Understanding Software Architecture

Lotfi ben Othmane

Goal

1. What is software architecture?
2. Why is it important?

Software Architecture in Layman Terms

Software architecture is about the important stuff

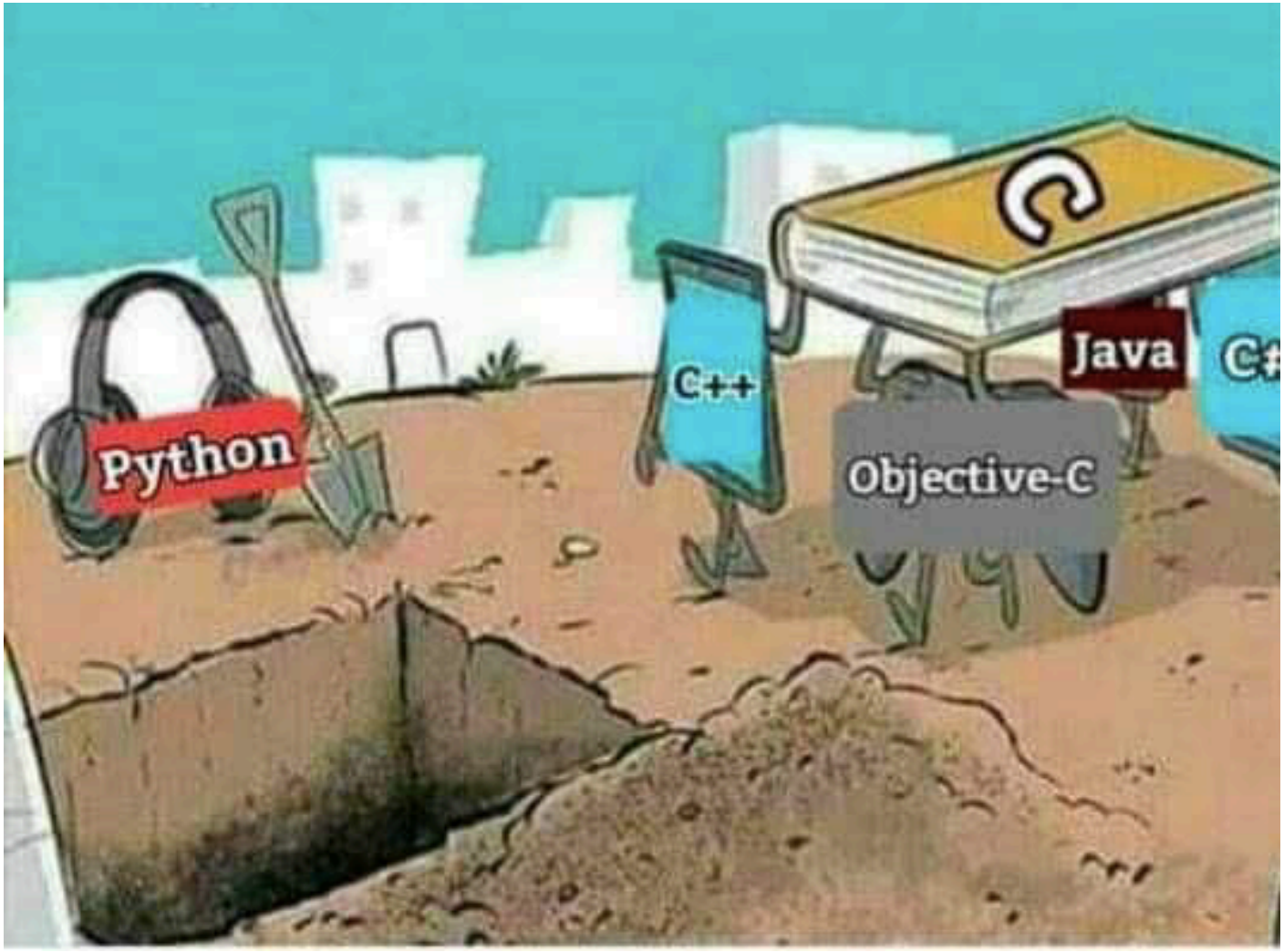
Computer scientist Ralph Johnson, who co-authored *Design Patterns: Elements of Reusable Object-Oriented Software*, once said:

"Architecture is about the important stuff. Whatever that is."

Software projects vary, and the amount of design effort, time, focus, and documentation devoted to particular aspects of a software architecture differ. Ultimately, software architecture consists of important design decisions that shape the system. It is made up of the structures and components that are significant to the quality, longevity, and usefulness of the system.

