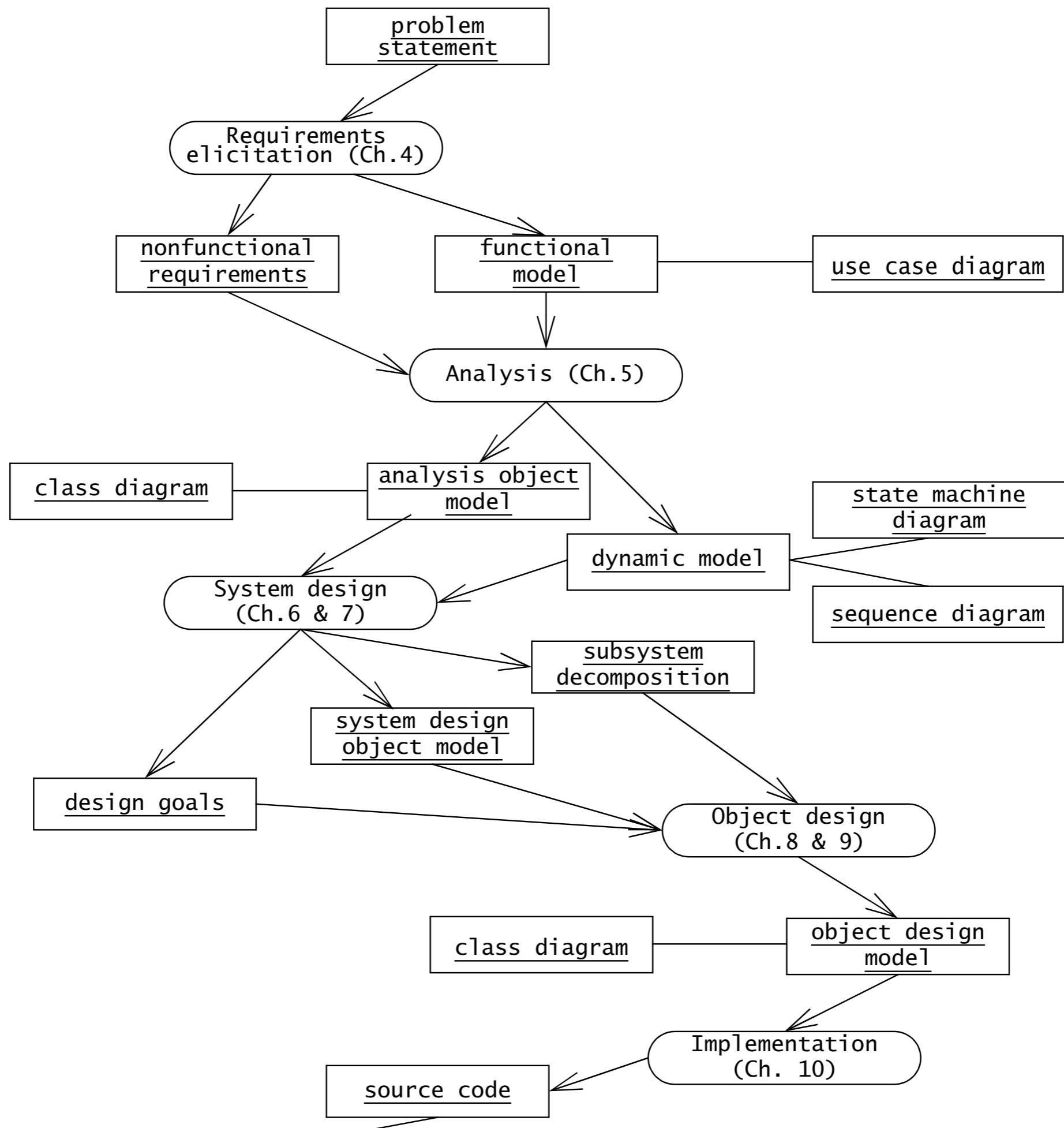


# Modellare i requisiti

Ingegneria del Software

*parte di queste slide sono state tratte dal libro di testo e dal materiale del Prof. Ciancarini, Dott. Favini e Di Iorio,  
Dott.ssa Zuppiroli*



# Requirements Analysis Document

## 1. Introduction

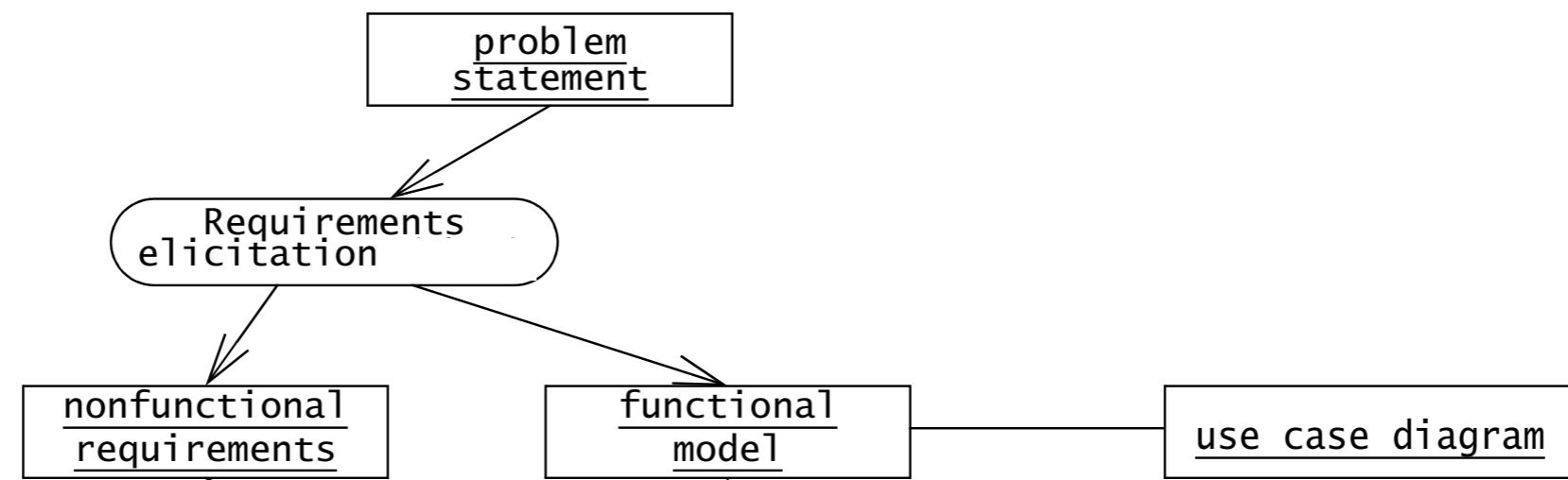
- 1.1 Purpose of the system
- 1.2 Scope of the system
- 1.3 Objectives and success criteria of the project
- 1.4 Definitions, acronyms, and abbreviations
- 1.5 References
- 1.6 Overview

