Ingegneria del Software

Il Processo Software (I parte)

Antonino Staiano
e-mail: **antonino.staiano@uniparthenope.it**

Obiettivi della lezione

- Comprendere il concetto di Processo Software
- Comprendere il concetto di Modello di Processo Software
  - Cascata
  - Basato su riuso
  - Evolutivo
  - Trasformazionale
- Il modello incrementale
- Il modello a spirale

Processi SW: introduzione

- Un processo software è un insieme di attività atte alla creazione di un prodotto software

- Lo sviluppo può partire da zero in un linguaggio di programmazione standard oppure il nuovo software è sviluppato estendendo e modificando sistemi esistenti

Processi SW: introduzione

- I processi sw sono complessi e basati su decisioni e pregiudizi delle persone
  - Successo limitato dei tentativi di automazione del processo
  - Gli strumenti CASE (Computer Aided Software Engineering) sono in grado di semplificare solo alcune attività del processo
    - Non è ancora pensabile sostituire la creatività umana nella progettazione

Processi SW: introduzione

- Non esiste un processo software ideale
  - Esistono molteplici processi software differenti
    - Le grandi organizzazione hanno sviluppato propri approcci in base alle proprie necessità
  - I processi si sono evoluti anche in base alla specifiche caratteristiche dei sistemi da sviluppare
    - Ad esempio, per i sistemi critici (sistemi per cui un fallimento del sw può portare a gravi perdite economiche, danni fisici o minacce per la vita) un processo di sviluppo strutturato è più adeguato
    - Per i sistemi aziendali in cui i requisiti cambiano velocemente, potrebbe essere più efficace un processo agile e flessibile

Processi SW: introduzione

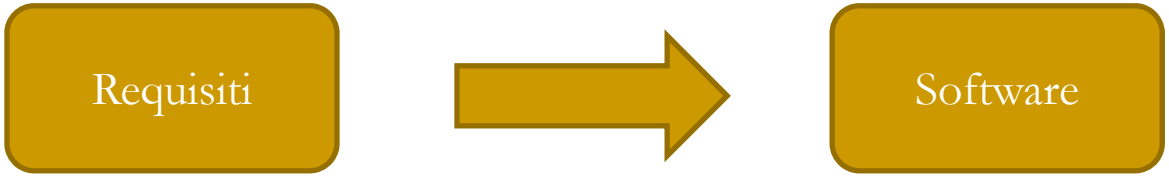
- I requisiti sono complessi e ambigui
  - Il cliente, inizialmente, non sa bene cosa fare
- I requisiti sono variabili
  - Cambiamenti tecnologici, organizzativi etc…
- Difficile gestire frequenti modifiche
  - Difficile stimare i costi ed identificare cosa consegnare al cliente
- I sistemi software interagiscono
  - Necessità di interagire con sistemi pre-esistenti

Processo software: standard IEEE 610.12-1990

- Processo di sviluppo del software: il processo per cui le necessità dell'utente sono tradotte in un prodotto software. Il processo prevede la traduzione delle necessità dell'utente nelle richieste del software, trasformare le richieste del software nel progetto, implementare il progetto in forma di codice, testing del codice e talvolta, installare e controllare il software per un uso operativo.
  - Nota: queste attività possono sovrapporsi o essere eseguite iterativamente.

Processo software

- In pratica …



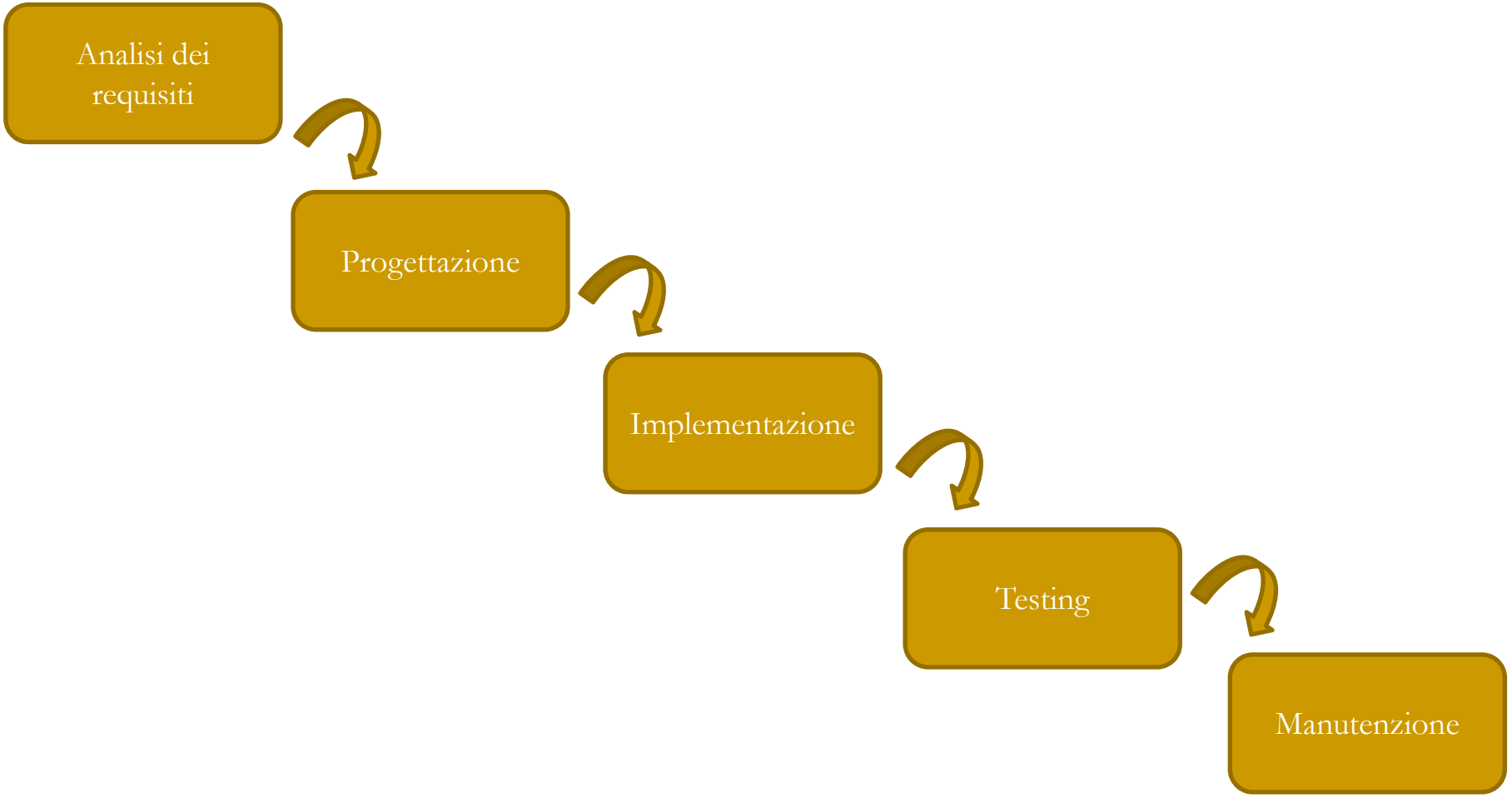
Identificazione delle attività del processo software

Analisi dei requisiti	Quale è il problema?	Dominio del problema
Progettazione del sistema	Quale è la soluzione?	
Progettazione del programma	Quali sono i meccanismi che meglio implementano la soluzione?	Dominio implementazione
Implementazione del programma	Come è costruita la soluzione?	
Testing	Il problema è stato risolto?	
Consegna	La soluzione (il prodotto) è utilizzabile dai clienti?	
Manutenzione	Necessitano evoluzioni?	

