

CS132: Software Engineering

Zhihao Jiang
SIST@ShanghaiTech

Lecture 1: Introduction

What is software engineering?

Science vs. Engineering

Science vs. Engineering

- Science: Theoretically how something can be achieved
 - How to generate Ammonia (NH_3)?
 - $N_2 + 3H_2 \leftrightarrow 2NH_3 \uparrow$
- Engineering: How to achieve the goal **efficiently** and **economically** with existing constraints
 - In 1774: $2NH_4Cl + Ca(OH)_2 \leftrightarrow CaCl_2 + 2NH_3 \uparrow + 2H_2O$
 - In 1898: $CaC_2 + N_2 \rightarrow CaCN_2 + C$ and $CaCN_2 + 3H_2O \rightarrow CaCO_3 + 2NH_3 \uparrow$
 - In 1908: $N_2 + 3H_2 \leftrightarrow 2NH_3 \uparrow$ under high pressure with catalysts

Software “Science”

- Algorithm
- Theory of computation
- What problems can be solved computationally?
- How can we express the solution rigorously without ambiguities?
 - Formal languages
- Can we solve problems efficiently?
 - Computational complexity theory

Software Engineering: Definition

- “The establishment and use of sound engineering principles in order to obtain economical software that is reliable and works efficiently on real machines ...” [Fritz Bauer, at the 1st NATO Conference on Software Engineering, 1969]

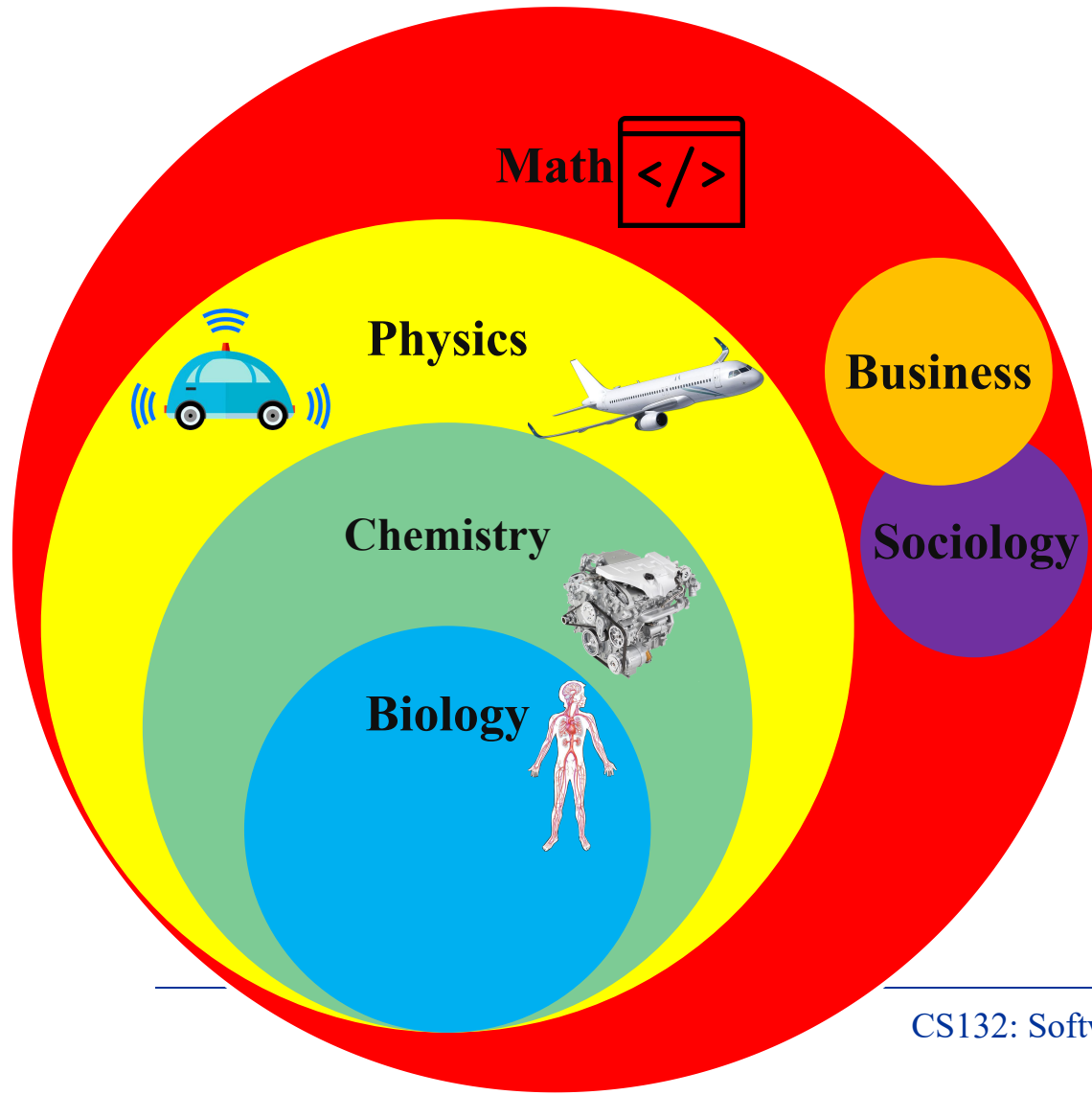
“The Software Crisis”

- The need to develop larger, more diverse software systems
- 28% of software projects are “success”
 - 51% seriously late, over budget and lacking expected features
 - 18% cancelled outright
- Increasing complexity
 - Use to be one single task on a specific computer
 - More functionalities
 - In more conditions
 - On more platforms
 - By more people

Software vs. Hardware Implementation



An Inter-disciplinary Field



- Software is in **math** domain
 - Amazon negative quantity bug
- The domain specific constraints have to be encoded or considered in the software
- The domains with more constraints are also less understood

Which fake news is easier to tell?

