

# Ingegneria del Software

Docente: Carmine Gravino  
Anno Accademico: 2023/2024

# Ingegneria del Software: Perché?

Facciamo un passo indietro, cercando di mettere insieme le conoscenze di programmazione da voi acquisite nel corso dei vostri primi anni di università.

Facciamo un passo indietro, cercando di mettere insieme le conoscenze di programmazione da voi acquisite nel corso dei vostri primi anni di università.

## 1° anno

- *Programmazione I.* Introduzione alla programmazione, oltre che a concetti quali specifica, precondizione, postcondizione e progettazione di soluzioni di programmazione.
- *Programmazione e strutture dati.* Complementi di programmazione, oltre che introduzione alle strutture dati e alla loro gestione.

# Ingegneria del Software: Perché?

Facciamo un passo indietro, cercando di mettere insieme le conoscenze di programmazione da voi acquisite nel corso dei vostri primi anni di università.

## 1° anno

- *Programmazione I.* Introduzione alla programmazione, oltre che a concetti quali specifica, precondizione, postcondizione e progettazione di soluzioni di programmazione.
- *Programmazione e strutture dati.* Complementi di programmazione, oltre che introduzione alle strutture dati e alla loro gestione.

## 2° anno

- *Programmazione Object-Oriented.* Introduzione alla programmazione Object-Oriented, con introduzione a concetti quali astrazione dei dati, encapsulamento dell'informazione, coesione ed accoppiamento, riuso del codice.
- *Progettazione di Algoritmi.* Divide-et-impera, programmazione dinamica e algoritmi su grafi.
- *Basi di dati.* Introduzione alla modellazione, oltre che ai metodi di gestione di basi di dati.
- *Tecnologie del Software per il Web.* Introduzione alle architetture software e allo sviluppo web.

# Ingegneria del Software: Perché?

Facciamo un passo indietro, cercando di mettere insieme le conoscenze di programmazione da voi acquisite nel corso dei vostri primi anni di università.

## 1° anno

- *Programmazione I.* Introduzione alla programmazione, oltre che a concetti quali specifica, precondizione, postcondizione e progettazione di soluzioni di programmazione.
- *Programmazione e strutture dati.* Complementi di programmazione, oltre che introduzione alle strutture dati e alla loro gestione.

## 2° anno

- *Programmazione Object-Oriented.* Introduzione alla programmazione Object-Oriented, con introduzione a concetti quali astrazione dei dati, encapsulamento dell'informazione, coesione ed accoppiamento, riuso del codice.
- *Progettazione di Algoritmi.* Divide-et-impera, programmazione dinamica e algoritmi su grafi.
- *Basi di dati.* Introduzione alla modellazione, oltre che ai metodi di gestione di basi di dati.
- *Tecnologie del Software per il Web.* Introduzione alle architetture software e allo sviluppo web.

Arrivati al terzo anno, quindi, dovreste essere già dei buoni programmati che sanno progettare algoritmi efficienti, programmarli in linguaggi procedurali ed orientati agli oggetti e gestire dati modellando e gestendo una base di dati. E allora perché un corso di IS?

# Ingegneria del Software: Perché?

Potenzialmente, tra meno di un anno potreste essere assunti da un'azienda. Lavoriamo di immaginazione, proiettandoci nel futuro.

# Ingegneria del Software: Perché?

Potenzialmente, tra meno di un anno potreste essere assunti da un'azienda. Lavoriamo di immaginazione, proiettandoci nel futuro.



E' il 3 Marzo 2025. Siete laureati, il futuro vi sorride! Avete da poco trovato lavoro in un'azienda di informatica, la quale vi ha proposto un buon contratto ed una prospettiva sul futuro.

# Ingegneria del Software: Perché?

Potenzialmente, tra meno di un anno potreste essere assunti da un'azienda. Lavoriamo di immaginazione, proiettandoci nel futuro.



E' il 3 Marzo 2025. Siete laureati, il futuro vi sorride! Avete da poco trovato lavoro in un'azienda di informatica, la quale vi ha proposto un buon contratto ed una prospettiva sul futuro.

Dopo un periodo di integrazione all'interno di un team di sviluppo, il vostro project manager vi chiede di far parte di un nuovo progetto commissionato da un'importante multinazionale.

# Ingegneria del Software: Perché?

Potenzialmente, tra meno di un anno potreste essere assunti da un'azienda. Lavoriamo di immaginazione, proiettandoci nel futuro.



E' il 3 Marzo 2025. Siete laureati, il futuro vi sorride! Avete da poco trovato lavoro in un'azienda di informatica, la quale vi ha proposto un buon contratto ed una prospettiva sul futuro.

Dopo un periodo di integrazione all'interno di un team di sviluppo, il vostro project manager vi chiede di far parte di un nuovo progetto commissionato da un'importante multinazionale.

In particolare, il progetto ha l'obiettivo di supportare i pagamenti della pubblica amministrazione tramite un'applicazione web-based che dovrà essere resiliente ad attacchi esterni ed effettuare predizioni sul grado di affidabilità creditizia degli utenti.

# Ingegneria del Software: Perché?

Potenzialmente, tra meno di un anno potreste essere assunti da un'azienda. Lavoriamo di immaginazione, proiettandoci nel futuro.



E' il 3 Marzo 2025. Siete laureati, il futuro vi sorride! Avete da poco trovato lavoro in un'azienda di informatica, la quale vi ha proposto un buon contratto ed una prospettiva sul futuro.

Dopo un periodo di integrazione all'interno di un team di sviluppo, il vostro project manager vi chiede di far parte di un nuovo progetto commissionato da un'importante multinazionale.

In particolare, il progetto ha l'obiettivo di supportare i pagamenti della pubblica amministrazione tramite un'applicazione web-based che dovrà essere resiliente ad attacchi esterni ed effettuare predizioni sul grado di affidabilità creditizia degli utenti.

*Deadline per la prima release: 30 Giugno 2025*



# Ingegneria del Software: Perché?

Potenzialmente, tra meno di un anno potreste essere assunti da un'azienda. Lavoriamo di immaginazione, proiettandoci nel futuro.



E' il 3 Marzo 2025. Siete laureati, il futuro vi sorride! Avete da poco trovato lavoro in un'azienda di informatica, la quale vi ha proposto un buon contratto ed una prospettiva sul futuro.

Dopo un periodo di integrazione all'interno di un team di sviluppo, il vostro project manager vi chiede di far parte di un nuovo progetto commissionato da un'importante multinazionale.

In particolare, il progetto ha l'obiettivo di supportare i pagamenti della pubblica amministrazione tramite un'applicazione web-based che dovrà essere resiliente ad attacchi esterni ed effettuare predizioni sul grado di affidabilità creditizia degli utenti.

*Deadline per la prima release: 30 Giugno 2025*

*Costo stimato dall'azienda: 375.000 €*



# Ingegneria del Software: Perché?

Potenzialmente, tra meno di un anno potreste essere assunti da un'azienda. Lavoriamo di immaginazione, proiettandoci nel futuro.



E' il 3 Marzo 2025. Siete laureati, il futuro vi sorride! Avete da poco trovato lavoro in un'azienda di informatica, la quale vi ha proposto un buon contratto ed una prospettiva sul futuro.

Dopo un periodo di integrazione all'interno di un team di sviluppo, il vostro project manager vi chiede di far parte di un nuovo progetto commissionato da un'importante multinazionale.

In particolare, il progetto ha l'obiettivo di supportare i pagamenti della pubblica amministrazione tramite un'applicazione web-based che dovrà essere resiliente ad attacchi esterni ed effettuare predizioni sul grado di affidabilità creditizia degli utenti.

*Deadline per la prima release: 30 Giugno 2025*

*Costo stimato dall'azienda: 375.000 €*

*Unità a disposizione per lo sviluppo: 13*



# Ingegneria del Software: Perché?

Potenzialmente, tra meno di un anno potreste essere assunti da un'azienda. Lavoriamo di immaginazione, proiettandoci nel futuro.



E' il 3 Marzo 2025. Siete laureati, il futuro vi sorride! Avete da poco trovato lavoro in un'azienda di informatica, la quale vi ha proposto un buon contratto ed una prospettiva sul futuro.

Dopo un periodo di integrazione all'interno di un team di sviluppo, il vostro project manager vi chiede di far parte di un nuovo progetto commissionato da un'importante multinazionale.

In particolare, il progetto ha l'obiettivo di supportare i pagamenti della pubblica amministrazione tramite un'applicazione web-based che dovrà essere resiliente ad attacchi esterni ed effettuare predizioni sul grado di affidabilità creditizia degli utenti.

*Deadline per la prima release: 30 Giugno 2025*

*Costo stimato dall'azienda: 375.000 €*

*Unità a disposizione per lo sviluppo: 13*

*Penale per il mancato raggiungimento dell'obiettivo: 150.000 €*



# Ingegneria del Software: Perché?

Potenzialmente, tra meno di un anno potreste essere assunti da un'azienda. Lavoriamo di immaginazione, proiettandoci nel futuro.



E' il 4 Marzo 2024. Siete laureati, il futuro vi sorride! Avete da poco trovato lavoro in un'azienda di informatica, la quale vi ha proposto un buon contratto ed una prospettiva sul futuro.

Dopo un periodo di integrazione all'interno di un team di sviluppo, il vostro project manager vi chiede di far parte di un nuovo progetto commissionato da un'importante multinazionale.

In particolare, il progetto ha l'obiettivo di supportare i pagamenti della pubblica amministrazione tramite un'applicazione web-based che dovrà essere resiliente ad attacchi esterni ed effettuare predizioni sul grado di affidabilità creditizia degli utenti.

*Deadline per la prima release: 30 Giugno 2025*

*Costo stimato dall'azienda: 375.000 €*

*Unità a disposizione per lo sviluppo: 13*

*Penale per il mancato raggiungimento dell'obiettivo: 150.000 €*

*Danni di immagine per eventuali difetti emersi durante l'esercizio: 1.872.000 €*



# Ingegneria del Software: Perché?

Potenzialmente, tra meno di un anno potreste essere assunti da un'azienda. Lavoriamo di immaginazione, proiettandoci nel futuro.



E' il 4 Marzo 2024. Siete laureati, il futuro vi sorride! Avete da poco trovato lavoro in un'azienda di informatica, la quale vi ha proposto un buon contratto ed una prospettiva sul futuro.

Dopo un periodo di integrazione all'interno di un team di sviluppo, il vostro project manager vi chiede di far parte di un nuovo progetto commissionato da un'importante multinazionale.

In particolare, il progetto ha l'obiettivo di supportare i pagamenti della pubblica amministrazione tramite un'applicazione web-based che dovrà essere resiliente ad attacchi esterni ed effettuare predizioni sul grado di affidabilità creditizia degli utenti.

*Deadline per la prima release: 30 Giugno 2025*

*Costo stimato dall'azienda: 375.000 €*

*Unità a disposizione per lo sviluppo: 13*

*Penale per il mancato raggiungimento dell'obiettivo: 150.000 €*

*Danni di immagine per eventuali difetti emersi durante l'esercizio: 1.872.000 €*

**Un incubo? No, è solo il mondo reale!** Per arrivare al raggiungimento dell'obiettivo non basta conoscere gli algoritmi, non basta conoscere le strutture dati o un linguaggio di programmazione. C'è bisogno di *progettazione, comunicazione e collaborazione!*



# Ingegneria del Software: Perché?

Analizziamo meglio lo scenario proposto...

*Dopo un periodo di integrazione all'interno di un **team di sviluppo**, il vostro **project manager** vi chiede di far parte di un nuovo progetto commissionato da un'importante multinazionale.*

Questo implica l'esistenza di una struttura complessa, molto spesso gerarchica, composta da un project manager ed una serie di sviluppatori (con diverse competenze ed esperienze):

# Ingegneria del Software: Perché?

Analizziamo meglio lo scenario proposto...

*Dopo un periodo di integrazione all'interno di un **team di sviluppo**, il vostro **project manager** vi chiede di far parte di un nuovo progetto commissionato da un'importante multinazionale.*

Questo implica l'esistenza di una struttura complessa, molto spesso gerarchica, composta da un project manager ed una serie di sviluppatori (con diverse competenze ed esperienze):

**“[...] supportare i pagamenti della pubblica amministrazione [...]”**

Il sistema software da realizzare avrà un obiettivo principale...

# Ingegneria del Software: Perché?

Analizziamo meglio lo scenario proposto...

*Dopo un periodo di integrazione all'interno di un **team di sviluppo**, il vostro **project manager** vi chiede di far parte di un nuovo progetto commissionato da un'importante multinazionale.*

Questo implica l'esistenza di una struttura complessa, molto spesso gerarchica, composta da un project manager ed una serie di sviluppatori (con diverse competenze ed esperienze):

**“[...] supportare i pagamenti della pubblica amministrazione [...]”**

Il sistema software da realizzare avrà un obiettivo principale...

*tramite un'applicazione web-based che dovrà essere **resiliente ad attacchi esterni** ed **effettuare predizioni sul grado di affidabilità creditizia degli utenti***

... ma anche diversi obiettivi tangenziali e vincoli, che dovranno essere rispettati in tempi utili e con l'adeguata qualità onde evitare conseguenze disastrose.

# Ingegneria del Software: Perché?

Analizziamo meglio lo scenario proposto...

*Dopo un periodo di integrazione all'interno di un **team di sviluppo**, il vostro **project manager** vi chiede di far parte di un nuovo progetto commissionato da un'importante multinazionale.*

Questo implica l'esistenza di una struttura complessa, molto spesso gerarchica, composta da un project manager ed una serie di sviluppatori (con diverse competenze ed esperienze):

**“[...] supportare i pagamenti della pubblica amministrazione [...]”**

Il sistema software da realizzare avrà un obiettivo principale...

*tramite un'applicazione web-based che dovrà essere **resiliente ad attacchi esterni** ed **effettuare predizioni sul grado di affidabilità creditizia degli utenti***

... ma anche diversi obiettivi tangenziali e vincoli, che dovranno essere rispettati in tempi utili e con l'adeguata qualità onde evitare conseguenze disastrose.

