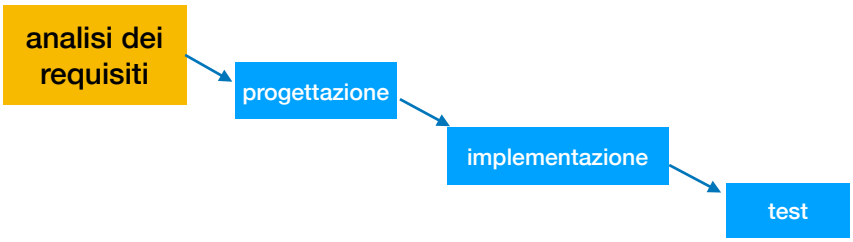
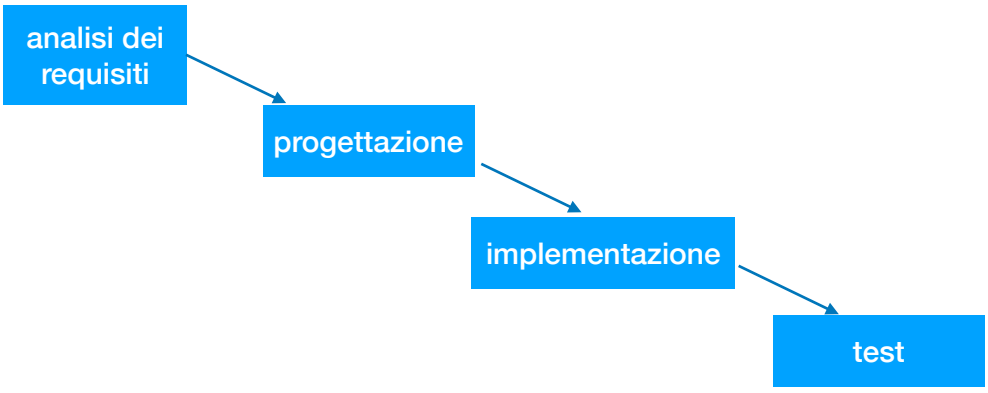
Analisi dei requisiti



- serve negoziare compromessi tra desideri, bisogni, costi e fattibilità.
- infine si compila un **documento di specifica dei requisiti del software,** che costituisce un accordo scritto tra le parti:
  - guida allo sviluppo del software e si usa per dirimere eventuali controversie

- Spessissimo i requisiti iniziali sono stati **incompresi**, vanno **modificati** o **integrati**, causando ritardi, costi ed errori
- Serve una comunicazione continua e diretta con le parti interessate al progetto

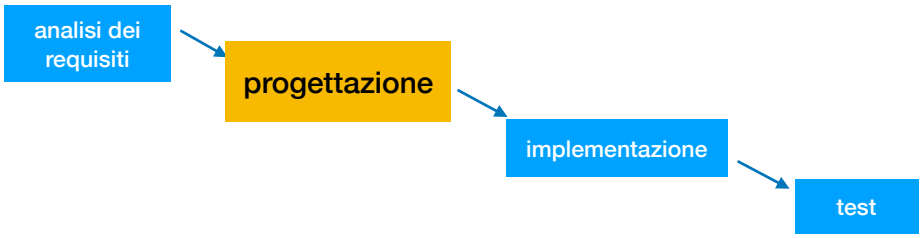
Fasi di sviluppo



Esempi: **Uniweb** oppure **Moodle**

- Provare a pensare cosa significano queste diverse fasi nello sviluppo e nella manutenzione di due software complessi come Uniweb e Moodle
- Pensare al ciclo di vita di questi software