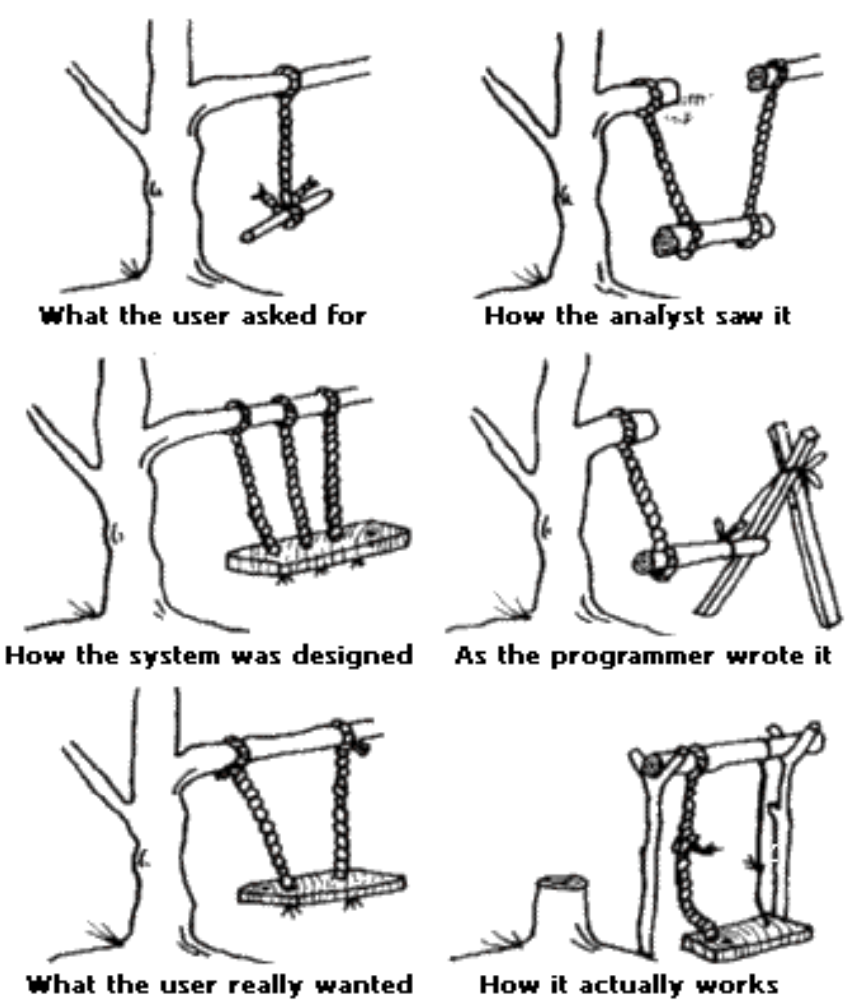
Processi SW: attività del processo software

- Le attività del processo software fondamentali comuni a tutti i processi sono:
  - **Specifiche del software**: si definiscono le funzionalità del sw e fissati i vincoli operativi
  - **Progettazione e implementazione**: si sviluppa il sw in moda da realizzare i requisiti individuati
  - **Convalida**: si verifica che il sw garantisca le funzionalità richieste dal cliente
  - **Evoluzione**: il sw deve poter essere modificato in modo da soddisfare i cambiamenti dei requisiti dell'utente

Processo SW: una possibile organizzazione



I modelli di processo



Modelli di processo

- Un modello di processo è un'astrazione del processo software
  - Rappresenta il processo software da una particolare prospettiva
- Sono modelli generici (**paradigmi di processo**)
  - rappresentano strutture di processo estensibili e adattabili per creare processi più specifici

Modelli di ciclo di vita del software o di processo di sviluppo del software

- "Un modello del ciclo di vita del software è una caratterizzazione descrittiva o prescrittiva di come un sistema software viene o dovrebbe essere sviluppato"

  W. Scacchi - Encyclopedia of Software Engineering Vol II pag 860

- I modelli di processo software sono descrizioni precise e formalizzate delle attività, degli oggetti, delle trasformazioni e degli eventi per realizzare e/o ottenere l'evoluzione del software

- NB molti autori usano i termini *processo di sviluppo del software e ciclo di vita del software come* sinonimi.

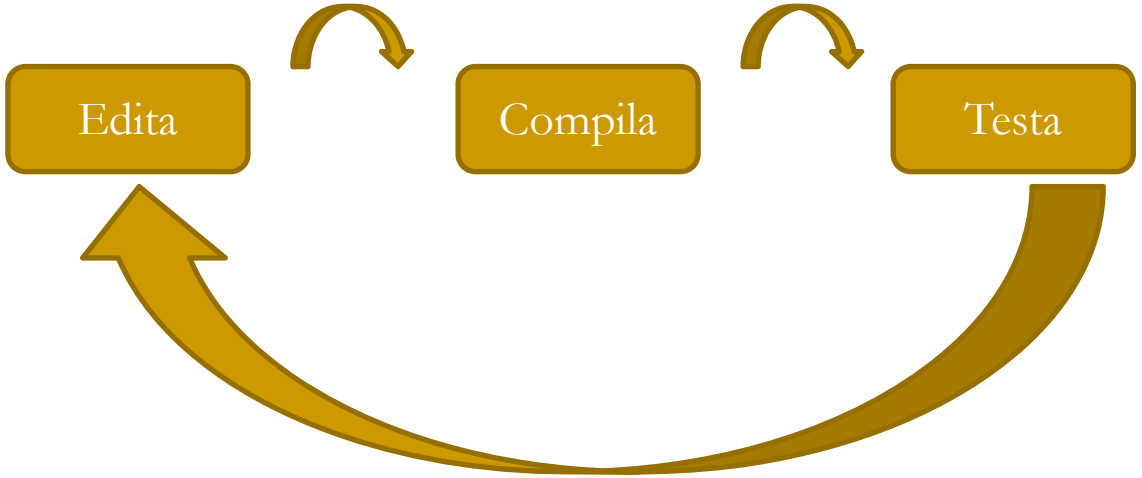
Modelli di processo: caratteristiche (1)

- Una strutturazione dell'organizzazione del lavoro nelle fabbriche del software
  - … fasi della produzione,
  - … tipi di attività,
  - … collegamento ed interfacciamento,
  - … controllo e misura,
- ma anche linee guida per: organizzare, pianificare, dimensionare personale, assegnare budget, schedulare e gestire, …

Modelli di processo: caratteristiche (2)

- definire e prescrivere prodotti e documenti da rilasciare al committente (**deliverables)**
- determinare e classificare metodi e strumenti più adatti a supportare le attività previste
- framework per analizzare, stimare, migliorare ...

Il più semplice (e peggiore) modello di processo



- Molto veloce, feedback rapido
- Molti strumenti disponibili
- Specializzato per codifica
- Non incoraggia la documentazione
- Ingestibile durante manutenzione

Modelli di processo software

- Modello a cascata (Waterfall)
  - Le principali attività di processo sono rappresentate come fasi separate
- Sviluppo evoluzionistico
  - Intreccia le attività di specifica, sviluppo e convalida.
    - Parte con specifiche astratte e poco precise per sviluppare un sistema iniziale.
    - Il sistema viene perfezionato secondo gli input dei clienti in base alle loro necessità.
- Sviluppo basato su riuso
  - Sfrutta un numero di componenti esistenti e riutilizzabili
  - Il processo di sviluppo si concentra sull'integrazione di tali componenti

Modelli di processo software

- I tre modelli di processo generici si utilizzano nella pratica corrente dell'Ingegneria del SW
  - Non si escludono reciprocamente
  - Usati insieme nello sviluppo di sistemi di grandi dimensioni
    - I sottosistemi all'interno di un sistema più ampio possono essere sviluppati usando differenti approcci

Modelli di processo SW: il modello trasformazionale

- Sono state proposte diverse varianti rispetto ai tre modelli generici

- La più nota consiste nello sviluppo dei sistemi formali
  - Viene creato un modello matematico e modificato usando trasformazioni matematiche che preservano la sua consistenza fino all'eseguibile finale

