Ingegneria del Software
dalla programmazione individuale al processo produttivo del software

Obiettivi.
Presentare le caratteristiche del ciclo di vita di un prodotto software.
Proporre un modello di realizzazione e spiegare perché si usano i prototipi.
Evidenziare gli aspetti salienti della fase di definizione dei requisiti.
Illustrare le principali strategie di progettazione.
Presentare fasi e modalità di test e collaudo.

Fulvio Sbroiavacca

Ingegneria del Software
Ciclo e Modelli

Obiettivi.
Presentare l'evoluzione del software verso il prodotto di un processo industriale e l'introduzione dell'Ingegneria del Software o Software Engineering (SWE).
Illustrare le caratteristiche del ciclo di vita di un prodotto software.
Rispondere ad alcune domande: come cambiano le esigenze dell'utenza, quali sono le ricadute sui costi, cosa si intende per manutenzione?
Proporre un modello di realizzazione e spiegare perché si usano i prototipi.
Presentare i principali Modelli per lo sviluppo del software.
Spiegare il significato del riuso del software.

Fulvio Sbroiavacca

Evoluzione del software

- I primi calcolatori sono stati sviluppati negli anni ’40 del 1900, la “programmazione” avveniva con il saldatore
- Dagli anni ’50 del 1900 furono resi disponibili gli strumenti di programmazione primordiali: i linguaggi macchina, poi i linguaggi assemblativi, poi i linguaggi di alto livello come Fortran (per applicazioni scientifiche) e Cobol (per applicazioni gestionali)
- La programmazione avveniva a livello individuale: un programma era di dimensioni limitate, e veniva sviluppato e mantenuto da una persona
- La fine degli anni ’60 vede il verificarsi della crisi del software:
  - Le macchine sempre più potenti ed il mercato richiedevano la realizzazione di programmi sempre più complessi
  - La forte lievitazione del costo del software

Software come prodotto di un processo industriale

- Il passaggio dalla programmazione individuale alla programmazione di squadra
  - il cambiamento del modo di sviluppare il software:
    da prodotto artigianale a prodotto industriale

- Nasce l'Ingegneria del Software o Software Engineering (SWE) per:
  - minimizzare il costo complessivo di un prodotto software
  - ottenere prodotti di qualità (che certamente costano di più inizialmente in termini monetari, ma che sono più sicuri ed affidabili)

- Il software diventa un prodotto industriale attraverso:
  - compiti differenziati del personale coinvolto nel processo
  - metodologie di sviluppo

I tipici problemi che presentava il software prima dell'introduzione del SWE (1)

- Tempi di consegna
  Non venivano rispettati
  (ad es. il successo del MacIntosch fu ritardato di molti anni a causa di ritardi software)
  Questo è un problema ancora attuale, anche se attenuato

- Costi di produzione
  Non venivano rispettati
  Non era raro che dal momento della commissione al momento della consegna i costi si fossero moltiplicati per 10

I tipici problemi che presentava il software prima dell'introduzione del SWE (2)

- Bisogni dell'utente
  Era frequentissima la mancata soddisfazione dei bisogni dell'utente
  Veniva consegnato un prodotto che faceva cose diverse rispetto i bisogni

- Scarsa affidabilità
  Il sistema corrispondeva ai requisiti dell'utente ma si bloccava spesso
  Oppure perdeva interi database

- Rigidità
  Al più piccolo cambiamento delle necessità dell'utente il sistema non era più adatto ed era difficile o impossibile modificarlo

A cosa sono dovuti i tipici problemi del software? (1)

- Tempi di consegna e Costi di produzione
  Derivano dal fatto che il produttore non ha le idee chiare
  su come deve avvenire il processo produttivo
  E' necessario un modello teorico ed affidabile del processo di produzione del software

- Bisogni dell'utente
  Derivano da una cattiva comunicazione tra cliente e produttore
  Il produttore non comprende ciò che il cliente vuole ottenere ed il cliente non comprende ciò che il produttore gli propone
  Vi è quindi una mancata comprensione dei requisiti

A cosa sono dovuti i tipici problemi del software? (2)

- Scarsa affidabilità
  Deriva da errori non trovati in fase di testing
  Può essere conseguenza di una cattiva organizzazione del programma, che rende il test più complesso
  Può anche interessare l'hardware del calcolatore (memoria) oppure le caratteristiche del sistema operativo (mancanza delle performance desiderate)
  Vi sono quindi varie carenze nelle fasi di progettazione e sviluppo

