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

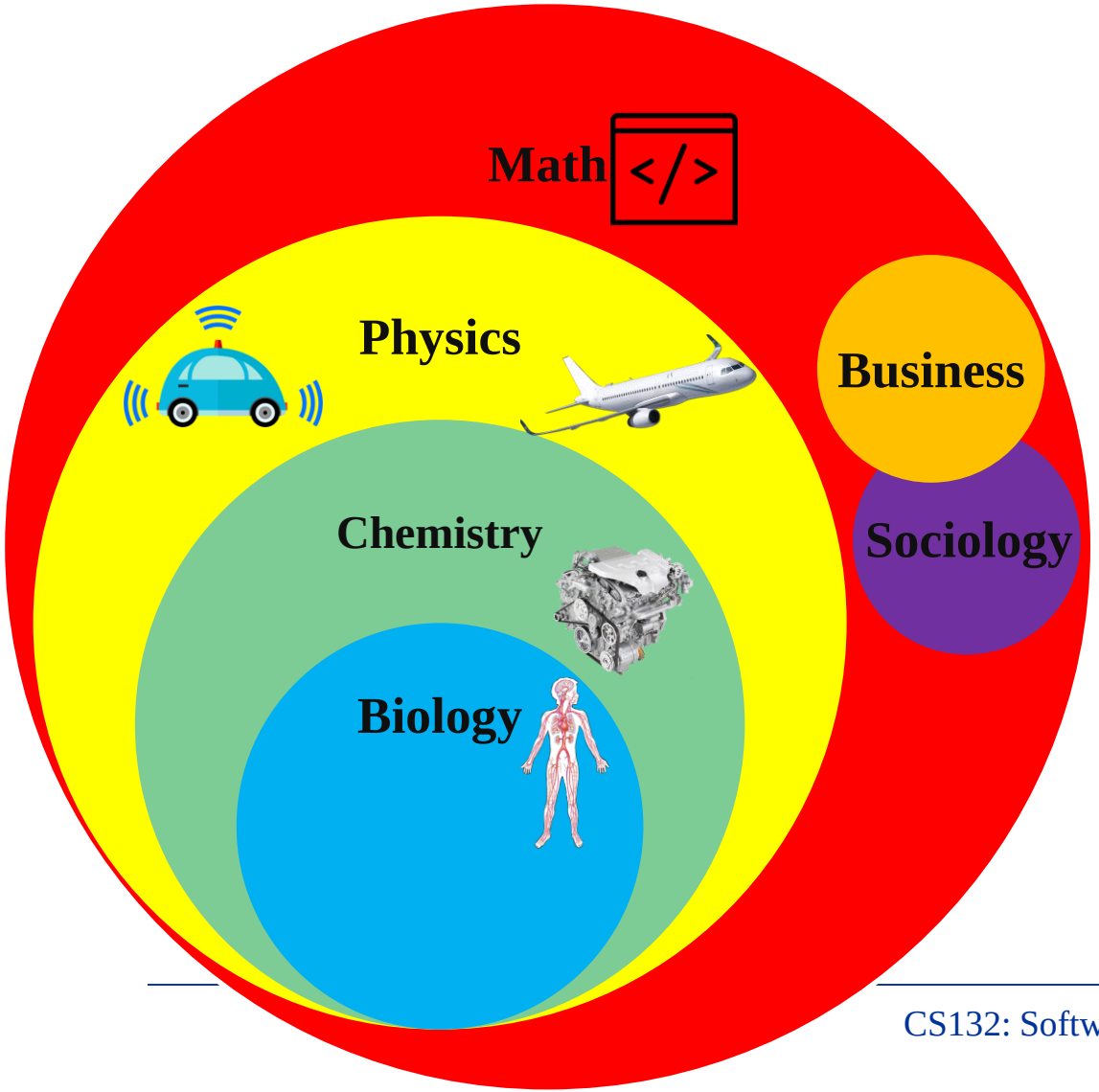
Software Engineering: Definition

- "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