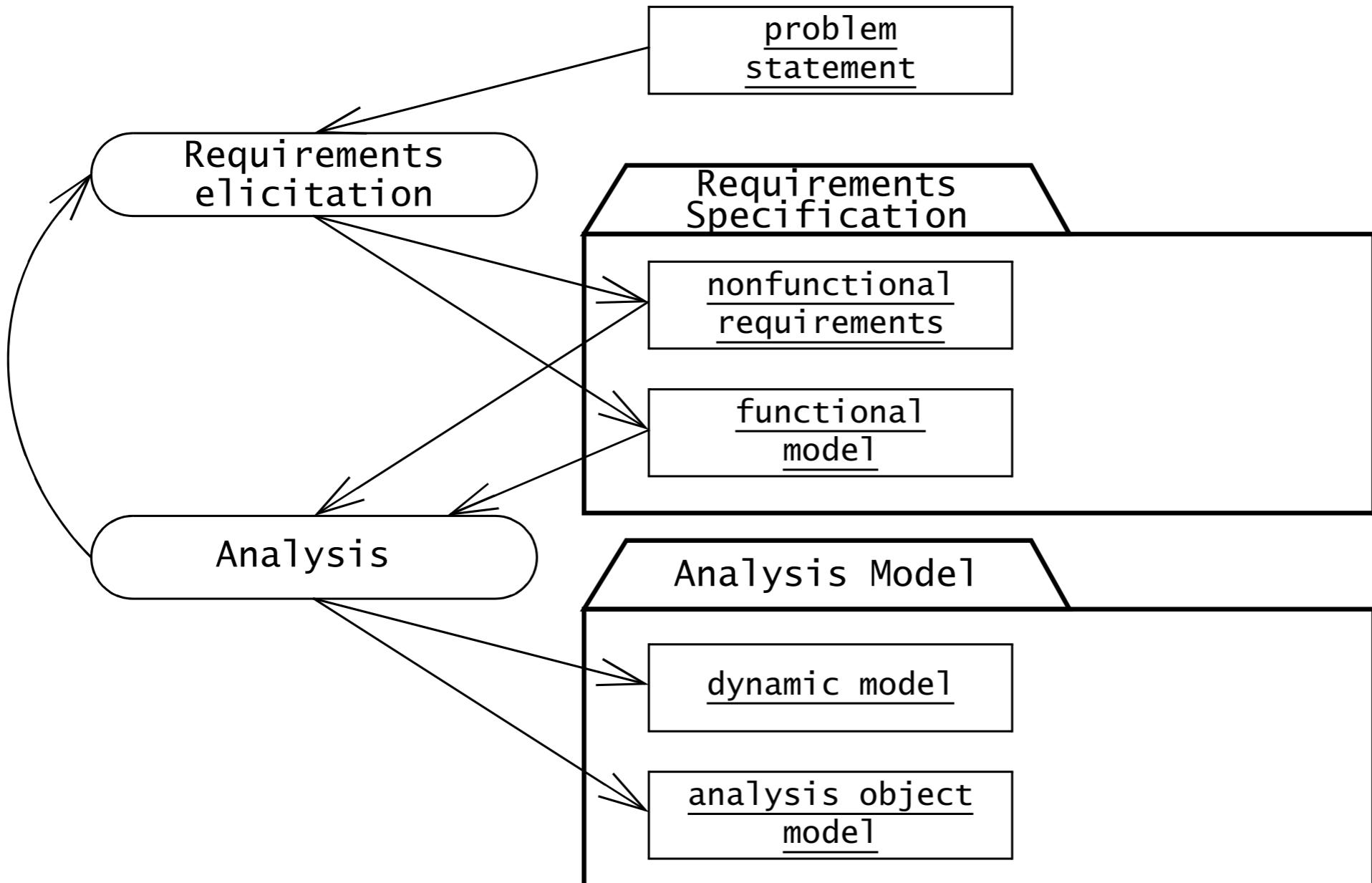
## 2. Current system

## 3. Proposed system

- 3.1 Overview
- 3.2 Functional requirements
- 3.3 Nonfunctional requirements
  - 3.3.1 Usability
  - 3.3.2 Reliability
  - 3.3.3 Performance
  - 3.3.4 Supportability
  - 3.3.5 Implementation
  - 3.3.6 Interface
  - 3.3.7 Packaging
  - 3.3.8 Legal
- 3.4 System models
  - 3.4.1 Scenarios
  - 3.4.2 Use case model
  - 3.4.3 *Object model*
  - 3.4.4 *Dynamic model*
  - 3.4.5 User interface—navigational paths and screen mock-ups

## 4. Glossary





---

**Figure 4-1** Products of requirements elicitation and analysis (UML activity diagram).

# Raccolta dei requisiti

- Include le seguenti attività:
  - Identificare gli attori
  - Identificare gli scenari
  - Identificare i casi d'uso
  - Raffinare i casi d'uso
  - Identificare le relazioni tra i casi d'uso
  - Identificare i requisiti non funzionali

# Raccolta dei requisiti

- Identificare gli attori. Durante questa attività, gli sviluppatori identificano i diversi tipi di utenti che il futuro sistema supporterà.
- Identificare gli scenari. Durante questa attività, gli sviluppatori osservano gli utenti e sviluppano una serie di scenari dettagliati per le funzionalità tipiche fornite dal sistema futuro. Gli scenari sono esempi concreti del sistema futuro in uso. Gli sviluppatori usano questi scenari per comunicare con l'utente e approfondire la loro comprensione del dominio dell'applicazione.
- Identificare i casi d'uso. Una volta che gli sviluppatori e gli utenti sono d'accordo su un insieme di scenari, gli sviluppatori derivano dagli scenari un insieme di casi d'uso che rappresentano completamente il sistema futuro. Mentre gli scenari sono esempi concreti che illustrano un singolo caso, i casi d'uso sono astrazioni che descrivono tutti i casi possibili. Quando si descrivono i casi d'uso, gli sviluppatori determinano lo scopo del sistema.

# Raccolta dei requisiti

- Raffinare i casi d'uso. Durante questa attività, gli sviluppatori si assicurano che la specifica dei requisiti sia completa dettagliando ogni caso d'uso e descrivendo il comportamento del sistema in presenza di errori e condizioni eccezionali.
- Identificare le relazioni tra i casi d'uso. Durante questa attività, gli sviluppatori identificano le dipendenze tra i casi d'uso. Inoltre consolidano il modello dei casi d'uso eliminando le funzionalità comuni. Questo assicura che la specifica dei requisiti sia coerente.
- Identificare i requisiti non funzionali. Durante questa attività, gli sviluppatori, gli utenti e i clienti si accordano su aspetti che sono visibili all'utente, ma non direttamente collegati alla funzionalità. Questi includono i vincoli sulle prestazioni del sistema, la sua documentazione, le risorse che consuma, la sua sicurezza e la sua qualità.

# Raccolta dei requisiti

- Esistono due tipi di requisiti:
  - Requisiti **funzionali** - descrivono il comportamento che il sistema dovrebbe avere quindi le interazioni tra il sistema ed il suo ambiente **indipendentemente** dalla sua implementazione
  - Requisiti **non funzionali** - descrivono proprietà o vincoli specifici del sistema, aspetti del sistema che non sono direttamente collegati con il funzionamento del sistema.

