


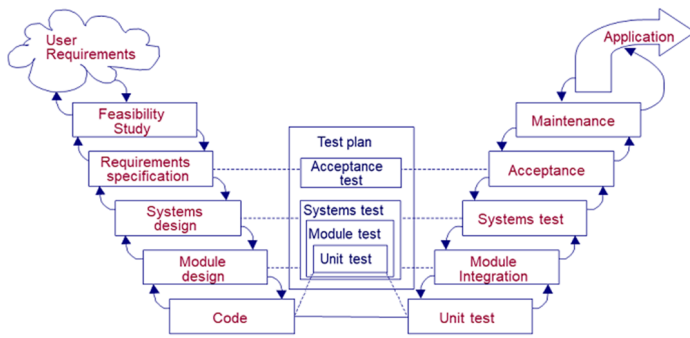
81614 - Laboratory of Software Systems

[SITO WEB di Ateneo](#)
[LabSS-lectures site](#)

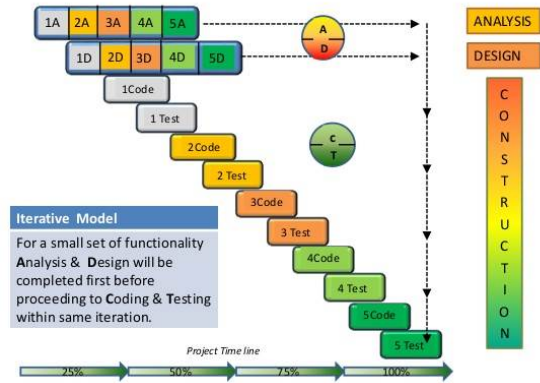
The context

<p style="text-align: center;">Informatico: cosa è?</p> <ul style="list-style-type: none"> • Un <i>tecnico</i> (che “pilota” tecnologie) ? • Un <i>analista</i> (che valuta problemi)? • Un <i>manager</i> (che gestisce processi di produzione)? • Un <i>progettista</i> (di prodotti e tecnologie)? • Uno <i>scienziato</i> (che studia e crea le fondazioni)? • Un <i>venditore</i> (che analizza le esigenze del mercato)?  <ul style="list-style-type: none"> • L'informatica: una commodity o qualcosa da costruire? 	<p style="text-align: center;">Scienza o ingegneria?</p> <div style="display: flex; justify-content: space-around;"> <div style="width: 45%;"> <p>Scienza</p> <ul style="list-style-type: none"> • Attività speculativa intesa ad analizzare, definire e interpretare la realtà sulla base di criteri rigorosi e coerenti </div> <div style="width: 45%;"> <p>Ingegneria</p> <ul style="list-style-type: none"> • Disciplina che studia l'applicabilità delle conoscenze scientifiche alle necessità della vita civile e del suo sviluppo socio-economico </div> </div>
<p style="text-align: center;">Computer science & software engineering</p> <div style="display: flex; justify-content: space-around;"> <div style="width: 45%;"> <p>Computer Science</p> <ul style="list-style-type: none"> • is concerned with the theories and methods which underlie computers and software systems. </div> <div style="width: 45%;"> <p>Software Engineering</p> <ul style="list-style-type: none"> • is concerned with the practical problem of producing software. • In the belief that software could be engineered on the same footing as traditional engineering disciplines, a NATO study group coined the term “<i>Software Engineering</i>” in 1967. </div> </div>	<p>Software Engineering is an engineering discipline which is concerned with all aspects of software production from the early stages of system requirements through to maintaining the system after it has gone into use.</p>
<p style="text-align: center;">Software process development</p> <div style="display: flex; justify-content: space-around;"> <div style="width: 45%;"> <p>Agile software development</p> <p>a group of software development methods based on <i>iterative and incremental development</i>, where requirements and solutions <i>evolve</i> through collaboration between <i>self-organizing, cross-functional teams</i>. It promotes adaptive planning, evolutionary development and delivery, a time-boxed iterative approach, and encourages rapid and flexible response to change.</p> </div> <div style="width: 45%;"> <p>Model-driven engineering (MDE)</p> <p>focuses on creating and exploiting domain models (that is, abstract representations of the knowledge and activities that govern a particular application domain), rather than on general purpose computing (or algorithmic) concepts.</p> </div> </div>	<p>https://en.wikipedia.org/wiki/Software_engineering</p> <p>http://en.wikipedia.org/wiki/Model-driven_engineering</p> <p>http://en.wikipedia.org/wiki/Agile_software_development</p> <p>http://en.wikipedia.org/wiki/Scrum_(software_development)</p> <p>https://www.comp.nus.edu.sg/~damithch/pages/SE-quotes.htm</p>