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

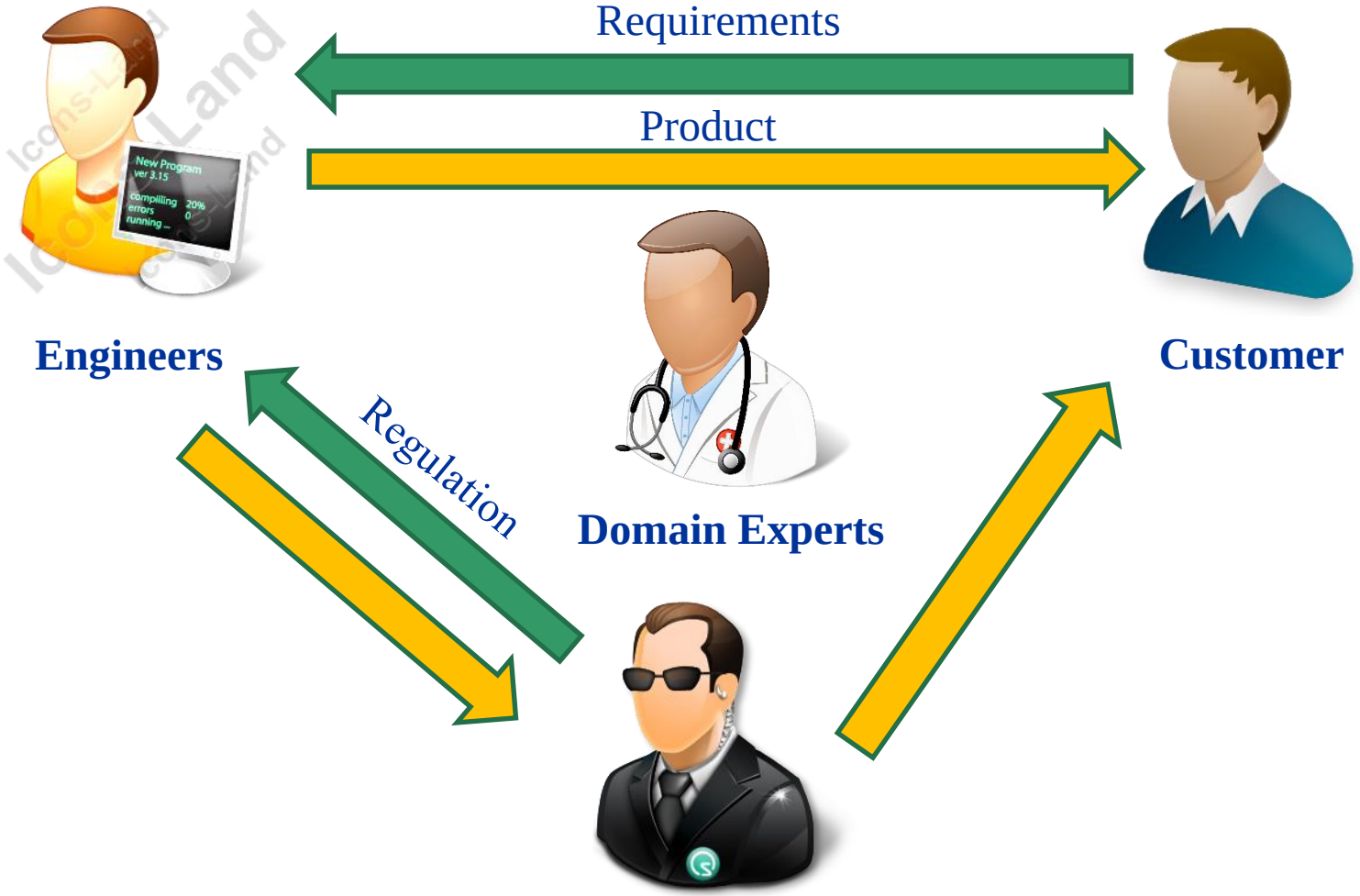
2. Risk Management
   - How to balance conflicting judging criteria?

