MODELLI DI PROCESSI SOFTWARE

LEZIONE 3
06/10/2025
VINCENZO RICCIO

RIFERIMENTI

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

- Sommerville - Capitolo 2.1, 2.3

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

MODELLI DI PROCESSI SOFTWARE

DEFINIZIONE

- *W. Scacchi in Encyclopedia of Software Engineering* definisce il modello di processo software come **modello del ciclo di vita del software (CVS)**:

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

DEFINIZIONE STANDARD IEEE 12207.0-1996

"Software Life Cycle: a framework containing the processes, activities, and tasks in the development, operation and maintenance of a software product, spanning the life of the system from the definition of its requirements to the termination of its use "

PROCESSO VS MODELLO DI PROCESSO

- Non esiste un processo universale → Ciascuna organizzazione utilizza il proprio processo, un'organizzazione può usare processi diversi per prodotti diversi
- Ogni processo è complesso, comprende attività, sotto-attività, prodotti e persone con responsabilità
- **Modello di processo software:** descrizione semplificata e astratta del processo software, osservato da un determinato punto di vista

PROCESSO VS MODELLO DI PROCESSO

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

- (Come abbiamo visto) ci si riferisce a questi modelli con molti alias: Modelli di processo software o ciclo di vita del software o paradigmi di processo
- Descrizioni astratte di alto livello dei processi software, che possono essere utilizzate per spiegare i diversi approcci allo sviluppo del software
- Strutture di processo da estendere e adattare per creare processi concreti e specifici

MODELLO DI PROCESSO SOFTWARE: PUNTI DI VISTA

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

- Ogni processo ha diverse possibili descrizioni:
  - Architetturale – descrive la sequenza di attività senza fornire dettagli sulle specifiche attività (utilizzato in questo corso);
  - Data-flow – evidenzia le trasformazioni dei dati operate dalle attività del processo;
  - Role/action – I ruoli delle persone coinvolte nel processo e le relative responsabilità

DESCRIZIONE DEL PROCESSO SOFTWARE

- La descrizione del processo software include:
  - **Attività** da svolgere e l'ordine tra tali attività
  - **Prodotti**, risultanti da ciascuna attività
  - **Ruoli**, persone coinvolte nel processo e loro responsabilità
  - **Pre- e Post-condizioni**, condizioni che devono verificarsi prima e dopo che un'attività del processo sia svolta. Stabiliscono i criteri di transizione per progredire da uno stadio del processo al successivo

QUALE MODELLO DI PROCESSO SCEGLIERE?

- Esistono diversi modelli, adatti a differenti tipi di prodotto
- In questo corso ci focalizziamo su due tipologie di modelli di processo:
  - Plan-driven (in questo blocco di slides)
  - Agili (nel prossimo blocco di slides)
- Per grandi sistemi occorre trovare un compromesso tra processi pianificati e processi agili

RIEPILOGO TERMINOLOGIA

- **Processo:** un insieme di attività concentrate nel tempo finalizzate alla realizzazione di un particolare output
- **Processo Software:** Un insieme strutturato di attività necessarie per lo sviluppo di un sistema software
- **Modello di Processo Software (o modello di ciclo di vita del software):** Rappresentazione semplificata di un processo. Tale modello fornisce una descrizione del processo da una particolare prospettiva (e pertanto fornisce solo informazioni parziali)

TIPOLOGIE DI MODELLI DI SVILUPPO SOFTWARE

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

In questa lezione:

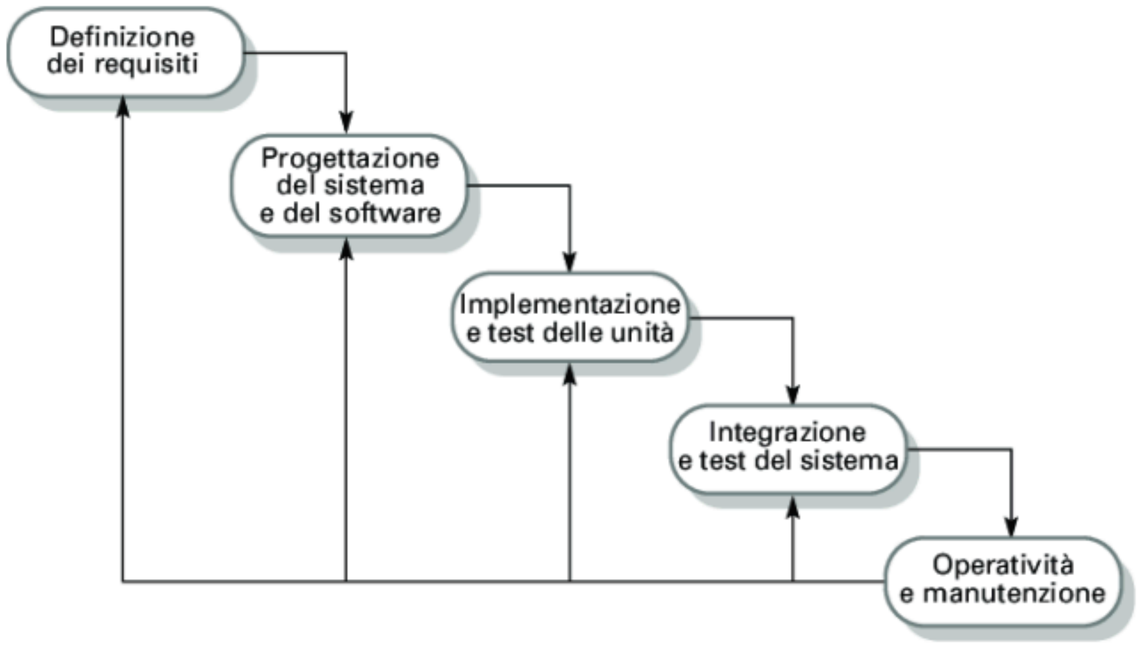
1. Modelli a cascata
2. Modelli evolutivi
3. Modelli orientati al riuso
4. Modelli trasformazionali (cenni)

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

MODELLI DI PROCESSI SOFTWARE

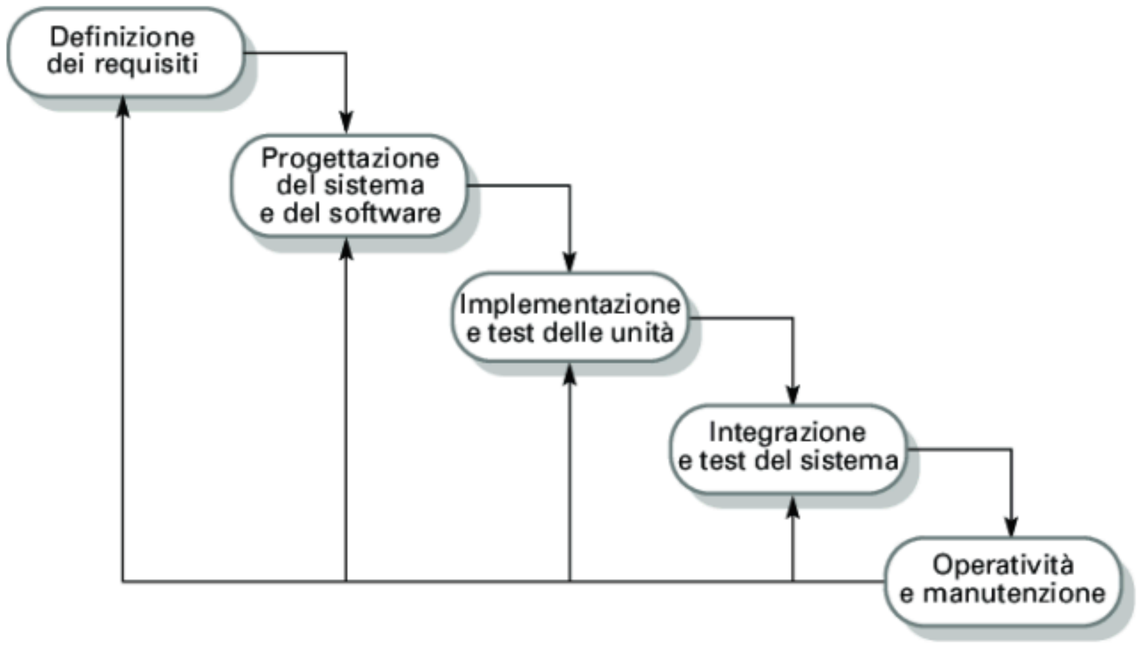
MODELLO A CASCATA E VARIANTI

MODELLO A CASCATA (WATERFALL MODEL)



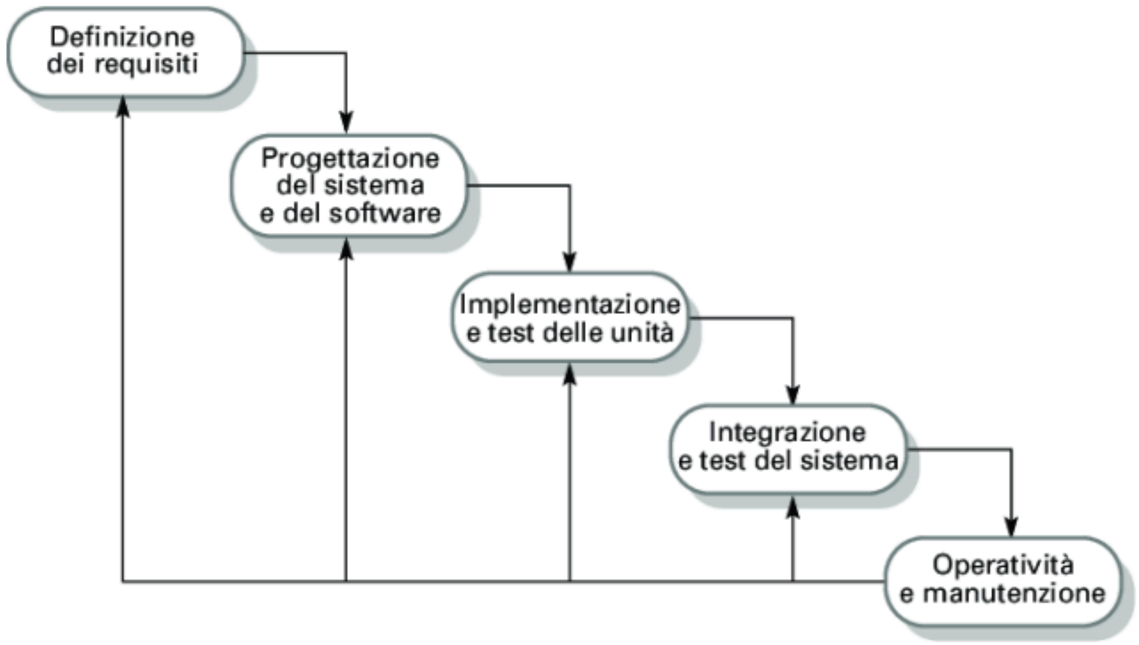
- Processo **plan-driven**
- Ciascuna fase segue quella successiva
- Bisogna pianificare tutte le attività di processo prima di iniziare lo sviluppo del sw
- Gli output di una fase sono gli input della fase successiva
- Le fasi del modello a cascata riflettono direttamente le attività di sviluppo fondamentali del software

MODELLO A CASCATA (WATERFALL MODEL)



- Processo **document-centric**, guidato dalla produzione di documentazione
- Il risultato di ogni fase è costituito da uno o più documenti approvati
- La fase successiva non dovrebbe partire prima che quella precedente sia finita ed i relativi documenti completati ed approvati

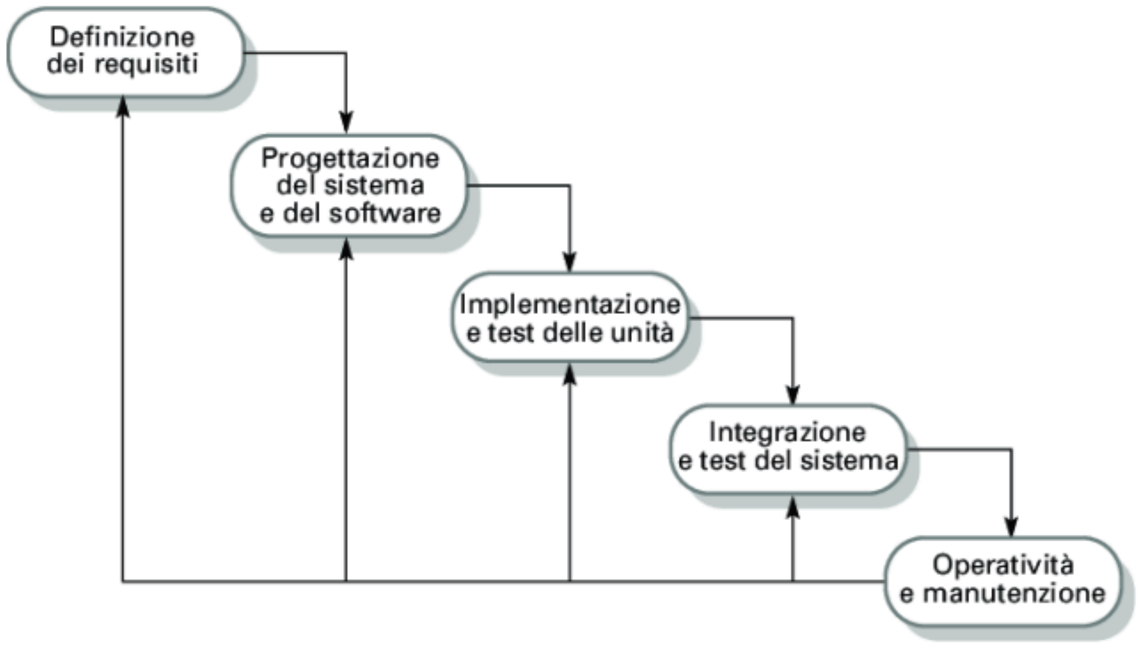
MODELLO A CASCATA (WATERFALL MODEL)



