# Richiami di Ingegneria del Software

**Roma, a.a. 2005-2006**



Docente: Dr. Anna Rita Laurenzi, Ph.D. - Docente aggiunto: Dr. Ercole Colonese

# Indice

- Principali concetti dell'ingegneria del software
- Processo di produzione software
  - Caratteristiche principali
- Modelli di Processo
  - Cascata
  - Evolutivo
  - Prototipale
  - Rapid Access Development (RAD)
  - Trasformazionale
  - Ad assemblaggio di componenti
  - Incrementale
  - A spirale di Boehm
  - Processo unificato

# Ingegneria del software

- 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.*

# Ingegneria del software

- L'Ingegneria del Software può essere vista come una parte dell'Ingegneria di Sistema.

- L'Ingegneria di Sistema ha come oggetto tutti gli aspetti dello sviluppo di un sistema basato su computers, inclusi gli aspetti hardware, software e di processo.

# Ingegneria del software

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***

# Come nasce l'ing. del software

- Uno dei più diffusi modi di lavorare per produrre software,è quello del programmatore che, in solitaria, dopo avere ascoltato le esigenze del proprio committente - utilizzatore ("utente"), si tuffa direttamente a scrivere il codice che le soddisferà. E' un modo di lavorare sbrigativo e senza troppe pretese, ma può essere efficace, a patto che:
  - il problema sia molto semplice
  - l'utente formuli il problema in modo chiaro, ed il programmatore capisca con precisione ciò che l'utente si aspetta
  - il programmatore possa effettivamente lavorare da solo, senza la collaborazione di altri

- Se però una di queste condizioni viene meno, il metodo sbrigativo non funziona.

# Come nasce l'ing. del sw.

- Ne consegue che, prima di mettersi a scrivere il codice, è opportuno effettuare un'analisi dei requisiti che il sistema dovrà soddisfare, e una progettazione (disegno) dell'architettura tecnica del sistema

- E' in atto da decenni un lavoro enorme di definizione di standard e di strategie per far raggiungere all'industria del software un livello di maturità accettabile a cura del ISO e del SEI, lavoro sponsorizzato dal governo americano, dalle maggiori aziende statunitensi, dai progetti finanziati dalla Comunità Europea

  – ISO=Organizzazione Internazionale per la Standardizzazione
  – SEI= Software Engineering Institute

# Ingegneria del software

- La tecnologia, in quanto si basa su un'infrastruttura tecnologica
- Le tecniche, gli algoritmi o le procedure necessarie per trasformare un prodotto in input per ottenere quello in output
- I metodi, ossia un insieme di tecniche che le consentono di cooperare per costruire un prodotto con predefiniti requisiti
- I processi software, ossia un insieme di attività che costruiscono prodotti intermedi e finali che sostituiscono il software desiderato, utilizzando uno o più metodi
- La gestione dei progetti software, ossia tecniche manageriali per gestire le risorse disponibili, onde eseguire le attività dei processi, minimizzando i tempi ed i costi.

# I diversi tipi di software

- Software di sistema
  - Collezione di programmi al servizio di altri programmi (compilatori, editor, strumenti per la gestione di file...)
- Software real-time
  - E' il software che sorveglia, analizza, controlla eventi esterni mentre avvengono
- Software gestionale
  - Elaborazione di dati aziendali
- Software scientifico e per l'ingegneria
  - Algoritmi di calcolo intensivo (astronomia, vulcanologia, biologia molecolare, terremoti...)

# I diversi tipi di software

- Software embedded
  - Risiede generalmente in memorie per sola lettura e ha lo scopo di controllare prodotti e sistemi di consumo o industriali
- Software per i personal computer
  - Elaborazione testi, fogli elettronici, grafica, programmi multimediali...
- Software basato sul Web
  - CGI, PHP, JSP...
- Software per l'intelligenza artificiale
  - Algoritmi non numerici (euristici) per la risoluzione di problemi complessi

# Caratteristiche del software: ISO 9126

- Funzionalità
- Affidabilità
- Usabilità
- Efficienza
- Manutenibilità
- Portabilità

# Caratteristiche del software

- **Funzionalità**: Un Software è considerato funzionale nella misura in cui le procedure in esso contenute coincidono con le funzioni richieste.
  - In altre parole la funzionalità esprime la conformità del software alle richieste effettuate dall'utente.

- **Affidabilità**: Un software è ritenuto affidabile quando è in grado di mantenere il livello di prestazioni sotto determinate condizioni e per un determinato periodo di tempo.
  - Un software è considerato affidabile nel momento in cui non incorre in errori quando sottoposto a condizioni di stress e per un certo periodo di tempo.

# Caratteristiche del software

- **Usabilità**: Un software è considerato usabile in proporzione alla facilità con cui gli utenti operano per sfruttare appieno le funzionalità che il software realizza.
  - Generalmente il termine utenti può essere riferito direttamente agli utenti di un software interattivo, per cui gli utenti possono essere operatori al terminale, utenti finali o utenti indiretti, ovvero tutti coloro che sono sotto l'influenza o dipendono dall'uso del software.

# Caratteristiche del software

- **Efficienza**: L'efficienza di un software è proporzionale al rapporto tra il livello generale di prestazioni del software e l'ammontare delle risorse necessarie al suo funzionamento.
  - L'efficienza è una qualità legata allo sfruttamento delle risorse computazionali in senso vasto, laddove anche il tempo impiegato per la computazione è considerata come una risorsa.

- **Manutenibilità**: È l'attitudine del software ad essere modificato.
  - Le modifiche possono includere modifiche correttive, modifiche migliorative o modifiche adattive per mantenere il software al passo con i cambiamenti dell'ambiente in cui opera o per rendere il software conforme alle nuove specifiche prodotte per esso.

# Caratteristiche del software

- **Portabilità**: Un software è considerato portabile quando è possibile trasferirlo da un ambiente ad un altro.
  - Con ambiente si intende sia l'ambiente informatico composto dal sistema operativo, dall'hardware, dal software di supporto, sia il contesto relativo all'utenza in cui il software si inserisce.

# Sottocaratteristiche del software

- Alle caratteristiche principali elencate in precedenza, la normativa **ISO 9126** aggiunge un insieme di sotto caratteristiche così definite

# Le sottocaratteristiche del software

## Funzionalità

- **Utilità**: attributi del software che influenzano la presenza e l'appropriatezza dell'insieme di funzioni per uno specifico obiettivo.
- **Accuratezza**: attributi del software che concernono la generazione di risultati o azioni che siano corrette.
- **Interoperabilità**: attributi del software che influenzano la capacità di interagire con specifici sistemi.
- **Aderenza**: attributi del software che rendono il software aderente agli standard, convenzioni o regolamenti legislativi applicabili all'applicativo.
- **Sicurezza**: attributi del software che permettono di prevenire accessi non autorizzati, siano essi accidentali o deliberati, ai dati o ai programmi.

# Le sottocaratteristiche del software

## Affidabilità

- **Maturità**: attributi del software che influenzano la frequenza dei fallimenti dovuti a errori nel software.
- **Fault Tolerance**: attributi del software che permettono al software di mantenere uno specificato livello di prestazioni in caso di errore del software.
- **Recuperabilità**: attributi del software che permettono al software di ristabilire il suo livello di prestazioni e di ricuperare i dati persi in occasione di errori.

# Le sottocaratteristiche del software

## Usabilità

- **Comprensibilità**: attributi del software che influenzano lo sforzo compiuto dall'utente nel riconoscere i concetti logici e la loro applicabilità all'interno del software.
- **Apprendimento**: attributi del software che influenzano lo sforzo compiuto dall'utente nell'imparare ad usare l'applicativo, come per esempio le operazioni di controllo, di input o di output.
- **Operabilità**: attributi del software che influenzano lo sforzo compiuto dall'utente nel controllo delle capacità del software.

# Le sottocaratteristiche del software

## Efficienza

- **Comportamento in tempo**: attributi del software che influenzano i tempi di risposta e di esecuzione delle funzionalità del software.
- **Comportamento in risorse**: attributi del software che influenzano l'ammontare e l'utilizzo nel tempo di risorse durante l'esecuzione delle funzionalità del software.

# Le sottocaratteristiche del software

## Manutenibilità

- **Analizzabilità**: attributi del software che influenzano lo sforzo compiuto per le fasi di diagnostica delle cause degli errori, e per l'identificazione delle parti di software da modificare.
- **Modificabilità**: attributi del software che influenzano lo sforzo necessario per le modifiche, la rimozione degli errori o per i cambi ambientali.
- **Stabilità**: attributi del software che influenzano i rischi legati ad eventi inaspettati o modifiche.
- **Verificabilità**: attributi del software che influenzano lo sforzo necessario alla validazione di un software modificato.

# Le sottocaratteristiche del software

## Portabilità

- **Adattabilità**: attributi del software che influenzano la possibilità di adattamento a differenti ambienti senza ricorrere ad altre azioni che quelle previste per il software considerato.
- **Installabilità**: attributi del software che influenzano lo sforzo necessario alla installazione del software in uno specifico ambiente.
- **Conformità**: attributi del software che rendono il software aderente agli standard o alle convenzioni relative alla portabilità.
- **Sostituibilità**: attributi del software relativi alla possibilità di sostituire altri specifici software all'interno del loro ambiente.

