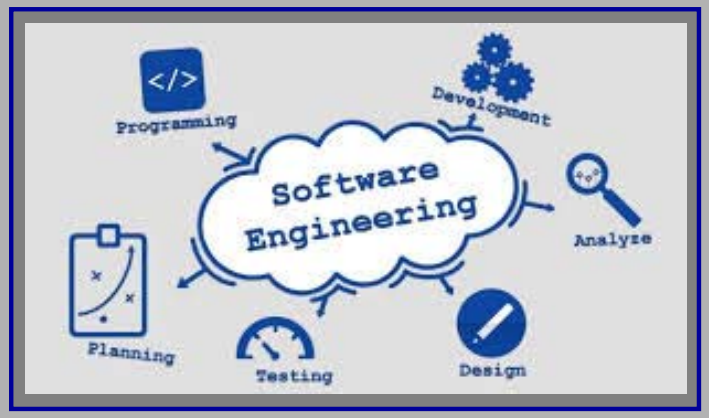
Introduzione



Anno accademico 2020/2021

Ingegneria del Software

Tullio Vardanega, tullio.vardanega@unipd.it

Cosa facciamo in questo corso – 1

- **Apprendiamo metodi e pratiche di lavoro alla base della professione informatica**
  - **Gestire il tempo**
    - Disponibilità, scadenze, conflitti, priorità
  - **Collaborare**
    - Fissare obiettivi, dividersi compiti, verificare progressi, riportare difficoltà
  - **Assumersi responsabilità**
    - Fare quanto pattuito, agire al meglio delle proprie capacità, auto-valutarsi prima di valutare
  - **Auto-apprendere**
    - "Imparare a imparare", essenziale competenza trasversale

- **Integriamo la teoria con la pratica**

Come vogliamo imparare

- La conoscenza passa dalla comprensione profonda, sperimentata, dei significati
  - Non ricordare, ma riconoscere (so chi sei …)
- Vogliamo fissare tali conoscenze in un glossario
  - Raccolta di termini/concetti centrali al dominio SWE e non auto-evidenti
  - Registrati in modo da facilitarne la localizzazione
  - Corredati dalla nostra personale specifica del loro significato e ogni altra informazione utile a riconoscerli
- Vogliamo raffinarne la comprensione, legando la teoria con la pratica

Come lo facciamo – 1

- Tramite un progetto didattico collaborativo
  - Promosso da un proponente esterno
  - Con esigenze e obiettivi funzionali innovativi
  - Complesso, impegnativo, visionario
  - Tecnologicamente avanzato
- Confermando le conoscenze acquisite tramite una prova scritta

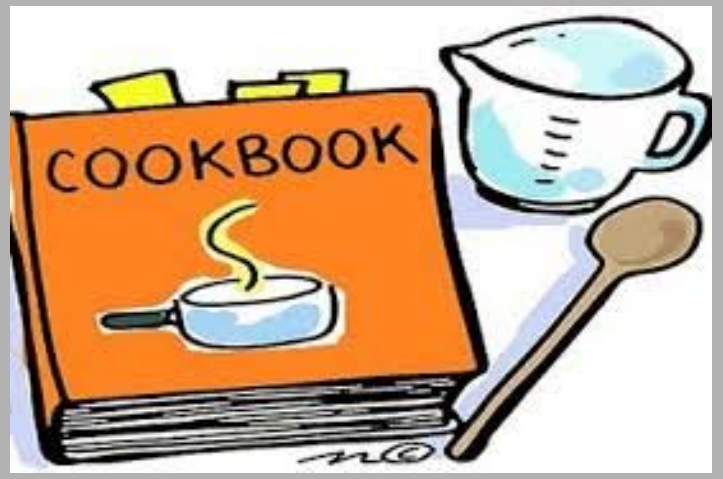
Glossario

Fonte: Harold Kerzner (1940-), uno dei maggiori esperti mondiali di *project management*

- **Progetto**
  - **Insieme di attività che**
    - Devono raggiungere determinati obiettivi a partire da determinate specifiche
    - Hanno una data d'inizio e una data di fine fissate
    - Dispongono di risorse limitate (persone, tempo, denaro, strumenti)
    - Consumano tali risorse nel loro svolgersi
  - **L'uscita di un progetto è un prodotto composito**
    - SW sorgente/eseguibile, librerie, documenti, manuali

