

# ITC INFORMATION AND COMMUNICATIONS TECHNOLOGY ACADEMY

---

MODULO: Progettazione  
UNITÀ: Progettazione.1

Prof. Toni Mancini  
Dipartimento di Informatica  
Sapienza Università di Roma



Introduzione e cenni di  
ingegneria del software

# ITC INFORMATION AND COMMUNICATIONS TECHNOLOGY ACADEMY

---

MODULO: Progettazione

UNITÀ: Progettazione.1

Prof. Toni Mancini

Dipartimento di Informatica  
Sapienza Università di Roma



I.1

Introduzione e cenni di ingegneria  
del software

Obiettivi

# Cosa impareremo in questa unità?

- Impareremo a progettare applicazioni software reali e di dimensioni non banali
  - impossibile definire direttamente il codice
  - gran parte del tempo deve essere speso nella comprensione dei dati di interesse e nelle loro interrelazioni
  - non esiste una soluzione unica, bisogna fare delle scelte ragionate
  - non esiste una ricetta che “basta applicare”
  - il metodo deve essere molto generale, e va applicato e adattato intelligentemente al problema in esame

Di conseguenza, impareremo a

- ragionare logicamente
- considerare le conseguenze delle nostre scelte
- valutare le alternative
- scomporre i problemi complessi in sottoproblemi

# Esempio 1

- Ci si richiede di progettare una applicazione che permetta di mantenere informazioni su un insieme di contatti (telefonici e email)
- Requisiti:
  - Per ogni contatto è necessario mantenere:
    - nome e cognome
    - numeri di telefono di casa, ufficio e mobile
    - indirizzo email
  - I contatti possono appartenere a gruppi
  - L'applicazione deve permettere all'utente di:
    - aggiungere un nuovo contatto
    - modificare i dati di un contatto
    - cancellare un contatto
    - assegnare/rimuovere contatti a/da un gruppo
    - ricercare i contatti per nome e/o cognome.

Non c'è bisogno di progettare molto:  
probabilmente siamo già in grado di definire un programma Python per l'applicazione

# Esempio 2

- Si vuole sviluppare un sistema che permetta ad una banca di gestire i conti correnti dei clienti, i loro investimenti, oltre che la propria rete di promotori finanziari
- Il sistema deve tenere traccia di tutti gli acquisti e vendite di azioni, obbligazioni, etc. effettuati dai clienti, e deve poter calcolare in tempo reale la valorizzazione corrente del loro portafoglio
- Inoltre, l'applicazione deve assistere i promotori finanziari nella scelta degli strumenti finanziari più adeguati da proporre ai clienti, e deve permettere ai responsabili di agenzia di controllare la professionalità dei promotori

Impossibile scrivere direttamente lo schema relazionale e il programma per l'applicazione!

## **Progetto software complesso:**

- pool di Ingegneri del SW, progettisti, programmati: 1.5 anni (spannometricamente...)
  - tempo per capire il problema: 6 mesi (33%)
  - tempo per la progettazione: 9 mesi (50%)
  - tempo per la realizzazione: 3 mesi (solo il 17%)
  - più: tempo per test&verifica, etc. etc.

**Oggetto di questo corso: saper realizzare questo genere di progetti come applicazioni software da dispiegare nel cloud**

# ITC INFORMATION AND COMMUNICATIONS TECHNOLOGY ACADEMY

---

MODULO: Progettazione  
UNITÀ: Progettazione.1

Prof. Toni Mancini  
Dipartimento di Informatica  
Sapienza Università di Roma



I.2

Introduzione e cenni di ingegneria del software  
**Contesto  
organizzativo**

# Contesto organizzativo

Attori principali coinvolti nel ciclo di progettazione, sviluppo ed esercizio del software:

- Committente
- Esperti del dominio
- Analisti
- Progettisti
- Programmatori
- Utenti finali
- Manutentori

# Esempio

- Il Comune di XYZ intende automatizzare la gestione delle informazioni relative alle contravvenzioni elevate sul suo territorio
- In particolare, intende dotare ogni vigile di una app per smartphone che gli consenta di comunicare al sistema informatico il veicolo a cui è stata comminata la contravvenzione, il luogo in cui è stata elevata e la natura dell'infrazione
- Il sistema informatico provvederà a notificare, tramite posta ordinaria, la contravvenzione al cittadino interessato.
- Il Comune bandisce opportune gare per la progettazione, realizzazione e manutenzione del sistema, che vengono vinte, rispettivamente, dalle ditte ITsolutions s.p.a., Develop s.r.l. e Ops s.r.l.