# Raccolta dei requisiti

- Tecniche per individuare i requisiti:
  - Interviste
  - Questionari
  - Workshop

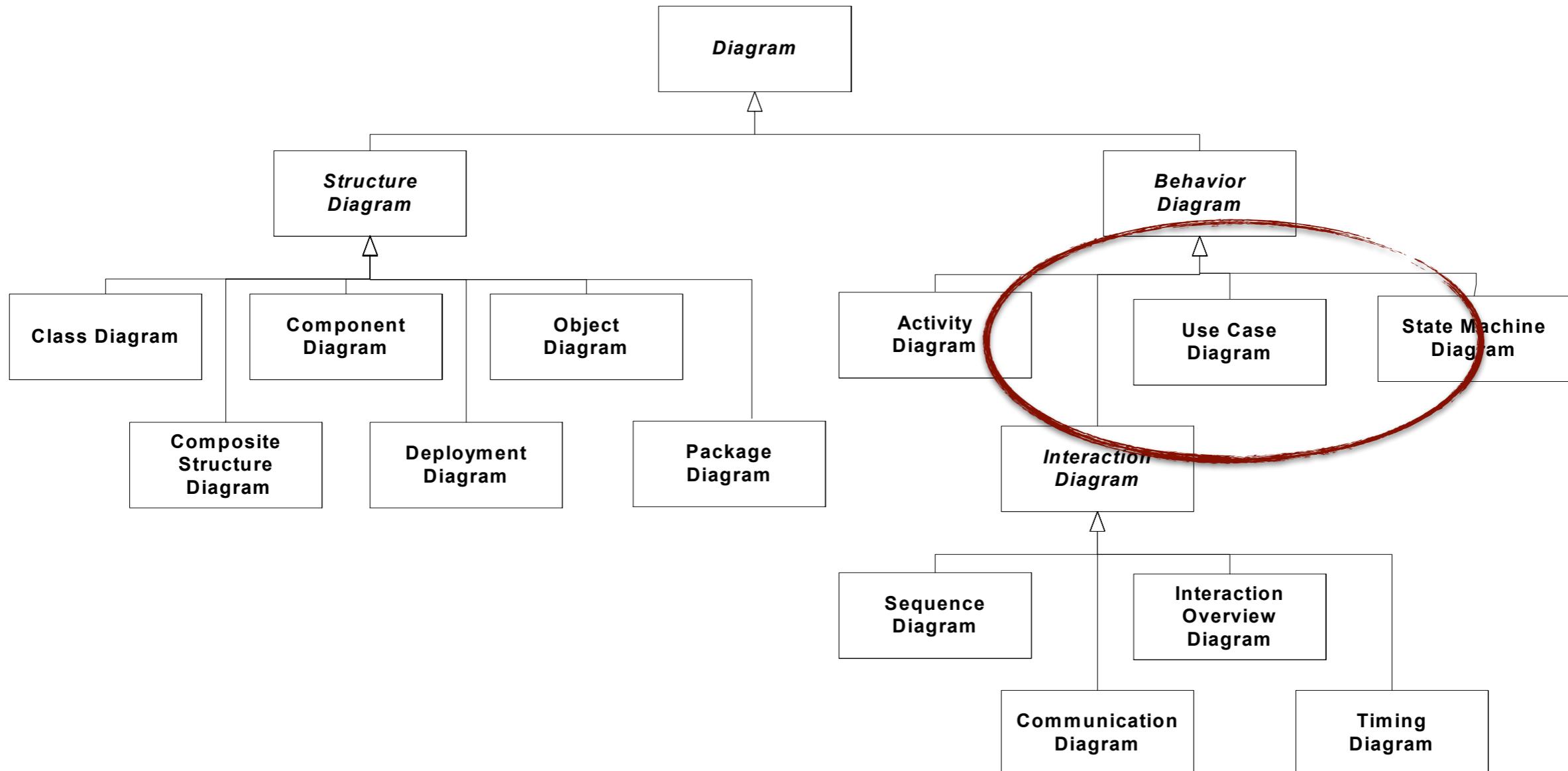
# Tassonomia dei diagrammi UML

- UML 2 definisce 13 diagrammi divisi in due categorie
  - structure diagrams: come è fatto il sistema
  - behaviour diagrams: come funziona (o si comporta) il sistema

<b>Structure</b>	<b>Behavior</b>
<p>Class diagram</p> <p>Object diagram</p> <p>Package diagram*</p> <p>Composite Structure diagram*</p> <p>Component diagram</p> <p>Deployment diagram</p>	<p>Use Case diagram</p> <p>Activity diagram</p> <p>State Machine diagram</p> <p>Sequence diagram</p> <p>Communication diagram</p> <p>Interaction Overview diagram*</p> <p>Timing diagram*</p>

\* : non esiste in UML 1.x

# Tassonomia dei diagrammi UML 2



# Diagramma dei casi d'uso

- E' un diagramma che rappresenta un comportamento desiderato o offerto
- L'oggetto esaminato è il sistema o una sua parte
- Individua:
  - chi o cosa ha a che fare con il sistema
  - che cosa l'attore può fare

# Modellare i casi d'uso

- La modellazione dei casi d'uso è una tecnica di ingegneria dei requisiti che segue la seguente procedura:
  - Individuare il subject
  - Individuare gli attori
  - Individuare i casi d'uso
  - Il subject definisce cosa fa parte del sistema e cosa è esterno ad esso