Software Project Lifecycle ("Waterfall")

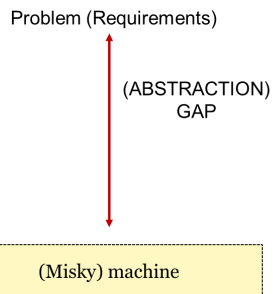


ITERATIVE Model



Machines and problems

Overcoming the abstraction gap

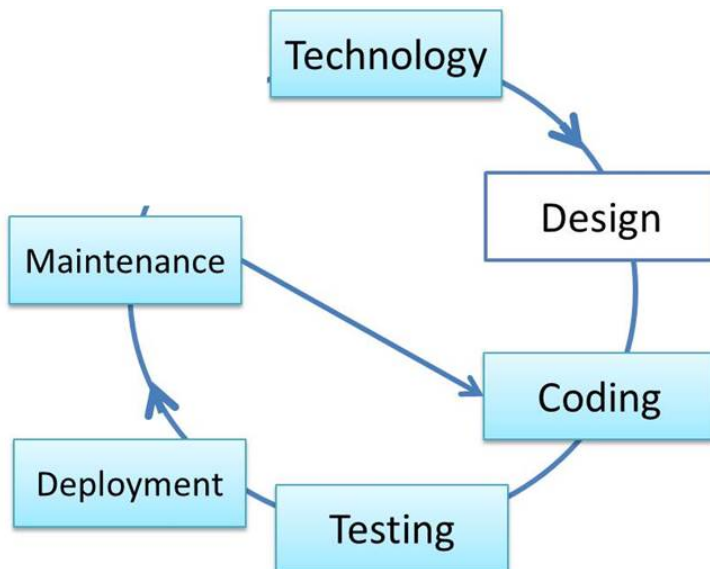


- TOP DOWN: from the problem to the machine
- BOTTOM UP: from the machine to the problem

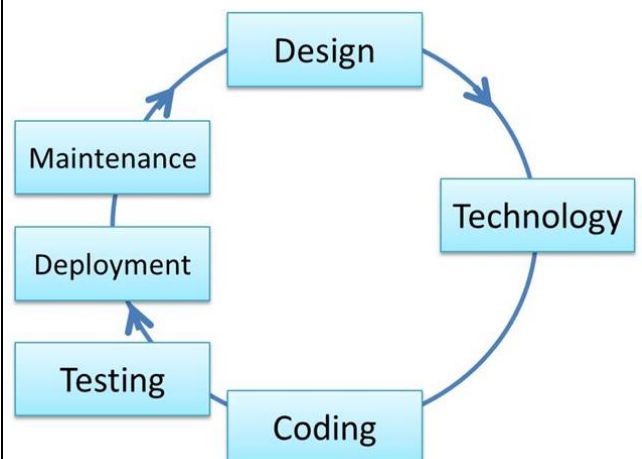
Minsky machine

- CLR (r): CLear register r . (Set r to zero.)
- INC (r): INCrement the contents of register r .
- DEC (r): DECrement the contents of register r .
- JZ (r, z): IF register r contains Zero THEN Jump to instruction z ELSE continue in sequence.