Quali sono gli attori coinvolti in questa applicazione software?

- Committente?
- Analisti?
- Progettisti?
- Programmatori?
- Utenti finali?
- Manutentori?
- Esperti del dominio?

# Esempio

- Il Comune di XYZ intende automatizzare la gestione delle informazioni relative alle contravvenzioni elevate sul suo territorio
- In particolare, intende dotare ogni vigile di una app per smartphone che gli consenta di comunicare al sistema informatico il veicolo a cui è stata comminata la contravvenzione, il luogo in cui è stata elevata e la natura dell'infrazione
- Il sistema informatico provvederà a notificare, tramite posta ordinaria, la contravvenzione al cittadino interessato.
- Il Comune bandisce opportune gare per la progettazione, realizzazione e manutenzione del sistema, che vengono vinte, rispettivamente, dalle ditte ITSolutions s.p.a., Develop s.r.l. e Ops s.r.l.

Quali sono gli attori coinvolti in questa applicazione software?

- Committente?
- Analisti?
- Progettisti?
- Programmatori?
- Utenti finali?
- Manutentori?
- Esperti del dominio?



- Comune di XYZ
- Personale ITSolutions s.p.a.
- Personale ITSolutions s.p.a.
- Personale Develop s.r.l.
- Polizia locale XYZ
- Personale Ops s.r.l.
- funzionario del Comune, o altro professionista designato, esperto del Codice della Strada

# ITC INFORMATION AND COMMUNICATIONS TECHNOLOGY ACADEMY

---

MODULO: Progettazione  
UNITÀ: Progettazione.1

Prof. Toni Mancini  
Dipartimento di Informatica  
Sapienza Università di Roma



I.3

Introduzione e cenni di ingegneria del software  
**Ciclo di vita del  
software**

# Ciclo di vita del software: macro-fasi principali

## 1. Studio di fattibilità

- comprendere i requisiti di alto livello
- valutare costi e benefici
- pianificare le attività e le risorse del progetto
- individuare l'ambiente di programmazione (hardware/software)

## 2. Raccolta dei requisiti

- Raccolta dei requisiti presso i diversi attori
- Stesura e sintesi iniziali
- Raffinamento dei requisiti

## 3. Analisi concettuale dei requisiti

- Obiettivo: produrre lo **schema concettuale** dell'applicazione, che definisca in dettaglio **cosa** l'applicazione dovrà realizzare, indipendentemente dal **come**
- Lo schema concettuale:
  - Modella i dati di interesse, le loro articolazioni, interrelazioni ed evoluzioni possibili
  - Specifica i servizi computazionali che l'applicazione dovrà offrire ai diversi utenti
  - Lo schema concettuale è un modello logico-matematico dell'applicazione, e sarà la base da cui partire per le successive attività di progettazione

4. ...

## 4. Progetto (design) dell'applicazione

- Obiettivo: specificare **come** l'applicazione dovrà realizzare le sue funzioni (ora che il **cosa** è stato esplicitato in modo non ambiguo e in tutti i dettagli)
  - Scegliere il mix tecnologico ottimale per l'applicazione
  - Definire l'architettura
  - Definire le strutture di rappresentazione dei dati (in memoria centrale e di massa)

## 5. Realizzazione (implementazione) dell'applicazione

- Scrivere il codice
- Scrivere la documentazione

## 6. Integrazione dei componenti e verifica dell'applicazione

- Le diverse componenti dell'applicazione, sviluppate separatamente, vengono integrate
- Si valuta se l'applicazione svolge correttamente, completamente ed efficientemente i suoi compiti

