema Definition*
- XSL – *eXtensible Stylesheet Language*
- XSLT – XSL *for Transformations*

- COM, COM++, DCOM, CORBA, EJB
- .NET, ADO, LinQ, XAML, WPF, WF, WCF
- DBMS, SQL, ORM

Processo di sviluppo del software

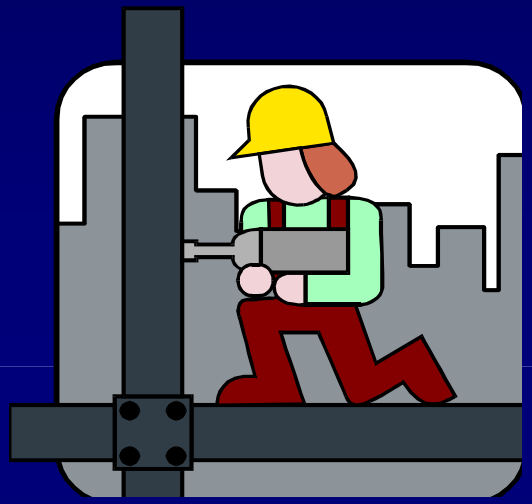
Tecnologie e Linguaggi



Processo di sviluppo del software

Integrazione e collaudo del sistema

- Si integrano i singoli moduli e/o programmi tra loro e si esegue il test del sistema completo per assicurarsi che le specifiche siano soddisfatte



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

Controllo della qualità

- Uso di programmi di test
- Sollecitazione dei programmi (condizioni limite)
- Controllo dell'utilizzo di standard prestabiliti
- …

Installazione e *training*

- *Deployment*

  Il sistema software viene:

  - consegnato al cliente
  - installato
  - messo in funzione

- *Training*

Processo di sviluppo del software

Utilizzo e manutenzione

- Il sistema viene utilizzato…
- Fase di manutenzione
  è la fase più lunga del ciclo di vita di un prodotto software
  50-80% dei costi complessivi

Processo di sviluppo del software

Utilizzo e manutenzione

- Manutenzione correttiva
  correzione degli errori che non sono stati scoperti nelle fasi precedenti
- Manutenzione adattativa
  aumento dei servizi forniti dal sistema in seguito alla definizione di nuovi requisiti
- Manutenzione perfettiva
  miglioramento delle caratteristiche delle unità del sistema

Utilizzo e manutenzione

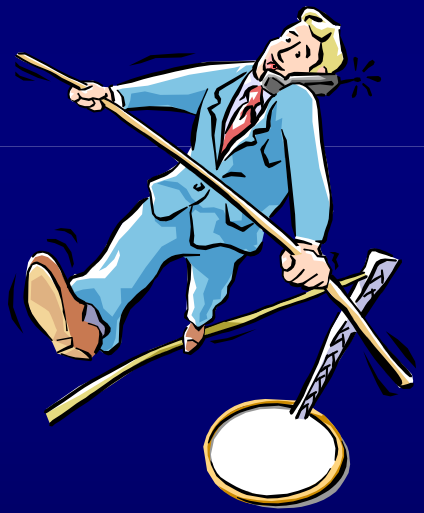
- Spesso il software non viene progettato per essere modificato facilmente
- Vengono apportate modifiche direttamente sui programmi, senza modificare
  - la documentazione di progetto
  - la documentazione di test
  - la specifica dei requisiti
  - ...

Processo di sviluppo del software

Utilizzo e manutenzione

- *Re-ingegnerizzazione* del software
  - Ristrutturazione del codice (*refactoring*)
  - Conversione del linguaggio
  - *Reverse engineering*

- Pianificazione, controllo e gestione del processo di sviluppo
  - Analisi dei costi
  - Gestione del gruppo di lavoro
- Gestione dei rischi
  - Rischi relativi ai requisiti
  - Rischi legati alle risorse umane
  - Rischi tecnologici
  - Rischi politici



Processo di sviluppo del software

- Modello a cascata
- Modelli evolutivi
- Sviluppo incrementale – iterativo
- Modello a spirale
- Modelli specializzati
  - Sviluppo a componenti
  - Modello dei metodi formali
  - Sviluppo aspect-oriented
  - Sviluppo model driven
  - *Unified Process* (UP - RUP)