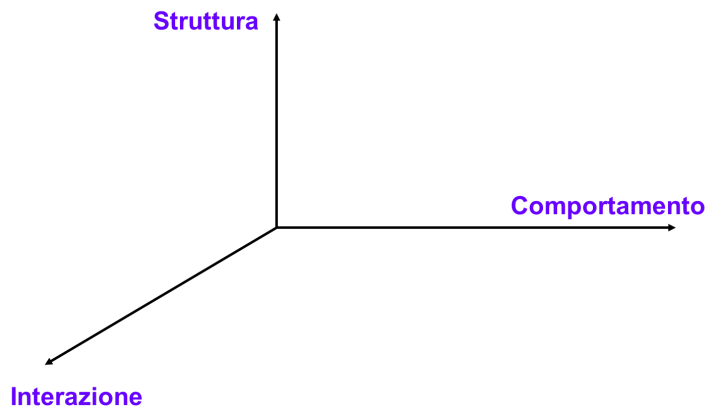
Bottom-up



Top-down

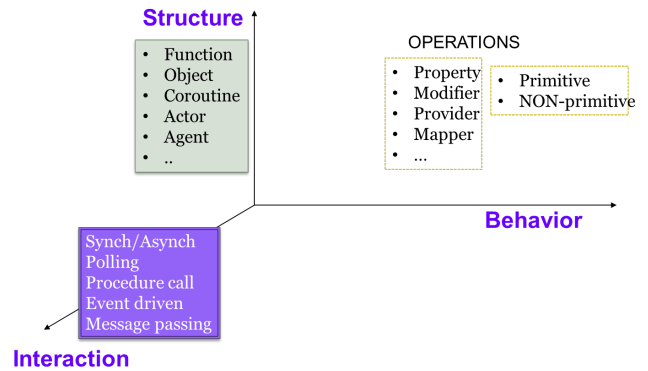


Architecture: basic dimensions



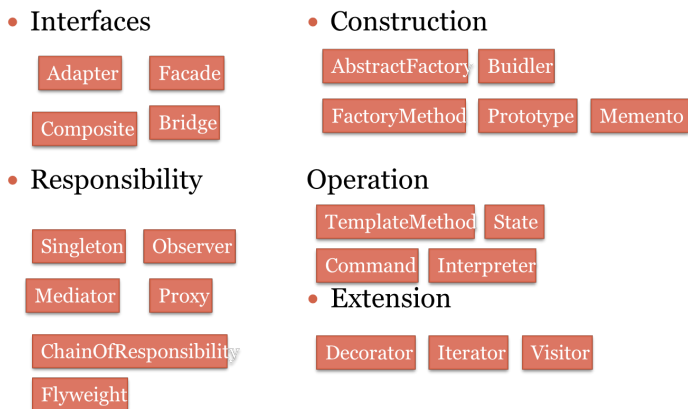
18/09/2019 10:00:00 AM

Languages: basic concepts



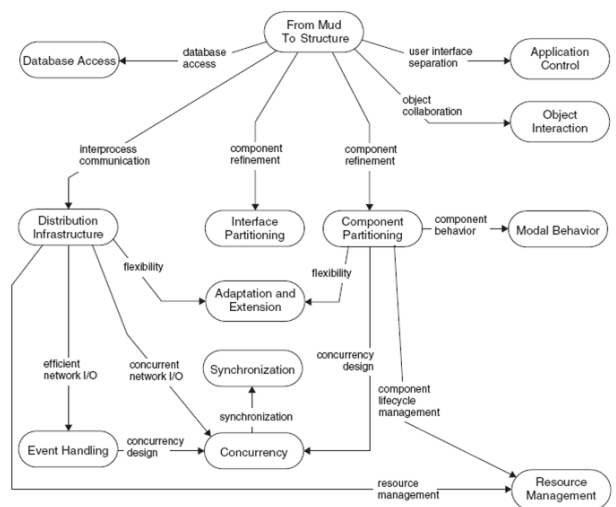
18/09/2019 10:00:00 AM

Design pattern GOF



18/09/2019 10:00:00 AM

Design pattern POSA4



Franck Bushmann, Kevin Henney, Douglas C. Schmidt:
Pattern-Oriented Software Architecture: Volume 4 A Pattern
Language for Distributed computing, John Wiley & Sons, 2007

Motto (per il costruttore di software)

Non c'è codice senza progetto, non c'è progetto
senza analisi del problema, non c'è problema
senza requisiti.

- Nel processo di produzione del software il lavoro dei diversi attori (product-owner, analista, progettista, sviluppatore, etc.) si basa su un insieme di conoscenze e decisioni **che spesso rimangono implicite** all'interno del processo di produzione.
- Uno degli scopi dei processi model-based è **rendere esplicite** queste conoscenze attraverso la costruzione di 'versioni semplificate del sistema' (modelli) espressi con notazioni formali ...
- ... attraverso l'uso di un **linguaggio** (testuale e/o grafico) con una **ben precisa sintassi e semantica**.
Ad es. : [UML2](#)

Modello nella ingegneria del software

- Nei **processi di costruzione del software**, il termine **modello** va inteso come un **insieme di concetti e proprietà** volti a **catturare aspetti essenziali** di un sistema, collocandosi in un **preciso spazio concettuale**.
- Per l'**ingegnere del software** quindi un **modello** costituisce una **visione semplificata** di un sistema che rende il sistema stesso più accessibile alla comprensione e alla valutazione e facilita il trasferimento di informazione e la collaborazione tra persone.