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

1. Effective communication

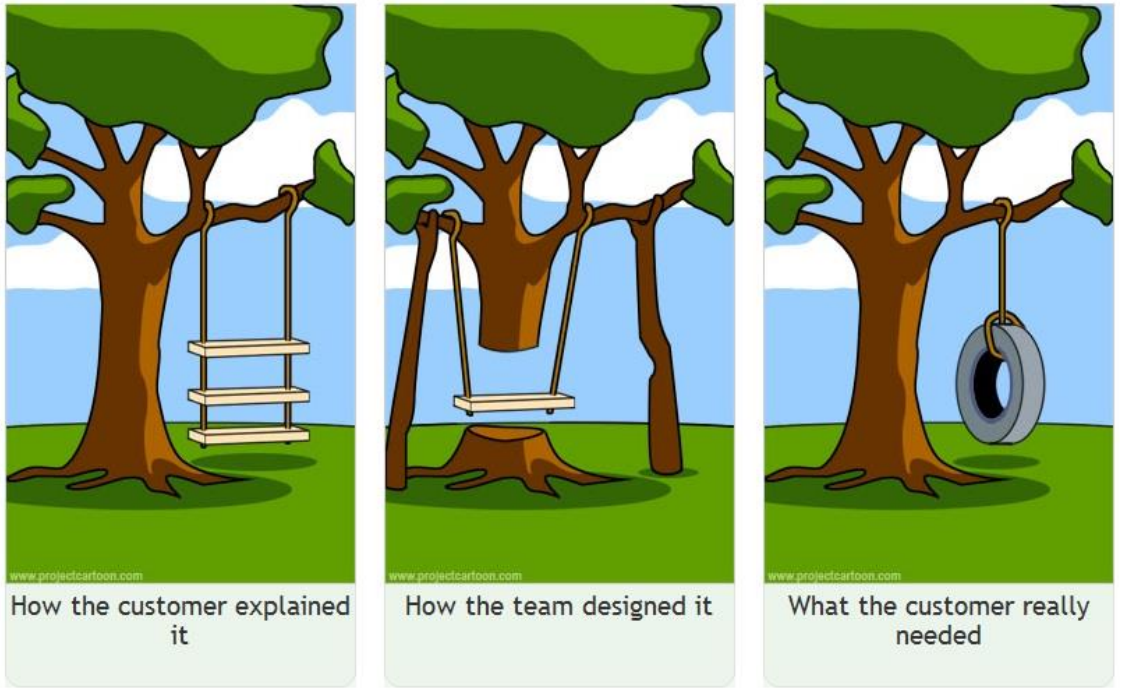
1. Effective Communication

Stakeholders for software



Miscommunications

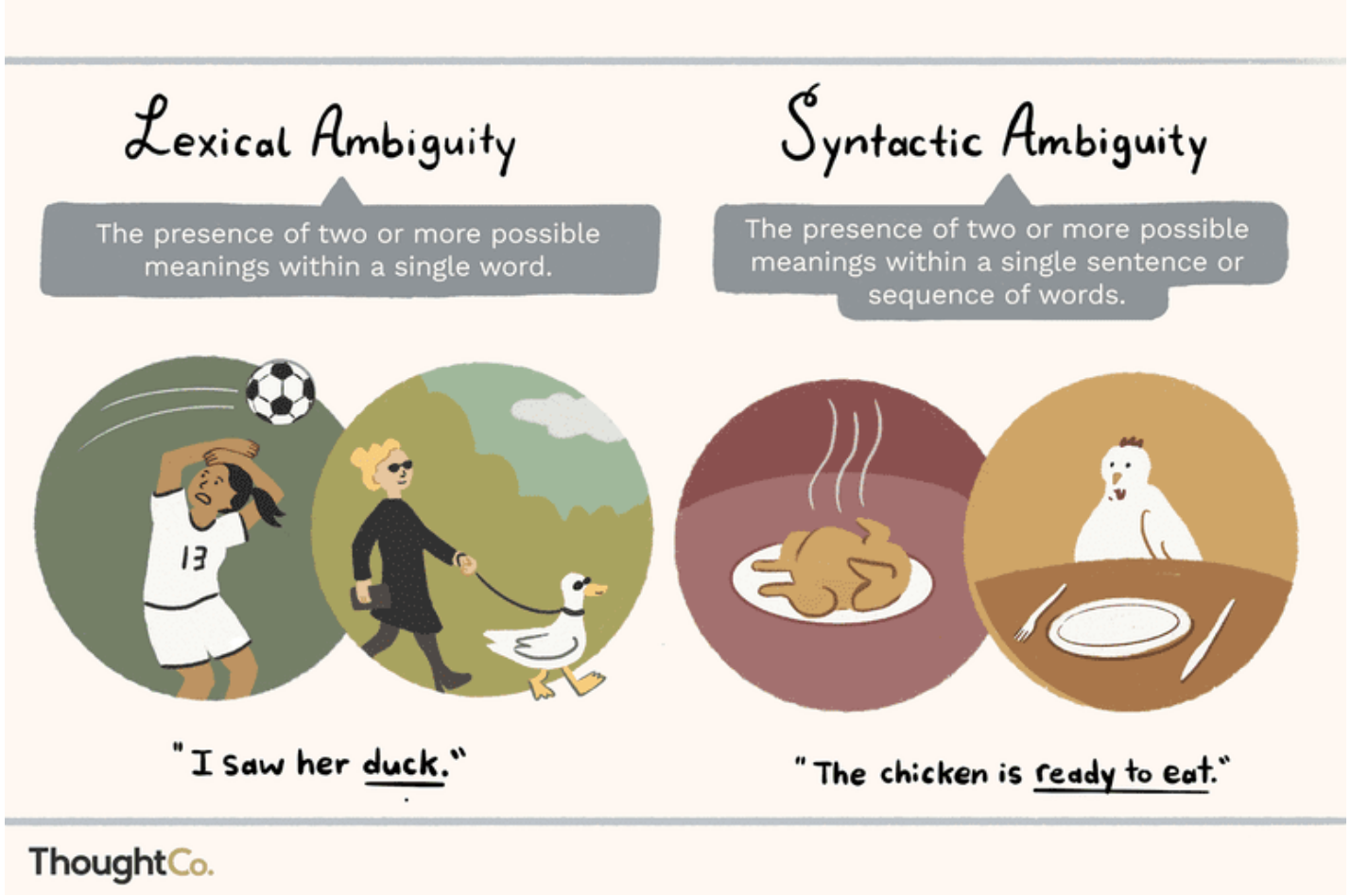
- The customer fails to explain their needs well.
- The customer may not know what he/she wants
- The analyst may not understand the customer's need.
- Your analyst may not convey the requirements to the development team



1. Effective Communication

Domain Knowledge: “The Expert”