“NASA说因为今天地球完美的重力角度，是唯一一天可以让扫把独自站立的日子。”

想看大家参与「NASA立扫把挑战」
BroomChallenge」的图 😂



Key Challenges in Software Engineering

1. Effective communication

- Between the engineering team and other stakeholders
- Within the engineering team

2. Risk Management

- How to balance conflicting judging criteria?

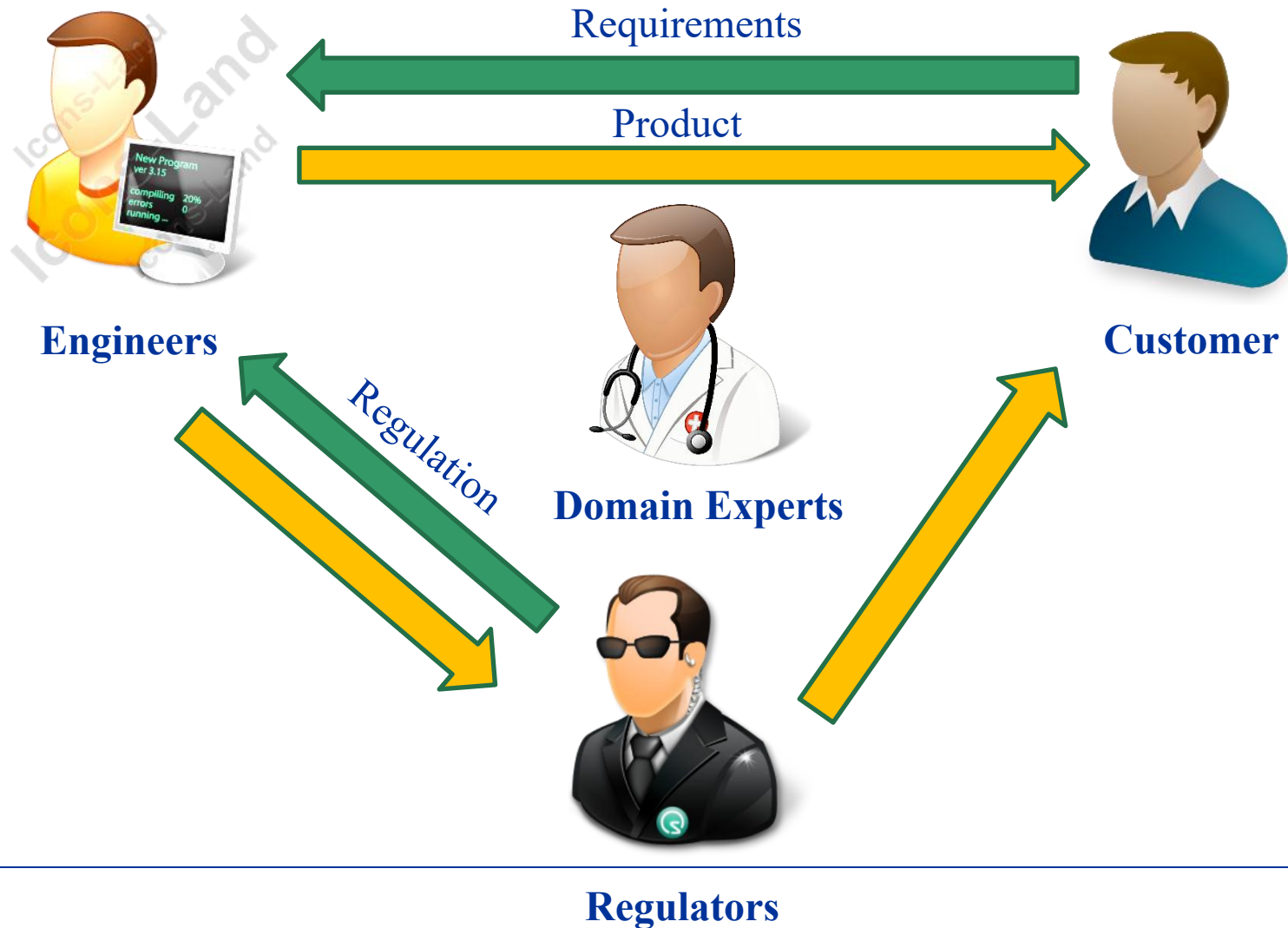
3. Validation

- How do you convince all stakeholders that the software is effective/safe/secure?

1. Effective communication

1. Effective Communication

Stakeholders for software



1. Effective Communication

Miscommunications

- The customer fails to explain their needs well.
- The customer may not know what he/she wants
- The analyst may not understand the customer's need.
- Your analyst may not convey the requirements to the development team