- Processo **rigido**
- i prodotti di una fase vengono "congelati", ovvero non sono più modificabili se non innescando un processo formale e sistematico di modifica
- La fine di ogni fase è un punto rilevante del processo (*milestone*)
- La definizione precisa di milestone e output è importante per misurare il progresso di un progetto

MODELLO A CASCATA (WATERFALL MODEL)

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine



- Processo **monolitico**
- Il cliente “vede” il software solo al completamento di tutte le fasi
- Se si commette un errore nei requisiti, viene rilevato solo alla fine, dopo il rilascio, con costi elevati
- Nella pratica, i processi non sono lineari ma ciascuna fase scambia feedback con le altre

MODELLO A CASCATA (WATERFALL MODEL): PRO E CONTRO

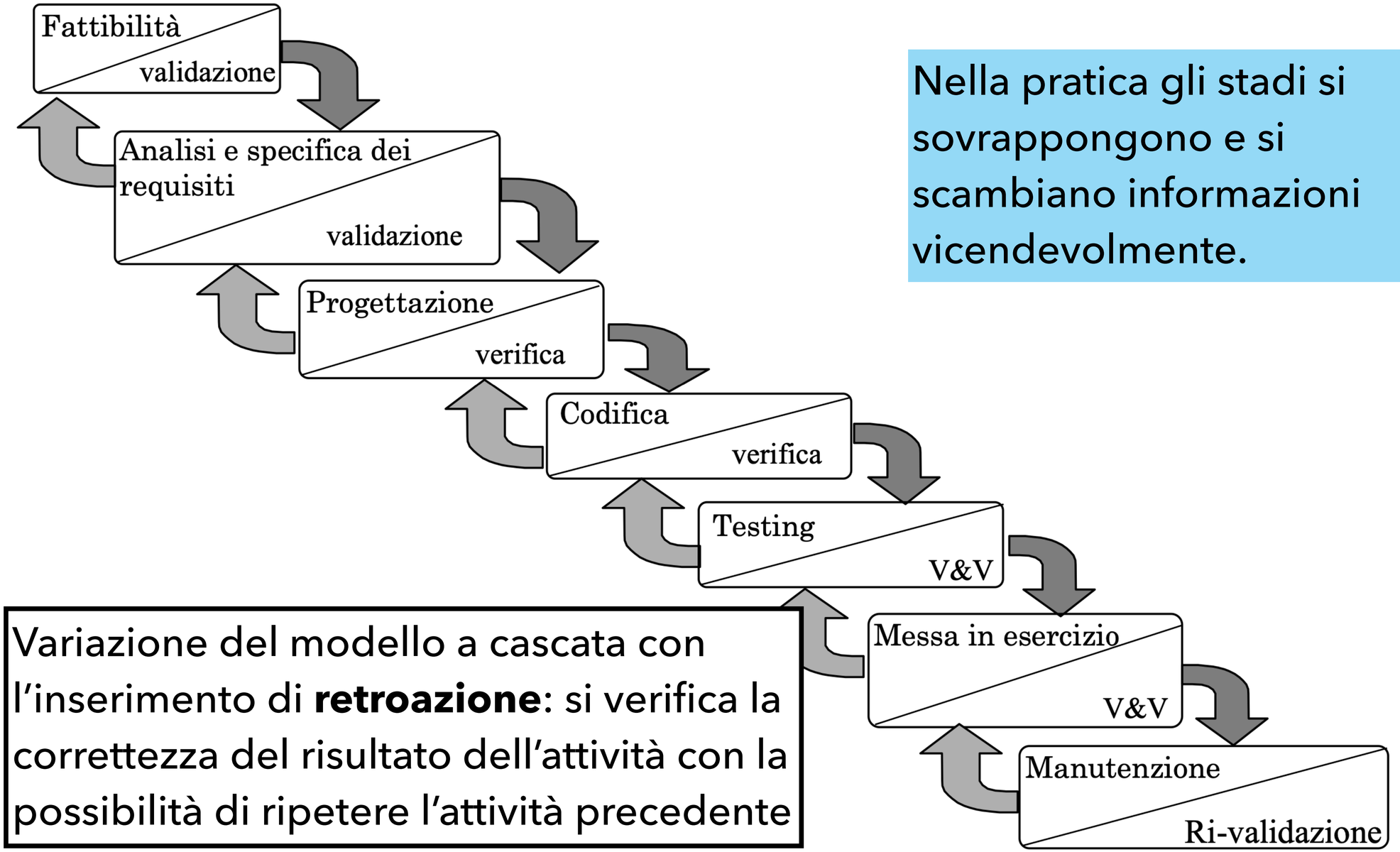
- **Vantaggi**
  - Le fasi sono ben definite
  - Gli output di ciascuna fase sono precisamente individuati
- **Svantaggi**
  - Richiede conoscenza immediata e stabilità dei requisiti. Ma è difficile avere requisiti congelati all'inizio del progetto, spesso sono poco chiari anche al cliente
  - Sviluppo di eccessiva documentazione, spesso non richiesta
  - Poco flessibile: difficile gestire necessità di modifiche che emergono durante l'esecuzione del processo, ad es. nuove richieste del cliente

MODELLO A CASCATA (WATERFALL MODEL)

- Adatto a software che richiedono una documentazione estremamente dettagliata come sistemi critici o grossi sistemi sviluppati da più società
- Adatto a sistemi integrati in cui il software deve interfacciarsi con sistemi hardware non flessibili
- Non adatto quando i requisiti cambiano rapidamente o in piccoli team ove la comunicazione avviene a livello informale
- In sintesi, il modello a cascata è adatto solo se i requisiti sono ben chiari fin dall'inizio ed è difficile che cambino durante lo sviluppo. Ma, nella realtà pochi software presentano requisiti stabili

MODELLO A CASCATA CON RETROAZIONE

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

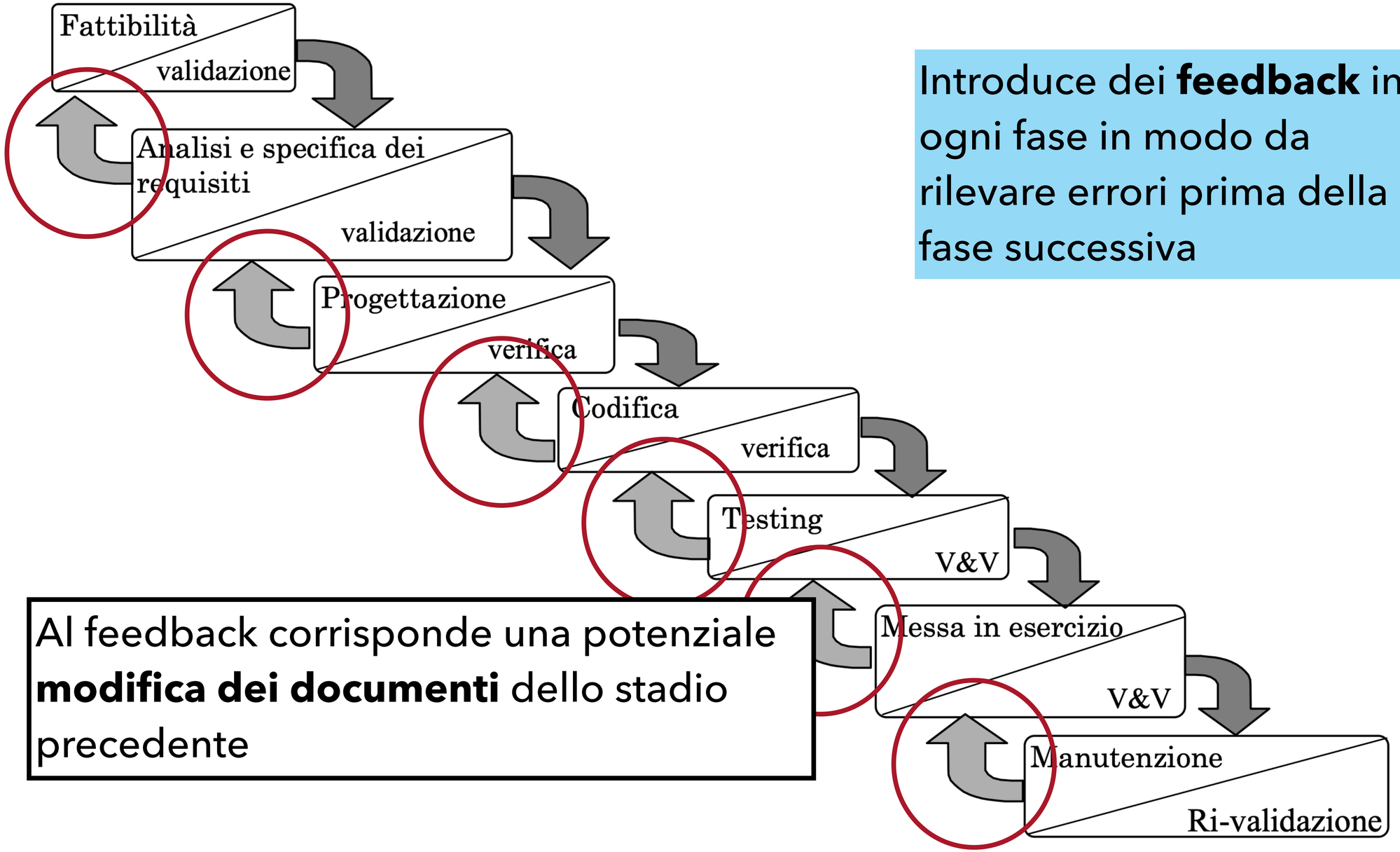


Nella pratica gli stadi si sovrappongono e si scambiano informazioni vicendevolmente.

Variazione del modello a cascata con l'inserimento di **retroazione**: si verifica la correttezza del risultato dell'attività con la possibilità di ripetere l'attività precedente

MODELLO A CASCATA CON RETROAZIONE

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine



Introduce dei **feedback** in ogni fase in modo da rilevare errori prima della fase successiva

Al feedback corrisponde una potenziale **modifica dei documenti** dello stadio precedente

MODELLO A CASCATA CON RETROAZIONE

- Mitiga la monoliticità del modello a cascata tradizionale: non bisogna aspettare il termine del processo per modificare il prodotto
- Non è comunque completamente flessibile rispetto a cambiamenti che possono avvenire in qualunque momento del processo
- Modello utile quando si prevede che il sistema sarà poco soggetto a cambiamenti