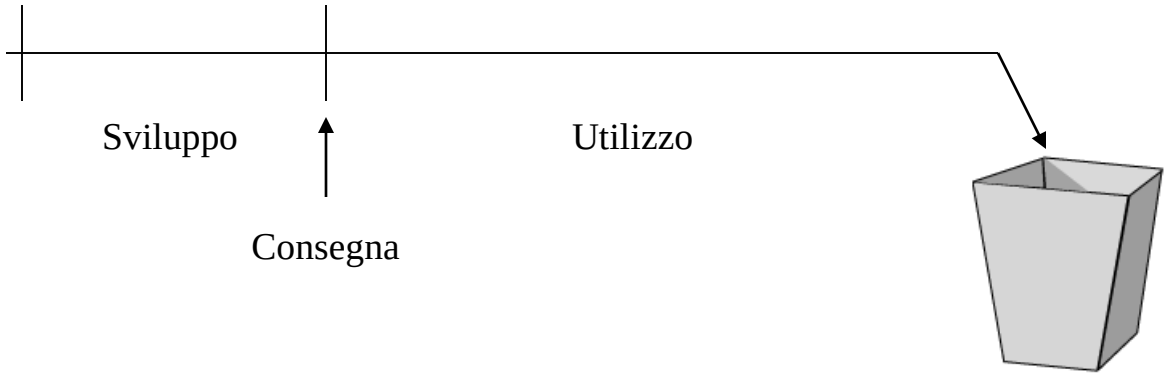
- Rigidità
  Dipende da problemi di accoppiamento tra parti e mancanza di modularità
  Si verifica per la poca indipendenza tra i moduli del programma
  Denota un forte accoppiamento
  Vi è quindi una errata scomposizione in fase di progettazione

Perché viene acquistato un prodotto software?

- *Viene acquistato se i benefici che produrrà*
  saranno maggiori
  del suo costo di acquisto
  (analisi costi/benefici)

- Per fare un buon prodotto software un produttore deve riuscire a:
  - *indovinare i costi,*
  - *consegnare il prodotto in tempo,*
  - *soddisfare i requisiti del cliente,*
  - *con un prodotto affidabile*
  - *e modificabile*

Ciclo di vita di un prodotto software



- Lo sviluppo avviene nell'azienda del produttore, la sua durata è normalmente meno lunga della fase di utilizzo
- La fase di utilizzo inizia al momento della consegna del prodotto e dura fino a quando viene eliminato
  L'utilizzo avviene nell'azienda del cliente
  Il produttore è spesso chiamato in causa per interventi di manutenzione

Cosa si intende per manutenzione? (1)

- *Per manutenzione di un prodotto software si intendono i diversi tipi di intervento che si rendono necessari durante la fase di utilizzo*

- *Quali sono i tipi di intervento e quindi di manutenzione che si rendono necessari nella fase di utilizzo?*

- Eliminare errori residui, trovati dopo la consegna
  - Si dice *manutenzione correttiva*
  - E' molto difficile, se non impossibile, produrre software privo di errori (ad esempio alcune misure indicano che i programmi di mercato hanno in media 40 errori ogni 1000 righe di codice, la maggior parte dei quali si verificano solamente in situazioni molto particolari ed improbabili)
  - *Il produttore sa che consegna un prodotto contenente degli errori*
  - *Deve fare in modo che siano facilmente eliminabili*

Cosa si intende per manutenzione? (2)

- Adattare a nuove configurazioni (hardware e software di base)
  - Si dice *manutenzione adeguativa*
  - Visto anche il suo costo, il tempo di utilizzo del software è più lungo del tempo di utilizzo dell'hardware
  - Il produttore non deve comunque vincolare l'utente ad operare con un hardware che può rapidamente diventare obsoleto
  - *Questo punto deve essere esplicitamente contrattato tra produttore ed utente (ad esempio devono sempre essere dichiarate le caratteristiche minime dell'hardware necessario per un certo prodotto)*

- Adattare a cambiamenti dell'ambiente applicativo
  - Si dice *manutenzione evolutiva*
  - I requisiti del cliente cambiano nel tempo
  - *Deve essere possibile modificare almeno parzialmente il prodotto per soddisfare nuovi requisiti del cliente*

I diversi tipi di manutenzione

- *La manutenzione è quindi il complesso degli interventi pianificati, progettati ed eseguiti su un prodotto software, già operante ed accettato dal committente (e quindi rilasciato dal processo di sviluppo), rivolti a:*
  - ripristinarne le funzionalità in seguito a malfunzionamenti (m. correttiva)
  - migliorarne le prestazioni, l'organizzazione o la qualità (m. migliorativa)
  - adeguarlo alle innovazioni tecnologiche (m. adeguativa)
  - renderlo rispondente alle nuove esigenze del committente, o a nuove normative, che comportino variazioni al software esistente, od implichino la necessità di integrazione con nuove funzioni, senza variarne gli obiettivi primari (m. ordinaria)
  - renderlo rispondente alle nuove esigenze del committente, o modifiche di normative o regolamenti che comportino lo sviluppo di nuove funzionalità o rendano necessari interventi di manutenzione sul prodotto già in gestione, che trascendano le modifiche di manutenzione ordinaria (m. evolutiva)

Come cambiano le esigenze del cliente?