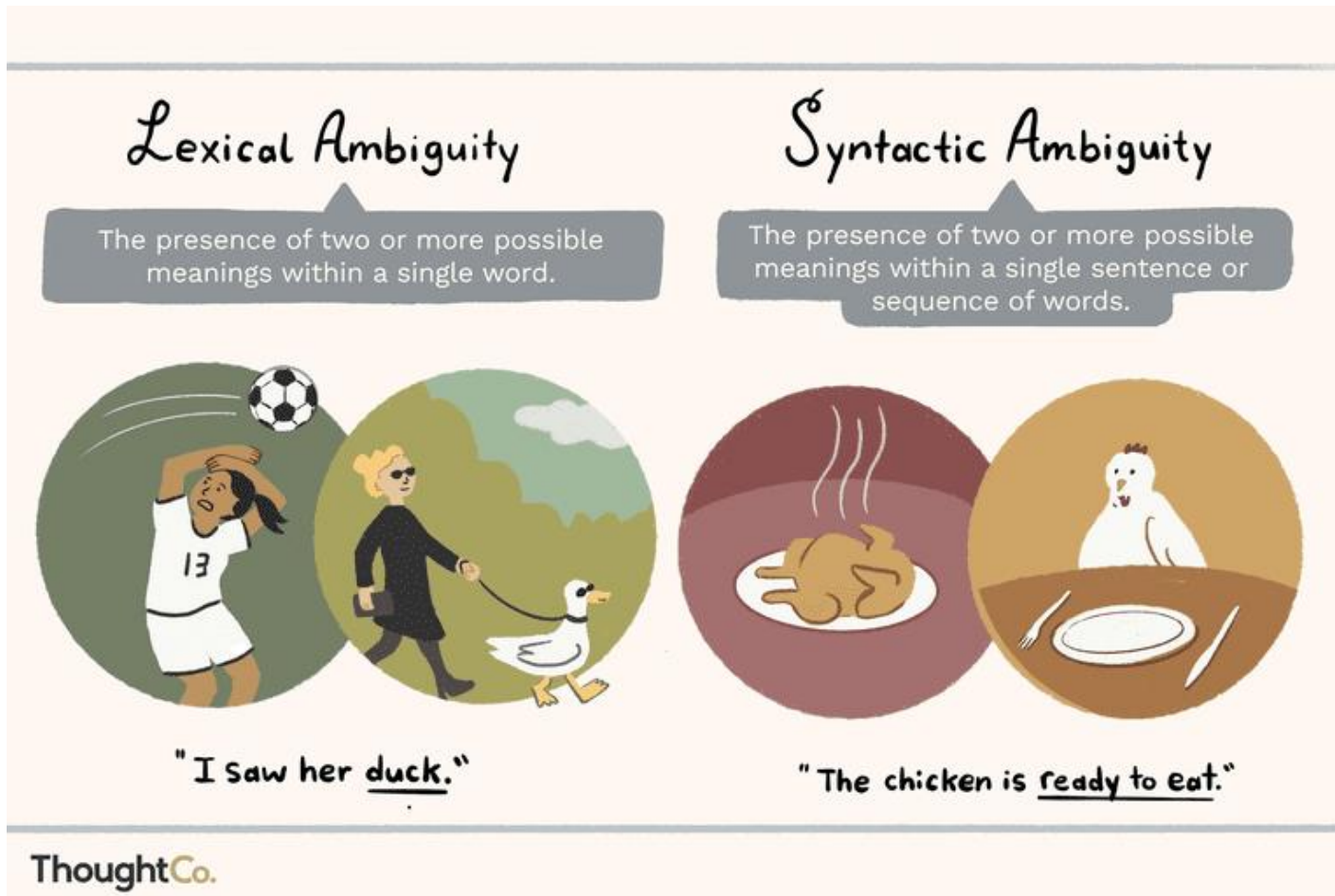


1. Effective Communication

Domain Knowledge: “The Expert”



Natural Languages Are Prone to Ambiguities



1. Effective Communication

Communications among various stakeholders

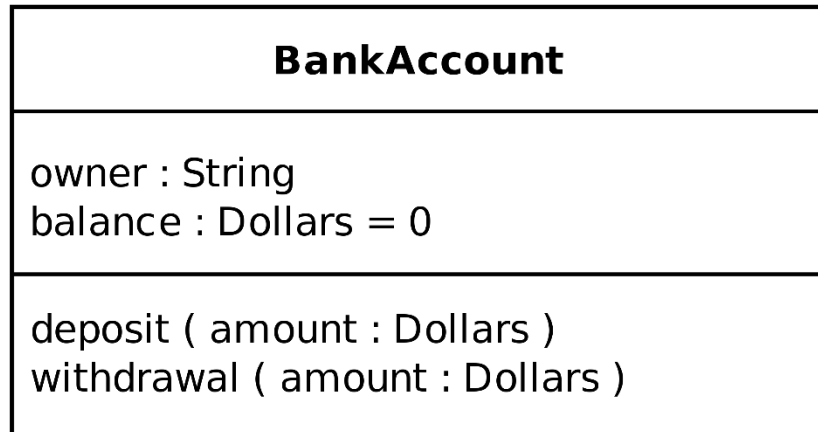
- Need a common language for communication
- Unified Modeling Language (UML)
- Recognized as an international standard
- It's just a tool, not a solution