Principali costituenti di un progetto

- **Pianificazione**
  - Gestire risorse (persone, tempo, denaro, strumenti) responsabilmente, in funzione degli obiettivi
- **Analisi dei requisiti**
  - Definire cosa bisogna fare
- **Progettazione (*design*)**
  - Definire come farlo
- **Realizzazione**
  - Farlo, perseguendo qualità
  - Accertando l'assenza di errori od omissioni
  - Accertando che i risultati soddisfino le attese



Cosa non è un progetto?

- Nella filosofia greca, arte (μίμησις) significa copia / riproduzione bella e significativa
  - Della natura tangibile o spirituale
- In latino, ars significa «abilità professionale»
  - Significato rimasto in vigore fino all'Illuminismo
- Con il Romanticismo, arte è divenuta «espressione di contenuto emozionante»
  - Ma Kant (1724-1804) riteneva l'arte giudicabile solo tramite qualità formali
- Un progetto non è arte romantica, ma kantiana

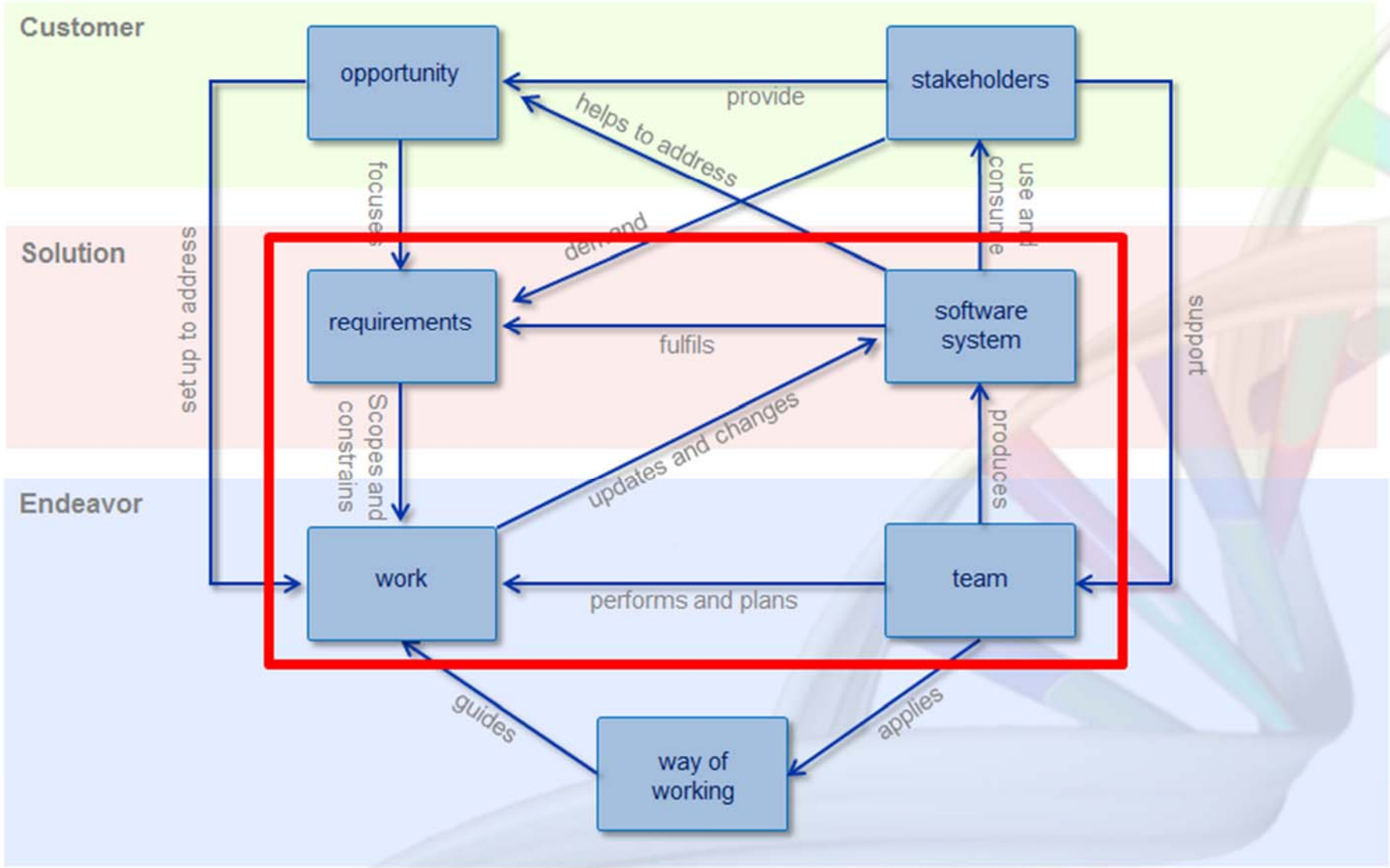
Cosa non è un progetto – 2

- ***One is blinded to the fundamental uselessness of their products, by the sense of achievement one feels in getting them to work at all***
- ***In other words, their fundamental design flaws are completely hidden by their superficial design flaws***

Fonte: Douglas Adams, "The Hitchhikers Guide to the Galaxy", 1979

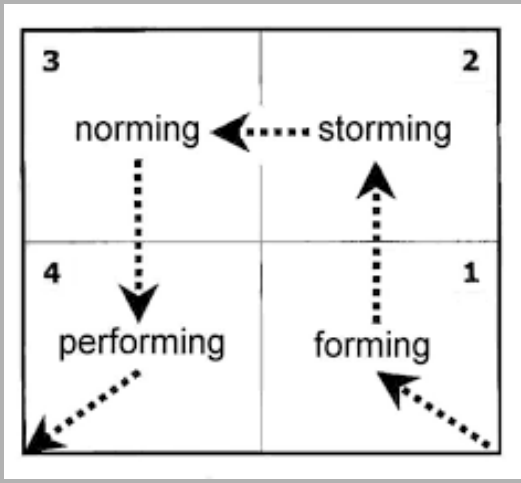
Gli elementi di un progetto

SEMAT
semat.org



Glossario

- ***Teamwork***
  - **Lavoro collaborativo che punta a raggiungere un obiettivo comune in modo efficace ed efficiente**
  - **I membri del *team* sono inter-dipendenti**
  - **La gestione di questa inter-dipendenza richiede il rispetto di regole e di buone pratiche**
