CS3213 Project – Week 3

Requirement Modeling | 26-01-2022

❏ Discussion Requirement Elicitation
❏ Discussion Requirement Models
❏ Comments to Software Architecture
❏ Assignment 3
❏ Assignment 4

Group & Project Selection Deadlines

~~**Deadline** for group registration[1]:
Friday, Jan 21, 10 am~~

Deadline for project selection[1]:
Friday, Jan 28, 10 am

---

[1] https://docs.google.com/spreadsheets/d/15sk6WnvQHTjClhMi_TUuyDkyIow6lOmMHVhD5n3Qu-I/edit?usp=sharing

Comment for „late“ submissions

> **Assignment 1: Requirements Analysis & Elicitation**
>
> CS3213 Foundations of Software Engineering (AY21/22 Sem2)
>
> Submission Deadline: **Tue 18/01/2022, 10 pm**
> Discussion: Week 2 and 3
>
> ---
>
> - You must strictly comply with the noted deadline. No late submissions!

→ See CS3213 assignments as a project and manage your deadlines accordingly.

Sidenote: also check the naming scheme!

Last two weeks

- ❑ A1: Requirements Analysis & Elicitation
- ❑ Requirements Elicitation with Stakeholders
- ❑ A2: Requirements Modeling

Requirements Analysis & Elicitation

will be discussed in the lab sessions

some remarks today

Requirement Elicitation
– Closing Remarks

❏ begin **gentle** and proceed with **caution**

❏ prepare your **catalogue of questions** and ask **systematically**

❏ reveal **contradictions**

❏ **special cases** usually require more effort as the default case – you need to explore **all eventualities** in the system **with** the customer

❏ do not forget the **„as-is" state**

❏ Jewish motherhood (example of the *door access system*)

Discussion: Requirement Models

❏ *submitted models will be discussed in the lab sessions*

❏ different models have different **purposes**

  ❏ Goal Model: Stakeholders become more aware of potential alternatives for meeting their goals, and are therefore less likely to over-specify by prematurely committing to certain technological solutions.

  ❏ Use Case Model: overview functional features of system, easy understandale description of scenarios and special cases

  ❏ Activity Diagram: process flows and their actions/activities

  ❏ Sequence Diagram: interaction between objects

  ❏ State Transition Chart: object states and their transitions

  ❏ …

Common Modeling Purposes

❏ clarifying requirements

    ❏ modeling techniques need to support "why" and "how else" types of reasoning analysis

    ❏ incremental process

❏ provide traceability of rationales

❏ management of change

❏ verification of achievement of requirements

❏ support of reuse

Requirement Engineering – Common Challenges

❏ Limited access to project stakeholders

❏ Project stakeholders do not know what they want

❏ Project stakeholders change their minds

❏ Conflicting priorities

❏ Developers don't understand the problem domain

❏ Developers don't understand the requirements

?

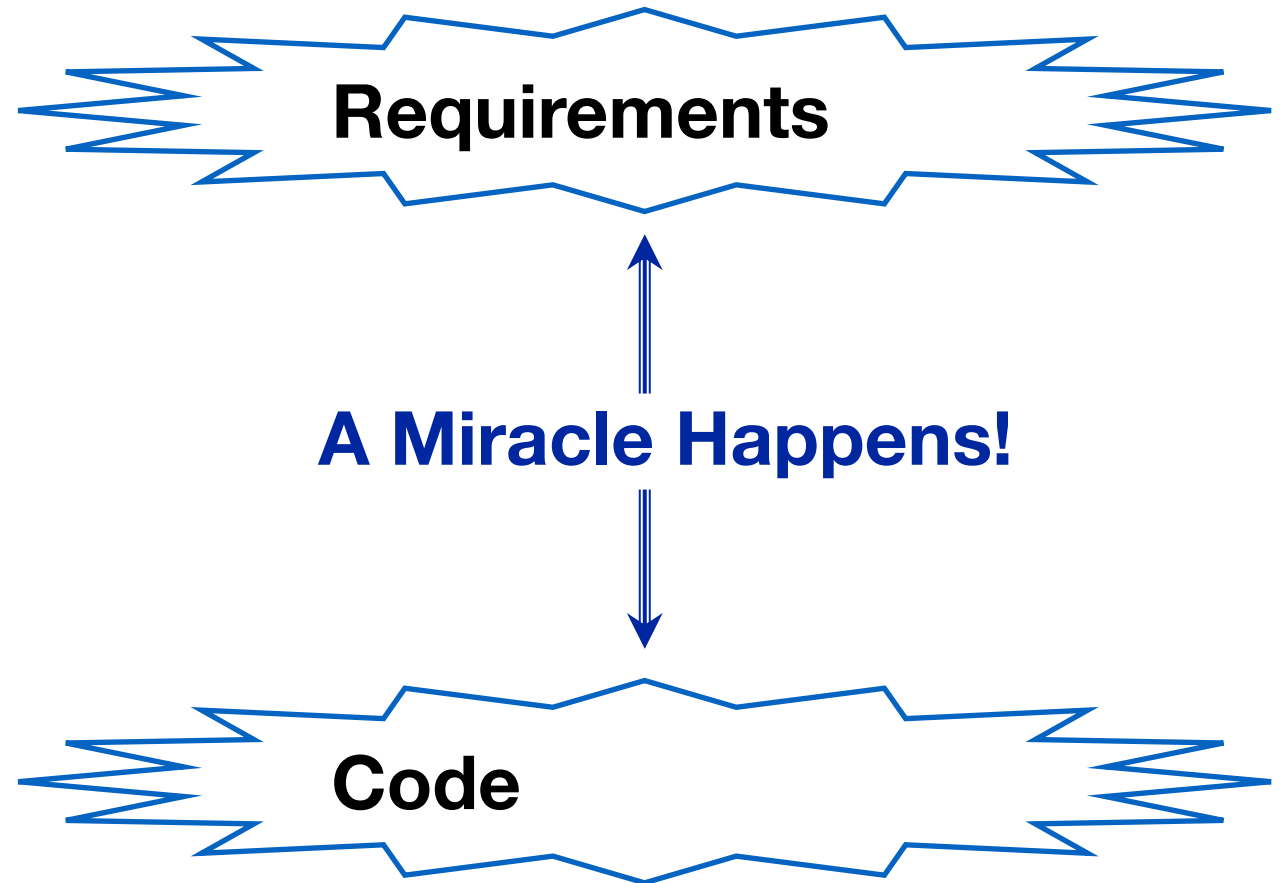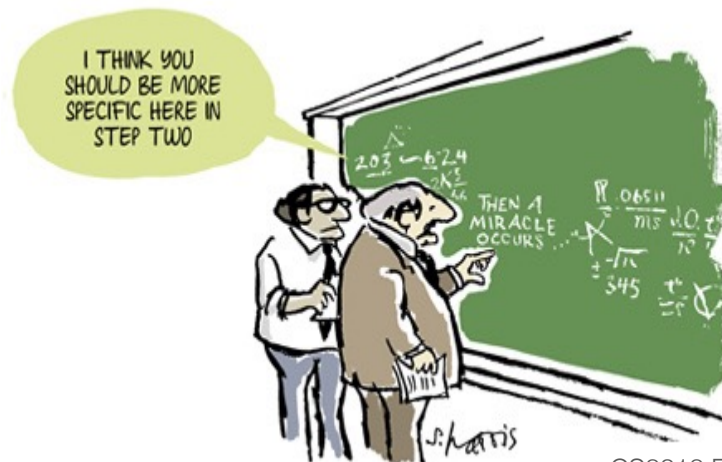# Any remaining question about requirements engineering?