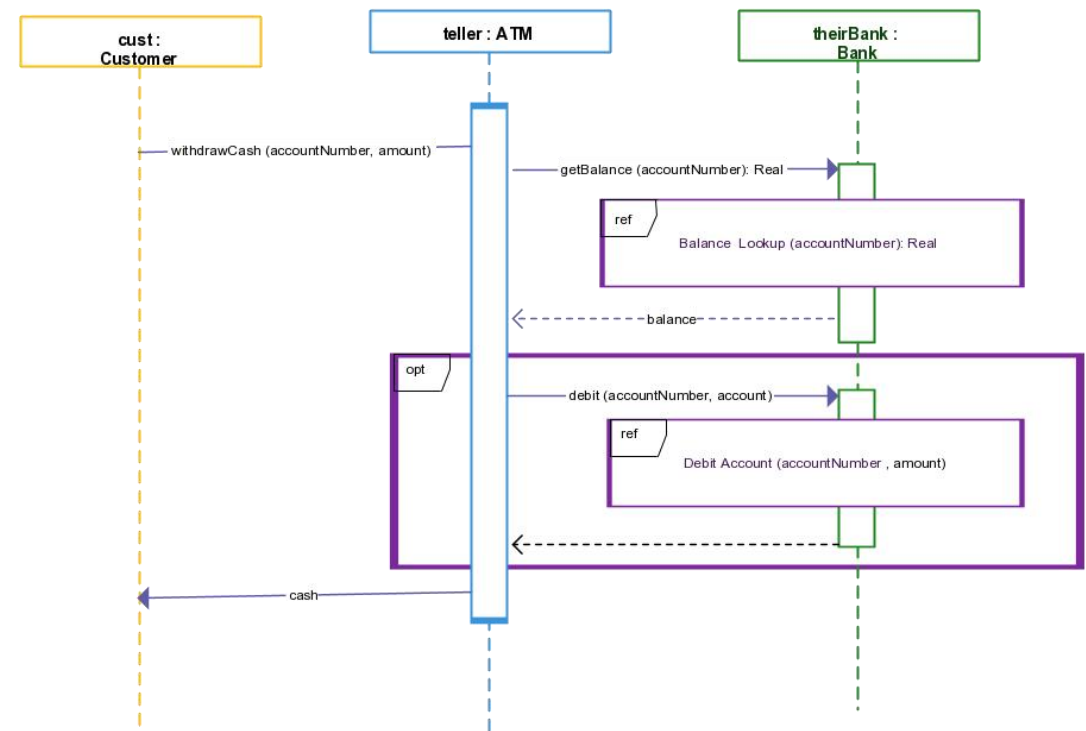
1. Effective Communication

Examples of UML Diagrams

- Structural diagram



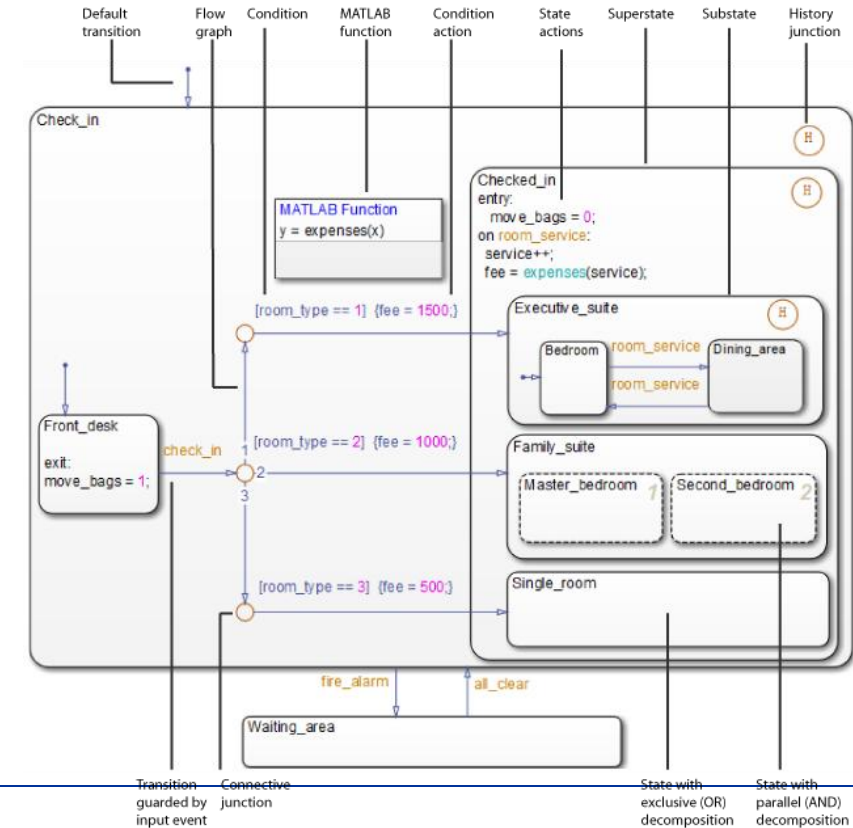
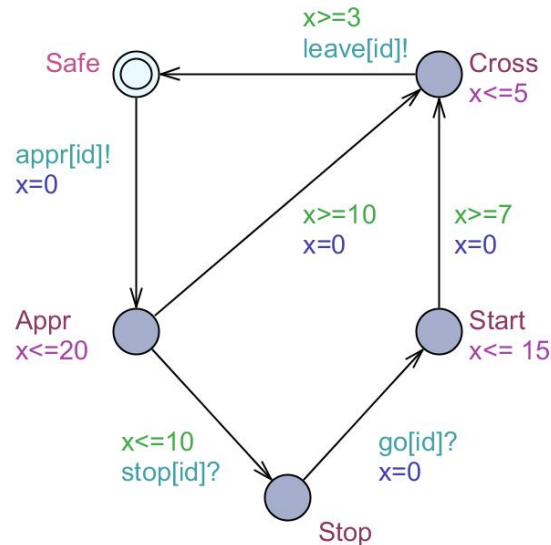
- Behavioral Diagram



1. Effective Communication

Communication within the team

- Formal models
 - Timed automata
- Simulation models
 - Simulink



The lifecycle of a successful software is long



- Other team members should understand your code and documentation
- Other team members should be able to easily “inherent” code from you