Per avere successo rispettando le scadenze, è  
spesso necessario lavorare come un team, ovvero  
suddividendo i compiti ed integrando il risultato  
(l'idea del divide-et-impera, dopotutto)

# Ingegneria del Software: Perché?

Analizziamo meglio lo scenario proposto...

*Dopo un periodo di integrazione all'interno di un **team di sviluppo**, il vostro **project manager** vi chiede di far parte di un nuovo progetto commissionato da un'importante multinazionale.*

Questo implica l'esistenza di una struttura complessa, molto spesso gerarchica, composta da un project manager ed una serie di sviluppatori (con diverse competenze ed esperienze):

**“[...] supportare i pagamenti della pubblica amministrazione [...]”**

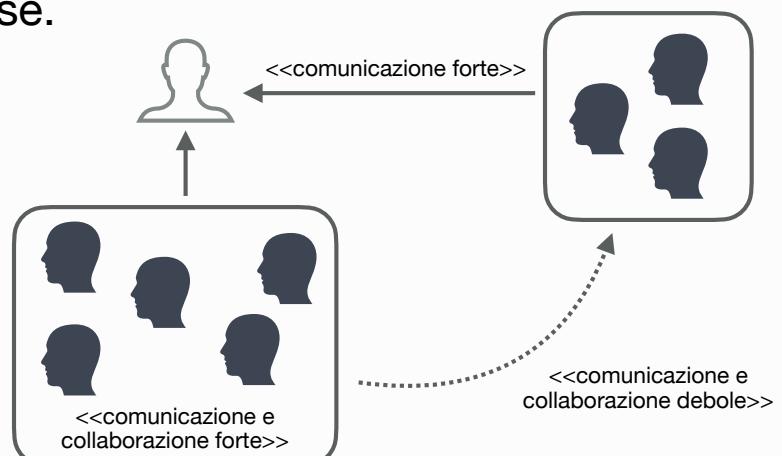
Il sistema software da realizzare avrà un obiettivo principale...

*tramite un'applicazione web-based che dovrà essere **resiliente ad attacchi esterni** ed **effettuare predizioni sul grado di affidabilità creditizia degli utenti***

... ma anche diversi obiettivi tangenziali e vincoli, che dovranno essere rispettati in tempi utili e con l'adeguata qualità onde evitare conseguenze disastrose.

Per avere successo rispettando le scadenze, è spesso necessario lavorare come un team, ovvero suddividendo i compiti ed integrando il risultato (l'idea del divide-et-impera, dopotutto)

Quindi, lo sviluppo potrebbe tradursi graficamente in qualcosa del genere.



# Ingegneria del Software: Perché?

In altri termini, lo sviluppo software è principalmente uno *sforzo collaborativo e comunicativo*, all'interno del quale la *progettazione* consente di raggiungere gli obiettivi prestabiliti con il committente.

# Ingegneria del Software: Perché?

In altri termini, lo sviluppo software è principalmente uno *sforzo collaborativo e comunicativo*, all'interno del quale la *progettazione* consente di raggiungere gli obiettivi prestabiliti con il committente.

Da qui il termine **Ingegneria**: “*Insieme di studi e tecniche che utilizzano le conoscenze delle varie branche delle scienze, unite a quelle tecnologiche, per risolvere problemi applicativi e per progettare e realizzare opere di diversa natura*” [Treccani]

# Ingegneria del Software: Perché?

In altri termini, lo sviluppo software è principalmente uno *sforzo collaborativo e comunicativo*, all'interno del quale la *progettazione* consente di raggiungere gli obiettivi prestabiliti con il committente.

Da qui il termine **Ingegneria**: “*Insieme di studi e tecniche che utilizzano le conoscenze delle varie branche delle scienze, unite a quelle tecnologiche, per risolvere problemi applicativi e per progettare e realizzare opere di diversa natura*” [Treccani]

Già nel corso di questa lezione, approfondiremo come la definizione viene mappata al software e quali sono le principali problematiche che rendono un processo ingegneristico inevitabile.

# Ingegneria del Software: Perché?

In altri termini, lo sviluppo software è principalmente uno *sforzo collaborativo e comunicativo*, all'interno del quale la *progettazione* consente di raggiungere gli obiettivi prestabiliti con il committente.

Da qui il termine **Ingegneria**: “*Insieme di studi e tecniche che utilizzano le conoscenze delle varie branche delle scienze, unite a quelle tecnologiche, per risolvere problemi applicativi e per progettare e realizzare opere di diversa natura*” [Treccani]

Già nel corso di questa lezione, approfondiremo come la definizione viene mappata al software e quali sono le principali problematiche che rendono un processo ingegneristico inevitabile.

In questo corso, apprenderete le implicazioni di queste problematiche e come gestirle - ma fortunatamente, lo apprenderete in un ambiente “controllato” in cui le conseguenze non sono catastrofiche né irreparabili!

# Ingegneria del Software: Perché?

In altri termini, lo sviluppo software è principalmente uno *sforzo collaborativo e comunicativo*, all'interno del quale la *progettazione* consente di raggiungere gli obiettivi prestabiliti con il committente.

Da qui il termine **Ingegneria**: “*Insieme di studi e tecniche che utilizzano le conoscenze delle varie branche delle scienze, unite a quelle tecnologiche, per risolvere problemi applicativi e per progettare e realizzare opere di diversa natura*” [Treccani]

Già nel corso di questa lezione, approfondiremo come la definizione viene mappata al software e quali sono le principali problematiche che rendono un processo ingegneristico inevitabile.

In questo corso, apprenderete le implicazioni di queste problematiche e come gestirle - ma fortunatamente, lo apprenderete in un ambiente “controllato” in cui le conseguenze non sono catastrofiche né irreparabili!

Apprenderete che le capacità tecniche, da sole, contano ben poco e che lo sviluppo software ha componenti e dinamiche umane, sociali ed organizzative paragonabili alla costruzione di qualunque altro prodotto architettonale ed ingegneristico - pertanto, richiede disciplina.

# Ingegneria del Software: Perché?

In altri termini, lo sviluppo software è principalmente uno *sforzo collaborativo e comunicativo*, all'interno del quale la *progettazione* consente di raggiungere gli obiettivi prestabiliti con il committente.

Da qui il termine **Ingegneria**: “*Insieme di studi e tecniche che utilizzano le conoscenze delle varie branche delle scienze, unite a quelle tecnologiche, per risolvere problemi applicativi e per progettare e realizzare opere di diversa natura*” [Treccani]

Già nel corso di questa lezione, approfondiremo come la definizione viene mappata al software e quali sono le principali problematiche che rendono un processo ingegneristico inevitabile.

In questo corso, apprenderete le implicazioni di queste problematiche e come gestirle - ma fortunatamente, lo apprenderete in un ambiente “controllato” in cui le conseguenze non sono catastrofiche né irreparabili!

Apprenderete che le capacità tecniche, da sole, contano ben poco e che lo sviluppo software ha componenti e dinamiche umane, sociali ed organizzative paragonabili alla costruzione di qualunque altro prodotto architettonale ed ingegneristico - pertanto, richiede disciplina.

Apprenderete infine che la progettazione di cui parleremo non è niente di più che la naturale evoluzione di ciò che avete già parlato nei precedenti corsi.

# Ingegneria del Software: Perché?

In altri termini, lo sviluppo software è principalmente uno *sforzo collaborativo e comunicativo*, all'interno del quale la *progettazione* consente di raggiungere gli obiettivi prestabiliti con il committente.

Da qui il termine **Ingegneria**: “*Insieme di studi e tecniche che utilizzano le conoscenze delle varie branche delle scienze, unite a quelle tecnologiche, per risolvere problemi applicativi e per progettare e realizzare opere di diversa natura*” [Treccani]

Già nel corso di questa lezione, approfondiremo come la definizione viene mappata al software e quali sono le principali problematiche che rendono un processo ingegneristico inevitabile.

In questo corso, apprenderete le implicazioni di queste problematiche e come gestirle - ma fortunatamente, lo apprenderete in un ambiente “controllato” in cui le conseguenze non sono catastrofiche né irreparabili!

Apprenderete che le capacità tecniche, da sole, contano ben poco e che lo sviluppo software ha componenti e dinamiche umane, sociali ed organizzative paragonabili alla costruzione di qualunque altro prodotto architettonico ed ingegneristico - pertanto, richiede disciplina.

Apprenderete infine che la progettazione di cui parleremo non è niente di più che la naturale evoluzione di ciò che avete visto nei precedenti corsi.

In che senso? Facciamo un esempio...

# Ingegneria del Software: Perché?

Ripercorriamo i passi eseguiti durante lo sviluppo di un algoritmo. Consideriamo ciò che, volontariamente o no, avete fatto per sviluppare un algoritmo di visita in ampiezza di un albero.

# Ingegneria del Software: Perché?

Ripercorriamo i passi eseguiti durante lo sviluppo di un algoritmo. Consideriamo ciò che, volontariamente o no, avete fatto per sviluppare un algoritmo di visita in ampiezza di un albero.

Analisi dei Requisiti

Prima di iniziare a sviluppare, vi siete sicuramente posti la domanda: “Cosa devo sviluppare?”. In altri termini, avete svolto analisi dei requisiti.

# Ingegneria del Software: Perché?

Ripercorriamo i passi eseguiti durante lo sviluppo di un algoritmo. Consideriamo ciò che, volontariamente o no, avete fatto per sviluppare un algoritmo di visita in ampiezza di un albero.

Analisi dei Requisiti

Prima di iniziare a sviluppare, vi siete sicuramente posti la domanda: “Cosa devo sviluppare?”. In altri termini, avete svolto analisi dei requisiti.

Progettazione di alto livello

Dopo aver capito cosa, avrete sicuramente iniziato a pensare al *come*, magari attraverso lo sviluppo e il raffinamento di un’idea di base.

# Ingegneria del Software: Perché?

Ripercorriamo i passi eseguiti durante lo sviluppo di un algoritmo. Consideriamo ciò che, volontariamente o no, avete fatto per sviluppare un algoritmo di visita in ampiezza di un albero.

Analisi dei Requisiti

Prima di iniziare a sviluppare, vi siete sicuramente posti la domanda: “Cosa devo sviluppare?”. In altri termini, avete svolto analisi dei requisiti.

Progettazione di alto livello

Dopo aver capito cosa, avrete sicuramente iniziato a pensare al *come*, magari attraverso lo sviluppo e il raffinamento di un’idea di base.

Probabilmente, tutto ciò ha portato allo sviluppo di pseudocodice...

```
1  funzione BFS( $G, v$ ):
2      crea coda  $Q$ 
3      inserisci  $v$  in  $Q$ 
4      marca  $v$ 
5      while  $Q$  non è vuota:
6           $t \leftarrow Q.\text{toglidallacoda}()$ 
7          if  $t$  è quello che stavamo cercando:
8              return  $t$ 
9          for all lati  $e$  in  $G.\text{latiincidenti}(t)$  do
12              $u \leftarrow G.\text{nodiadiacenti}(t, e)$ 
13             if  $u$  non è marcato:
14                 marca  $u$ 
15                 inserisci  $u$  in  $Q$ 
16     return none
```

# Ingegneria del Software: Perché?

Ripercorriamo i passi eseguiti durante lo sviluppo di un algoritmo. Consideriamo ciò che, volontariamente o no, avete fatto per sviluppare un algoritmo di visita in ampiezza di un albero.

Analisi dei Requisiti

Prima di iniziare a sviluppare, vi siete sicuramente posti la domanda: “Cosa devo sviluppare?”. In altri termini, avete svolto analisi dei requisiti.

Progettazione di alto livello

Dopo aver capito cosa, avrete sicuramente iniziato a pensare al *come*, magari attraverso lo sviluppo e il raffinamento di un’idea di base.

Probabilmente, tutto ciò ha portato allo sviluppo di pseudocodice...

```
1  funzione BFS( $G, v$ ):
2      crea coda  $Q$ 
3      inserisci  $v$  in  $Q$ 
4      marca  $v$ 
5      while  $Q$  non è vuota:
6           $t \leftarrow Q.\text{toglidallacoda}()$ 
7          if  $t$  è quello che stavamo cercando:
8              return  $t$ 
9          for all lati  $e$  in  $G.\text{latiincidenti}(t)$  do
12              $u \leftarrow G.\text{nodiadiacenti}(t,e)$ 
13             if  $u$  non è marcato:
14                 marca  $u$ 
15                 inserisci  $u$  in  $Q$ 
16     return none
```

Progettazione di basso livello

Sulla base della progettazione di alto livello, avete poi pensato agli strumenti da usare per implementare realmente l’algoritmo (es., il linguaggio da usare).

# Ingegneria del Software: Perché?

Ripercorriamo i passi eseguiti durante lo sviluppo di un algoritmo. Consideriamo ciò che, volontariamente o no, avete fatto per sviluppare un algoritmo di visita in ampiezza di un albero.

Analisi dei Requisiti

Prima di iniziare a sviluppare, vi siete sicuramente posti la domanda: “Cosa devo sviluppare?”. In altri termini, avete svolto analisi dei requisiti.

Progettazione di alto livello

Dopo aver capito cosa, avrete sicuramente iniziato a pensare al *come*, magari attraverso lo sviluppo e il raffinamento di un’idea di base.