Comments to Software Architecture
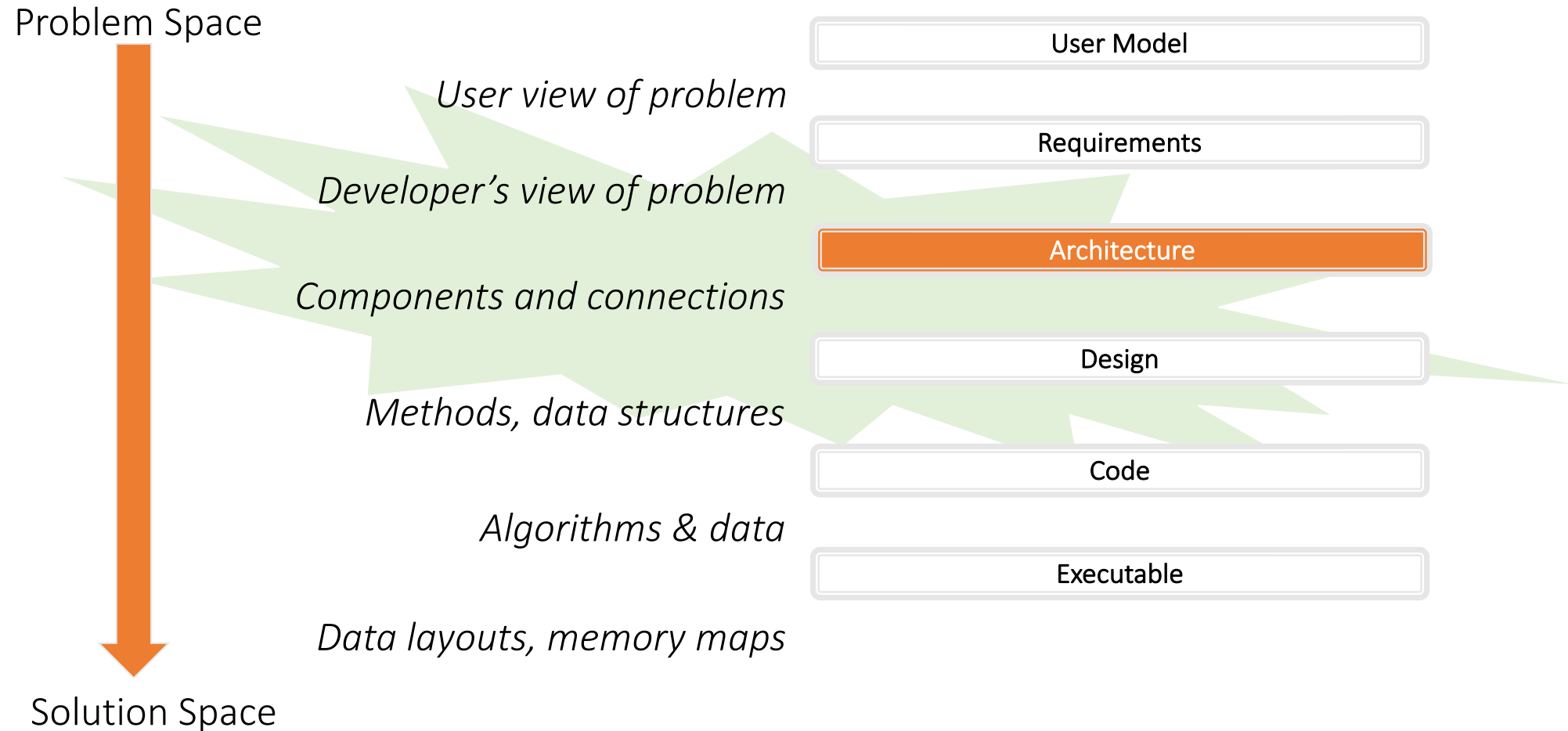
How to bridge the gap between requirements and code?