1. Effective Communication

What we want you to learn in this course

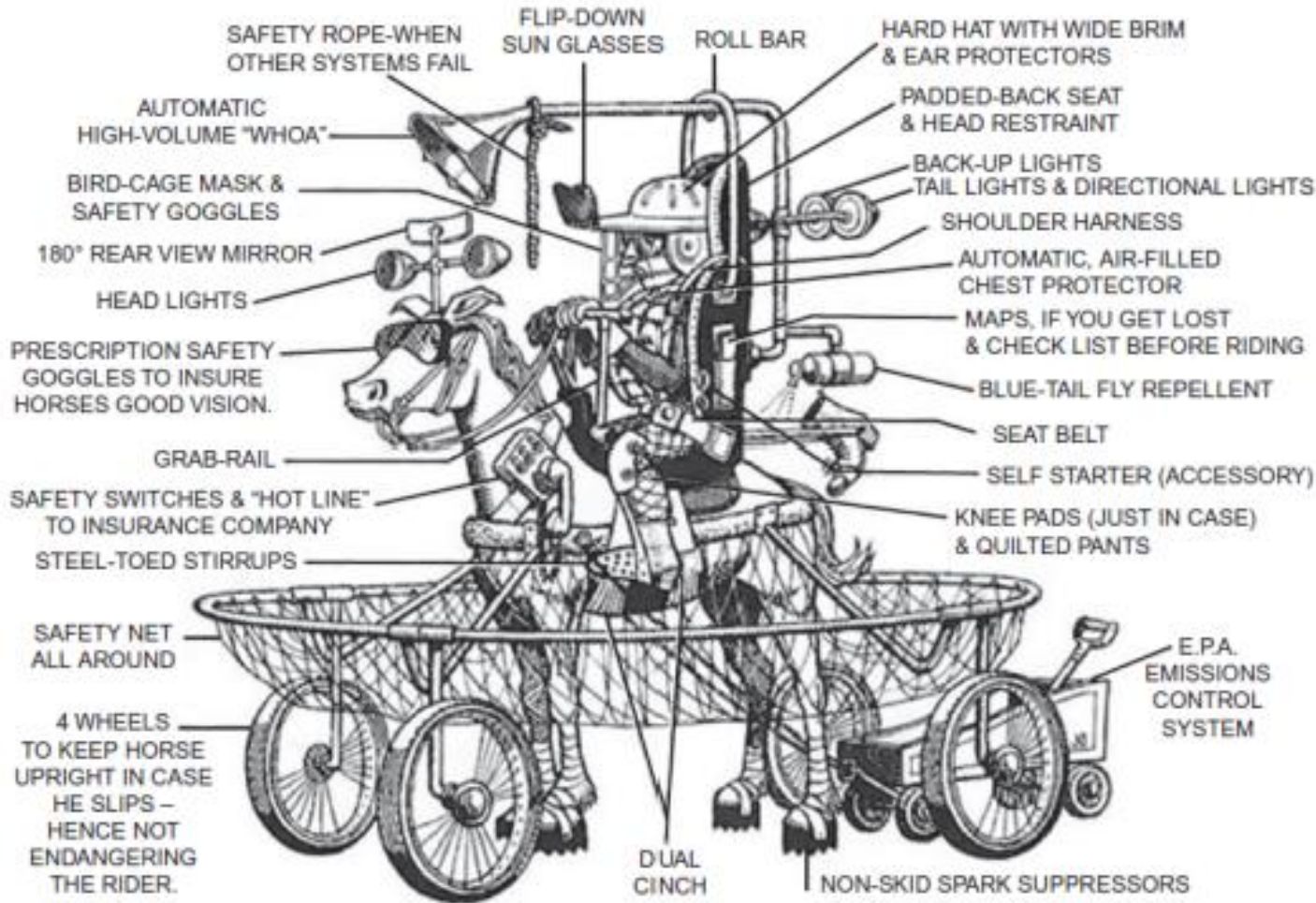
- Communication skills are important no matter what you do in the future
 - Documentation
 - Meetings and presentations
- Be mentally prepared
 - Respect other people's domain expertise
 - Accept that other people may not know what you know
- How to analyze problems in other domains?

2. Risk Management

Are they good aircrafts?



The “All-around” Solution

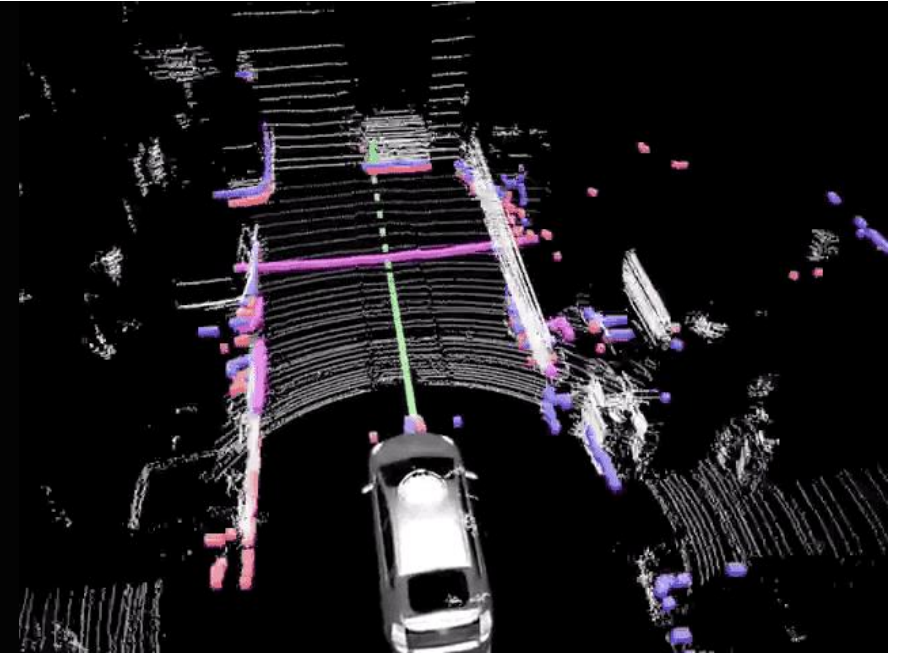


- Risk control measures may affect other system properties
- Justify that benefits outweigh the risks