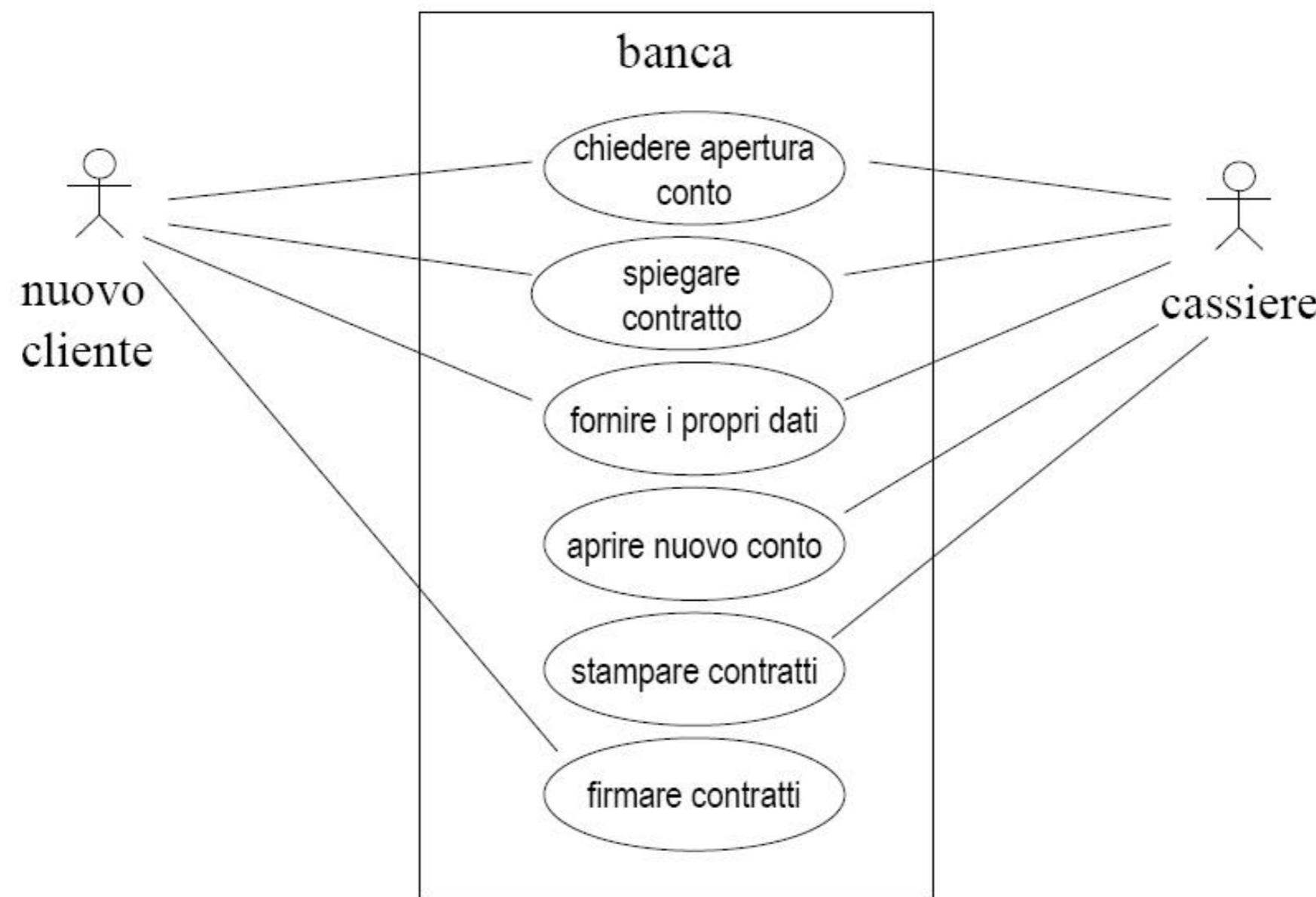
# Modellare i casi d'uso

- Gli attori rappresentano i ruoli delle entità esterne al sistema che interagiscono direttamente con esso
- Si possono individuare gli attori ragionando su chi o cosa utilizza o si interfaccia con il sistema
- Il tempo è spesso un attore

# Modellare i casi d'uso

- I casi d'uso rappresentano le funzioni che il sistema persegue per conto di qualche attore
- È possibile individuare i casi d'uso considerando quali funzioni il sistema offre agli attori
- I casi d'uso vengono sempre attivati da un attore
- I casi d'uso vengono sempre scritti dal punto di vista degli attori

# Esempio: conto in banca



# Modellare i casi d'uso

- La specifica di un caso d'uso contiene:
  - Il nome del caso d'uso
  - Un identificatore univoco
  - Una breve descrizione - l'obiettivo del caso d'uso
  - Attori
    - Attori primari: avviano il caso d'uso
    - Attori secondari: interagiscono con il caso d'uso dopo che è stato avviato
  - Le precondizioni: vincoli iniziali sul sistema che impattano sulla esecuzione del caso d'uso
  - Sequenza degli eventi principali: sequenza dichiarativa ordinata temporalmente
  - Postcondizioni: vincoli finali sul sistema dovuti all'esecuzione del caso d'uso
  - Sequenza degli eventi alternativa: un elenco di alternative alla sequenza principale

# Modellare i casi d'uso

- Ricordiamo:
  - Un caso d'uso modella il **cosa** ci si aspetta da un sistema
  - Nasconde **come** il sistema lo implementa
  - E' una sequenza di azioni che riproduce come l'attore interagisce con il sistema - il punto di vista è quello dell'attore
  - Si usa per descrivere i requisiti (fase iniziale), convalidare l'architettura e verificare il sistema



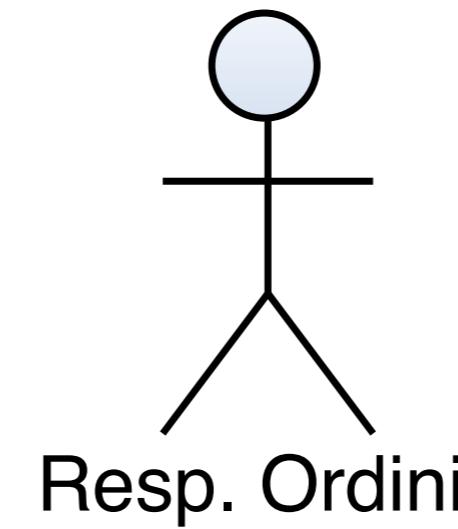
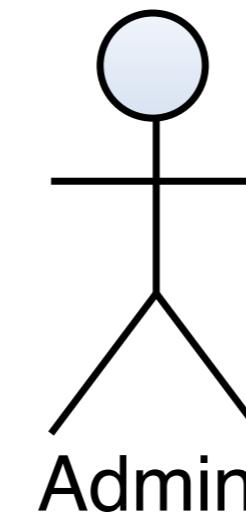
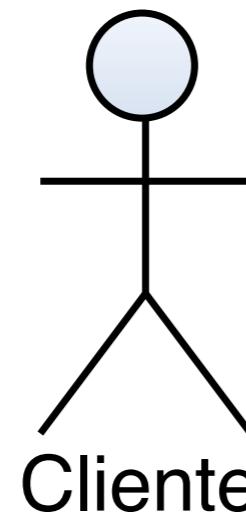
# Estrarre i requisiti

- Chiedersi:
  - chi interagisce con il sistema?
  - quindi quali attori?
  - cosa fanno gli attori quando usano il software?
  - esempio: il correntista della banca inserisce il PIN, l'utente seleziona una destinazione ...

# Attori

- Un attore specifica un ruolo assunto da un utente o altra entità che interagisce col sistema nell'ambito di un'unità di funzionamento (caso d'uso).
- Un attore è esterno al sistema.  
Non è necessariamente umano: oggetto fisico, agente
- software, condizioni ambientali, etc.
- Gli attori eseguono casi d'uso: prima si cercano gli attori, poi i loro casi d'uso!

# Notazione UML per gli attori



# Come individuare gli attori

- Domande per identificare gli attori
  - Quali gruppi di utenti sono supportati dal sistema per svolgere il loro lavoro?
  - Quali gruppi di utenti eseguono le funzioni principali del sistema?
  - Quali gruppi di utenti svolgono funzioni secondarie, come la manutenzione e l'amministrazione?
  - Con quale sistema hardware o software esterno interagirà il sistema?
  - Ci sono funzioni attivate periodicamente?
  - Chi o cosa ottiene (o fornisce) informazioni dal sistema?



# Come identificare gli scenari

- Nella raccolta dei requisiti, gli sviluppatori e gli utenti scrivono e perfezionano una serie di scenari al fine di ottenere una visione condivisa di ciò che il sistema dovrebbe fare.

# Come identificare gli scenari

- Domande per identificare gli scenari
- Quali sono i compiti che l'attore vuole che il sistema esegua?
- A quali informazioni accede l'attore? Chi crea quei dati? Può essere modificato o rimosso? Da chi?
- Di quali cambiamenti esterni l'attore deve informare il sistema? Quanto spesso? Quando?
- Di quali eventi il sistema deve informare l'attore? Con quale latenza?

# Identificare i casi d'uso

- Uno **scenario** è un'istanza di un **caso d'uso**
- Un **caso d'uso** specifica tutti i possibili scenari per una data funzionalità.
- Un caso d'uso è iniziato da un attore.
- Dopo il suo inizio, un caso d'uso può interagire anche con altri attori.
- Un caso d'uso rappresenta un flusso completo di eventi attraverso il sistema, nel senso che descrive una serie di interazioni correlate che risultano dal suo inizio.