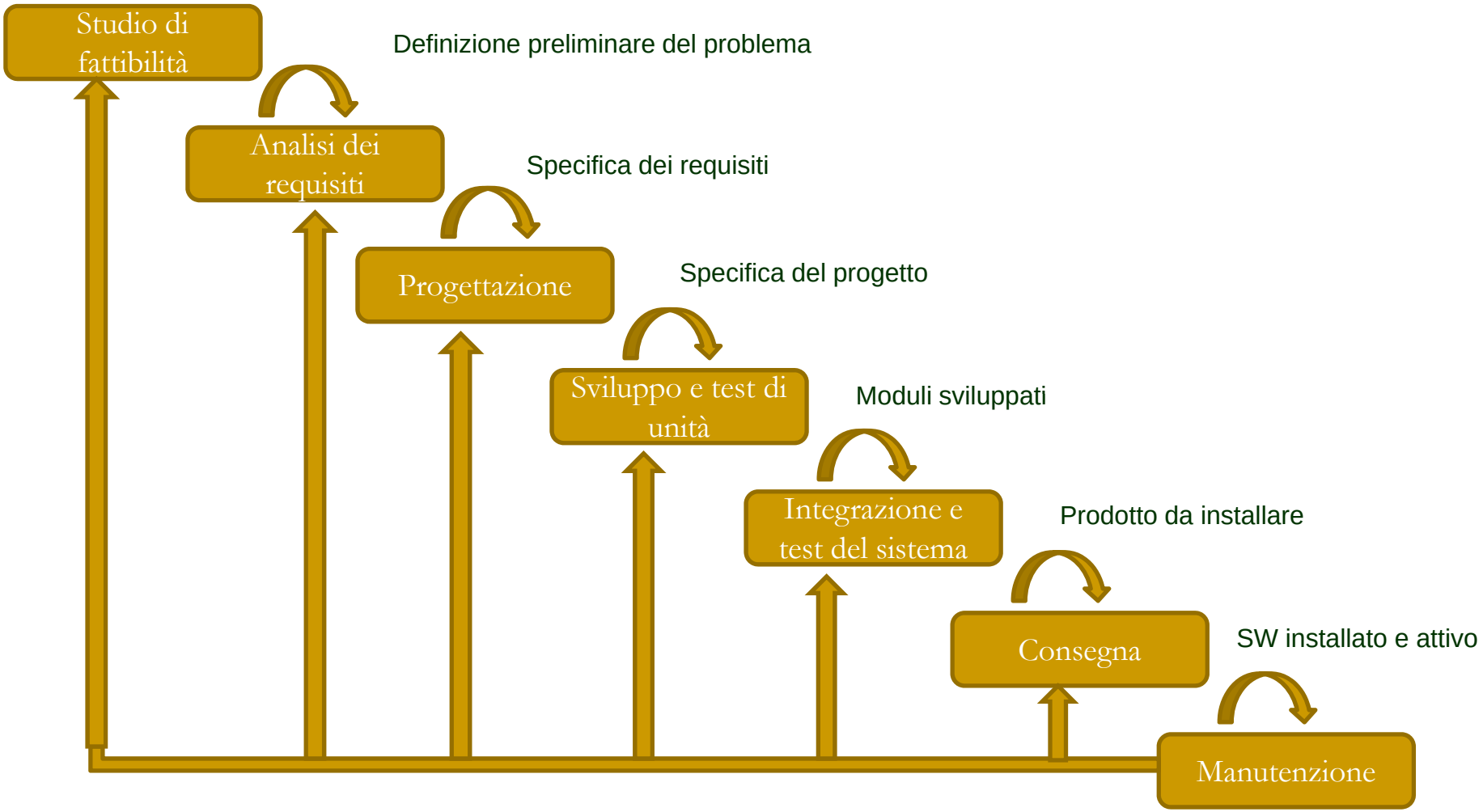
Modello a Cascata (Waterfall) - ’70

- Popolare negli anni ’70
  - reazione al “code and fix” originario
  - ispirazione dall’industria manifatturiera
- Modello sequenziale lineare
  - progressione sequenziale (in cascata) di fasi, senza ricicli, al fine di meglio controllare tempi e costi
  - definisce e separa le varie fasi e attività del processo
    - nullo (o minimo) overlap fra le fasi
  - uscite intermedie: semilavorati del processo (documentazione di tipo cartaceo, programmi)
    - formalizzati in struttura e contenuti
  - consente un controllo dell’evoluzione del processo
    - attività trasversali alle diverse fasi

Modello a cascata: organizzazione sequenziale delle fasi

- Ogni fase raccoglie un insieme di attività omogenee per metodi, tecnologie, skill del personale, etc.
- Ogni fase è caratterizzata da:
  - Attività (tasks),
  - Prodotti di tali attività (deliverables),
  - Controlli di qualità/stato (quality control measures)
- La fine di ogni fase è un punto rilevante del processo (**milestone)**
- I risultati (semilavorati) di una fase sono input alla fase successiva
- I prodotti di una fase vengono "congelati", ovvero non sono più modificabili se non innescando un processo formale e sistematico di modifica
- Il limite del modello a cascata è la difficoltà ad effettuare cambiamenti nel corso del processo

Modello a cascata: fasi



Modello a cascata: fasi

- Analisi e definizione dei requisiti
  - Si individuano i servizi del sistema, i suoi vincoli e obiettivi mediante riunioni con i clienti o utenti del sistema
  - Sono definiti nel dettaglio e usati come specifiche del sistema
- Progettazione del sistema e del software
  - Si suddividono i requisiti sul sistema HW o sul SW
  - Si stabilisce l'architettura generale del sistema
  - La progettazione coinvolge l'identificazione e la descrizione delle astrazioni fondamentali del sistema e le relative relazioni

Modello a cascata: fasi

- Implementazione e test delle unità
  - Il SW è realizzato come un insieme di unità di programma
  - Il test delle unità verifica che ognuna soddisfi le proprie specifiche
- Integrazione e test del sistema
  - Le singole unità di programma sono integrate e testate come programma completo in modo da verificare che siano soddisfatti i requisiti del software
  - Dopo il test finale avviene la consegna al cliente

Modello a cascata: fasi

- Operatività e manutenzione
  - Fase più lunga del ciclo di vita
  - Il sistema è installato e messo in opera
  - Si correggono errori non individuati nelle fasi iniziali del ciclo di vita
    - Si migliorano l'implementazione delle unità di sistema e si aggiornano i servizi forniti dal sistema quando sono individuati nuovi requisiti

Modello a cascata: considerazioni

- In teoria
  - Ogni fase si conclude con uno o più documenti approvati
    - La fase successiva non dovrebbe iniziare prima che sia terminata la precedente
- In pratica
  - Gli stadi sono sovrapposti e si scambiano informazioni vicendevolmente
    - Durante la progettazione sono identificati problemi nei requisiti
    - Durante la programmazione sono individuati problemi di progettazione ecc.
- Il processo SW non è un modello lineare
  - Consiste in una sequenza di cicli attraverso le attività di sviluppo

Modello a cascata: considerazioni

- I cicli sono dispendiosi
  - I documenti sono costosi da produrre e approvare
  - Comportano consistenti rielaborazioni
  - Dopo un determinato numero di ripetizioni si congelano alcune parti dello sviluppo (ad esempio, le specifiche) e si continua con gli stadi successivi
  - I problemi che permangono sono risolti successivamente, aggirati o ignorati del tutto

Modello a cascata: considerazioni

- Durante la fase di operatività e manutenzione il SW viene messo in produzione
  - Sono scoperti gli errori e le omissioni nei requisiti originali
  - Emergono errori di programmazione e progettazione
  - Si identificano nuove funzionalità
- Il sistema deve evolvere per poter rimanere utile
  - I cambiamenti da apportare richiedono potenzialmente la ripetizione dei precedenti stadi del processo

Modello a cascata: vantaggi e svantaggi

- Vantaggi
  - ha definito molti concetti utili
  - ha rappresentato un punto di partenza importante per lo studio dei processi SW
  - facilmente comprensibile e applicabile
  - Ogni fase del processo produce la relativa documentazione
- Svantaggi
  - interazione con il committente solo all'inizio e alla fine
    - requisiti congelati alla fine della fase di analisi
    - requisiti utente spesso imprecisi: "l'utente sa quello che vuole solo quando lo vede"
    - Errori nei requisiti scoperti solo alla fine del processo
  - il nuovo sistema software diventa installabile solo quando è totalmente finito
    - né l'utente né il management possono giudicare prima della fine l'adesione del sistema alle proprie aspettative

Modello a cascata: quando usarlo

- Il modello a cascata dovrebbe essere utilizzato
  - Quando i requisiti del software sono ben compresi fin dall'inizio e difficilmente cambiano radicalmente durante lo sviluppo del sistema
- Oggigiorno usato quando fa parte di un progetto più vasto di ingegneria dei sistemi

