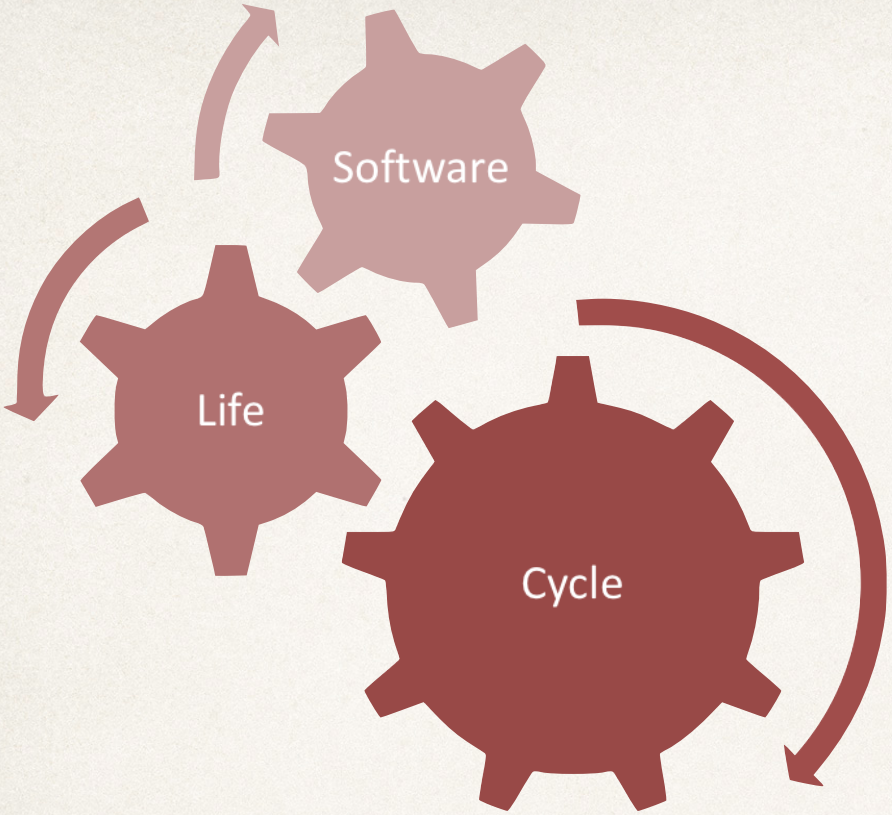


Modelli di processo di sviluppo di software

Alberto Gianoli

Univ. Ferrara - Corso di laurea in Informatica

Dove?

- Pressmann, cap. 3, parte cap. 4 (xp)
- Sommerville, cap. 4, parte cap. 17 (xp)
- Se vi interessa scrum: Ken Schwaber, Jeff Sutherland, La Guida a Scrum. La Guida Definitiva a Scrum: Le Regole del Gioco (PDF), scrumguides.org

Processo di produzione del software

- Insieme coerente di attività per la specifica, il progetto, l'implementazione, la verifica di sistemi software: è il percorso per sviluppare il prodotto sw
- Un modello di processo è una rappresentazione astratta di un processo
  - descrive il processo da una particolare prospettiva
- Modelli di processo generici:
  - a cascata (waterfall)
  - sviluppo evolutivo
  - sviluppo formale
  - meccanismi basati sul riutilizzo
  - modelli agile (extreme programming)
  - modello a spirale

Ciclo di risoluzione di un problema

- Status quo: stato della situazione attuale
- Definizione del problema: individuare lo specifico problema da risolvere
- Sviluppo tecnico: risoluzione di un problema mediante una opportuna tecnologia
- Integrazione: consegna dei risultati al committente
- Il ciclo si può applicare a diversi livelli di dettaglio:
  - a livello più alto (l'intero sistema)
  - ai singoli sottosistemi
  - alle singole funzioni

Modello

- Modellare il processo significa strutturarlo
  - dividerlo in attività
  - per ogni attività specificare
    - cosa
    - quali prodotti
    - quando
- Esempio: ISO 12207

Modello di ciclo di vita

- Come organizzo le attività
  - ordinamento delle attività
  - criteri per terminare una attività e passare alla successiva

- Esempio: preparare una torta
  - Modello di sviluppo generico
    - fare la spesa; scaldare il forno; impastare; mettere nella teglia; infornare
  - Non è la ricetta del dolce, sono le attività generiche, è indipendente dal dolce

Come sono evoluti i modelli di ciclo di vita?

- Build-and-Fix: non ho un modello…
  - attività non identificate né organizzate
  - il progetto non è gestito
- Modelli prescrittivi
  - cascata
  - incrementale
  - rapid prototyping
  - spirale

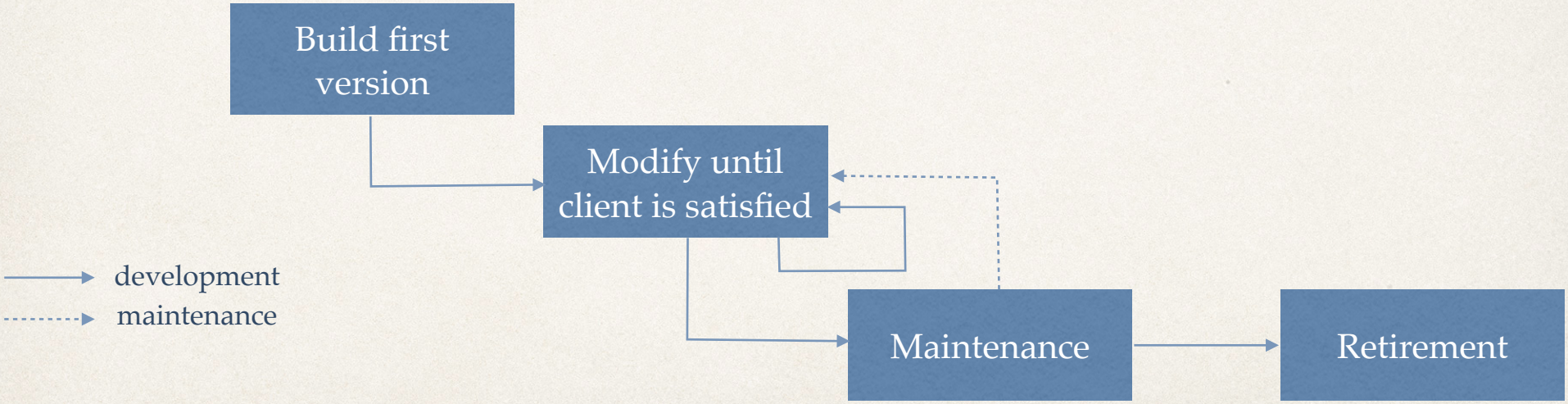
Tipici di una qualsiasi branca ingegneristica

- Unified Process
- Modelli agili
  - eXtreme Programming
  - scrum

Tipici dell'ingegneria del software

Build-and-Fix

- Il prodotto viene sviluppato senza specifica, nessun tentativo di progettazione: “stile libero” insomma…
  - Chi sviluppa comincia a scrivere il programma…
  - … che poi viene modificato più volte finché non va bene al committente