Progettazione

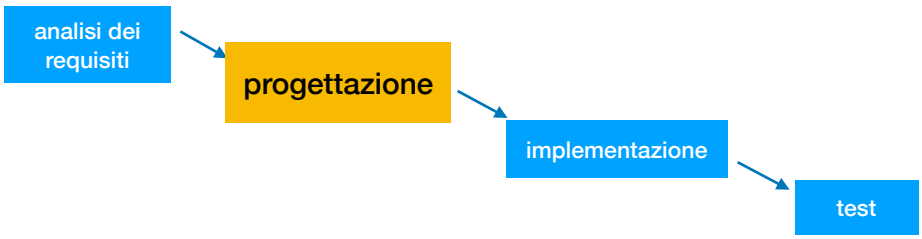


- Se l'**analisi dei requisiti** specifica il problema da risolvere, ***il che cosa***, la **progettazione** specifica la soluzione al problema, ***il come***

- in questa fase si definisce la struttura interna del sistema software, ad un livello di dettaglio tale da poter essere convertito poi in programma.

  Esempio: è richiesto il **sito Web di un negozio di mobili**
  - interfaccia del sito
  - un sistema di acquisto online
  - integrazione con canali social
  - un planner di mobile componibile

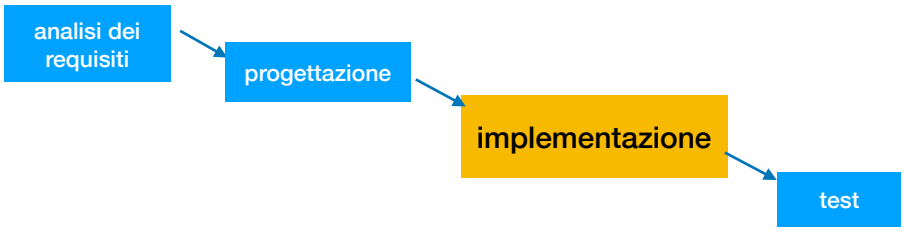
Progettazione



- Se l'**analisi dei requisiti** specifica il problema da risolvere, ***il che cosa***, la **progettazione** specifica la soluzione al problema, ***il come***

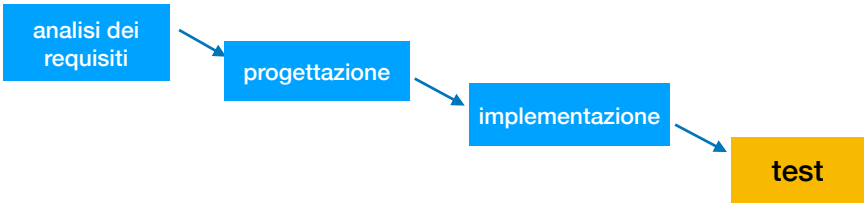
- in questa fase si definisce la struttura interna del sistema software, ad un livello di dettaglio tale da poter essere convertito poi in programma.
  - Ci sono varie metodologie e notazioni per la progettazione, es. modelli e diagrammi di vario genere, che costituiscono la **documentazione** di questa fase. Ma ***non sono notazioni stabili e standard***, come ad esempio lo sono quelle degli architetti.

Implementazione



- **Stesura del codice** vera e propria.
- Creazione di file di dati e sviluppo di **database**

test



- testing del codice: per controllare che non ci siano errori e che **rispetti la specifica dei requisiti**

- servono test di accuratezza di **tutte le fasi** di sviluppo: **software quality assurance**
  - seguire buone prassi di raccolta requisiti,
  - qualità e aggiornamento della documentazione,
  - revisioni periodiche tra le diverse parti coinvolte nello sviluppo.

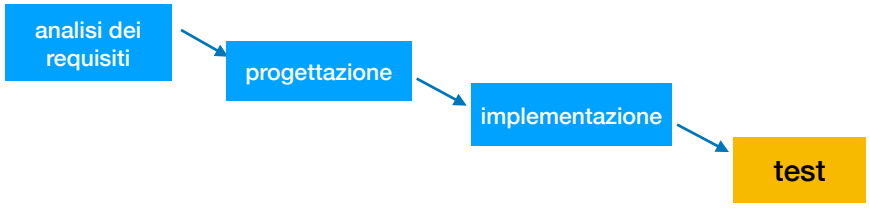
Documentazione

1. **Documentazione utente**: caratteristiche del prodotto e ***come si usa***
   - pensata per l'utente del software, quindi non tecnica
   - manuale d'uso separato oppure incluso nel software (informazioni in piccoli *help package* visualizzatili mentre si usa il software, magari che compaiono in automatico se si indugia troppo)