## 7. Messa in esercizio dell'applicazione

- L'applicazione viene messa in esercizio ed inizia a funzionare

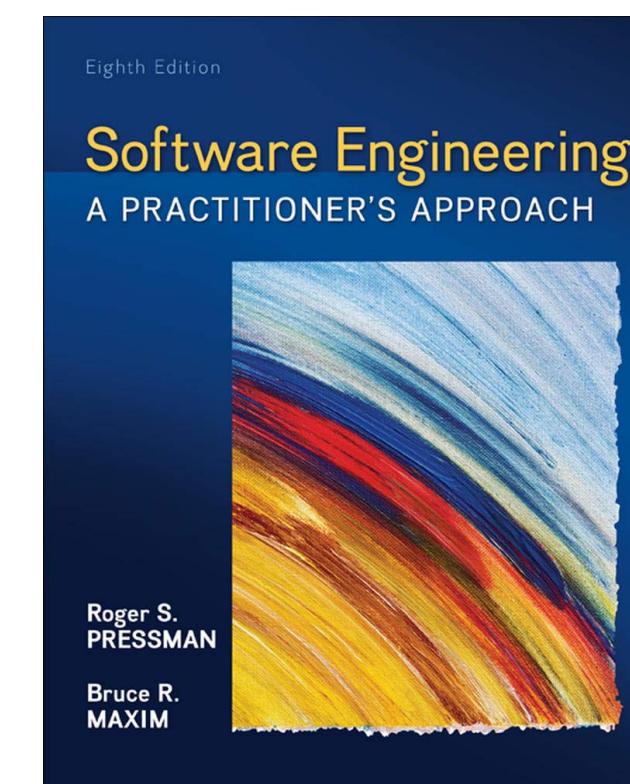