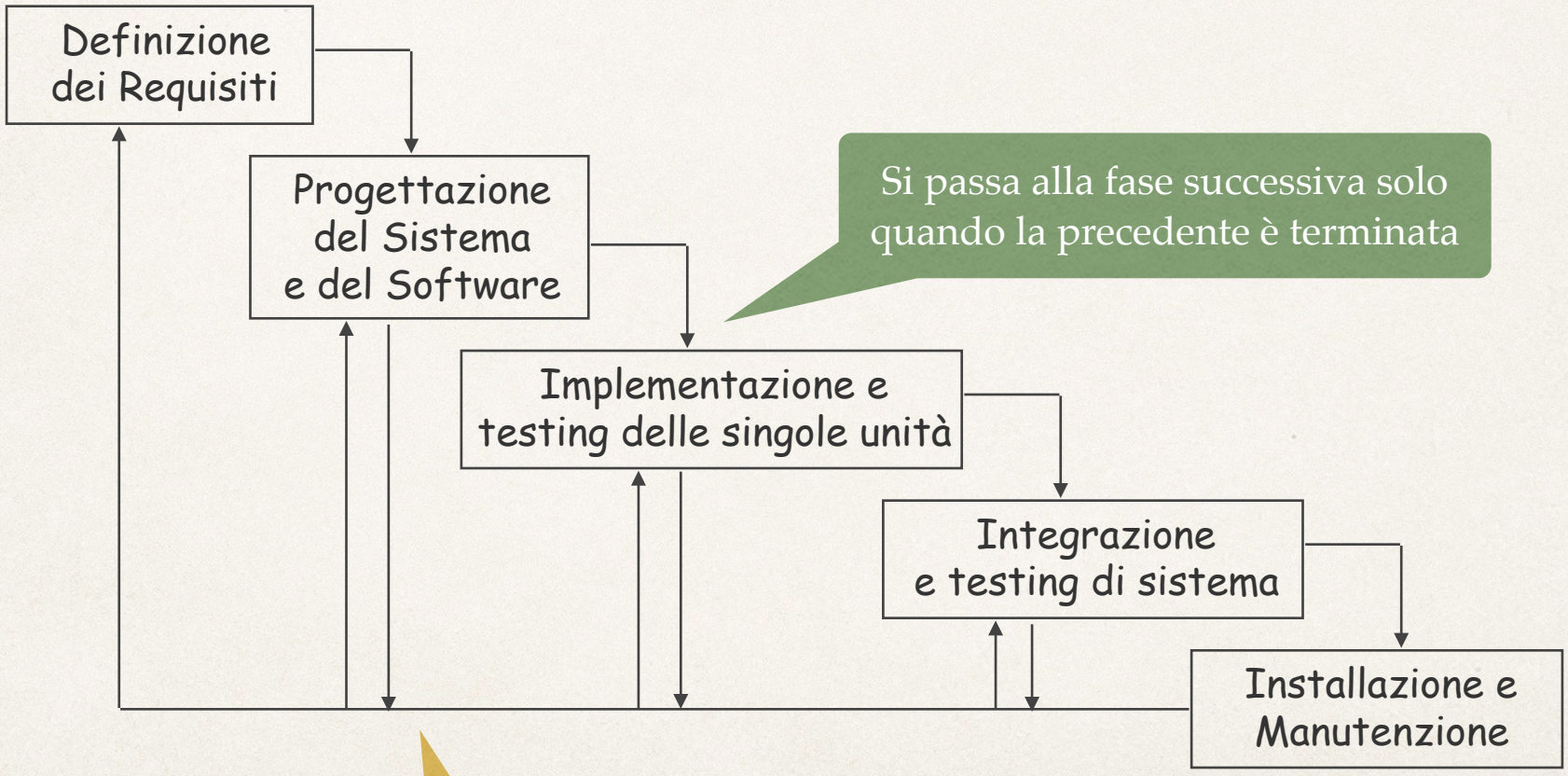


Build-and-fix

- Può andare bene per un progetto molto piccolo (100 linee codice?)
- Non è proponibile per prodotti di ragionevoli dimensioni
- Non esiste specifica né documentazione: la manutenzione è molto difficile

Il modello a cascata

Distinguiamo e definiamo le fasi: analisi e progettazione vengono prima della codifica



Si passa alla fase successiva solo quando la precedente è terminata

Ogni fase produce un documento che deve essere approvato da un gruppo di valutatori

Fasi del modello a cascata

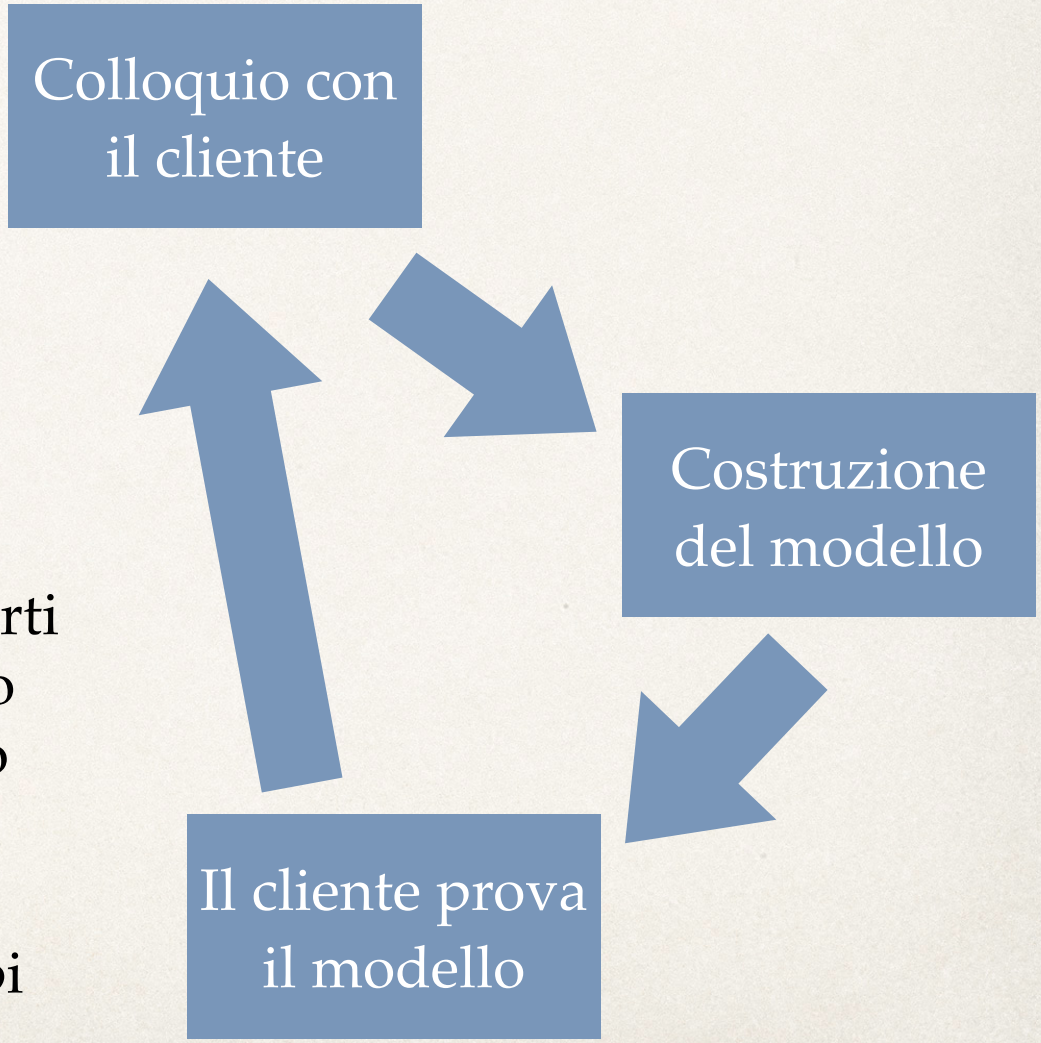
- Analisi e definizione dei requisiti
- Progettazione del sistema e/o del software
- Implementazione e test delle singole unità
- Integrazione e test del sistema
- Installazione e mantenimento del sistema

Problemi legati al modello a cascata

- Il grosso limite del modello a cascata è la difficoltà a effettuare cambiamenti nel corso del processo.
  - è molto difficile soddisfare cambiamenti nei requisiti da parte del committente
- Manca interazione con il cliente: c'è solo a prodotto finito
- La suddivisione in fasi può sembrare arbitraria o artificiosa
  - in realtà questo modello è derivato dal ben consolidato modello ingegneristico per la risoluzione dei problemi
- Il modello a cascata è adeguato quando i requisiti sono ben compresi dall'inizio e non sono soggetti a modifiche

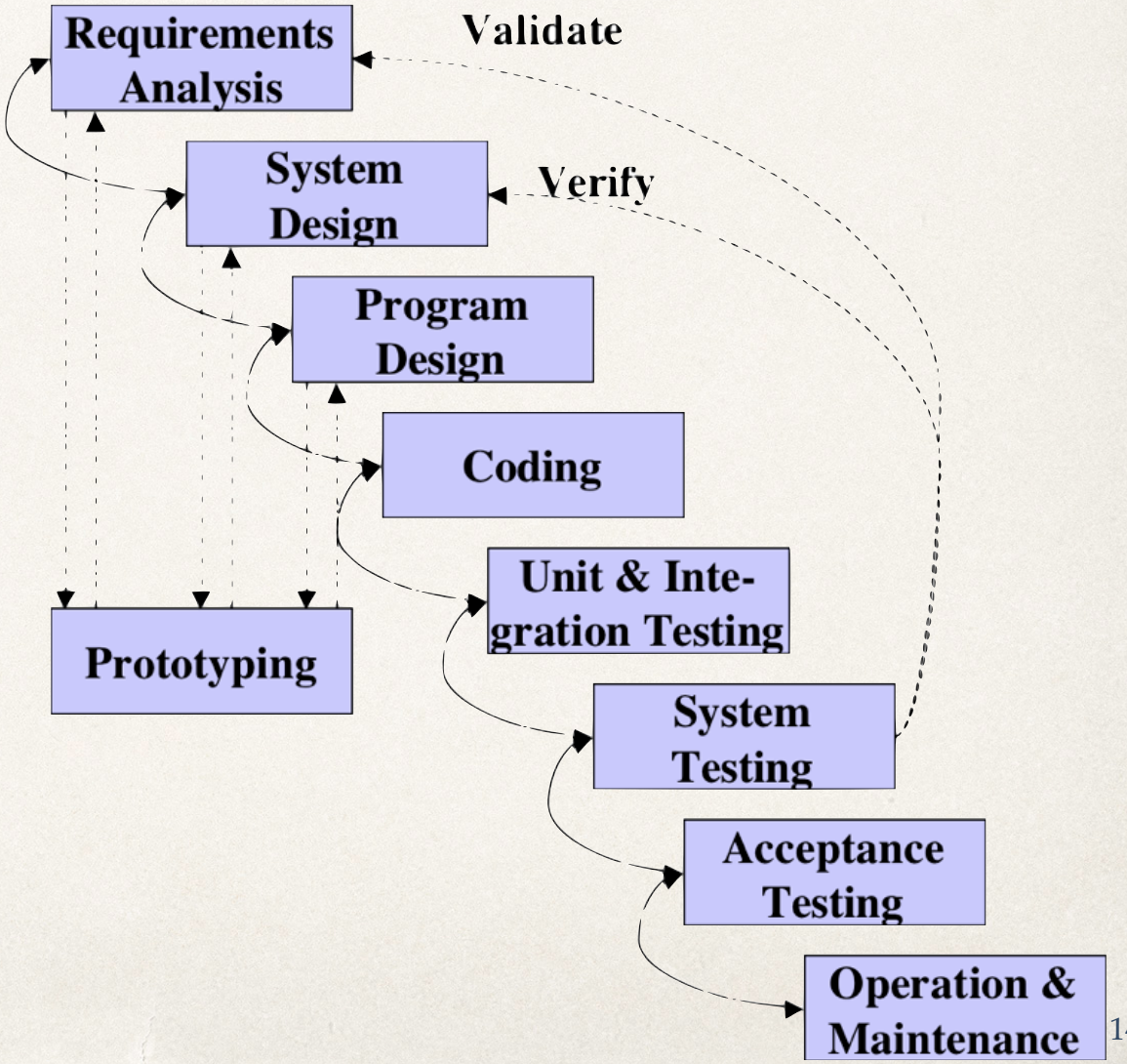
Prototipazione

- Spesso i requisiti non sono sufficientemente chiari
  - il committente stabilisce gli scopi generali de sistema software, ma non chiarisce subito i requisiti in modo completo
  - oppure gli sviluppatori sono incerti sul significato di alcuni requisiti o su come strutturare l'interfaccia o gli algoritmi
- Possibilità di procedere per prototipi



Prototipazione

- Possiamo anche combinare questo modo di procedere con il modello a cascata...