Processo di sviluppo del software

Modello a cascata *(waterfall model)*

- Fasi distinte, in cascata tra loro con retroazione finale

Modello a cascata

- Il modello si fonda sul presupposto che introdurre cambiamenti sostanziali nel software in fasi avanzate dello sviluppo ha costi troppo elevati pertanto, ogni fase deve essere svolta in maniera esaustiva prima di passare alla successiva, in modo da non generare retroazioni
- Le uscite che una fase produce come ingresso per la fase successiva sono i cosiddetti semilavorati del processo di sviluppo:
  - Documentazione di tipo cartaceo
  - Codice dei singoli moduli
  - Sistema complessivo

Modello a cascata

- Oltre alle fasi, è fondamentale definire:
  - Semilavorati
    al fine di garantire che ci possa essere un'attività di controllo della qualità dei semilavorati
  - Date
    entro le quali devono essere prodotti i semilavorati al fine di certificare l'avanzamento del processo secondo il piano stabilito

Modello a cascata

- I limiti sono dati dalla sua rigidità
  in particolare da due assunti di fondo molto discutibili:
  - Immutabilità dell'analisi
    gli utenti sono in grado di esprimere esattamente le loro esigenze e di conseguenza
    in fase di analisi iniziale è possibile definire tutte le funzionalità che il software deve realizzare
  - Immutabilità del progetto
    è possibile progettare l'intero sistema prima di aver scritto una sola riga di codice

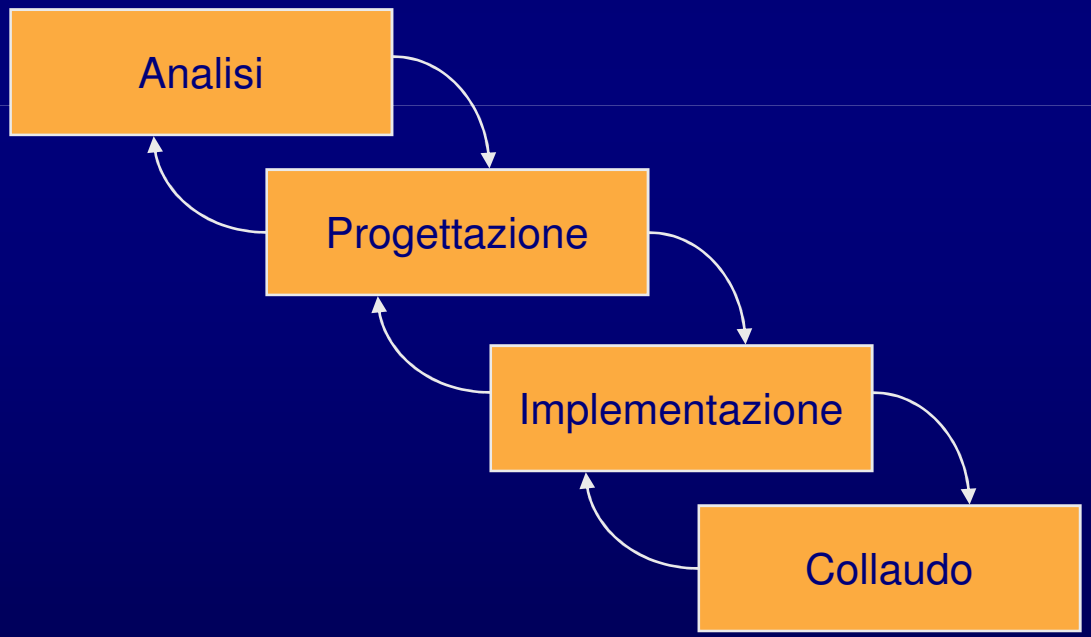
Modello a cascata

- Nella realtà:
  - La visione che gli utenti hanno del sistema evolve man mano che il sistema prende forma e quindi le specifiche cambiano in continuazione
  - Nel campo della progettazione "le idee migliori vengono in mente per ultime", quando si comincia a vedere qualcosa di concreto
    Inoltre problemi di prestazioni costringono spesso a rivedere le scelte di progetto

Processo di sviluppo del software

Modello a cascata

- Evoluzioni successive al modello originale ammettono forme limitate di retroazione a un livello