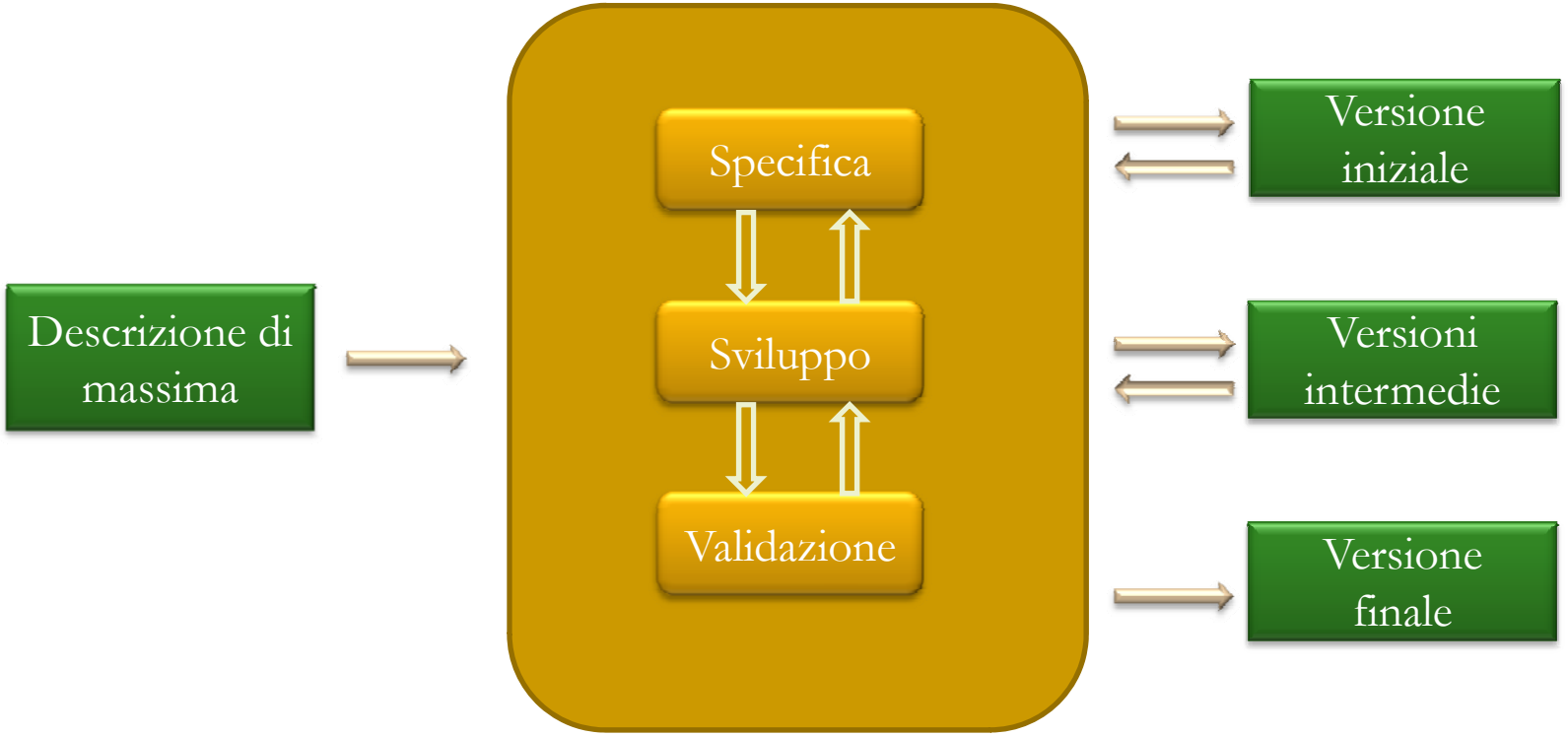
Modello evolutivo

- Si basa sullo sviluppo di una implementazione iniziale che viene fornita immediatamente agli utenti
  - Si perfeziona attraverso molte versioni fino ad ottenere un sistema adeguato
- Le attività di specifica, sviluppo e convalida sono intrecciate piuttosto che separate
  - Ci sono continui e veloci feedback tra le varie attività

Modello evolutivo

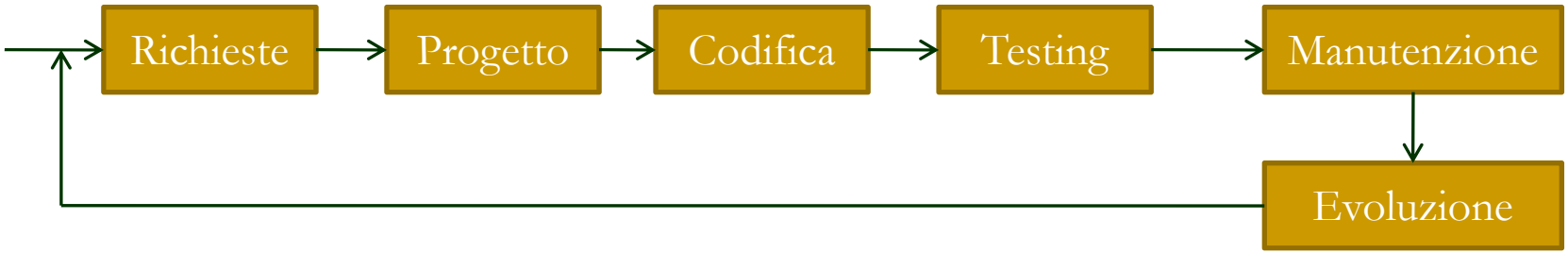
- Prototipazione di tipo evolutivo (sviluppo esplorativo)
  - L'obiettivo è lavorare con il cliente ed evolvere verso il sistema finale a partire da una specifica di massima.
  - Lo sviluppo inizia con le parti del sistema che **sono già ben** specificate, aggiungendo via via nuove caratteristiche
- Prototipazione di tipo usa e getta (throw-away)
  - L'obiettivo è capire i requisiti del sistema e quindi sviluppare una definizione migliore dei requisiti.
  - Il prototipo sperimenta le parti del sistema che **non sono** ancora ben comprese

Modello Evolutivo



Modello evolutivo: Sviluppo esplorativo

- Si comincia a sviluppare ciò che è ben chiaro del problema.

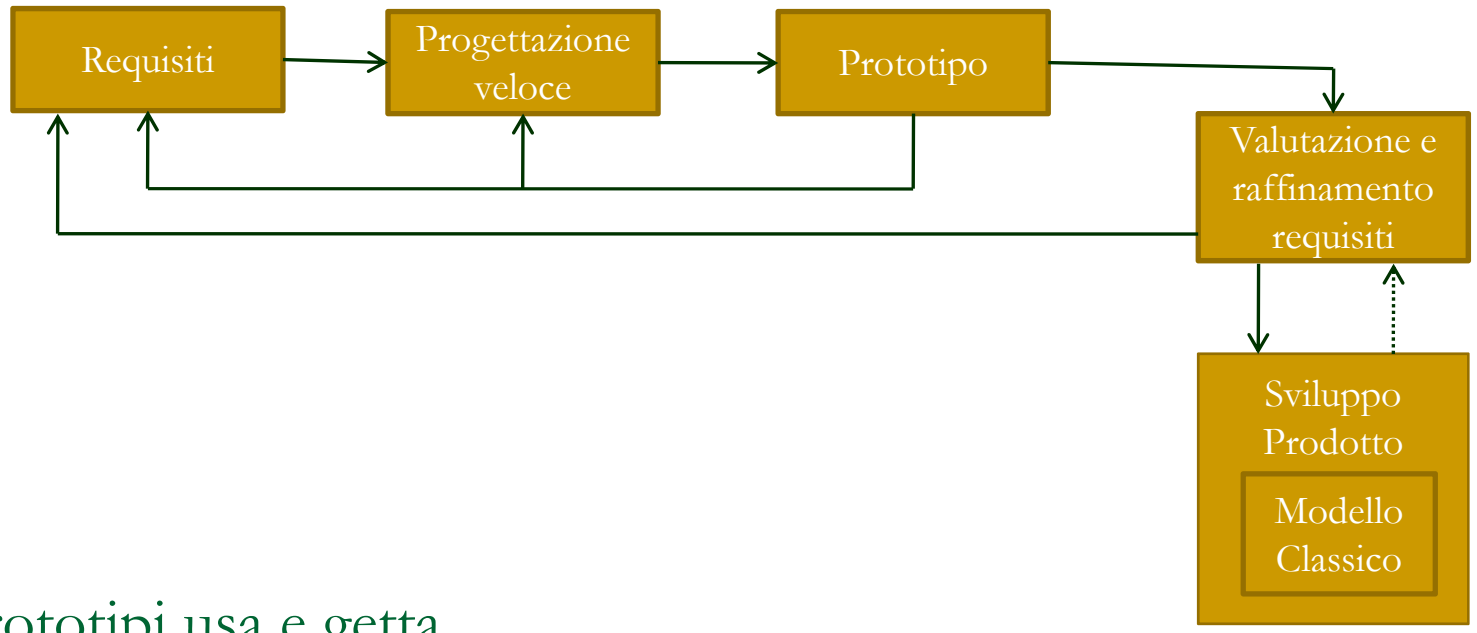


- Nella fase di evoluzione:
  - si analizza l'esperienza di uso del SW sul campo e si utilizza la maggiore conoscenza per definire nuovi obiettivi,
  - si determinano esigenze e nuove funzionalità emerse o non coperti in precedenza,
  - si riprende il ciclo dalla definizione dei requisiti alla messa in esercizio e manutenzione del nuovo SW nato dall'arricchimento del precedente.

