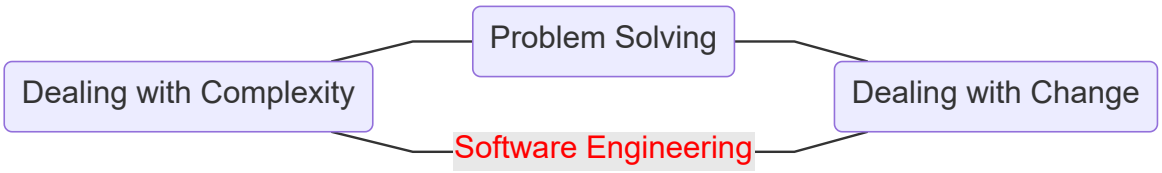
Robin Weiland

EIST Summary

01 Introduction



Abstraction

> *The 7 ± 2 phenomena*
> *Our short term memory cannot store more than 7 ± 2 pieces at the same time*
> *→ chunking: Group collection of objects to reduce complexity*

- thought process ⟶ **activity**
- result ⟶ **entity**

> *Abstraction as a model of a priorly, currently or not yet existing system.*

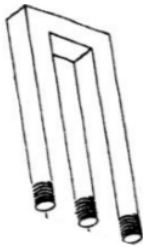
Typical Models

Object Model	**entities**	**structure** of the system?
Functional Model	**use cases**	**functions** of the system?
Dynamic Model	**activities**	system's **reaction to external events**

> *System model: object model + functional model + dynamic model*

Difficulties in software development

- Problem's **ambiguity**

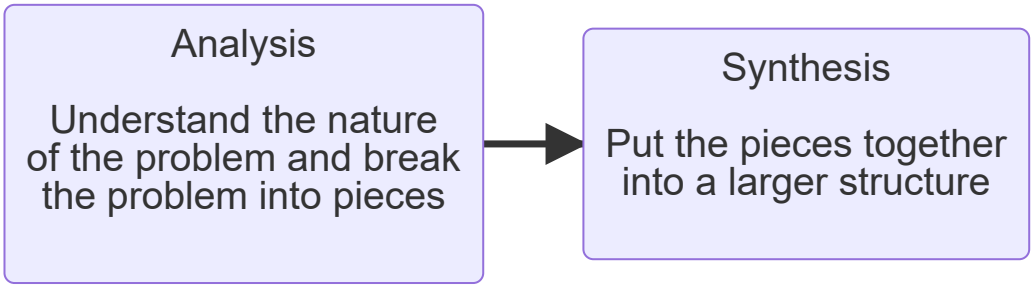


impossible trident

- **Requirements** are usually **unclear** and **change** when they become clearer

- The **problem domain** (also **application domain**) is **complex**, and so is the **solution domain**
- The **development process** is **difficult to manage**
- Software is a **discrete** system
    - Continuous systems have no hidden surprises
    - Discrete systems can have hidden surprises! (Parnas)