Modello a cascata

- Per evitare problemi, prima di iniziare a lavorare sul sistema vero e proprio è meglio realizzare un prototipo in modo da fornire agli utenti una base concreta per meglio definire le specifiche
- Una volta esaurito il compito, il prototipo viene abbandonato (throw-away prototyping) e si procede a costruire il sistema reale secondo i canoni del modello a cascata
- Questo approccio risulta però quasi sempre così dispendioso da annullare i vantaggi economici che il modello a cascata dovrebbe garantire

Prototipo

- Modello approssimato dell'applicazione
  Obiettivo: 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
  - con costi minimi
- Alternativa interessante in tutti i casi in cui lo sviluppo dell'applicazione parta inizialmente con requisiti non perfettamente noti o instabili

Prototipo

- Prototipazione "*throw-away*"
  - Il prototipo che si realizza è del tipo *usa e getta*
  - Obiettivo: comprendere meglio le richieste del cliente e quindi migliorare la definizione dei requisiti del sistema
  - Il prototipo si concentra sulle parti che sono mal comprese con l'obiettivo di contribuire a chiarire i requisiti

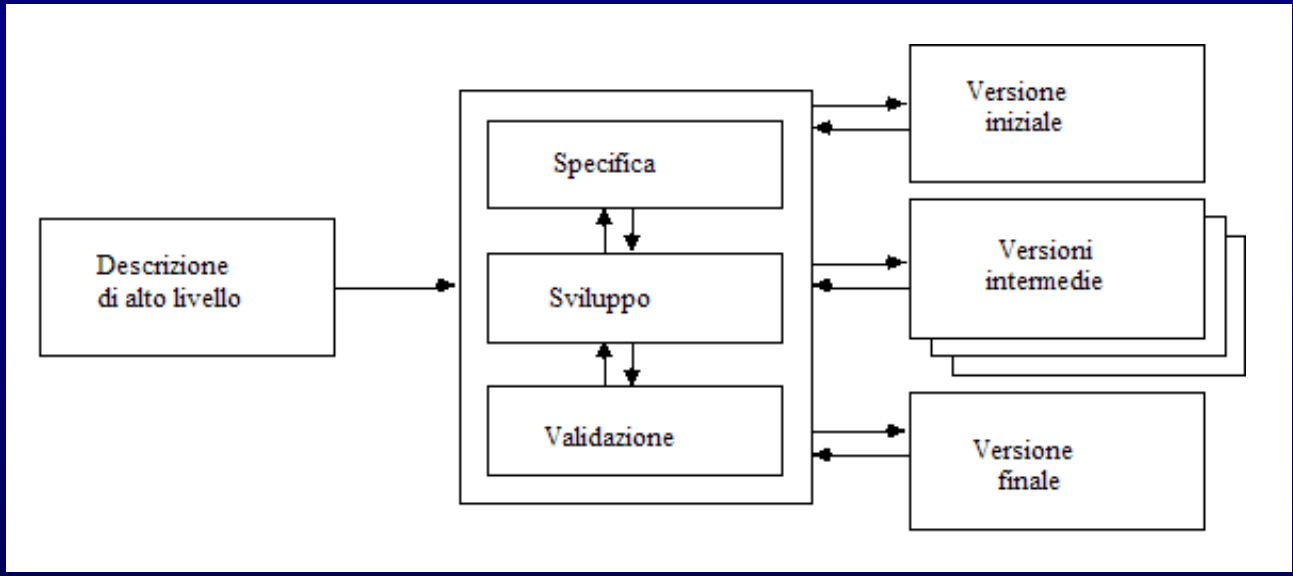
Prototipo

- Programmazione esplorativa
  - Il prototipo si trasforma progressivamente nel prodotto finale
  - Obiettivo: lavorare a stretto contatto con il cliente per
    - Chiarire completamente i requisiti
    - Giungere a un prodotto finale
  - Prima, si sviluppano le parti del sistema che sono ben chiare (requisiti ben compresi)
  - Quindi, si aggiungono nuove parti/funzionalità come proposto dal cliente

Processo di sviluppo del software

Modelli evolutivi

- Partendo da specifiche molto astratte, si sviluppa un primo prototipo
  - Da sottoporre al committente
  - Da raffinare successivamente