Modelli basati su prototipo throw-away (1)

- Un prototipo per aiutare a comprendere i requisiti o per valutare la fattibilità di un approccio
- Realizzazione di una prima implementazione (prototipo), più o meno incompleta da considerare come una 'prova', con lo scopo di:
  - accertare la fattibilità del prodotto
  - validare i requisiti
- Il prototipo è un mezzo attraverso il quale si interagisce con il committente per:
  - accertarsi di aver ben compreso le sue richieste
  - specificare meglio tali richieste
  - valutare la fattibilità del prodotto
- Dopo la fase di utilizzo del prototipo, questo viene buttato (throw-away) e si passa alla produzione della versione definitiva del Sistema SW mediante un modello che, in generale, è di tipo waterfall

Modelli basati su prototipo throw-away (2)



- Prototipi usa e getta
  - Il prototipo è uno strumento di identificazione dei requisiti di utente; è incompleto, approssimativo, realizzato utilizzando parti già possedute.
  - Il prototipo deve essere buttato

Modello evolutivo: vantaggi

- Più efficace rispetto ad un approccio a cascata quando è necessario soddisfare le immediate necessità del cliente
  - Le specifiche si possono sviluppare in modo incrementale
    - Quando l'utente comprende meglio i suoi problemi la cosa si riflette sul sistema software

Modello evolutivo: svantaggi

- Da un punto di vista ingegneristico e manageriale
  - Il processo non è visibile: i manager devono avere consegne regolari per attivare i progressi
    - Non è economico produrre la documentazione che rifletta ogni versione del sistema quando esso è sviluppato velocemente
  - Sistemi strutturati male
    - I continui cambiamenti finiscono con il corrompere la struttura del software

Modello evolutivo: applicabilità

- Sviluppo di sistemi medio/piccoli (fino a 500.000 linee di codice)
- Meno utile quando:
  - Sviluppo di grandi sistemi per cui lavorano più team di sviluppo su parti diverse del sistema
    - Difficile definire un'architettura stabile per cui l'integrazione del prodotto del lavoro dei vari team è ardua
    - Preferibile usare un approccio misto
      - Sviluppo di un prototipo usa e getta per i requisiti incerti
      - Reimplementazione del sistema con un approccio più strutturato

Modelli di sviluppo basato su riuso

- Basati su un riuso sistematico, per sistemi ottenuti integrando componenti esistenti (sistemi COTS (commercial off-the shelf))
- C'è molto interesse intorno a questo approccio ma ci sono ancora poche esperienze
- Consente uno sviluppo rapido delle applicazioni
- Full ReuseModel (Basili, 1990)
  - prevede repository di componenti riusabili a diversi livelli di astrazione, prodotti durante le diverse fasi del ciclo di vita
    - specifiche, progetti, codice, test case, ...
  - durante lo sviluppo di un nuovo sistema:
    - riuso di componenti esistenti
    - popolamento delle repository con nuove componenti
- Particolarmente adatti per sviluppo di software object-oriented
  - Component based, Design Patterns, Web Services, etc...

Modelli di sviluppo basato su riuso

- Gli stadi iniziali di specifica dei requisiti e di convalida sono simili a quelli degli altri processi
- Gli stadi intermedi sono molto diversi
  - Analisi dei componenti
    - Data una specifica dei requisiti, si cercano i componenti per implementarla
  - Modifica dei requisiti
    - Si analizzano i requisiti usando le informazioni sui componenti scoperti e si modificano sulla base dei componenti disponibili. Se non è possibile fare modifiche si riesegue l'analisi dei componenti per trovare soluzioni laternative

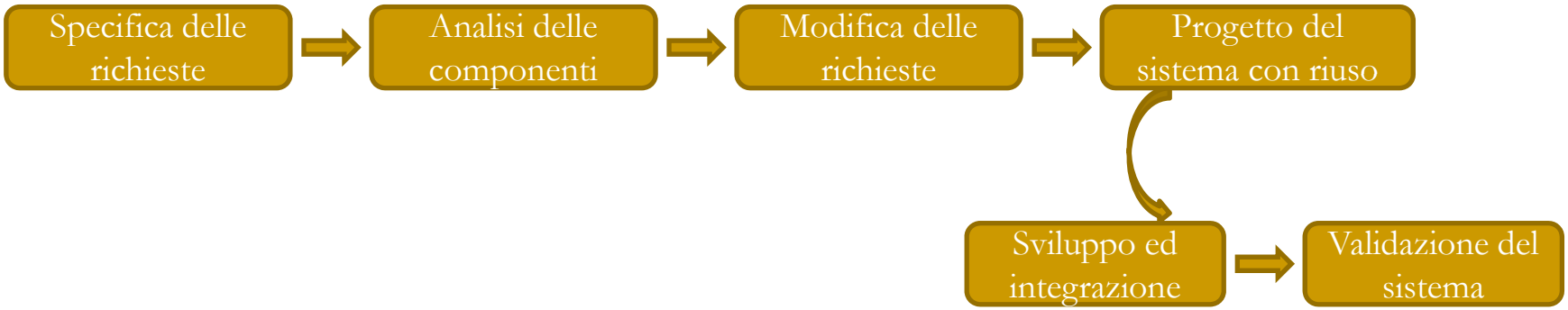
Modelli di sviluppo basato su riuso

- Progettazione del sistema con riuso
  - È progettata la struttura del sistema o utilizzata una esistente. Potrebbe essere necessario progettare alcuni software nuovi se non sono disponibili componenti riusabili
- Sviluppo e integrazione
  - Si sviluppa il software non acquisibile esternamente e si integrano i sistemi COTS e i componenti in modo da creare il nuovo sistema.

Modelli di sviluppo basato su riuso: vantaggi e svantaggi

- Vantaggi
  - Riduce la quantità di software da sviluppare
  - Riduce costi e rischi
  - Consegne veloci
- Svantaggi
  - Compromessi nei requisiti che portano a sistemi che non soddisfano pienamente le necessità dei clienti
  - Perdita del controllo sulla evoluzione del sistema
    - Le nuove versioni dei componenti riutilizzabili non sono sotto il controllo dell'organizzazione che li usa

Sviluppo basato su riuso



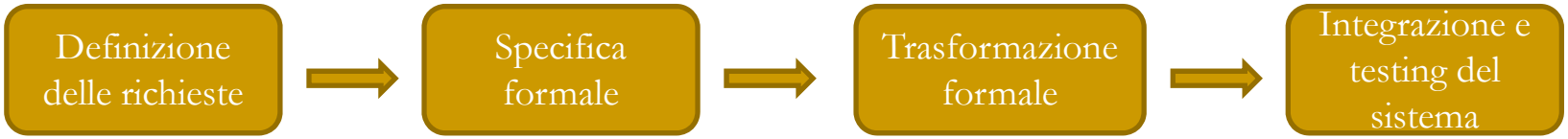
Modello trasformazionale

- Basato sulla trasformazione di una specifica matematica in un programma eseguibile attraverso trasformazioni eseguibili, che permettono di passare da una rappresentazione formale ad un'altra.
- Le trasformazioni devono preservare la correttezza. Questo garantisce che il programma soddisfi la specifica.
- Tra i più famosi: CleanRoom della IBM, Metodo B