Natural Languages Are Prone to Ambiguities



1. Effective Communication

Communications among various stakeholders

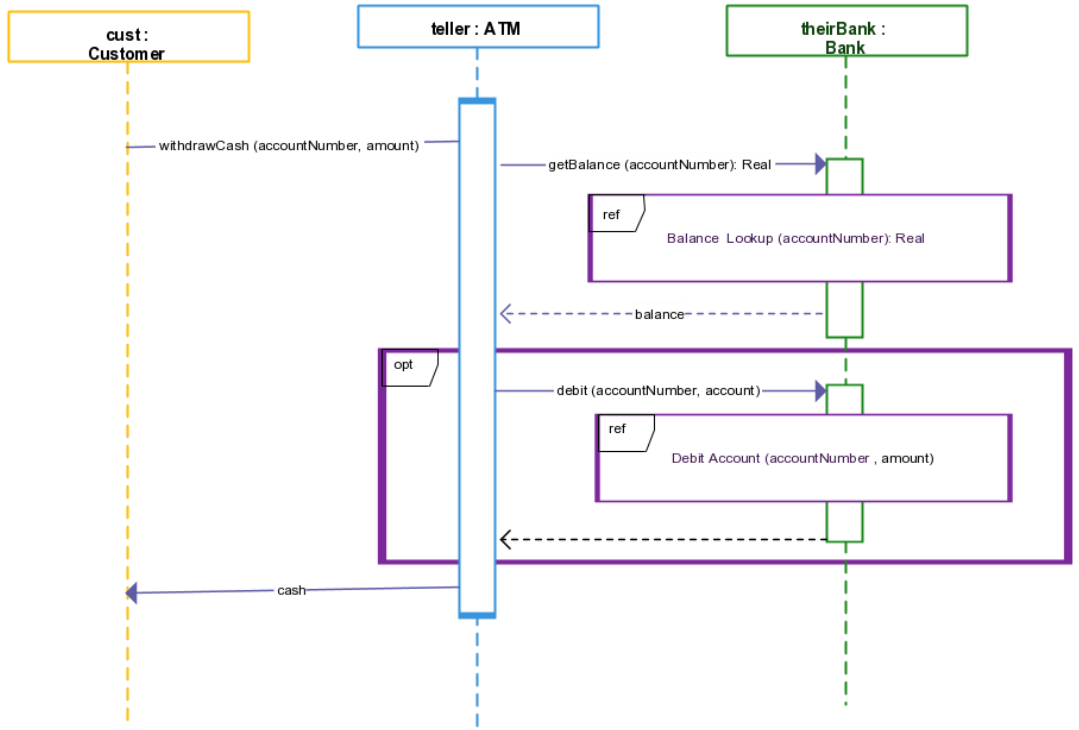
- Need a common language for communication
- Unified Modeling Language (UML)
- Recognized as an international standard
- It's just a tool, not a solution

Examples of UML Diagrams

- Structural diagram

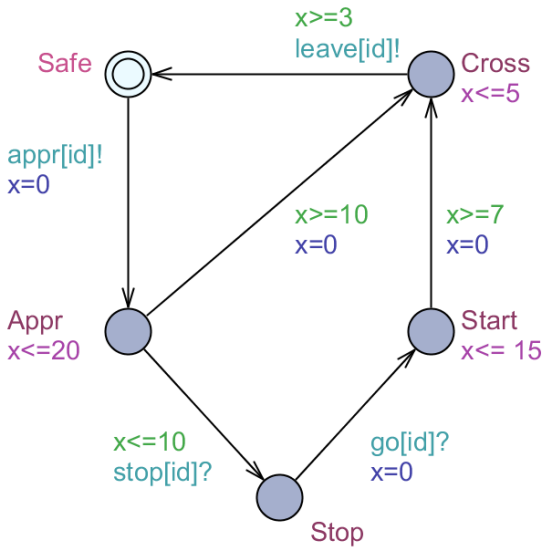
BankAccount
owner : String balance : Dollars = 0
deposit (amount : Dollars) withdrawal (amount : Dollars)