Probabilmente, tutto ciò ha portato allo sviluppo di pseudocodice...

```
1 funzione BFS( $G, v$ ):
2     crea coda  $Q$ 
3     inserisci  $v$  in  $Q$ 
4     marca  $v$ 
5     while  $Q$  non è vuota:
6          $t \leftarrow Q.\text{toglidallacoda}()$ 
7         if  $t$  è quello che stavamo cercando:
8             return  $t$ 
9         for all lati  $e$  in  $G.\text{latiincidenti}(t)$  do
12             $u \leftarrow G.\text{nodiadiacenti}(t,e)$ 
13            if  $u$  non è marcato:
14                marca  $u$ 
15                inserisci  $u$  in  $Q$ 
16    return none
```

Solo dopo avete implementato. E magari qualcuno di voi ha anche commentato il codice prodotto per ragioni di comprensibilità...

```
coda = [R]
while coda:
    # operazione di dequeue
    vertice = coda.pop(0)
    # operazione di enqueue su ogni figlio
    coda.extend(vertice.figli)
```



Progettazione di basso livello

Sulla base della progettazione di alto livello, avete poi pensato agli strumenti da usare per implementare realmente l’algoritmo (es., il linguaggio da usare).

# Ingegneria del Software: Perché?

Ripercorriamo i passi eseguiti durante lo sviluppo di un algoritmo. Consideriamo ciò che, volontariamente o no, avete fatto per sviluppare un algoritmo di visita in ampiezza di un albero.

Analisi dei Requisiti

Prima di iniziare a sviluppare, vi siete sicuramente posti la domanda: “Cosa devo sviluppare?”. In altri termini, avete svolto analisi dei requisiti.

Progettazione di alto livello

Dopo aver capito cosa, avrete sicuramente iniziato a pensare al *come*, magari attraverso lo sviluppo e il raffinamento di un’idea di base.

Probabilmente, tutto ciò ha portato allo sviluppo di pseudocodice...

```
1 funzione BFS( $G, v$ ):
2     crea coda  $Q$ 
3     inserisci  $v$  in  $Q$ 
4     marca  $v$ 
5     while  $Q$  non è vuota:
6          $t \leftarrow Q.\text{toglidallacoda}()$ 
7         if  $t$  è quello che stavamo cercando:
8             return  $t$ 
9         for all lati  $e$  in  $G.\text{latiincidenti}(t)$  do
12             $u \leftarrow G.\text{nodiadiacenti}(t,e)$ 
13            if  $u$  non è marcato:
14                marca  $u$ 
15                inserisci  $u$  in  $Q$ 
16    return none
```

Solo dopo avete implementato. E magari qualcuno di voi ha anche commentato il codice prodotto per ragioni di comprensibilità...



```
coda = [R]
while coda:
    # operazione di dequeue
    vertice = coda.pop(0)
    # operazione di enqueue su ogni figlio
    coda.extend(vertice.figli)
```

Progettazione di basso livello

Sulla base della progettazione di alto livello, avete poi pensato agli strumenti da usare per implementare realmente l’algoritmo (es., il linguaggio da usare).

Testing

A valle dell’implementazione, avrete sicuramente “provato” il vostro codice, esercitandolo per verificarne la sua corretta implementazione.

# Ingegneria del Software: Perché?

In altri termini, lo sviluppo software è principalmente uno *sforzo collaborativo e comunicativo*, all'interno del quale la *progettazione* consente di raggiungere gli obiettivi prestabiliti con il committente.

Da qui il termine **Ingegneria**: “*Insieme di studi e tecniche che utilizzano le conoscenze delle varie branche delle scienze, unite a quelle tecnologiche, per risolvere problemi applicativi e per progettare e realizzare opere di diversa natura*” [Treccani]

Già nel corso di questa lezione, approfondiremo come la definizione viene mappata al software e quali sono le principali problematiche che rendono un processo ingegneristico inevitabile.

In questo corso, apprenderete le implicazioni di queste problematiche e come gestirle - ma fortunatamente, lo apprenderete in un ambiente “controllato” in cui le conseguenze non sono catastrofiche né irreparabili!

Apprenderete che le capacità tecniche, da sole, contano ben poco e che lo sviluppo software ha componenti e dinamiche umane, sociali ed organizzative paragonabili alla costruzione di qualunque altro prodotto architettonico ed ingegneristico - pertanto, richiede disciplina.

Apprenderete infine che la progettazione di cui parleremo non è niente di più che la naturale evoluzione di ciò che avete già parlato nei precedenti corsi.

In altri termini, avete SEMPRE eseguito attività tipiche dell'Ingegneria del Software. Lo avete però fatto nel “piccolo”, mentre questo corso vi fornirà gli strumenti per sviluppare sistemi software complessi e di grandi dimensioni!

# Ingegneria del Software: Overview del Corso

**M1. Introduzione all'Ingegneria  
del software**

**M2. Analisi e specifica dei  
requisiti**

**M3. Progettazione di alto livello  
ed architetturale**

**M4. Progettazione di basso  
livello**

**M5. Verifica e validazione del  
software**

# Ingegneria del Software: Overview del Corso

## M1. Introduzione all'Ingegneria del software

## M2. Analisi e specifica dei requisiti

## M3. Progettazione di alto livello ed architetturale

## M4. Progettazione di basso livello

## M5. Verifica e validazione del software

- Concetti di base: Processo e prodotto software;
- Cicli di vita del software;
- Cenni su qualità del software, project management, lavoro collaborativo e software configuration management;
- Introduzione alla modellazione orientata agli oggetti e a Unified Modeling Language (UML).

# Ingegneria del Software: Overview del Corso

**M1. Introduzione all'Ingegneria  
del software**

**M2. Analisi e specifica dei  
requisiti**

**M3. Progettazione di alto livello  
ed architetturale**

**M4. Progettazione di basso  
livello**

**M5. Verifica e validazione del  
software**

- Raccolta dei requisiti, scenari e casi d'uso;
- Modellazione ad oggetti e diagramma delle classi UML;
- Modellazione dinamica: diagramma di interazione, diagrammi di stato e diagramma delle attività;
- Documento di analisi dei requisiti

# Ingegneria del Software: Overview del Corso

**M1. Introduzione all'Ingegneria  
del software**

**M2. Analisi e specifica dei  
requisiti**

**M3. Progettazione di alto livello  
ed architetturale**

**M4. Progettazione di basso  
livello**

**M5. Verifica e validazione del  
software**

- Progettazione e decomposizione in moduli del sistema software;
- Principi di coesione ed accoppiamento;
- Stili architetturali;
- Architettura fisica di un sistema software, component diagram e deployment diagram UML;
- Documento di System Design;
- Gestione del razionale di design.

# Ingegneria del Software: Overview del Corso

**M1. Introduzione all'Ingegneria  
del software**

**M2. Analisi e specifica dei  
requisiti**

**M3. Progettazione di alto livello  
ed architetturale**

**M4. Progettazione di basso  
livello**

**M5. Verifica e validazione del  
software**

- Riuso del software e design pattern;
- Specifica delle interfacce delle classi;
- Introduzione all'Object Constraint Language (OCL);
- Mapping dei modelli al codice sorgente;
- Refactoring e reverse engineering;
- Documento di Object Design.

# Ingegneria del Software: Overview del Corso

**M1. Introduzione all'Ingegneria  
del software**

**M2. Analisi e specifica dei  
requisiti**

**M3. Progettazione di alto livello  
ed architetturale**

**M4. Progettazione di basso  
livello**

**M5. Verifica e validazione del  
software**

- Verifica e convalida del software;
- Livelli di testing: unità, integrazione e sistema;
- Attività di pianificazione del testing;
- Documentazione di testing;
- Principali tecniche di testing funzionale (black-box) e strutturale (white-box).

# Ingegneria del Software: Overview del Corso

**M1. Introduzione all'Ingegneria  
del software**

**M2. Analisi e specifica dei  
requisiti**

**M3. Progettazione di alto livello  
ed architetturale**

**M4. Progettazione di basso  
livello**

**M5. Verifica e validazione del  
software**

- Verifica e convalida del software;
- Livelli di testing: unità, integrazione e sistema;
- Attività di pianificazione del testing;
- Documentazione di testing;
- Principali tecniche di testing funzionale (black-box) e strutturale (white-box).

*Il “solito” corso di Ingegneria del Software, no?*

# Ingegneria del Software: Overview del Corso

**M1. Introduzione all'Ingegneria  
del software**

**M2. Analisi e specifica dei  
requisiti**

**M3. Progettazione di alto livello  
ed architetturale**

**M4. Progettazione di basso  
livello**

**M5. Verifica e validazione del  
software**

- Verifica e convalida del software;
- Livelli di testing: unità, integrazione e sistema;
- Attività di pianificazione del testing;
- Documentazione di testing;
- Principali tecniche di testing funzionale (black-box) e strutturale (white-box).

*Il “solito” corso di Ingegneria del Software, no?*

Non proprio - e non lasciatevi fuorviare dai commenti dei vostri predecessori! Ci saranno diverse novità.

# Ingegneria del Software: Overview del Corso

**M1. Introduzione all'Ingegneria  
del software**

**M2. Analisi e specifica dei  
requisiti**

**M3. Progettazione di alto livello  
ed architetturale**

**M4. Progettazione di basso  
livello**

**M5. Verifica e validazione del  
software**

- Verifica e convalida del software;
- Livelli di testing: unità, integrazione e sistema;
- Attività di pianificazione del testing;
- Documentazione di testing;
- Principali tecniche di testing funzionale (black-box) e strutturale (white-box).

*Il “solito” corso di Ingegneria del Software, no?*

Non proprio - e non lasciatevi fuorviare dai commenti dei vostri predecessori! Ci saranno diverse novità.

- La parte di laboratorio cambierà, nell'ottica di aggiornare e migliorare;

# Ingegneria del Software: Overview del Corso

**M1. Introduzione all'Ingegneria  
del software**

**M2. Analisi e specifica dei  
requisiti**

**M3. Progettazione di alto livello  
ed architetturale**

**M4. Progettazione di basso  
livello**

**M5. Verifica e validazione del  
software**

- Verifica e convalida del software;
- Livelli di testing: unità, integrazione e sistema;
- Attività di pianificazione del testing;
- Documentazione di testing;
- Principali tecniche di testing funzionale (black-box) e strutturale (white-box).

*Il “solito” corso di Ingegneria del Software, no?*

Non proprio - e non lasciatevi fuorviare dai commenti dei vostri predecessori! Ci saranno diverse novità.

- La parte di laboratorio cambierà, nell'ottica di aggiornare e migliorare
- I vincoli e la gestione dei progetti subiranno piccoli cambiamenti;

# Ingegneria del Software: Overview del Corso

**M1. Introduzione all'Ingegneria  
del software**

**M2. Analisi e specifica dei  
requisiti**

**M3. Progettazione di alto livello  
ed architetturale**

**M4. Progettazione di basso  
livello**

**M5. Verifica e validazione del  
software**

- Verifica e convalida del software;
- Livelli di testing: unità, integrazione e sistema;
- Attività di pianificazione del testing;
- Documentazione di testing;
- Principali tecniche di testing funzionale (black-box) e strutturale (white-box).

*Il “solito” corso di Ingegneria del Software, no?*

Non proprio - e non lasciatevi fuorviare dai commenti dei vostri predecessori! Ci saranno diverse novità che renderanno il corso profondamente diverso rispetto agli anni passati.

- La parte di laboratorio cambierà, nell'ottica di aggiornare e migliorare
- I vincoli e la gestione dei progetti subiranno piccoli cambiamenti;
- Il materiale didattico sarà rimodulato in modo da dare maggior peso ad alcuni aspetti.

# Ingegneria del Software: Overview del Corso

**M1. Introduzione all'Ingegneria  
del software**

**M2. Analisi e specifica dei  
requisiti**

**M3. Progettazione di alto livello  
ed architetturale**

**M4. Progettazione di basso  
livello**

**M5. Verifica e validazione del  
software**

- Verifica e convalida del software;
- Livelli di testing: unità, integrazione e sistema;
- Attività di pianificazione del testing;
- Documentazione di testing;
- Principali tecniche di testing funzionale (black-box) e strutturale (white-box).

*Il “solito” corso di Ingegneria del Software, no?*

Non proprio - e non lasciatevi fuorviare dai commenti dei vostri predecessori! Ci saranno diverse novità che renderanno il corso profondamente diverso rispetto agli anni passati.

- La parte di laboratorio cambierà, nell'ottica di aggiornare e migliorare
- I vincoli e la gestione dei progetti subiranno piccoli cambiamenti;
- Il materiale didattico sarà rimodulato in modo da dare maggior peso ad alcuni aspetti.

Pertanto, dimenticate ciò che è stato.

# Ingegneria del Software: Overview del Corso

**M1. Introduzione all'Ingegneria  
del software**

**M2. Analisi e specifica dei  
requisiti**

**M3. Progettazione di alto livello  
ed architetturale**

**M4. Progettazione di basso  
livello**

**M5. Verifica e validazione del  
software**

- Verifica e convalida del software;
- Livelli di testing: unità, integrazione e sistema;
- Attività di pianificazione del testing;
- Documentazione di testing;
- Principali tecniche di testing funzionale (black-box) e strutturale (white-box).

*Già guardando l'overview, sembra ci saranno tante scartoffie da preparare o, in alcuni casi, tanti documenti degli scorsi anni da copiare!*

# Ingegneria del Software: Overview del Corso

**M1. Introduzione all'Ingegneria  
del software**

**M2. Analisi e specifica dei  
requisiti**

**M3. Progettazione di alto livello  
ed architetturale**

**M4. Progettazione di basso  
livello**

**M5. Verifica e validazione del  
software**

- Verifica e convalida del software;
- Livelli di testing: unità, integrazione e sistema;
- Attività di pianificazione del testing;
- Documentazione di testing;
- Principali tecniche di testing funzionale (black-box) e strutturale (white-box).

*Già guardando l'overview, sembra ci saranno tante scartoffie da preparare o, in alcuni casi, tanti documenti degli scorsi anni da copiare!*

Decisamente no.

# Ingegneria del Software: Overview del Corso

**M1. Introduzione all'Ingegneria  
del software**

**M2. Analisi e specifica dei  
requisiti**

**M3. Progettazione di alto livello  
ed architetturale**

**M4. Progettazione di basso  
livello**

**M5. Verifica e validazione del  
software**

- Verifica e convalida del software;
- Livelli di testing: unità, integrazione e sistema;
- Attività di pianificazione del testing;
- Documentazione di testing;
- Principali tecniche di testing funzionale (black-box) e strutturale (white-box).