    - Comunicazioni aperte e trasparenti: risoluzione dei conflitti
    - Costruzione e preservazione delle fiducia reciproca: condivisione e collaborazione
    - Assunzione di responsabilità: coordinamento
    - Condivisione dei rischi
  - **La sua base è un solido *way of working***



Glossario

- ***Stakeholder*** **(portatore di interesse)**
  - **Tutti coloro che a vario titolo hanno influenza sul prodotto e sul progetto**
    - La comunità degli utenti (che usa il prodotto)
    - Il committente (che compra il prodotto)
    - Il fornitore (che sostiene i costi di realizzazione)
    - Eventuali regolatori (che verificano la qualità del lavoro)
- ***Way of working***
  - **La maniera di organizzare al meglio le attività di progetto**

Ingredienti

- **Per svolgere un progetto potendo confidare nel suo successo serve ingegneria**

> *Engineering*: *application of scientific and mathematical principles to practical ends*
> **Fonte: American Heritage Dictionary**

  - **Applicazione (non creazione!) di principi noti e autorevoli: *best practice***
  - ***Practical ends* spesso civili e sociali, associati a responsabilità etiche e professionali**

Glossario

- ***Software engineering*** **[SWE]**
  - **Disciplina per la realizzazione di prodotti SW così impegnativi da richiedere il dispiego di attività collaborative**
  - **Capacità di produrre "in grande" e "in piccolo"**
  - **Garantendo qualità: efficacia**
  - **Contenendo il consumo di risorse: efficienza**
  - **Lungo l'intero periodo di sviluppo e di uso del prodotto: ciclo di vita**