ESTENSIONE DEL MODELLO A CASCATA: MODELLO A V

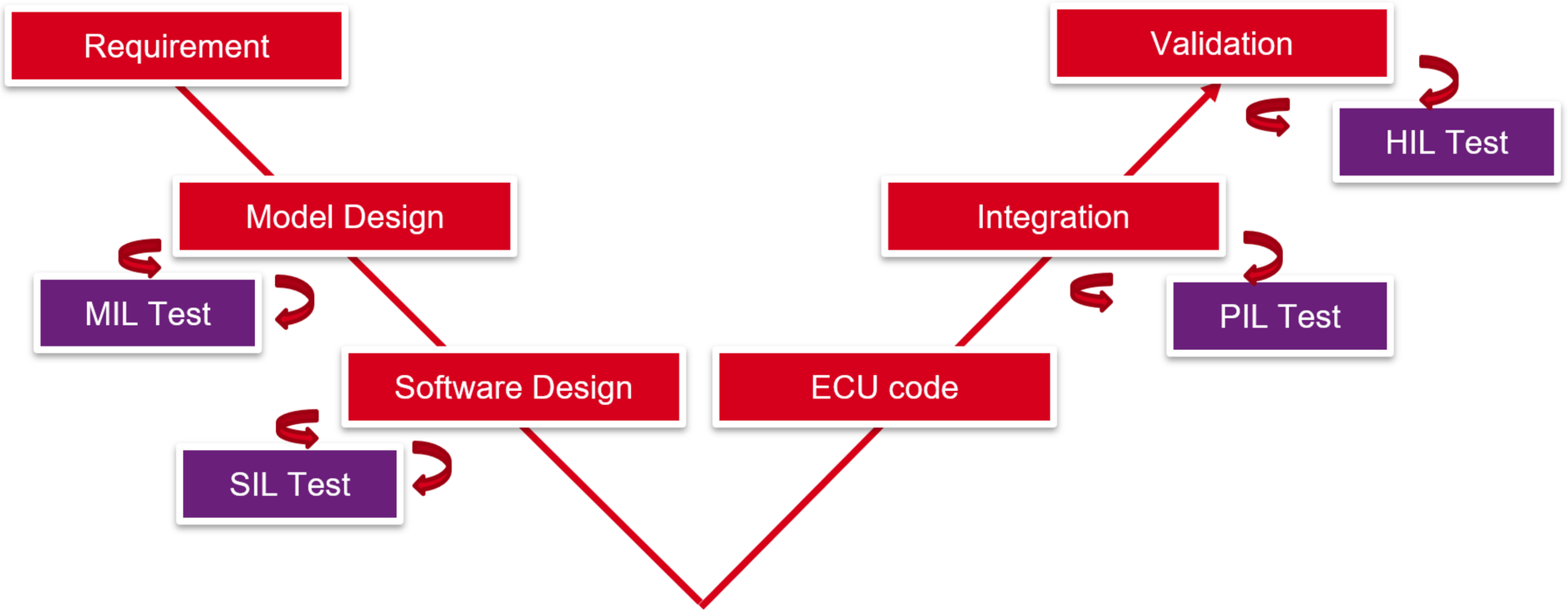
- Le attività del ramo superiore (**progetto**) sono collegate a quelle del ramo inferiore (**V&V**)
- In ogni fase di progetto il team di progetto definisce il corrispondente piano di test della fase di V&V
- V&V è guidato da un insieme di piani di test ed è eseguito da un team indipendente da quello di sviluppo

ESTENSIONE DEL MODELLO A CASCATA: MODELLO A V

- Se si trova un errore in una fase di V&V si rieseguono le fasi di progetto collegate
- Questo modello può anticipare la validazione dei requisiti da parte del cliente e quindi garantire il rilevamento precoce di eventuali errori

ESEMPIO: SVILUPPO ELECTRONIC CONTROL UNIT IN AMBITO AUTOMOTIVE

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine



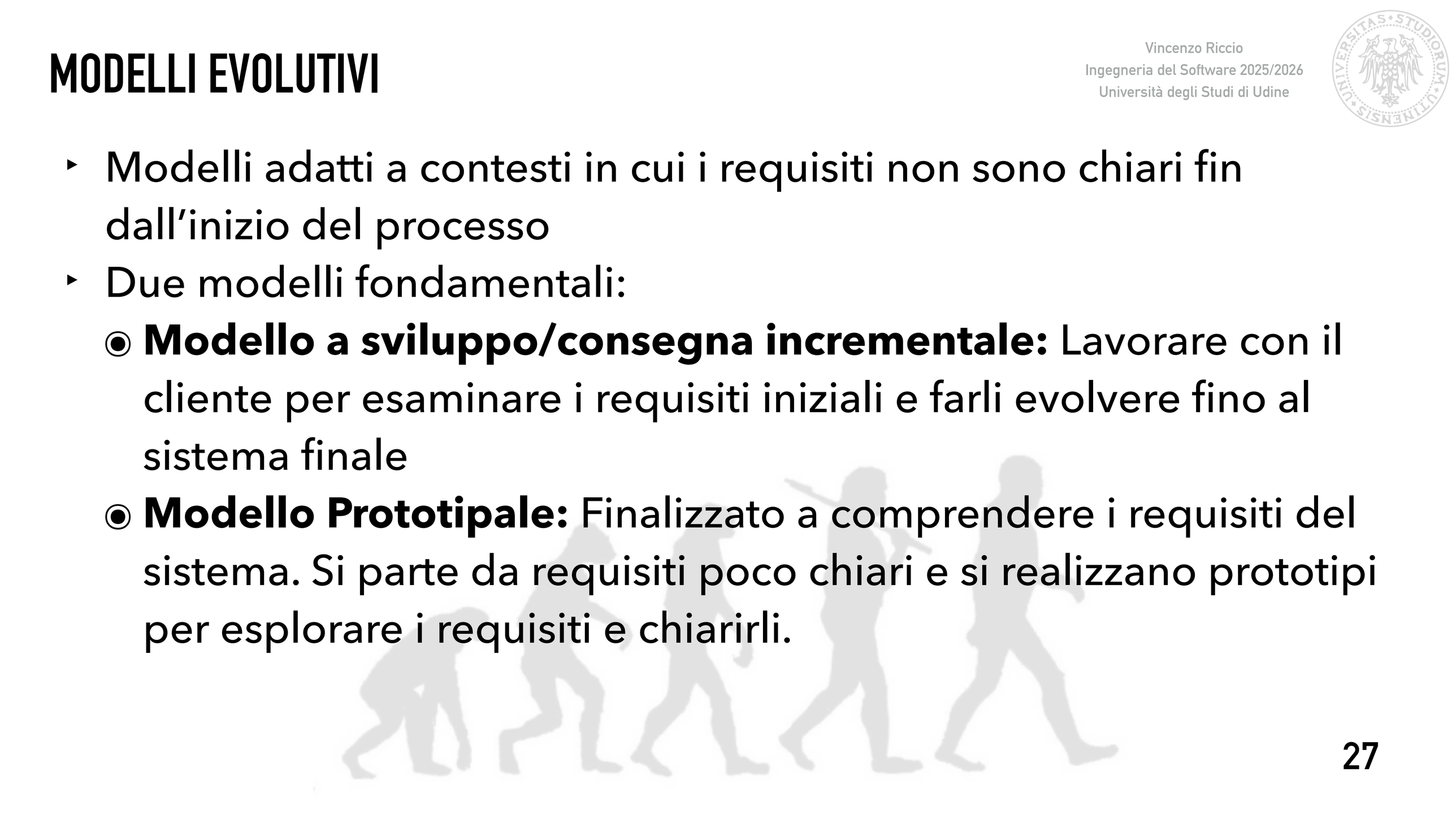
Fonte: https://blog.nashtechglobal.com/

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

MODELLI DI PROCESSI SOFTWARE

MODELLI EVOLUTIVI

MODELLI EVOLUTIVI

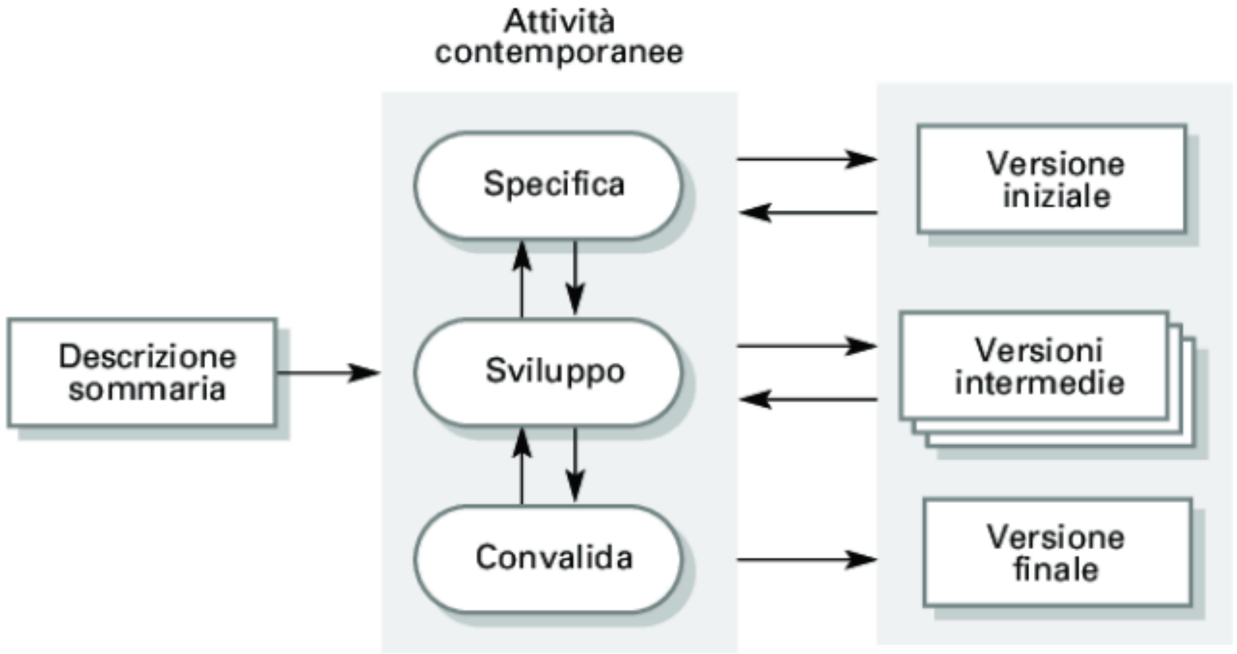


- Modelli adatti a contesti in cui i requisiti non sono chiari fin dall'inizio del processo
- Due modelli fondamentali:
  - **Modello a sviluppo/consegna incrementale:** Lavorare con il cliente per esaminare i requisiti iniziali e farli evolvere fino al sistema finale
  - **Modello Prototipale:** Finalizzato a comprendere i requisiti del sistema. Si parte da requisiti poco chiari e si realizzano prototipi per esplorare i requisiti e chiarirli.

MODELLO A SVILUPPO INCREMENTALE

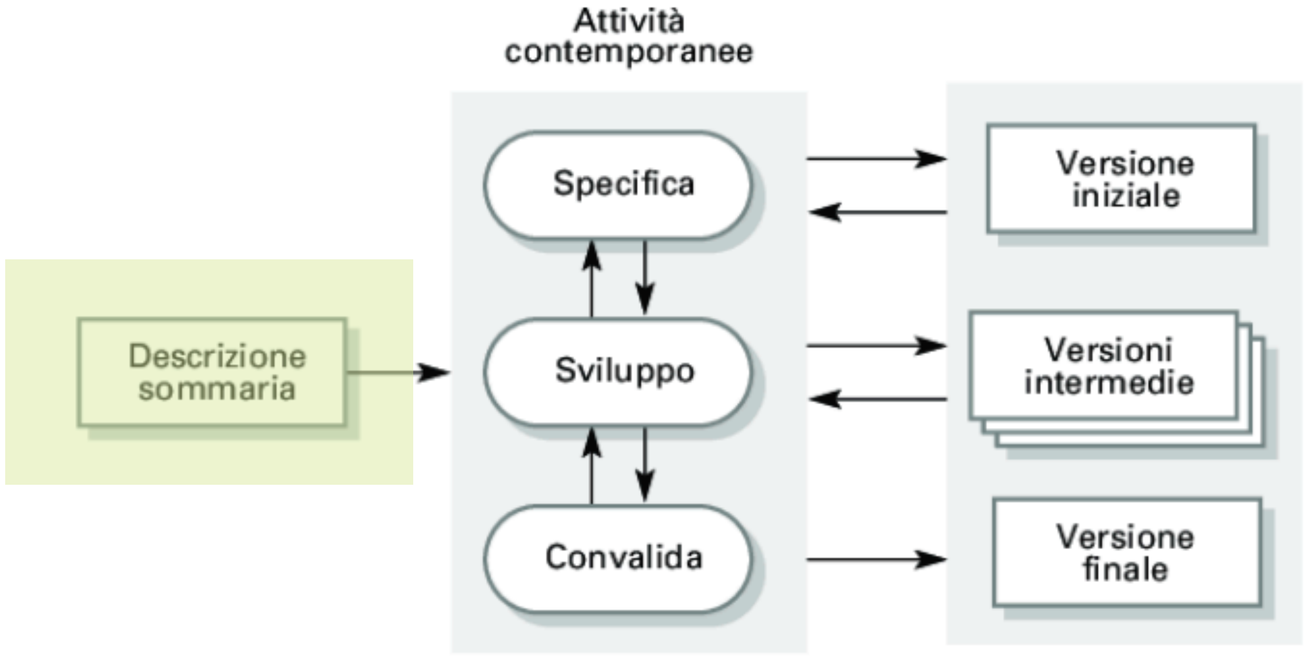
Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

- Basato sull'idea di sviluppare un'implementazione iniziale, esporla agli utenti e perfezionarla attraverso versioni successive (**incrementi**), finché non si ottiene il sistema richiesto