*Già guardando l'overview, sembra ci saranno tante scartoffie da preparare o, in alcuni casi, tanti documenti degli scorsi anni da copiare!*

Decisamente no.

- La documentazione **non** è vostra nemica, ma un supporto concreto allo sviluppo di sistemi software reali! Se seguirete con attenzione, lo capirete.

# Ingegneria del Software: Overview del Corso

**M1. Introduzione all'Ingegneria  
del software**

**M2. Analisi e specifica dei  
requisiti**

**M3. Progettazione di alto livello  
ed architetturale**

**M4. Progettazione di basso  
livello**

**M5. Verifica e validazione del  
software**

- Verifica e convalida del software;
- Livelli di testing: unità, integrazione e sistema;
- Attività di pianificazione del testing;
- Documentazione di testing;
- Principali tecniche di testing funzionale (black-box) e strutturale (white-box).

*Già guardando l'overview, sembra ci saranno tante scartoffie da preparare o, in alcuni casi, tanti documenti degli scorsi anni da copiare!*

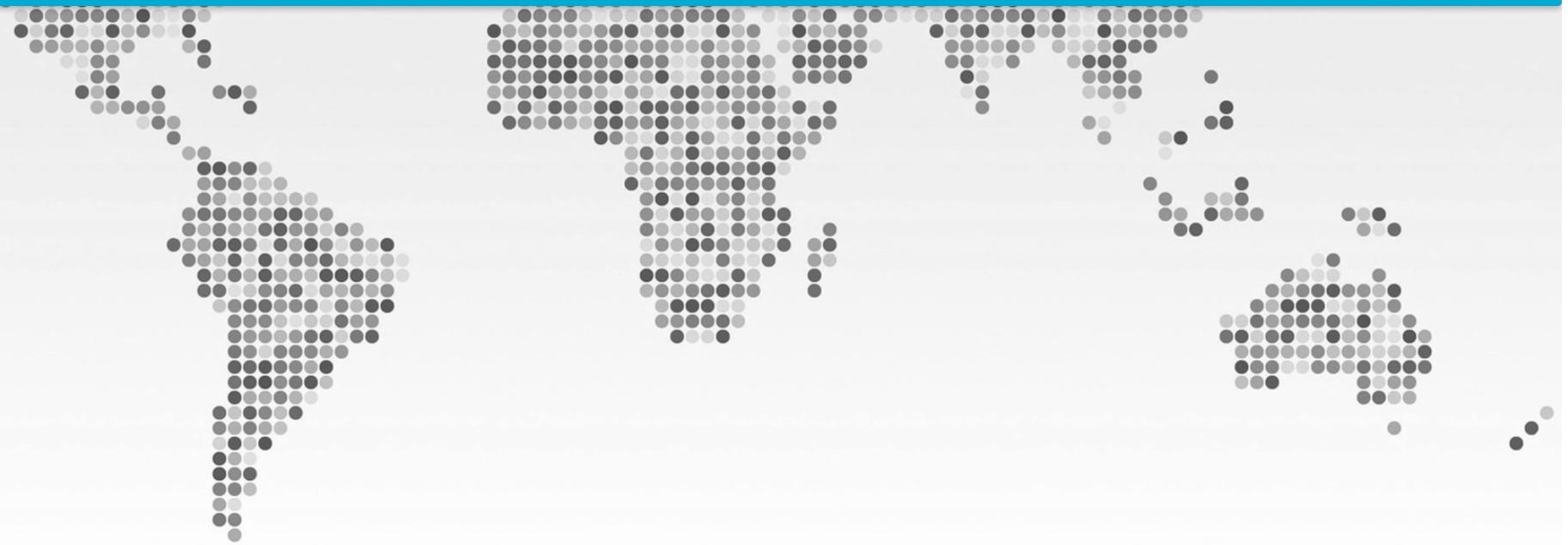
Decisamente no.

- La documentazione **non** è vostra nemica, ma un supporto concreto allo sviluppo di sistemi software reali! Se seguirete con attenzione, lo capirete.
- Copiare significa sbagliare nel 99.9% dei casi. Inoltre, un documento copiato è visibile già dalle prime righe. I docenti se ne accorgono sempre, non siate ingenui.

**Carmine Gravino**

SeSa Lab - University of Salerno

# Ingegneria del Software



# Ingegneria del Software: il docente



Carmine Gravino

Docente di: Ingegneria del Software (Triennale L-31), Metriche e Qualità del Software (Magistrale LM-18), Software Engineering for Secure Cloud Systems (Magistrale LM-66)

E-mail: [gravino@unisa.it](mailto:gravino@unisa.it)

Sito web: <https://docenti.unisa.it/004724/home>

Google Scholar: <https://scholar.google.com/citations?user=fGpVbKwAAAAJ&hl=en>

Twitter: [@GravinoCarmine](https://twitter.com/GravinoCarmine)

## SeSa Lab Software Engineering @ Salerno

[sesalab@unisa.it](mailto:sesalab@unisa.it) 

[sesa\\_lab](#)   

<https://sesalabunisa.github.io/> 



Andrea De Lucia



Filomena Ferrucci



Carmine Gravino



Dario Di Nucci



Fabio Palomba



Giammaria Giordano



Valeria Pontillo



Emanuele Iannone



Francesco Casillo



Stefano Lambiase



Giulia Sellitto



Vincenzo De Martino



Dario Di Dario



Gilberto Recupito



Gabriele De Vito



Alexandra Sheykina



Giusy Annunziata

## SeSa Lab Software Engineering @ Salerno

[sesalab@unisa.it](mailto:sesalab@unisa.it) 

[sesa\\_lab](#)   

<https://sesalabunisa.github.io/> 

### Education

- Ingegneria del Software (x3);
- Fondamenti di Intelligenza Artificiale;
- Programmazione Object-Oriented.
- Gestione dei Progetti Software;
- Ingegneria, Gestione ed Evoluzione del Software;
- Metriche e Qualità del Software;
- Software Engineering for AI;
- Software Dependability;
- Didattica dell'Informatica.
- Software Engineering for Secure Cloud Systems
- 100+ tesisti all'anno;
- Occasioni di tirocini esterni in azienda.
- Più info: <https://sesalabunisa.github.io/it/didattica.html>.



## SeSa Lab Software Engineering @ Salerno

[sesalab@unisa.it](mailto:sesalab@unisa.it)   
[sesa\\_lab](https://sesalabunisa.github.io/)     
<https://sesalabunisa.github.io/> 

### Education

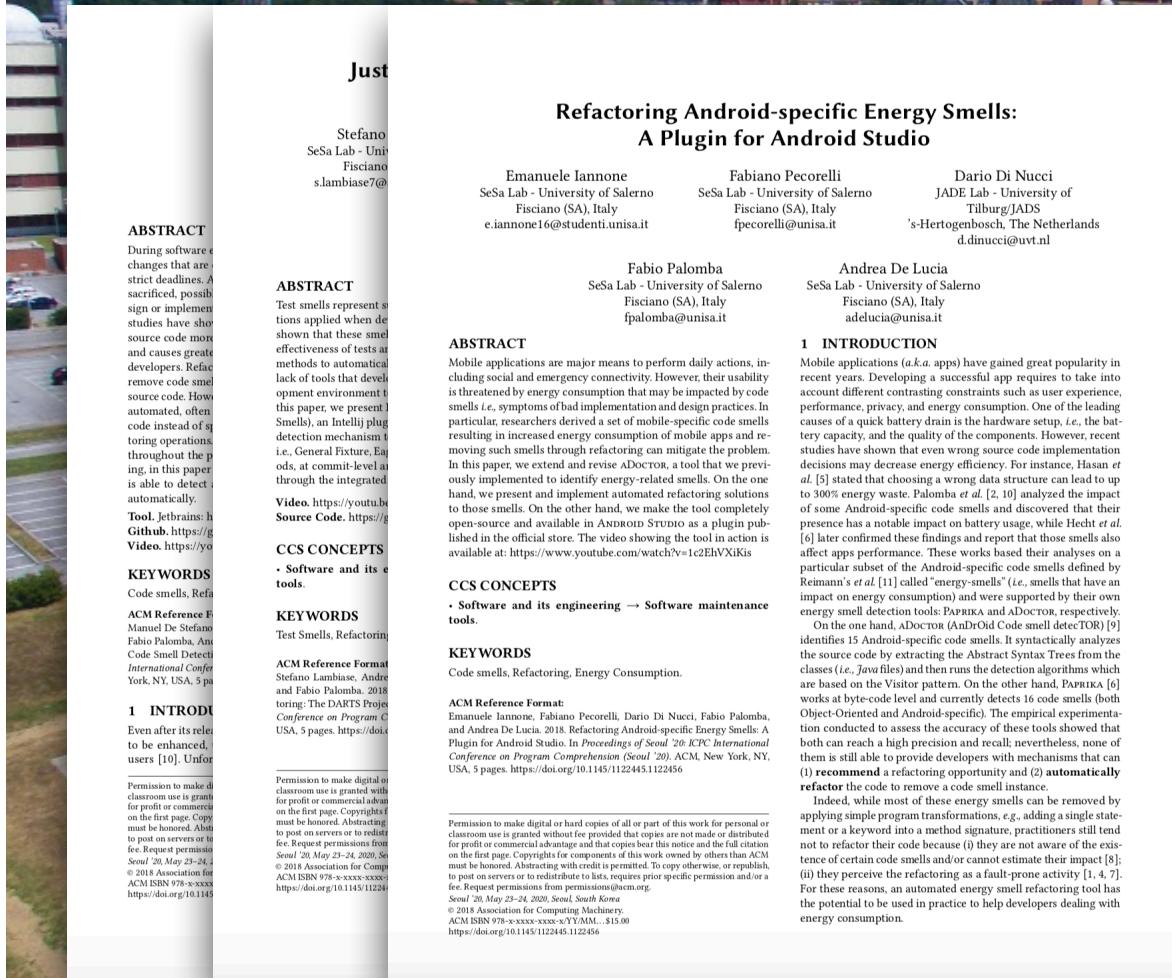
- Ingegneria del Software (x3);
- Fondamenti di Intelligenza Artificiale;
- Programmazione Object-Oriented.
- Gestione dei Progetti Software;
- Ingegneria, Gestione ed Evoluzione del Software;
- Metriche e Qualità del Software;
- Software Engineering for AI;
- Software Dependability;
- Didattica dell'Informatica.
- Software Engineering for Secure Cloud Systems
- 100+ tesisti all'anno;
- Occasioni di tirocini esterni in azienda.
- Più info: <https://sesalabunisa.github.io/it/didattica.html>.

### Research

- Ingegneria dei requisiti;
- Manutenzione ed evoluzione del software;
- Fondamenti di Data Science e Machine Learning per l'Ingegneria del Software;
- Sicurezza software;
- Verifica e convalida del software;
- Aspetti umani e gestione dei progetti software, inclusi sistemi ML e IoT.
- Quantum Software Engineering;
- Accessibilità e usabilità del software;
- Sviluppo del software sostenibile.
- Più info: <https://sesalabunisa.github.io/it/index.html>.

## SeSa Lab Software Engineering @ Salerno

[sesalab@unisa.it](mailto:sesalab@unisa.it)   
[sesa\\_lab](https://sesalab.unisa.it)     
<https://sesalabunisa.github.io/> 



Just another WordPress site

Stefano Lambiase  
SeSa Lab - University of Salerno  
Fisciano (SA), Italy  
s.lambiase7@unisa.it

**ABSTRACT**  
During software engineering, changes that are made to code often come at strict deadlines. A developer may sacrifice time spent on design or implementation to meet a deadline. Studies have shown that this leads to source code more prone to errors and causes greater costs for the organization and causes greater frustration for developers. Refactoring is a process that can remove code smells from the source code. However, refactoring is often automated, often by tools that do not support refactoring operations. Throughout the paper, we show that, in this paper, we present a tool that is able to detect and remove code smells automatically.

Tool: Jetrbrain's IntelliJ IDEA  
Github: <https://github.com/se-sa-lab/Android-Smells-Refactor>  
Code Source: <https://github.com/se-sa-lab/Android-Smells-Refactor>  
Video: <https://youtu.be/1c2EhVXikis>

**CCS CONCEPTS**  
• Software and its e-tools.

**KEYWORDS**  
Code smells, Refactoring, Energy consumption.

**ACM Reference Format:**  
Manuel De Stefano, Fabio Palomba, and Stefano Lambiase. 2018. Refactoring: The DARTS Project. In *Proceedings of the International Conference on Program Comprehension (Seoul '20)*. ACM, New York, NY, USA, 5 pages. <https://doi.org/10.1145/1122445.1122456>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted for profit or commercial advantage without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting or reusing portions of this work in whole or in part is permitted. To copy otherwise, or to republish, post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permission from permissions@acm.org.  
Seoul '20, May 23–24, 2020, Seoul, South Korea  
© 2020 Association for Computing Machinery  
ACM ISBN 978-x-xxxx-xxxx-x  
<https://doi.org/10.1145/1122445.1122456>

Refactoring Android-specific Energy Smells:  
A Plugin for Android Studio

Emanuele Iannone  
SeSa Lab - University of Salerno  
Fisciano (SA), Italy  
e.iannone16@studenti.unisa.it

Fabiano Pecorelli  
SeSa Lab - University of Salerno  
Fisciano (SA), Italy  
fpecorelli@unisa.it

Dario Di Nucci  
JADE Lab - University of  
Tilburg/JADS  
's-Hertogenbosch, The Netherlands  
d.diucci@uvt.nl

Fabio Palomba  
SeSa Lab - University of Salerno  
Fisciano (SA), Italy  
fpalomba@unisa.it

Andrea De Lucia  
SeSa Lab - University of Salerno  
Fisciano (SA), Italy  
adelucia@unisa.it

**ABSTRACT**  
Test smells represent situations applied when developing code. This paper shows that these smells are effective in terms of tests and methods to automatical lack of tools that development environment to detect them. In this paper, we present a tool (called Energy Smells), an IntelliJ plugin for Android Studio that detects mechanism i.e., General Fixture, Eadods, at commit-level automatically through the integrated refactoring tool. The tool is able to detect and remove code smells automatically.