Glossario

- **Efficacia**
  - Misura della capacità di raggiungere l'obiettivo prefissato
- **Efficienza**
  - Misura dell'abilità di raggiungere l'obiettivo impiegando le risorse minime indispensabili

Glossario

- **Ciclo di vita**
  - Gli stati che il prodotto SW richiesto assume dal suo concepimento al ritiro

- ***Best practice***
  - Modo di fare (*way of working*) noto, che abbia mostrato di garantire i migliori risultati in circostanze note e specifiche

SWE rispetto alle altre discipline

- ***SWE is <u>not</u> a branch of computer science; it is an engineering discipline that relies on computer science, in the same way that mechanical engineering relies on physics***
- ***In addition to computer science, the scientific disciplines of reference to SWE also include certain areas of discrete mathematics and operation research, statistics, psychology and economics***

Fonte: Lionel Briand, IEEE Software 49(4), 93-95

SWE rispetto a se stessa

- **Un sistema SW è tanto più utile quanto più è usato**
  - **Metrica: integrale del suo uso (o del suo #utenti) nel tempo**
- **Più lunga la vita d'uso di un prodotto, maggiore il suo costo di manutenzione**
  - **Manutenzione: insieme di attività necessarie a garantire l'uso continuativo del prodotto**
    - Reattivamente (per correzione dopo malfunzionamento) o preventivamente
- **Il costo di manutenzione ha varie componenti**
  - **Mancato guadagno, perdita di reputazione, recupero o reclutamento esperti, sottrazione di risorse ad altre attività**
- **I principi SWE puntano ad abbassare tali costi**
  - **Sviluppando SW più facilmente manutenibile**

Cos'è l'ingegneria del *software* – 1

- **Nasce nel 1968**
  - **Conferenza NATO 7-11/10/1968 @ Garmisch (D)**
- **Raccogliere, organizzare, consolidare la conoscenza (*body of knowledge*) necessaria a realizzare progetti SW con efficacia ed efficienza**
  - **Collezione e manutenzione migliorativa di *best practice***
- **Applicare principi ingegneristici calati nella produzione del SW**

Cos'è l'ingegneria del *software* – 2

L'approccio sistematico, disciplinato e quantificabile allo sviluppo, l'uso, la manutenzione e il ritiro del SW
Fonte: Glossario IEEE

- **Sistematico**
  - Modo di lavorare metodico e rigoroso
  - Che conosce, usa ed evolve le *best practice* di dominio
- **Disciplinato**
  - Che segue le regole che si è dato
- **Quantificabile**
  - Che permette di misurare l'efficienza e l'efficacia del suo agire

Figure professionali – 1

- ***Software engineer*** **≠ programmatore**
- **Il programmatore**
  - **Figura professionale dominante nei primi decenni dell'informatica**
  - **Scrive programmi da solo, sotto la propria responsabilità tecnica**
  - **Svolge un'attività creativa fortemente personalizzata (arte in senso romantico)**