Modelli evolutivi

- Esistono diversi modelli di tipo evolutivo, ma tutti in sostanza propongono un ciclo di sviluppo in cui un prototipo iniziale evolve gradualmente verso il prodotto finito attraverso un certo numero di iterazioni
- Il vantaggio fondamentale è che ad ogni iterazione è possibile
  - confrontarsi con gli utenti per quanto riguarda le specifiche e le funzionalità (raffinamento dell'analisi)
  - rivedere le scelte di progetto (raffinamento del *design*)

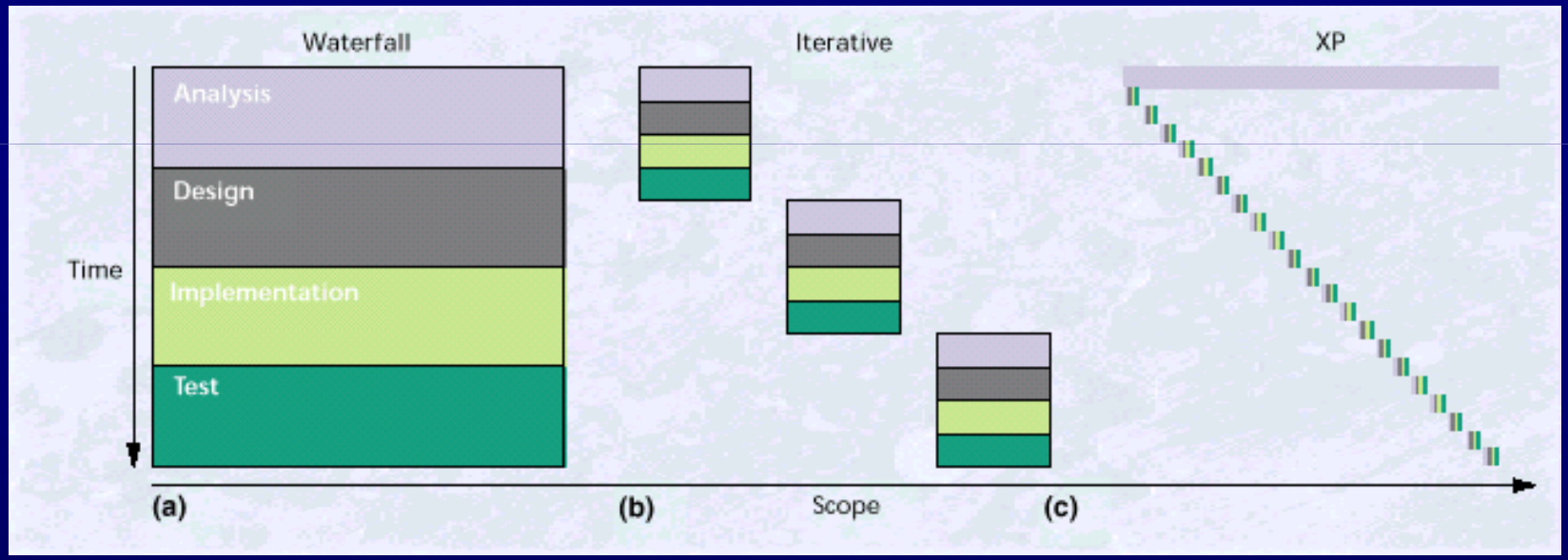
Modelli evolutivi

- I modelli evolutivi si sono orientati verso cicli sempre più brevi e iterazioni sempre più veloci, fino ad arrivare al modello più “radicale” che prende il nome di *Extreme Programming* (XP)

Processo di sviluppo del software

Modelli evolutivi

- L'illustrazione, tratta da un articolo di Kent Beck, mostra l'evoluzione dal modello a cascata, all'*extreme programming*



Processo di sviluppo del software

Modelli evolutivi

- Problemi
  - Il processo di sviluppo non è visibile (ad es., documentazione non disponibile)
  - Il sistema sviluppato è poco strutturato (modifiche frequenti)
  - È richiesta una particolare abilità nella programmazione (*team* ristretto)
- Applicabilità
  - Sistemi di piccole dimensioni
  - Sistemi che avranno breve durata
  - Parti di sistemi più grandi

Extreme Programming

- Si basa su:
  - Comunicazione: ci deve essere grande comunicazione sia tra gli sviluppatori, sia tra sviluppatori e clienti
  - Testing: è necessario avere più codice per il test che per il programma vero e proprio - ogni programmatore deve scrivere il programma di test parallelamente se non prima di scrivere il codice effettivo