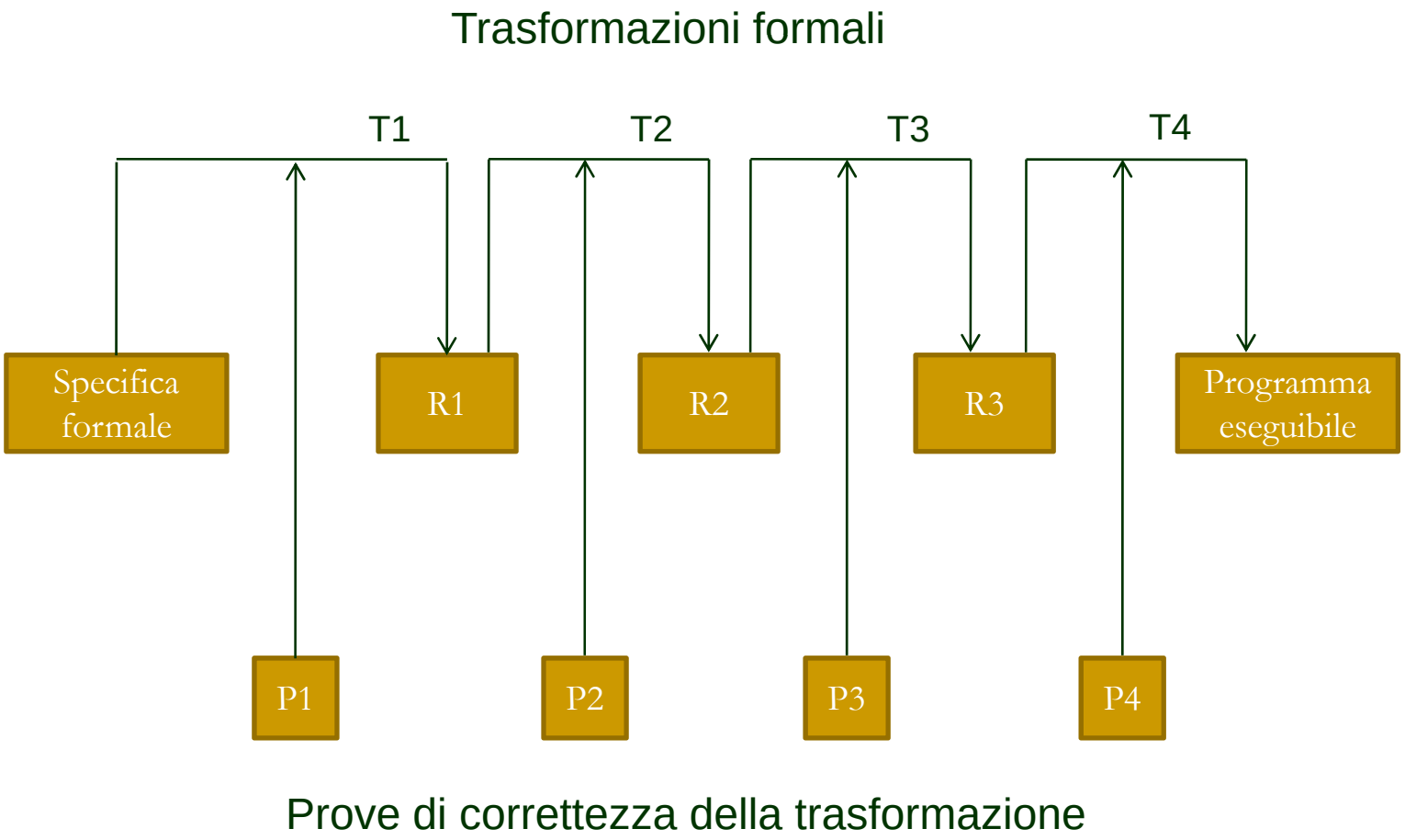
Modello trasformazionale

- Cleanroom
  - Ogni incremento al software è specificato formalmente e la sua specifica trasformata in implementazione
  - L'esattezza del software è dimostrata usando un approccio formale
    - Non è previsto alcun test per rilevarne i difetti
    - Il test del sistema consiste nella stima della sua attendibilità

Sviluppo formale dei sistemi



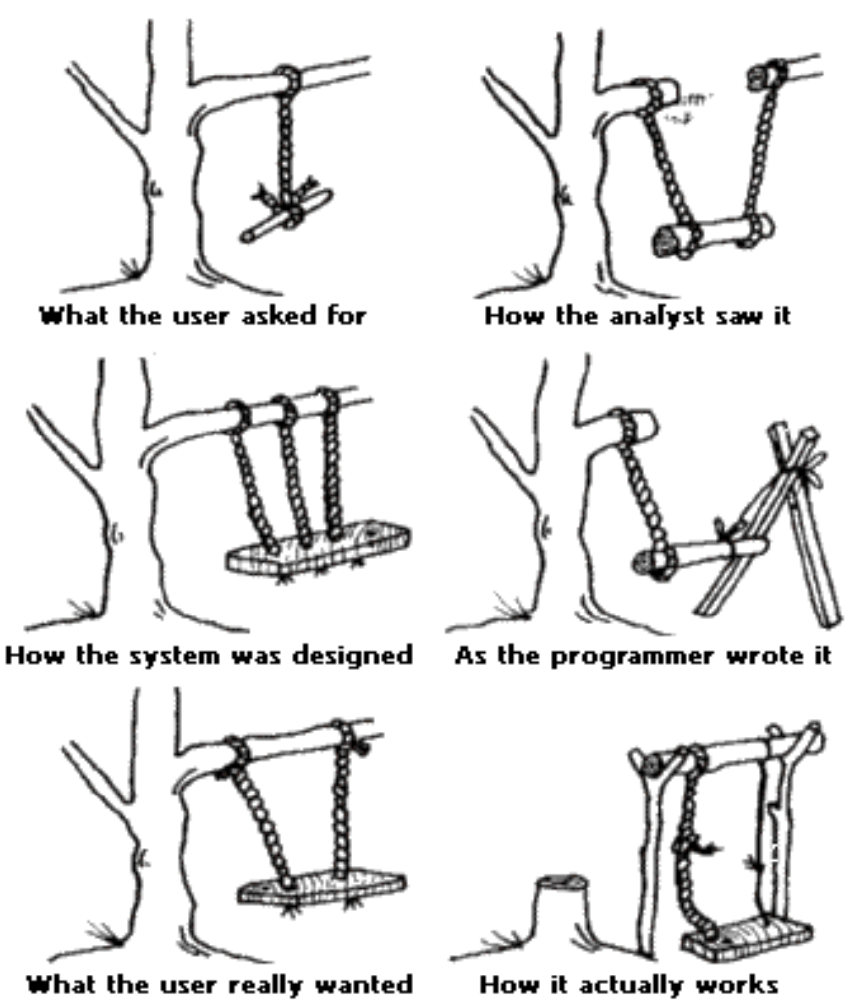
Modello Trasformazionale



Sviluppo formale

- Problemi
  - Richiede una conoscenza specializzata ed esperienza nell'applicazione della tecnica
  - Difficoltà nello specificare formalmente alcuni aspetti del sistema come per esempio le interfacce
  - Pochi esempi di applicazione per sistemi complessi
- Applicabilità
  - Uso di questo modello in altri modelli, per migliorare il processo di sviluppo di sistemi critici (specialmente quelli in cui la sicurezza è un aspetto importante)

Cicli di Processo



Cicli di processo

- I cambiamenti in tutti i grandi progetti software sono all'ordine del giorno
  - Cambiano i requisiti
  - Cambiano le priorità nella gestione
  - Cambiano i progetti e le implementazioni con l'evoluzione tecnologica
- Il processo software non è un processo unico
  - Le attività si ripetono regolarmente quando il sistema è rielaborato in risposta ai cambiamenti richiesti

Modelli ciclici

- Sono stati progettati due modelli di processo specificamente per il supporto dei cicli di processo
  - Consegna incrementale
    - Specifiche, progettazione e implementazione sono suddivisi in una serie di incrementi sviluppati una alla volta
  - Sviluppo a spirale
    - Lo sviluppo del sistema segue una spirale verso l'esterno a partire da una descrizione sommaria iniziale fino al sistema finale
- Risolvono la difficoltà a produrre l'intero sistema in una sola volta nel caso di grandi progetti SW (sia per problemi del produttore che del committente - quest'ultimo potrebbe non avere l'immediata disponibilità finanziaria necessaria per l'intero progetto)