- L'insieme dei modelli che descrivono un sistema dovrebbe formare una descrizione **completa, coerente, consistente e non (troppo) ridondante**.

18/09/2019 10:00:00 AM

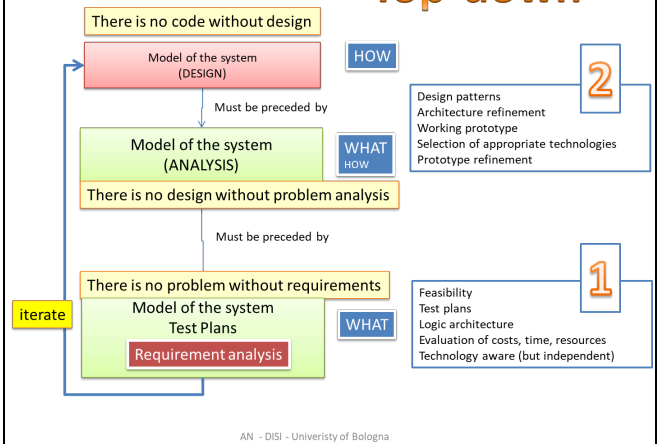
Requirements and Testing

More than the act of testing, the act of **designing tests** is one of the best bug preventers known.

https://en.wikipedia.org/wiki/Software_testing

<http://softwaretestingfundamentals.com/software-testing-quotes/>

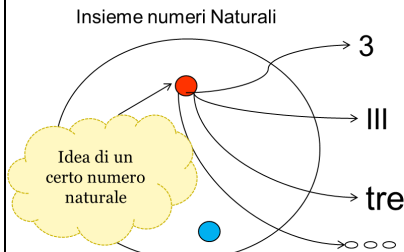
Top-down



AN - DISI - University of Bologna

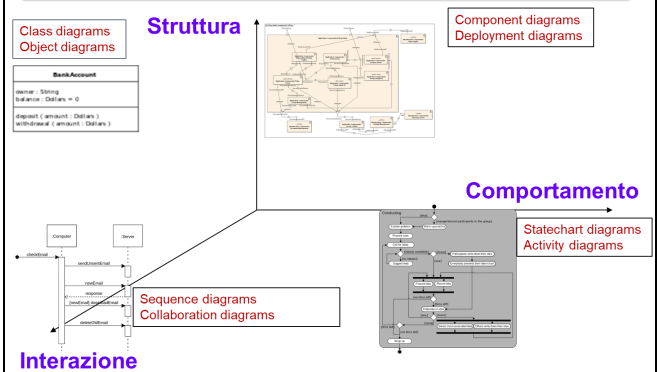
Modelli e linguaggi

- Quattro diverse rappresentazioni
- espresse in quattro differenti linguaggi
- dello stesso modello



AN - DISI - University of Bologna

Architecture: UML diagrams



AN - DISI - University of Bologna

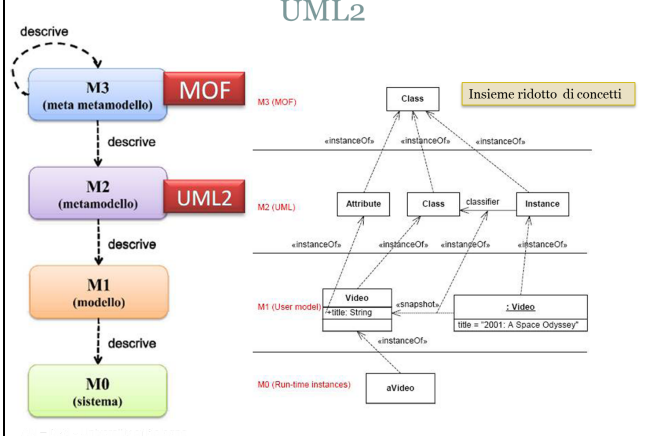
Metamodeling

The Object Management Group (OMG) has developed a metamodeling architecture to define the UML, called the **Meta-Object Facility**, designed as a four-layered architecture.