Requirements

???

Code
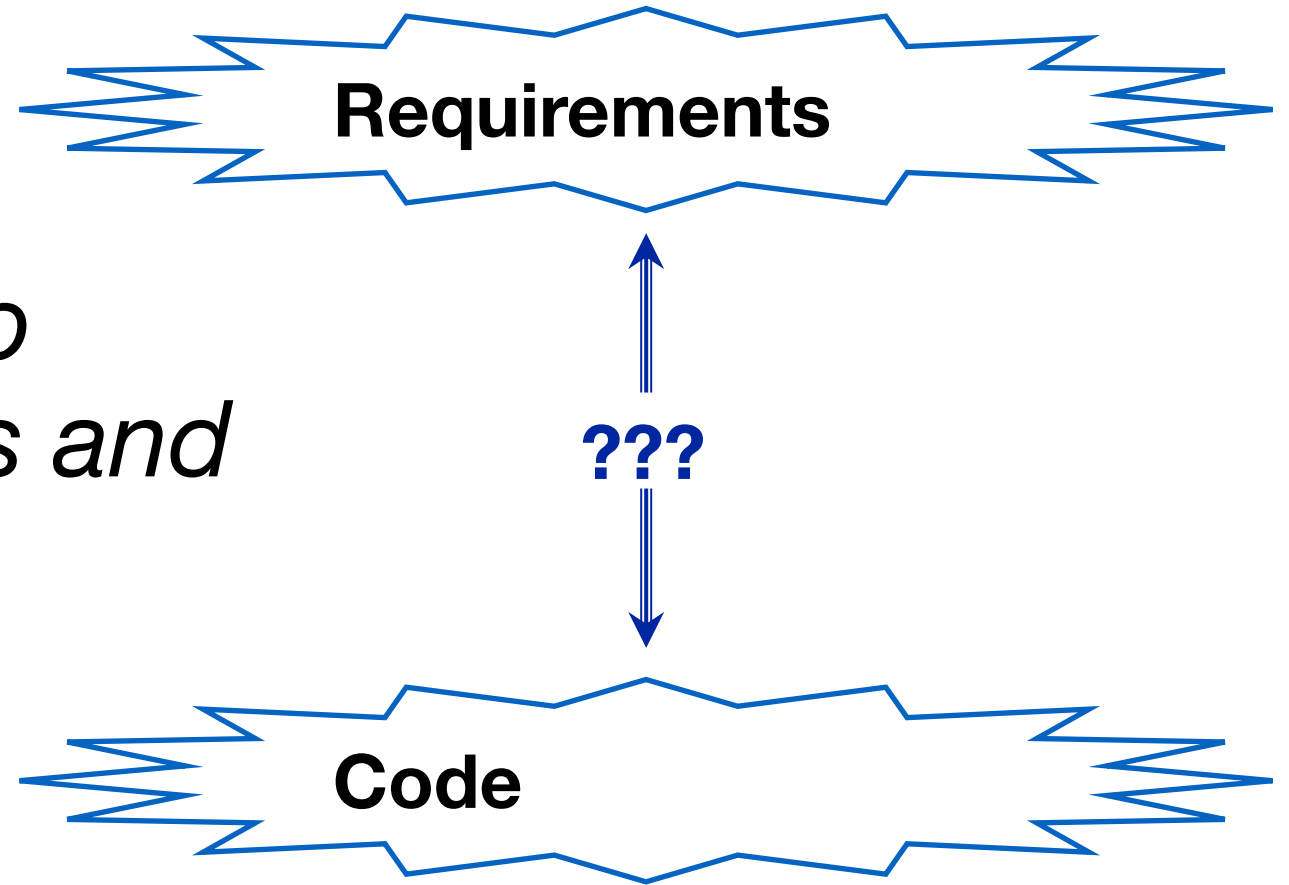
The Traditional Answer

- ❏ Ad hoc
- ❏ Requires gurus
- ❏ Unpredictable
- ❏ Costly

Requirements

A Miracle Happens!

Code

Role of Software Architecture: Bridging the Gap

Problem Space

User Model

User view of problem

Requirements

Developer's view of problem

Architecture

Components and connections

Design

Methods, data structures

Code

Algorithms & data

Executable

Data layouts, memory maps

Solution Space

Architecture as Tool for Managing Complexity

→ is a result of software engineering challenges and software's nature (overlay of all challenges, imprecise specifications, …)

Let's view *Software Architecture* as *conceptual tool* for dealing with the **complexity**. That is, architecture is a set of concepts which impose **order on complexity**.

Stuctures!