Tool: Jetrbrain's IntelliJ IDEA  
Github: <https://github.com/se-sa-lab/Android-Smells-Refactor>  
Code Source: <https://github.com/se-sa-lab/Android-Smells-Refactor>  
Video: <https://youtu.be/1c2EhVXikis>

**CCS CONCEPTS**  
• Software and its engineering → Software maintenance tools.

**KEYWORDS**  
Code smells, Refactoring, Energy Consumption.

**ACM Reference Format:**  
Emanuele Iannone, Fabiano Pecorelli, Dario Di Nucci, Fabio Palomba, and Andrea De Lucia. 2018. Refactoring Android-specific Energy Smells: A Plugin for Android Studio. In *Proceedings of Seoul '20: ICPC International Conference on Program Comprehension (Seoul '20)*. ACM, New York, NY, USA, 5 pages. <https://doi.org/10.1145/1122445.1122456>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting or reusing portions of this work in whole or in part is permitted. To copy otherwise, or to republish, post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permission from permissions@acm.org.  
Seoul '20, May 23–24, 2020, Seoul, South Korea  
© 2020 Association for Computing Machinery  
ACM ISBN 978-x-xxxx-xxxx-x  
<https://doi.org/10.1145/1122445.1122456>

**1 Best Paper Award**  
**International Conference on Software Code, Analysis, and Manipulation**

**1 ACM/SIGSOFT Distinguished Paper Award**  
**International Conference on Automated Software Engineering**

**30+ articoli scientifici con studenti in conferenze e riviste internazionali**

**14 nuovi docenti negli ultimi 10 anni di attività**

<https://sesalabunisa.github.io/it/index.html>

# Il Software Engineering (SeSa) Lab - Collaborazioni Recenti



E altri...

# Informazioni amministrative generali

---

**Docente:** Carmine Gravino

**Tutor:** Da definire

**Importunateli solo se  
necessario!**

**Orario Lezioni:** Lunedì: 11:00 - 14:00 - Lab P13;  
Giovedì: 09:00 - 11:00 - Aula F1;  
Venerdì: 11:00 - 13:00 - Aula F1.

**Piattaforma E-learning:** <http://elearning.informatica.unisa.it/el-platform/course/view.php?id=819>

**Email:** [gravino@unisa.it](mailto:gravino@unisa.it)

**Orario di risposta alle  
email:** Lunedì—Venerdì 12:30 - 13:30

**Appuntamento sempre  
richiesto.**

**Orario di ricevimento:** Giovedì 11:00 - 12:00; Mercoledì 9:00 - 11:00.

---

# Informazioni amministrative - Comunicazione

Destinatario	<b><u>gravino@unisa.it</u></b>
Oggetto	<b>[IS] &lt;Oggetto esplicito, breve e conciso&gt;</b>
Corpo	<p><b>&lt;Saluto&gt;,</b></p> <p><b>&lt;presentazione del mittente&gt;</b></p> <p><b>&lt;chiara e sintetica descrizione della richiesta/comunicazione&gt;</b></p> <p><b>&lt;segnalazione di eventuali allegati&gt;</b></p> <p><b>&lt;Saluto&gt;</b></p> <p><b>&lt;Firma&gt;</b></p>

## Altre semplici avvertenze e suggerimenti:

- (1) La punteggiatura è importante, usatela appropriatamente;
- (2) Usate le lettere maiuscole, il grassetto e l'italico appropriatamente;
- (3) Siate ordinati, ad esempio dividendo il corpo in paragrafi;
- (4) Prima di inviare l'email, verificate la presenza di errori ortografici e/o di battitura;
- (5) Date del Lei, non del Voi (il periodo fascista è finito 80 anni fa!)
- (6) Le e-mail che non seguiranno il formato riportato **NON verranno considerate!**

# Informazioni amministrative - Comunicazione

Destinatario

**gravino@unisa.it**

Oggetto

**[IS] <Oggetto esplicito, breve e conciso>**

Corpo

**<Saluto>,**  
**<presentazione del mittente>**  
**<chiara e sintetica descrizione della richiesta/comunicazione>**  
  
**<segnalazione di eventuali allegati>**  
  
**<Saluto>**  
**<Firma>**

## Altre semplici avvertenze e suggerimenti:

- (1) La punteggiatura è importante, usatela appropriatamente;
- (2) Usate le lettere maiuscole, il grassetto e l'italico appropriatamente;
- (3) Siate ordinati, ad esempio dividendo il corpo in paragrafi;
- (4) Prima di inviare l'email, verificate la presenza di errori ortografici e/o di battitura;
- (5) Date del Lei, non del Voi (il periodo fascista è finito 80 anni fa!)
- (6) Le e-mail che non seguiranno il formato riportato **NON verranno considerate!**

# Informazioni amministrative - Comunicazione

Destinatario

**gravino@unisa.it**

Oggetto

**[IS] <Oggetto esplicito, breve e conciso>**

Corpo

**<Saluto>,**

**<presentazione del mittente>**

**<chiara e sintetica descrizione della richiesta/comunicazione>**

**<segnalazione di eventuali allegati>**

**<Saluto>**

**<Firma>**

**Ricorda di rispondere a tutti  
(funzione reply-all)**

**Se non ricevi risposta entro 48h,  
invia nuovamente la mail!**

## Altre semplici avvertenze e suggerimenti:

- (1) La punteggiatura è importante, usatela appropriatamente;
- (2) Usate le lettere maiuscole, il grassetto e l'italico appropriatamente;
- (3) Siate ordinati, ad esempio dividendo il corpo in paragrafi;
- (4) Prima di inviare l'email, verificate la presenza di errori ortografici e/o di battitura;
- (5) Date del Lei, non del Voi (il periodo fascista è finito 80 anni fa!)
- (6) Le e-mail che non seguiranno il formato riportato **NON verranno considerate!**

# Informazioni amministrative - Comunicazione

Destinatario

**gravino@unisa.it**

**Se la tua richiesta è a nome del tuo  
gruppo, ricorda di mettere in copia tutti i  
membri del team (funzione cc)**

Oggetto

**[IS] Richiesta di Esenzione da Prova Progettuale**

Corpo

Gentile Prof. Gravino,

Sono Mario Rossi, uno studente del corso di Ingegneria del Software.

Essendo uno studente lavoratore ed iscritto in modalità part-time, non ho modo di partecipare attivamente alle lezioni e sviluppare un progetto per il superamento dell'esame. Pertanto, vorrei chiedere l'esenzione dalla prova progettuale. Invio in allegato la richiesta, documentata così come da indicazioni fornite sulla piattaforma e-learning.

In attesa di un riscontro, grazie e buona giornata,  
Mario Rossi

Allegati



richiesta\_esenzione\_rossi.pdf

**A meno di richieste esplicite, i file si inviano  
SOLO ed ESCLUSIVAMENTE in formato pdf**

Altre informazioni sulla cosiddetta “netiquette”: <https://www.linkedin.com/pulse/la-netiquette-il-galateo-di-internet-15-regole-per-rete-gianquinto/?originalSubdomain=it>

## Informazioni amministrative - Il Ruolo dei Tutor

Come detto, i tutor ci assisteranno durante l'intero semestre. E' doveroso quindi spiegare meglio quali saranno i loro compiti ed in che modo è necessario interagire con loro.

## Informazioni amministrative - Il Ruolo dei Tutor

Come detto, i tutor ci assisteranno durante l'intero semestre. E' doveroso quindi spiegare meglio quali saranno i loro compiti ed in che modo è necessario interagire con loro.

(1) I tutor **non** sono tenuti a dare spiegazioni su argomenti trattati durante le lezioni frontali dai docenti — per quello, ci sono i docenti! Quindi, in caso di dubbi sugli argomenti trattati, vanno contattati i docenti (secondo le modalità descritte prima).

## Informazioni amministrative - Il Ruolo dei Tutor

Come detto, i tutor ci assisteranno durante l'intero semestre. E' doveroso quindi spiegare meglio quali saranno i loro compiti ed in che modo è necessario interagire con loro.

- (1) I tutor **non** sono tenuti a dare spiegazioni su argomenti trattati durante le lezioni frontali dai docenti — per quello, ci sono i docenti! Quindi, in caso di dubbi sugli argomenti trattati, vanno contattati i docenti (secondo le modalità descritte prima).
- (2) I tutor saranno spesso coinvolti durante le ore di laboratorio e presenteranno strumenti e/o nozioni utili per lo sviluppo del progetto. Quindi, **possono** essere contattati per eventuali dubbi e/o problemi nell'utilizzo degli strumenti.

## Informazioni amministrative - Il Ruolo dei Tutor

Come detto, i tutor ci assisteranno durante l'intero semestre. E' doveroso quindi spiegare meglio quali saranno i loro compiti ed in che modo è necessario interagire con loro.

- (1) I tutor **non** sono tenuti a dare spiegazioni su argomenti trattati durante le lezioni frontali dai docenti — per quello, ci sono i docenti! Quindi, in caso di dubbi sugli argomenti trattati, vanno contattati i docenti (secondo le modalità descritte prima).
- (2) I tutor saranno spesso coinvolti durante le ore di laboratorio e presenteranno strumenti e/o nozioni utili per lo sviluppo del progetto. Quindi, **possono** essere contattati per eventuali dubbi e/o problemi nell'utilizzo degli strumenti.

**NB:** Tuttavia, non sono dei tecnici —> il supporto non riguarda l'installazione/configurazione di tool o errori di compilazione del codice sorgente, ma riguarda l'uso appropriato di tool.

## Informazioni amministrative - Il Ruolo dei Tutor

Come detto, i tutor ci assisteranno durante l'intero semestre. E' doveroso quindi spiegare meglio quali saranno i loro compiti ed in che modo è necessario interagire con loro.

- (1) I tutor **non** sono tenuti a dare spiegazioni su argomenti trattati durante le lezioni frontali dai docenti — per quello, ci sono i docenti! Quindi, in caso di dubbi sugli argomenti trattati, vanno contattati i docenti (secondo le modalità descritte prima).
- (2) I tutor saranno spesso coinvolti durante le ore di laboratorio e presenteranno strumenti e/o nozioni utili per lo sviluppo del progetto. Quindi, **possono** essere contattati per eventuali dubbi e/o problemi nell'utilizzo degli strumenti.  
**NB:** Tuttavia, non sono dei tecnici —> il supporto non riguarda l'installazione/configurazione di tool o errori di compilazione del codice sorgente, ma riguarda l'uso appropriato di tool.
- (3) I tutor saranno spesso coinvolti nel monitoraggio dei vostri progetti. **Possono** supportarvi nella definizione e sviluppo del vostro progetto.

## Informazioni amministrative - Il Ruolo dei Tutor

Come detto, i tutor ci assisteranno durante l'intero semestre. E' doveroso quindi spiegare meglio quali saranno i loro compiti ed in che modo è necessario interagire con loro.

- (1) I tutor **non** sono tenuti a dare spiegazioni su argomenti trattati durante le lezioni frontali dai docenti — per quello, ci sono i docenti! Quindi, in caso di dubbi sugli argomenti trattati, vanno contattati i docenti (secondo le modalità descritte prima).
- (2) I tutor saranno spesso coinvolti durante le ore di laboratorio e presenteranno strumenti e/o nozioni utili per lo sviluppo del progetto. Quindi, **possono** essere contattati per eventuali dubbi e/o problemi nell'utilizzo degli strumenti.  
**NB:** Tuttavia, non sono dei tecnici —> il supporto non riguarda l'installazione/configurazione di tool o errori di compilazione del codice sorgente, ma riguarda l'uso appropriato di tool.
- (3) I tutor saranno spesso coinvolti nel monitoraggio dei vostri progetti. **Possono** supportarvi nella definizione e sviluppo del vostro progetto.  
**NB:** Tuttavia, non sono dei membri aggiuntivi del vostro team. Da loro si va per chiedere spiegazioni sui giusti metodi da utilizzare, ma non per far correggere i vostri artefatti.

# Informazioni amministrative - Il Ruolo dei Tutor

Come detto, i tutor ci assisteranno durante l'intero semestre. E' doveroso quindi spiegare meglio quali saranno i loro compiti ed in che modo è necessario interagire con loro.

- (1) I tutor **non** sono tenuti a dare spiegazioni su argomenti trattati durante le lezioni frontali dai docenti — per quello, ci sono i docenti! Quindi, in caso di dubbi sugli argomenti trattati, vanno contattati i docenti (secondo le modalità descritte prima).
- (2) I tutor saranno spesso coinvolti durante le ore di laboratorio e presenteranno strumenti e/o nozioni utili per lo sviluppo del progetto. Quindi, **possono** essere contattati per eventuali dubbi e/o problemi nell'utilizzo degli strumenti.  
**NB:** Tuttavia, non sono dei tecnici —> il supporto non riguarda l'installazione/configurazione di tool o errori di compilazione del codice sorgente, ma riguarda l'uso appropriato di tool.
- (3) I tutor saranno spesso coinvolti nel monitoraggio dei vostri progetti. **Possono** supportarvi nella definizione e sviluppo del vostro progetto.  
**NB:** Tuttavia, non sono dei membri aggiuntivi del vostro team. Da loro si va per chiedere spiegazioni sui giusti metodi da utilizzare, ma non per far correggere i vostri artefatti.
- (4) I tutor **non** sono tenuti a rispondere a domande su argomenti che non riguardano il corso di Ingegneria del Software - ad esempio, non possono colmare le lacune in Basi di Dati.

# Informazioni amministrative - Il Ruolo dei Tutor

Come detto, i tutor ci assisteranno durante l'intero semestre. E' doveroso quindi spiegare meglio quali saranno i loro compiti ed in che modo è necessario interagire con loro.