Extreme Programming

- Si basa su:
  - Semplicità: il codice deve essere il più semplice possibile - complicare il codice tentando di prevedere futuri riutilizzi è controproducente sia come qualità di codice prodotto, sia come tempo necessario - d'altra parte, il codice semplice e comprensibile è il più riutilizzabile
  - Coraggio: non si deve avere paura di modificare il sistema che deve essere ristrutturato in continuazione, ogni volta che si intravede un possibile miglioramento (*refactoring*)

Processo di sviluppo del software

Modelli ibridi

- Sistemi composti di sotto-sistemi
- Per ogni sotto-sistema è possibile adottare un diverso modello di sviluppo
  - Modello evolutivo
    per sotto-sistemi con specifiche ad alto rischio
  - Modello a cascata
    per sotto-sistemi con specifiche ben definite
- Di norma conviene creare e raffinare prototipi funzionanti del sistema o di sue parti, secondo l'approccio incrementale - iterativo

Sviluppo incrementale

- Sviluppo incrementale
  si costruisce il sistema sviluppandone sistematicamente e in sequenza parti ben definite
  una volta costruita una parte, essa non viene più modificata
  - è di fondamentale importanza essere in grado di specificare perfettamente i requisiti della parte da costruire, prima della sua implementazione

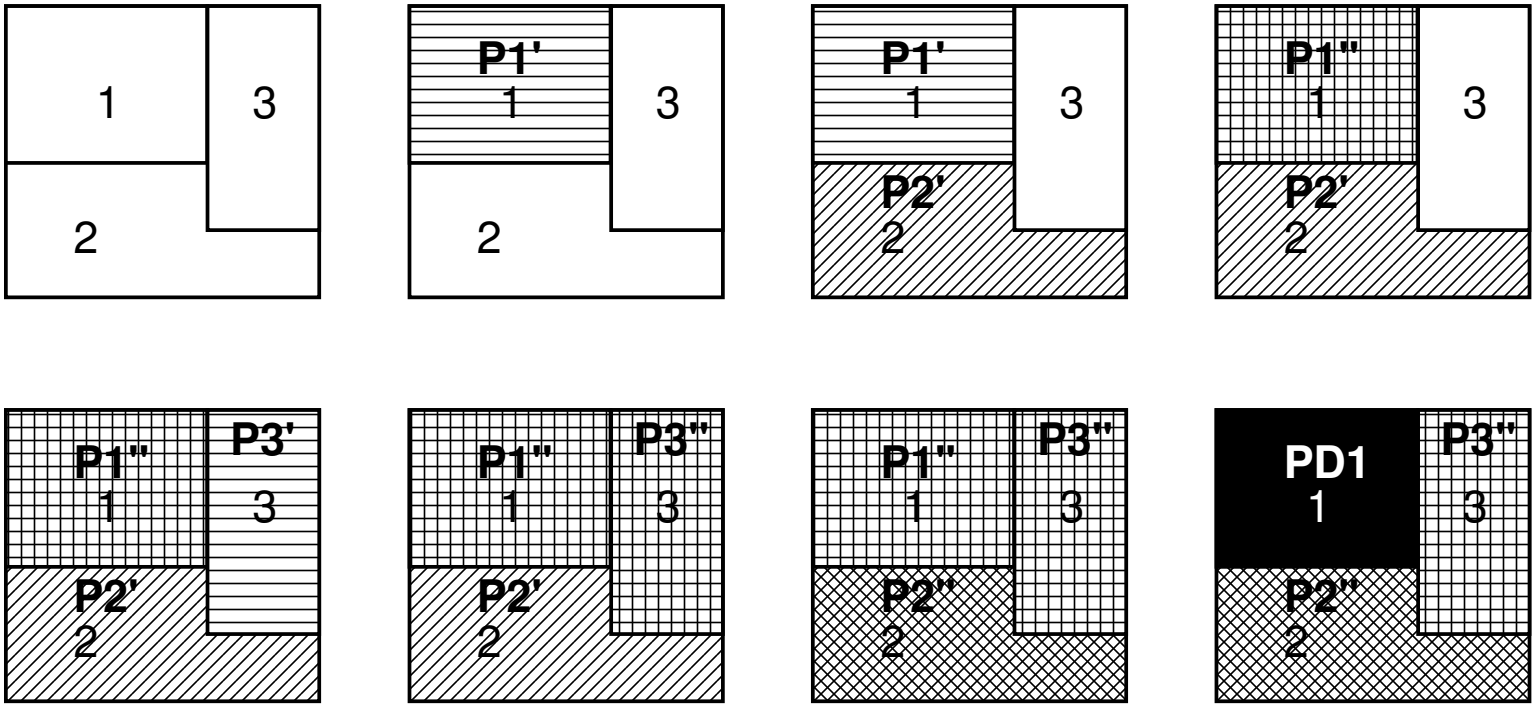
Sviluppo iterativo

- Sviluppo iterativo
  si effettuano molti passi dell'intero ciclo di sviluppo del software, per costruire iterativamente tutto il sistema, aumentandone ogni volta il livello di dettaglio
  - non funziona bene per progetti significativi

Sviluppo incrementale - iterativo

- Sviluppo incrementale - iterativo
  - Si individuano sottoparti relativamente autonome
  - Si realizza il prototipo di una di esse
  - Si continua con altre parti
  - Si aumenta progressivamente l'estensione e il dettaglio dei prototipi, tenendo conto delle altre parti interagenti
  - E così via…