# Sistema Software

- È un sistema applicativo di alto livello di qualità, completo di tutta la documentazione che descrive: i requisiti, la progettazione che spiega la sua struttura e le decisioni che hanno giustificato la sua strutturazione. Essi devono avere, per ripagare gli alti costi di produzione, un largo bacino di utenza anche con piattaforme diverse, pertanto: la utilizzabilità e la portabilità sono caratteristiche chiavi.

# Sistema Software

- Un **sistema software semplice** è:
  - (quasi) completamente specificato da un insieme piccolo di comportamenti
  - comprensibile nella sua totalità da una sola persona
  - convenientemente sostituibile da un sistema nuovo e diverso quando sia necessario modificarlo o estenderne le funzionalità

- Un **sistema software complesso** (industrial-strength) è:
  - specificato in termini di un insieme ricco di comportamenti
  - difficile o impossibile da comprendere in tutti i suoi aspetti per un singolo individuo
  - un sistema che non possiamo permetterci di buttare via, e che viene pertanto modificato a costo di tenere in vita sistemi di sviluppo obsoleti per il suo solo mantenimento

# Il processo software

Il concetto di processo software o *ciclo di vita* del software è stato introdotto alla fine degli anni '60 con l'obiettivo di individuare e fissare tutti i principi ingegneristici per la produzione di sistemi complessi.

Il processo di produzione software è un insieme di attività il cui fine è lo sviluppo oppure la modifica di un prodotto software

# Attività generiche in un processo software

- **Attività portanti**
  - Fase di definizione
    - Strutturazione del sistema o delle informazioni
    - Pianificazione del progetto
    - Analisi dei requisiti (definizione dei requisiti)
  - Fase di sviluppo
    - Progettazione del software (Design)
    - Generazione di codice
    - Collaudo del software
  - Fase di manutenzione
- **Attività ausiliarie**
  - Controllo e guida del progetto
  - Revisioni tecniche formali
  - Assicurazione di qualità del software
  - Gestione delle configurazioni software
  - Preparazione e produzione dei documenti
  - Gestione della riusabilità
  - Misurazioni
  - Gestione dei rischi

# Il processo software o ciclo di vita

- Le fasi precedenti, opportunamente collegate tra loro, costituiscono appunto **il ciclo di vita del software o processo software.**
- Esistono diversi cicli di vita o processi software a seconda delle interazioni delle varie fasi/attività tra loro.
- Il modello base, e quello anche più diffuso, è detto a cascata (in inglese *waterfall*) perché prevede di eseguire le varie fasi rigorosamente in sequenza.

# Caratteristiche del processo

- ***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)

# Caratteristiche del processo

- ***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`)

# Problemi nel processo software

- Specifiche incomplete/incoerenti
- Mancanza di distinzione tra specifica, progettazione e implementazione
- Assenza di un sistema di validazione
- Il software non si consuma: la manutenzione non significa riparare alcune componenti rotte, ma modificare il prodotto rispetto a nuove esigenze

# Costi nel processo di produzione

- Circa il 60% dei costi è legato allo sviluppo, il 40% sono costi per la verifica e validazione (testing).
- I costi variano a seconda del tipo di sistema che deve essere sviluppato e da requisiti quali le prestazioni o l'affidabilità del sistema.
- La distribuzione dei costi nelle varie fasi del processo di produzione del software dipende dal modello di processo.

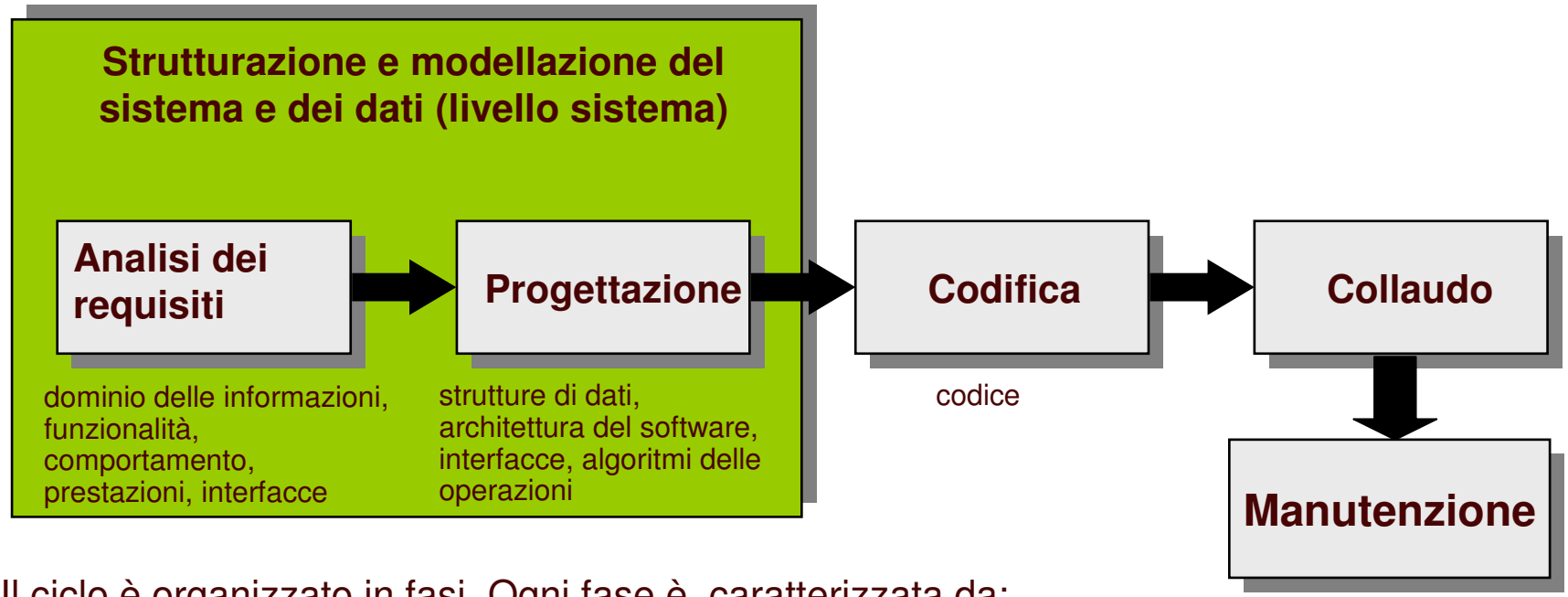
# Scelta del modello del processo

- Elementi che influiscono sulla scelta del modello del processo:
  - la natura del progetto e dell'applicazione
  - le competenze specifiche e l'esperienza acquisita in progetti precedenti dai membri del team di sviluppo
  - i metodi e strumenti che si vogliono utilizzare
  - i controlli e prodotti richiesti.

# Modelli di processo

- Cascata
- Evolutivo
- Prototipale
- Rapid Access Development (RAD)
- Trasformazionale
- Ad assemblaggio di componenti
- Incrementale
- A spirale di Boehm
- Processo unificato

# Modello “a cascata”




Il ciclo è organizzato in fasi. Ogni fase è caratterizzata da:

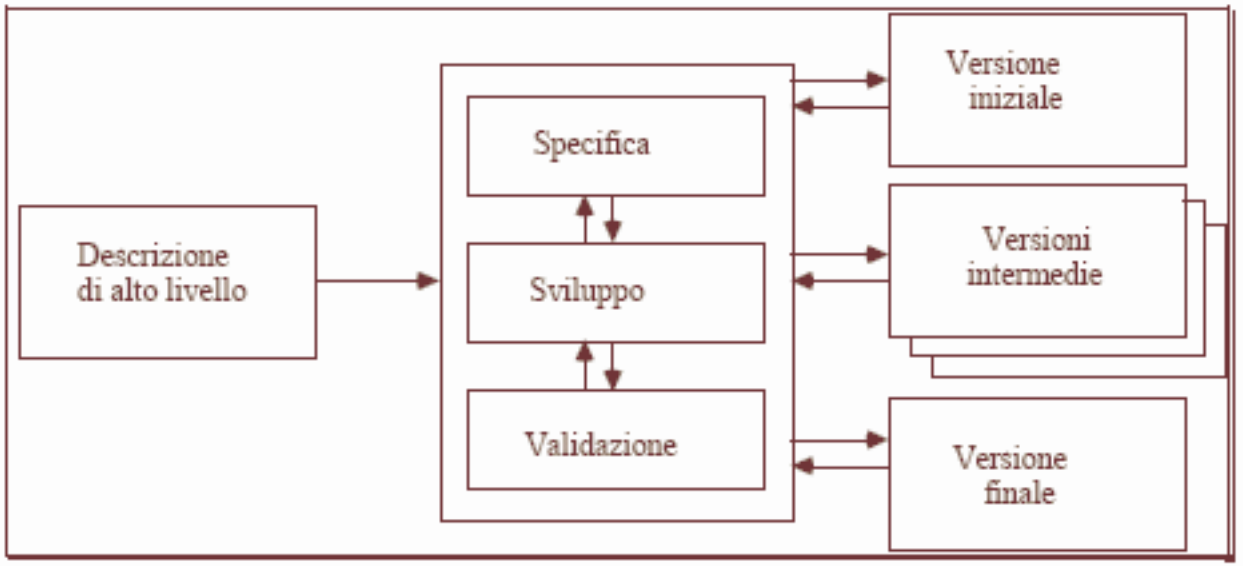
1. attività (tasks),
2. prodotti (deliverables), -sono prodotti di input per la fase successiva, perciò vengono congelati”
3. controlli e misure(V&V,Qualità) La fine di ogni fase è un punto rilevante per il progetto (milestones)