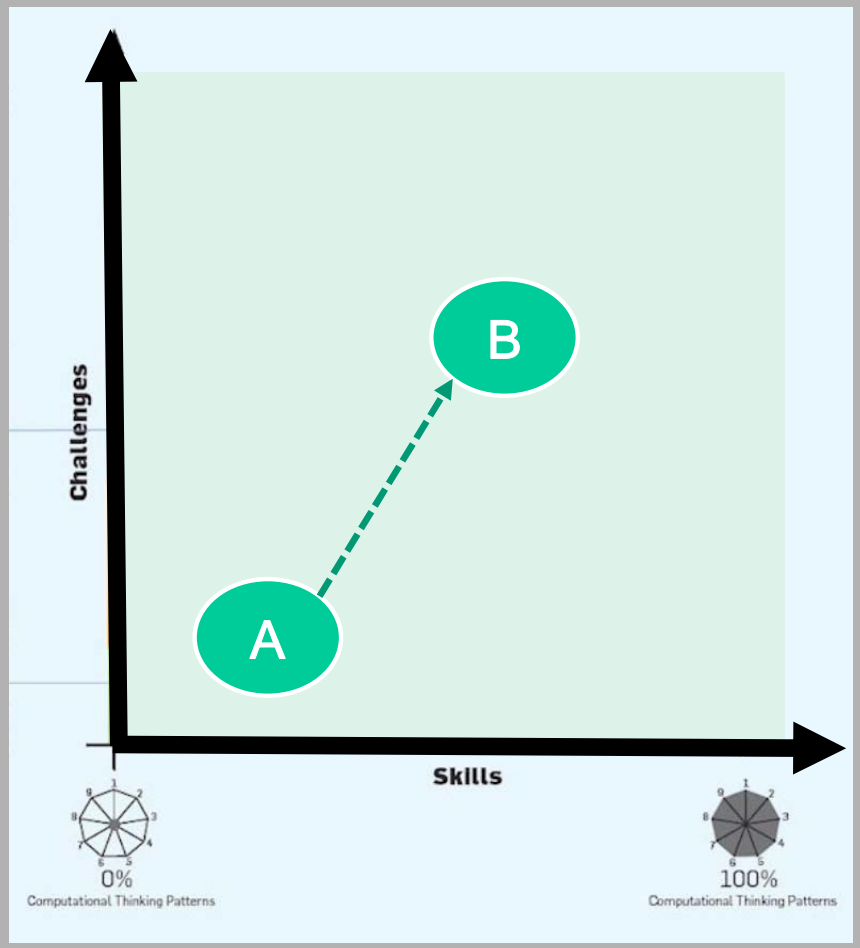
Figure professionali – 2

- **Il *software engineer***
  - **Realizza parte di un sistema complesso con la consapevolezza che potrà essere usato, completato e modificato da altri**
  - **Comprende il contesto in cui si colloca il sistema cui contribuisce**
    - La dimensione "sistema" include ma non si limita al SW
  - **Attua compromessi intelligenti e lungimiranti tra visioni e spinte contrapposte**
    - Costi – qualità
    - Risorse – disponibilità
    - Esperienza utente – facilità di realizzazione

Cosa facciamo in questo corso – 2

- Studiamo tutte le attività di progetto
- Proviamo a metterle in pratica
  - Nel progetto didattico
- Verifichiamo il grado di apprendimento
  - In itinere: tramite revisioni di avanzamento
  - In fine: tramite una prova scritta