2. Risk Management

Balancing among risks

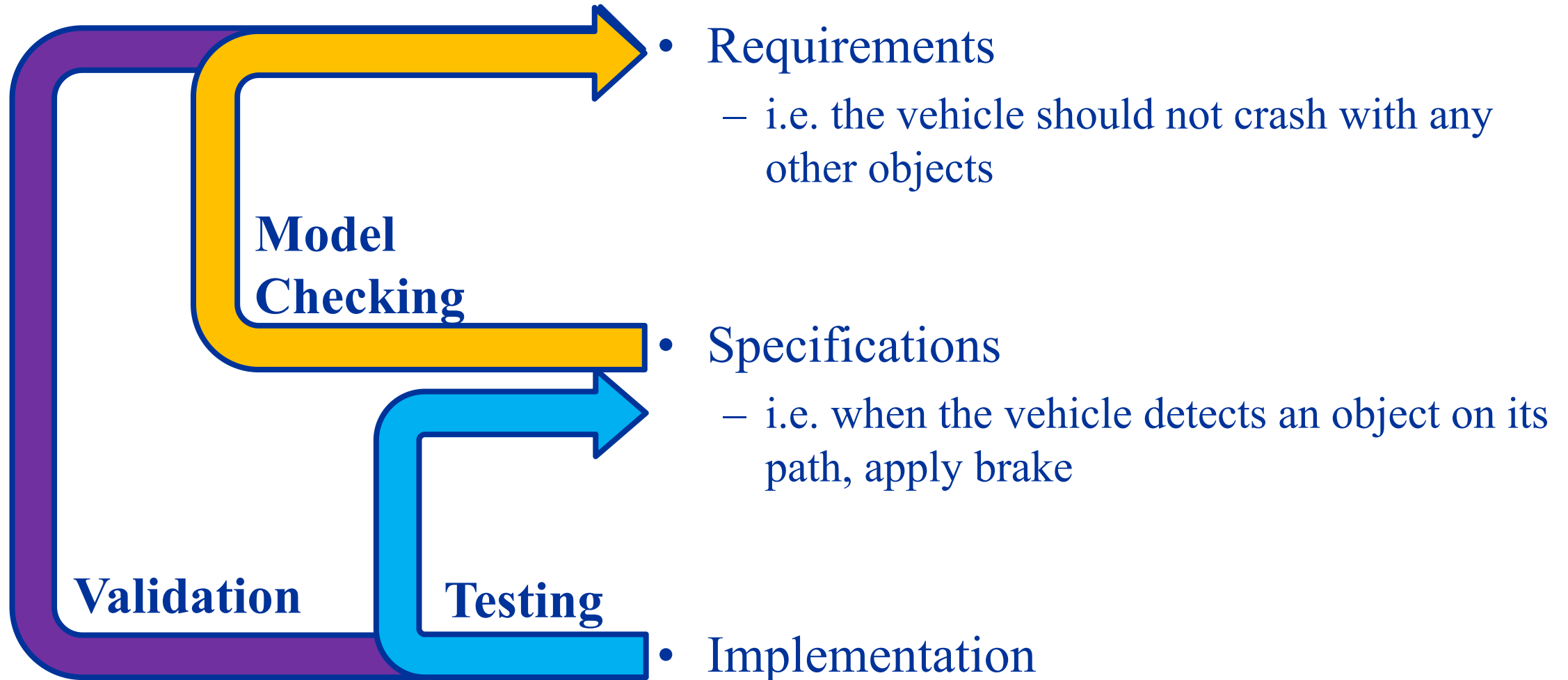
- Uber Autonomous Vehicle Accident (March 18, 2018)
- Pedestrian Identification: balancing false-positives vs. false-negatives



3. Validation

3. Validation

Software Development Process



3. Validation

What should we validate?

- Efficacy: The system can do its job as designed
- Safety: Under intended use, the system will not harm the user and its surroundings
- Security: Prevent malicious use of the system