# Problemi del modello “a cascata”

- Difficile operare cambiamenti una volta che il processo è in corso
- I progetti reali non si conformano allo schema sequenziale
- Molte volte l’analista non comprende il dominio del problema e questo fa si che il dominio applicativo è instabile, per cui non è possibile definire esattamente i requisiti a scapito della correttezza e completezza dei requisiti
- Spesso si verificano “Stati bloccanti” per colpa della sincronizzazione tra le attività o tra i membri del team
- Il software è disponibile solo alla fine del processo
- Le attività sono sequenziali, quindi un imprevisto in una di esse si ripercuote su tutte

# Modelli evolutivi: schema

- 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

# Tipi di modelli evolutivi

- Prototipazione “throw-away”
- Programmazione “esplorativa”

# 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

# 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

# Applicabilità del modello evolutivo

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

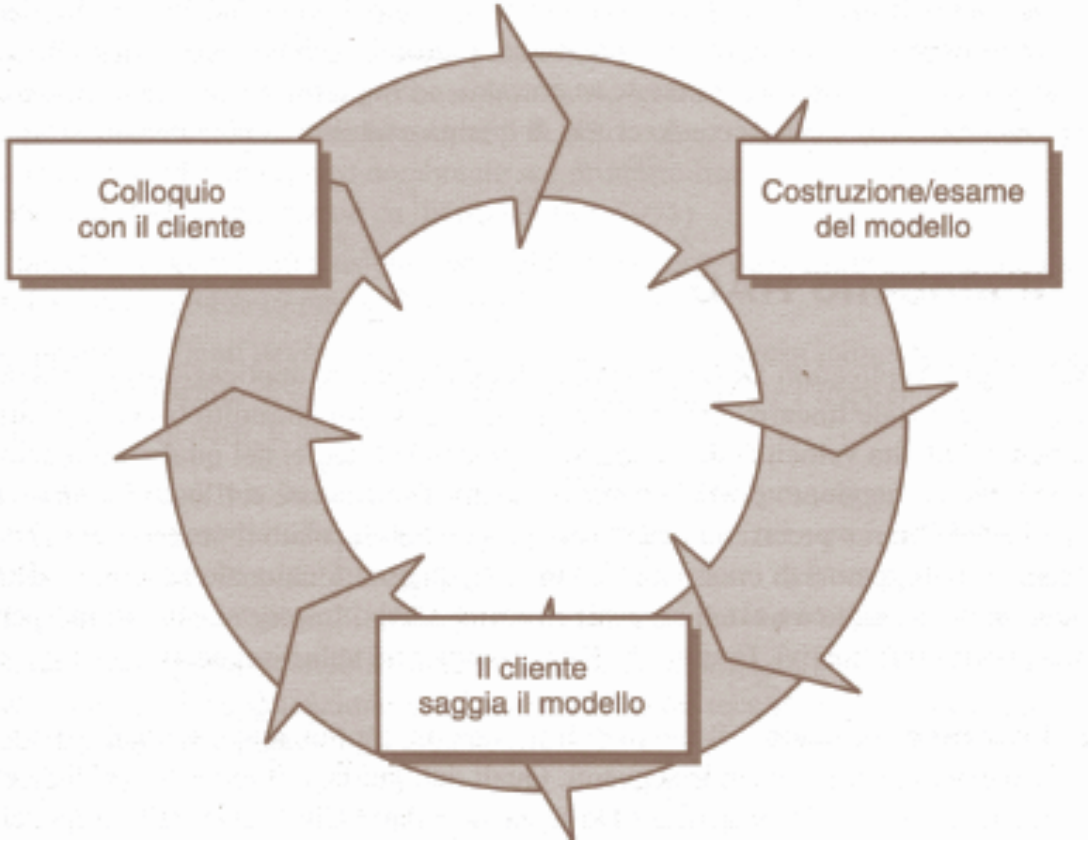
# Limiti dei modelli evolutivi

- 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

# Modello di Prototipazione

Questo modello di processo, tende a superare la comunicazione tra committente e analista nella definizione e validazione dei requisiti



# Passi del modello di Prototipazione

- Si realizza il prototipo e si sottopone alla visione dell'utilizzatore
- Questi, visionandolo rileva i difetti rispetto alle sue aspettative
- Il prototipo viene modificato e poi rivisto dall'utente
- Terminato il prototipo, questo costituisce la rappresentazione formale dei requisiti che realizza

# Svantaggi del modello di Prototipazione

- Valutazione di caratteristiche specifiche. Molto spesso l'utente valuta caratteristiche differenti da quelle per cui esso è stato costituito

- Utilizzo esteso del prototipo: quando il prototipo è operativo, gli sviluppatori tendono ad utilizzarlo come base per la costruzione dell'applicazione finale, trascurando le carenze costruttive

# Che fare del prototipo alla fine?

Nella maggior parte dei progetti, il sistema realizzato per primo è a malapena utilizzabile. Può risultare troppo lento, troppo grande, di uso complicato o tutte e tre le cose. Non c'è altro da fare che ricominciare da capo, facendo tesoro dell'esperienza e realizzare una nuova versione, nella quale i problemi siano risolti... Quando si applica una nuova concezione o una nuova tecnologia, è inevitabile costruire un sistema pilota destinato a essere cestinato, perché anche con la migliore pianificazione non si può prevedere tutto e centrare il bersaglio al primo colpo. La questione non è dunque se costruire un sistema pilota e poi gettarlo. Ciò è inevitabile. La questione è se pianificare in anticipo di costruire un oggetto destinato a essere gettato, oppure promettere al cliente di consegnargli quell'oggetto

Frederick P. Brooks, Jr.
"The Mythical Man-Month"

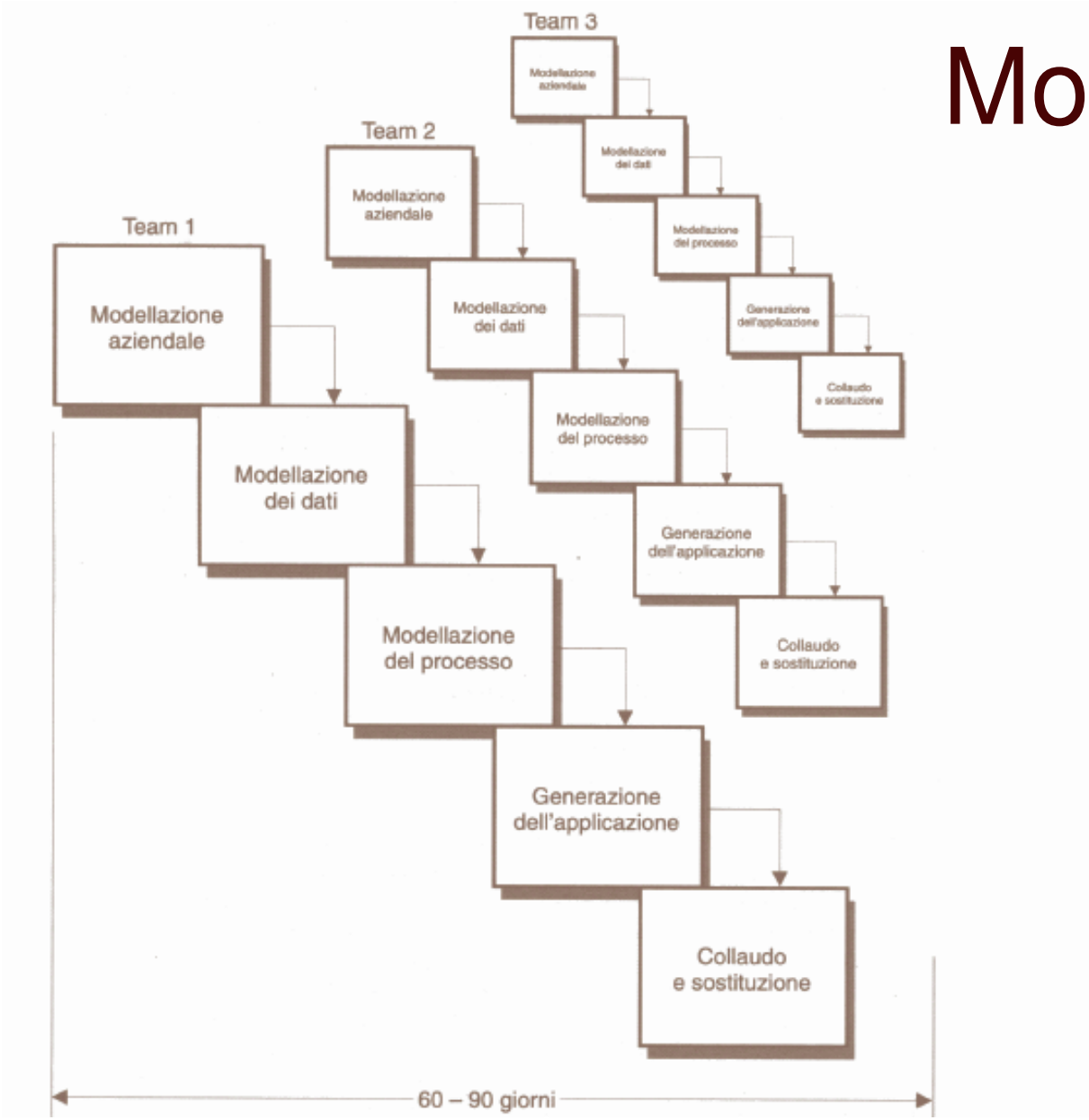
# Modello Rapid Application Development (RAD)