Software Engineering as a problem solving activity



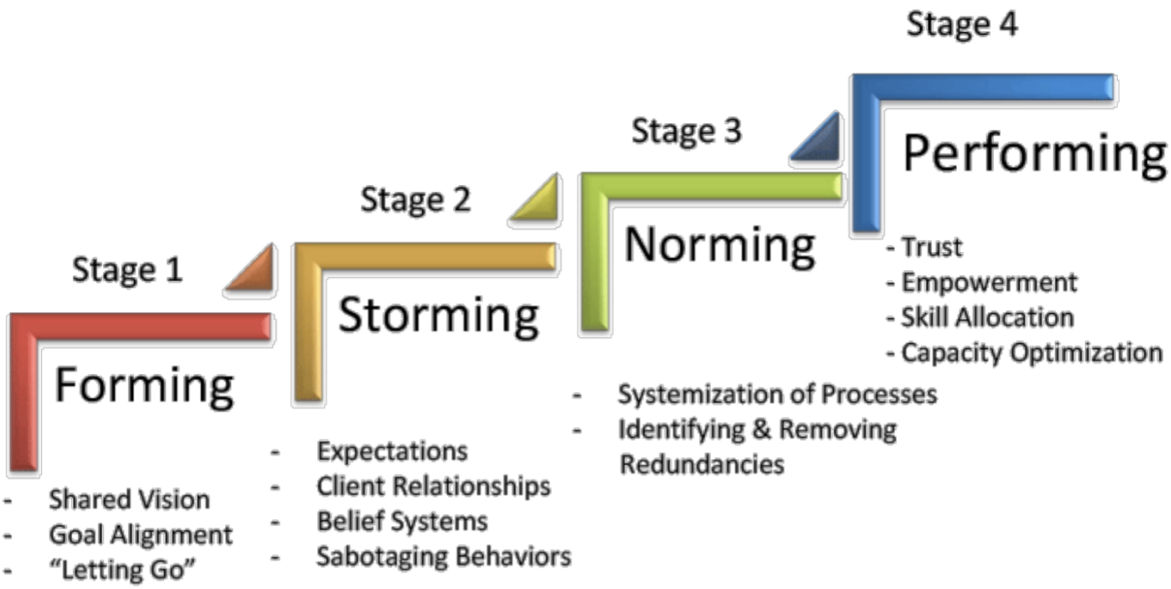
Techniques, methodologies and tools

Techniques	methodologies	tools
Formal procedures for producing results using some **well defined notation**	**Collection of techniques** applied across software development and unified by a philosophical approach	**Instruments or automated systems** to accomplish a technique
recipe, quick sort algorithm	cookbook, object oriented analysis and design, functional decomposition	compiler, editor, debugger, IDE, CASE

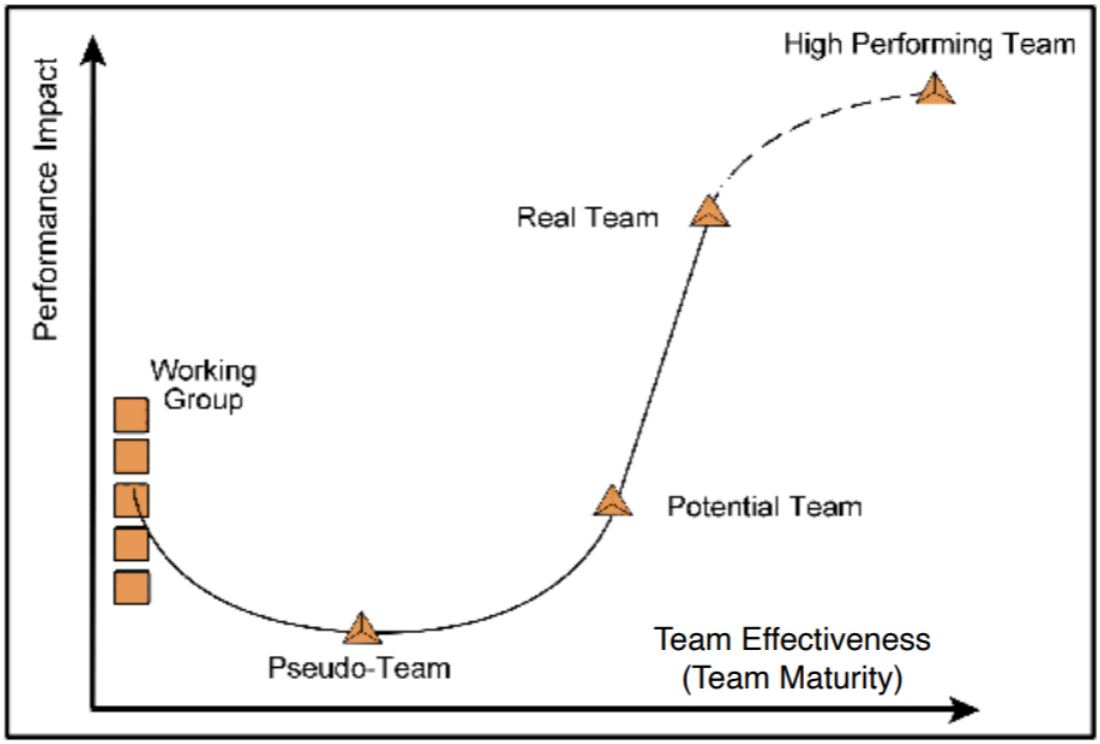
[Team]

I don't know whether that's important for the exam, but it feels like some bs they could ask in a bad mood.