Software architecture consists of some of the earliest decisions that are made for a software system and some of the hardest to change. ...

Software Architecture in Layman Terms

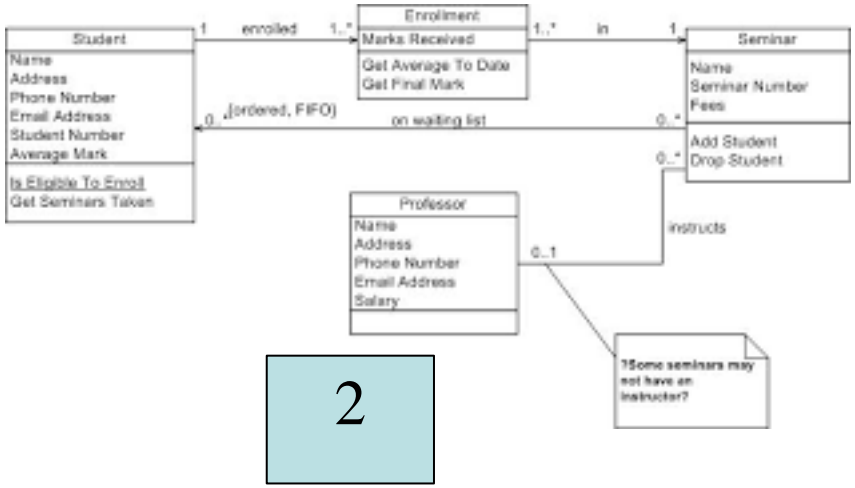


Software Architecture in Layman Terms

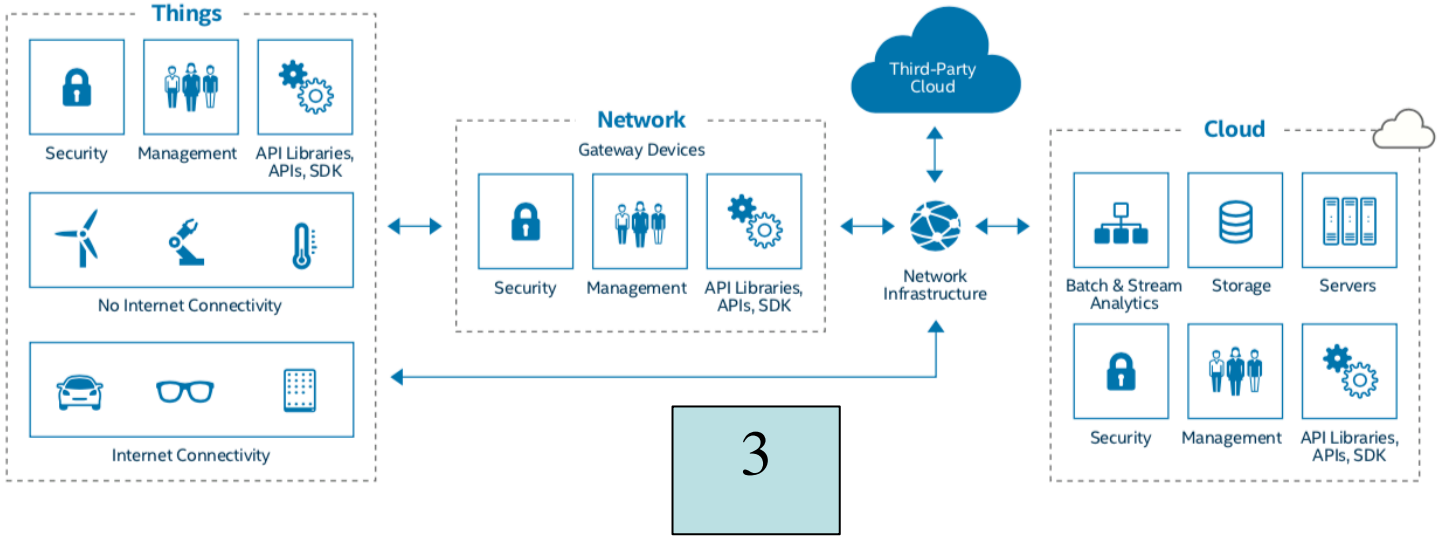
Which of the following Represents an Architecture?

```
'''
Created on December 19, 2018

@author: Lotfi ben Othmane
'''
import csv  # to reasesd the csv file
import sys

#import seaborn as sns; sns.set()  # for plot styling
#import numpy as np # data science package
from numpy import vstack,array
from numpy.random import rand
from scipy.cluster.vq import  vq, kmeans, whiten
import copy
import collections
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import stats
import gc


count_learnethod=[0]*17

'''
...This method returns the list of selected prefered learning methods for a given student
'''
def getLearnMethod(listmethods):
    methods=[0 for i in range(17)]
    lm = listmethods.split(';')

    for l in range(len(lm)):
        methods[int(lm[l])-1]=1
        count_learnethod[int(lm[l])-1]+=1

    # return an array that marks the used meth
    return methods

'''
...This method returns
'''
def EvaluateCognitivelevel(level, CognitiveValue):
    isCorrect  = 0 # This indicates wether the question was correct or not
```