Sviluppo incrementale - iterativo



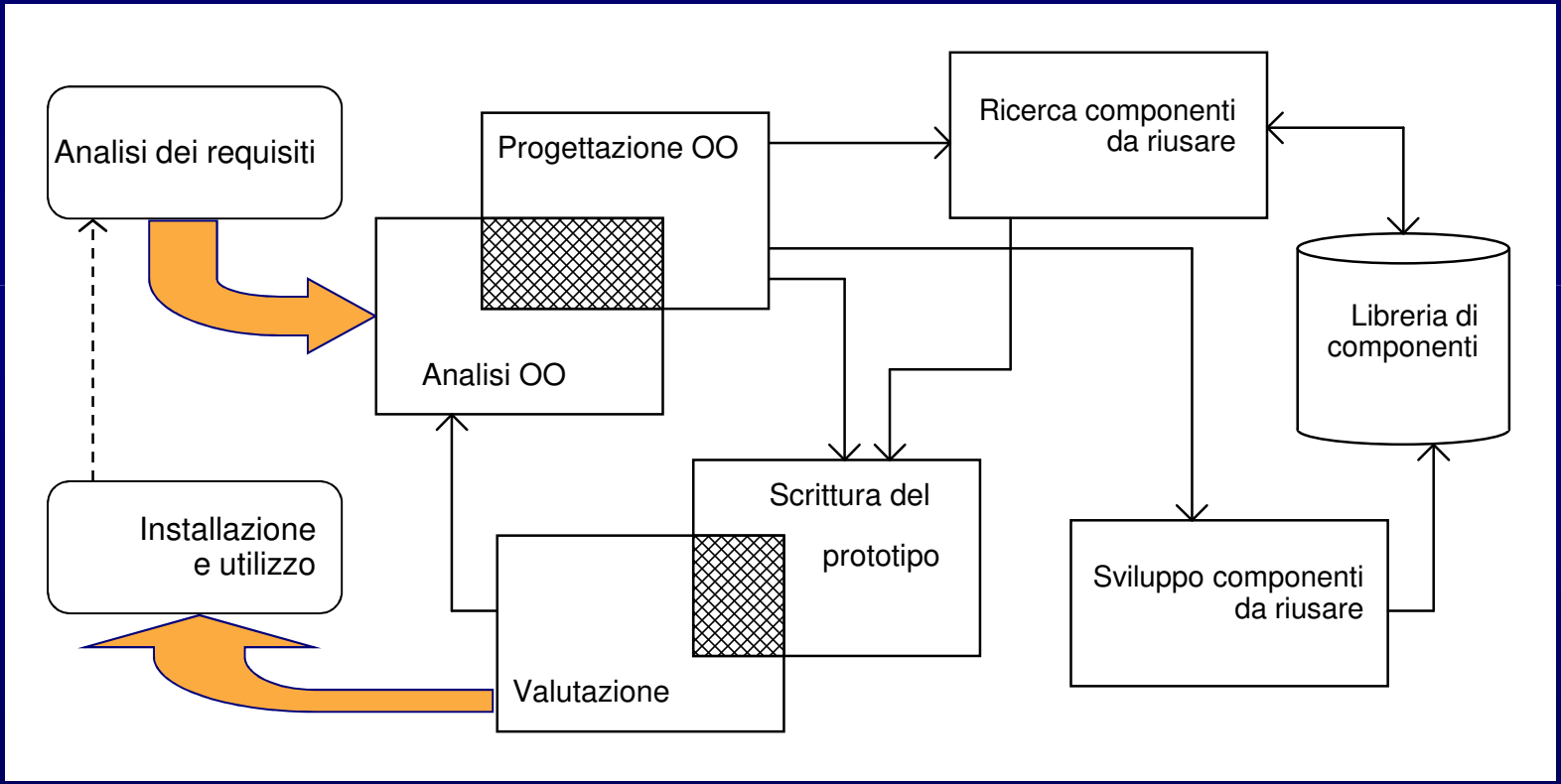
Pn' : Primo prototipo

Pn'' : Secondo prototipo

PDn : Prototipo definitivo

Processo di sviluppo del software

Sviluppo a componenti



Linguaggi di modellazione

- Permettono di descrivere (modellare) un sistema di qualche natura
- Possono essere grafici o testuali
  - Se grafici, sono basati su uno o più tipi di diagrammi, costruiti a partire da simboli grafici con una semantica ben definita
- Possono essere interpretabili
  - Un modello può essere eseguito
    simulazione più o meno completa del comportamento del sistema modellato
  - Un modello può essere tradotto
    generazione del codice sorgente utilizzabile nell'implementazione del sistema

Linguaggi di modellazione

- Modelli semantici dei dati
  - Entità-Relazioni (E-R)
- Modelli orientati all'elaborazione dati
  - Diagrammi di Flusso dei Dati (*Data-Flow Diagrams*, DFD)
- Modelli orientati alla classificazione
  - Modelli orientati agli oggetti
- Modelli operazionali
  - Automi a stati finiti
  - Reti di Petri
- Modelli descrittivi
  - Logica del primo ordine
  - Logica temporale

Linguaggi di modellazione

- "*Nessun cliente vi ringrazierà per avergli fornito diagrammi accurati; quello che vorranno da voi è software funzionante*"
- Perché usare un linguaggio di modellazione?
- Dobbiamo risolvere un problema di comunicazione
  - tra i progettisti
  - tra i progettisti e il committente
  - tra i progettisti presenti e i progettisti futuri…

Linguaggi di modellazione

- Il linguaggio naturale è troppo impreciso