Stages of team development



Team productivity and performance



Forming > Storming > Norming > Performing

Difference between group and team

- Group a number of people that have some relationship to one another
    - Participants are loosely connected
    - Do not focus on specific outcomes or a common purpose
    - Every individual works on his own
- Team any group of people involved in the same activity with a common goal, especially referring to sports and work
    - Participants are strongly connected
    - Focus on a specific outcome, requires coordination of tasks and activities
    - Members need to work together

Phenomenon vs. Concept

Phenomenon
: An **object** in the **world of a domain** as it is **perceived**
This EIST lecture at 9:25 *my black watch*

Concept
: **Common properties** of phenomena
All lectures on software engineering *All black watches*

	A concept is a 3-tuple
Name	The name distinguishes the concept from other concepts
Purpose	Properties that determine if a phenomenon is a member of a concept
Members	Set of phenomena which are part of the concept

> ***Abstraction:*** *classification of phenomena into concepts*
>
> ***Modelling:*** *development of abstractions to answer specific questions about a set of phenomena while ignoring irrelevant details*

Systems, models and views

Systems

- Organized set of communicating parts
  - Natural system: a system whose ultimate purpose is not known
  - Engineered system: designed and built by engineers for a specific purpose
- The parts of the system can be considered as systems again
  - In this case we call them subsystems

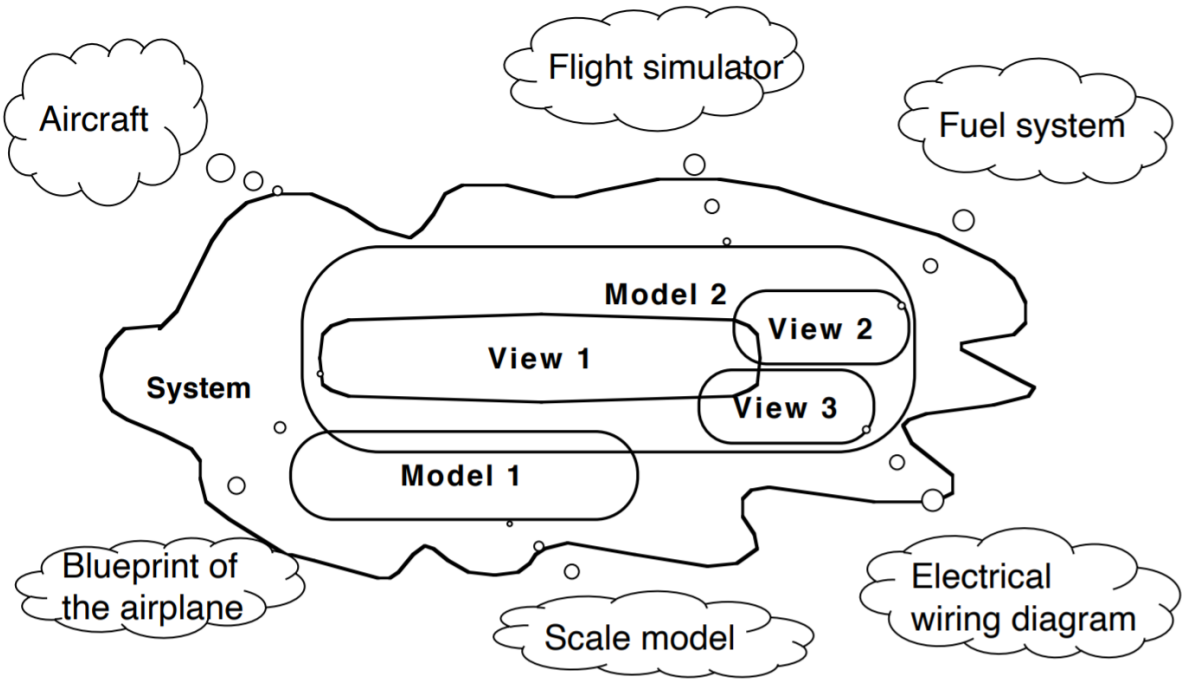
natural systems	**engineered systems**	**subsystems**
universe, earth, ocean	airplane, watch, GPS	jet engine, battery, satellite

Model and View

- Model: Abstraction of a system
- View: Selected aspects of a model
- Notation: Set of Graphical or textual rules for depicting models and views
  - Informal ("napkin design")
  - Formal (UML)

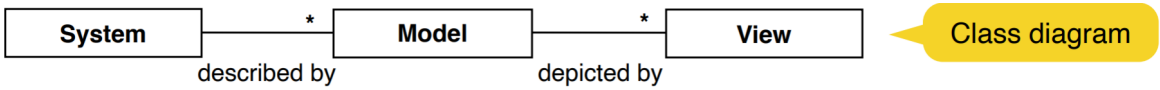
System	Model	View
Airplane	Flight simulator Scale model	Blueprint of the airplane electric wiring diagram an airplane breaking the sound barrier