- *Automatizzare una parte del sistema informativo del cliente vuol dire fotografare la realtà informativa della sua organizzazione (analisi del flusso dell'informazione) e immetterne la logica in un calcolatore: questo è il software*

- L'introduzione del prodotto nell'organizzazione modifica però la realtà informativa, cambiano pertanto modalità ed esigenze

- Nel contempo aumenta la cultura informatica dell'utente che vuole quindi sfruttare meglio il suo sistema informatico, viene stimolato a concepire nuove funzionalità da automatizzare

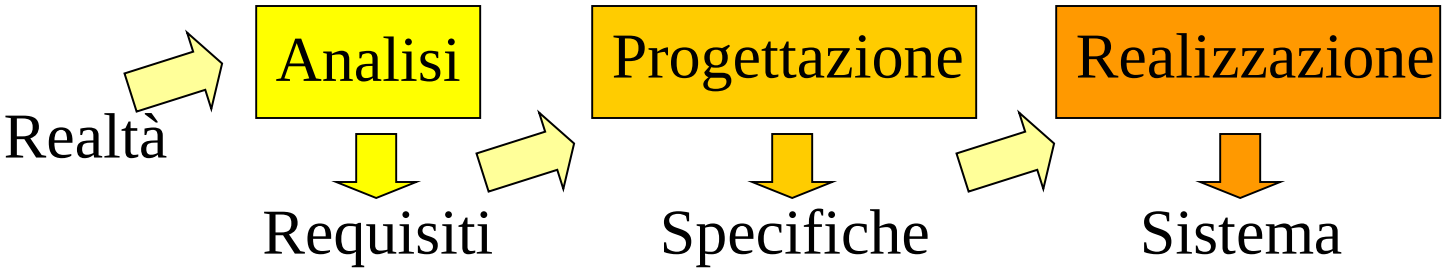
- *Le esigenze in genere sono proporzionali alle risorse disponibili* (ad esempio alla potenza di calcolo, allo spazio di memoria di massa disponibile: più spazio è disponibile più in fretta si riempie)

Quali sono le ricadute sui costi?
Costi del software

- I costi sono distribuiti durante tutto il ciclo di vita del software
- Tipi di costi:
  - Costi diretti quando il cliente chiama il produttore per effettuare una modifica del prodotto
  - Costi indiretti dovuti all'inutilizzabilità del sistema in seguito al verificarsi di un errore bloccante
- Generalmente la maggioranza dei costi ricade nella parte di utilizzo
  - ogni volta che il cliente interagisce con il produttore deve sostenere dei costi
- L'idea alla base del SWE è abbattere i costi globali
  - aumentare i costi di sviluppo per diminuire i costi di utilizzo
  - il cliente deve spendere di più inizialmente per un prodotto di qualità maggiore, mentre il costo di utilizzo diventa molto più lieve
- Un software che rende minimi i costi di utilizzo deve essere:
  - non rigido, affidabile e modificabile