2. **Documentazione tecnica/di sistema**: caratteristiche interne del software e ***come mantenerlo***
   - molto più tecnica, anche informazioni su come il software va installato, configurato e mantenuto, utili al tecnico amministratore di sistema
   - ci sono strumenti automatici che aiutano a generare e mantenere aggiornati i documenti tecnici

test

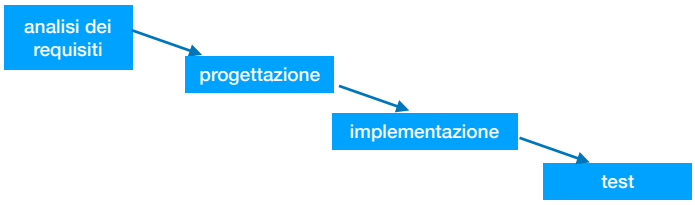
Program testing can only show the presence of bugs, but not their absence!



- testing del codice: per controllare che non ci siano errori e che **rispetti la specifica dei requisiti**
- servono test di accuratezza di **tutte le fasi** di sviluppo: **software quality assurance**
  - seguire buone prassi di raccolta requisiti,
  - qualità e aggiornamento della documentazione,
  - revisioni periodiche tra le diverse parti coinvolte nello sviluppo.

- nonostante il testing esaustivo restano **quasi sempre errori**, magari **nascosti per anni** ma che poi causano gravi danni

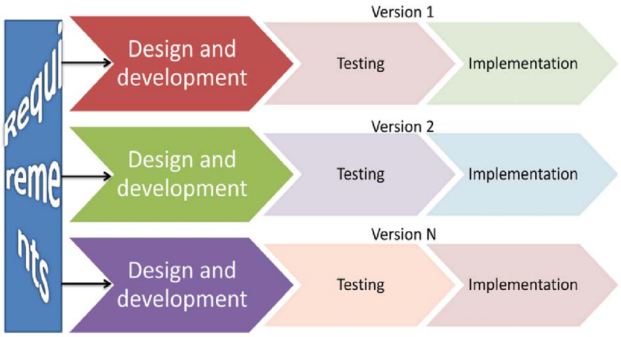
Modello a cascata



- Spessissimo i requisiti iniziali sono stati **incompresi**, vanno **modificati** o **integrati**, causando ritardi, costi ed errori
- Serve una comunicazione continua e diretta con le parti interessate al progetto

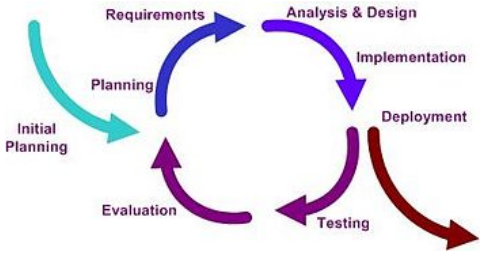
Modello incrementale

si ***estende*** una versione iniziale aggiungendo nuove funzionalità



Modello iterativo

si ***raffina*** una versione iniziale, magari partendo da un ***prototipo*** o un ***mock-up*** (versione non funzionante ma dimostrativa)



qualità del software

- estremamente complesso descrivere in maniera **precisa** ed **esaustiva**
  - **cosa deve fare** un programma (specifica/algoritmo)
  - **cosa fa** un programma e **come lo fa** (implementazione)

e "dimostrare" che corrispondono!