- Behavioral Diagram

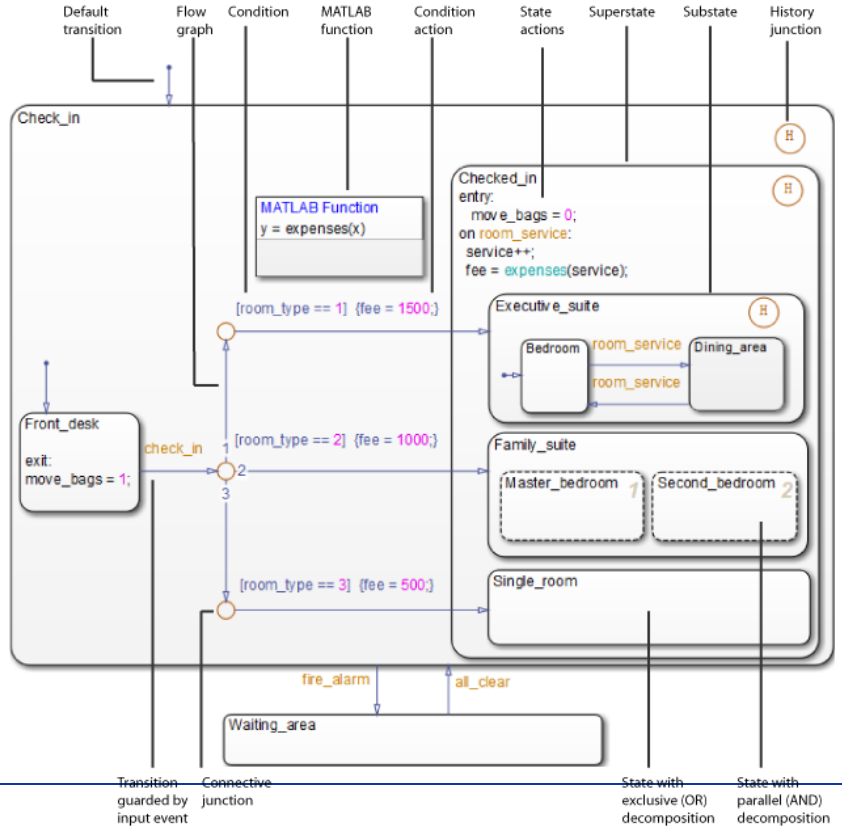


Communication within the team

- 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

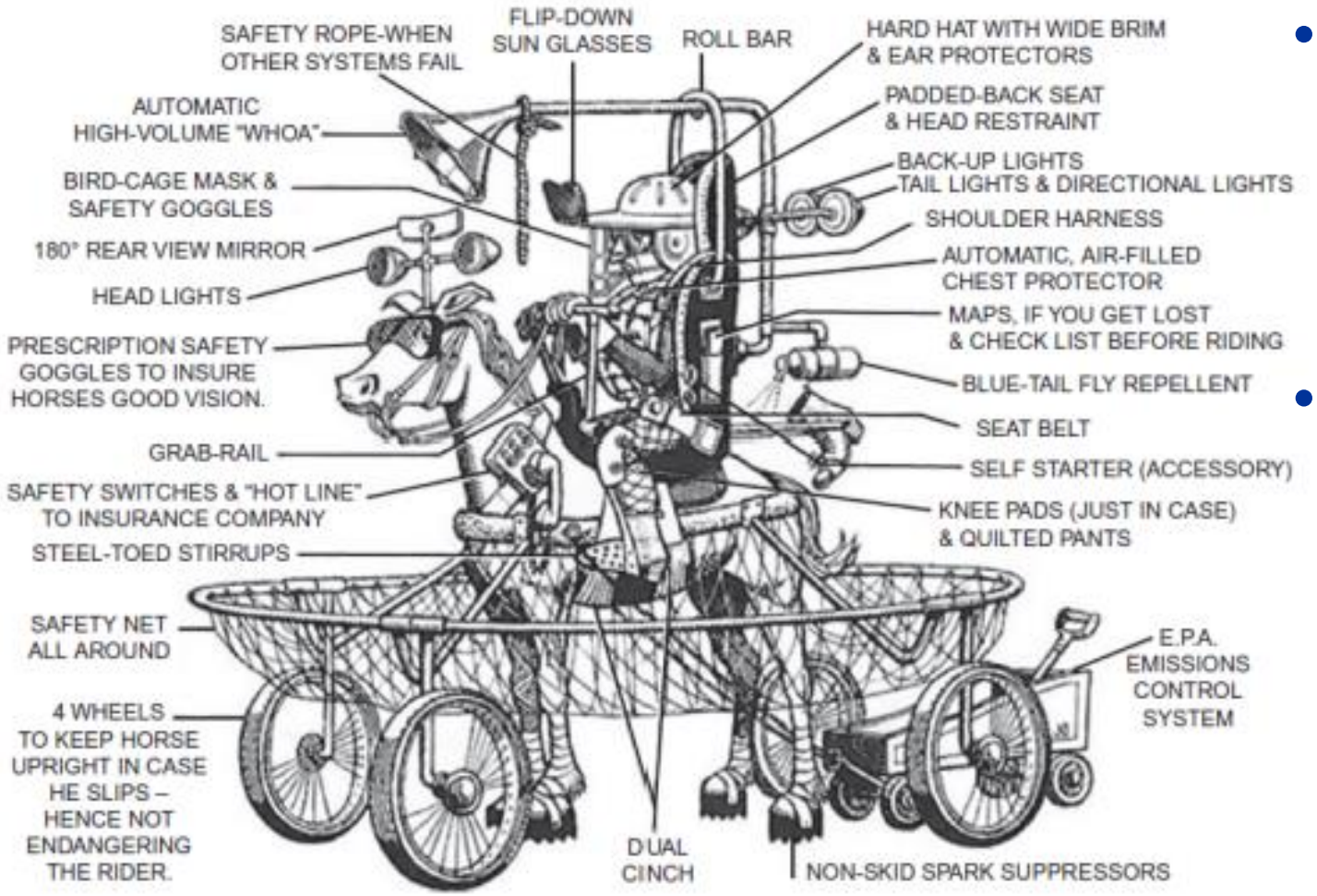
What we want you to learn in this course