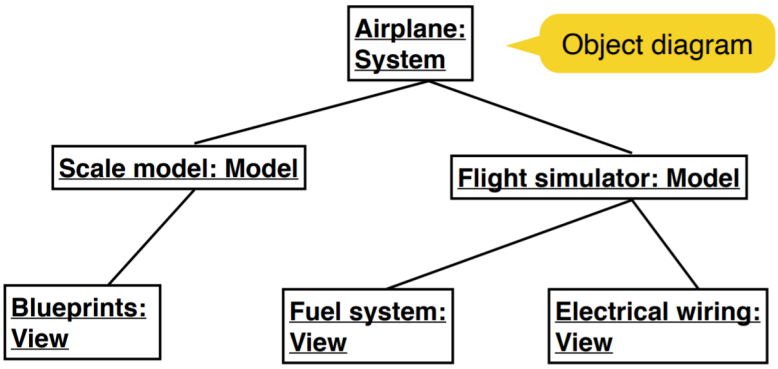
Napkin Notation



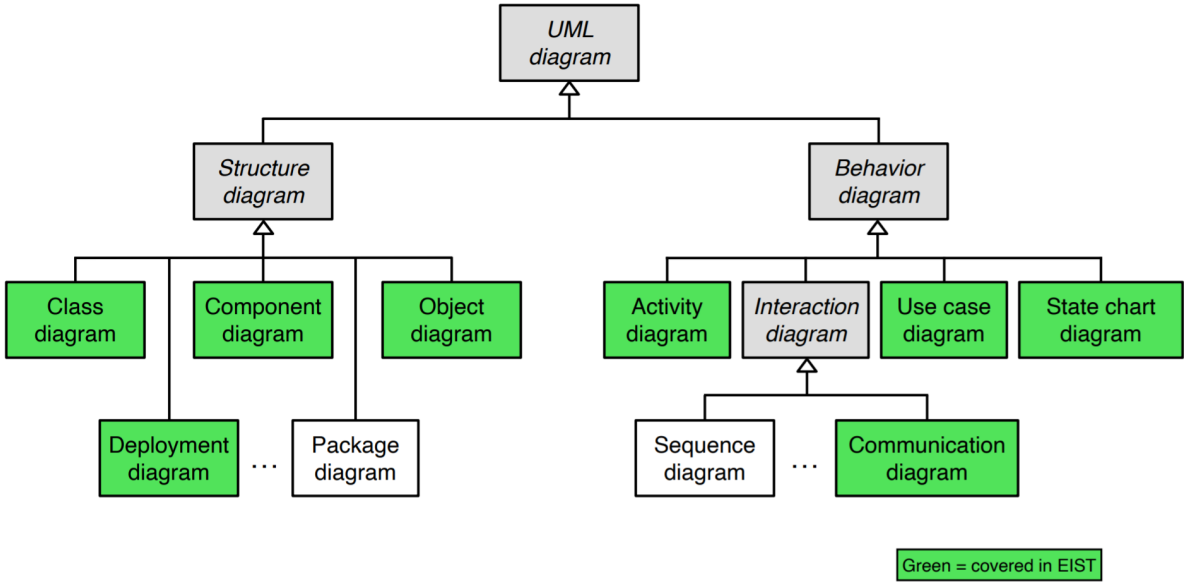
Views and models of a complex system usually overlap

UML Notation





Overview of UML diagrams



OOP principles (some with maybe not so obvious terminology)

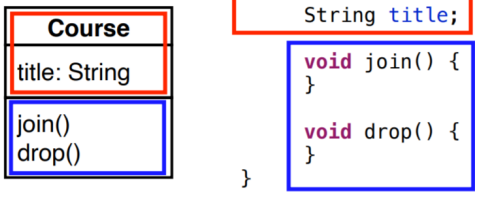
Type & instance

name	purpose	members
int	integral number	$\mathbb{N}_{\text{in memory range}}$
boolean	logical	{true, false}

These relationships are similar

- Type $\longleftrightarrow$ variable
- Concept $\longleftrightarrow$ Phenomenon
- Class $\longleftrightarrow$ Object

Encapsulation

> *Encapsulation means creating classes for such objects to define*
>
> - *Structure / state by using attributes*
> - *Functionality / behavior by providing methods*
>
> Java supports encapsulation by using classes with attributes for structuring and methods for describing functionality
>
> **Problem Statement:** "A course has a title, students can join and drop the course."
>
> 
>
>
> ```
> class Course {
>     String title;
>     void join() {
>     }
>     void drop() {
>     }
> }
> ```

02 Model based Software Engineering

03 Requirements Analysis

04 System Design I

05 System Design II

06 Object Design I

07 Object Design II

08 Testing

1o Software Configuration Management

11.1 Software Quality Management

11.2 Guest Lecture: Dr. Elmar Jürgens

12 Project Management