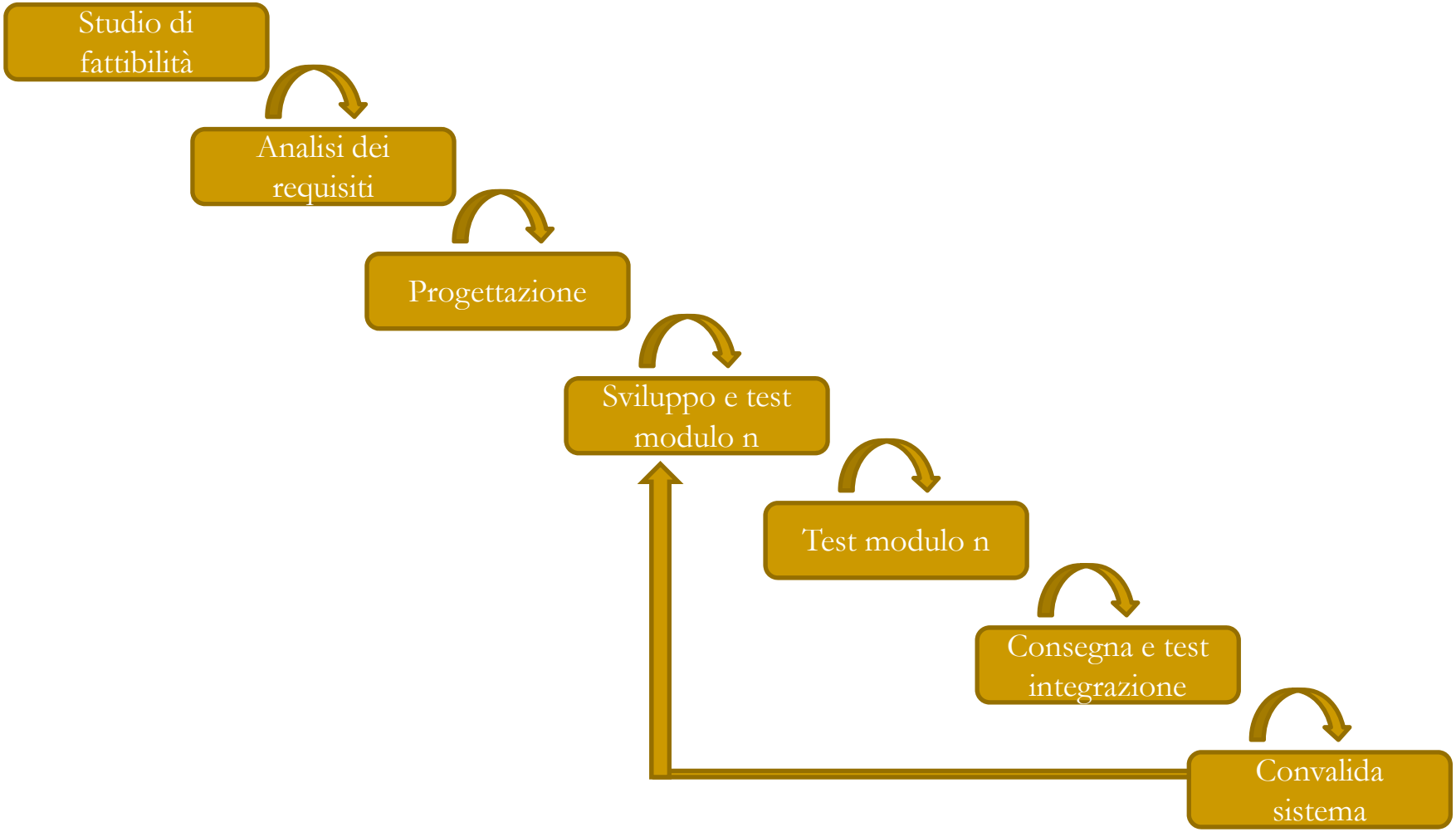
Modello a consegna incrementale

- La consegna incrementale è una via di mezzo tra modello a cascata e modello evolutivo e combina i vantaggi di tali modelli
  - Il cliente identifica a grandi linee i requisiti e i servizi che il sistema deve fornire, ordinandoli per importanza
  - È definito un numero di incrementi ognuno dei quali fornisce delle funzionalità del sistema
    - L'assegnazione dei servizi agli incrementi dipende dal priorità del servizio

Modello a consegna incrementale

- Identificati gli incrementi …
  - Si definiscono dettagliatamente i requisiti del primo incremento e parte lo sviluppo
  - Durante lo sviluppo viene svolta l'analisi dei requisiti per gli incrementi successivi
    - I cambiamenti per i requisiti dell'incremento in corso di sviluppo non vengono accettati
  - Quando un incremento è completato
    - Viene consegnato al cliente e messo in uso
      - Consegna rapida di alcune parti del sistema
      - Sperimentazione al fine di chiarire i requisiti per gli incrementi successivi e versioni successive dell'incremento corrente
  - Completati i nuovi incrementi
    - Si integrano con quelli esistenti in modo da migliorare le funzionalità del sistema ad ogni consegna

Modello a consegna incrementale



Modello a consegna incrementale: vantaggi

- I clienti non devono aspettare il completamento dell'intero sistema perché avvenga la consegna
  - Il primo incremento soddisfa i requisiti più critici in modo da utilizzare il software immediatamente
- I clienti usano i primi incrementi come prototipi
  - Acquisiscono esperienza per migliorare i requisiti degli incrementi seguenti
- Minor rischio di fallimento del progetto
- I servizi con maggiore priorità sono consegnati prima e quindi testati più intensamente
  - Minor rischio di fallimento nelle parti più importanti del software

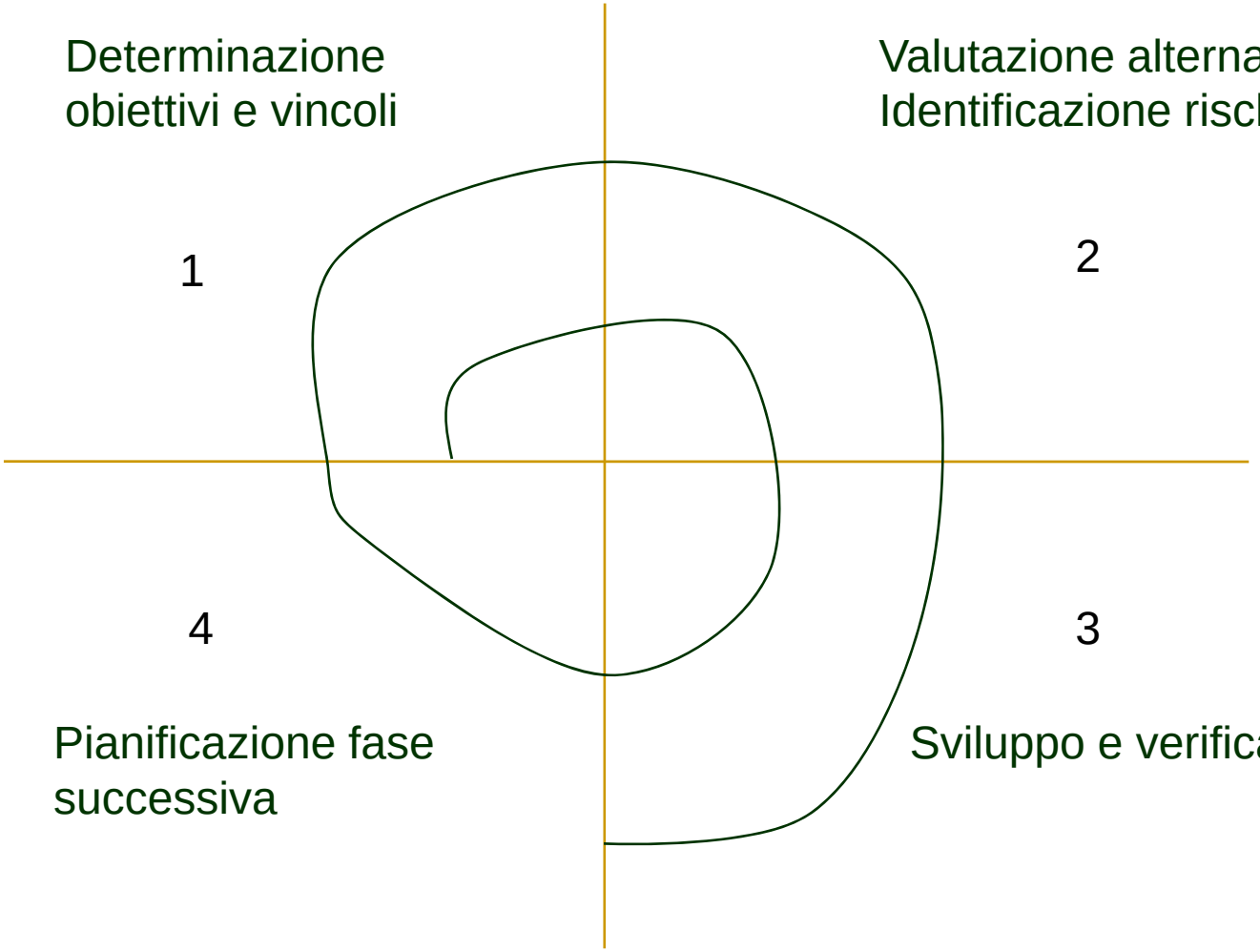
Modello a consegna incrementale: svantaggi