MODELLO A SVILUPPO INCREMENTALE

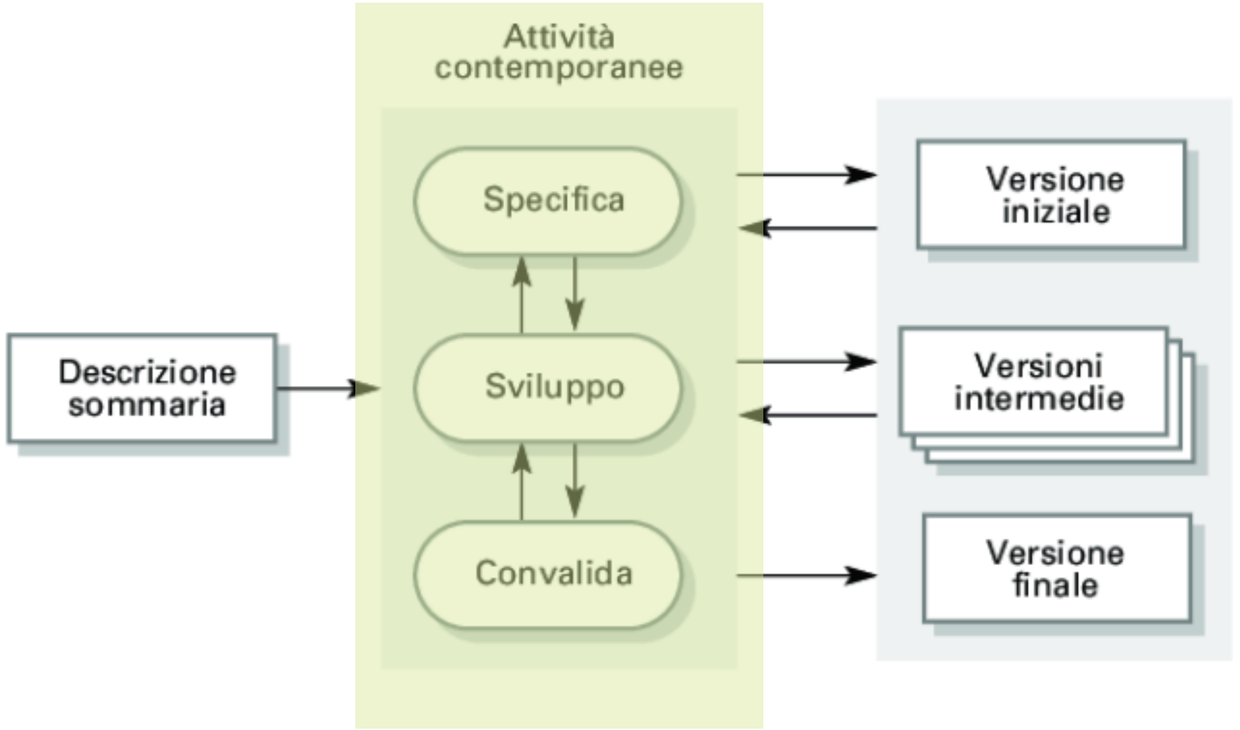
Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine



- Parte da pochi requisiti ben compresi o da una descrizione sommaria dei requisiti

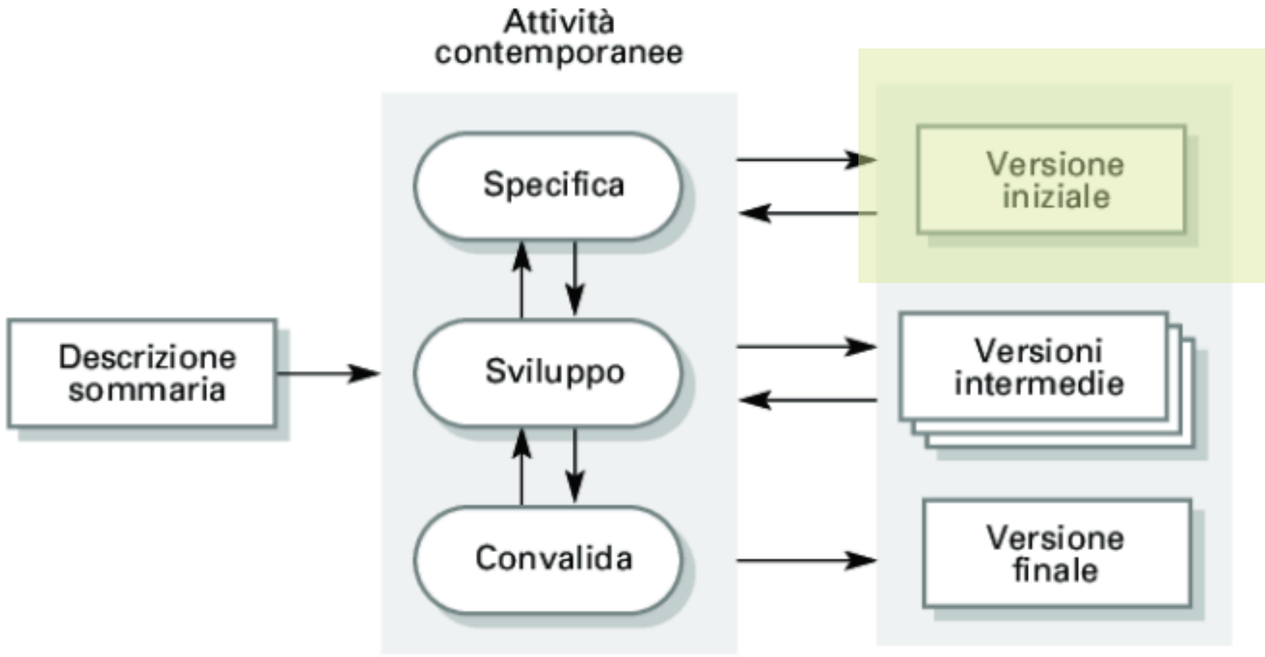
MODELLO A SVILUPPO INCREMENTALE

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine



- Le attività di specifica, sviluppo e convalida sono intrecciate anziché separate, con feedback veloci tra le varie attività

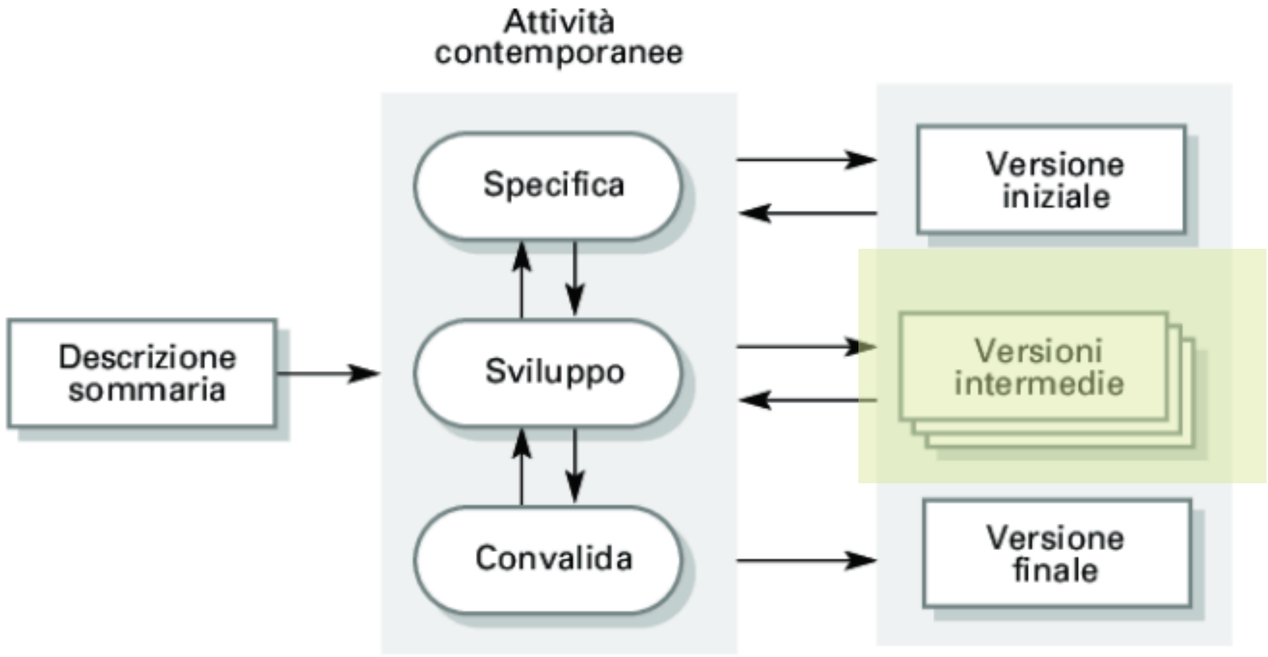
MODELLO A SVILUPPO INCREMENTALE



- La versione iniziale che implementa i requisiti fondamentali è esposta agli utenti (clienti o a proxy del clienti)
- Più facile dare feedback su una versione iniziale (es. demo) piuttosto che su documenti

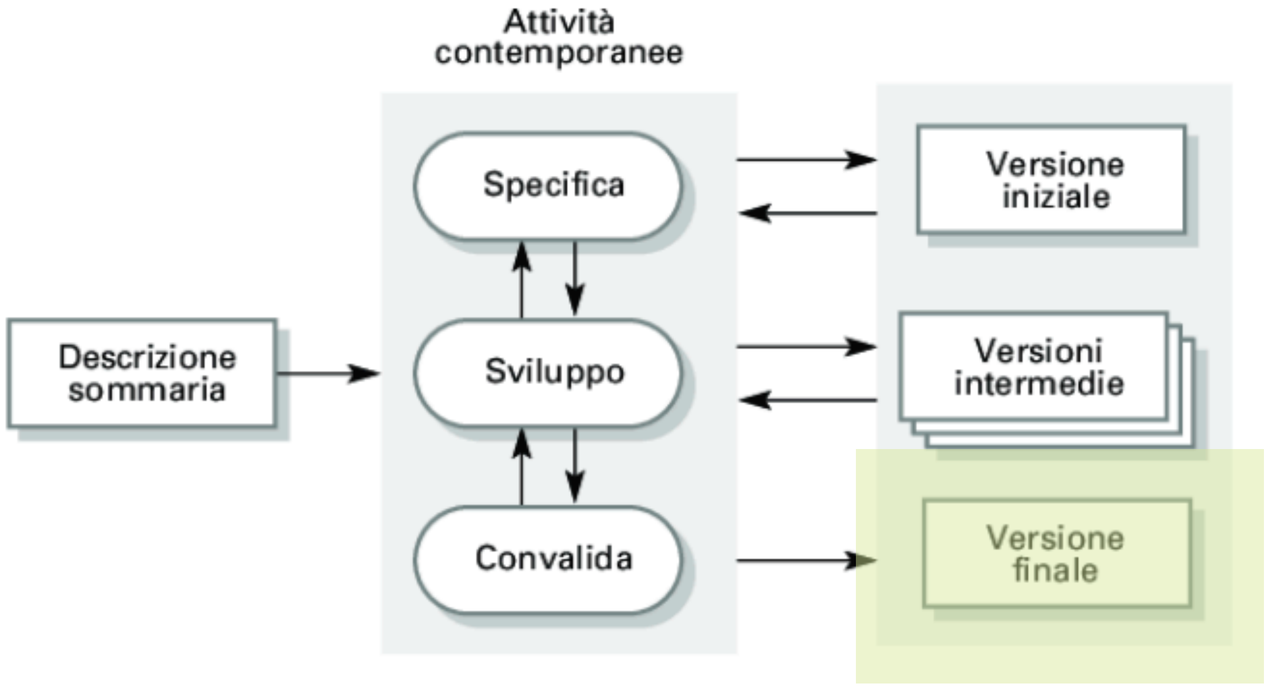
MODELLO A SVILUPPO INCREMENTALE

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine



- Ulteriori funzionalità e requisiti (ad es. proposti dal cliente) sono implementati in versioni intermedie successive
- Le versioni intermedie non sono solitamente mostrate al cliente

MODELLO A SVILUPPO INCREMENTALE



- L'ultimo incremento è la versione finale del sistema che viene rilasciata al cliente
- La versione finale corrisponde ad un'evoluzione della versione iniziale

MODELLO A CONSEGNA INCREMENTALE

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

- Il sistema non è consegnato direttamente nella sua forma finale alla fine del progetto. Alcuni degli incrementi sviluppati (non per forza tutti) sono consegnati ai clienti e installati nel loro ambiente operativo
- **Vantaggio:** il cliente usa l'incremento nell'ambiente operativo reale, pertanto il feedback è più realistico
- **Limitazione:** Gli utenti devono avere tempo sufficiente per sperimentare ciascun incremento