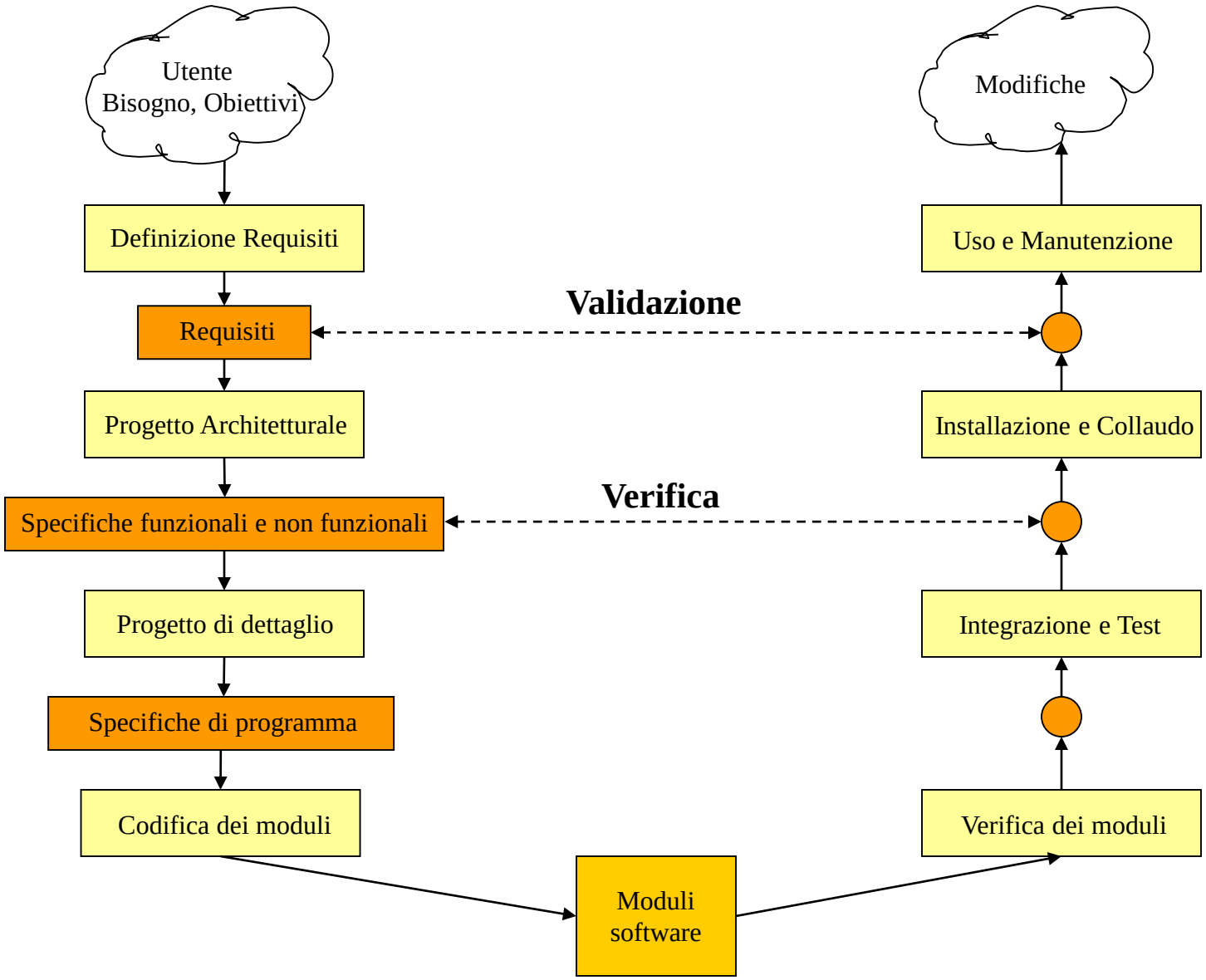
Modelli per lo sviluppo del software

- Consideriamo un modello del processo produttivo del software
- Un modello è una formalizzazione di una realtà, una prospettiva attraverso la quale guardare e descrivere un fenomeno con un certo grado di precisione



- Nel nostro caso la Realtà sarà soggetta ad una attività di Analisi, che produrrà dei Requisiti utili alla Progettazione, che esprimerà delle Specifiche necessarie per l'attività di Realizzazione del Sistema finale
- Il primo modello che analizziamo si chiama Modello a Cascata