- Gli incrementi dovrebbero essere piccoli (20.000 righe di codice)
  - Difficile predisporre i requisiti del cliente in incrementi della giusta dimensione
- Difficile identificare le funzionalità comuni necessarie per tutti gli incrementi
  - Molti sistemi richiedono un insieme di funzionalità di base utilizzate da diverse parti del sistema.
    - I requisiti non sono definiti in dettaglio fino a che l'incremento non è implementato

Modello a spirale

- Il processo software viene rappresentato da una spirale
  - Ogni giro della spirale corrisponde ad una fase del processo
    - Il giro più interno si occupa della fattibilità del sistema
    - Il successivo della definizione dei requisiti
    - Il successivo della progettazione etc.
  - Ogni giro della spirale è diviso in quattro settori
    - Determinazione obiettivi, vincoli, alternative
    - Valutazione alternative, analisi dei rischi
    - Sviluppo, verifica e convalida
    - Pianificazione prossimo ciclo

Modello a spirale di Boehm



Rischi

- Rischi = fattori variabili che influiscono negativamente sulla riuscita di un progetto
  - Di tipo tecnico
  - Di tipo non tecnico
- Esempi:
  - personale non qualificato per le attività da effettuare
  - budget e pianificazione irrealistici
  - errori di definizione delle funzionalità
  - errori di definizione dell'interfaccia utente
  - funzionalità non richieste ma costose
  - instabilità dei requisiti
  - difetti in componenti sviluppati da terze parti
  - errori in attività svolte all'esterno dell'azienda
  - uso di tecnologie non consolidate o inaffidabili

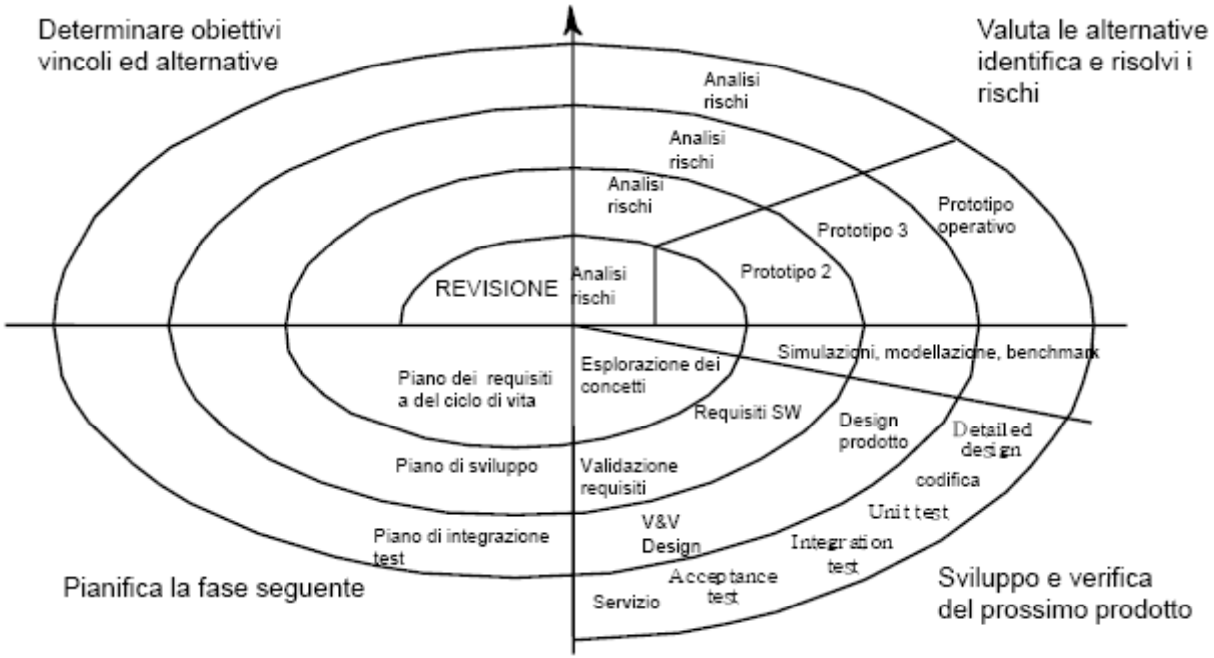
Gestione dei rischi

- Compito di chi gestisce (il manager) è minimizzare i rischi
- Il rischio è insito in tutte le attività umane ed è una misura dell'incertezza sul risultato dell'attività
- Alti rischi provocano ritardi e costi imprevisti
- Il rischio è collegato alla quantità e qualità delle informazioni disponibili: meno informazione si ha più alti sono i rischi

I modelli e la gestione dei rischi

- Cascata
  - alti rischi per sistemi nuovi, non familiari per problemi di specifica e progetto
  - Bassi rischi nello sviluppo di applicazioni familiari con tecnologie note
- Prototipazione
  - bassi rischi per le nuove applicazioni, specifica e sviluppo vanno di pari passo
  - alti rischi per la mancanza di un processo definito e visibile
- Trasformazionale
  - alti rischi per le tecnologie coinvolte e le professionalità richieste

Il modello a spirale tradizionale



Esempio

- Obiettivi
  - Migliorare la qualità del software in modo significativo
- Vincoli:
  1. In tre anni
  2. Senza grandi investimenti
  3. Senza un cambiamento radicale degli standard dell'azienda
- Alternative
  1. Riutilizzare software certificato già esistente
  2. Introdurre specifiche e verifiche formali
  3. Investire in prodotti di testing e di validazione

Esempio (cont.)

- Rischi
  - Migliorare la qualità del software può aumentare eccessivamente i costi
  - I nuovi metodi possono indurre il personale a licenziarsi
- Risoluzione dei rischi
  1. Studio della letteratura esistente
  2. Avvio di un progetto pilota
  3. Analisi delle componenti potenzialmente riutilizzabili
  4. Valutazione degli strumenti di supporto già esistenti
  5. Addestramento e rimotivazione del personale

Esempio (cont.)

- Risultati
  1. I miglioramenti sono difficili da quantificare per la scarsa esperienza nell'utilizzo di metodi formali
  2. Gli strumenti di supporto disponibili sono insufficienti rispetto allo standard dei sistemi di sviluppo dell'azienda
  3. Componenti software riusabili, ma scarso contributo degli strumenti di supporto alla riusabilità
- Pianificazione della fase successiva
  - Finanziare una fase di studio di altri 18 mesi
  - Studiare in maggior dettaglio le opzioni di riuso del software
  - Sviluppare strumenti di supporto al riuso di software
  - Esplorare uno schema di certificazione delle componenti

Modello a spirale: vantaggi e svantaggi

- Vantaggi
  - Rende esplicita la gestione dei rischi
  - Focalizza l'attenzione sul riuso
  - Aiuta a determinare errori nelle fasi iniziali
  - Obbliga a considerare gli aspetti di qualità
  - Integra sviluppo e manutenzione
  - Costituisce un framework di sviluppo hardware/software
- Svantaggi
  - Per contratto di solito si specifica a priori il modello di processo e i "deliverables". Lo sviluppo richiede un contratto nel contratto: vanno specificati vincoli, modello di processo, tempi di consegna e artefatti da consegnare
  - Richiede persone in grado di valutare i rischi
  - Per poter essere usato deve essere adattato alla realtà aziendale e/o al team

Modelli di processo e domini applicativi

- Per sistemi o sottosistemi di cui si ha una buona conoscenza si può adottare il modello a cascata: la fase di analisi dei rischi ha costi limitati
- Sistemi critici per le persone o cose (safety critical) con requisiti stabili, possono essere sviluppati con approcci trasformazionali
- Zone non completamente specificate, interfacce utente possono impiegare il modello a prototipi