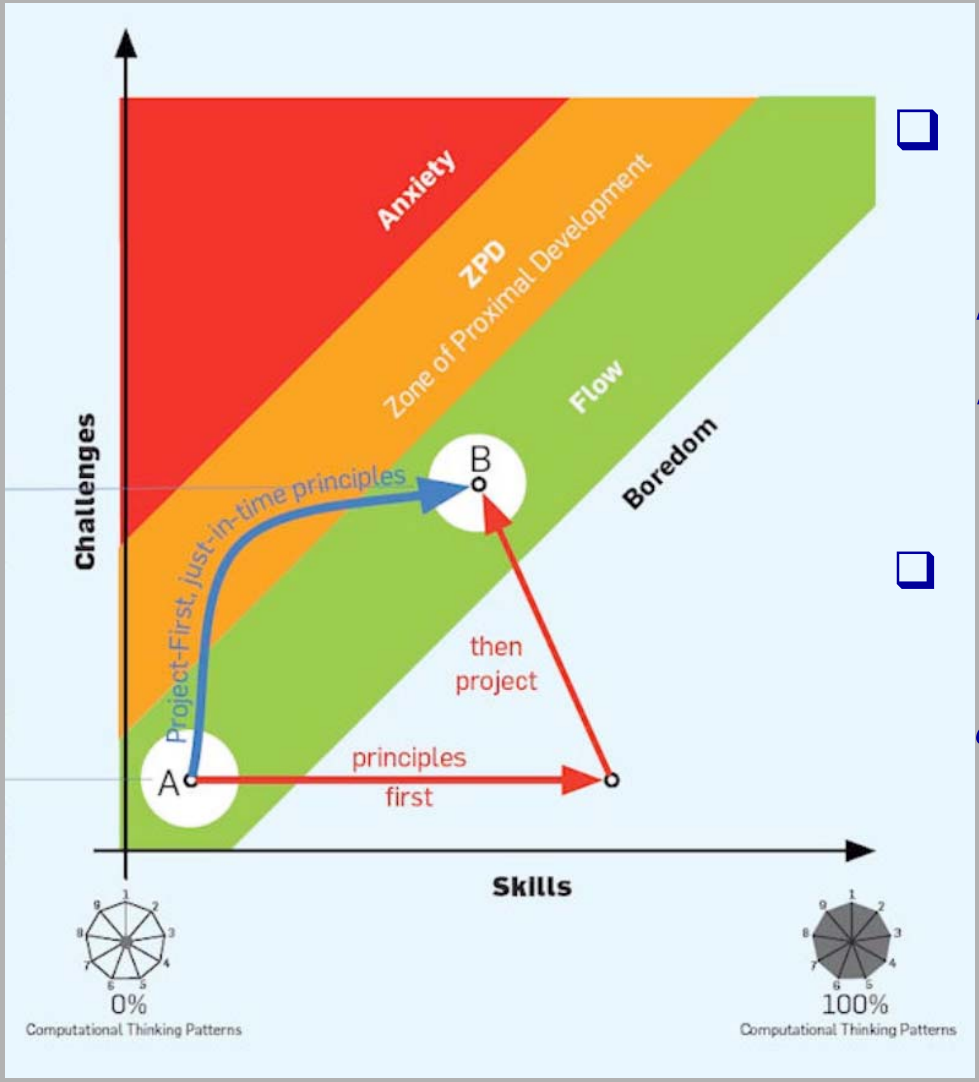
Perché lo facciamo così – 1



- ***Student acquisition of [methodical] skills advances in response to challenges***
- ***Pedagogical approaches can be described as instructional trajectories connecting a skill/challenge starting point (A) with a destination point (B)***

Fonte: *D.C. Webb, A. Repenning, K.H. Koh, "Toward an emergent theory of broadening participation in computer science education", Proc. 43rd ACM Computer Science Education symposium, 173-178 (SIGCSE '12)*

Perché lo facciamo così – 2



- ***The project-first just-in-time-principles approach lies in the Zone of Proximal Flow (ZPF)***
  - ***The ideal condition for learning***
- ***The ZPF orchestrates students' take in best practices with assistance and tool use***

Con quale quantità di impegno

- 13 crediti → 325 ore di lavoro complessivo
- 75 ore in lezioni, esercitazioni, seminari
  - 64 di esse nel I semestre
- 150 ore nel progetto didattico
  - Fino a 105 in attività rendicontate
  - Fino a ulteriori 45 per auto-formazione su strumenti e metodi di lavoro utili al progetto
- 100 ore per studio personale in preparazione alla prova scritta e revisioni di avanzamento

Regole e vincoli – 1

- **Svolgere attività collaborative**
  - ~7 **persone / gruppo → condividere, ripartire, coordinare, verificare**
- **Cercare soluzioni sostenibili a problemi complessi**
  - **Tipologia utenti, dominio d'uso, risorse disponibili, prospettive**
  - **Auto-apprendimento di tecnologie e metodi di lavoro**
- **Adottare un approccio ingegneristico**
  - **Lavorare in modo disciplinato, sistematico, quantificabile**
  - $[85\,..\,105]$ **ore di impegno individuale → costo esterno rendicontato per attività obbligatorie**
  - ≈ 45 **ore di esplorazione tecnologica → costo interno per attività integrative (da condividere, ripartire e contenere)**
  - **Ore produttive, non tempo trascorso**

Regole e vincoli – 2

- **Partecipa solo chi ha soddisfatto le propedeuticità**
  - **Basi di Dati (superamento completo)**
  - **Programmazione a oggetti (≥ prova scritta)**
- **Chi ha altri "arretrati", li sani prima di cimentarsi con il progetto**

- **I gruppi sono formati in sessione pubblica**
  - **Gli aventi diritto si registrano in tabellone @ Google Docs**
  - **Gli altri studenti si registrano in altro tabellone, specificando le propedeuticità non soddisfatte**
    - Per costoro varranno ulteriori regole e scadenze di ingresso