1



2



3

What is Software Architecture?

[Software architecture goes] beyond the algorithms and data structures of the computation; designing and specifying the overall system structure emerges as a new kind of problem. Structural issues include gross organization and global control structure; protocols for communication, synchronization, and data access; assignment of functionality to design elements; physical distribution; composition of design elements; scaling and performance; and selection among design alternatives.

D. Garlan, M. Shaw, An Introduction to Software Architecture, Advances in Software Engineering and Knowledge Engineering, Volume I, World Scientific, 1993

What is Software Architecture?

Architecture is defined by the recommended practice as the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution.

ANSI/IEEE Std 1471-2000, Recommended Practice for Architectural Description of Software-Intensive Systems

What is Software Architecture?

The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them.

L.Bass, P.Clements, R.Kazman, Software Architecture in Practice (2nd edition), Addison-Wesley 2003

What is Software Architecture?

Which of of these items defines software architecture?

1. Refactor the code to have a clean structure
2. Show how data moves between the front end, the back end and the database
3. Partition the system into components considering requirements and constraints
4. Specify the technology to use in developing the software

Property 1 - Architecture Addresses Non-Functional Requirements

- Architecture addresses the questions:
  - *What* does the application do?
  - *And…*

Property 1 - Architecture Addresses Non-Functional Requirements

- Architecture addresses the questions:
  - *What* does the application do?
  - *How* does the application provide the functionality?

- Areas of non-functional requirements
  - Technical constraints
  - Business constraints
  - Quality attributes

Property 1 - Architecture Addresses non Functional Requirements

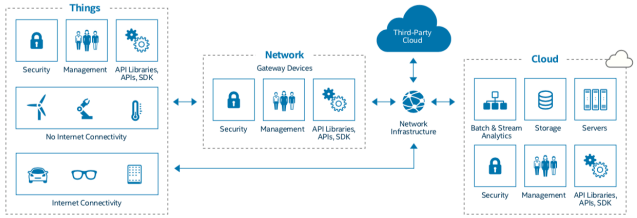
Examples of non-functional requirements

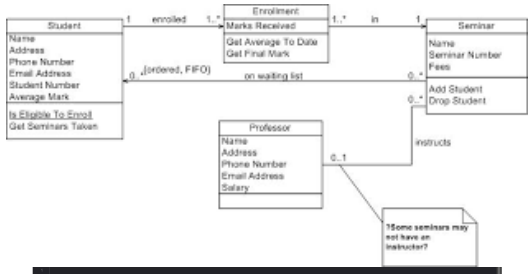
1. The users shall access the software remotely without installing a client application
2. The team has experience with Node.js. The server application shall be a Node.js application
3. To register for the system, the user must provide their name, address, email, phone number, and social security number
4. The system should support 500 concurrent users
5. The server application shall be deployed as a cloud service

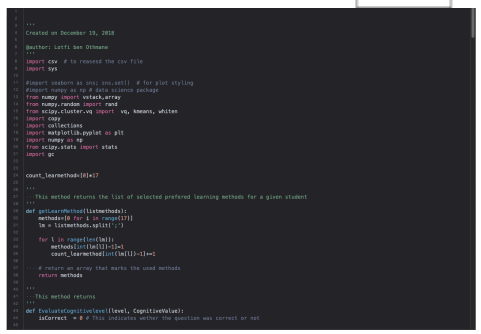
Property 2 - Architecture Is an Abstraction

- The architecture describes the system in sufficient details for the stockholders to collaborate
- The architecture suppresses or ignores details to focus the attention on the salient features and the constraints.
- Hierarchical decomposition is used to refine the system and ensure correctness.