- Il codice è preciso, ma troppo dettagliato
- La soluzione migliore è utilizzare un linguaggio di modellazione
  - Sufficientemente **preciso**
  - **Flessibile** dal punto di vista descrittivo
    per poter arrivare a un qualunque livello di dettaglio
  - Possibilmente **standard**

Processo di sviluppo del software

Linguaggi di modellazione

- Esiste ormai una convergenza su un unico linguaggio di modellazione di tipo grafico: **UML** (*Unified Modeling Language*)
- Sviluppato verso la metà degli anni '90 da Grady Booch, James Rambaugh, Ivar Jacobson
- Nel 1999, OMG (*Object Management Group*) ha definito la versione 1.3 del linguaggio (http://www.omg.org)
- La versione corrente è la 2.1 (http://www.uml.org)

Processo di sviluppo del software

Unified Modeling Language

- È un linguaggio di modellazione grafico
- Permette di creare varie descrizioni più o meno astratte di un sistema (non necessariamente software)
- È indipendente dal processo di sviluppo
- Prevede meccanismi di estensione del linguaggio stesso
- È orientato agli oggetti
- È uno standard OMG

Unified Modeling Language

- Diagrammi strutturali
  - Diagramma delle classi
  - Diagramma degli oggetti
  - Diagramma dei componenti
  - Diagramma dei package
  - Diagramma di deployment
  - Diagramma delle strutture composite (UML 2.0)
- Diagrammi comportamentali
  - Diagramma dei casi d'uso
  - Diagramma delle attività
  - Diagramma degli stati
  - Diagramma di sequenza
  - Diagramma di comunicazione (ex di collaborazione)
  - Diagramma dei tempi (UML 2.0)
  - Diagramma di sintesi dell'interazione (UML 2.0)