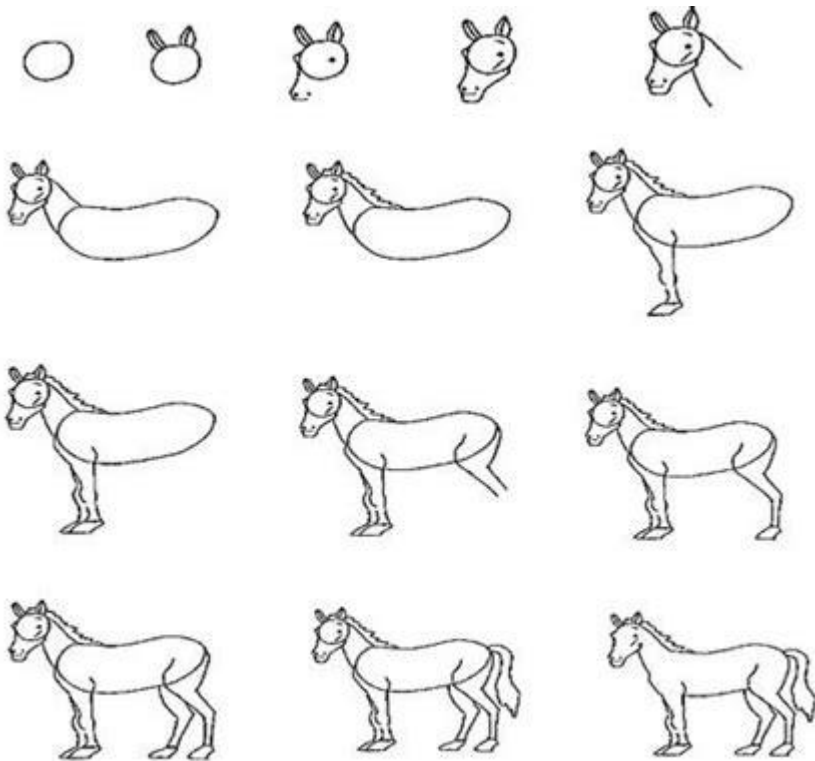
How to convince others that your system is good?

- Rigorousness of the development process
- Rigorousness of the techniques
- Demonstration of effort
- “All or nothing”

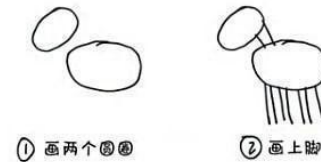


Iterative Software Development

- Develop “validatable” artifacts early

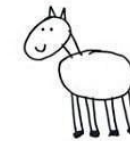


怎样画马

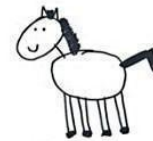


① 画两个圆圈

② 画上脚



③ 画上脸



④ 画上毛发



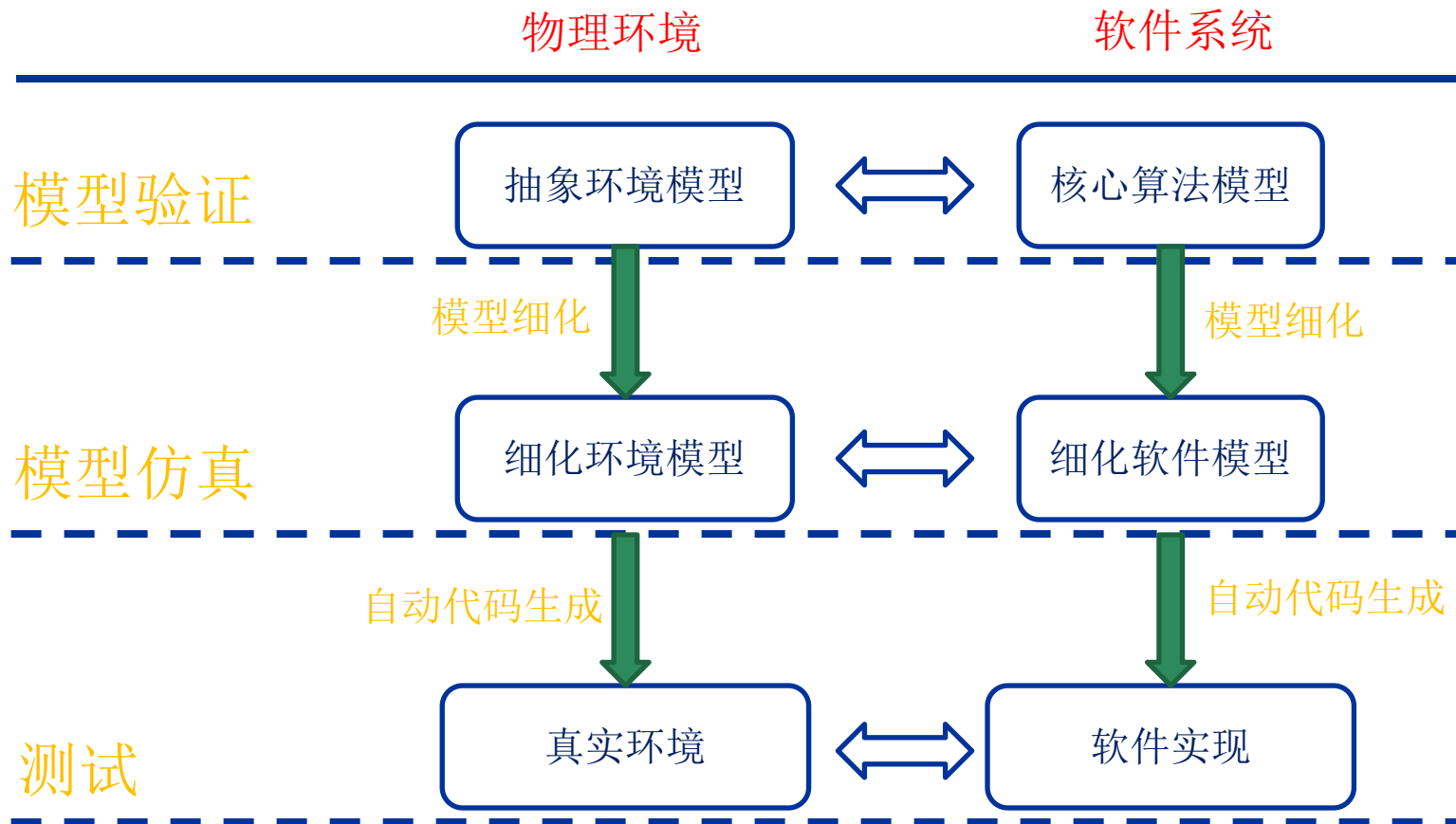
⑤ 再添加其他细节
就大功告成了!