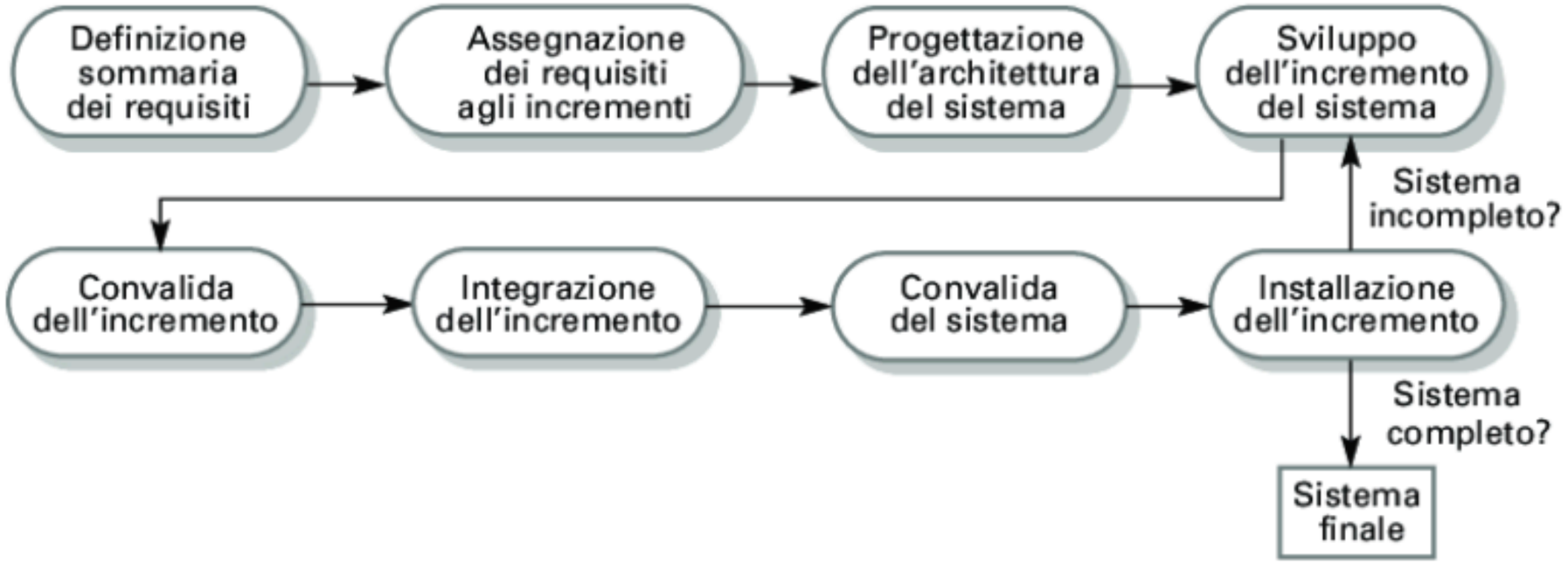
MODELLO A CONSEGNA INCREMENTALE

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

- Ogni incremento rilascia parte delle funzionalità richieste
- Ai requisiti utente vengono assegnati livelli di priorità. I requisiti a priorità maggiore vengono rilasciati per primi
- I requisiti di un incremento sono “congelati” (non possono essere modificati) dopo che tale incremento è stato consegnato. Gli altri requisiti invece possono evolvere

MODELLO A CONSEGNA INCREMENTALE

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine



- Funzionalità comuni a più requisiti dovrebbero essere individuate tempestivamente e implementate all'inizio del processo

ESEMPIO DI MODELLO A CONSEGNA INCREMENTALE PLAN-DRIVEN

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

Requisiti: R1, R2, R3
Architettura: M1, M2, M3, M4
Pianificazione: 3 incrementi

Iterazione 1
R1, richiede M1, M2
Sviluppare e integrare M1, M2
Consegnare R1

Iterazione 2
R2, richiede M1, M3
Sviluppare M3, integrare M1, M2, M3
Consegnare R1 + R2

Iterazione 3
R3, richiede M3, M4
Sviluppare M4, integrare M1, M2, M3, M4
Consegnare R1 + R2 + R3

CONSEGNA VS SVILUPPO INCREMENTALE

- Nello **sviluppo incrementale** la valutazione della prima versione è effettuata da un proxy degli utenti finali in un ambiente operativo diverso da quello target. Le versioni intermedie non sono solitamente rilasciate al cliente

- La **consegna incrementale** invece permette una valutazione più realistica e rappresentativa dell'utilizzo reale del software perché ciascun incremento può essere rilasciato agli utenti finali nell'ambiente operativo del sistema

CONSEGNA E SVILUPPO INCREMENTALE

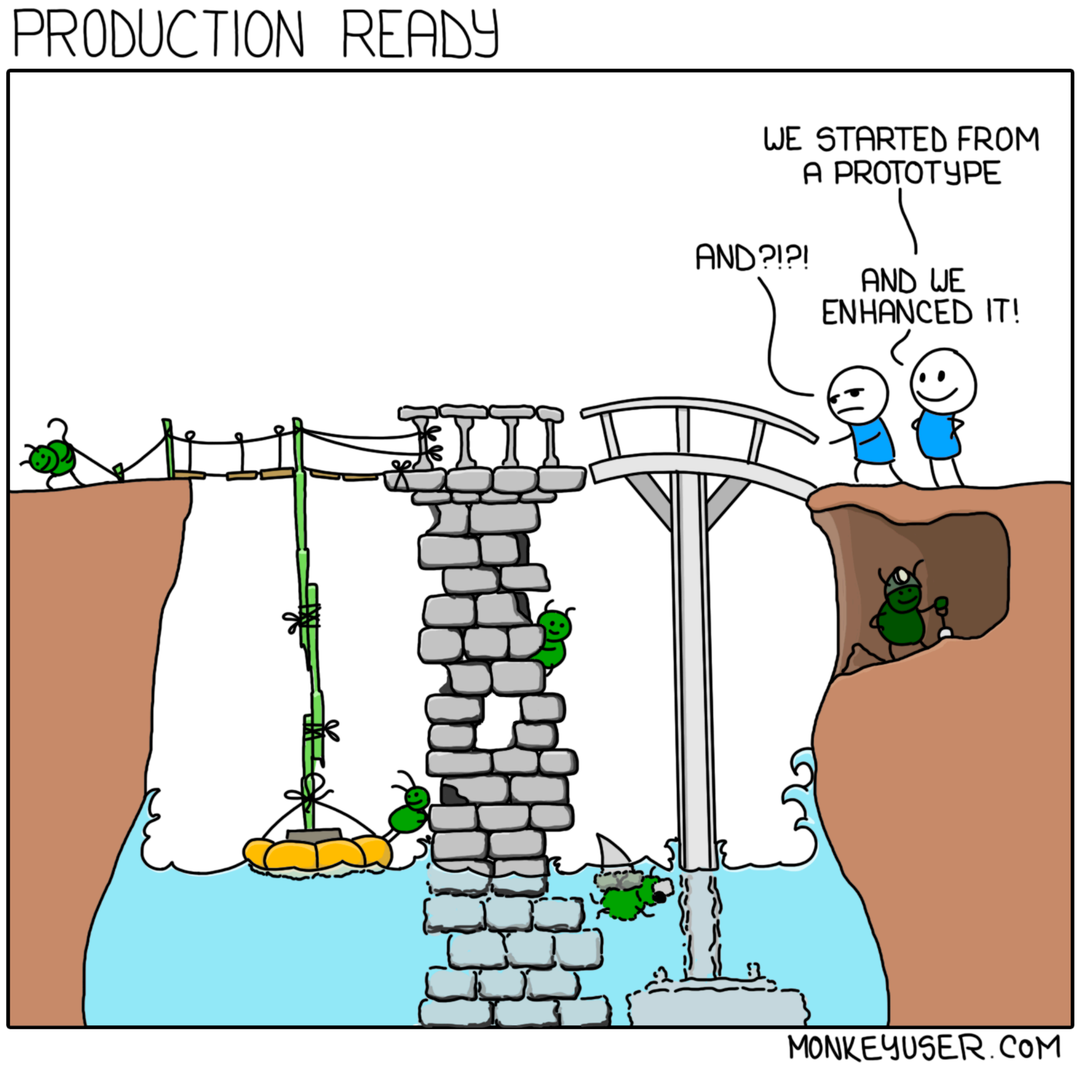
- **Vantaggi:**
  - Rapido feedback del cliente su una versione preliminare del software, invece che su documenti di progetto
  - Possibilità di far cambiare i requisiti prima della consegna finale del prodotto, riducendo i costi di modifica
  - Possibilità di consegnare ai clienti versioni preliminari in cui le funzionalità fondamentali sono già implementate
  - I primi incrementi possono essere utilizzati per dedurre requisiti per gli incrementi successivi
  - Le funzionalità con priorità più elevata sono testate più approfonditamente

CONSEGNA E SVILUPPO INCREMENTALE

- **Problemi**
  - Mancanza di visibilità del processo (è anti-economico documentare ogni versione del sistema)
  - I sistemi diventano spesso mal strutturati per i continui cambiamenti
- **Applicabilità**
  - Componenti di piccole e medie dimensioni (per es. l'interfaccia utente)
  - Sistemi destinati a vita breve
  - Sistemi i cui requisiti è probabile che cambino durante lo sviluppo

MODELLO PROTOTIPALE

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine



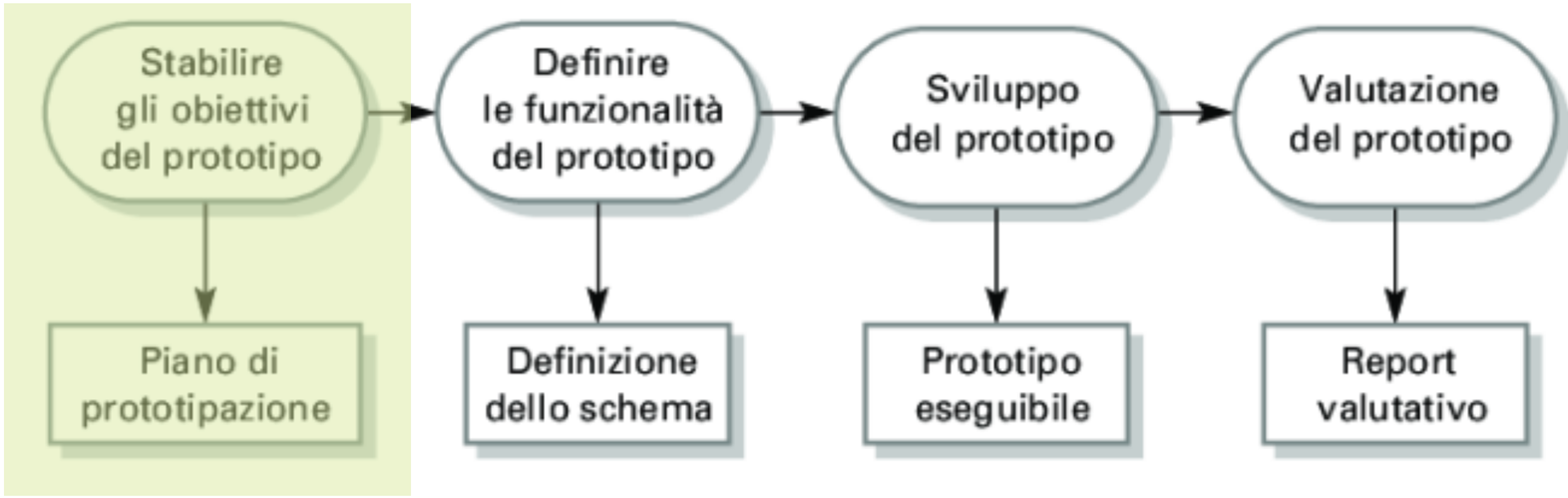
- **Prototipo:** versione iniziale di un intero sistema software o di parte di esso
- Sviluppato rapidamente per contenere i costi e poter sperimentare con il cliente prima della consegna, nelle fasi iniziali del processo

MODELLO PROTOTIPALE

- Il prototipo è **usa e getta**, deve essere scartato dopo la sua validazione poiché non è una buona base per sviluppare il sistema finale
- Pur realizzando le funzionalità richieste, potrebbe non rispettare aspetti fondamentali come le prestazioni o il rispetto di standard aziendali
- Potrebbe non essere documentato in modo appropriato
- La rapidità dello sviluppo ed i frequenti cambiamenti potrebbero deteriorarne la qualità

MODELLO PROTOTIPALE

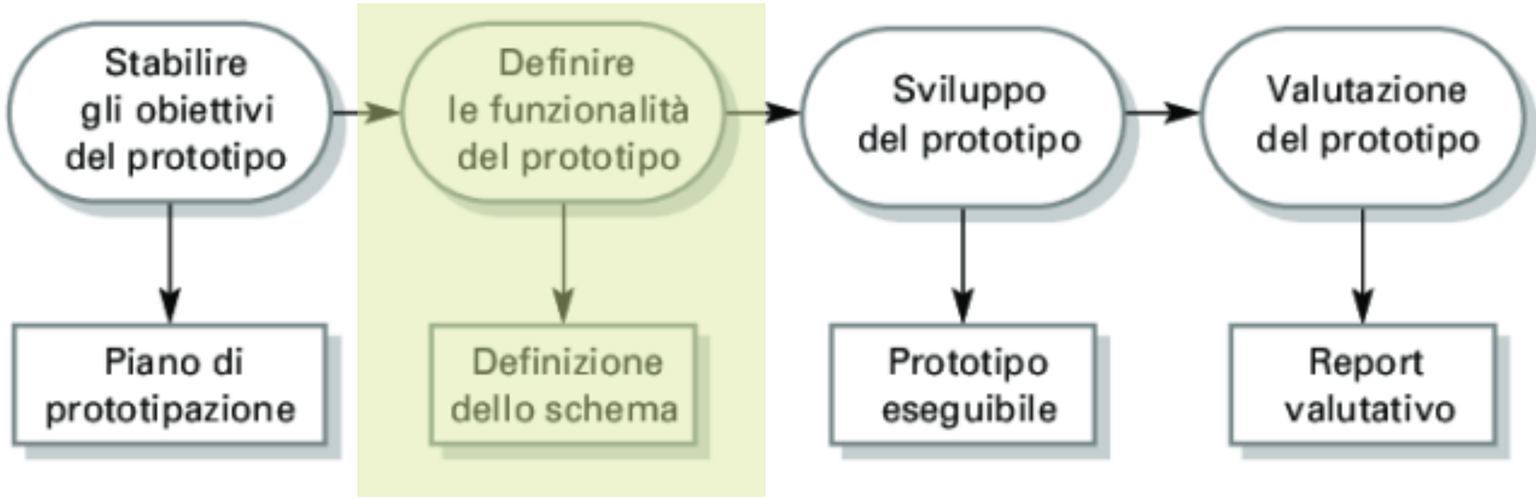
Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine