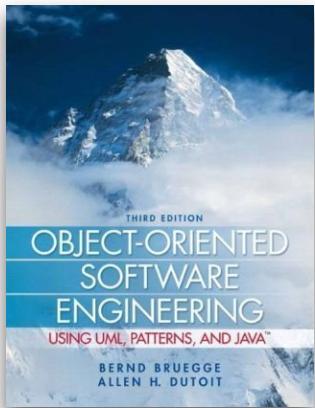
- (1) I tutor **non** sono tenuti a dare spiegazioni su argomenti trattati durante le lezioni frontali dai docenti — per quello, ci sono i docenti! Quindi, in caso di dubbi sugli argomenti trattati, vanno contattati i docenti (secondo le modalità descritte prima).
- (2) I tutor saranno spesso coinvolti durante le ore di laboratorio e presenteranno strumenti e/o nozioni utili per lo sviluppo del progetto. Quindi, **possono** essere contattati per eventuali dubbi e/o problemi nell'utilizzo degli strumenti.

**NB:** Tuttavia, non sono dei tecnici —> il supporto non riguarda l'installazione/configurazione di tool o errori di compilazione del codice sorgente, ma riguarda l'uso appropriato di tool.
- (3) I tutor saranno spesso coinvolti nel monitoraggio dei vostri progetti. **Possono** supportarvi nella definizione e sviluppo del vostro progetto.

**NB:** Tuttavia, non sono dei membri aggiuntivi del vostro team. Da loro si va per chiedere spiegazioni sui giusti metodi da utilizzare, ma non per far correggere i vostri artefatti.
- (4) I tutor **non** sono tenuti a rispondere a domande su argomenti che non riguardano il corso di Ingegneria del Software - ad esempio, non possono colmare le lacune in Basi di Dati.
- (5) I tutor hanno impostato una e-mail dedicata alla quale richiedere informazioni. Per la nostra classe la mail è: **[tutoratois2@gmail.com](mailto:tutoratois2@gmail.com)**. Utilizzate preferibilmente questa e-mail per interagire con loro, inviando l'eventuale materiale soggetto a chiarimento; ad esempio, se avete dubbi sulla scrittura di un sequence diagram, inviate ciò che avete prodotto prima di richiedere chiarimenti - in questo modo, la comunicazione sarà più proficua ed efficiente.

# Informazioni amministrative - Materiale Didattico

## Libri

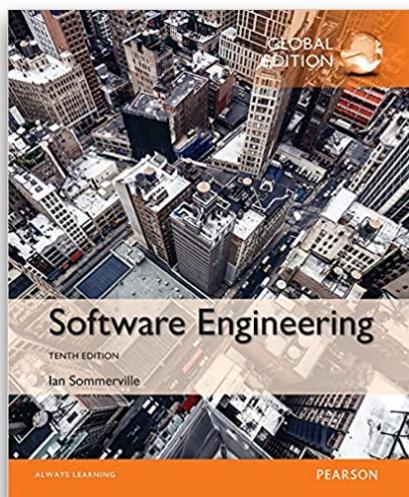


### Libro suggerito.

Bruegge, Bernd, and Allen H. Dutoit. "Object-oriented software engineering. using uml, patterns, and java." *Learning* 5.6 (2009): 7.

### Alternative

Per approfondimenti e/o ulteriori punti di vista sulle tematiche affrontate, i testi di seguito rappresentano buone alternative.



## Slide e materiale aggiuntivo

Disponibile sulla piattaforma e-learning del corso.

# Informazioni amministrative - Modalità di Esame

## Modalità di Esame

Come ogni esame, ci sarà un accertamento individuale della conoscenza dei contenuti trattati durante il corso. Questa avverrà in due momenti.

## Modalità di Esame

Come ogni esame, ci sarà un accertamento individuale della conoscenza dei contenuti trattati durante il corso. Questa avverrà in due momenti.



### **Prova scritta a fine corso.**

La prova sarà composta da domande a risposta chiusa, la cui valutazione potrà arrivare ad un massimo di 30.

# Informazioni amministrative - Modalità di Esame

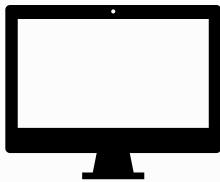
## Modalità di Esame

Come ogni esame, ci sarà un accertamento individuale della conoscenza dei contenuti trattati durante il corso. Questa avverrà in due momenti.



### **Prova scritta a fine corso.**

La prova sarà composta da domande a risposta chiusa, la cui valutazione potrà arrivare ad un massimo di 30.



### **Progetto di gruppo + Discussione del progetto a fine corso.**

Il progetto dovrà essere sviluppato con le metodologie/strumenti trattati a lezione, nonché rispettando i vincoli stabiliti - approfondimento a breve. La valutazione del progetto potrà arrivare ad un massimo di 30 e abiliterà l'eventuale lode.

# Informazioni amministrative - Modalità di Esame

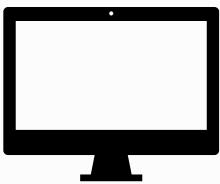
## Modalità di Esame

Come ogni esame, ci sarà un accertamento individuale della conoscenza dei contenuti trattati durante il corso. Questa avverrà in due momenti.



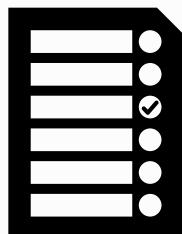
### **Prova scritta a fine corso.**

La prova sarà composta da domande a risposta chiusa, la cui valutazione potrà arrivare ad un massimo di 30.



### **Progetto di gruppo + Discussione del progetto a fine corso.**

Il progetto dovrà essere sviluppato con le metodologie/strumenti trattati a lezione, nonché rispettando i vincoli stabiliti - approfondimento a breve. La valutazione del progetto potrà arrivare ad un massimo di 30 e abiliterà l'eventuale lode.



### **Votazione finale.**

In linea generale, una media tra le valutazioni delle due prove. In casi eccezionali (ad esempio, adempimento di tutti i criteri di premialità nel progetto), il voto finale potrebbe essere incrementato fino a tre punti.

# Informazioni amministrative - Modalità di Esame

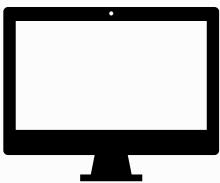
## Modalità di Esame

Come ogni esame, ci sarà un accertamento individuale della conoscenza dei contenuti trattati durante il corso. Questa avverrà in due momenti.



### Prova scritta a fine corso.

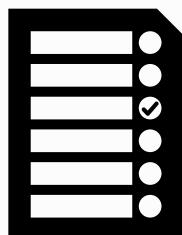
La prova sarà composta da domande a risposta chiusa, la cui valutazione potrà arrivare ad un massimo di 30.



### Progetto di gruppo + Discussione del progetto a fine corso.

Il progetto dovrà essere sviluppato con le metodologie/strumenti trattati a lezione, nonché rispettando i vincoli stabiliti - approfondimento a breve. La valutazione del progetto potrà arrivare ad un massimo di 30 e abiliterà l'eventuale lode.

---



### Votazione finale.

In linea generale, una media tra le valutazioni delle due prove. In casi eccezionali (ad esempio, adempimento di tutti i criteri di premialità nel progetto), il voto finale potrebbe essere incrementato fino a tre punti.

### Studenti impossibilitati a fare il progetto di gruppo.

Per essere esonerati è necessaria una motivazione valida (es: studenti lavoratori). Se esonerati, l'esame comprenderà solamente la prova scritta, ma la votazione massima sarà 24. Il modulo di esonero è disponibile sulla piattaforma e-learning del corso.

## Modalità di Esame - Alcuni Chiarimenti sui Progetti

- Team di 3/4 persone;
- Proposta di progetto presentata dal team;

## Modalità di Esame - Alcuni Chiarimenti sui Progetti

- Team di 3/4 persone;
- Proposta di progetto presentata dal team;

## Modalità di Esame - Alcuni Chiarimenti sui Progetti

### 1 *Greenfield Engineering*

- Focus su **progettazione** e testing;
- Dalle esigenze, si definiscono i requisiti, la progettazione, lo sviluppo, il testing.

## Modalità di Esame - Alcuni Chiarimenti sui Progetti

- Team di 3/4 persone;
- Proposta di progetto presentata dal team;

## Modalità di Esame - Alcuni Chiarimenti sui Progetti; Tipologie in base al focus del progetto

**1**

*Greenfield Engineering*

- Focus su **progettazione** e testing;
- Dalle esigenze, si definiscono i requisiti, la progettazione, lo sviluppo, il testing.

**2**

*Open-Source*

- Focus su **testing** e progettazione;
- Da un progetto open-source esistente, l'obiettivo è individuare failure.

## Modalità di Esame - Alcuni Chiarimenti sui Progetti

- Team di 3/4 persone;
- Proposta di progetto presentata dal team;

## Modalità di Esame - Alcuni Chiarimenti sui Progetti

**1**

### *Greenfield Engineering*

- Focus su **progettazione** e testing;
- Dalle esigenze, si definiscono i requisiti, la progettazione, lo sviluppo, il testing.

**2**

### *Open Source*

- Focus su **testing** e progettazione;
- Da un progetto open-source esistente, l'obiettivo è individuare failure.

Sebbene il focus sia sul testing, sarà comunque necessario documentare parte del progetto.

## Modalità di Esame - Alcuni Chiarimenti sui Progetti; Ulteriore possibilità

Se la tipologia di progetto selezionata è di greenfield engineering, è possibile realizzare **combinare** i progetti di Ingegneria del Software con quelli di Fondamenti di Intelligenza Artificiale.

## Modalità di Esame - Alcuni Chiarimenti sui Progetti; Ulteriore possibilità

Se la tipologia di progetto selezionata è di greenfield engineering, è possibile realizzare **combinare** i progetti di Ingegneria del Software con quelli di Fondamenti di Intelligenza Artificiale.

In tal caso, il progetto dovrà integrare una componente “intelligente” come richiesto dall’insegnamento di Fondamenti di Intelligenza Artificiale. Maggiori dettagli saranno forniti durante la prima lezione del corso di FIA.

## Modalità di Esame - Alcuni Chiarimenti sui Progetti; Ulteriore possibilità

Se la tipologia di progetto selezionata è di greenfield engineering, è possibile realizzare **combinare** i progetti di Ingegneria del Software con quelli di Fondamenti di Intelligenza Artificiale.

In tal caso, il progetto dovrà integrare una componente “intelligente” come richiesto dall’insegnamento di Fondamenti di Intelligenza Artificiale. Maggiori dettagli saranno forniti durante la prima lezione del corso di FIA.

NB: **Non** è indispensabile che tutti i membri di un team di IS facciano FIA. Tuttavia, è chiaro che della componente intelligente dovranno occuparsene gli studenti di FIA.

## Modalità di Esame - Alcuni Chiarimenti sui Progetti; Ulteriore possibilità

Se la tipologia di progetto selezionata è di greenfield engineering, è possibile realizzare **combinare** i progetti di Ingegneria del Software con quelli di Fondamenti di Intelligenza Artificiale.

In tal caso, il progetto dovrà integrare una componente “intelligente” come richiesto dall’insegnamento di Fondamenti di Intelligenza Artificiale. Maggiori dettagli saranno forniti durante la prima lezione del corso di FIA.

NB: **Non** è indispensabile che tutti i membri di un team di IS facciano FIA. Tuttavia, è chiaro che della componente intelligente dovranno occuparsene gli studenti di FIA.

## Modalità di Esame - Alcuni Chiarimenti sui Progetti; Scadenze

### Scadenze:

- I step: Formazione gruppi e inizio proposte di progetto.
- II step:
  - Kick-off meeting;
  - Gruppi illustrano la proposta di progetto ai docenti.

9/10

16/10

### Scadenze consegne intermedie/definitive:

- Inizio Dicembre: Coloro che vogliono, possono consegnare quanto fatto fino al SDD.
- Le altre scadenze saranno in concomitanza con gli appelli di esame.
- Chiaramente, chi consegna a Dicembre dovrà presentare esclusivamente la seconda parte.

### Studenti con DSA:

- Fanno pervenire prima possibile ai docenti le indicazioni da parte dell’ufficio diritto allo studio.
- Si individueranno le modalità più opportune.

## Vincoli progettuali

Come già detto in precedenza, i vincoli progettuali hanno subito cambiamenti sostanziali al fine di implementare un'idea di base: **è necessario fare meno, ma meglio!** O, in altri termini: **meno quantità ma più qualità!**

## Vincoli progettuali

Come già detto in precedenza, i vincoli progettuali hanno subito cambiamenti sostanziali al fine di implementare un'idea di base: **è necessario fare meno, ma meglio!** O, in altri termini: **meno quantità ma più qualità!**

### Ingegneria del Software: Perché?

In altri termini, lo sviluppo software è principalmente uno *sforzo collaborativo e comunicativo*, all'interno del quale la *progettazione* consente di raggiungere gli obiettivi prestabiliti con il committente.

Da qui il termine **Ingegneria**: *"Insieme di studi e tecniche che utilizzano le conoscenze delle varie branche delle scienze, unite a quelle tecnologiche, per risolvere problemi applicativi e per progettare e realizzare opere di diversa natura"* [Treccani]

Già nel corso di questa lezione, approfondiremo come la definizione viene mappata al software e quali sono le principali problematiche che rendono un processo ingegneristico inevitabile.

In questo corso, apprenderete le implicazioni di queste problematiche e come gestirle - ma fortunatamente, lo apprenderete in un ambiente "controllato" in cui le conseguenze non sono catastrofiche né irreparabili!

Apprenderete che le capacità tecniche, da sole, contano ben poco e che lo sviluppo software ha componenti e dinamiche umane, sociali ed organizzative paragonabili alla costruzione di qualunque altro prodotto architettonico ed ingegneristico - pertanto, richiede disciplina.

Apprenderete infine che la progettazione di cui parleremo non è niente di più che la naturale evoluzione di ciò che avete già parlato nei precedenti corsi.