# Come individuare i casi d'uso

- Ciascun attore che funzioni si aspetta?
- Il sistema gestisce (archivia/fornisce) informazioni?
- Se sì quali sono gli attori che provocano questo comportamento?
- Alcuni attori vengono informati quando il sistema cambia stato?
- Gli attori devono informare il sistema di cambiamenti improvvisi?
- Alcuni eventi esterni producono effetti sul sistema?
- Quali casi d'uso manutengono il sistema?
- I requisiti funzionali sono tutti coperti dai casi d'uso?

# Modellare i casi d'uso

- Un caso d'uso non deve essere scomposto in sequenza degli eventi alternative a meno che questi non aggiungano valore al modello
- La modellazione dei casi d'uso è particolarmente indicata per sistemi che:
  - Sono dominati da requisiti funzionali
  - Hanno molti tipi utente
  - Hanno molte interfacce con altri sistemi
- La modellazione dei casi d'uso è poco indicata per sistemi che:
  - Sono dominati da requisiti non funzionali
  - Hanno pochi utenti
  - Hanno poche interfacce con altri sistemi

# Come descrivere un caso d'uso

- Descrivere in modo generico e sequenziale il flusso di eventi di un caso d'uso
- Descrivere la precondizione (stato iniziale del sistema) Elencare la sequenza di passi
- Descrivere le interazioni con gli attori e i messaggi scambiati
- Aggiungere eventuali punti di estensione
- Descrivere in modo chiaro, preciso e breve



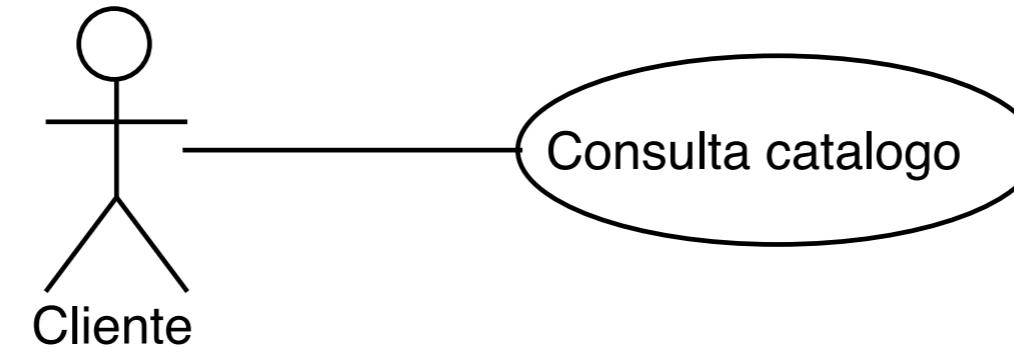
# Elemento notazione per il caso d'uso

Consulta catalogo

# Modellare i casi d'uso

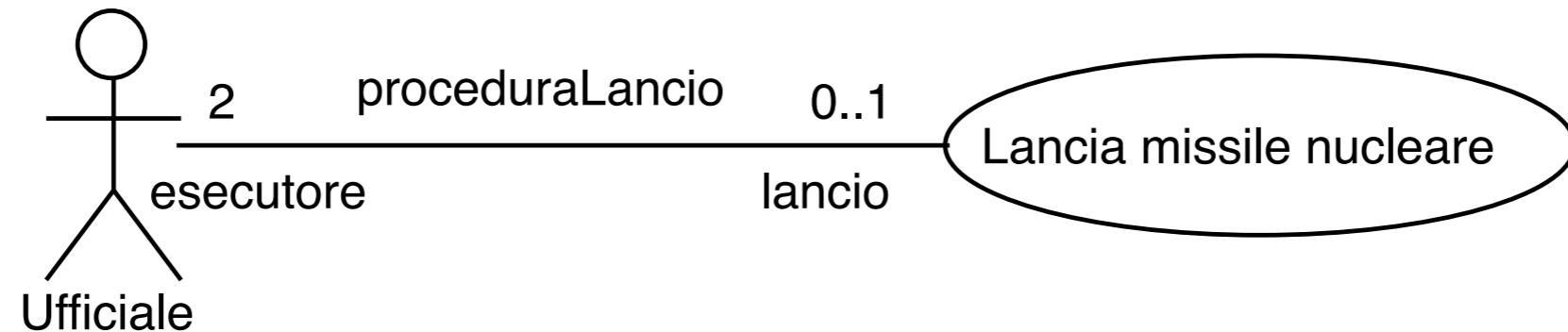
- Relazioni tra casi d'uso:
  - Generalizzazione tra attori
  - Generalizzazione tra casi d'uso
  - Include
  - Extend
- Suggerimenti:
  - Mantenere di casi d'uso brevi e semplici
  - Concentrarsi sul cosa e non sul come
  - Evitare la scomposizione funzionale

# Associazione tra attore e caso d'uso



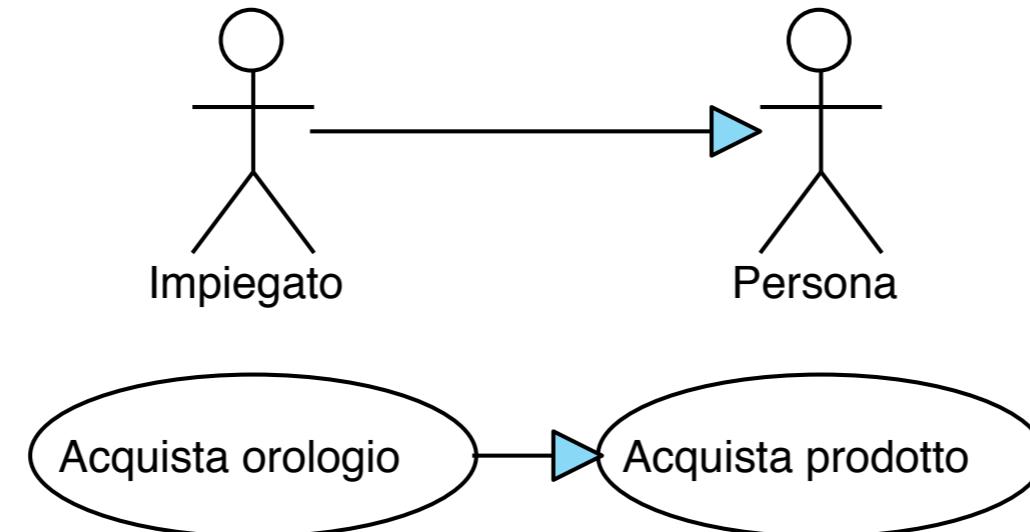
- Collega gli attori ai casi d'uso.
- Un attore si può associare solo a casi d'uso.
- Un caso d'uso non dovrebbe essere associato ad altri casi d'uso riguardanti lo stesso argomento
-

# Associazione tra attore e caso d'uso



- Alcune caratteristiche opzionali comuni a tutte le associazioni in UML:
  - nome
  - molteplicità
  - ruoli