Modello a Cascata

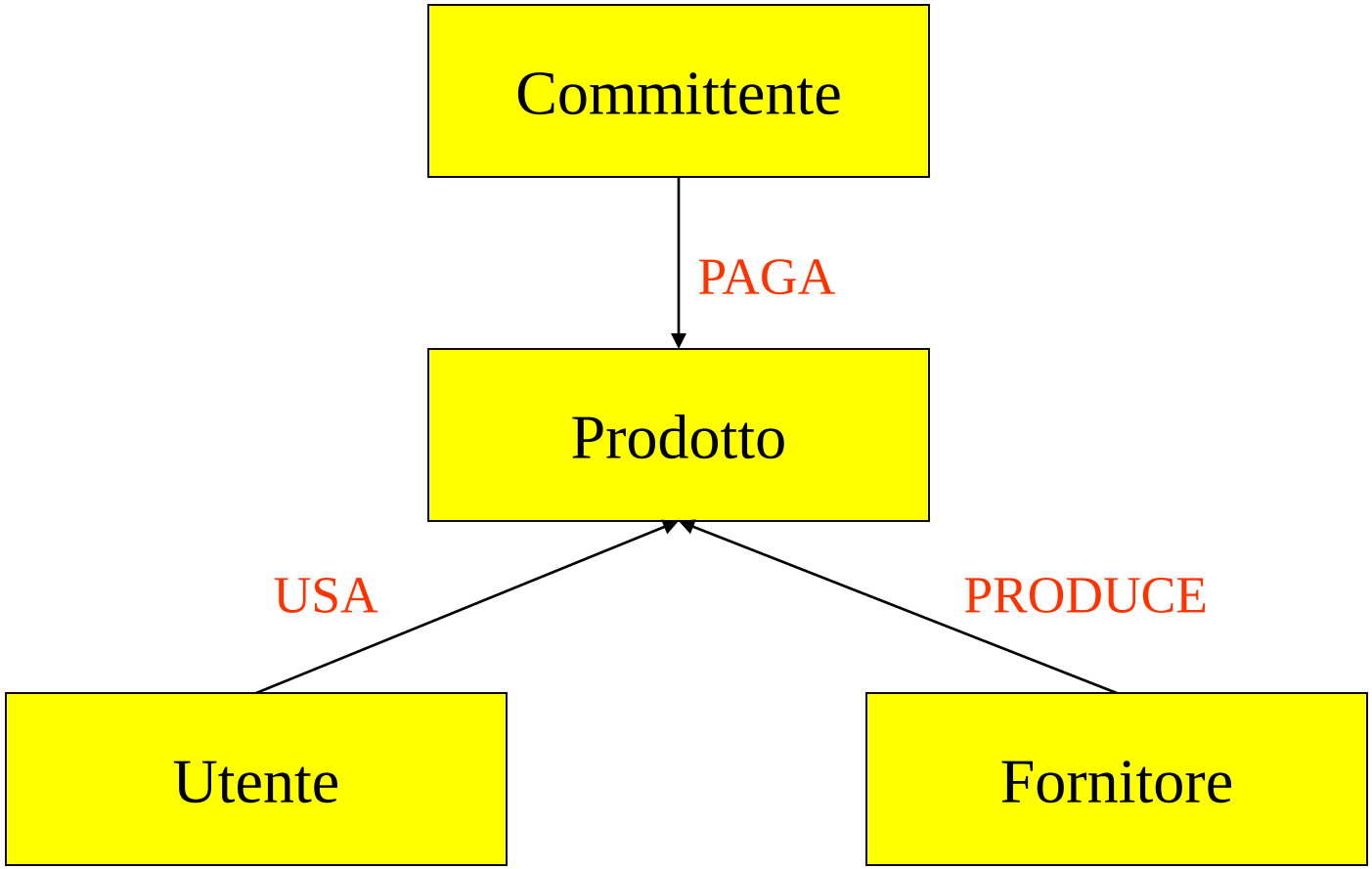


Modello a Cascata (1)

Utente e Definizione dei requisiti

- L'utente esprime
  - un bisogno, lo stato nel quale si potrebbe trovare rispetto al presente
  - degli obiettivi, una percorso per superare un bisogno
- Il cliente quando va dal produttore per commissionare il prodotto desiderato, normalmente ha un'idea molto vaga dello stesso
- Il fornitore cerca capire quali sono i requisiti del prodotto che soddisfano i bisogni ed obiettivi del cliente
  - tratta direttamente con gli utenti
  - analizza lo stato presente dell'azienda del committente, il suo sistema informativo attuale
- Bisogna distinguere i vari ruoli:
  - il fornitore o produttore
  - il committente: chi paga
  - l'utente: chi userà il prodotto

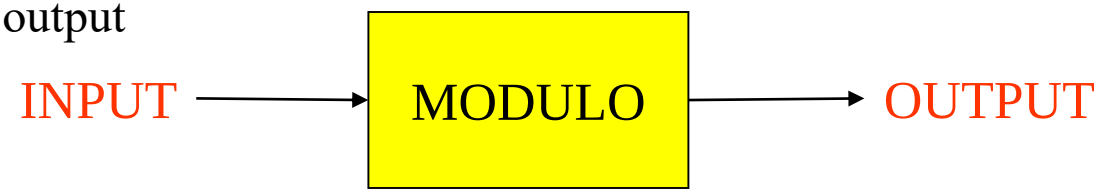
Ruoli nel Modello a Cascata



Modello a Cascata (2)

Progetto architetturale e Specifiche

- Durante la fase di progetto architetturale vengono definite le specifiche del prodotto per definire in modo preciso che cosa deve fare il software da produrre
- Le specifiche sono di due tipi:
  - Specifiche funzionali che descrivono una funzionalità del software: ogni modulo è visto come una funzione, ad esempio viene descritto il rapporto tra l'input e l'output



  - Specifiche non funzionali che rappresentano i vincoli ai quali deve sottostare il software, ad esempio: per compatibilità con un sistema esistente si vuole che l'input sia formulato secondo una particolare codifica (ad esempio XML), per un sistema real-time che l'output sia prodotto in meno di mezzo secondo

Modello a Cascata (3)