- E' un modello sequenziale lineare che punta ad un ciclo di sviluppo molto breve
- L'obbiettivo di uno sviluppo rapido è raggiunto mediante il riuso di componenti
- Quando applicarlo
  - Quando i requisiti sono chiari
  - Quando il processo di sviluppo è ben vincolato
- Nei casi in cui si può applicare, il modello RAD può portare allo sviluppo del software in tempi brevi rispetto al modello a cascata classico

# Modello RAD

- E' adatto alle situazioni in cui il sistema da costruire può essere partizionato facilmente fin dall'inizio
- Ogni parte deve poter essere sviluppata indipendentemente dalle altre, e in tempi brevi (<3 mesi)

# Modello RAD


Team 3
Modellazione aziendale
Modellazione dei dati
Modellazione del processo
Generazione dell'applicazione
Collaudo e sostituzione
Team 2
Modellazione aziendale
Modellazione dei dati
Modellazione del processo
Generazione dell'applicazione
Collaudo e sostituzione
Team 1
Modellazione aziendale
Modellazione dei dati
Modellazione del processo
Generazione dell'applicazione
Collaudo e sostituzione
60 – 90 giorni

# Attività del modello RAD

- **Modello dei dati**: Struttura dei dati così come è capita da progettista dopo un rapido colloquio con l'analista
- **Modello delle funzioni**: Struttura delle funzioni scaturite dallo stesso colloquio con l'analista
- **Generazione dell'applicazione**: Generazione automatica dell'applicazione, utilizzando anche componenti già operative

# Fasi del modello RAD

- **Validazione**: il committente valida che l'applicazione costruita soddisfi le sue aspettative e dichiara i requisiti che non trova nell'applicazione
- **Raffinamento dell'applicazione**: Rifinitura dell'applicazione per soddisfare gli scostamenti rilevati nella validazione
- **Collaudo dell'applicazione**: prova del corretto funzionamento della applicazione e del soddisfacimento di tutti i requisiti. Molte componenti sono state riutilizzate da altri sistemi
- **Ricostruzione della documentazione**: ricostruzione della documentazione necessaria per rendere completo il software secondo il suo sistema di riferimento
- **Manutenzione**: come per gli altri processi

# Limiti del modello RAD

- Per progetti di grandi dimensioni (purché partizionabili) è necessario disporre di risorse umane sufficienti
- Sviluppatori e clienti devono impegnarsi a completare il sistema in tempi rapidi
- RAD è inadatto quando:
  - ...Il sistema non è partizionabile
  - ...Le prestazioni sono un aspetto cruciale e dipendono da interfacce appositamente definite
  - ...Si sviluppano sistemi con tecnologie innovative, con alto rischio di incontrare problemi strada facendo

# Problemi del modello RAD

- **Competenze dell'analista e del progettista**: L'analista deve conoscere bene il dominio dell'applicazione e deve sapere utilizzare l'ambiente RAD, in modo che quanto viene prodotto sia sufficientemente vicino a quello che si aspetta l'utente. Altrimenti, il raffinamento dell'applicazione richiederebbe molti ricicli e sarebbe molto oneroso

- **Integrabilità**: le componenti prodotte devono essere integrabili con i software già operativi e con quelli generati da altri team e che sono parti di uno stesso software

# Modello Trasformazionale

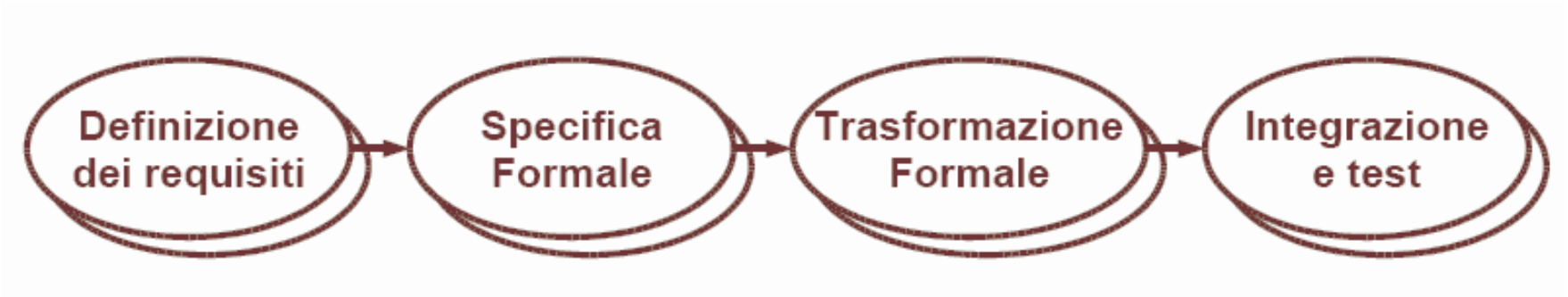
- Si basa sulla trasformazione di specifiche matematiche di un sistema in differenti rappresentazioni
- Le trasformazioni devono preservare la correttezza
  - In tal modo è banale verificare che il prodotto finale soddisfa la specifica

# Concetti alla base del modello

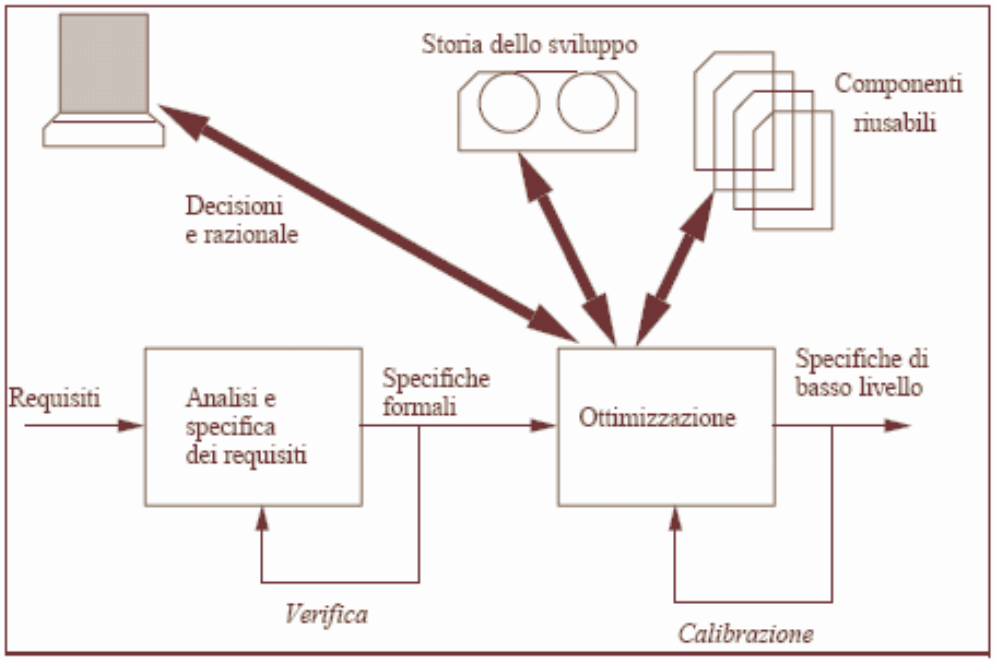
**prototipazione** di tipo evolutivo
deriva automaticamente
dalla eseguibilita` delle
specifiche formali