- Bisogna stabilire esplicitamente gli obiettivi del prototipo all'inizio del processo
- Se gli obiettivi non sono espliciti o chiari, il management o gli utenti finali possono fraintendere la funzione del prototipo e la prototipazione risulta inefficace

MODELLO PROTOTIPALE

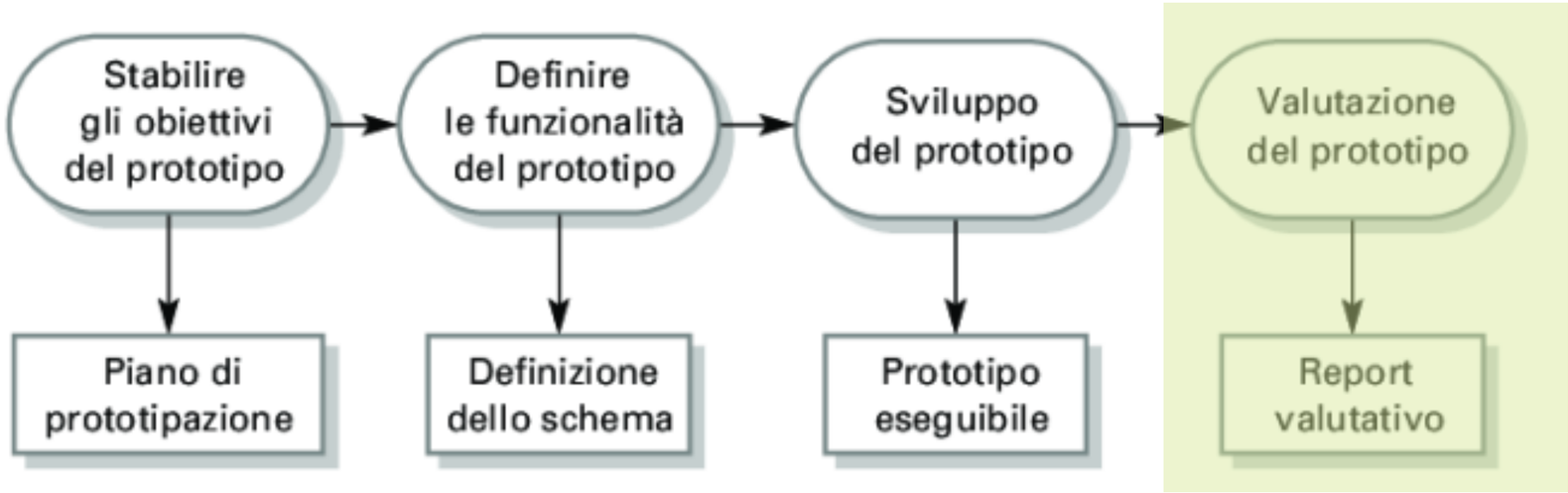
Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine



- Non tutte le funzionalità del sistema finale devono essere incluse nel prototipo, in modo da ridurre i costi di prototipazione
- Ad esempio, il prototipo può focalizzarsi su aree ancora non comprese bene

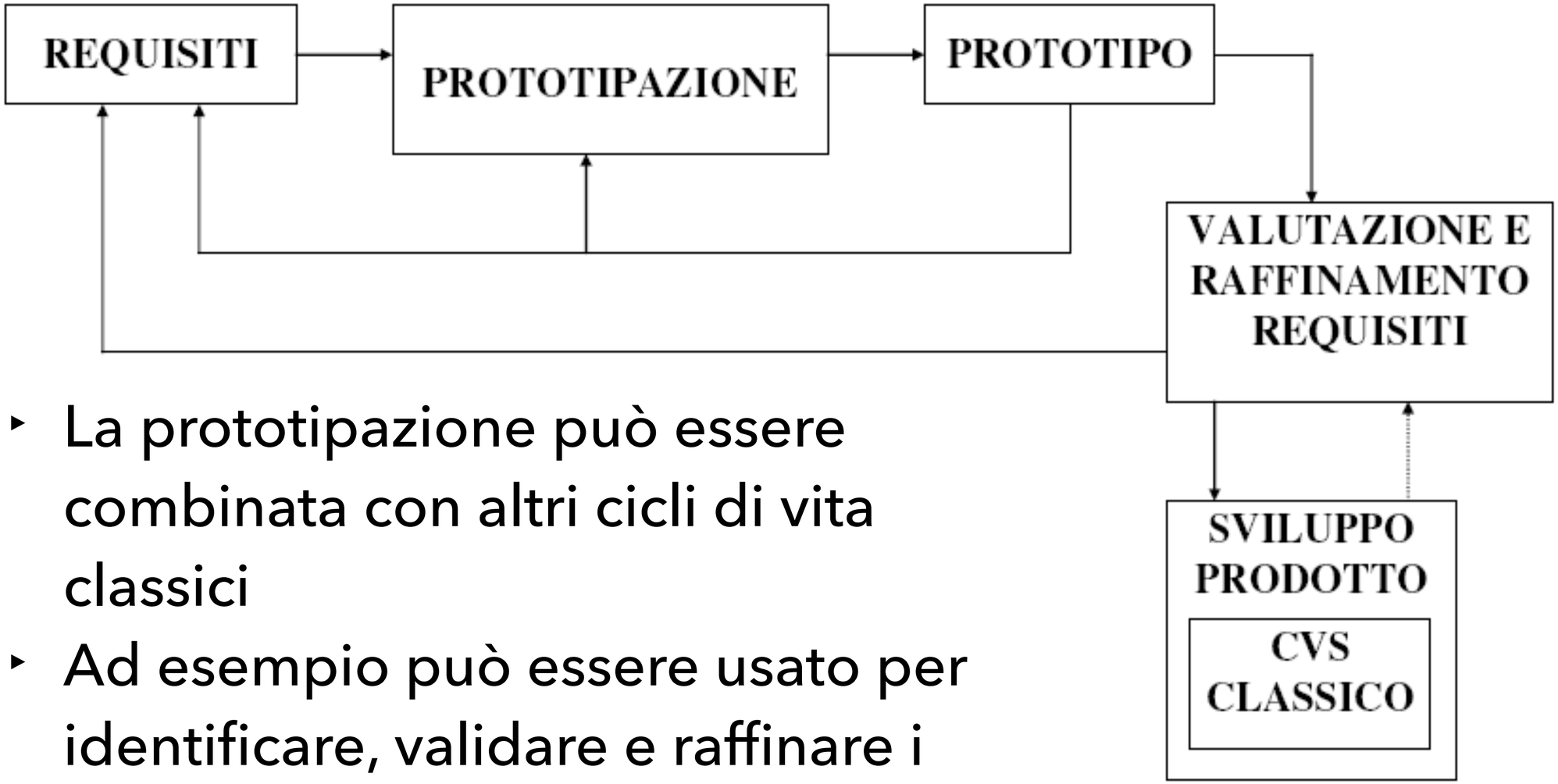
MODELLO PROTOTIPALE

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine



- Necessaria formazione utenti sull'utilizzo di ciascun prototipo prima di valutarlo
- Problema della rappresentatività:
  - Il prototipo non è parte del sistema reale (a differenza degli incrementi)
  - I valutatori potrebbero essere non rappresentativi degli utenti finali
  - Il modo di usare il prototipo può differire dall'utilizzo del sistema reale

MODELLO PROTOTIPALE

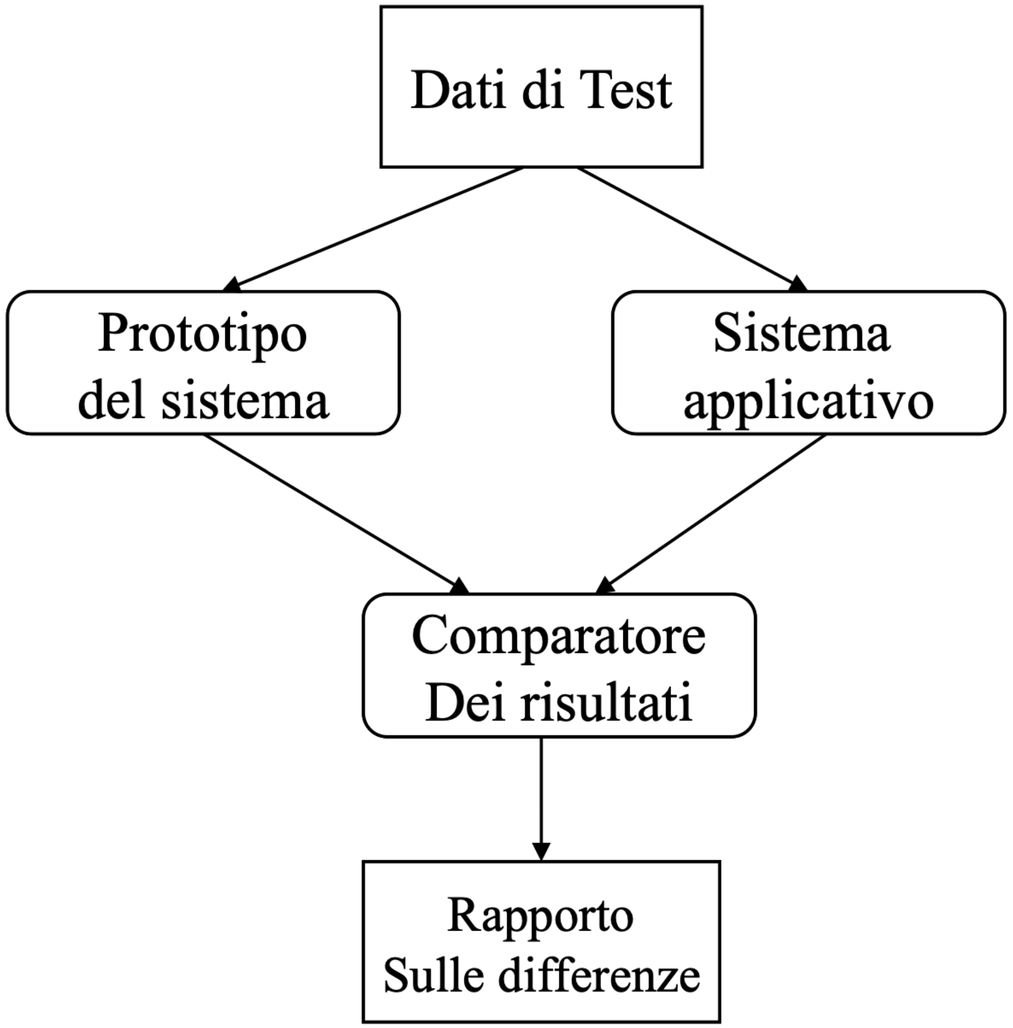


- La prototipazione può essere combinata con altri cicli di vita classici
- Ad esempio può essere usato per identificare, validare e raffinare i requisiti

MODELLO PROTOTIPALE

- Durante la fase di progettazione in un modello a cascata, la prototipazione può essere usata per valutare opzioni alternative nella progettazione

MODELLO PROTOTIPALE: BACK-TO-BACK TESTING



- Il prototipo può essere utilizzato nella fase di Validazione per controllare che il sistema sviluppato si comporti come modellato nel prototipo nelle fasi iniziali del progetto