The most prominent example of a Layer 2 MOF model is the UML metamodel, which describes the UML itself. These M2-models describe elements of the M1-layer, and thus M1-models. These would be, for example, models written in UML. The last layer is the M0-layer or data layer. It is used to describe runtime instances of the system.

The meta-model can be extended using a mechanism called **stereotyping**. This has been criticised as being insufficient/untenable.

UML2



AN - DISI - University of Bologna

Xtext

Xtext is a framework for development of programming languages and domain-specific languages. With Xtext you define your language using a powerful grammar language. As a result you get a full infrastructure, including parser, linker, typechecker, compiler as well as editing support for Eclipse.

<https://www.eclipse.org/Xtext/>

Definition of a custom metamodel: example

```
grammar it.unibo.xtext.intro19.lot
with org.eclipse.xtext.common.Terminals
generate iot "http://www.unibo.it/xtext/intro19/lot"
```

```
lotSystem: "System" spec=lotSystemSpec ;
terminal VARID : ('A'..'Z'|_ ('a'..'z'|'A'..'Z'|_ ('0'..'9')*)
QualifiedName : ID ('.' ID)* ;
```

```
lotSystemSpec: name=ID
(mqttBroker = BrokerSpec)? // Optional
(message += Message)* // N >= 0
;
BrokerSpec : "mqttBroker" brokerHost=STRING ":" brokerPort=INT ;
```

```
Message : Event | Dispatch ;
```

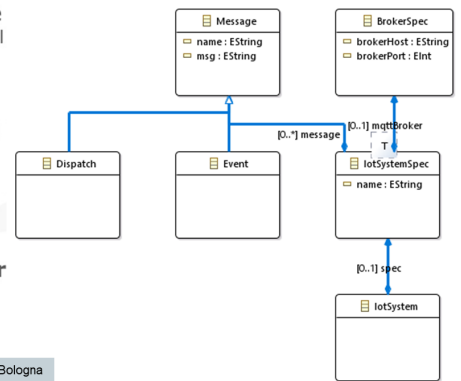
```
Event: "Event" name=ID ":" msg = STRING ;
Dispatch: "Dispatch" name=ID ":" msg = STRING ;
```

Eclipse Modeling Framework (EMF) Ecore

- The EMF core metamodel

- Definition of metamodels (or domain models)

- A de-facto standard for modeling...

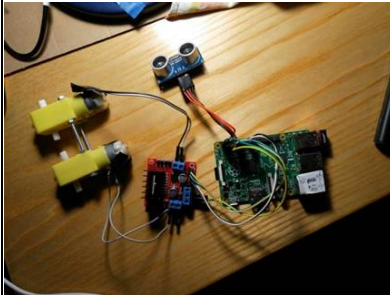


AN – DISI - University of Bologna

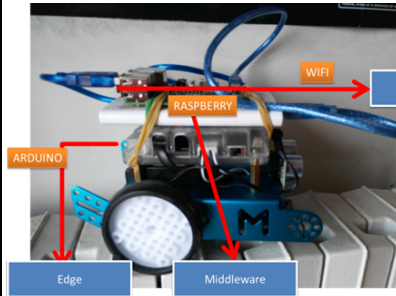
Our Goal

- (Learn to) build in a **concrete way**
 - working individually or **in a team**
 - following **agile and/or model-centered** methodology
- distributed software systems in the domain of **IOT (WOT)** applications