- Avere a disposizione **il codice sorgente** offre completa **trasparenza**, ma non completa **intelligibilità**.

qualità del software

- **garantire** che un programma, *eseguito con qualsiasi input*, non ha errori , è estremamente difficile:
  - cos'è un errore?
    - risultato errato, non terminazione, interruzione improvvisa (errore runtime), falla di sicurezza, lentezza delle prestazioni....
    - errore rispetto a cosa? specifica dei requisiti non facile da fare in modo chiaro, corretto, completo.
  - qualsiasi input! possono essere infiniti, testing non esaustivo

- **non è la stessa cosa** ad esempio **per l'ingegneria** meccanica, o altri tipi di tecnologie, che offrono maggiori garanzie di affidabilità.

Ingegneria del software

A differenza dell'ingegneria non digitale

- mancano delle metriche quantitative per **misurare le proprietà del software**:
  - **quanto è *corretto*? quanto è *grande*? quanto *consuma*?** hanno spesso poco senso, o un senso poco confrontabile con altri software

- **approccio applicativo**,
  - test empirici, linee guida e buone prassi, standard di qualità di processo
- **approccio teorico**
  - dimostrare (in modo automatico) l'assenza di specifici errori

ancora grossi problemi di affidabilità

Collection of Software Bugs

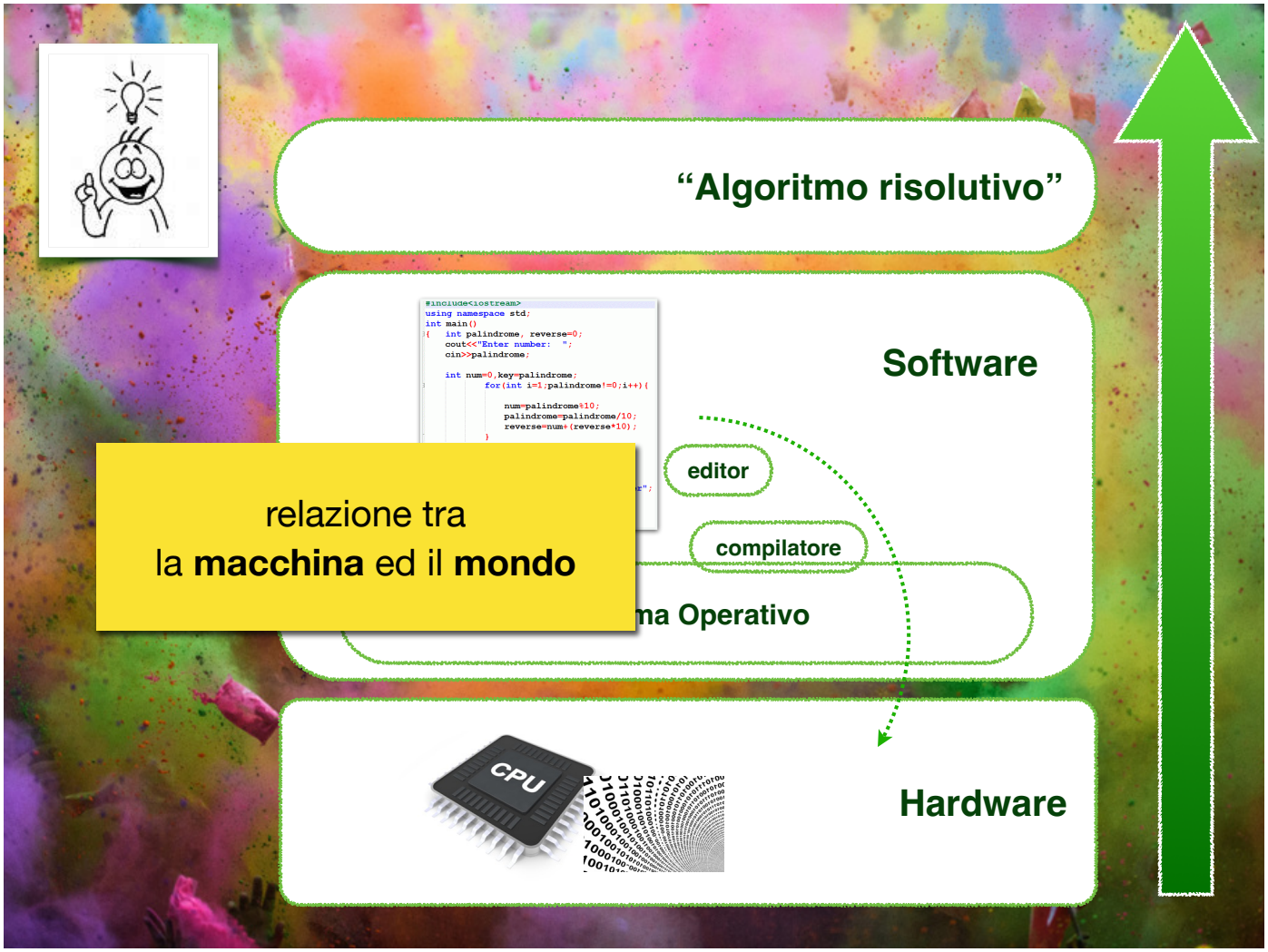
https://www5.in.tum.de/~huckle/bugse.html