- ❏ What elements are there?
- ❏ How are they interconnected?
- ❏ What does the connection mean?
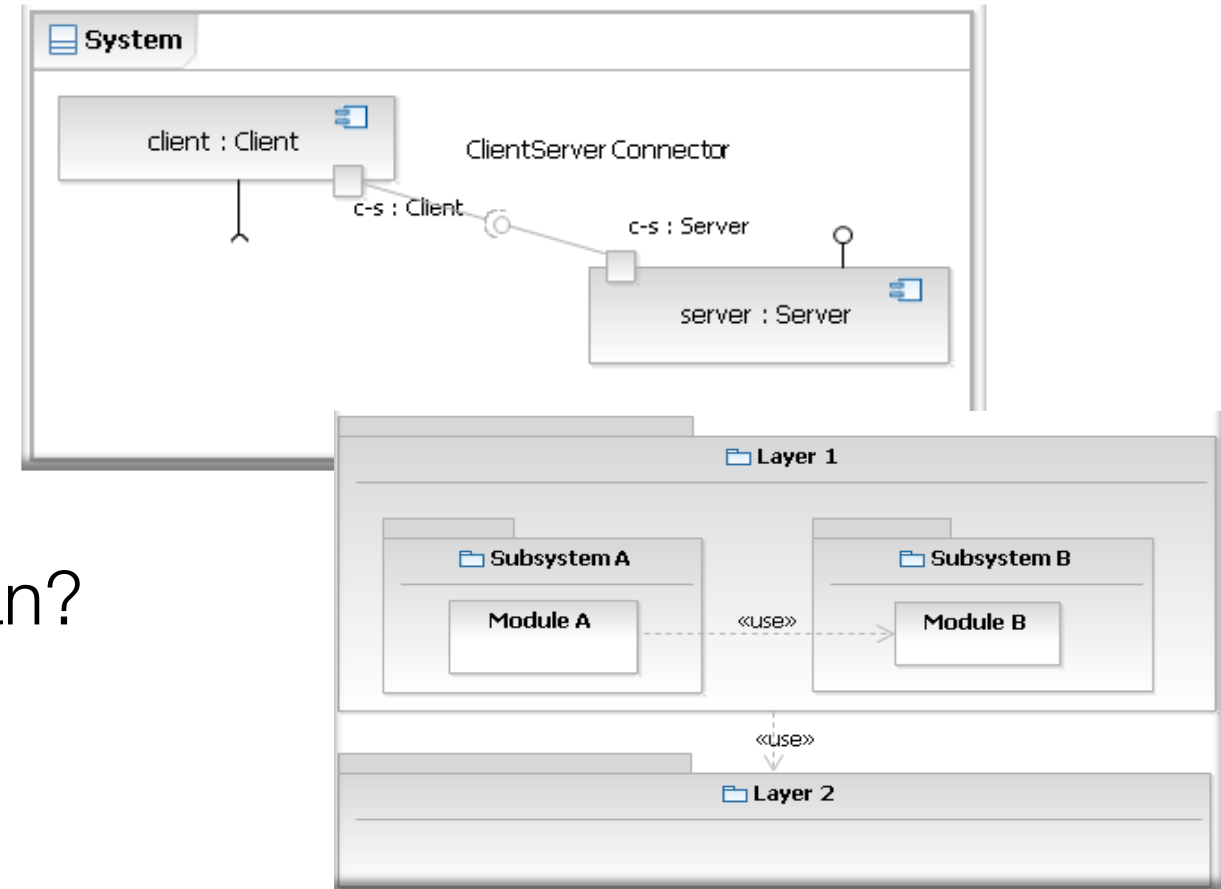
Overall conceptual idea:
→ Each part can be built fairly independently of the other parts
→ However, these parts must be put together to solve the larger problem in the end
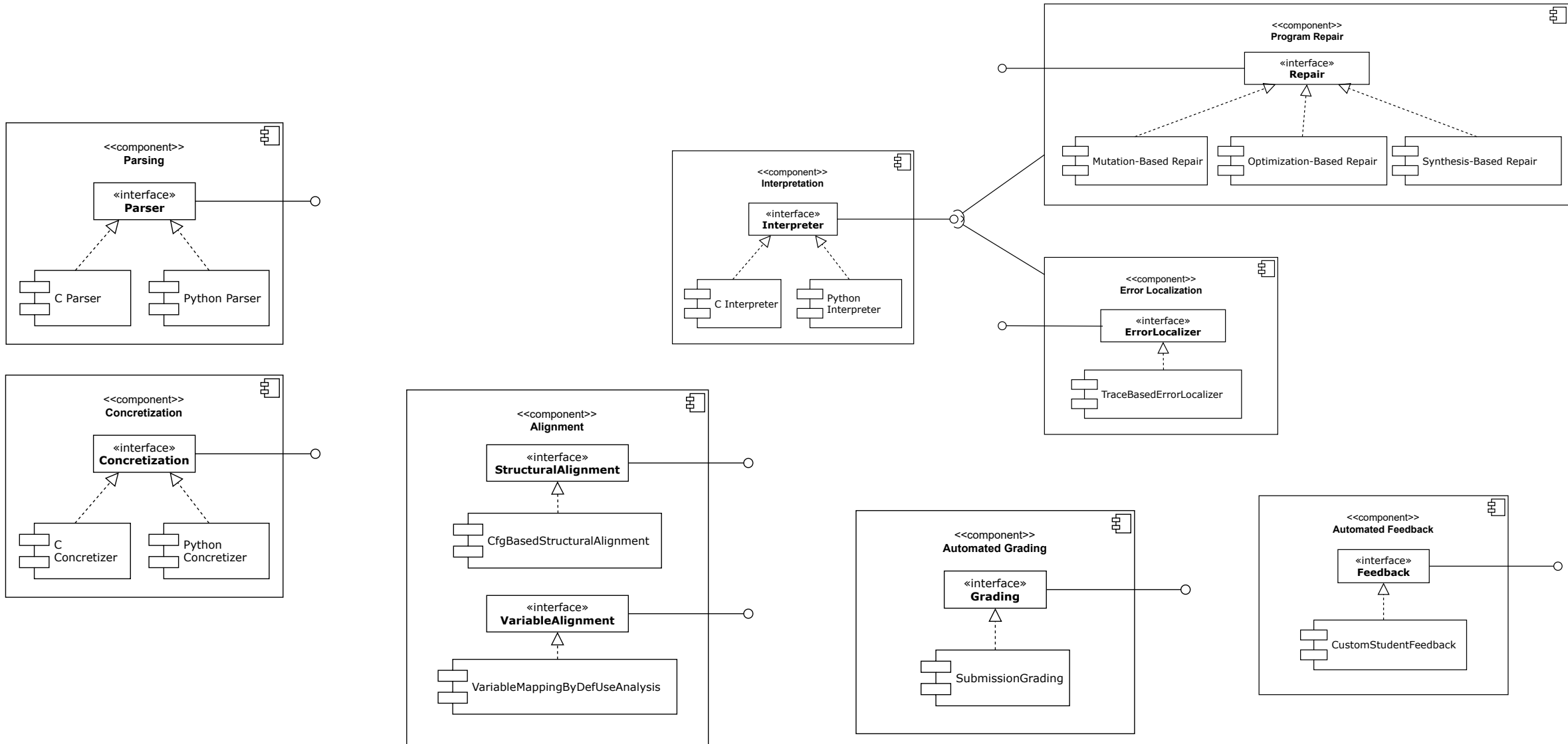
Static Structures: Focus on Components