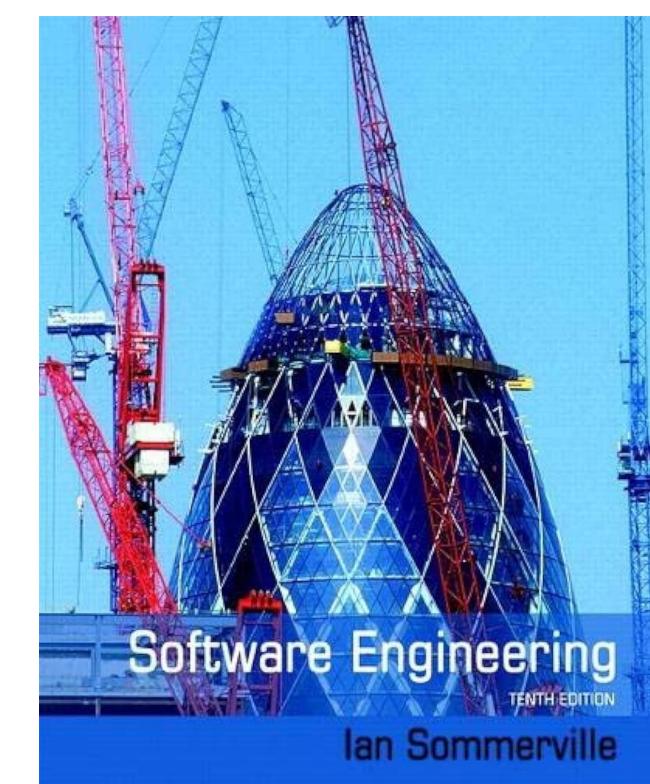
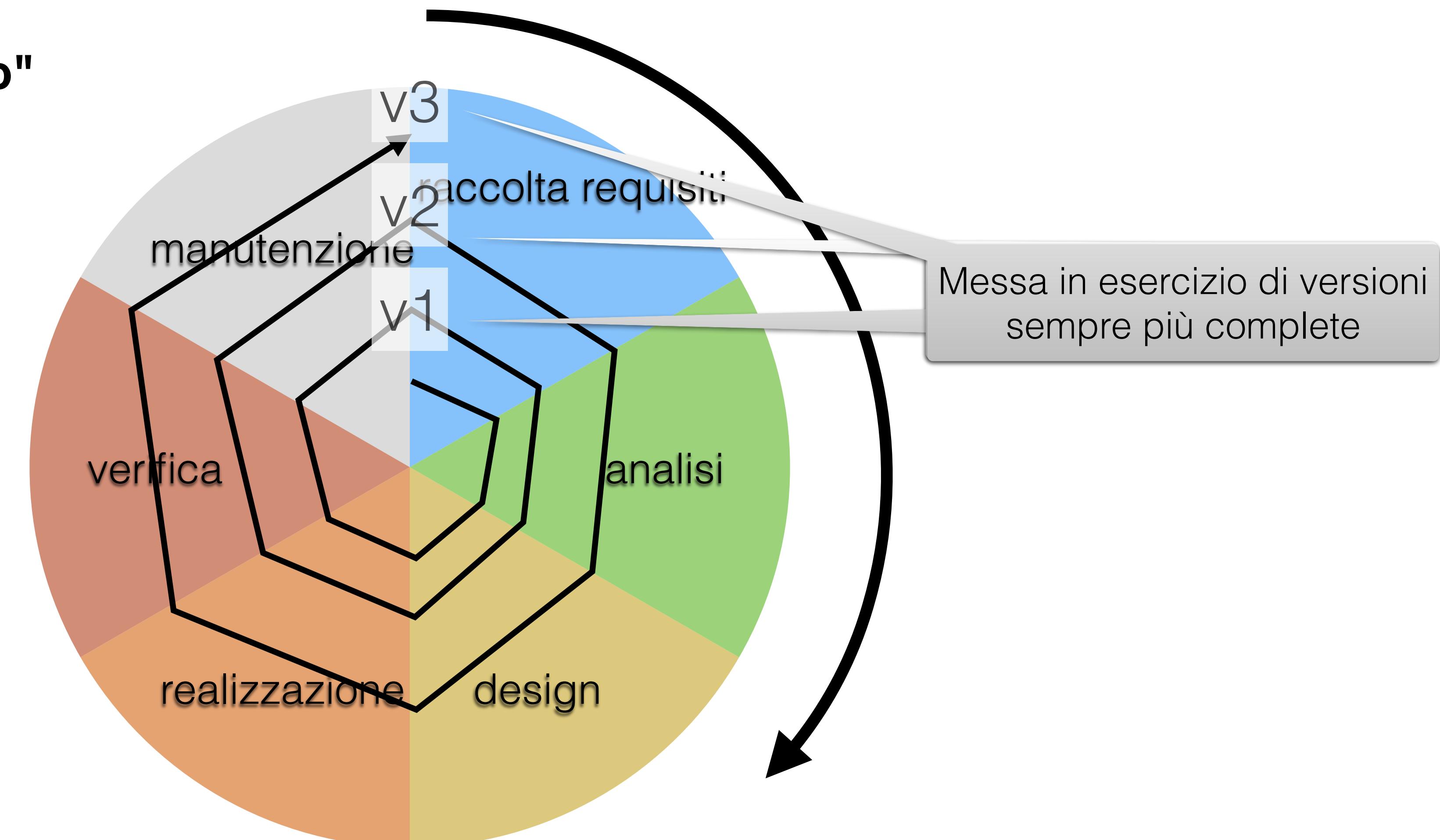
## 8. Manutenzione dell'applicazione