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

# Fasi del modello trasformazionale

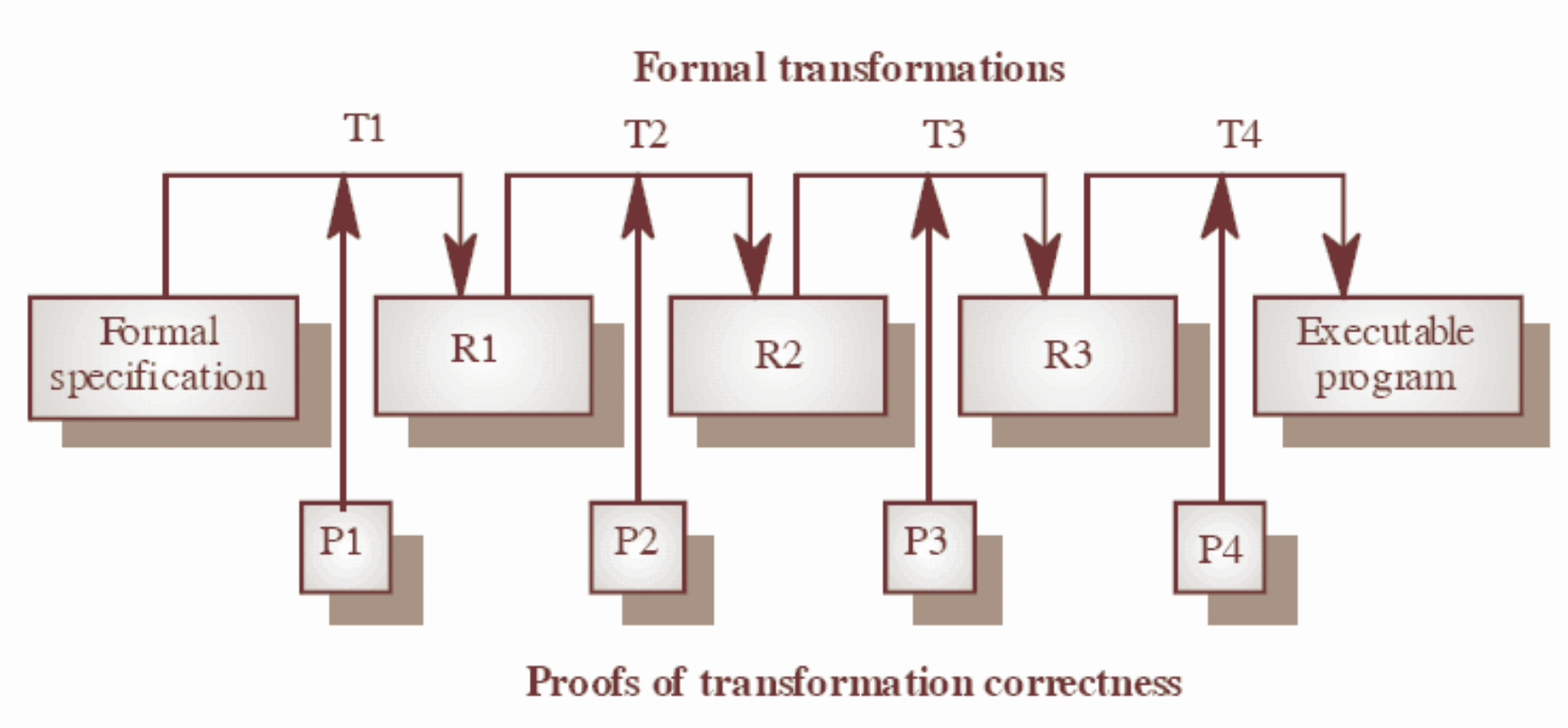


# Dettaglio delle fasi




- ***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)

# Trasformazioni formali



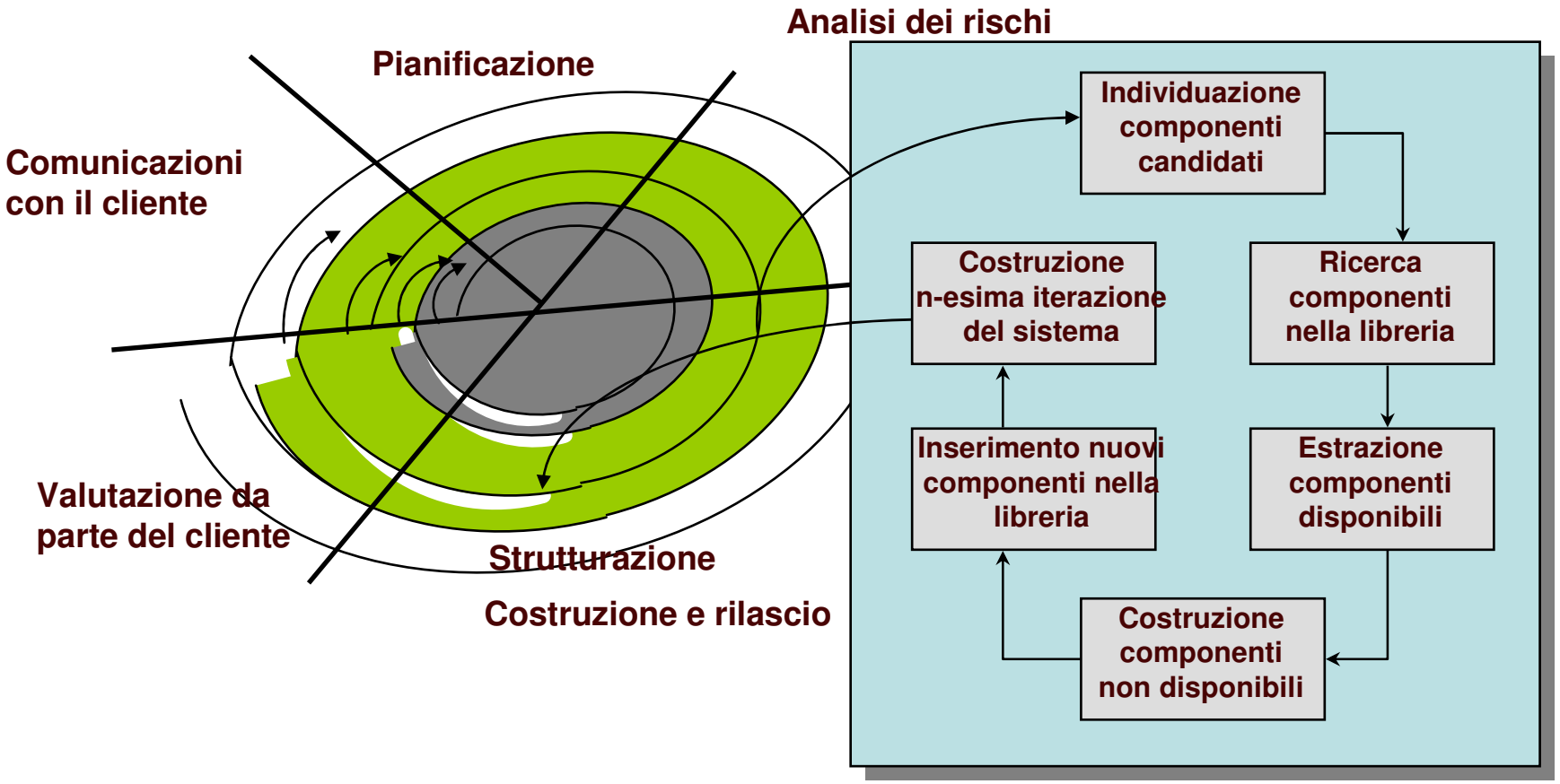
# Limiti e applicabilità

- Problemi
  - Richiede conoscenze specializzate e addestramento per essere applicato
  - Certe parti del sistema (es. interfaccia utente) sono difficili da specificare formalmente
- Quando applicarlo
  - Sistemi critici, in cui sicurezza o affidabilità sono essenziali e devono essere *provabilmente* raggiunti

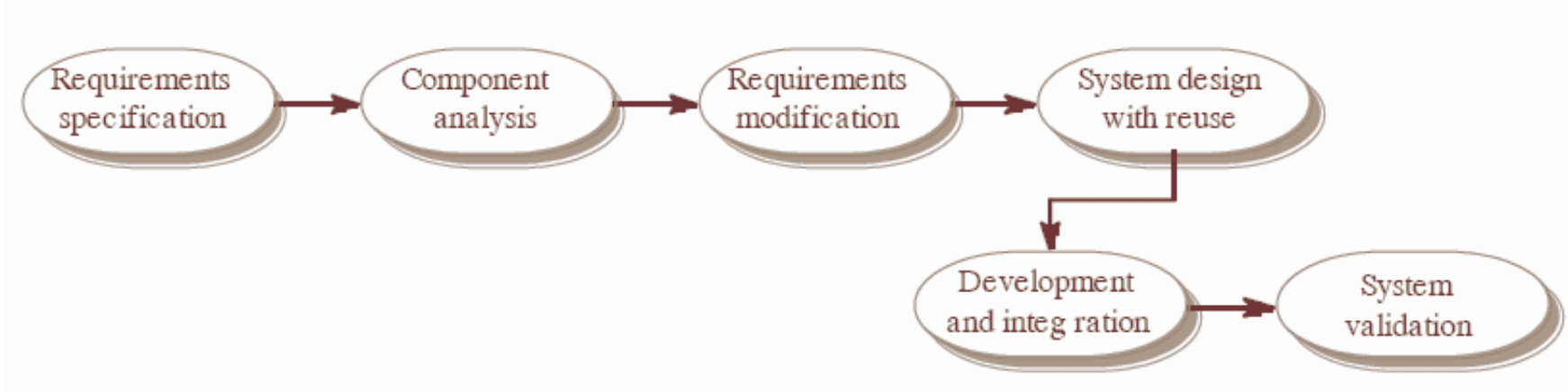
# Modello ad assemblaggio di componenti

- Basato sul sistematico riuso di componenti *off-the-shelf*, integrate opportunamente
- Fasi del processo
  - Analisi delle componenti
  - Adattamento dei requisiti
  - Disegno del sistema
  - Integrazione