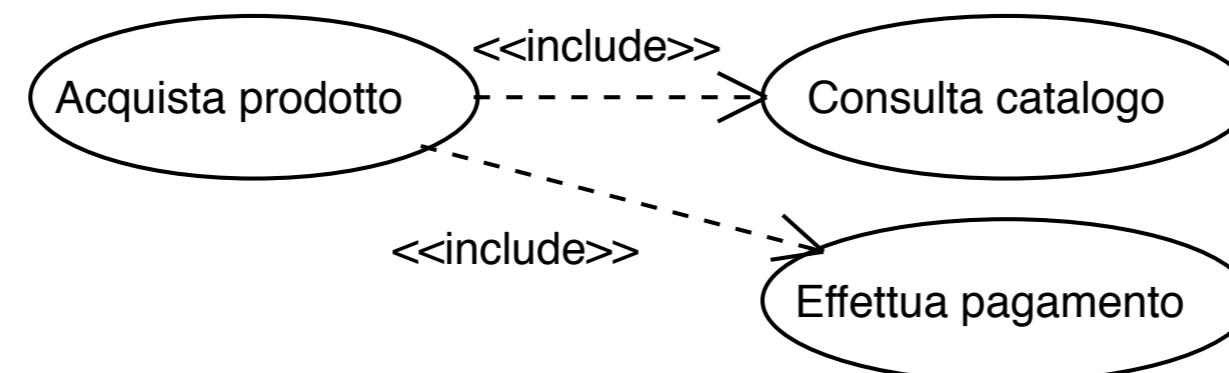
# Generalizzazione



- Collega un attore o caso d'uso ad un altro più generale.

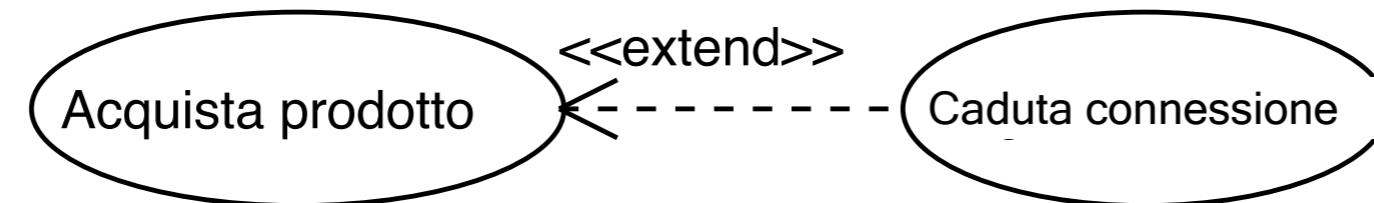
-

# Include

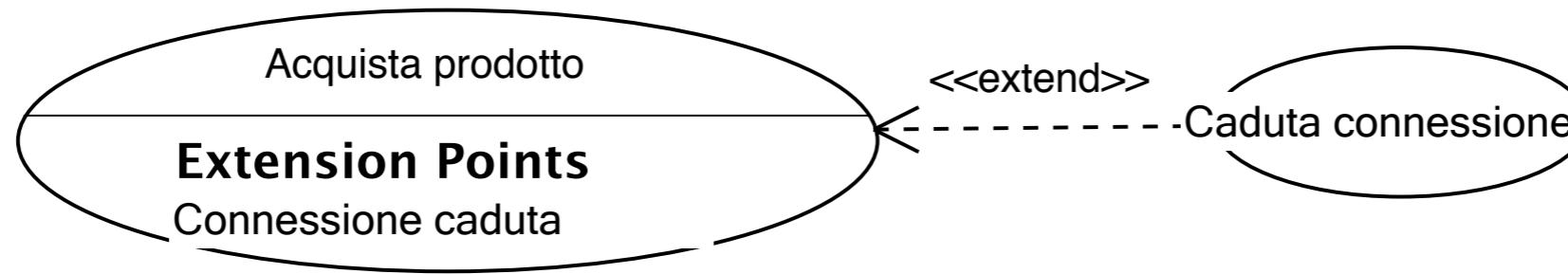


- Una **dipendenza** tra casi d'uso; il caso incluso fa parte del comportamento di quello che lo include.
- L'inclusione **non** è **opzionale** ed avviene in ogni istanza del caso d'uso.
- La corretta esecuzione del caso d'uso che include dipende da quella del caso d'uso incluso.
- Non si possono formare cicli di include.
- Usato per **riutilizzare parti comuni** a più casi d'uso.