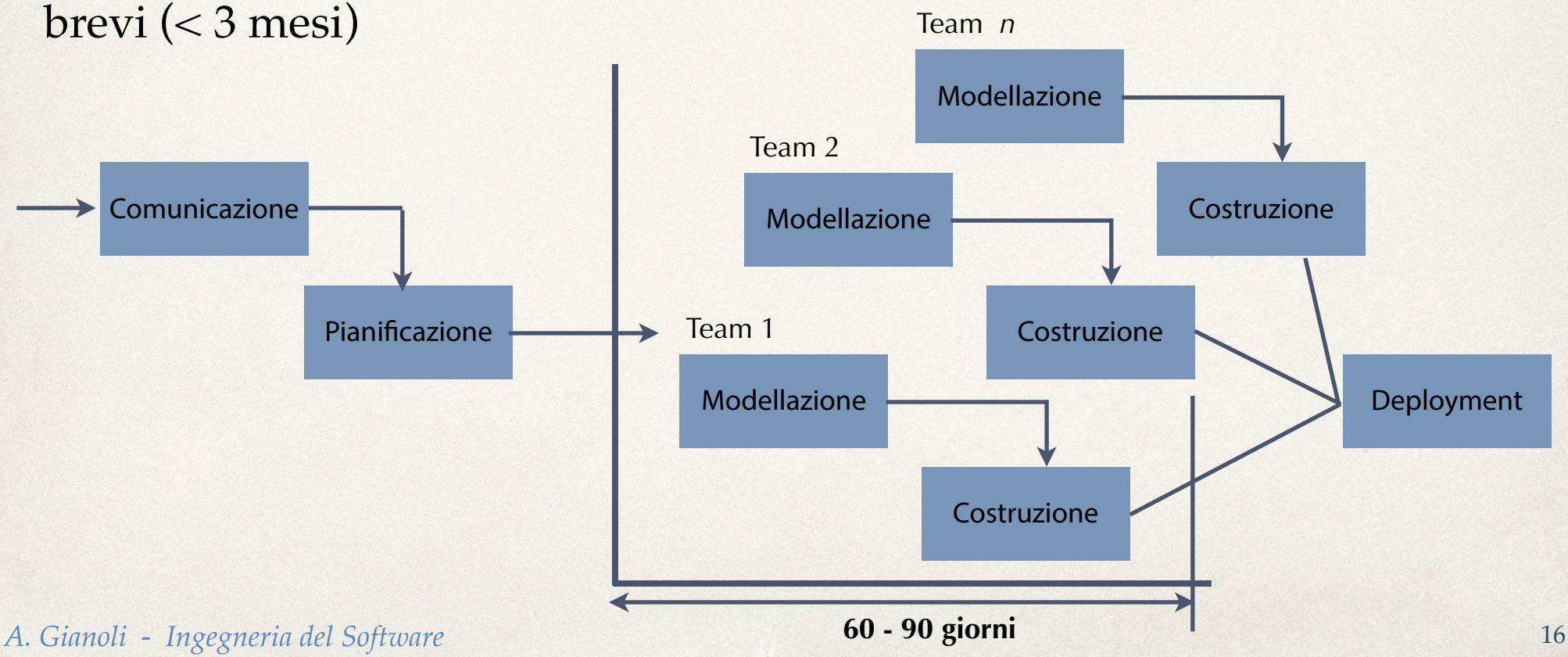


RAD (Rapid Application Development)

- Modello sequenziale lineare per avere un ciclo di sviluppo molto breve
- Lo sviluppo rapido è ottenuto mediante il riuso di componenti
- Quando applicarlo:
  - requisiti chiari
  - processo di sviluppo ben vincolato
- nei casi in cui si può applicare, il modello RAD può portare allo sviluppo di software in tempi brevi rispetto al modello a cascata classico

RAD

- adatto alle situazioni in cui il sistema da implementare può essere facilmente partizionato fin dall'inizio
- ogni parte deve potere essere sviluppata indipendentemente e in tempi brevi (< 3 mesi)



Le fasi del modello RAD

- Comunicazioni
  - servono per comprendere il problema e i dati che devono essere considerati dal software
- Pianificazione
  - come suddividere il lavoro tra i vari team
- Modellazione
  - modellazione business: quali dati guidano il processo? quali dati vengono generati? da chi? chi li utilizza?
  - modellazione dati: definirli in termini di oggetti o di ADT
  - modellazione del processo: definire le operazioni sui dati
- Costruzione
  - generare l'applicazione con linguaggi di alto livello, riuso di componenti, ...
  - collaudo singoli componenti
- Deployment
  - integrazione dei vari componenti e collaudo finale

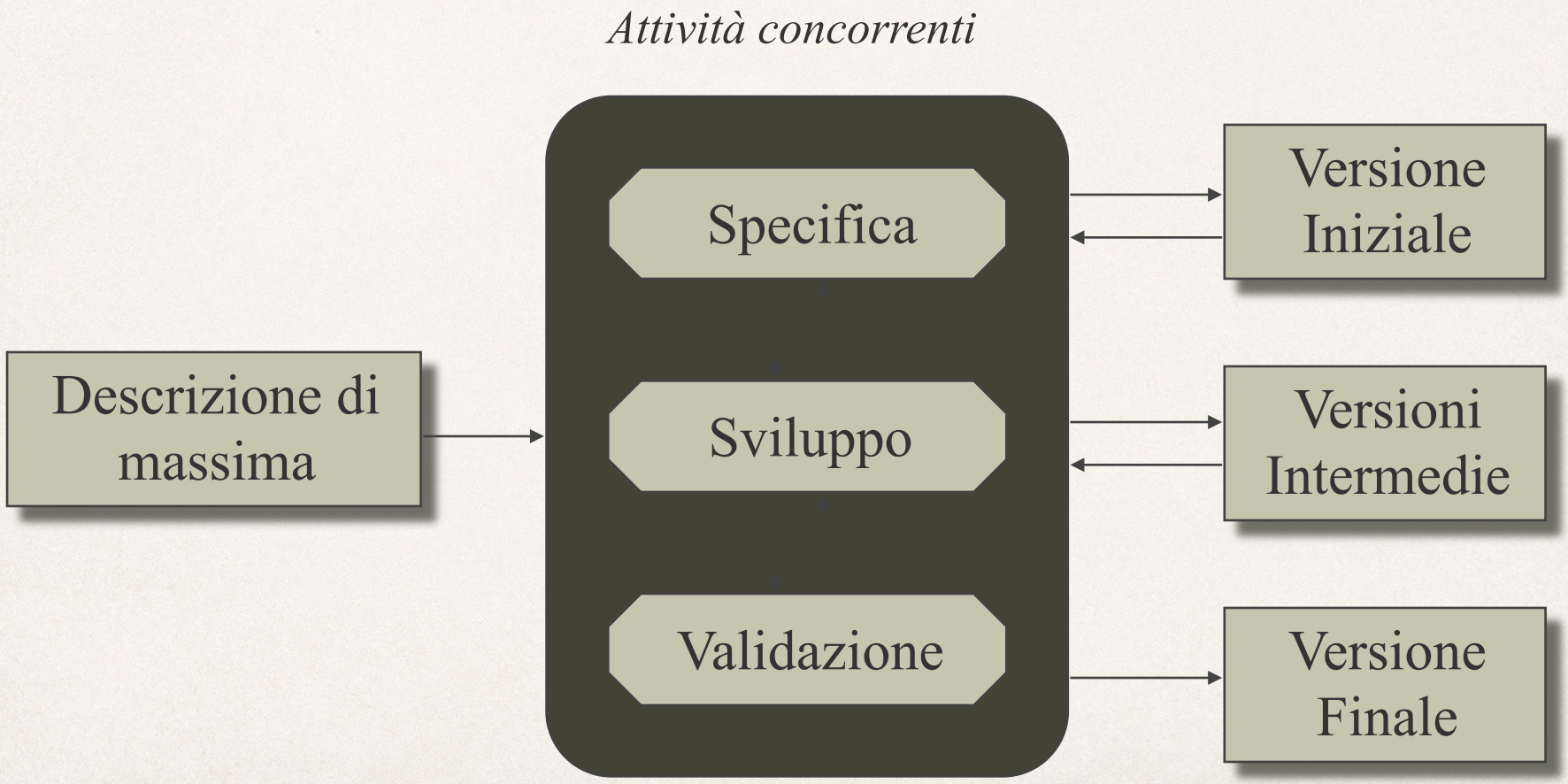
Limiti del modello RAD

- progetti estesi ma partizionabili
  - bisogna avere risorse umani sufficienti a creare il numero corretto di team
- sviluppatori e clienti devono essere disponibili a completare il sistema in tempi rapidi
- RAD non è il modello adatto quando
  - il sistema non può essere modularizzato correttamente
  - si richiedono alte prestazioni che dipendono da interfacce appositamente definite
  - si usano tecnologie innovative con alto rischio di incontrare problemi strada facendo