In che senso? Facciamo un esempio...

### In che modo?

Simulando più fedelmente un caso reale. Ripartiamo dall'inizio: lo sviluppo software è **principalmente** uno sforzo collaborativo e comunicativo, **all'interno del quale** la progettazione consente di raggiungere gli obiettivi prestabiliti con il committente.

## Vincoli progettuali

Come già detto in precedenza, i vincoli progettuali hanno subito cambiamenti sostanziali al fine di implementare un'idea di base: **è necessario fare meno, ma meglio!** O, in altri termini: **meno quantità ma più qualità!**

### Ingegneria del Software: Perché?

In altri termini, lo sviluppo software è principalmente uno sforzo collaborativo e comunicativo, all'interno del quale la progettazione consente di raggiungere gli obiettivi prestabiliti con il committente.

Da qui il termine **Ingegneria**: "Insieme di studi e tecniche che utilizzano le conoscenze delle varie branche delle scienze, unite a quelle tecnologiche, per risolvere problemi applicativi e per progettare e realizzare opere di diversa natura" [Treccani]

Già nel corso di questa lezione, approfondiremo come la definizione viene mappata al software e quali sono le principali problematiche che rendono un processo ingegneristico inevitabile.

In questo corso, apprenderete le implicazioni di queste problematiche e come gestirle - ma fortunatamente, lo apprenderete in un ambiente "controllato" in cui le conseguenze non sono catastrofiche né irreparabili.

Apprenderete che le capacità tecniche, da sole, contano ben poco e che lo sviluppo software ha componenti e dinamiche umane, sociali ed organizzative paragonabili alla costruzione di qualunque altro prodotto architettonico ed ingegneristico - pertanto, richiede disciplina.

Apprenderete infine che la progettazione di cui parleremo non è niente di più che la naturale evoluzione di ciò che avete già parlato nei precedenti corsi.

In che senso? Facciamo un esempio...

### In che modo?

Simulando più fedelmente un caso reale. Ripartiamo dall'inizio: lo sviluppo software è **principalmente** uno sforzo collaborativo e comunicativo, **all'interno del quale** la progettazione consente di raggiungere gli obiettivi prestabiliti con il committente.

Quindi, l'effort sarà più orientato agli aspetti collaborativi e comunicativi piuttosto che all'aspetto tecnico.

## Vincoli progettuali

Come già detto in precedenza, i vincoli progettuali hanno subito cambiamenti sostanziali al fine di implementare un'idea di base: **è necessario fare meno, ma meglio!** O, in altri termini: **meno quantità ma più qualità!**

### Ingegneria del Software: Perché?

In altri termini, lo sviluppo software è principalmente uno *sforzo collaborativo e comunicativo*, all'interno del quale la *progettazione* consente di raggiungere gli obiettivi prestabiliti con il committente.

Da qui il termine **Ingegneria**: *"Insieme di studi e tecniche che utilizzano le conoscenze delle varie branche delle scienze, unite a quelle tecnologiche, per risolvere problemi applicativi e per progettare e realizzare opere di diversa natura"* [Treccani]

Già nel corso di questa lezione, approfondiremo come la definizione viene mappata al software e quali sono le principali problematiche che rendono un processo ingegneristico inevitabile.

In questo corso, apprenderete le implicazioni di queste problematiche e come gestirle - ma fortunatamente, lo apprenderete in un ambiente "controllato" in cui le conseguenze non sono catastrofiche né irreparabili.

Apprenderete che le capacità tecniche, da sole, contano ben poco e che lo sviluppo software ha componenti e dinamiche umane, sociali ed organizzative paragonabili alla costruzione di qualunque altro prodotto architettonico ed ingegneristico - pertanto, richiede disciplina.

Apprenderete infine che la progettazione di cui parleremo non è niente di più che la naturale evoluzione di ciò che avete già parlato nei precedenti corsi.

In che senso? Facciamo un esempio...

### *In che modo?*

Simulando più fedelmente un caso reale. Ripartiamo dall'inizio: lo sviluppo software è **principalmente** uno sforzo collaborativo e comunicativo, **all'interno del quale** la progettazione consente di raggiungere gli obiettivi prestabiliti con il committente.

Quindi, l'effort sarà più orientato agli aspetti collaborativi e comunicativi piuttosto che all'aspetto tecnico.

Questo implica nuovi vincoli - il ché rende i progetti completamente diversi dagli anni scorsi.

## Vincoli progettuali

Come già detto in precedenza, i vincoli progettuali hanno subito cambiamenti sostanziali al fine di implementare un'idea di base: **è necessario fare meno, ma meglio!** O, in altri termini: **meno quantità ma più qualità!**

### Ingegneria del Software: Perché?

In altri termini, lo sviluppo software è principalmente uno sforzo collaborativo e comunicativo, all'interno del quale la progettazione consente di raggiungere gli obiettivi prestabiliti con il committente.

Da qui il termine **Ingegneria**: "Insieme di studi e tecniche che utilizzano le conoscenze delle varie branche delle scienze, unite a quelle tecnologiche, per risolvere problemi applicativi e per progettare e realizzare opere di diversa natura" [Treccani]

Già nel corso di questa lezione, approfondiremo come la definizione viene mappata al software e quali sono le principali problematiche che rendono un processo ingegneristico inevitabile.

In questo corso, apprenderete le implicazioni di queste problematiche e come gestirle - ma fortunatamente, lo apprenderete in un ambiente "controllato" in cui le conseguenze non sono catastrofiche né irreparabili.

Apprenderete che le capacità tecniche, da sole, contano ben poco e che lo sviluppo software ha componenti e dinamiche umane, sociali ed organizzative paragonabili alla costruzione di qualunque altro prodotto architettonico ed ingegneristico - pertanto, richiede disciplina.

Apprenderete infine che la progettazione di cui parleremo non è niente di più che la naturale evoluzione di ciò che avete già parlato nei precedenti corsi.

In che senso? Facciamo un esempio...

### In che modo?

Simulando più fedelmente un caso reale. Ripartiamo dall'inizio: lo sviluppo software è **principalmente** uno sforzo collaborativo e comunicativo, **all'interno del quale** la progettazione consente di raggiungere gli obiettivi prestabiliti con il committente.

Quindi, l'effort sarà più orientato agli aspetti collaborativi e comunicativi piuttosto che all'aspetto tecnico.

Questo implica nuovi vincoli - il ché rende i progetti completamente diversi dagli anni scorsi.

Approfondiamo. Innanzitutto, avremo:

- **Vincoli collaborativi e comunicativi.** Limitazioni su metodi e strumenti di comunicazione e collaborazione da utilizzare nel progetto.

## Vincoli progettuali

Come già detto in precedenza, i vincoli progettuali hanno subito cambiamenti sostanziali al fine di implementare un'idea di base: **è necessario fare meno, ma meglio!** O, in altri termini: **meno quantità ma più qualità!**

### Ingegneria del Software: Perché?

In altri termini, lo sviluppo software è principalmente uno *sforzo collaborativo e comunicativo*, all'interno del quale la *progettazione* consente di raggiungere gli obiettivi prestabiliti con il committente.

Da qui il termine **Ingegneria**: “*Insieme di studi e tecniche che utilizzano le conoscenze delle varie branche delle scienze, unite a quelle tecnologiche, per risolvere problemi applicativi e per progettare e realizzare opere di diversa natura*” [Treccani]

Già nel corso di questa lezione, approfondiremo come la definizione viene mappata al software e quali sono le principali problematiche che rendono un processo ingegneristico inevitabile.

In questo corso, apprenderete le implicazioni di queste problematiche e come gestirle - ma fortunatamente, lo apprenderete in un ambiente “controllato” in cui le conseguenze non sono catastrofiche né irreparabili.

Apprenderete che le capacità tecniche, da sole, contano ben poco e che lo sviluppo software ha componenti e dinamiche umane, sociali ed organizzative paragonabili alla costruzione di qualunque altro prodotto architettonico ed ingegneristico - pertanto, richiede disciplina.

Apprenderete infine che la progettazione di cui parleremo non è niente di più che la naturale evoluzione di ciò che avete già parlato nei precedenti corsi.

In che senso? Facciamo un esempio...

### In che modo?

Simulando più fedelmente un caso reale. Ripartiamo dall'inizio: lo sviluppo software è **principalmente** uno sforzo collaborativo e comunicativo, **all'interno del quale** la progettazione consente di raggiungere gli obiettivi prestabiliti con il committente.

Quindi, l'effort sarà più orientato agli aspetti collaborativi e comunicativi piuttosto che all'aspetto tecnico.

Questo implica nuovi vincoli - il ché rende i progetti completamente diversi dagli anni scorsi.

Approfondiamo. Innanzitutto, avremo:

- **Vincoli collaborativi e comunicativi.** Limitazioni su metodi e strumenti di comunicazione e collaborazione da utilizzare nel progetto.
- **Vincoli tecnici.** Limitazioni sugli artefatti da realizzare.

## Vincoli progettuali

Come già detto in precedenza, i vincoli progettuali hanno subito cambiamenti sostanziali al fine di implementare un'idea di base: **è necessario fare meno, ma meglio!** O, in altri termini: **meno quantità ma più qualità!**

### Ingegneria del Software: Perché?

In altri termini, lo sviluppo software è principalmente uno *sforzo collaborativo e comunicativo*, all'interno del quale la *progettazione* consente di raggiungere gli obiettivi prestabiliti con il committente.

Da qui il termine **Ingegneria**: *"Insieme di studi e tecniche che utilizzano le conoscenze delle varie branche delle scienze, unite a quelle tecnologiche, per risolvere problemi applicativi e per progettare e realizzare opere di diversa natura"* [Treccani]

Già nel corso di questa lezione, approfondiremo come la definizione viene mappata al software e quali sono le principali problematiche che rendono un processo ingegneristico inevitabile.

In questo corso, apprenderete le implicazioni di queste problematiche e come gestirle - ma fortunatamente, lo apprenderete in un ambiente "controllato" in cui le conseguenze non sono catastrofiche né irreparabili.

Apprenderete che le capacità tecniche, da sole, contano ben poco e che lo sviluppo software ha componenti e dinamiche umane, sociali ed organizzative paragonabili alla costruzione di qualunque altro prodotto architettonico ed ingegneristico - pertanto, richiede disciplina.

Apprenderete infine che la progettazione di cui parleremo non è niente di più che la naturale evoluzione di ciò che avete già parlato nei precedenti corsi.

In che senso? Facciamo un esempio...

### In che modo?

Simulando più fedelmente un caso reale. Ripartiamo dall'inizio: lo sviluppo software è **principalmente** uno sforzo collaborativo e comunicativo, **all'interno del quale** la progettazione consente di raggiungere gli obiettivi prestabiliti con il committente.

Quindi, l'effort sarà più orientato agli aspetti collaborativi e comunicativi piuttosto che all'aspetto tecnico.

Questo implica nuovi vincoli - il ché rende i progetti completamente diversi dagli anni scorsi.

Approfondiamo. Innanzitutto, avremo:

- **Vincoli collaborativi e comunicativi.** Limitazioni su metodi e strumenti di comunicazione e collaborazione da utilizzare nel progetto.
- **Vincoli tecnici.** Limitazioni sugli artefatti da realizzare.
- **Criteri di accettazione.** Criteri che, se non rispettati, portano al fallimento del progetto.

## Vincoli progettuali

Come già detto in precedenza, i vincoli progettuali hanno subito cambiamenti sostanziali al fine di implementare un'idea di base: **è necessario fare meno, ma meglio!** O, in altri termini: **meno quantità ma più qualità!**

### Ingegneria del Software: Perché?

In altri termini, lo sviluppo software è principalmente uno *sforzo collaborativo e comunicativo*, all'interno del quale la *progettazione* consente di raggiungere gli obiettivi prestabiliti con il committente.

Da qui il termine **Ingegneria**: *"Insieme di studi e tecniche che utilizzano le conoscenze delle varie branche delle scienze, unite a quelle tecnologiche, per risolvere problemi applicativi e per progettare e realizzare opere di diversa natura"* [Treccani]

Già nel corso di questa lezione, approfondiremo come la definizione viene mappata al software e quali sono le principali problematiche che rendono un processo ingegneristico inevitabile.

In questo corso, apprenderete le implicazioni di queste problematiche e come gestirle - ma fortunatamente, lo apprenderete in un ambiente "controllato" in cui le conseguenze non sono catastrofiche né irreparabili.

Apprenderete che le capacità tecniche, da sole, contano ben poco e che lo sviluppo software ha componenti e dinamiche umane, sociali ed organizzative paragonabili alla costruzione di qualunque altro prodotto architettonico ed ingegneristico - pertanto, richiede disciplina.

Apprenderete infine che la progettazione di cui parleremo non è niente di più che la naturale evoluzione di ciò che avete già parlato nei precedenti corsi.

In che senso? Facciamo un esempio...

### In che modo?

Simulando più fedelmente un caso reale. Ripartiamo dall'inizio: lo sviluppo software è **principalmente** uno sforzo collaborativo e comunicativo, **all'interno del quale** la progettazione consente di raggiungere gli obiettivi prestabiliti con il committente.

Quindi, l'effort sarà più orientato agli aspetti collaborativi e comunicativi piuttosto che all'aspetto tecnico.

Questo implica nuovi vincoli - il ché rende i progetti completamente diversi dagli anni scorsi.

Approfondiamo. Innanzitutto, avremo:

- **Vincoli collaborativi e comunicativi.** Limitazioni su metodi e strumenti di comunicazione e collaborazione da utilizzare nel progetto.
- **Vincoli tecnici.** Limitazioni sugli artefatti da realizzare.
- **Criteri di accettazione.** Criteri che, se non rispettati, portano al fallimento del progetto.
- **Criteri di premialità.** Criteri che, se rispettati, portano a punti aggiuntivi al progetto.

## Vincoli progettuali

**Vincoli collaborativi e comunicativi.** Limitazioni su metodi e strumenti di comunicazione e collaborazione da utilizzare nel progetto.