❏ E.g. layered architecture

❏ Typical example often found in practice

❏ Meaning of boxes and lines is not defined

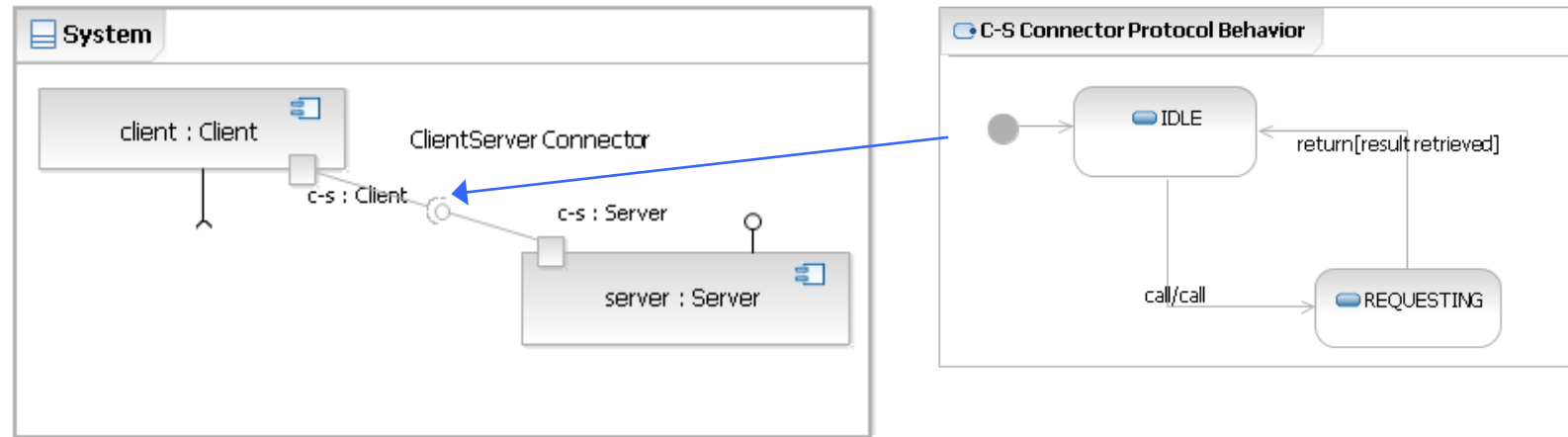❏ Incomplete representation of relationships existing in the presented structure