Modello evolutivo

- Prototipazione di tipo evolutivo
  - L'obiettivo è lavorare con il committente 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
  - 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

Schema del modello evolutivo



Modello evolutivo

- Problemi:
  - mancanza di visibilità del processo (quando produrre i documenti?)
  - sistemi spesso poco strutturati
  - possono essere richiesta capacità particolare (p.e. in linguaggi per la prototipazione rapida)
- attenzione a non trasformare il prototipo in un prodotto da porre in produzione:
  - al committente il prototipo potrebbe sembrare funzionante, ma può avere tanti e tali problemi da essere inutilizzabile in condizioni reali
  - il sistema va rifatto

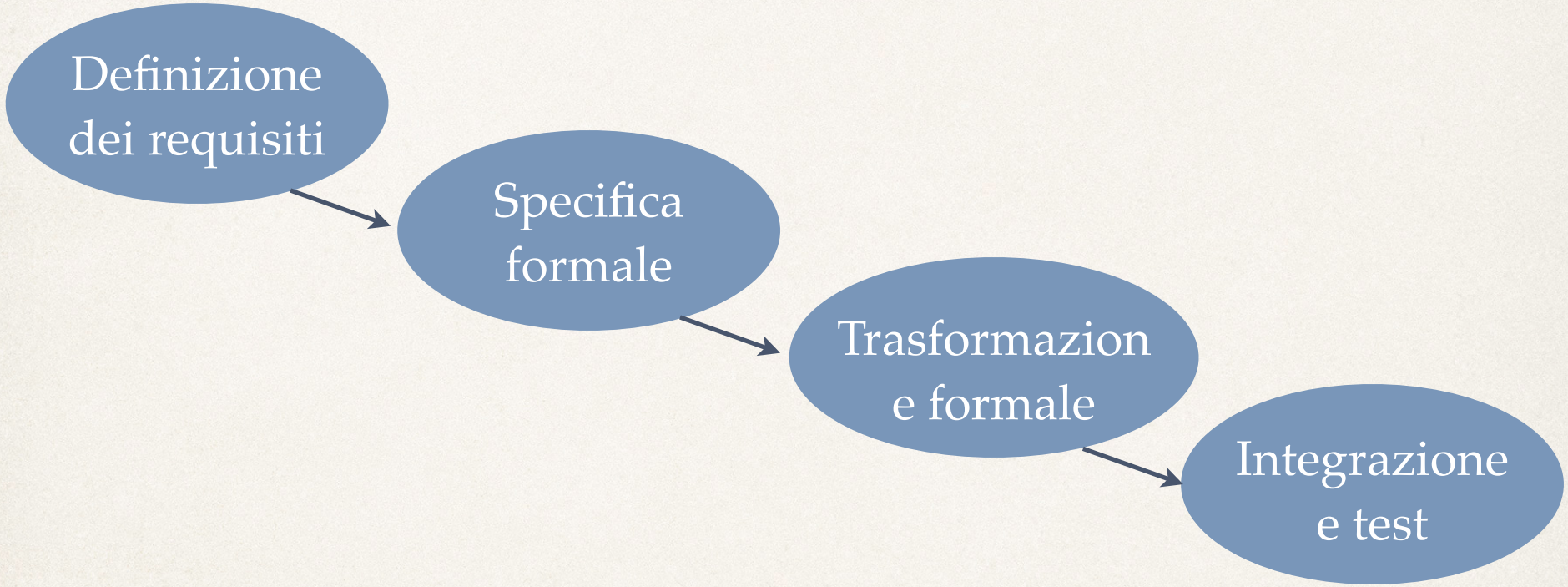
Modello evolutivo

- Applicabilità
  - sistemi interattivi di piccola o media dimensione
  - per parti di sistemi più grandi (p.e. l'interfaccia utente)
  - per sistemi con vita attesa corta
  - per sviluppare prototipi

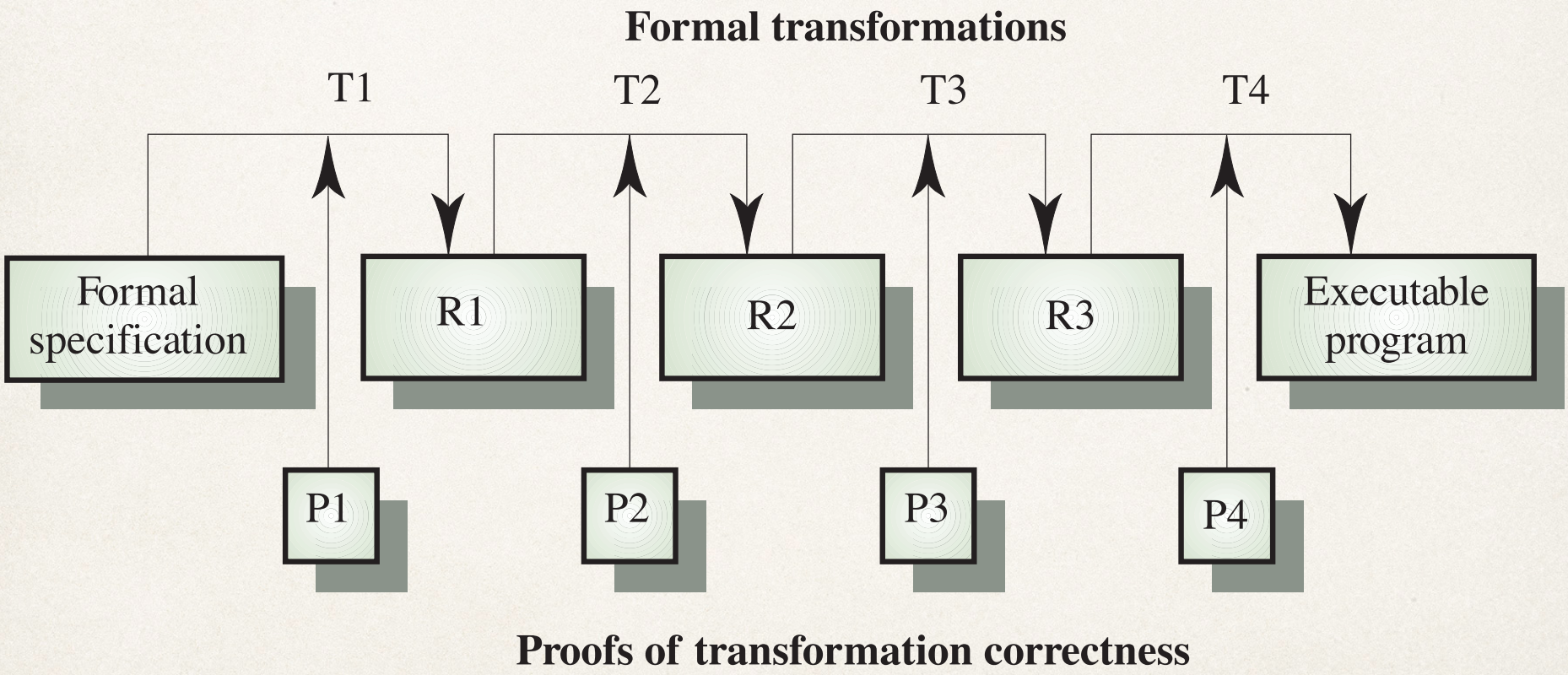
Modello trasformazionale (o modello formale)

- Basato sulla trasformazione di una specifica matematica in un programma eseguibile, attraverso trasformazioni che permettono di passare da una rappresentazione formale ad un altra
- Le trasformazioni devono preservare la correttezza
  - in questo modo è banale verificare che il programma soddisfa la specifica

Fasi del modello trasformazionale



Trasformazioni formali



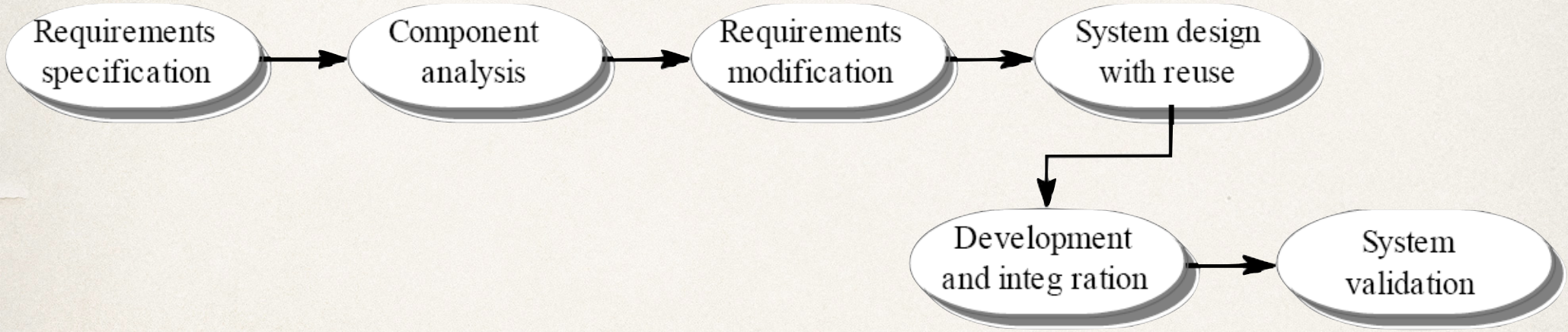
Modello trasformazionale

- Problemi
  - richiede conoscenze specializzate e addestramento per essere applicato
    - certe parti del sistema (p.e. l'interfaccia utente) sono difficili da specificare formalmente
- Quando applicarlo
  - sistemi critici, in cui sicurezza e/o affidabilità sono essenziale e devono non solo essere raggiunti, ma bisogna dimostrare che il sistema li può mantenere