# Modello ad assemblaggio di componenti



# Schema del modello a componenti

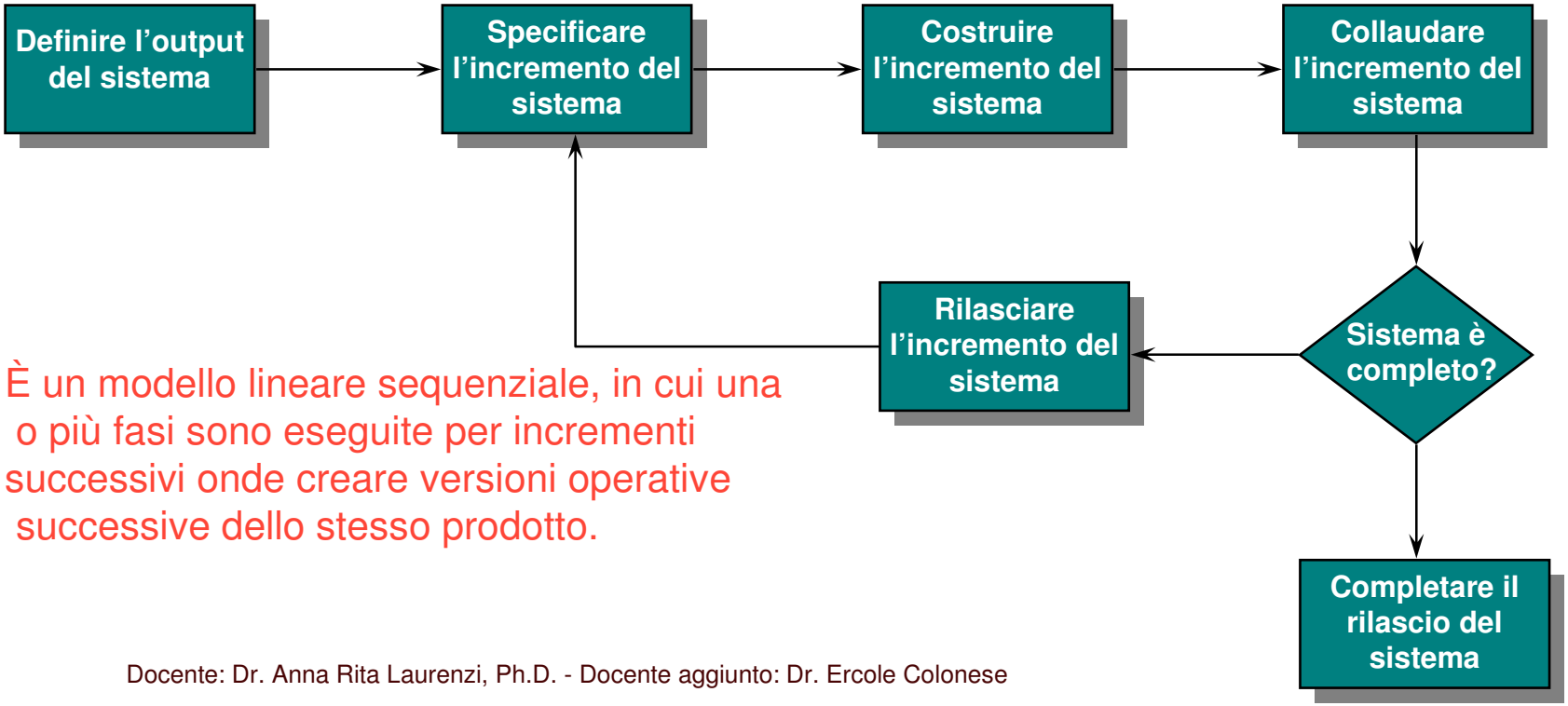


# Modello Incrementale

Combina approccio prototipale e a cascata

1. Non c'è tempo per una versione completa però dobbiamo uscire sul mercato.
2. Esiste un nucleo di requisiti di sistema ma i dettagli delle estensioni non esistono ancora.

Soluzione: modello di processo per un prodotto che evolve nel tempo




È un modello lineare sequenziale, in cui una o più fasi sono eseguite per incrementi successivi onde creare versioni operative successive dello stesso prodotto.

# Vantaggi del modello incrementale

- **Riuso delle componenti**: Tutte le componenti costruite con un incremento devono essere riusate nella costruzione degli incrementi successivi, altrimenti aumenta il costo di sviluppo e quello di manutenzione del prodotto

- **Integrazione delle componenti**: tutte le componenti devono essere integrabili e devono avere lo stesso livello di qualità perché sono frammenti di uno stesso software.

# Vantaggi del modello incrementale

- I requisiti a più alta priorità tendono ad ricevere maggiori test
  - Perché sono implementati per primi e testati ad ogni iterazione
- Il cliente riceve presto un prototipo funzionante, e può fornire feedback
  - I feedback vengono usati per l'iterazione successiva
- Minor rischio di fallimento dell'intero processo

# Modello a spirale di Boehm

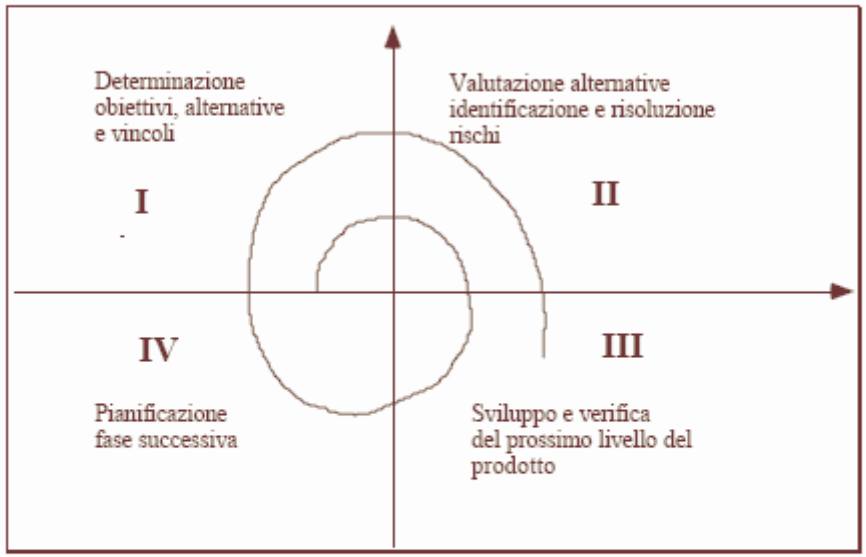
- Il processo di sviluppo è rappresentato come una spirale, piuttosto che come una sequenza
- Ogni ciclo nella spirale è una fase del processo
- Al termine di ogni "giro" il risultato può essere un *progetto*, un *prototipo*, un *sistema funzionante* o un *prodotto software* completo
- Non ci sono fasi predefinite. Il management del progetto deve decidere come strutturarlo in fasi

# Analisi dei rischi

- Il modello a spirale, a differenza degli altri, tiene conto dei *rischi*
- Rischio è evento imprevisto che può causare problemi o difficoltà
  - Esempio: Stiamo usando un compilatore per un nuovo linguaggio. C'è il rischio che il compilatore sia bacato.
- I rischi sono conseguenza di informazioni insufficienti
- Si risolvono acquisendo maggiori informazioni per ridurre l'incertezza

# Modello a spirale di Boehm

- 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.

- 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; trasformazionale, se la sicurezza e` piu` importante*

- **IV: pianificazione della fase successiva**

*si decide se continuare con un altro ciclo della spirale*

# Caratteristiche del modello

- 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***

# Voci da considerare

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

# Esempio: aumento qualità 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

# Esempio: aumento qualità prodotti

**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

# Esempio: aumento qualità prodotti

**Risultati**

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

**Piani**

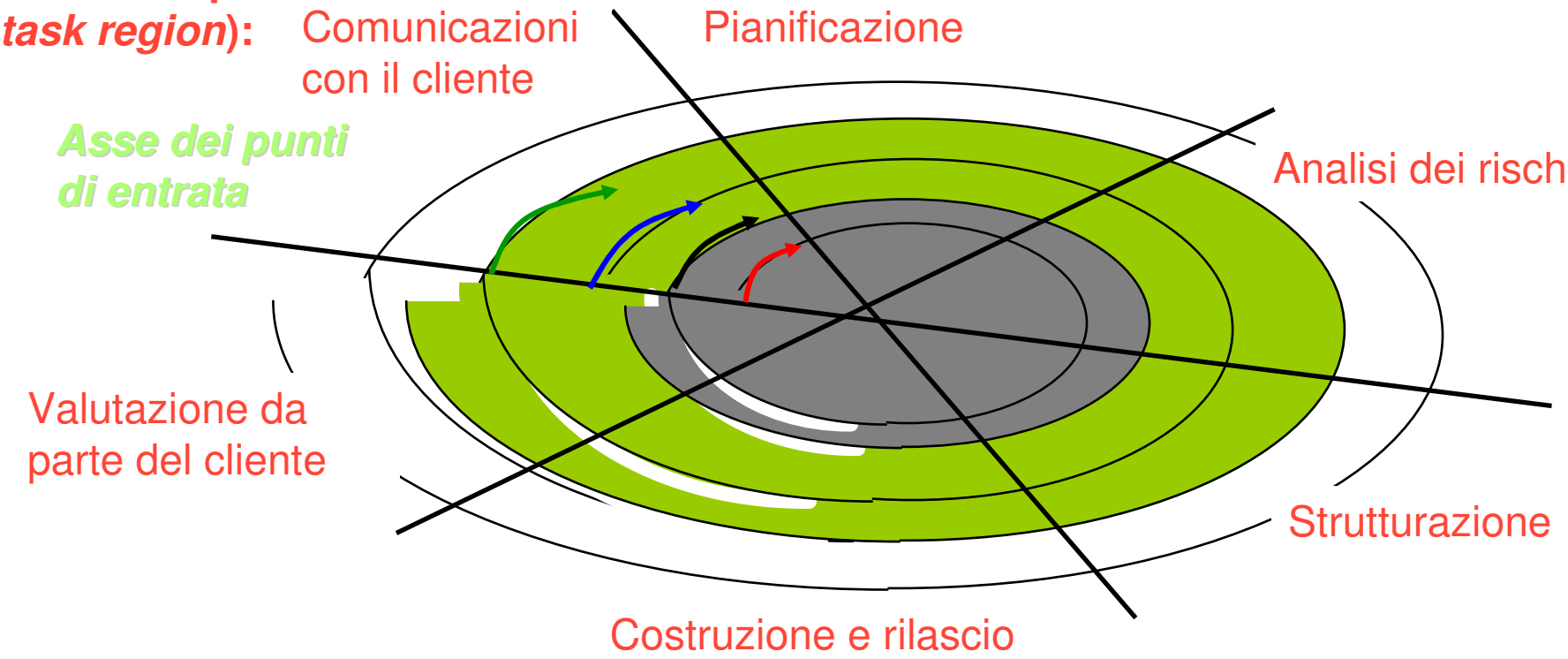
- Esplorare le opzioni per riutilizzo in maggiore dettaglio
- Sviluppare prototipi di sistemi di aiuto al riutilizzo
- Esplorare schemi di certificazione di componenti

**Decisione**

- Finanziare una fase di studio di 18 mesi

# Modello a spirale: schema

**Sei attività portanti (*task region*):**




Il modello abbina la natura iterativa della prototipazione e gli aspetti controllati e sistematici del modello sequenziale.