Progetto dettagliato, codifica e verifica dei moduli

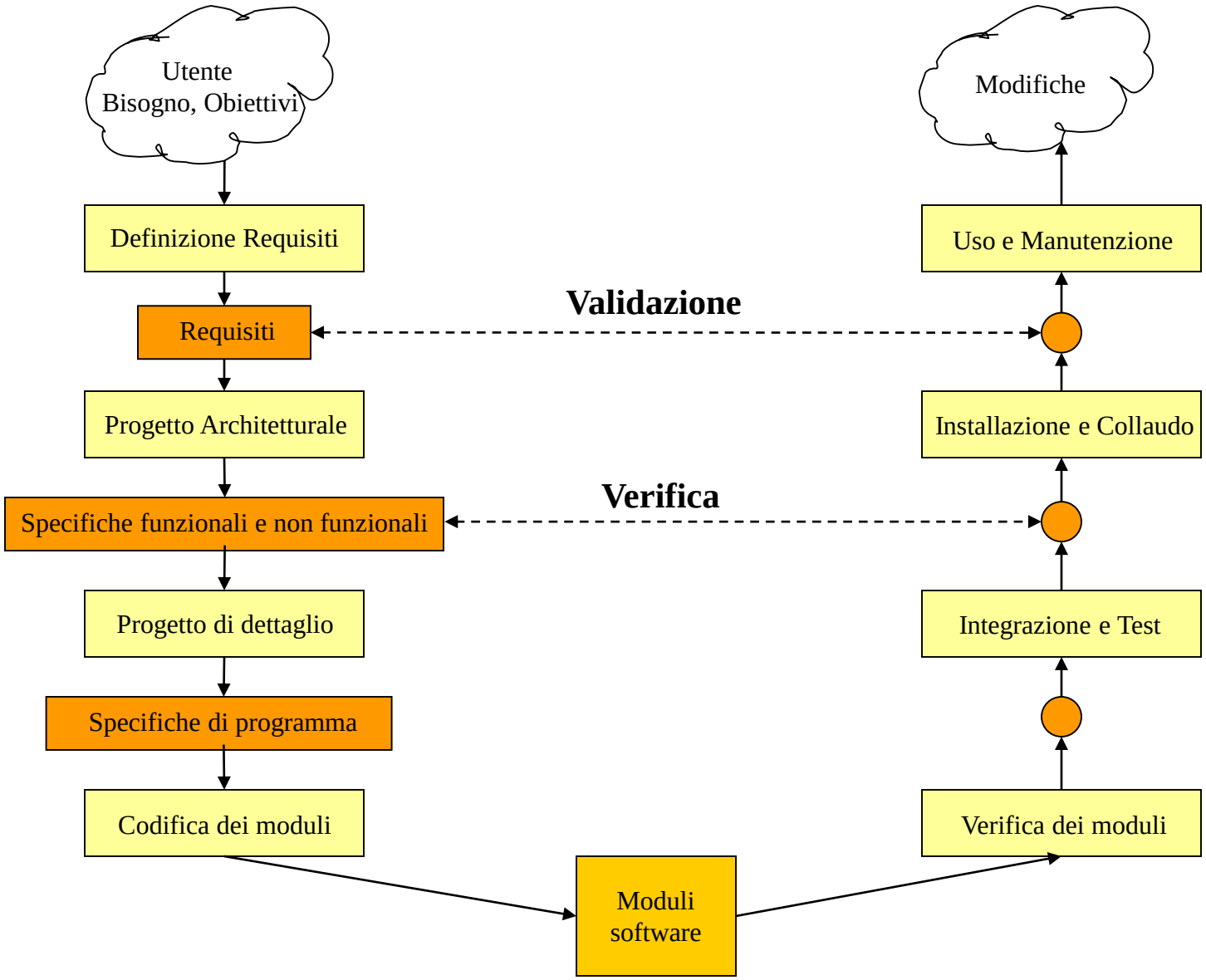
- Durante la fase di progetto dettagliato vengono definiti in modo preciso i moduli che comporranno il software e la loro struttura interna
  - viene definita l'interfaccia ed i dati globali
  - si fanno le scelte su come realizzare il software
- Nella fase di codifica vengono prodotti i moduli software
  - si tratta di una fase tecnica, le decisioni principali sono state prese nelle fasi precedenti
- La fase di verifica mira ad eliminare gli errori di realizzazione attraverso:
  - Testing: esecuzione del modulo su campioni di dati
  - Verifica statica: lettura del software con particolari criteri per verificarne il comportamento rispetto alle specifiche

Modello a Cascata (4)

Integrazione e test, installazione e collaudo

- Quando ogni modulo è individualmente corretto si passa all'integrazione dei moduli
  - viene testato il sistema complessivo confrontando le specifiche funzionali e non funzionali
  - *Verification: "Are we building the product right?"*
- Dopo l'installazione del prodotto si precede al collaudo
  - la prova da parte dell'utente che il prodotto soddisfa i suoi bisogni ed obiettivi
  - il sistema viene confrontato con i requisiti
  - *Validation: "Are we building the right product?"*
- Infine si entra nella fase di uso e mantenimento
  - è necessario un confronto con i nuovi bisogni che l'utente esprime nel tempo

Modello a Cascata



Qual è l'incidenza degli errori sul costo del prodotto? Il costo degli errori (1)

- L'esito finale della fase di sviluppo è la soddisfazione dei bisogni dell'utente
- *Se l'utente non è soddisfatto ci sono stati degli errori nella fase di sviluppo*

- Errore nella fase di definizione dei requisiti
  - Si ripercuote sulle fase successive
  - Anche se i successivi passi sono svolti in modo corretto il prodotto non potrà soddisfare l'utente, pur essendo funzionante
  - Ci si accorge di un tale errore al momento del collaudo
    Validation: "we don't build the right product"

Qual è l'incidenza degli errori sul costo del prodotto? Il costo degli errori (2)