www.bilibili.com

Model-based Software Design

- From verified model to verified code
 - Business model (in UML)
 - Analysis model (in UPPAAL)
 - Design model (in UML/Simulink)
 - Code (in Python)
- Verify analysis/design model
- Maintain traceability during development

Model-based Software Design



**“In a software engineering course, you
PREACH, not TEACH.”**

**-- Frederick P. Brooks, Jr.
UNC Chapel Hill**

Key Challenges in Software Engineering

1. Effective communication

- Between the customers and the engineering team
- Within the engineering team

2. Risk Management

- How to balance conflicting judging criteria?

3. Validation

- How do you convince all stakeholders that the software is effective/safe/secure?

Curriculum

- Software development lifecycle
- Capture software requirements using UML
- Strike a balance: risk management
- Early bug-finding using model checking
- Maintain traceability in model-based software design
- Software testing

Grading

- Homework: 4*5%
- Midterm: 20%
- Final Project: 60%

Project Logistics

- 3 students per team (1,2,3)
- 3 mini projects (a,b,c)
- Each project has 3 stages
 - Requirement (R)
 - Development (D)
 - Validation (V)
- Student 1: $a.R + b.D + c.V$
- Student 2: $b.R + c.D + a.V$
- Student 3: $c.R + a.D + b.V$

Checkpoints and Progression

- Team meeting every week
 - Report on what has been done and plan for next week
 - Part of the demonstration of “effort”
- 3 Customer Consultations
 - Chance to demonstrate initial results and ask for feedback
 - Please take them seriously!

Project Grading

- Overall Product (70%+10%)
 - Functional requirements (40%)
 - Non-functional requirements (15%)
 - Validation (15%)
 - Extra Credits (10%)
- Documentation (30%)
 - Requirement (10%)
 - Development (10%)
 - Validation (10%)

Grading Example

Project 1			Project 2			Project 3		
Overall Product		55	Overall Product		60	Overall Product		65
Documentation	Requirement	8	Documentation	Requirement	5	Documentation	Requirement	6
	Development	7		Development	4		Development	3
	Validation	5		Validation	9		Validation	7

	Score
Student 1 (1R2D3V)	$(55+60+65)/3+8+4+7=79\%$
Student 2 (2R3D1V)	$(55+60+65)/3+5+3+5=73\%$
Student 3 (3R1D2V)	$(55+60+65)/3+6+7+9=82\%$

Working as a team

- Team up with someone you trust
- You are responsible for some task doesn't mean you “only” need to complete the task
- “Make other's job easier”
- “The Black Sheep” will get severe penalties

You may complain about...

- “The requirements are too vague!”
 - It’s not a bug, it’s a feature!
- “I did my part, why do I get penalized for what others didn’t do?”
 - Because you are on the same team.
- Please do not ask
 - “Can we get/lose points if we implement/not implement a feature?”

Logistics

- Slides are released 1 day before each lecture on blackboard
- Important announcements are sent via emails

Academic Ethics

- Homework should be done alone
- Do not share code/documents among teams
- Feel free to reuse code segments **within the team**
- Do not use code/documents from other sources (previous years' or online)
- **Violators will receive severe penalties**

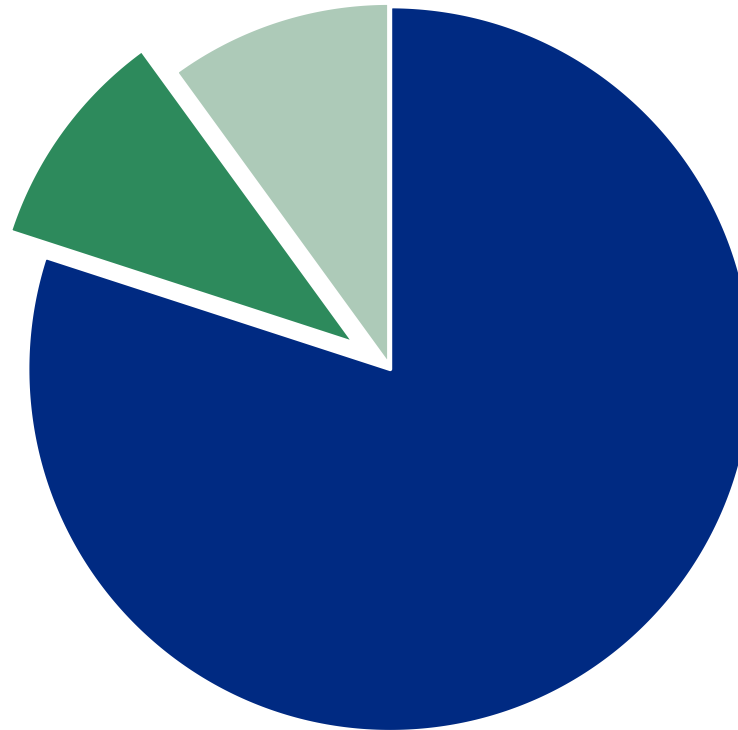
What you will learn during this course

Knowledge

- State of the art software development and validation
- Experience the whole software life cycle
- Limitations of different techniques

Skills

- Modeling methodologies
- Validation techniques



■ Knowledge ■ Skills ■ Mentality ■

Mentality

- Work as a group
- Inter-disciplinary communication
- System Engineering