Regole e vincoli – 3

- L'impegno necessario per raggiungere gli obiettivi di progetto deve avere limite superiore
  - Compatibile con gli altri propri obblighi personali
  - Questo sconsiglia la partecipazione con "arretrati"
- Gli obiettivi di progetto devono essere fissati in modo elastico
  - Tra un minimo accettabile e un massimo ambizioso, negoziati dinamicamente con il proponente

Gli argomenti che tratteremo

- Processi, ciclo di vita e modelli di sviluppo del SW
- Gestione di progetto
- Amministrazione IT
- Analisi dei requisiti
- Progettazione
- Documentazione
- Qualità
- Verifica e validazione

- UML: diagrammi dei casi d'uso
- UML: diagrammi delle classi e dei *package*
- UML: diagrammi di sequenza e di attività
- *Design pattern*: creazionali, comportamentali, architetturali
- Stili architetturali
- Principi SOLID

Come lo facciamo – 2

- **Tre lezioni in modalità *flipped***

Flipping the classroom means that students gain first exposure to new material **outside** of class (studying documents and online resources) and then use class time to do the **harder** work of assimilating that knowledge through problem-solving, discussion, debates

Come lo facciamo – 3

- **Strumenti di lavoro collaborativo (12 ottobre)**
  - **Gestione progetto (calendari, canali, *ticket*)**
  - **In concerto con l'insegnamento TOS**
    - Versionamento (p.es., git, SVN)
    - Configurazione e *build* (p.es, Maven, Ant, Gradle, …)
- ***Lessons learned* (12 novembre)**
  - **Ricercare ed elaborare "consigli dei veterani" su come organizzare la gestione collaborativa delle attività di documentazione**
- ***Ways of working* (11 dicembre)**
  - **SEMAT, sviluppo *agile* (p.es. Scrum)**

Fonti e risorse – 1

- **I libri di SWE si dividono in due categorie**
  - **Teorici : trattano la materia in modo privo di riflessi di esperienza concreta**
    - Principi esposti, ma spesso non vissuti
  - **Esperienziali : espongono l'esperienza dell'autore, senza relazionarla bene alla visione generale del problema e della disciplina**
    - Eccessiva enfasi sugli aspetti accidentali
- **Noi useremo**
  - ***Software Engineering*, 10th ed., 2014, di Ian Sommerville, edito da Addison Wesley (Pearson Education)**

Fonti e risorse – 2

- Faremo anche riferimento a
  - Guide to the Software Engineering Body of Knowledge (SWEBOK v3)
    IEEE Computer Society
    Software Engineering Coordinating Committee
  - https://www.computer.org/education/bodies-of-knowledge/software-engineering
- Che ci aiuta a familiarizzarci con le aree di conoscenza della disciplina SWE

Fonti e risorse – 3

Table I.1. The 15 SWEBOK KAs
Software Requirements
Software Design
Software Construction
Software Testing
Software Maintenance
Software Configuration Management
Software Engineering Management
Software Engineering Process
Software Engineering Models and Methods
Software Quality
Software Engineering Professional Practice
Software Engineering Economics
Computing Foundations
Mathematical Foundations
Engineering Foundations

Noi ci occupiamo di queste 10

Fonti e risorse – 4

- **Come altri testi di consultazione useremo**
  - **E. Gamma, R. Helm, R. Johnson, J. Vlissides**
    ***Design Patterns,* 2002**
    Addison-Wesley (Pearson Education Italia)
    - E le moltissime risorse digitali che li approfondiscono
  - **C. Larman**
    Applicare UML e i *pattern*
    Pearson Italia (5° edizione, 2020)

Come si studia SWE

- **Costruendo incrementalmente il proprio glossario**
  - **Basandolo inizialmente sulla teoria**
    - Individuazione dei termini, definizione dei significati
  - **Consolidandolo con la pratica**
    - Applicazione dei significati, confronto critico con l'esperienza
  - **Discutendolo con i colleghi**
    - Unendo conoscenze parziali, correggendosi reciprocamente
- **Integrando le diapositive con studio personale**
  - **Ricercando altre fonti e risorse autorevoli**