- Errore nella fase di progetto architetturale
  - Riguarda le specifiche
  - Se la specifica implicata è funzionale significa che è stata sbagliata la descrizione formale di una funzionalità
    - Ad esempio un modulo doveva effettuare il calcolo dell'età
      ed invece è stato fatto calcolare l'anno di nascita,
      il modulo è corretto,
      ma quando un altro modulo si aspetta la restituzione dell'età,
      ottiene invece l'anno di nascita
  - Ci si accorge di questi errori al momento dell'integrazione e test del sistema integrato
    Verification: "we don't build the product right"

Qual è l'incidenza degli errori sul costo del prodotto? Il costo degli errori (3)

- Errore nella fase di progetto dettagliato
  - Si tratta di errori riguardanti i moduli:
    un modulo non esegue ciò che gli viene richiesto
  - Ci si accorge di questi errori in fase di verifica e test

- Errore durante la codifica
  - L'errore viene segnalato direttamente dall'ambiente di sviluppo o compilatore

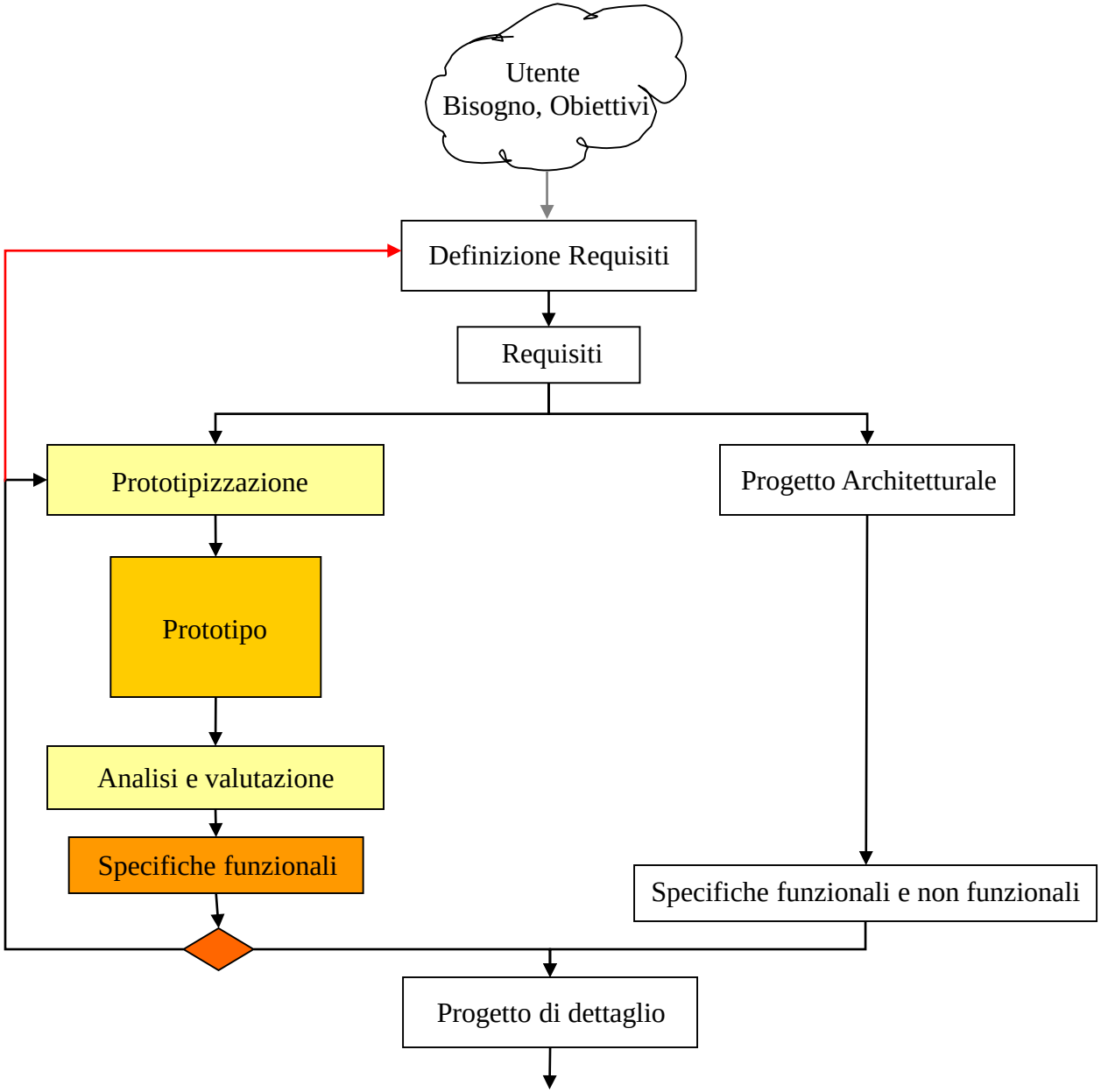
Applicabilità del modello a cascata

- Nel modello a cascata non vengono sviluppate strade alternative in parallelo
  - Viene scelta una soluzione e sviluppata, se non porta ai risultati sperati la si abbandona per sviluppare ex novo una soluzione alternativa
- Seguire soluzioni alternative è utile in particolare nelle prime fasi per esplorare strade diverse prima di scegliere quella ritenuta più opportuna
- *Progettazione, definizione dei requisiti, produzione delle specifiche, ecc. richiedono in realtà attività cicliche, iterative*
  - Si svolgono le attività affinando il risultato più volte fino ad ottenere un risultato soddisfacente
- Il modello a cascata è applicabile se è possibile specificare completamente il sistema prima della fase di codifica
  - Bisogna cioè saper fare il prodotto prima di realizzarlo
  - Vi sono casi nei quali questa situazione non è applicabile
    (ad esempio per lo sviluppo di un sistema esperto di diagnostica oppure di un sistema di relazione con i clienti senza conoscere le caratteristiche di questi ultimi)

Perché si sviluppano i prototipi?

- Se l'utente non è soddisfatto ci sono stati degli errori nel ciclo descritto
- Gli errori più costosi sono quelli che si verificano ai livelli più astratti
  - prima si sbaglia, più tardi si intercetta l'errore, più costoso diventa effettuare la correzione
- Un errore nella fase di definizione dei requisiti si ripercuote su tutte le fasi successive
  - anche se i successivi passi sono svolti in modo corretto il prodotto, pur essendo funzionante, non potrà soddisfare l'utente
- Lo sviluppo di un prototipo, anticipando parte delle attività delle fasi successive, consente di ridurre gli errori precoci
  - Progettazione, definizione dei requisiti, produzione delle specifiche, richiedono in realtà attività cicliche, iterative: si svolgono le attività affinando il risultato più volte fino ad ottenere un risultato soddisfacente