Modello basato sul riutilizzo (sviluppo a componenti)

- Si basa sul riutilizzo sistematico di componenti *off-the-shelf* opportunamente integrate
- Fasi del modello:
  - analisi delle componenti
  - adattamento dei requisiti
  - progettazione del sistema
  - integrazione

Sviluppo basato sul riutilizzo



Modello basato sul riutilizzo

- E' per natura evolutivo: richiede un approccio iterativo allo sviluppo del software
- Classico esempio: tecnologia a oggetti
- Vantaggi:
  - netta riduzione del ciclo di sviluppo
  - netta riduzione dei costi di progetto
  - maggiore produttività
- Svantaggi:
  - dipende tutto dal trovare componenti che possano servirci

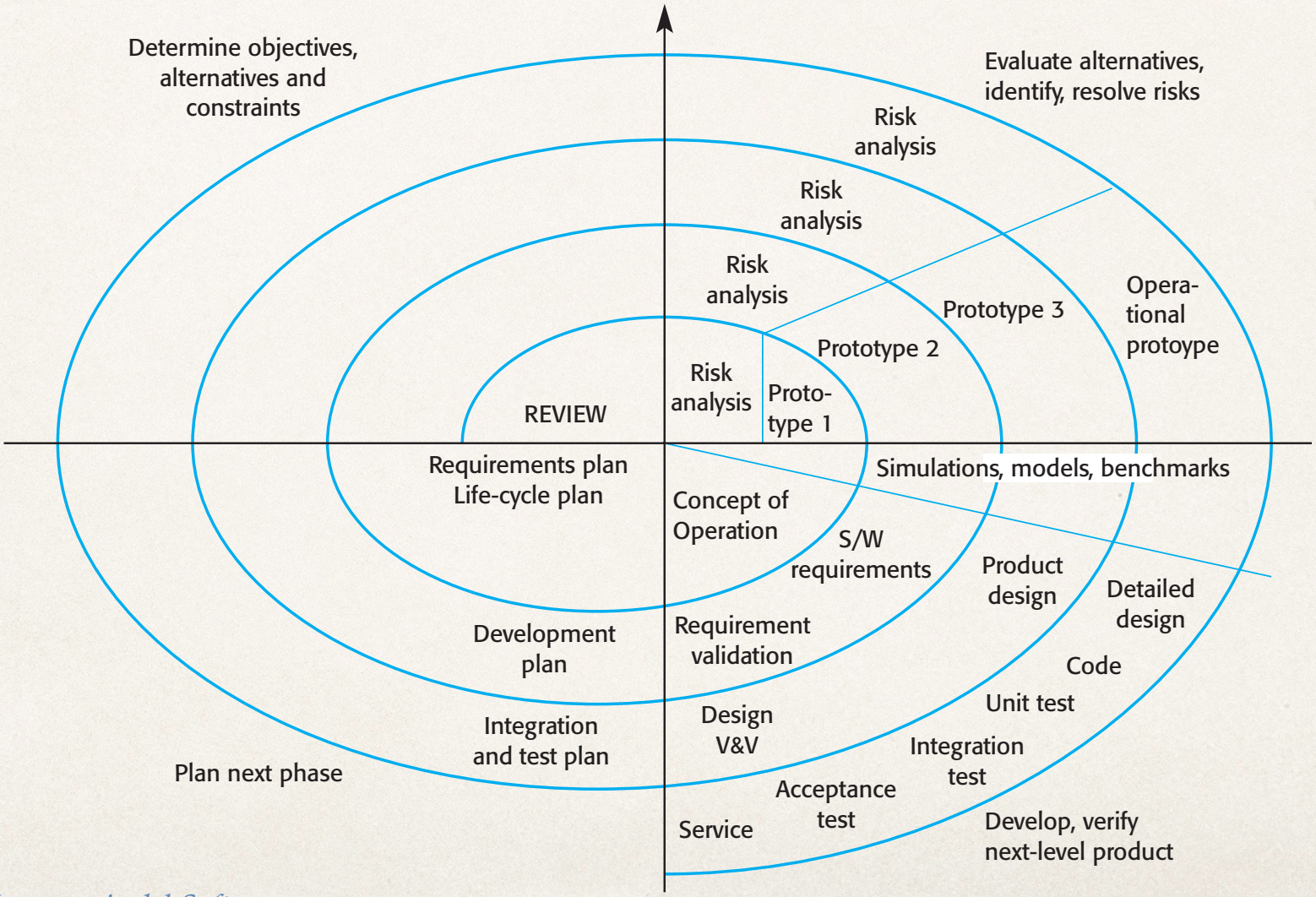
Modello a spirale

- Obiettivo principale = minimizzare i rischi
- Rischio = misura di incertezza del risultato di una attività
  - p.e.: stiamo usando un tool nuovo, c'è il rischio che ci siano dei bug.
- Meno informazione si ha, più alti sono i rischi
- Bisogna acquisire maggiori informazioni per ridurre l'incertezza

Modello a spirale

- 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* oppure un *prodotto software* completo
- Non ci sono fasi predefinite: sta al management del progetto decidere come strutturarlo in fasi

Schema del modello a spirale



Fasi del modello a spirale

- La spirale è divisa in spicchi:
  - *Comunicazione con il cliente*: specificare gli obiettivi e i vincoli di quella fase, identificare i rischi e eventualmente proporre strategie alternative
  - *Pianificazione*: definizione delle scadenze e delle risorse
  - *Analisi dei rischi*: stimare i rischi tecnici e di gestione
  - *Strutturazione*: costruire una o più rappresentazioni del sistema
  - *Costruzione e rilascio*: sviluppo effettuato secondo un modello generico (cascata, evolutivo,..)
  - *Valutazione da parte del committente*: ricevere le reazioni del cliente su quanto prodotto. Nei primi giri si creano idee o proposte, nei successivi prodotti o prototipi

Modello a spirale

- Vantaggi
- concentra l'attenzione sulle possibilità di riuso
- concentra l'attenzione sull'eliminazione degli errori
- valutazione esplicita dei rischi
- adatto allo sviluppo di progetti di grandi dimensioni
- interazione con il cliente a ogni giro della spirale

- Svantaggi
- richiede esperienza nella valutazione dei rischi
- errori nella valutazione dei rischi si ripercuotono nelle fasi successive
- richiede raffinamenti per un uso generale

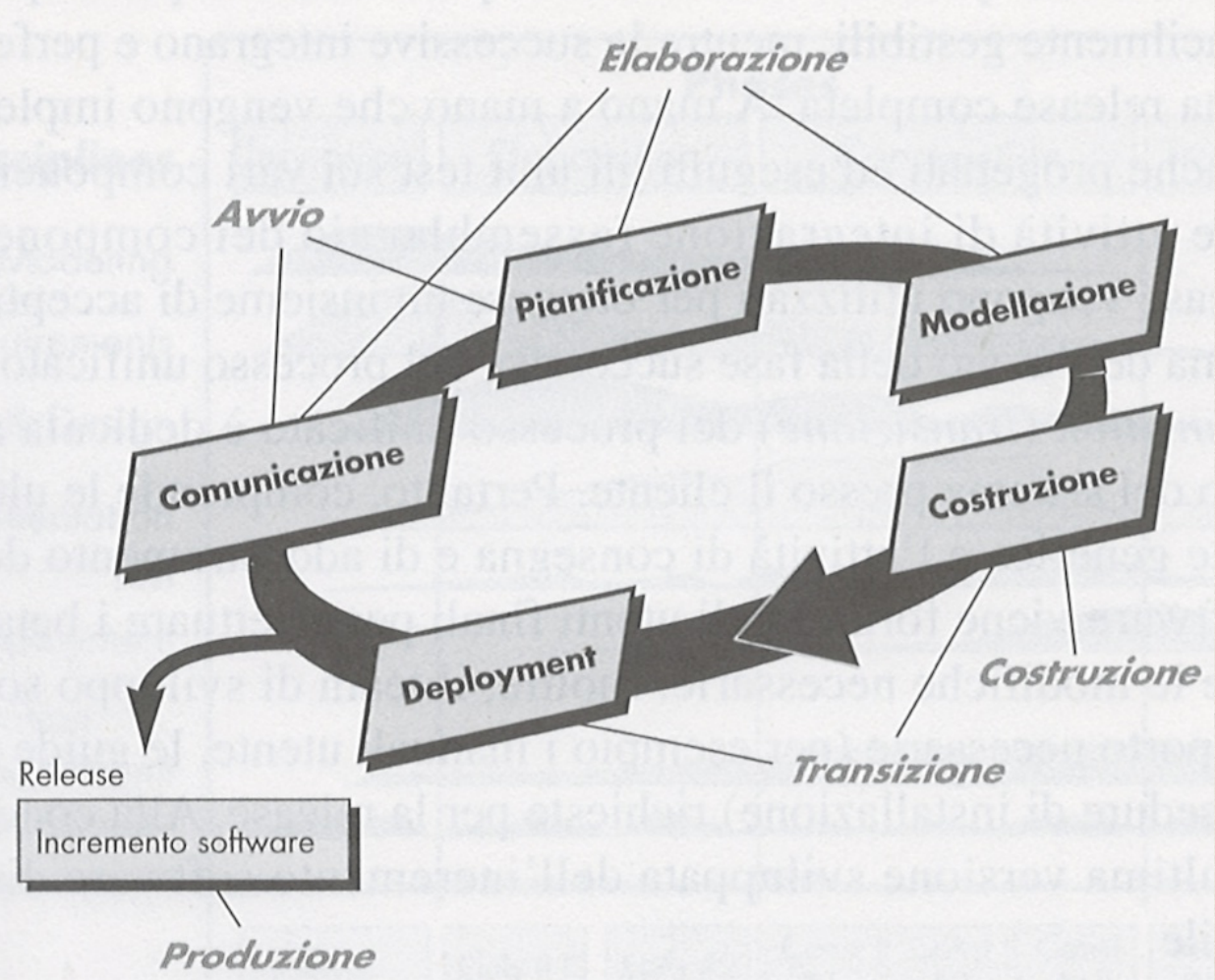
Rational Unified Process (RUP)
processo unificato