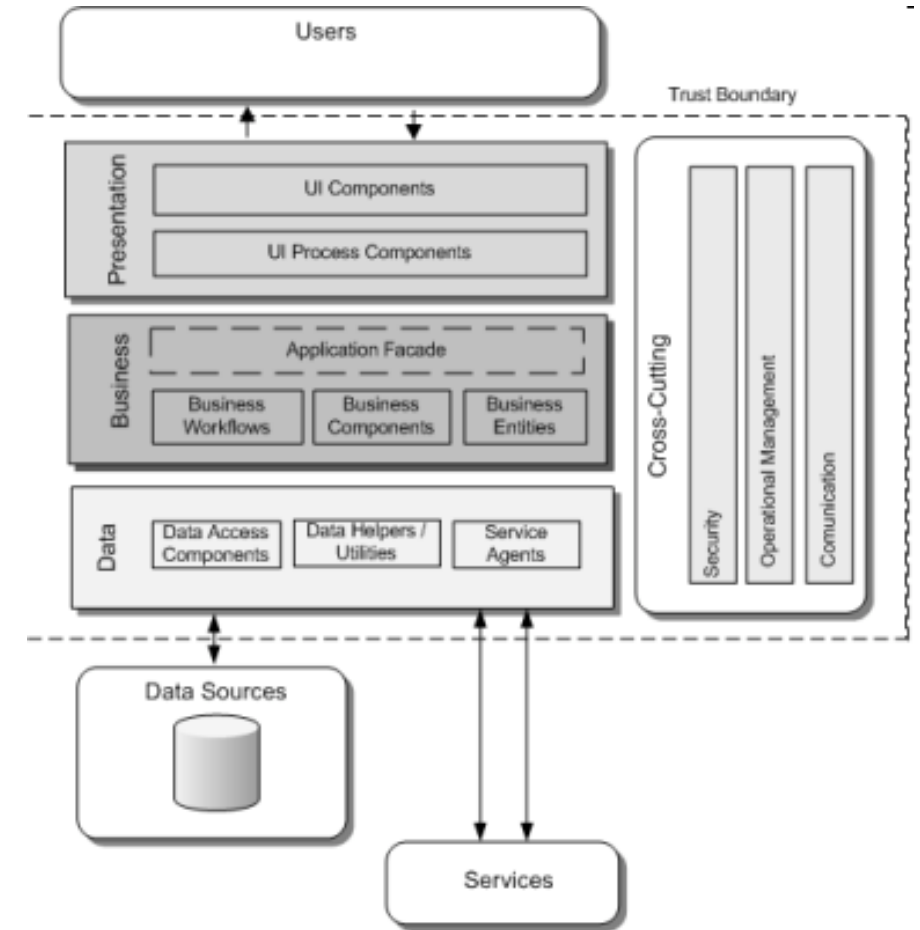
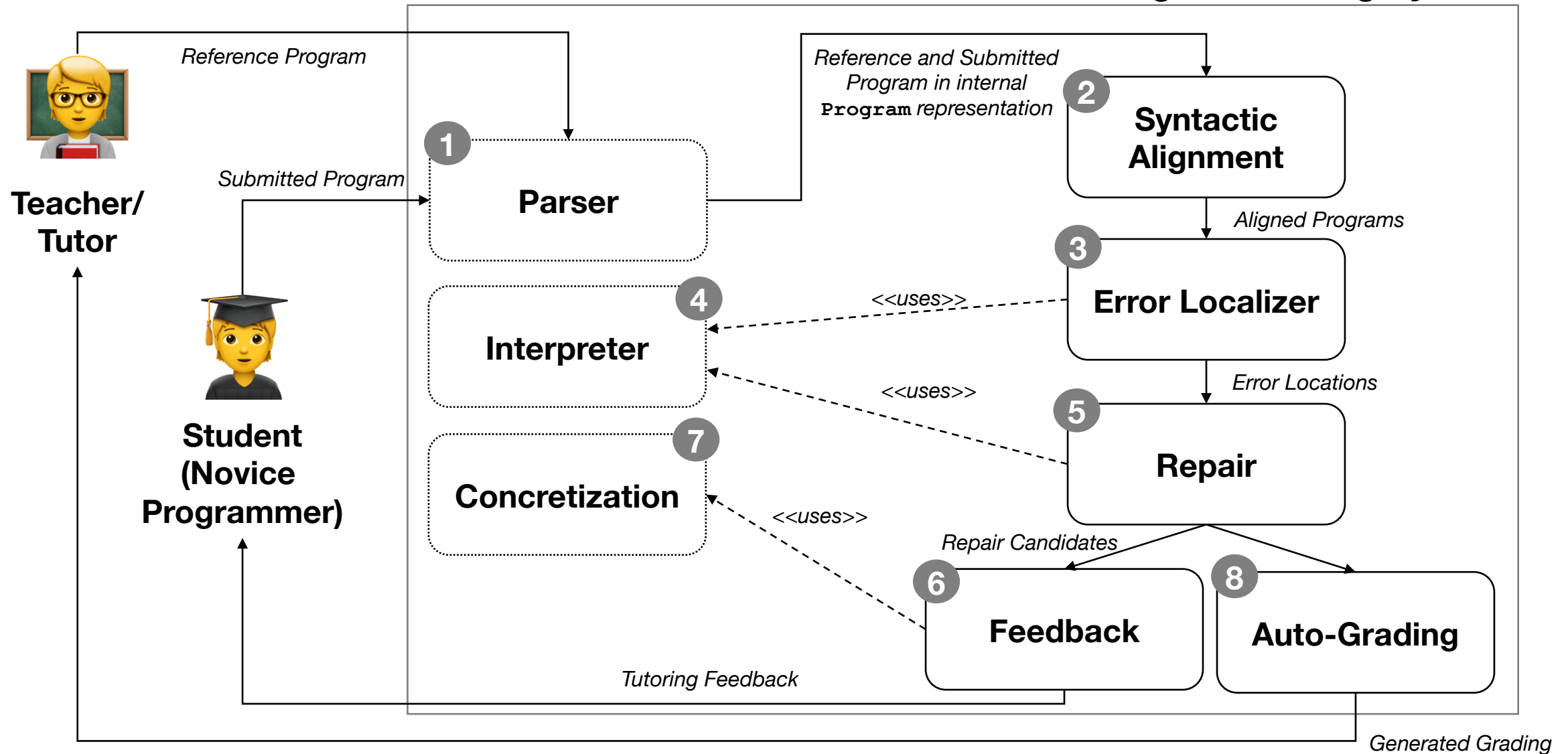[Source: Microsoft]

Static Structures

Dynamic Structures: Focus is On Connectors

❏ What is behind relationships?