# Extend

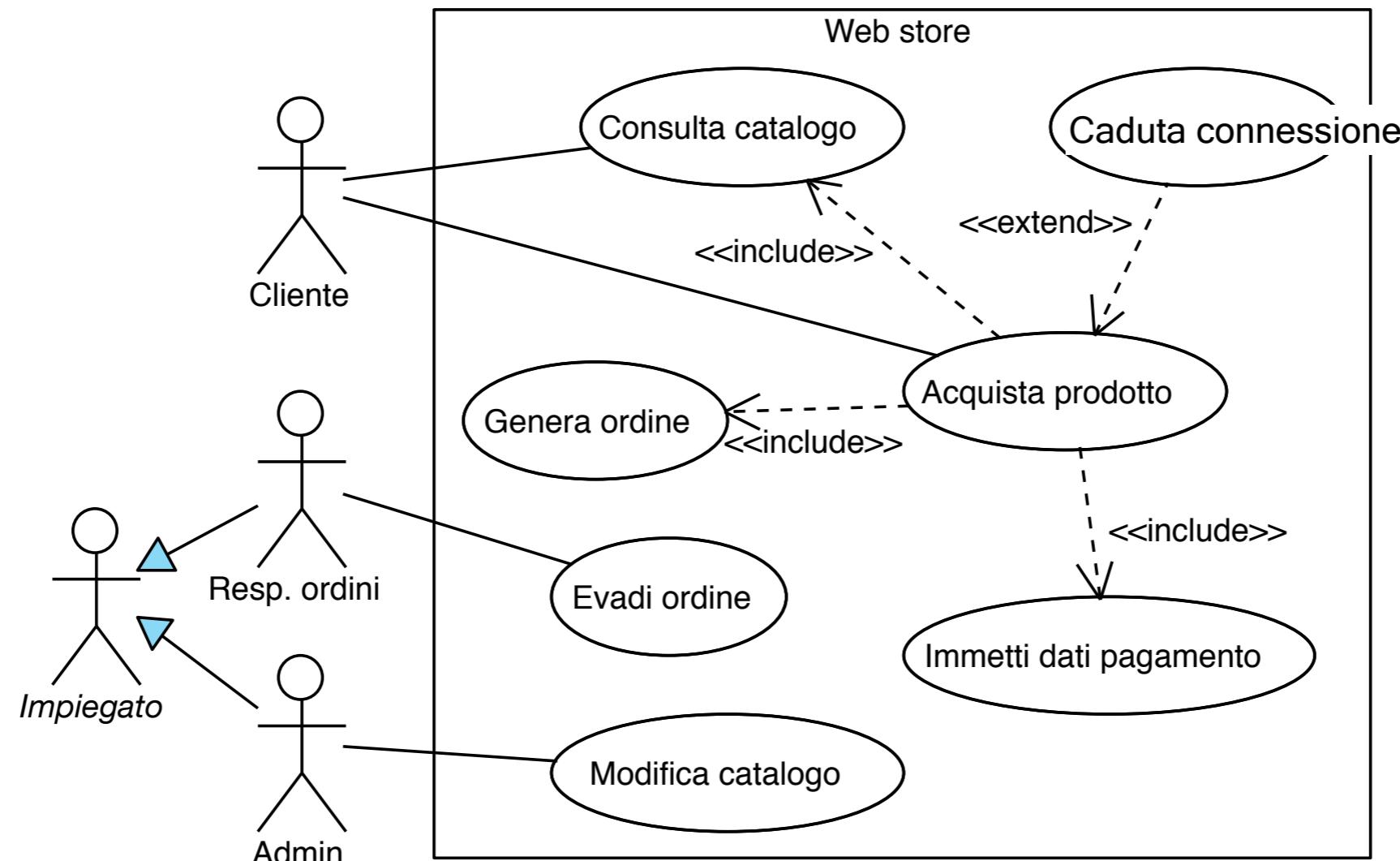


- Una **dipendenza** tra casi d'uso (notare il verso della freccia).
- Il caso d'uso che estende (client) specifica un incremento di comportamento a quello esteso (supplier).
- Si tratta di comportamento **supplementare** ed **opzionale** che gestisce **casi particolari** o non standard.
- Diverso da una generalizzazione tra casi d'uso:
  - in una generalizzazione, entrambi i casi d'uso sono ugualmente significativi
  - in un extend, il client non ha necessariamente senso se preso da solo



- Possono essere specificati gli extension points - cioè i punti e/o le condizioni in cui il comportamento viene esteso.

# Esempio di diagramma

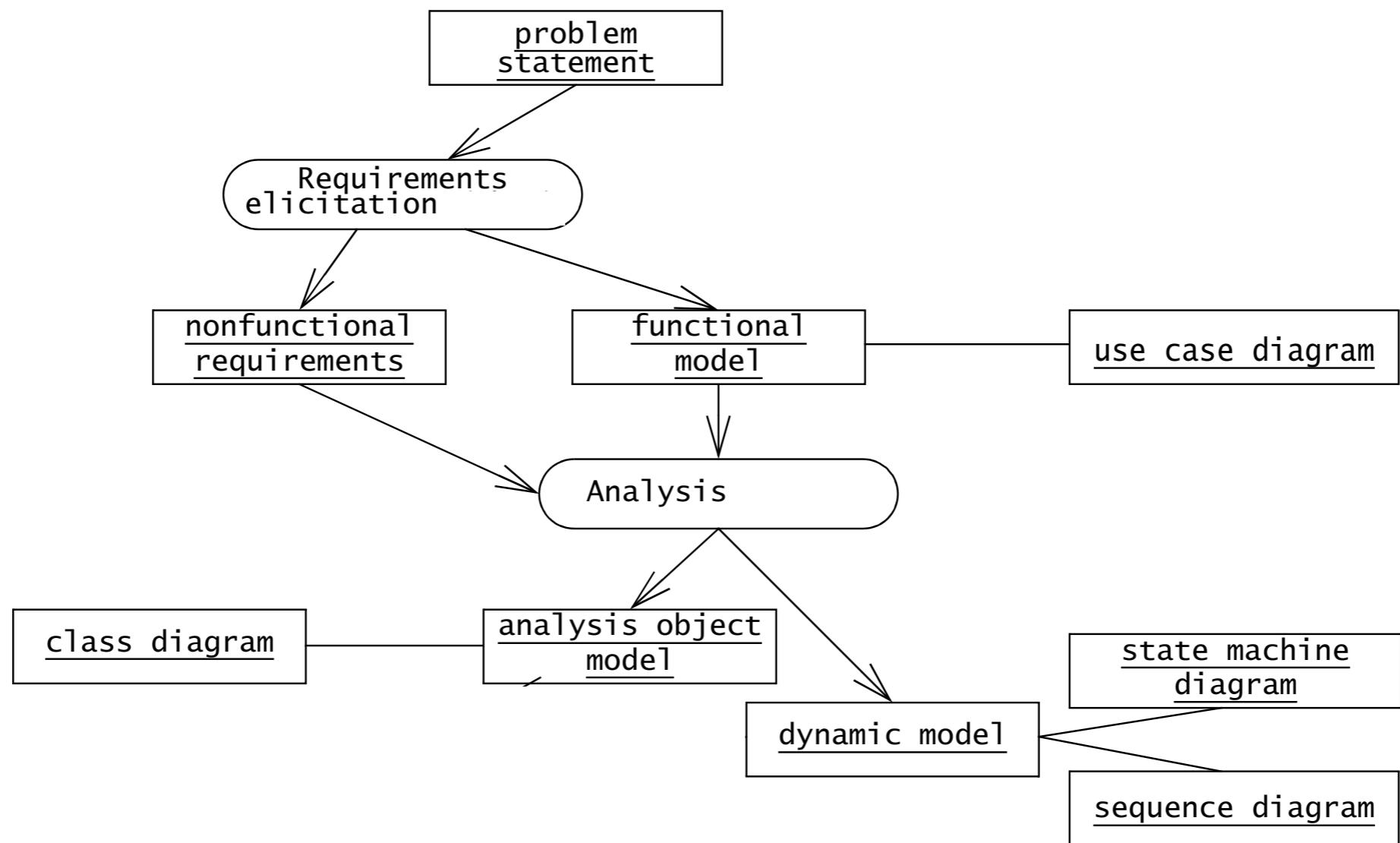


# Modellare i casi d'uso

- Linee guida:
- Generalizzazione tra attori: da usare solo se semplifica il modello
- Generalizzazione tra casi d'uso: meglio non usarla oppure usarla solo con casi d'uso generalizzati astratti
- Relazione include: da usare solo se semplifica il modello
- Un eccesso di include può trasformare il modello in una scomposizione funzionale del sistema

# Modellare i requisiti

- Sono sufficienti i diagrammi dei casi d'uso?
  - NO
  - Ogni caso d'uso va completato con la descrizione del flusso degli eventi che realizza la funzionalità.