Architecture







Code

Property 3 - Architecture Defines Structure

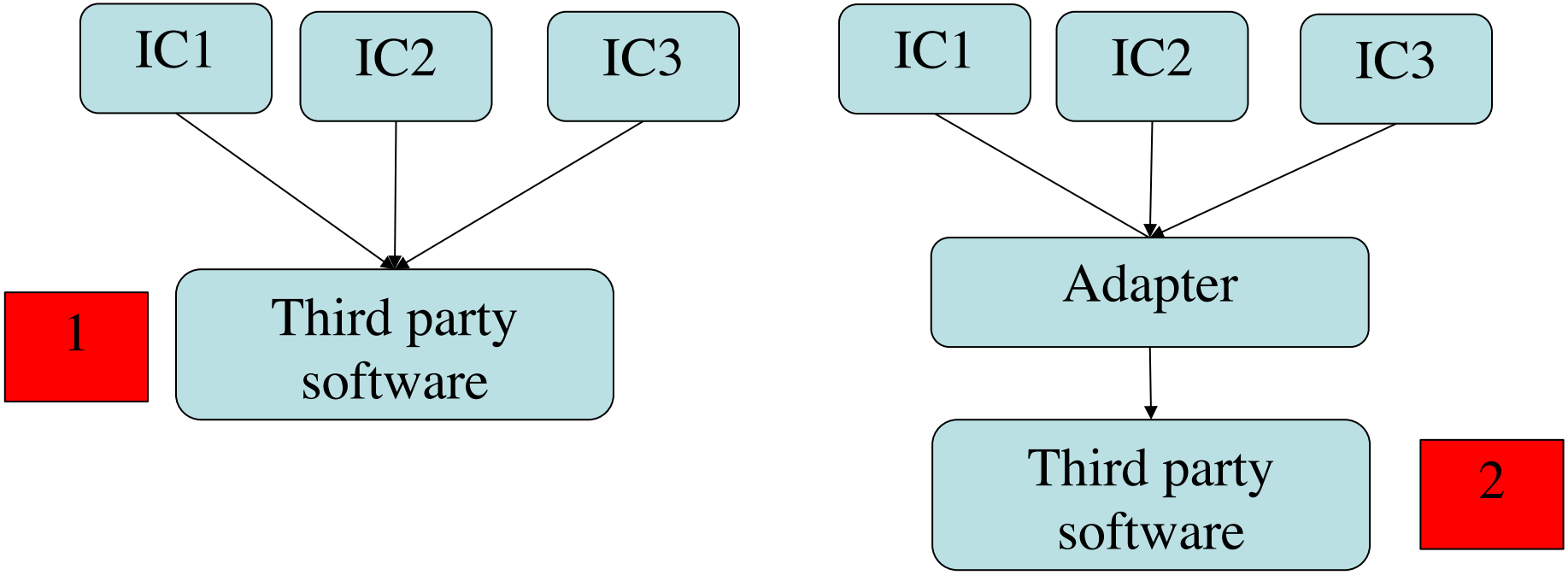
- Architects partition the system into components such that:
  - All the responsibilities of the software are assigned to the components
  - All the constraints and requirements are addressed

- A component is **“A recognizable chunk of software”**

- A key concern is to minimize the communication and dependencies between the components
  - High cohesive components
  - Loosely coupled architecture

Property 3 - Architecture Defines Structure

IC1, IC2, IC3 depend on the third party software



What is the impact of changing the third party software on the three components for both cases?