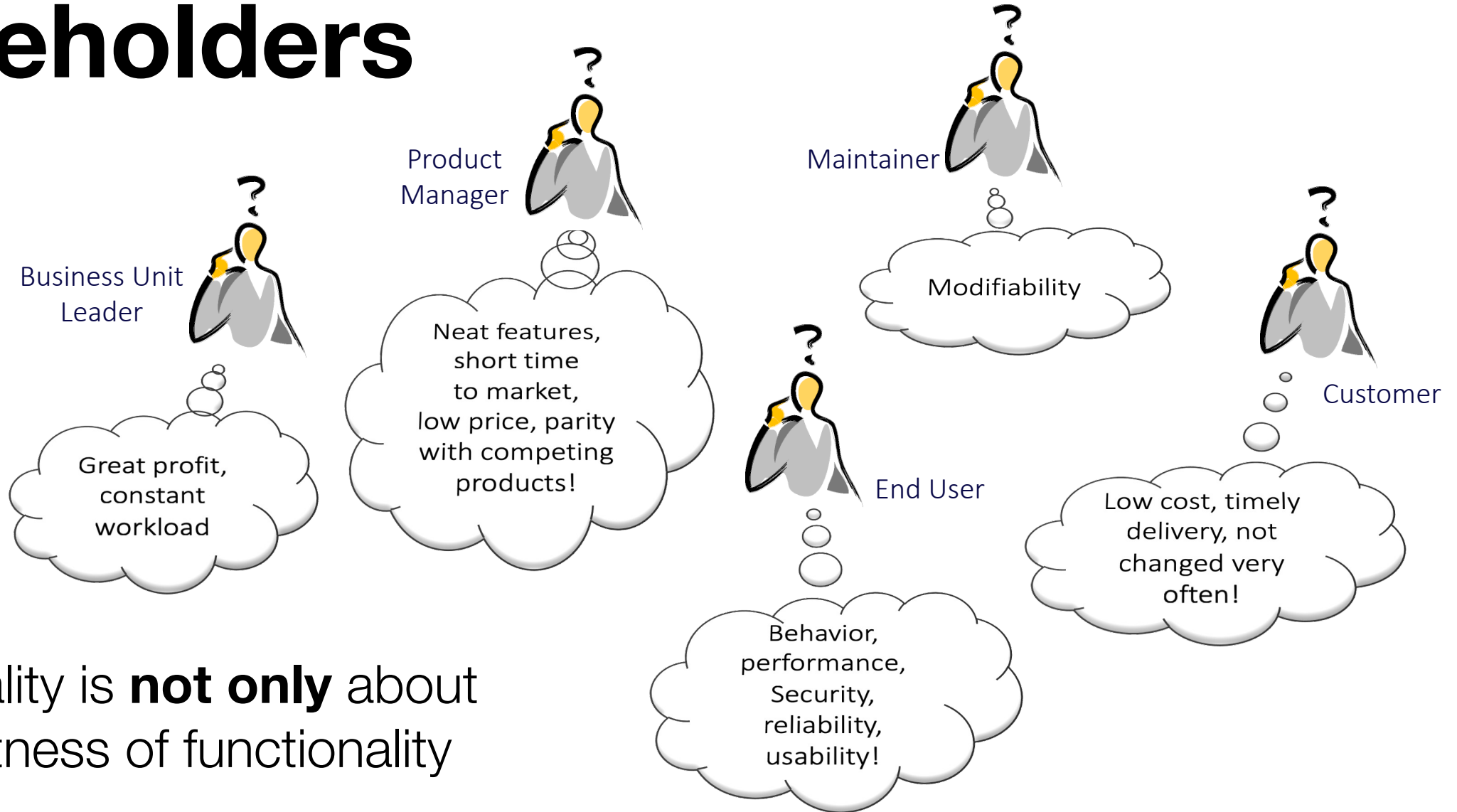❏ How do elements communicate?

❏ What assumptions do elements make?

Dynamic Structures (informal)

Intelligent Tutoring System

Teacher/ Tutor

Reference Program

Submitted Program

Student (Novice Programmer)

1 Parser

Reference and Submitted Program in internal `Program` representation

2 Syntactic Alignment

Aligned Programs

3 Error Localizer

<<uses>>

4 Interpreter

<<uses>>

Error Locations

5 Repair

7 Concretization

<<uses>>

Repair Candidates

6 Feedback

8 Auto-Grading

Tutoring Feedback

Generated Grading

Potential Concerns of Some Stakeholders

Business Unit Leader

Great profit, constant workload

Product Manager

Neat features, short time to market, low price, parity with competing products!

Maintainer

Modifiability

End User

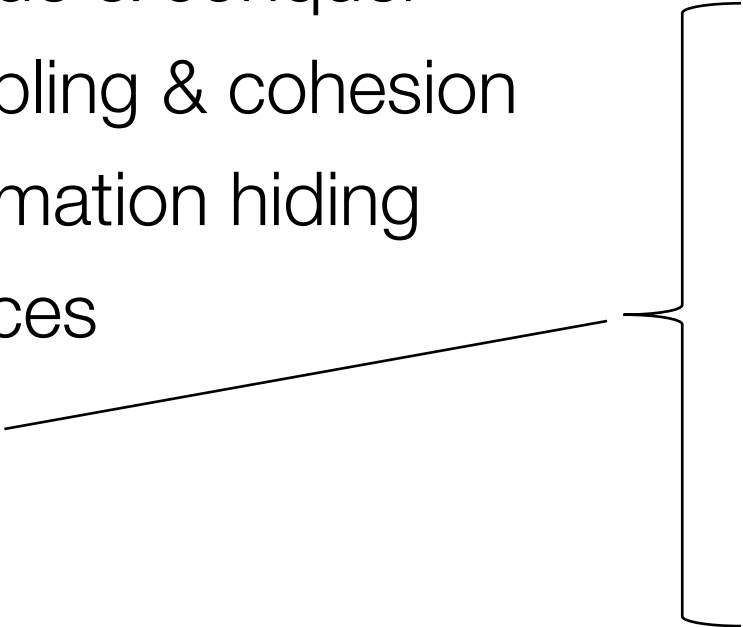Behavior, performance, Security, reliability, usability!