- Derivato da UML e dal relativo *Unified Software Development Process*
- Modello ibrido: prende elementi dai vari modelli generici, aiuta i cicli e illustra buone prassi nella specifica e nella progettazione
- Individua due prospettive
  1. **prospettiva tecnica**: tratta gli aspetti qualitativi, ingegneristici e di metodo di progettazione
  2. **prospettiva gestionale**: tratta gli aspetti finanziari, strategici, commerciali e cani

Iterativo e incrementale

RUP

- Consiste di 4 fasi, separate da milestone che possono essere superate solo se lo stato di avanzamento è sufficiente e l'analisi dei rischi soddisfacente
- Se non si supera una milestone, o si abortisce il progetto, oppure la fase viene iterata per raffinare il risultato e risolvere il problema



RUP: le fasi

- Inception (avvio): comprende le attività di comunicazione con il cliente e di pianificazione. Collaborando con cliente e utenti finali si identificano i requisiti business (tramite use case preliminari), si propone una architettura di base per il sistema e un piano per la natura iterativa e incrementale del progetto. Si ottiene indicazione dei principali sottosistemi e delle relative funzioni, da raffinare e ampliare in seguito. La pianificazione identifica risorse, rischi principali e stabilisce un piano dei tempi per le fasi successive

- Elaboration (elaborazione): comprende comunicazione con il cliente e modellazione. Si ampliano gli use case e si espande la rappresentazione dell'architettura con 5 viste: modello use case, modello analitico, modello di progettazione, modello implementativo, modello deployment. Alla fine di questa fase si ricontrolla la pianificazione

RUP: le fasi

- Construction (costruzione): sostanzialmente progettare, programmare e testare il sistema. Varie parti vengono sviluppate parallelamente e integrate. A seconda della dimensione del progetto, questa fase è divisa in varie iterazioni: inizialmente prototipi e poi raffinamenti e perfezionamento. Alla fine di questa fase si dovrebbe avere qualcosa di funzionante e la relativa documentazione

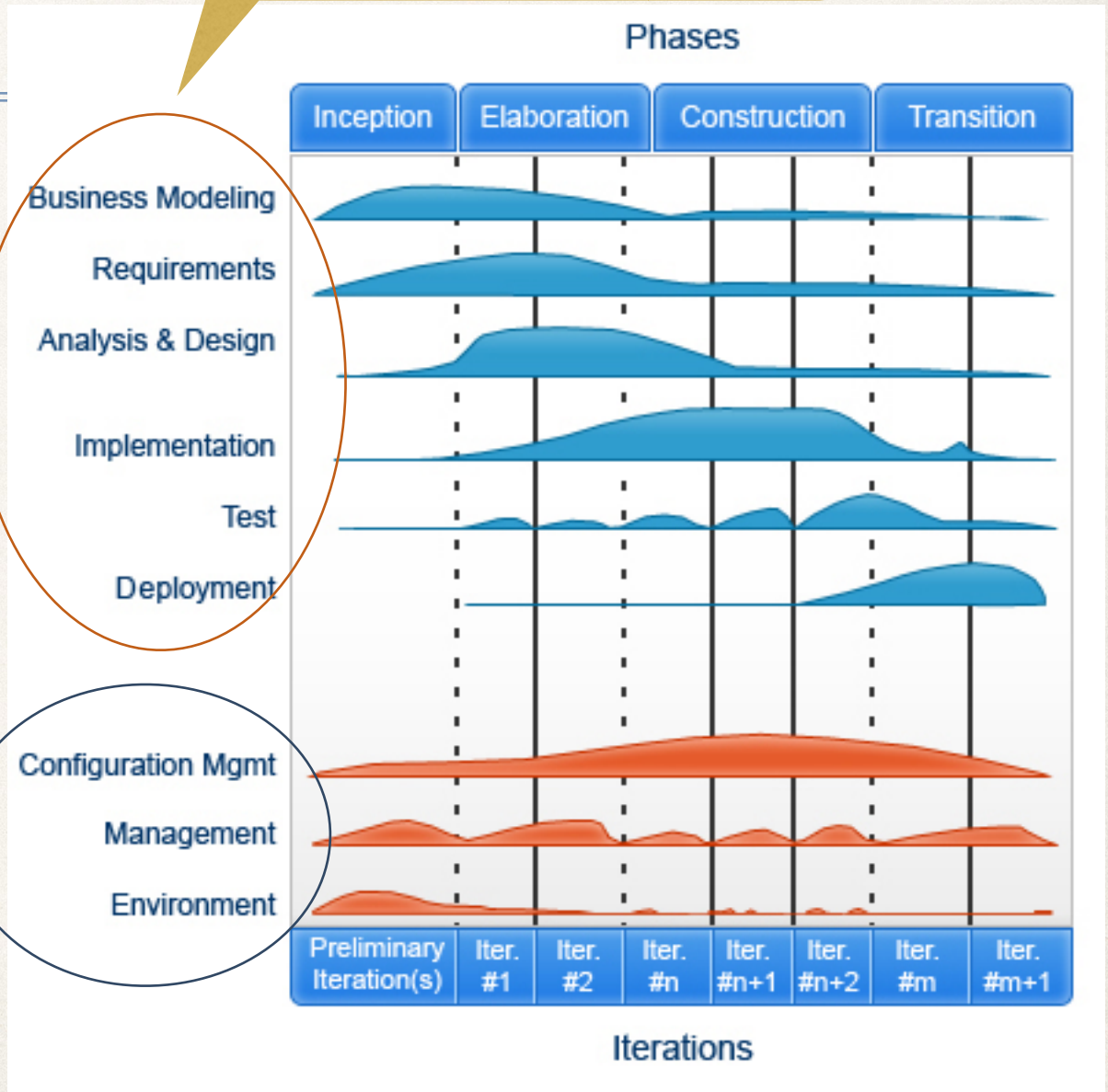
- Transition (transizione): spostare il sistema dalla comunità di sviluppo al cliente e farlo funzionare nell'ambiente reale. Alla fine di questa fase, l'ultima versione sviluppata dell'incremento software diventa una release software utilizzabile

- E' possibile che prima della fine di una release sia già iniziato il lavoro per il successivo incremento sw.

RUP: le fasi

- La ciclicità e l'incremento sono supportati sia in ogni fase, sia come intero insieme di fasi
- Visione statica: suddivisione in attività di produzione del software (workflow); sei workflow principali e tre di supporto

questi sono workflow di processo (tecnici)