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

MODELLI DI PROCESSI SOFTWARE

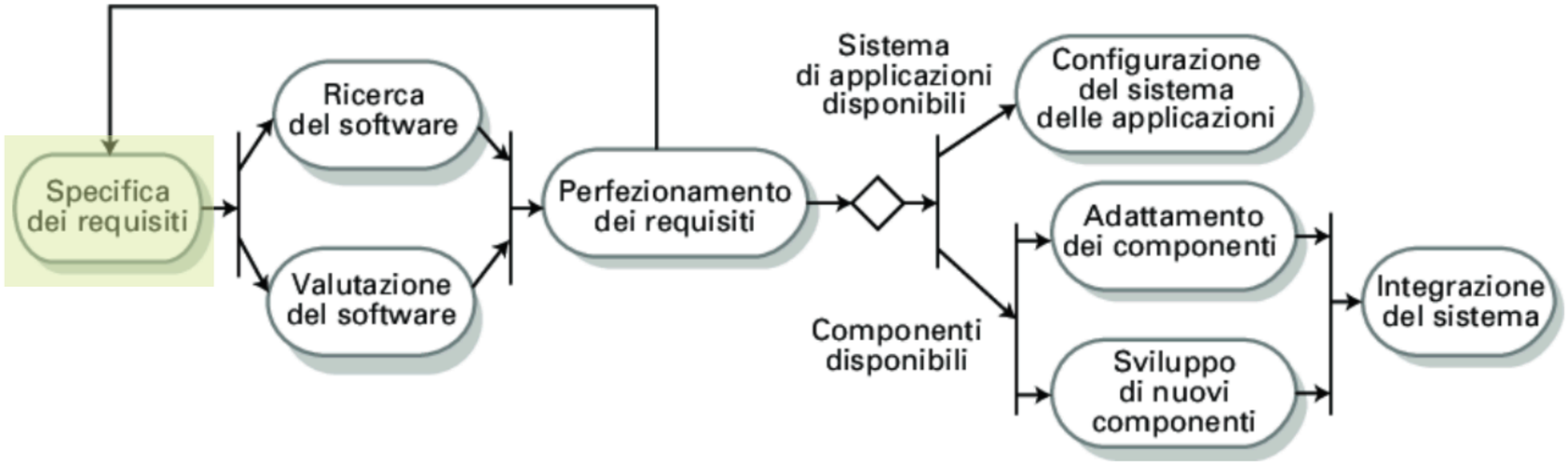
MODELLI ORIENTATI AL RIUSO

MODELLO ORIENTATO AL RIUSO

- Il riuso non avviene soltanto in maniera informale
- Approccio orientato al riuso di:
  - Componenti software riutilizzabili
  - Interi sistemi (COTS - Commercial-off-the-shelf)
- Sfrutta **framework di integrazione** per comporre i componenti
- **Componenti riutilizzabili** e **COTS** possono essere configurati per adattare il loro comportamento ai requisiti utente
- Approccio diffuso grazie ad appositi standard per la specifica dei componenti

MODELLO ORIENTATO AL RIUSO

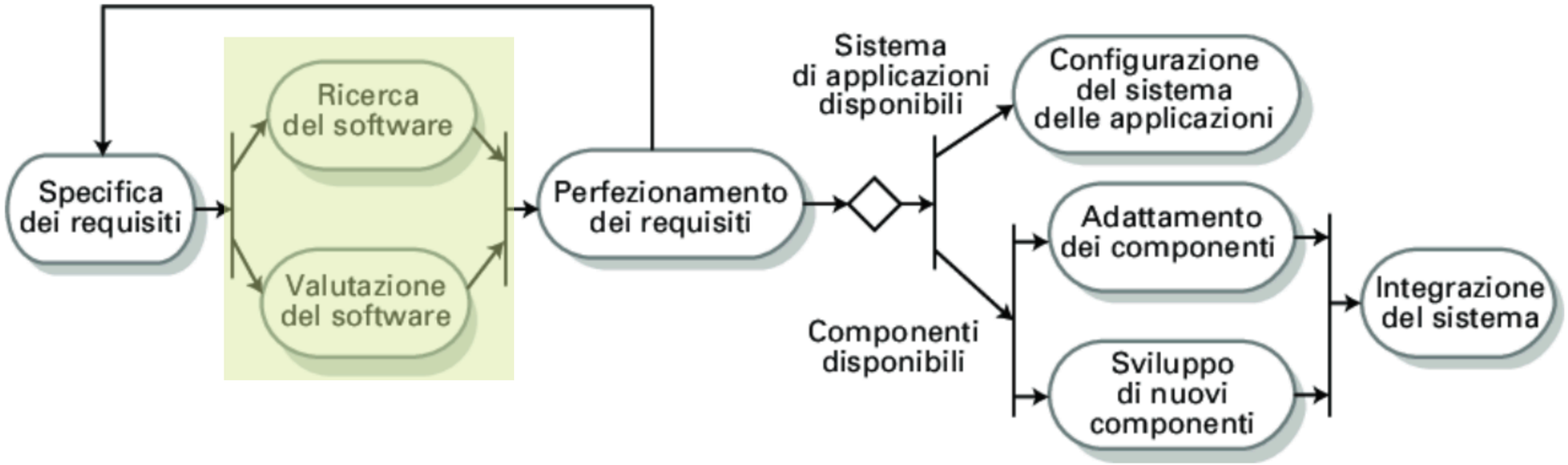
Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine



- I requisiti essenziali sono specificati in maniera non eccessivamente dettagliata (ad es., breve descrizione dei requisiti e delle funzionalità essenziali del sistema)

MODELLO ORIENTATO AL RIUSO

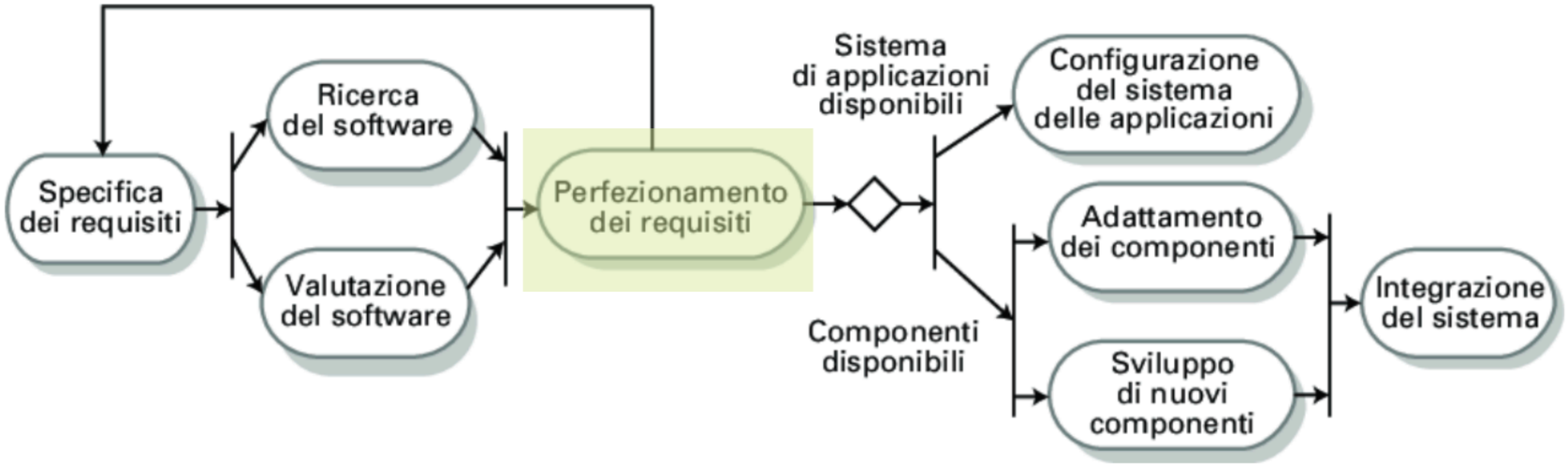
Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine



- Vengono ricercati i componenti e i sistemi che possono fornire le funzionalità specificate nei requisiti
- I candidati vengono valutati per vedere se soddisfano i requisiti essenziali e se sono disponibili per essere utilizzati

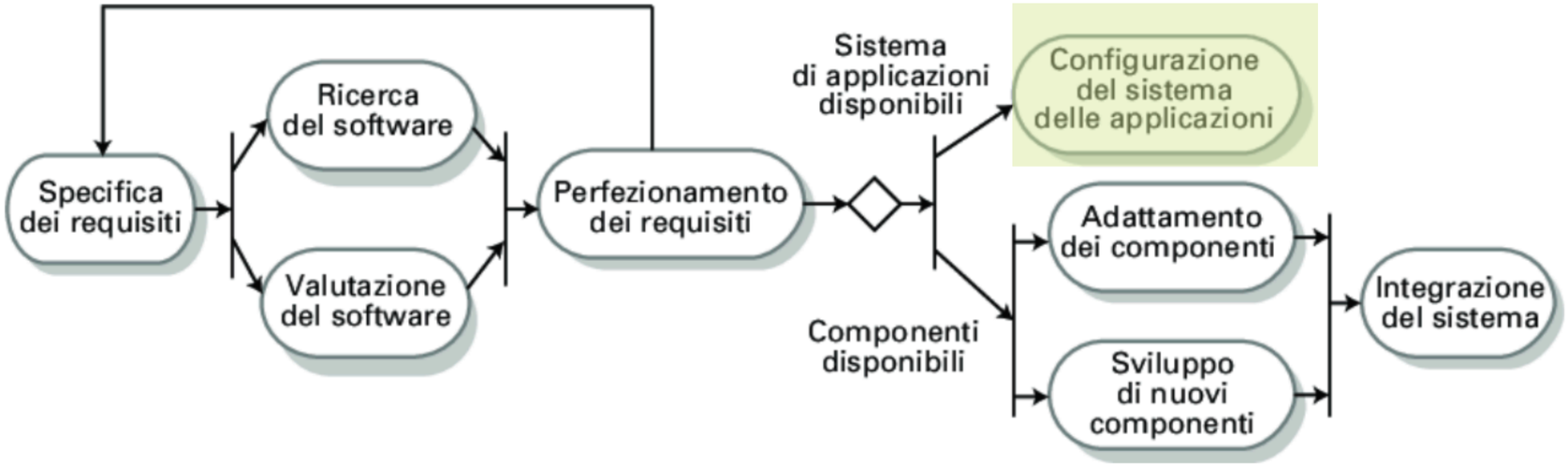
MODELLO ORIENTATO AL RIUSO

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine



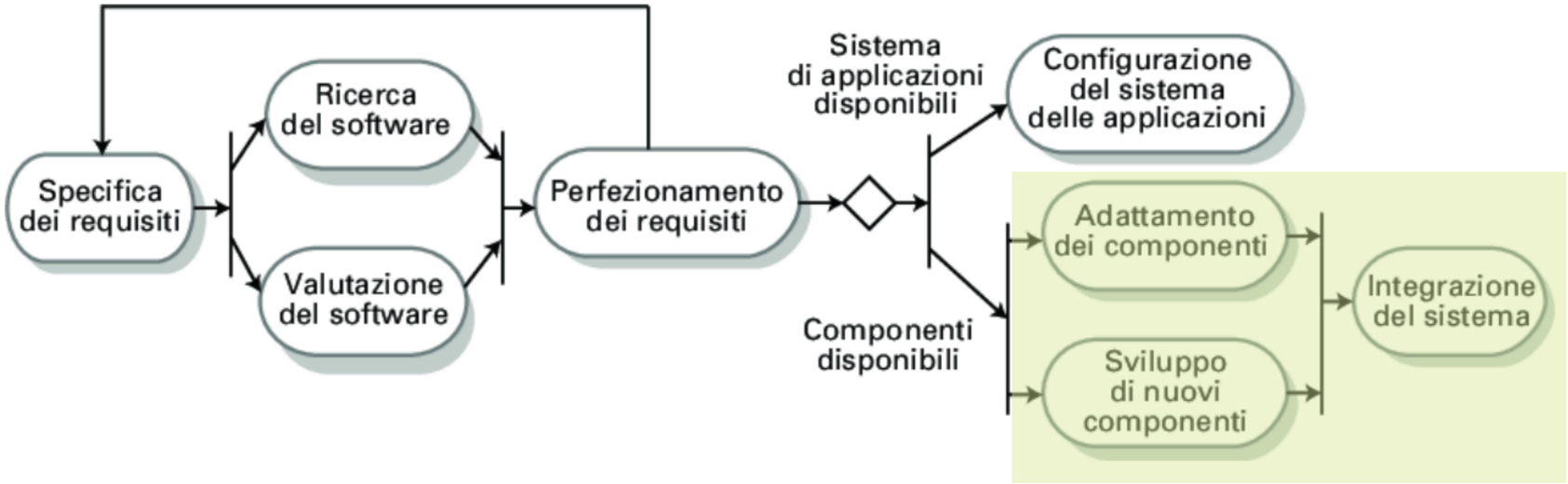
- I requisiti vengono perfezionati utilizzando le informazioni sulle applicazioni e sui componenti riutilizzabili che sono stati trovati
- La specifica viene aggiornata con i requisiti perfezionati

MODELLO ORIENTATO AL RIUSO



- Se è disponibile un sistema di applicazioni pronto all'uso che soddisfa i requisiti, esso può essere configurato per creare il nuovo sistema

MODELLO ORIENTATO AL RIUSO



- Se non è disponibile un sistema pronto all'uso, i singoli componenti riutilizzabili possono essere modificati e integrati con nuovi componenti appositamente sviluppati per creare il sistema finale

MODELLO ORIENTATO AL RIUSO: PRO E CONTRO

- **Vantaggi**
  - Riduce la quantità di software da sviluppare *ex novo*
  - Costi e rischi ridotti
  - Maggiore velocità nella consegna