Progetti di sviluppo di nuove idee
Progetti di sviluppo di un nuovo prodotto
Progetti di miglioramento di un prodotto
Progetti di manutenzione di un prodotto

# Caratteristiche del modello a spirale

## La spirale è divisa a "spicchi"

- **Comunicazione con il Cliente.** Specificare gli obiettivi e i vincoli di quella fase, identificare i rischi ed eventualmente proporre delle strategie alternative
- **Pianificazione.** Attività rivolte a definire scadenze e risorse
- **Analisi dei rischi.** Attività rivolte a stimare i rischi tecnici e di gestione
- **Strutturazione.** Attività rivolte a costruire una o più rappresentazioni (strutture alternative) del sistema
- **Costruzione e rilascio.** Attività di sviluppo, effettuata secondo un modello generico (cascata, evolutivo...)
- **Valutazione da parte del cliente.** Attività rivolte a ricevere le reazioni del cliente su quanto fin qui realizzato (nei primi giri della spirale si creano "idee" o "proposte", nei successivi anche "prodotti" o "prototipi")

# Modello a spirale

- Vantaggi
  - Valutazione esplicita dei rischi
  - Adatto allo sviluppo di sistemi complessi di grandi dimensioni
  - Interazione con il cliente ad ogni giro della spirale
- Svantaggi
  - La stima dei rischi richiede competenze specifiche
  - Se i rischi non vengono valutati correttamente, potranno sorgere problemi nelle fasi successive
  - Il modello è relativamente nuovo e poco sperimentato

# Processo Unificato

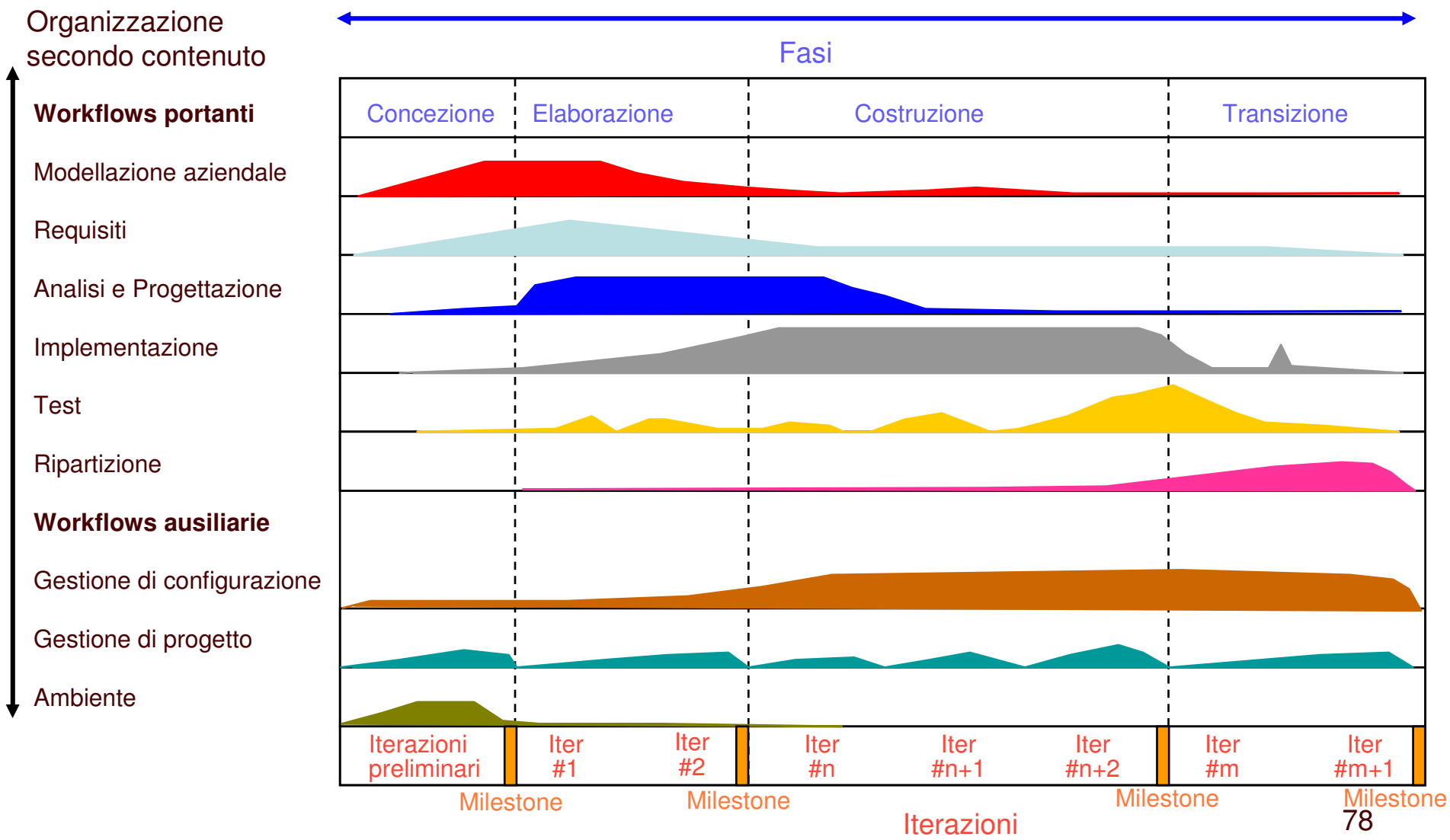
- Il Processo Unificato (UP) é un processo industriale per lo sviluppo di software che ha l'obiettivo di produrre applicazioni di alta qualità, rispondenti alle necessità degli utenti, con un approccio disciplinato e con un controllo accurato dei tempi e dei costi.

- Il Processo Unificato è stato inizialmente ideato da Ivar Jacobson, Grady Booch, and James Rumbaugh, ed è commercializzato dalla Rational Software come Rational Unified Process (RUP). L'UP raccoglie molte tra le "best practices" fondamentali dello sviluppo di applicazioni software, presentandole in una visione unitaria applicabile ad una vastissima tipologia di progetti ed organizzazioni