- 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

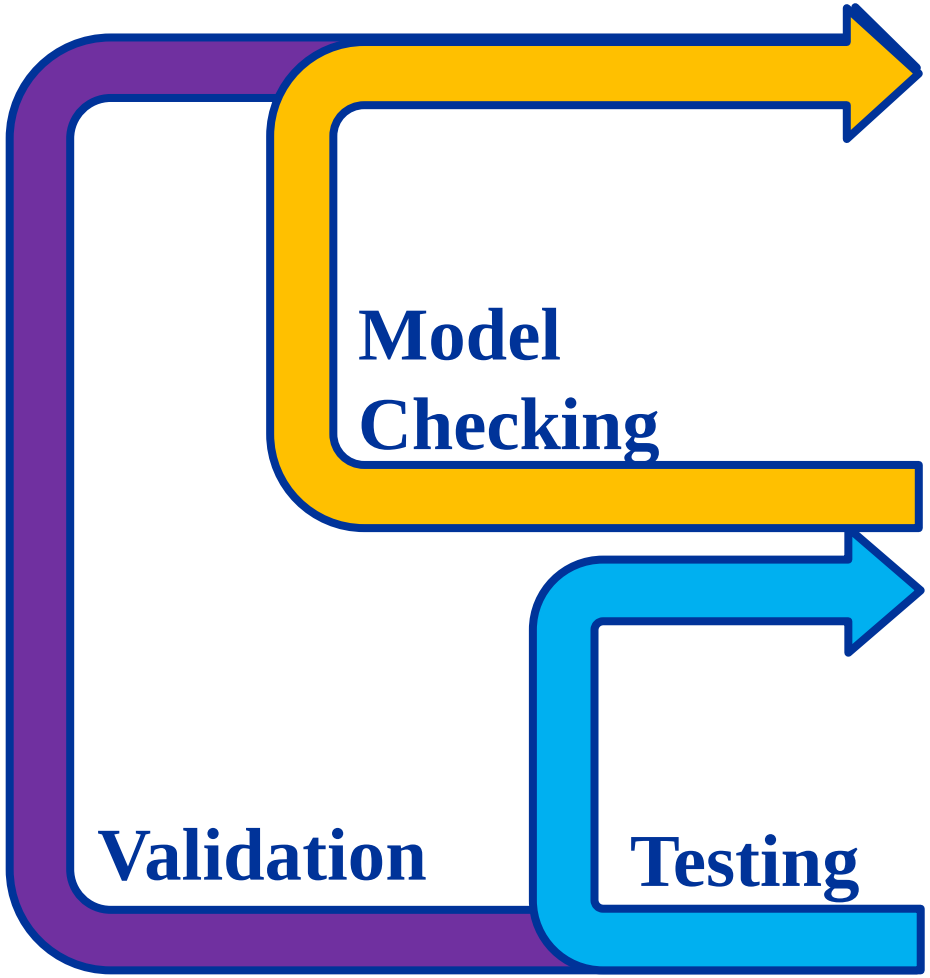
2. Risk Management

Balancing among risks

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

3. Validation

Software Development Process



- Requirements
  - i.e. the vehicle should not crash with any other objects
- Specifications
  - i.e. when the vehicle detects an object on its path, apply brake
- Implementation

3. Validation

What should we validate?

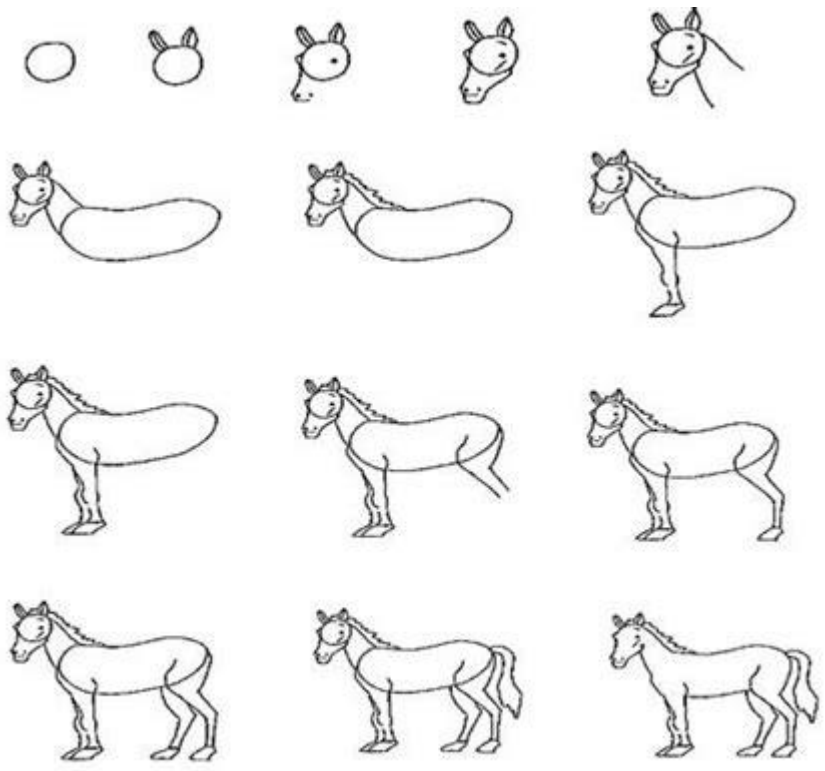
- Efficacy: The system can do its job as designed
- Safety: Under intended use, the system will not harm the user and its surroundings
- Security: Prevent malicious use of the system

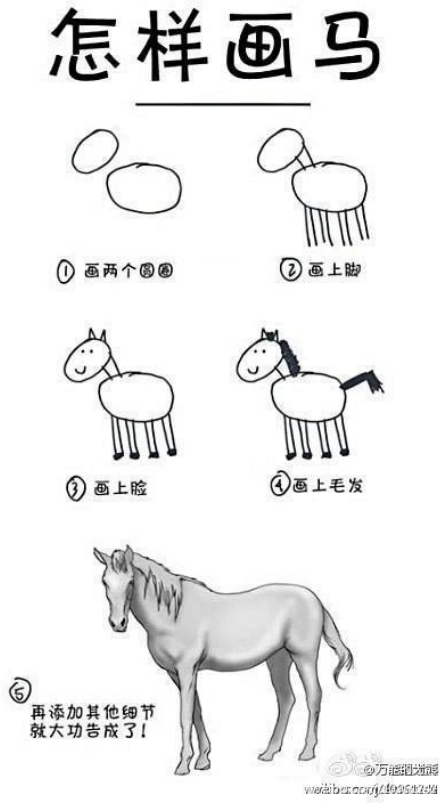
How to convince others that your system is good?