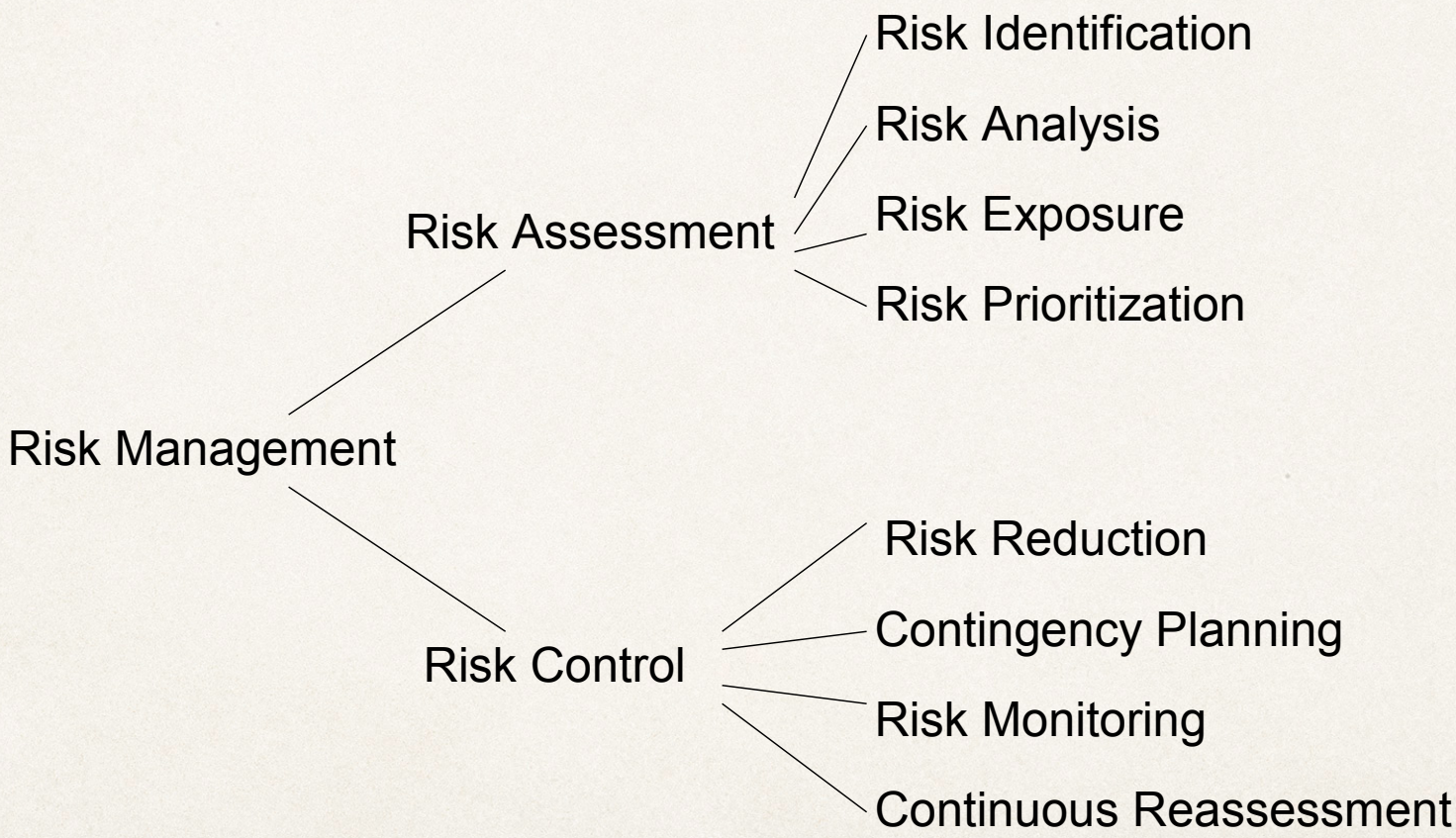


questi sono workflow di supporto (gestionali)

Valutazione dei rischi nei modelli di processo del software

- Modello a cascata
  - alto rischio per sistemi nuovi, per problemi di specifica e di progettazione
  - basso rischio per sviluppo di problemi se familiarità è già acquisita
- Modello evolutivo, prototipazione
  - basso rischio per nuovi problemi
  - alto rischio a causa della scarsa visibilità del processo (antieconomico produrre documentazione a ogni iterazione)
- Modello trasformazionale
  - alto rischio legato alla necessità di tecnologia avanzata e di elevate capacità da parte degli sviluppatori

Gestione dei rischi



Come identificare i rischi

- Comincia con una lista di errori tipici
- Guarda il piano di sviluppo
  - cammini critici
  - membri dello staff critici
  - consegne critiche
  - milestones critiche
- Guarda i requisiti
- Guarda la progettazione tecnica
- Guarda i progetti precedenti

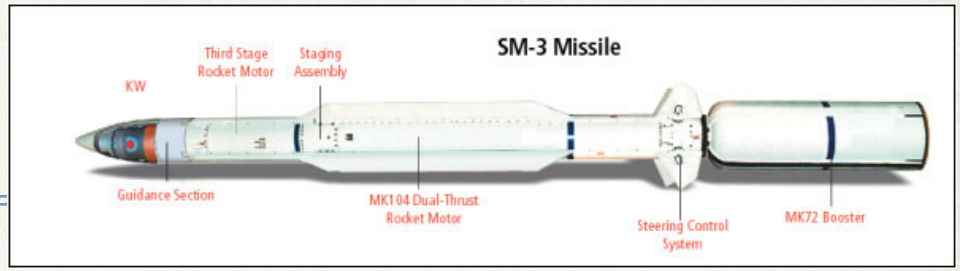
Come identificare i rischi

- Conduci delle sessioni di brainstorming con lo staff, gli utenti finali, i venditori e il management dedicate all'identificazione dei rischi
  - Try to assess the direction of thinking by third parties as they may give an indication of future requirements, expectations or vendor changes.
  - If you are dependent on vendors, try to understand their business situation
- Get as much input as possible!

Rischi: analisi, conseguenze e priorità

- Per ciascun rischio
  - determina la probabilità che esso si verifichi
  - determinane l'impatto
  - determinane le conseguenze: cosa perdiamo se il rischio si verifica?
- Per l'insieme dei rischi
  - mettili in ordine di priorità