Property 4 - Architecture Specifies Components Communication

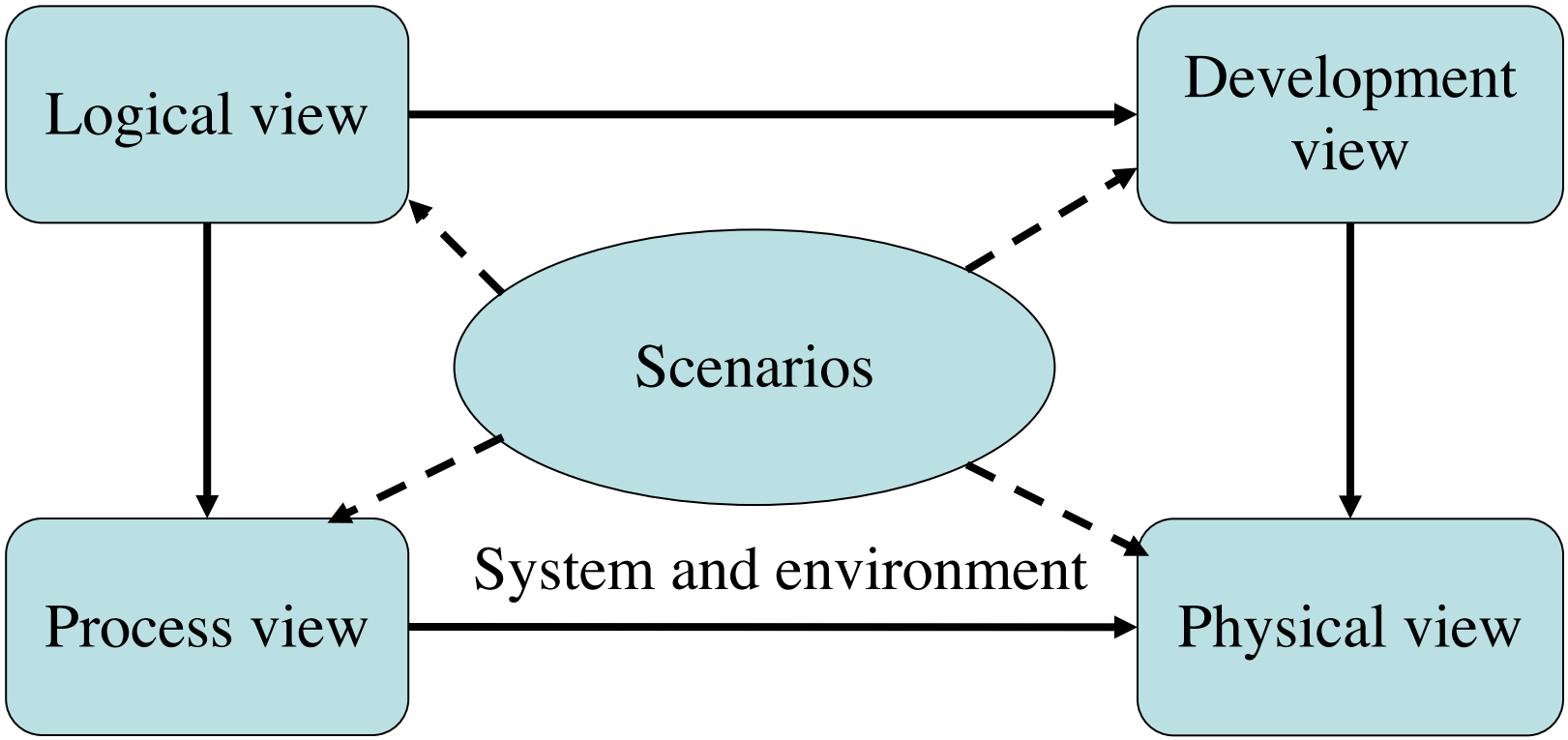
- The components of the architecture must exchange data and control to fulfill the use cases given the constraints.
- Communication needs include:
  1. Synchronous vs asynchronous communication
  2. Components are (1) in the same process or in different processes, (2) in the same node or different nodes, etc.
  3. Strategy to handle errors

- Architecture patterns address these
  - Communication: method calls, threads, protocols
  - Client-server software vs three tiers architecture

Architecture Views

End-users

Developers



Integration, performance scalability

System engineers

Architecture Views

- Another taxonomy
  - Module view
  - Components and connector view
  - Allocation view

Architect Role

- The role is defined by the environment

- The main required skills are
  - Liaison between the stakeholders
  - Design capabilities and good perception of the impacts of the decisions
  - Knowledge of the technologies – know what they do not know
  - Risk management – evaluate the risk associated to the design and be cautious

Architectures and Technologies

- Architects make design decisions often at beginning of the project and it is impossible often to validate them.
  - Prototypes help to gain confidence
- Technologies implement design patterns that could be used in the architecture
  - E.g., microservices, queue management, access control
- Architects need to understand the strengths and weaknesses of these technologies
- Projects often fail because of bad architecture choices

Discussion

Krutchen says: “The life of a software architect is a long (and sometimes painful) succession of sub-optimal decisions made partly in the dark.”

1. Do you agree with the statement?
2. Why the decisions are sub-optimal?
3. Why the decisions are made in the dark?

End

Prepare your question for the synchronous session on Tuesday or post it on Piazza. We will be happy to help.