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

Iterative Software Development

- Develop “validatable” artifacts early

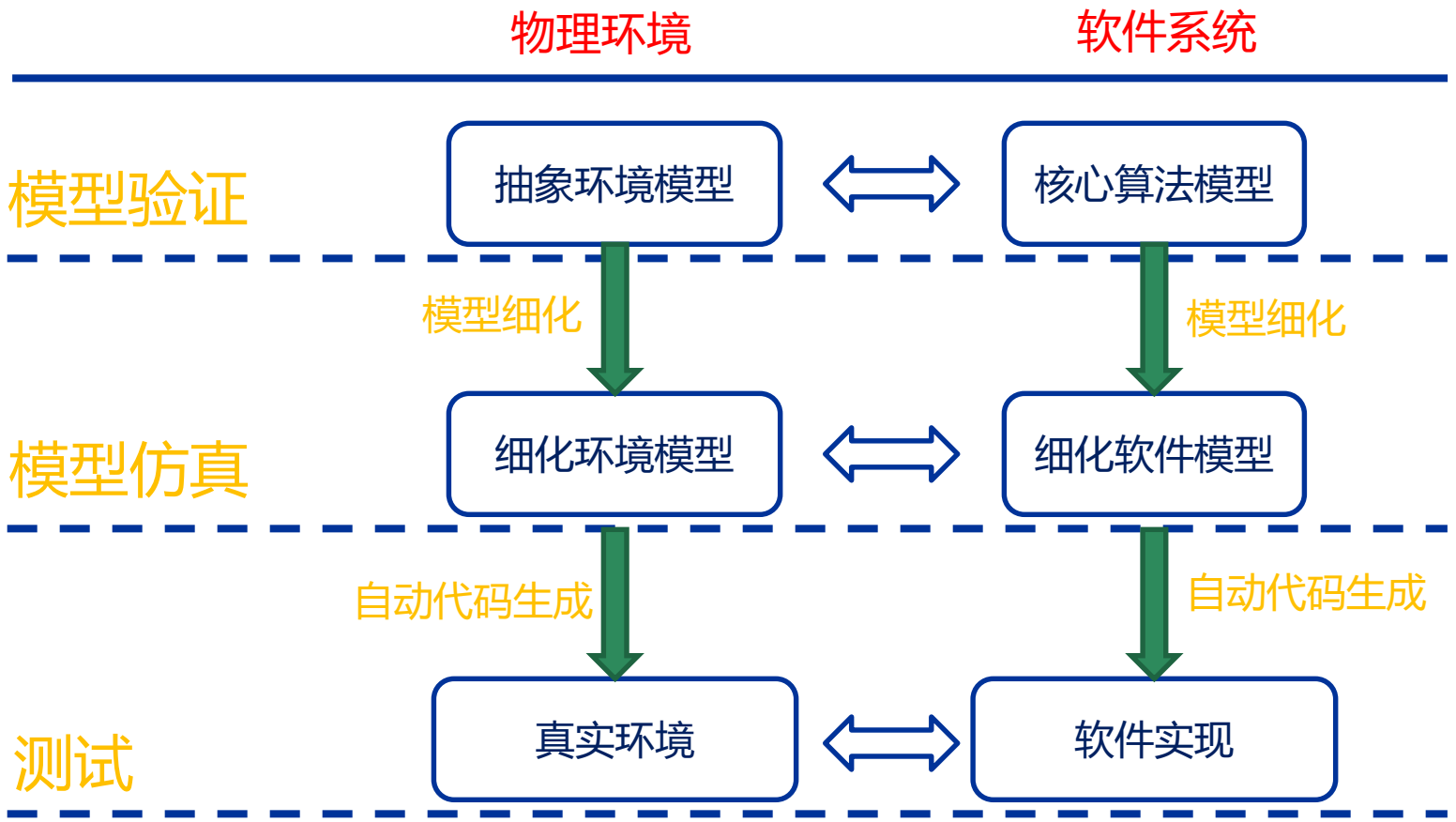




Model-based Software Design

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

Model-based Software Design



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!