Fattori di qualità del software

- Qualità esterne
  percepibili da un osservatore esterno che esamina il prodotto software come se fosse una scatola nera (*black-box*)
  - Affidabilità
  - Facilità d'uso
  - Velocità
  - …
- Devono essere garantite

Fattori di qualità del software

- Qualità interne
  osservabili esaminando la struttura interna del prodotto software, come se questo fosse una scatola trasparente (*white-box*)
  - Modularità
  - Leggibilità
  - …
- Influenzano le qualità esterne
- Sono un modo per realizzare le qualità esterne

Fattori di qualità del software

Correttezza *(correctness)*

- Data una definizione dei requisiti che il software deve soddisfare,
  il software si dice corretto se rispetta tali requisiti

I requisiti specificati risultano spesso incompleti rispetto ai requisiti reali

Fattori di qualità del software

Robustezza *(robustness)*

- Il software si dice robusto se si comporta in maniera accettabile anche in corrispondenza di situazioni anomale e comunque non specificate nei requisiti

- Nel caso di situazioni anomale, il software non deve causare disastri (perdita di dati o peggio)
  - o termina l'esecuzione in modo pulito
  - o entra in una modalità particolare, in cui non sono più attive alcune funzionalità (*graceful degradation mode*)

Fattori di qualità del software

Affidabilità *(reliability)*

- Il software si dice affidabile se
  - le funzionalità offerte corrispondono ai requisiti (il software è corretto)
  - in caso di guasto, non produce danni fisici o economici (il software è robusto)

Affidabilità = Correttezza + Robustezza

Fattori di qualità del software

Facilità d'uso *(ease of use)*

- Facilità con cui l'utilizzatore del software è in grado di:
  - Imparare ad usare il sistema
  - Utilizzare il sistema
  - Fornire i dati da elaborare
  - Interpretare i risultati
  - Gestire condizioni di errore

Fattori di qualità del software

Efficienza *(efficiency)*

- Buon utilizzo delle risorse hardware:
  - Processori (tempo di calcolo)
  - Memoria principale (occupazione di memoria)
  - Memorie secondarie
  - Canali di comunicazione

Fattori di qualità del software

Integrità *(integrity)*

- Protezione – sicurezza
  abilità del sistema software di proteggere i vari componenti (programmi, dati, documenti) da
  - accessi e/o modifiche non autorizzati

  di natura
  - sia volontaria
  - sia involontaria

Fattori di qualità del software

Estensibilità *(extensibility)*

- Facilità con cui il software può essere modificato per soddisfare nuovi requisiti
- La modifica di programmi di piccole dimensioni non è un problema serio
- I sistemi software di dimensioni medie e grandi possono soffrire di fragilità strutturale: modificando un singolo elemento della struttura si rischia di far collassare l'intera struttura

Fattori di qualità del software

Estensibilità *(extensibility)*

- Due principi essenziali:
  - Semplicità architetturale
    - più l'architettura del sistema è semplice
    - più è facile da modificare
  - Modularità e decentralizzazione
    - più il sistema è suddiviso in moduli autonomi
    - più è facile che una modifica coinvolga un numero limitato di moduli

Fattori di qualità del software

Riusabilità *(reusability)*

- Il software è riusabile se può essere riutilizzato completamente o in parte in nuove applicazioni

- Permette di non dover reinventare soluzioni a problemi già affrontati e risolti
- Influenza tutte le altre caratteristiche dei prodotti software

Fattori di qualità del software

Verificabilità *(verifiability)*

- Facilità con cui il prodotto software può essere sottoposto a test

Fattori di qualità del software

Portabilità *(portability)*

- Facilità con cui il prodotto software può essere trasferito su altre architetture hardware e/o software

Fattori di qualità del software

- Molti dei fattori di qualità del software sono definibili soltanto in maniera
  - Intuitiva
  - Poco rigorosa
- L'obiettivo di fornire meccanismi precisi di misura non è realisticamente raggiungibile

Fattori di qualità del software

- Alcune caratteristiche sono in contrapposizione, ad esempio:
  efficienza vs portabilità
- L'importanza dell'una o dell'altra caratteristica varia (può variare) a seconda del settore applicativo
- Il costo del software aumenta esponenzialmente se è richiesto un livello molto alto di una qualunque di tali caratteristiche