- ingenti danni in settore bancario, aerospaziale, medicina, aviazione…
- The DAO e smart contract …*errore o feature del codice?*
- **non solo bug**…anche pessime pratiche di sviluppo e uso di tool (e.g. fogli excel).

- **Documentazione in ogni fase!!!**

Comunicare il software

- **Come si comunica efficacemente il proprio software?**
  - che significa comunicare?
  - comunicare **cosa**?
    - il codice, la specifica, in che contesto va usato, che problemi risolve, che dipendenze ha…
  - a **chi** comunicare?
    - un programmatore, un utente, un commerciale, un giudice



Verso la conclusione...

Consapevolezza Digitale

Su cosa abbiamo riflettuto:

- che significa ***sviluppare*** software
- che significa ***eseguire*** software
- che significa ***valutare*** un software (correttezza, efficienza, qualità)
- che significa ***comunicare*** un software

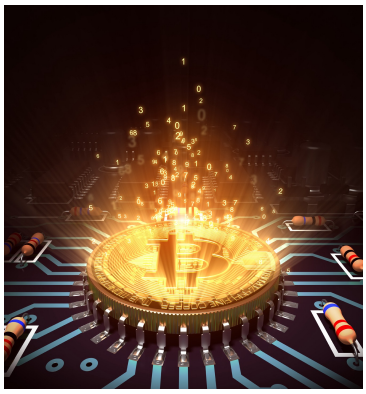
La tecnologia non è neutrale

le tecnologie che usiamo per ***mediare le relazioni*** con gli altri *esseri umani*, gli *oggetti* e i *luoghi* che abitiamo, **influenzano comportamenti**, modi di **lavorare**, imparare, comunicare e divertirsi ...**pensare**.

Consapevolezza Digitale

Serve sviluppare due capacità importanti:

1. la capacità di guardare ai sistemi digitali con uno **sguardo attento ai diversi livelli di astrazione di cui sono composti**, distinguendo le criticità che dipendono:
    - dai **requisiti di progettazione**,
    - dalla specifica **logica di funzionamento**,
    - dalla correttezza **dell'implementazione**,
    - **dall'ambiente di esecuzione e manutenzione**,
    - oppure **dal contesto di uso**. (AI, "*Soluzionismo Digitale*")



Consapevolezza Digitale

2. la capacità di **chiedere conto di come funziona un sistema software,** al fine di comprendere:
    - **come l'informazione è rappresentata nei suoi *dati***,
    - **che manipolazioni vengono effettuate**,
    - **cosa viene *trascurato***,
    - **chi e come ha fatto le *scelte*** progettuali e implementative,
    - **che garanzie di *qualità*** offre il software.

Servono dunque in tutti gli ambiti
professionisti capaci di approcci scientifici interdisciplinari,
capaci di **riconoscere la propria corresponsabilità**
nell'impatto sociale delle tecnologie digitali.

The End