- L'applicazione viene monitorata durante l'esercizio
- Correzioni ed aggiornamenti vengono prodotti ove e quando necessario

Al termine di ogni fase, se necessario, si può **tornare indietro**

# Modelli di ciclo di vita del software

- **Modello “a cascata” (“waterfall model”)**
  - Ogni attività inizia quando termina la precedente
  - Ha un interesse esclusivamente didattico!
- **Modello “a spirale” o “iterativo”**



I. Sommerville:  
Software Engineering

R. Presman, B. Maxim:  
Software Engineering

# ITC INFORMATION AND COMMUNICATIONS TECHNOLOGY ACADEMY

---

MODULO: Progettazione  
UNITÀ: Progettazione.1

Prof. Toni Mancini  
Dipartimento di Informatica  
Sapienza Università di Roma

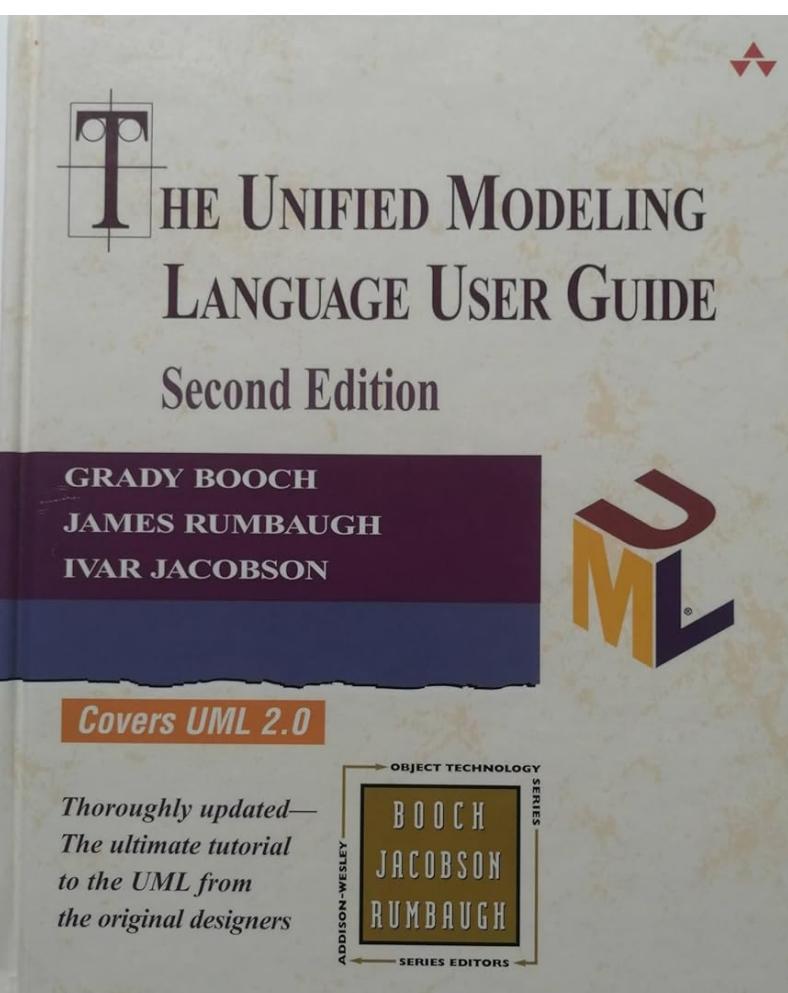


I.4

Introduzione e cenni di ingegneria del software  
**Unified Modeling  
Language**

# Il linguaggio UML

- UML sta per Unified Modeling Language
- Il progetto UML nasce nel 1994 come unificazione di:
  - Booch
  - Rumbaugh: OMT (Object Modeling Technique)
  - Jacobson: OOSE (Object-Oriented Software Engineering)
- Storia
  - 1995: Versione 0.8 (Booch, Rumbaugh)
  - 1996: Versione 0.9 (Booch, Rumbaugh, Jacobson)
  - Versione 1.0 (BRJ + Digital, IBM, HP, ...)
  - 1999/2001: Versione 1.3/1.4, descritta nei testi
  - 2003: Versione 1.5 + draft versione 2.0
  - 2005: Versione 2.0
  - ...
    - ...
  - 2017: Versione 2.5.1
- Riferimento:
  - G. Booch, J. Rumbaugh, I. Jacobson, “The Unified Modeling Language user guide”, 2nd Edition, Addison Wesley, 2005.
  - <http://www.uml.org>



# Diagrammi UML

- UML definisce 14 tipi di diagrammi per modellare l'applicazione sotto prospettive diverse.
- I principali tipi di diagramma sono:
  - Diagrammi strutturali:
    - Diagramma delle classi e degli oggetti (class and object diagram)
  - Diagrammi comportamentali:
    - Diagramma degli use case (use case diagram),
    - Diagramma degli stati e delle transizioni (state/transition diagram),
    - Interaction (Sequence e Collaboration diagram),
    - Activity diagram
  - Diagrammi architetturali:
    - Component diagram
    - Deployment diagram

# Uso di UML nella nostra metodologia

- La metodologia che illustriamo in questo modulo si basa su UML, ma non è esattamente la metodologia usualmente associata a UML
- In particolare, ci concentriamo solo sugli aspetti base e sui diagrammi più importanti
- Nell'analisi concettuale useremo solo i seguenti diagrammi (e di questi diagrammi useremo solo le caratteristiche più importanti):
  - Diagrammi strutturali:
    - Diagramma delle classi e degli oggetti (class and object diagram)
  - Diagrammi comportamentali:
    - Diagramma degli use case (use case diagram),
    - Diagramma degli stati e delle transizioni (state/transition diagram)
- Useremo inoltre UML con alcune limitazioni e regole precise