Customer

Low cost, timely delivery, not changed very often!

→ Quality is **not only** about correctness of functionality

Architecture Essentials – Design Principles

❏ Abstraction

❏ Separation of Concerns

❏ Decomposition: divide & conquer

❏ Modularization: coupling & cohesion

❏ Encapsulation: information hiding

❏ Well-Defined Interfaces

❏ Architectural Styles

- Pipe-and-Filter
- Shared-Data
- Publish-Subscribe
- Client Server Style
- Peer-to-Peer Style
- Communicating-Processes Style

Define Architecture: Design and Communication
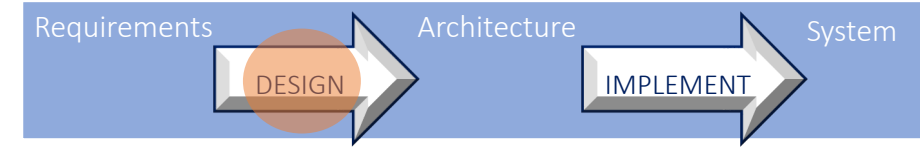
❏ Design is driven by a single architect (or a small group of architects with an identified leader)

❏ Basic architectural design process

     ❏ Choose the **architectural drivers**

     ❏ Drivers are derived from requirements most highly ranked

     ❏ Drivers thus combine specific set of functional and quality requirements that will dominantly 'shape' the architecture

❏ Choose an architectural style

❏ Instantiate module types and allocate functionality

Architectural Drivers

❏ **Business goals**

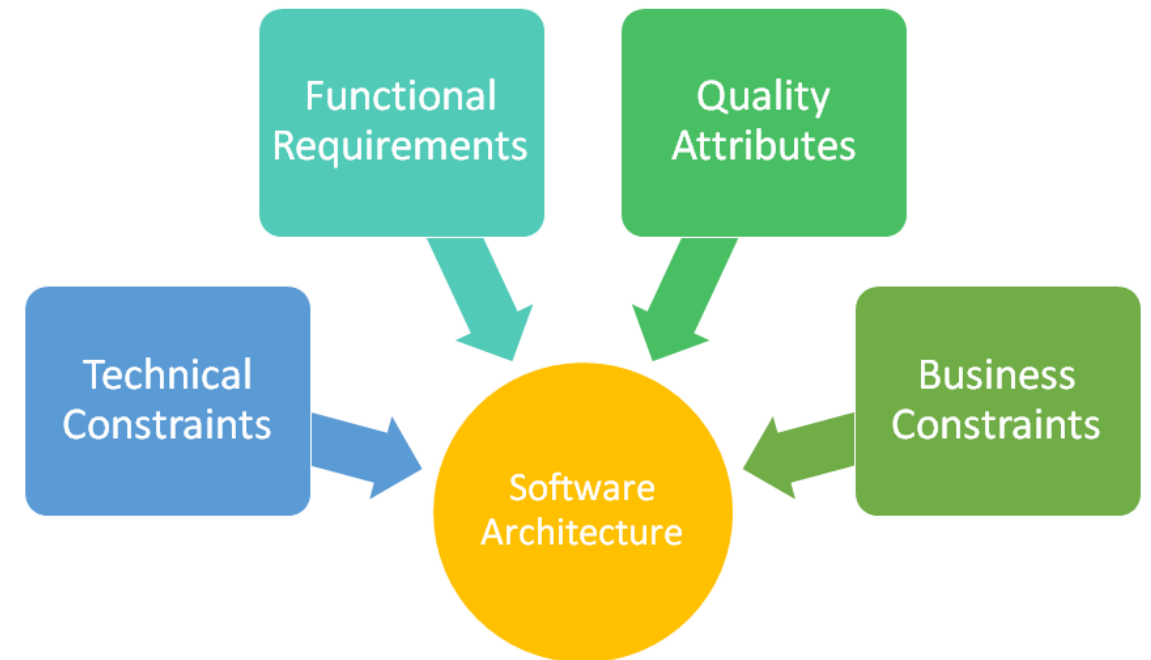   ❏ Customer organization

   ❏ Developing organization

❏ **Quality attributes**

❏ **Key functional requirements**

   ❏ Unique properties

   ❏ Make system viable

❏ **Constraints**

   ❏ Organizational and technical

   ❏ Cost and time

https://medium.com/@janerikfra/architectural-drivers-in-modern-software-architecture-cb7a42527bf2

Conclusion

❑ Requirement Analysis, Elicitation, and Modeling **needs training**

❑ Next step: entering the **solution space**

Next Week:
Chinese New Year

➤ ITS Architecture is already shared

➤ Assignment 4:
   (a) Module Design
   (b) Strategy Planning

In two weeks:
Module Design & Project Planning