- L'introduzione di un ciclo di prototipizzazione consente di ridurre le probabilità di errore nella definizione dei requisiti ed in parte anche nella definizione delle specifiche

Ciclo di Prototipizzazione (integrato nel Modello a Cascata)



Come si usano i prototipi

- Quando il fornitore non è certo di capire esattamente ciò che vuole l'utente, o su richiesta dell'utente stesso, viene anticipata una fase di codifica e di verifica all'inizio del ciclo di sviluppo del prodotto
- Il prototipo è un sistema in grado di mostrare tutte le funzionalità del prodotto finale, ma che non rispetta le specifiche non funzionali
  - può essere lento e poco affidabile
  - riguardare il solo sviluppo dell'interfaccia utente per la verifica delle modalità di interazione

- *Ciò che l'utente vede è la sua percezione del sistema,*
  per l'utente il sistema è l'interfaccia

- Il prototipo è sviluppato per capire le caratteristiche che l'utente vuole dal prodotto finale
- Normalmente una volta verificato con l'utente si butta via

Il modello esplorativo

- Se non si possono determinare le specifiche, non può essere definita la correttezza del sistema, ossia la corrispondenza con le specifiche
  - In questi casi viene utilizzato il concetto di adeguatezza, ossia corrispondenza con i bisogni e gli obiettivi del cliente
- Il modello esplorativo consiste in una successione di prototipi che viene fatta convergere verso una soluzione adeguata, soddisfacente per il cliente
  - Ad ogni passo viene costruito un prototipo e presentato alla valutazione critica del cliente, che lo può accettare come adeguato alle sue esigenze, oppure può indicare nuovi criteri e requisiti che il sistema deve possedere
  - Nei passi finali il prototipo viene affinato ad ogni ciclo finché viene accettato dal cliente
- Questo modello presenta due caratteristiche salienti:
  - Non vi è distinzione tra prototipo finale e prodotto
    Questo normalmente non risulta vantaggioso per il fornitore
  - Vi è un forte coinvolgimento del cliente nella fase di sviluppo (rispetto al modello a cascata)
    Con conseguenti costi anche per il cliente

Il riuso del software

- I metodi visti in precedenza riguardano la costruzione ex novo di un prodotto interamente nuovo per ogni specifica definita
- Con il miglioramento di strumenti e linguaggi di sviluppo, nel corso del tempo il software si accumula, risolvendo funzionalità di uso comune, vengono messe a disposizione componenti di validità generale, raccolte spesso in librerie
- *Il riutilizzo del software è un'idea che si basa sulla costruzione di nuovo software semplicemente assemblando componenti esistenti e sviluppando software di integrazione delle stesse*
- Perché ciò possa avvenire vantaggiosamente sono necessari strumenti automatici di supporto:
  - le componenti devono essere mantenute in un Database
  - devono essere corredate da un'ottima documentazione
  - è necessario un sistema di Information Retrieval per permetterne il reperimento