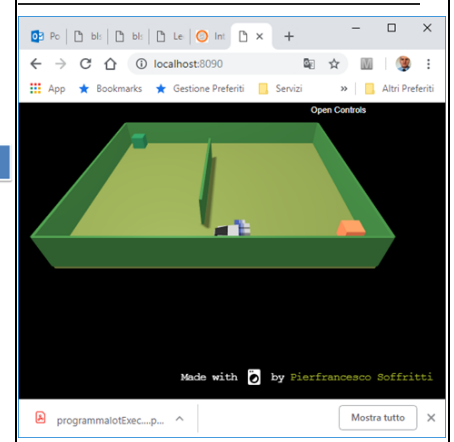
Basic hardware



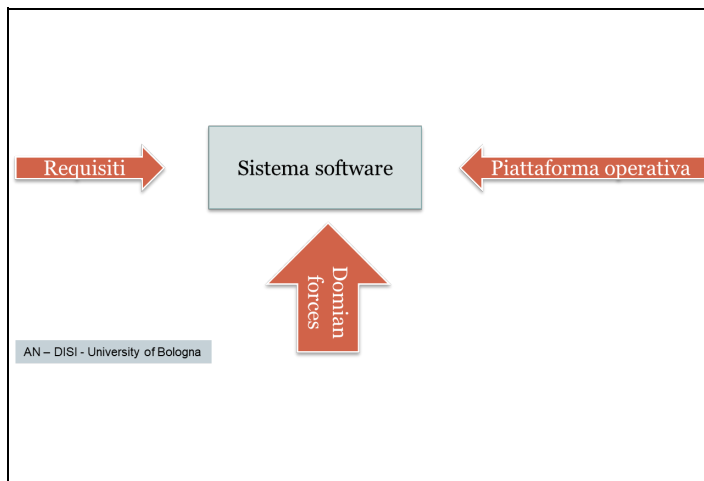
Real robot (MBOT)



Virtual environment and robot



- that must satisfy a set of pre-defined **requirements**
- running on a **heterogeneous** set of nodes (**PC, Arduino, RaspberryPi, Android**)



AN – DISI - University of Bologna

Costruire una sistema software

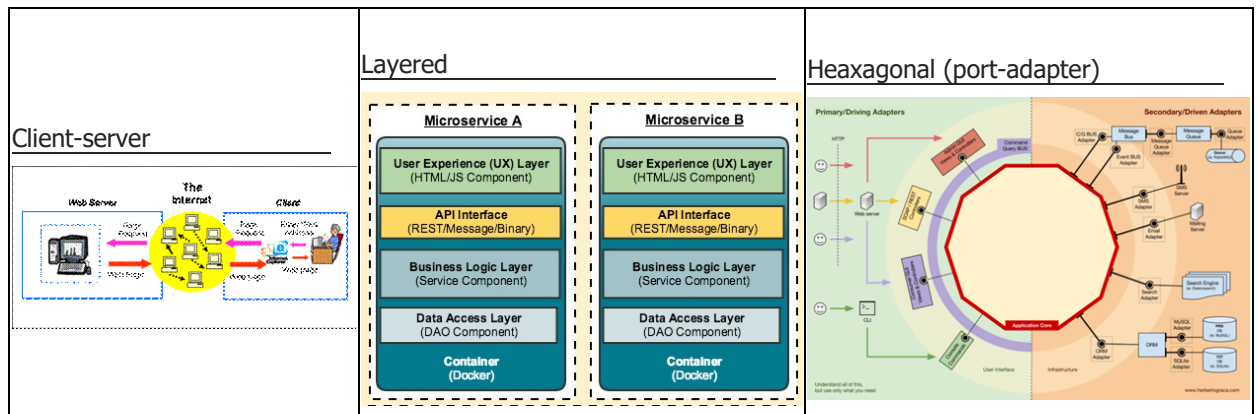
Occorre rispondere a un insieme di domande:

- Cosa deve fare (committente)
- Cosa deve essere (analisi)
Quale struttura-interazione-comportamento sono necessari tenendo conto dei vincoli che sorgono dai requisiti e dal problema
- Cosa è opportuno che sia (analisi)
Tenendo conto delle tecnologie, dell'andamento del mercato, delle risorse umane, economiche, temporali, etc.
- Come è fatto e come funziona (progetto)
Quale struttura-interazione-comportamento ha il sistema finale tenendo conto dei vincoli dall'analisi e dei criteri (pattern) usati per risolvere le forze (anche contrastanti) in gioco
- Come è validato
- Come è mantenuto