- **Svantaggi**
  - Compromessi nei requisiti → Il sistema potrebbe non soddisfare tutte le reali necessità degli utenti
  - L'evoluzione dei componenti riutilizzati (ad es., le nuove versioni) non è controllata direttamente

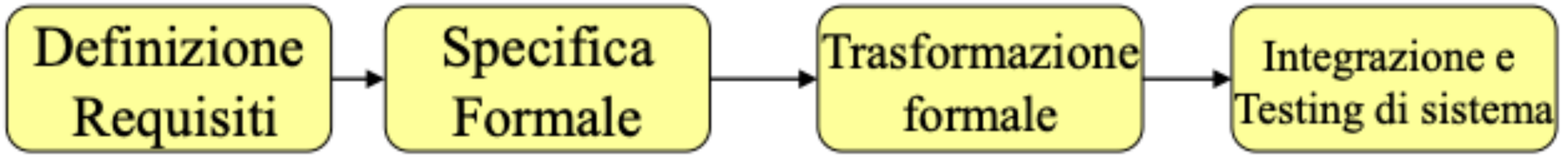
Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

MODELLI DI PROCESSI SOFTWARE

MODELLI TRASFORMAZIONALI

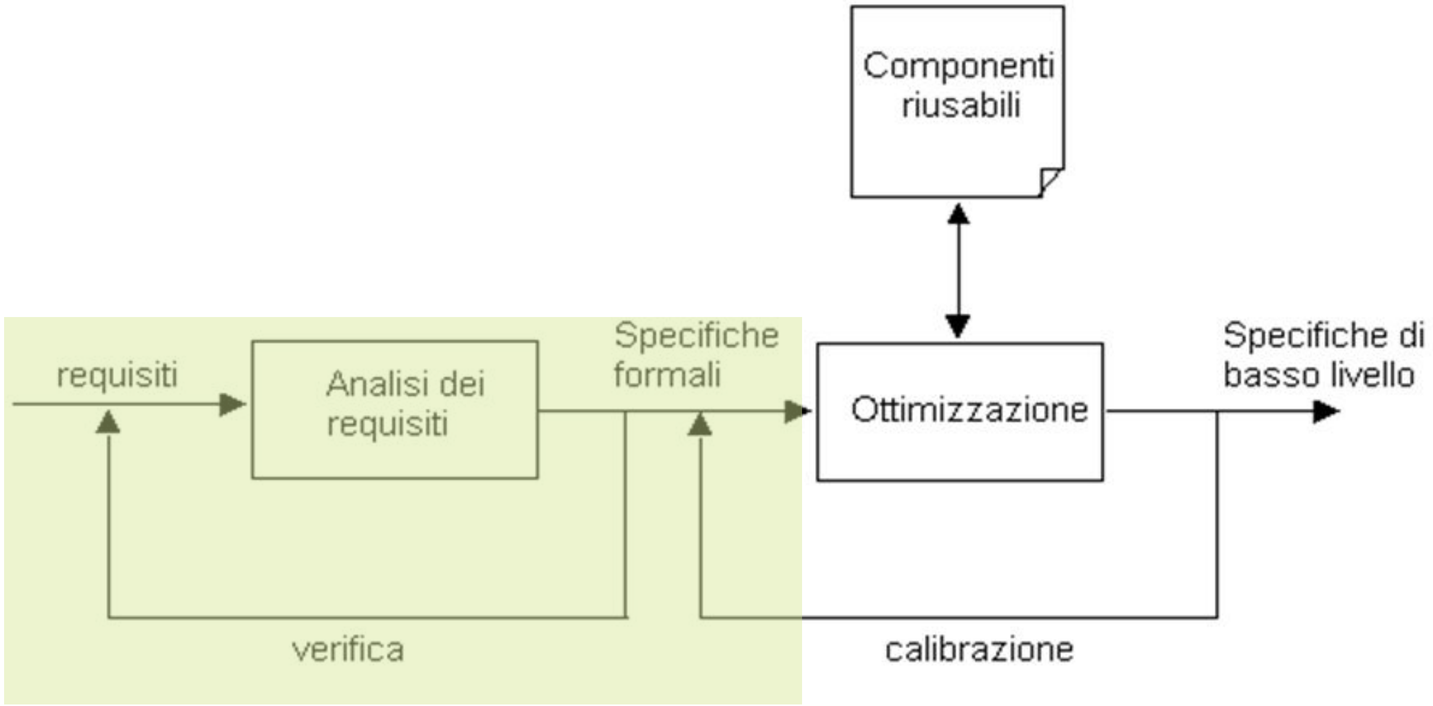
MODELLO TRASFORMAZIONALE

- Le specifiche sono definite attraverso linguaggi formali
  - Specifiche algebriche (es. per tipi di dato astratto)
  - Modelli di stato
- Uso di tecniche di model checking per provare la correttezza
- Le specifiche formali sono trasformate automaticamente in codice
  - La correttezza è preservata
  - La verifica è ottenuta implicitamente



MODELLO TRASFORMAZIONALE (CENNI)

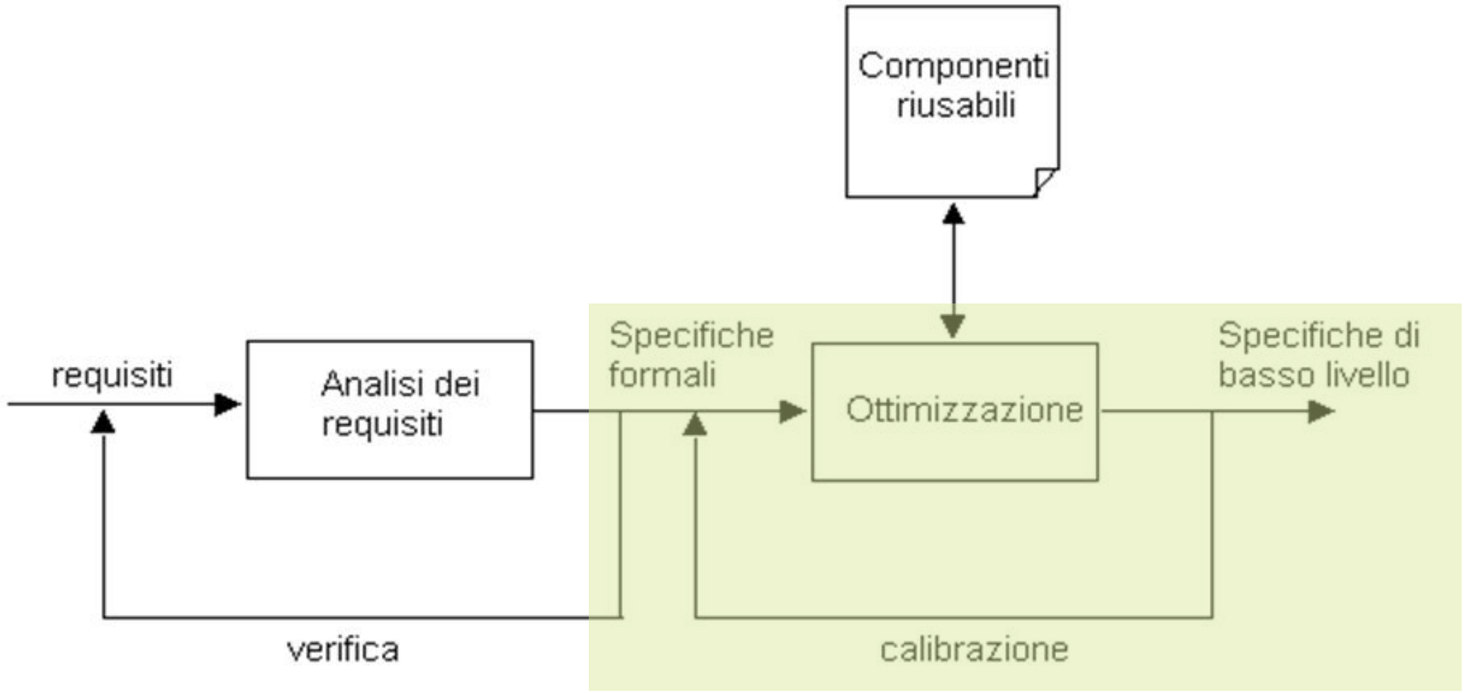
Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine



- I requisiti sono specificati formalmente nella fase di analisi
- Comprensione chiara e non ambigua dei requisiti
- Le specifiche sono verificate automaticamente prima di essere trasformate da opportuni strumenti

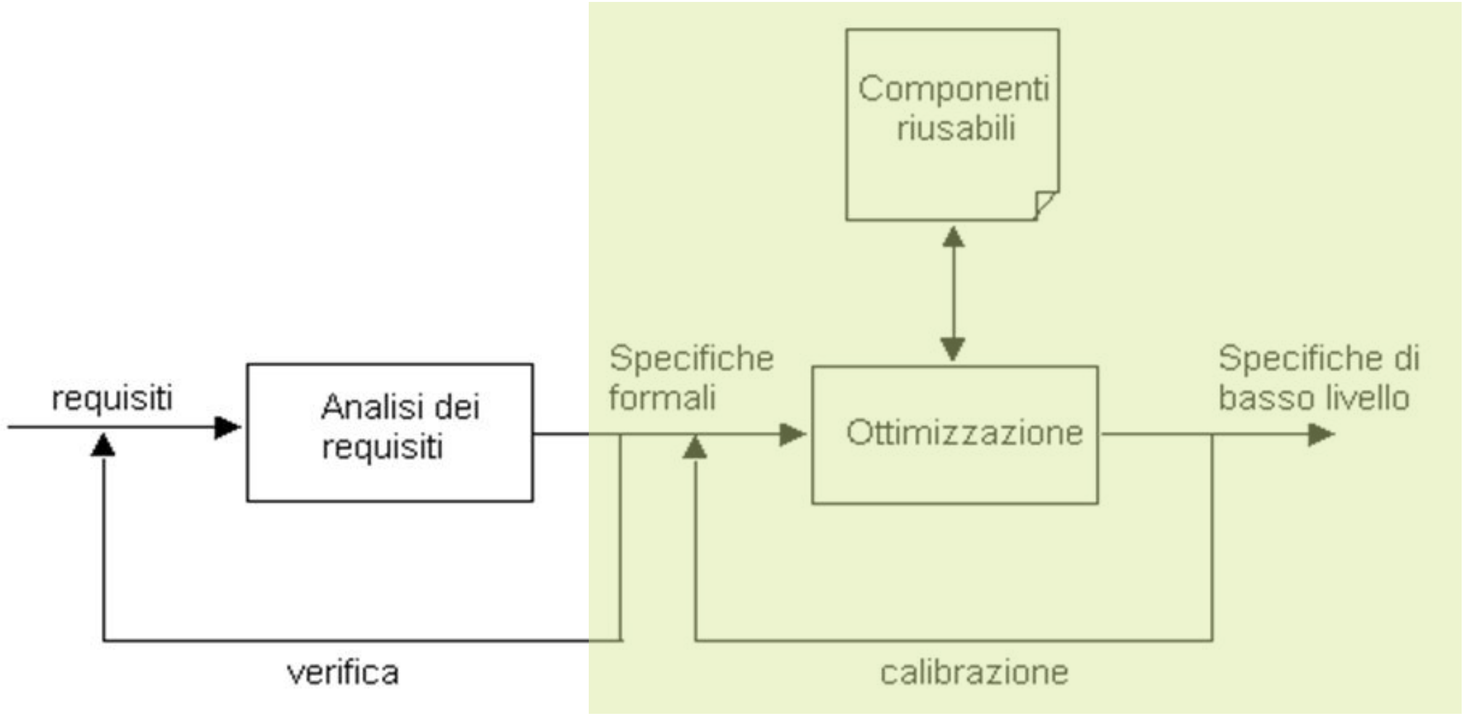
MODELLO TRASFORMAZIONALE (CENNI)

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine



- La descrizione formale è trasformata man mano in una meno astratta e più dettagliata, fino ad ottenere specifiche di basso livello eseguibili
- Le trasformazioni possono essere eseguite manualmente o supportate da appositi strumenti

MODELLO TRASFORMAZIONALE (CENNI)



- Il processo di trasformazione può trarre vantaggio da componenti riusabili

MODELLO TRASFORMAZIONALE

- **Problemi**
  - Necessità di competenze specifiche in linguaggi formali (es., matematici)
  - Difficile specificare formalmente alcune parti del sistema
  - Difficoltà del cliente nella convalida dei requisiti

- **Applicabilità**
  - Non adatti per sistemi di grandi dimensioni
  - Usati per parti critiche, ove la validità va dimostrata *by construction*

ESERCIZIO

Vincenzo Riccio
Ingegneria del Software 2024/2025
Università degli Studi di Udine

1. A quale dei modelli visti in questa lezione somiglia di più il tuo approccio nello sviluppo software?
2. Rappresenta lo sviluppo del tuo ultimo software come istanza di tale modello