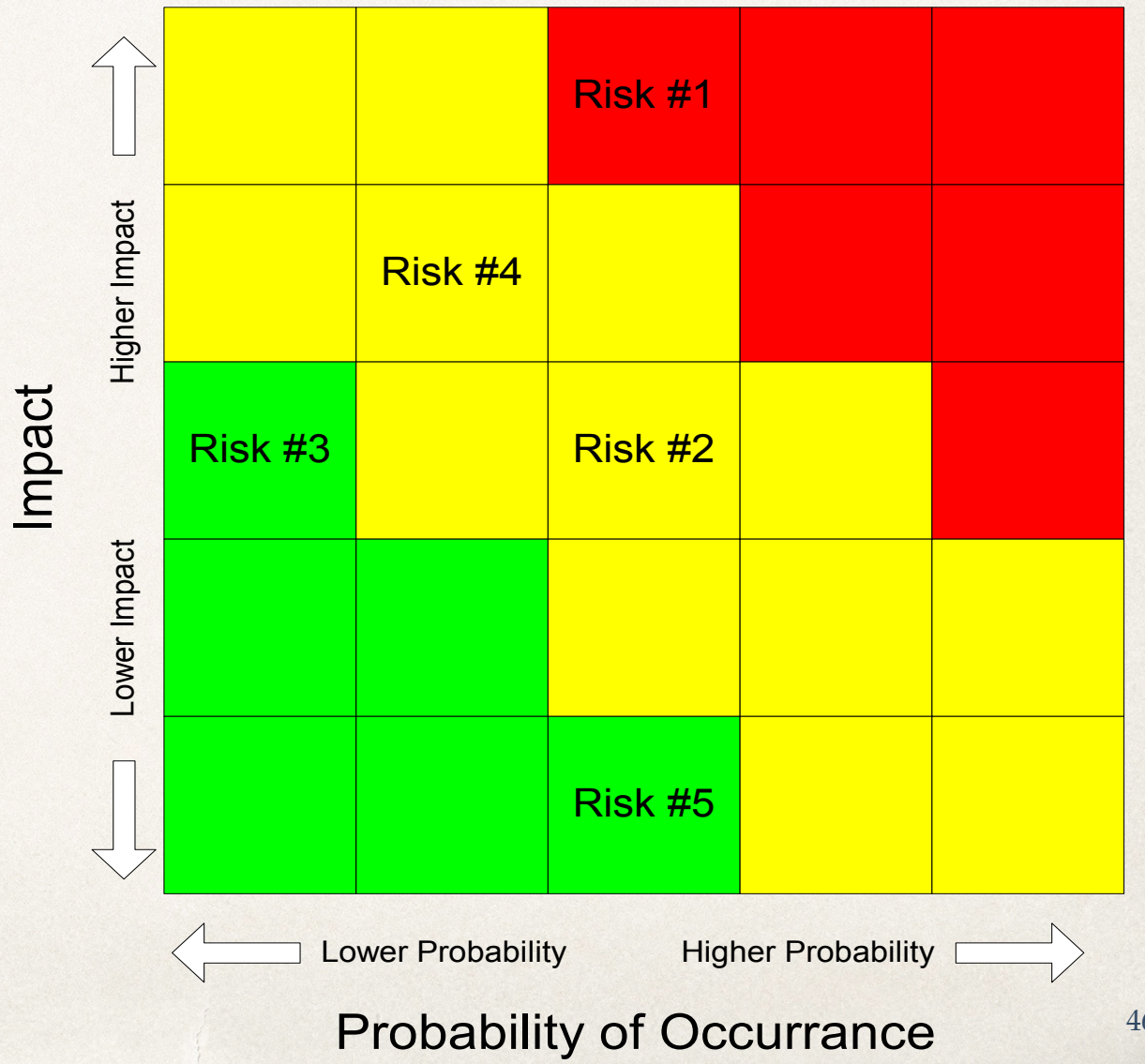
Esempio di classificazione dei rischi



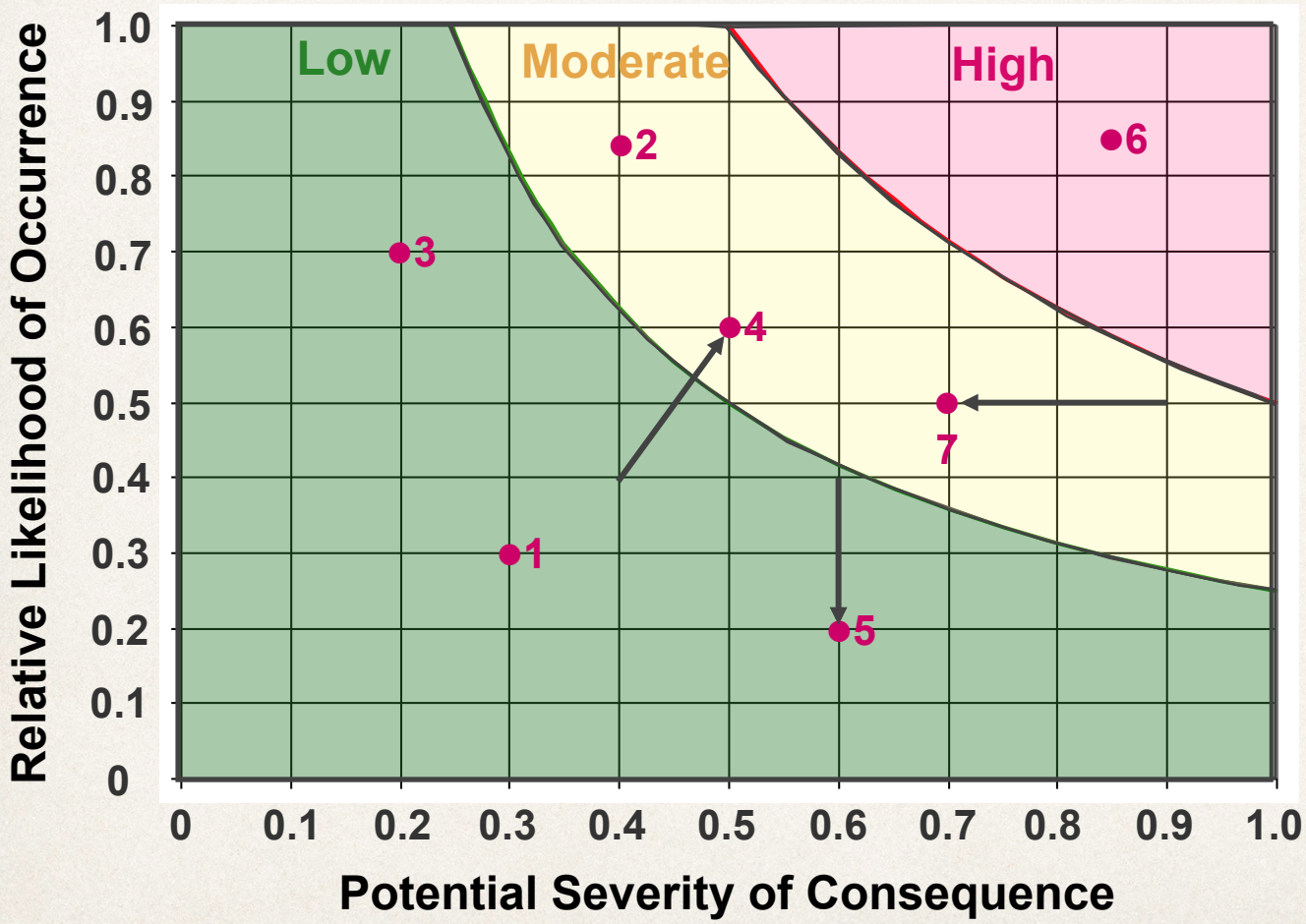
Risk no.	Risk	Consequences	Likelihood	Severity	Risk Factor
1	May not achieve specified total impulse	Overall missile performan-ce may not be acceptable	0,8	0,7	0,56
2	2nd thrust may not initiate properly	Would reduce mission effectiveness	0,4	0,8	0,32
3	May exceed weight allowance	Overall missile performan-ce may not be acceptable	0,5	0,6	0,3
4	Mechanical interfaces with other stages may be inadequate	Missile failure during transportation, launch or flight	0,2	0,9	0,18
5	Stage production cost may exceed allowance	Total missile cost may not be adequate	0,5	0,3	0,15
6	Propulsion development time may exceed plan	Development time (and cost) may not meet program objectives	0,7	0,3	0,21
7	Propulsion stage may not meet safety requirements	Missile may not be acceptable to customer	0,3	0,8	0,24

Classification & tracking methods

- Occorrenze e impatto
  - scala da 1 a 5
- Priorità
  - red - high
  - yellow - med
  - green - low
- Il piano va rivisto periodicamente!



Classification & tracking methods (alternativa)



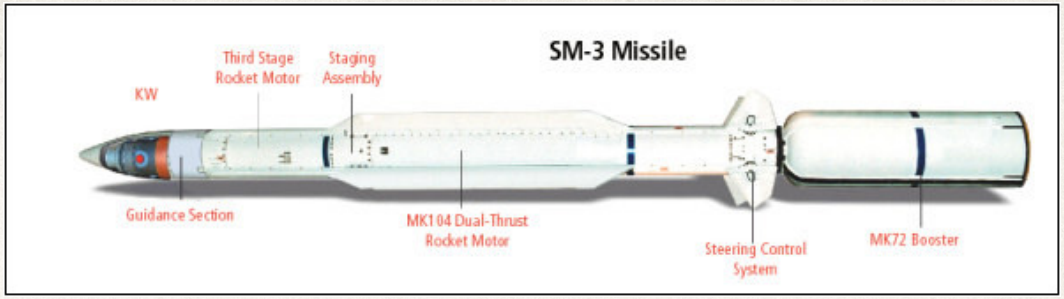
Arrows indicate changes from previous report

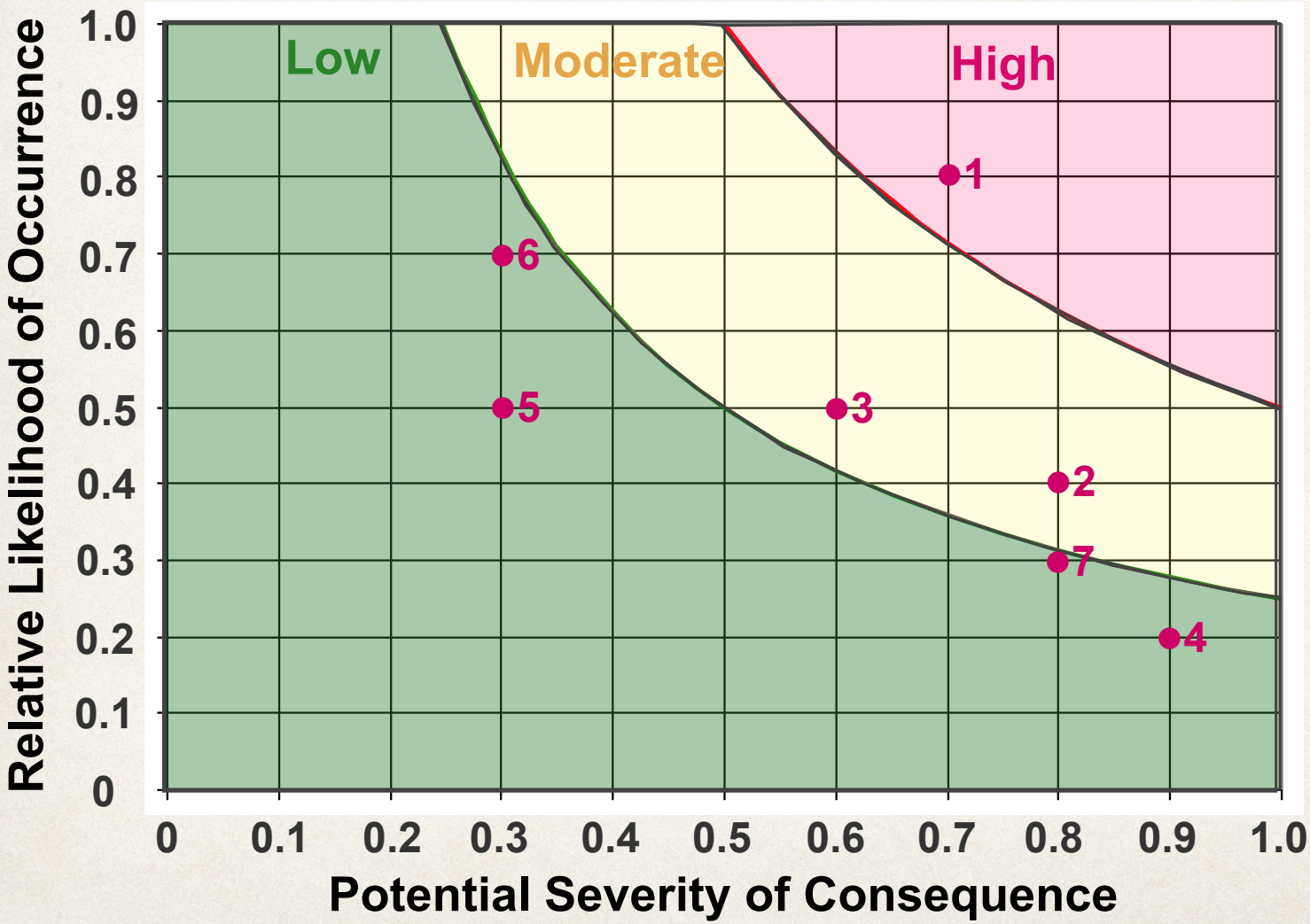
Legend
1 =
2 =
3 =
4 =
5 =
6 =
7 =

(Brief descriptions)

Report Date:

Esempio





Legend
1 = Total impulse
2 = 2nd thrust
3 = Stage weight
4 = Interfaces
5 = Stage cost
6 = Schedule
7 = Safety

Visibilità del processo software

- C'è bisogno di documentazione per poter valutare i progressi nel processo di sviluppo software
- Problema
  - la programmazione dei tempi di consegna dei "deliverables" può non combaciare con i tempi necessari per completare una attività
  - la necessità di produrre documentazione vincola l'iterazione del processo
  - il tempo necessario per approvare i documenti è significativo

Esempio di documentazione del modello a cascata

Activity	Output documents
Requirements analysis	Feasability study, Outline requirementes
Requirements definition	Requirements document
System specification	Functional specification, Acceptance test plan Draft user manual
Architectural design	Architectural specification, System test plan
Interface design	Interface specification, Integration test plan
Detailed design	Design specification, Unit test plan
Coding	Program code
Unit testing	Unit test report
Module testing	Module test report
Integration testing	ntegration test report Final user manual
System testing	System test report
Acceptance testing	Final system plus documentation

Visibilità dei modelli di processo

Process model	Process visibility
Waterfall model	Good visibility, each activity produces some deliverable
Evolutionary development	Poor visibility, uneconomic to produce documents during rapid iteration
Formal transformations	Good visibility, documents must be produced from each phase for the process to continue
Reuse-oriented development	Moderate visibility, it may be artificial to produce documents describing reuse and reusable components
Spiral model	Good visibility, each segment and each ring should produce some document