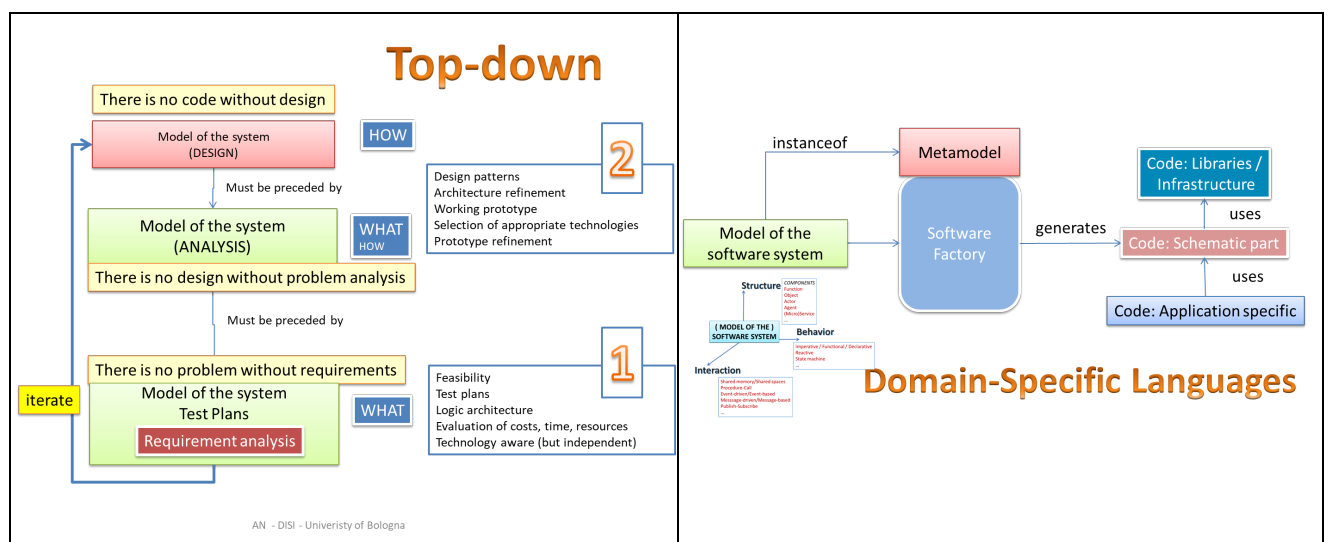
- using proper available technologies and tools

- in a way that is **'technology aware'** but also as much 'technology independent' as possible

- o focussing on the **architecture level**



4. and using proper custom meta-models (DSL) and platforms



More specifically

1. Sviluppiamo l'analisi dei requisiti e l'analisi del problema esprimendo fatti rilevanti attraverso **modelli eseguibili**
2. Discutiamo in modo sistematico avvalendoci di modelli basati su meta-modelli custom
 - o Usiamo i modelli **come se fossero il nuovo codice sorgente** costruendo generatori di codice usando Xtext
 - o Realizziamo **in modo automatico parti del sistema** avvalendoci (anche) dei design pattern per sistemi distribuiti
3. Impostiamo **piani di collaudo** ancor prima di avere iniziato la fase di progettazione
4. Realizziamo in **modo incrementale** (con nuovi modi di concepire il **riuso del software**) uno o più prototipi del sistema interagendo con il committente attraverso una successione di **SPRINT** in stile **Scrum**

Teaching and Assessment methods

1. Il laboratorio costituisce una parte essenziale del corso e la frequenza è pressochè indispensabile.
2. Gli studenti sono tenuti a progettare e costruire in modo incrementale sistemi software, che verranno valutati dal docente, in modo da acquisire una retroazione immediata sul lavoro svolto, che potrà essere utilizzata per modificare/migliorare quanto sviluppato.
3. La valutazione finale consiste nella presentazione e discussione degli artefatti prodotti in relazione alla costruzione di un sistema software (dello stesso tipo di quello discusso nei CaseStudy proposti durante il corso) che rispetti i requisiti pubblicati l'ultima settimana di lezione.

4. Questi artefatti possono essere prodotti in modo individuale oppure, preferibilmente, attraverso un lavoro cooperativo svolto in gruppi di 2/3 studenti.

Esempi di tema finale

- [Room cleaner \(pdf\)](#)
- [Bomb detector \(pdf\)](#)
- [Room Butler \(pdf\)](#)

Tools

Software

- Git, <https://github.com/anatali/iss2020Lab>
- IntelliJ , Eclipse DSL, Arduino IDE, Android Studio
- Gradle
- Java, Kotlin, Python, C++
- XText
- MQTT broker (Mosquitto)
- Jupyter

Hardware

- Arduino Uno
- RaspberryPi
- MBOT (o altro)
- GY521, GY271

By AN Unibo-DISI