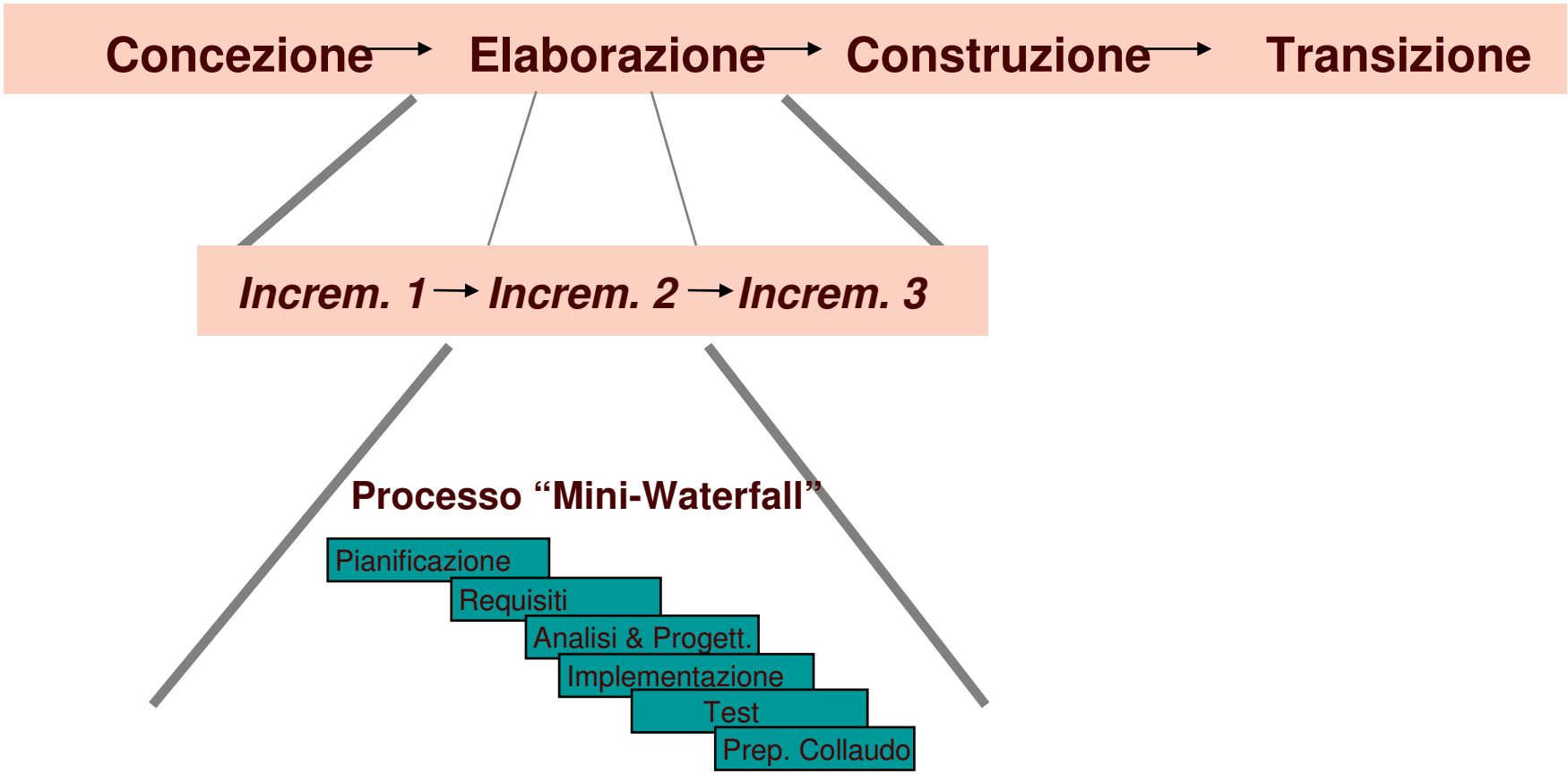
# Processo Unificato

- **Guidato dagli Use case**: la raccolta dei requisiti e i successivi passi sono guidati dagli UC.
- **Fondato sull'architettura**: fornire una visione generale del sistema adattabile al cambiamento dei requisiti
- **Iterativo e incrementale**: suddividere il progetto in miniprogetti, dove ogni miniprogetto rappresenta un'iterazione, il cui risultato è un'iterazione del prodotto da ottenere

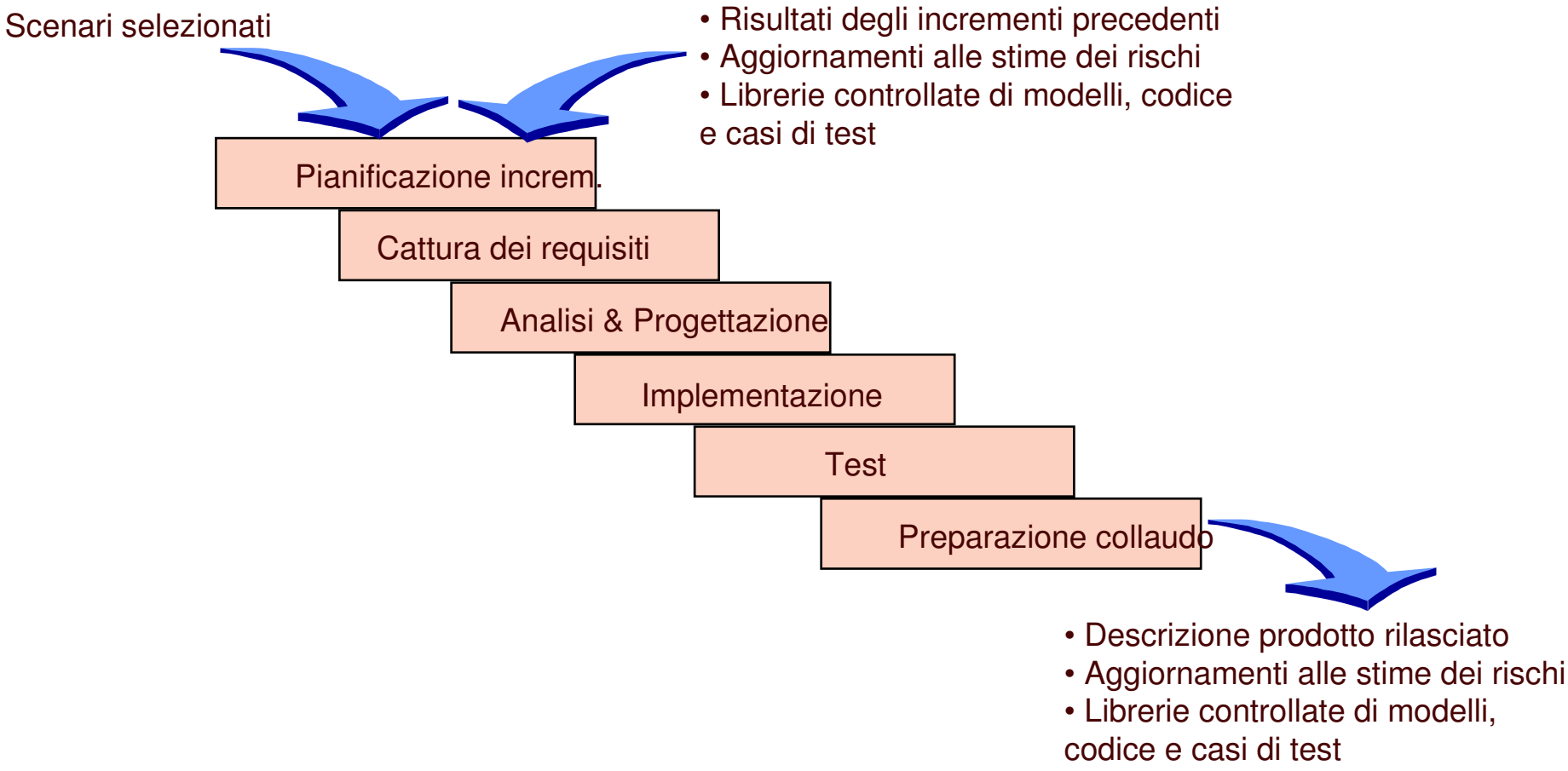
# Il Modello Iterativo



# Cicli di sviluppo iterativo



# Il mini ciclo a cascata



# Ciclo di vita per lo sviluppo del software OO

- Concezione
  - Raggruppa i rischi del progetto
  - Pianifica lo sviluppo
- Elaborazione
  - Sviluppa una descrizione dello scopo del sistema e della sua funzionalità esplorando gli scenari insieme ai clienti ed ai esperti del dominio
  - Stabilisce l'architettura del sistema
  - Progetta i meccanismi riguardanti le macro-decisioni dello sviluppo
- Costruzione
  - Dettaglia l'architettura portandola verso implementazione
  - Esegue iterazioni guidate da rischi e dai requisiti e controllate dai criteri di valutazione costantemente riesaminati in rapporto con lo stato del progetto
  - Integra continuamente
- Transizione-Installazione
  - Facilita l'accettazione del cliente, migliora le prestazioni, elimina i bugs
  - Misura il soddisfacimento del cliente
- Cicli post installazione
  - Continua l'approccio evolutivo
  - Mantiene l'integrità architetturale.

# Attività svolte durante un'iterazione (I)

- Pianificazione incremento
  - Prima di partire con la nuova iterazione, stabilire gli obiettivi generali dell'iterazione considerando:
    - Risultati degli incrementi precedenti ( se ne sono )
    - Aggiornamenti delle stime sui rischi del progetto
  - Determinare i criteri di valutazione per l'iterazione corrente
  - Realizzare un piano dettagliato dell'iterazione per includerlo nel piano generale dello sviluppo
    - Includere milestones intermedi per il monitoraggio del progetto
    - Includere walkthroughs e revisioni
- Cattura dei requisiti
  - Definire requisiti selezionando i casi d'uso da implementare
  - Creare il vocabolario dell'iterazione
  - Identificare gli attori
  - Identificare e descrivere i casi d'uso
  - Aggiornare il modello degli oggetti per includere le classi ed associazioni scoperte nell'ultima iterazione
  - Creare un modello concettuale iniziale
  - Sviluppare un piano di test per l'iterazione corrente

# Attività svolte durante un'iterazione (II)

- Analisi & Progettazione
  - Raffinare i casi d'uso
  - Raffinare il modello concettuale
  - Determinare le classi che debbono essere sviluppate o aggiornate durante l'iterazione
  - Raffinare il vocabolario
  - Definire i diagrammi di sequenze del sistema
  - Definire i contratti delle operazioni
  - Definire i diagrammi di transizioni di stato
  - Aggiornare il documento di architettura, se necessario
  - Definire i casi d'uso reali
  - Definire i report, le GUI e gli help
  - Definire i diagrammi di interazioni
  - Dettagliare i diagrammi di classi
  - Definire lo schema del database
  - Iniziare lo sviluppo delle procedure di test
- Implementazione
  - Generare automaticamente o manualmente codice basato sul modello della progettazione
  - Scrivere il codice delle operazioni
  - Completare le procedure di test
  - Realizzare i test di unità e di integrazione

# Attività svolte durante un'iterazione (III)

- Test
  - Integrare e testare il codice sviluppato con il resto del sistema ( le versioni precedenti)
  - Catturare e revisionare i risultati dei test
  - Valutare i risultati dei test rispetto ai criteri di valutazione stabiliti in precedenza)
  - Stimare l'iterazione
- Ripartizione e preparazione della descrizione della versione
  - Mettere d'accordo il codice con i modelli di progettazione
  - Includere i prodotti dell'iterazione nelle librerie controllate