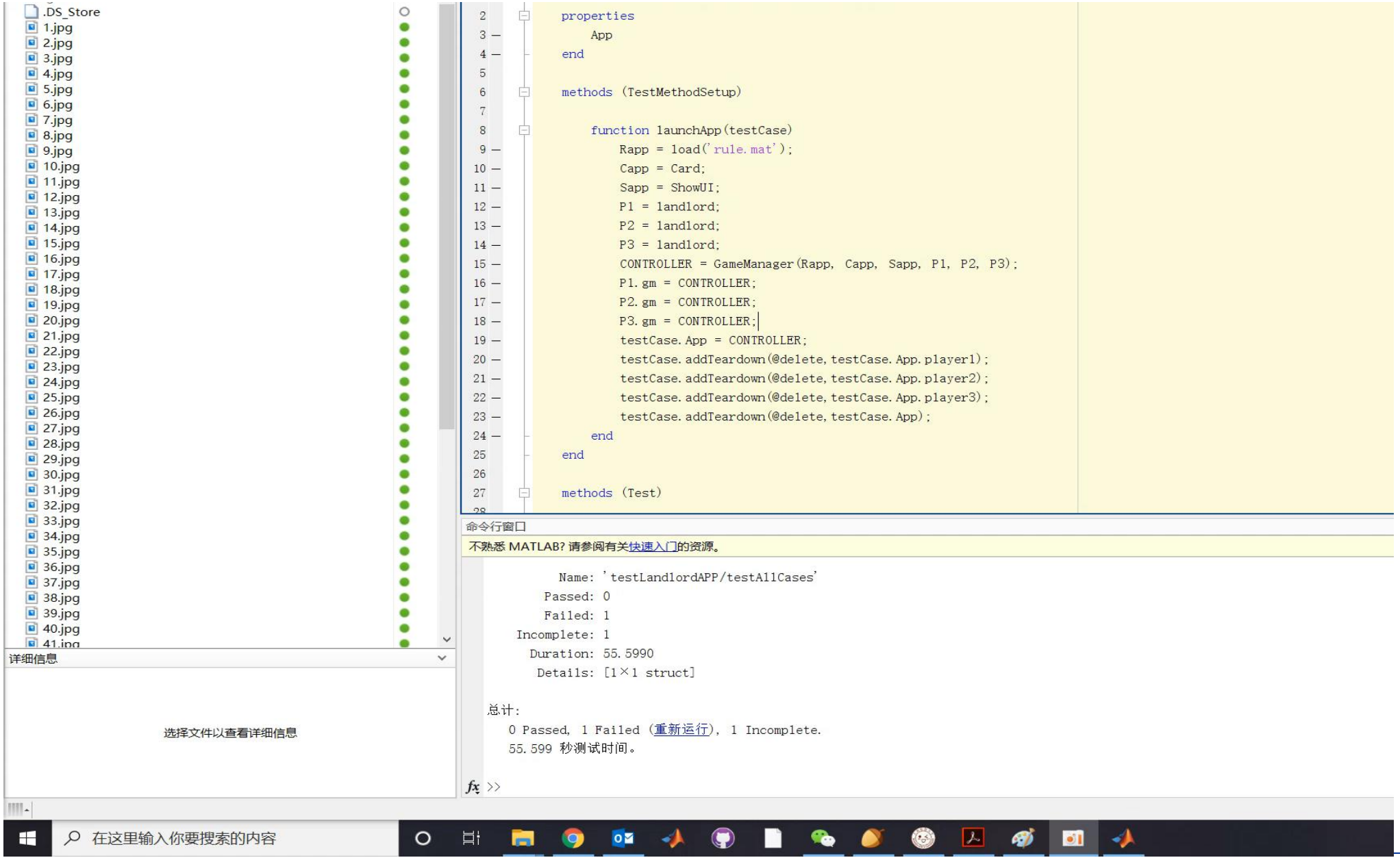
Project Grading

- 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%


.DS_Store
1.jpg
2.jpg
3.jpg
4.jpg
5.jpg
6.jpg
7.jpg
8.jpg
9.jpg
10.jpg
11.jpg
12.jpg
13.jpg
14.jpg
15.jpg
16.jpg
17.jpg
18.jpg
19.jpg
20.jpg
21.jpg
22.jpg
23.jpg
24.jpg
25.jpg
26.jpg
27.jpg
28.jpg
29.jpg
30.jpg
31.jpg
32.jpg
33.jpg
34.jpg
35.jpg
36.jpg
37.jpg
38.jpg
39.jpg
40.jpg
41.jpg
详细信息
选择文件以查看详细信息
properties
App
end
methods (TestMethodSetup)
function launchApp(testCase)
Rapp = load('rule.mat');
Capp = Card;
Sapp = ShowUI;
P1 = landlord;
P2 = landlord;
P3 = landlord;
CONTROLLER = GameManager(Rapp, Capp, Sapp, P1, P2, P3);
P1.gm = CONTROLLER;
P2.gm = CONTROLLER;
P3.gm = CONTROLLER;
testCase.App = CONTROLLER;
testCase.addTeardown(@delete,testCase.App.player1);
testCase.addTeardown(@delete,testCase.App.player2);
testCase.addTeardown(@delete,testCase.App.player3);
testCase.addTeardown(@delete,testCase.App);
end
end
methods (Test)
命令行窗口
不熟悉 MATLAB? 请参阅有关快速入门的资源。
Name: 'testLandlordAPP/testAllCases'
Passed: 0
Failed: 1
Incomplete: 1
Duration: 55.5990
Details: [1×1 struct]
总计:
0 Passed, 1 Failed (重新运行), 1 Incomplete.
55.599 秒测试时间。
>>
在这里输入你要搜索的内容

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, that's why.
- 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