- Rispetto delle scadenze intermedie/di fine progetto **definite nello statement of work.**
- Budget effort **non superiore** a 50 ore per ogni membro del team.
- Uso di sistemi di versioning - **GitHub** in particolare.
- Uso di tool per la gestione di task e attività - **Trello** o simili.
- Uso di un tool di comunicazione tracciabile - **Slack** in particolare.

Già nei primi due laboratori, parleremo dei sistemi di versioning e di sistemi per la gestione di task/attività e per la comunicazione, in modo da agevolare il vostro lavoro.

## Vincoli progettuali

**Vincoli collaborativi e comunicativi.** Limitazioni su metodi e strumenti di comunicazione e collaborazione da utilizzare nel progetto.

- Rispetto delle scadenze intermedie/di fine progetto **definite nello statement of work.**
- Budget effort **non superiore** a 50 ore per ogni membro del team.
- Uso di sistemi di versioning - **GitHub** in particolare.
- Uso di tool per la gestione di task e attività - **Trello** o simili.
- Uso di un tool di comunicazione tracciabile - **Slack** in particolare.

Già nei primi due laboratori, parleremo dei sistemi di versioning e di sistemi per la gestione di task/attività e per la comunicazione, in modo da agevolare il vostro lavoro.

**NB:** Patti chiari, amicizia lunga! Il mancato uso di GitHub, Trello e/o Slack porta **automaticamente** al fallimento del progetto —> il progetto non verrà accettato né discusso; nessuna eccezione.

## Vincoli progettuali

**Vincoli collaborativi e comunicativi.** Limitazioni su metodi e strumenti di comunicazione e collaborazione da utilizzare nel progetto.

- Rispetto delle scadenze intermedie/di fine progetto **definite nello statement of work.**
- Budget effort **non superiore** a 50 ore per ogni membro del team.
- Uso di sistemi di versioning - **GitHub** in particolare.
- Uso di tool per la gestione di task e attività - **Trello** o simili.
- Uso di un tool di comunicazione tracciabile - **Slack** in particolare.

Già nei primi due laboratori, parleremo dei sistemi di versioning e di sistemi per la gestione di task/attività e per la comunicazione, in modo da agevolare il vostro lavoro.

**NB:** Patti chiari, amicizia lunga! Il mancato uso di GitHub, Trello e/o Slack porta **automaticamente** al fallimento del progetto —> il progetto non verrà accettato né discusso; nessuna eccezione.

Chiaramente, non basta aver installato, ad esempio, Slack per considerare il vincolo rispettato, ma su questo ci arriveremo tra poco...

## Vincoli progettuali

**Vincoli collaborativi e comunicativi.** Limitazioni su metodi e strumenti di comunicazione e collaborazione da utilizzare nel progetto.

- Rispetto delle scadenze intermedie/di fine progetto **definite nello statement of work.**
- Budget effort **non superiore** a 50 ore per ogni membro del team.
- Uso di sistemi di versioning - **GitHub** in particolare.
- Uso di tool per la gestione di task e attività - **Trello** o simili.
- Uso di un tool di comunicazione tracciabile - **Slack** in particolare.

Già nei primi due laboratori, parleremo dei sistemi di versioning e di sistemi per la gestione di task/attività e per la comunicazione, in modo da agevolare il vostro lavoro.

**NB:** Patti chiari, amicizia lunga! Il mancato uso di GitHub, Trello e/o Slack porta **automaticamente** al fallimento del progetto —> il progetto non verrà accettato né discusso; nessuna eccezione.

Chiaramente, non basta aver installato, ad esempio, Slack per considerare il vincolo rispettato, ma su questo ci arriveremo tra poco...

## Vincoli progettuali

**Vincoli tecnici (1)**. Limitazioni sugli artefatti da realizzare.

*Analisi e Specifica dei Requisiti.*

- Specifica di **minimo** 2 e **massimo** 4 scenari per ogni membro del team;
- Specifica di **minimo** 2 e **massimo** 4 requisiti funzionali e non funzionali per ogni membro del team;
- **Esattamente** uno use case per ogni membro del team - i casi d'uso aggiuntivi **non** saranno valutati;
- **Esattamente** un sequence diagram ogni due membri del team - i sequence diagram aggiuntivi **non** saranno valutati;
- **Esattamente** un diagramma a scelta tra statechart e activity diagram ogni due membri del team - ulteriori diagrammi **non** verranno valutati;
- Specifica di un class diagram per team - eventuali object diagram **non** verranno valutati.

## Vincoli progettuali

**Vincoli tecnici (1).** Limitazioni sugli artefatti da realizzare.

*Analisi e Specifica dei Requisiti.*

- Specifica di **minimo** 2 e **massimo** 4 scenari per ogni membro del team;
- Specifica di **minimo** 2 e **massimo** 4 requisiti funzionali e non funzionali per ogni membro del team;
- **Esattamente** uno use case per ogni membro del team - i casi d'uso aggiuntivi **non** saranno valutati;
- **Esattamente** un sequence diagram ogni due membri del team - i sequence diagram aggiuntivi **non** saranno valutati;
- **Esattamente** un diagramma a scelta tra statechart e activity diagram ogni due membri del team - ulteriori diagrammi **non** verranno valutati;
- Specifica di un class diagram per team - eventuali object diagram **non** verranno valutati.

**NB:** L'analisi dei requisiti è storicamente la parte in cui gli studenti hanno maggiore difficoltà, oltre ad essere il momento in cui gli artefatti vengono modificati più spesso. Quindi, **non** strafate ed **attenetevi** ai vincoli! Se fate di più, non rispetterete i vincoli e, pertanto, **questo porterà ad avere un punteggio inferiore.**

## Vincoli progettuali

**Vincoli tecnici (1).** Limitazioni sugli artefatti da realizzare.

*Analisi e Specifica dei Requisiti.*

- Specifica di **minimo** 2 e **massimo** 4 scenari per ogni membro del team;
- Specifica di **minimo** 2 e **massimo** 4 requisiti funzionali e non funzionali per ogni membro del team;
- **Esattamente** uno use case per ogni membro del team - i casi d'uso aggiuntivi **non** saranno valutati;
- **Esattamente** un sequence diagram ogni due membri del team - i sequence diagram aggiuntivi **non** saranno valutati;
- **Esattamente** un diagramma a scelta tra statechart e activity diagram ogni due membri del team - ulteriori diagrammi **non** verranno valutati;
- Specifica di un class diagram per team - eventuali object diagram **non** verranno valutati.

**NB:** L'analisi dei requisiti è storicamente la parte in cui gli studenti hanno maggiore difficoltà, oltre ad essere il momento in cui gli artefatti vengono modificati più spesso. Quindi, **non** strafate ed **attenetevi** ai vincoli! Se fate di più, non rispetterete i vincoli e, pertanto, **questo porterà ad avere un punteggio inferiore.**

## Vincoli progettuali

**Vincoli tecnici (2)**. Limitazioni sugli artefatti da realizzare.

### System Design.

- Specifica di **minimo 2 e massimo 4 design goal** per ogni membro del team.
- Definizione di **un diagramma** di decomposizione dei sottosistemi per team, con annessa descrizione e motivazione all'uso.
- Definizione di **un deployment diagram** per team, con annessa descrizione e motivazione all'uso.

## Vincoli progettuali

**Vincoli tecnici (2)**. Limitazioni sugli artefatti da realizzare.

### System Design.

- Specifica di **minimo** 2 e **massimo** 4 design goal per ogni membro del team.
- Definizione di **un diagramma** di decomposizione dei sottosistemi per team, con annessa descrizione e motivazione all'uso.
- Definizione di **un deployment diagram** per team, con annessa descrizione e motivazione all'uso.

### Object Design.

- Uso di **minimo** uno e **massimo** due design pattern per team;
- Uso di UML;

## Vincoli progettuali

**Vincoli tecnici (2)**. Limitazioni sugli artefatti da realizzare.

### System Design.

- Specifica di **minimo** 2 e **massimo** 4 design goal per ogni membro del team.
- Definizione di **un diagramma** di decomposizione dei sottosistemi per team, con annessa descrizione e motivazione all'uso.
- Definizione di **un deployment diagram** per team, con annessa descrizione e motivazione all'uso.

### Object Design.

- Uso di **minimo** uno e **massimo** due design pattern per team;
- Uso di UML;

### Testing.

- Ogni studente dovrà effettuare il testing di unità, tramite category partition, di **esattamente** un metodo di una classe sviluppata.
- Ogni studente dovrà effettuare il testing di sistema, tramite category partition, di **esattamente** una funzionalità del sistema sviluppato.

## Vincoli progettuali

**Vincoli tecnici (2)**. Limitazioni sugli artefatti da realizzare.

### System Design.

- Specifica di **minimo** 2 e **massimo** 4 design goal per ogni membro del team.
- Definizione di **un diagramma** di decomposizione dei sottosistemi per team, con annessa descrizione e motivazione all'uso.
- Definizione di **un deployment diagram** per team, con annessa descrizione e motivazione all'uso.

### Object Design.

- Uso di **minimo** uno e **massimo** due design pattern per team;
- Uso di UML;

### Testing.

- Ogni studente dovrà effettuare il testing di unità, tramite category partition, di **esattamente** un metodo di una classe sviluppata.
- Ogni studente dovrà effettuare il testing di sistema, tramite category partition, di **esattamente** una funzionalità del sistema sviluppato.

**NB:** Sebbene il carico di lavoro richiesto a valle dell'analisi e specifica dei requisiti è stato notevolmente ridotto, sarà necessario comunque sviluppare una documentazione **snella** delle attività condotte nel progetto.

## Vincoli progettuali

**Criteri di accettazione.** Criteri che, se non rispettati, portano al fallimento del progetto.

- Utilizzo appropriato di GitHub, che preveda il **rispetto delle linee guida definite nel contesto del primo lab.**
- Adeguato utilizzo del pull-based development, che preveda il **rispetto delle linee guida definite nel contesto del primo lab.**
- Adeguato utilizzo di Slack, che preveda il **rispetto delle linee guida definite nel contesto del secondo lab.**
- Adeguato utilizzo di Trello, che preveda il **rispetto delle linee guida definite nel contesto del secondo lab.**
- Documentazione adeguata. Verranno usati tool di **plagiarism detection** per identificare casi in cui gli studenti hanno copiato da progetti di anni precedenti e/o da altre fonti.
- Appropriato test di unità di un metodo sviluppato, che preveda il **rispetto dei vincoli.**
- Appropriato test di sistema di una funzionalità del sistema sviluppato, che preveda il **rispetto dei vincoli.**

## Vincoli progettuali

**Criteri di accettazione.** Criteri che, se non rispettati, portano al fallimento del progetto.

- Utilizzo appropriato di GitHub, che preveda il **rispetto delle linee guida definite nel contesto del primo lab.**
- Adeguato utilizzo del pull-based development, che preveda il **rispetto delle linee guida definite nel contesto del primo lab.**
- Adeguato utilizzo di Slack, che preveda il **rispetto delle linee guida definite nel contesto del secondo lab.**
- Adeguato utilizzo di Trello, che preveda il **rispetto delle linee guida definite nel contesto del secondo lab.**
- Documentazione adeguata. Verranno usati tool di **plagiarism detection** per identificare casi in cui gli studenti hanno copiato da progetti di anni precedenti e/o da altre fonti.
- Appropriato test di unità di un metodo sviluppato, che preveda il **rispetto dei vincoli.**
- Appropriato test di sistema di una funzionalità del sistema sviluppato, che preveda il **rispetto dei vincoli.**

**NB:** Sembrano criteri di difficile applicazione, ma non lo sono affatto. Bisogna diventare dei professionisti e sviluppare software in maniera adeguata: se seguirete il corso, imparerete presto che fare le cose per bene è sempre un guadagno!

## Vincoli progettuali

***Criteri di premialità.*** Criteri che, se rispettati, portano a punti aggiuntivi al progetto.

- Uso adeguato di **sistemi di build**;
- Uso adeguato di un processo di **continuous integration** tramite **Travis**;
- Uso adeguato di tool di controllo della qualità (ad esempio, **CheckStyle**);
- Adozione di **processi di code review**;
- Uso adeguato di tool avanzati di testing (e.g., **Mockito**, **Cobertura**, etc.).

## Vincoli progettuali

***Criteri di premialità.*** Criteri che, se rispettati, portano a punti aggiuntivi al progetto.

- Uso adeguato di **sistemi di build**;
- Uso adeguato di un processo di **continuous integration tramite Travis**;
- Uso adeguato di tool di controllo della qualità (ad esempio, **CheckStyle**);
- Adozione di **processi di code review**;
- Uso adeguato di tool avanzati di testing (e.g., **Mockito**, **Cobertura**, etc.).

**NB:** Visti così, anche questi sembrano criteri di difficile applicazione, ma ancora una volta: basta “solo” essere dei professionisti! Mettetevi alla prova!

## Vincoli progettuali

**Criteri di premialità.** Criteri che, se rispettati, portano a punti aggiuntivi al progetto.

- Uso adeguato di **sistemi di build**;
- Uso adeguato di un processo di **continuous integration tramite Travis**;
- Uso adeguato di tool di controllo della qualità (ad esempio, **CheckStyle**);
- Adozione di **processi di code review**;
- Uso adeguato di tool avanzati di testing (e.g., **Mockito**, **Cobertura**, etc.).

**NB:** Visti così, anche questi sembrano criteri di difficile applicazione, ma ancora una volta: basta “solo” essere dei professionisti! Mettetevi alla prova!

**Statement of Work.** La prima scadenza progettuale da rispettare (9/10) è quella della presentazione dello statement of work, il documento che lista la vostra proposta di progetto e dettaglia i vincoli progettuali.

**Questionario.** Entro il 28/09, si prega di compilare il seguente questionario:  
<https://tinyurl.com/t92pxrze>

**Carmine Gravino**  
SeSa Lab - University of Salerno

# Ingegneria del Software