# Analisi - un overview

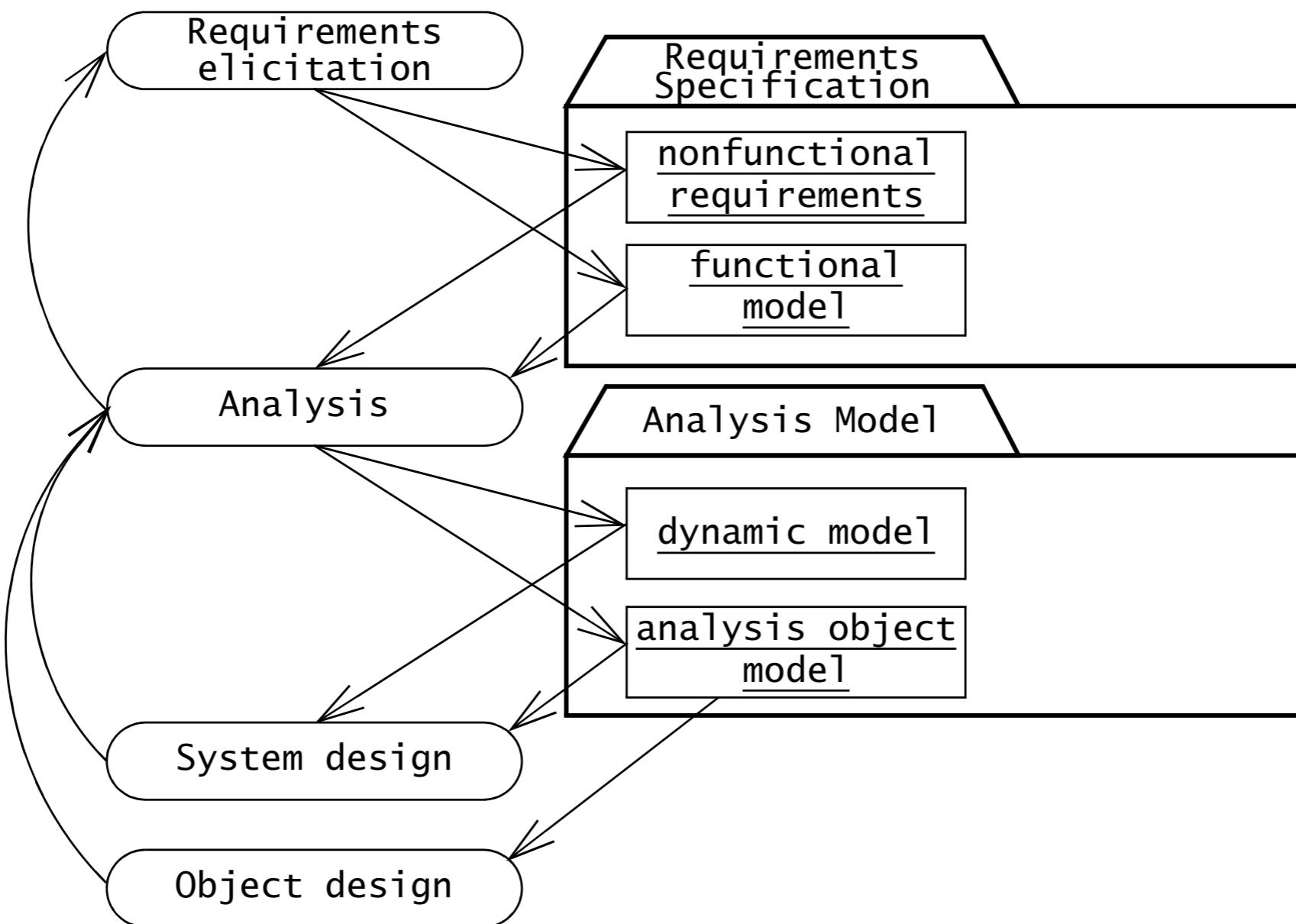
- L'analisi si focalizza sulla produzione di un modello del sistema:
  - Il modello di analisi, o *analysis model*
  - Corretto
  - Completo
  - Consistente
  - Verificabile
- L'analisi è diversa dalla identificazione dei requisiti
- Durante l'analisi lo sviluppatore si occupa di strutturare e formalizzare i requisiti ottenuti dall'utente

# Analisi - un overview

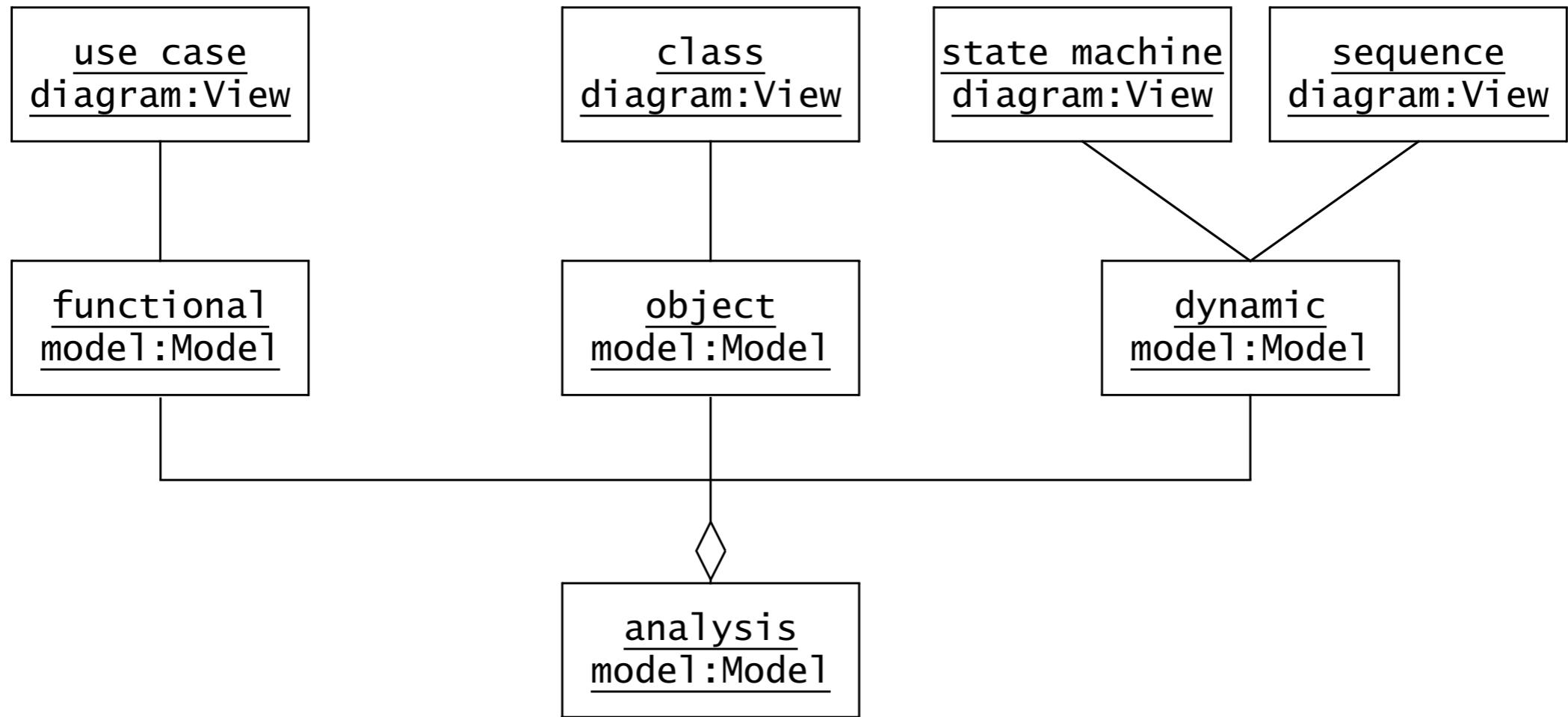
- Il **modello di analisi** composto da tre differenti modelli:
  - Il **modello funzionale** - casi d'uso e scenari
  - Il **modello degli oggetti** di analisi- rappresentato da un diagramma delle classi degli oggetti
  - Il **modello dinamico** - rappresentato da un diagramma di sequenza (*sequence diagram*) e da una macchina a stati (*state machine diagram*)

## An Overview of Analysis

---



**Figure 5-2** Products of requirements elicitation and analysis (UML activity diagram).



**Figure 5-3** The analysis model is composed of the functional model, the object model, and the dynamic model. In UML, the functional model is represented with use case diagrams, the object model with class diagrams, and the dynamic model with state machine and sequence diagrams.

