Software Engineering

@ U of Trento

Fabio Casati
University of Trento
(next semester)

Yannis Velegrakis
University of Trento

Alessandro Tomasi
University of Trento
(TA)

Course Schedule

- 2 hours lectures weekly
- 3 hours laboratory weekly

- Keep an eye on the schedule

- Instructor's email: velgias@disi.unitn.eu
- TA: Alessandro.Tomasi@unitn.it

- Project Course (No written exam)
  - Midterm Assignment
  - Final before Christmas

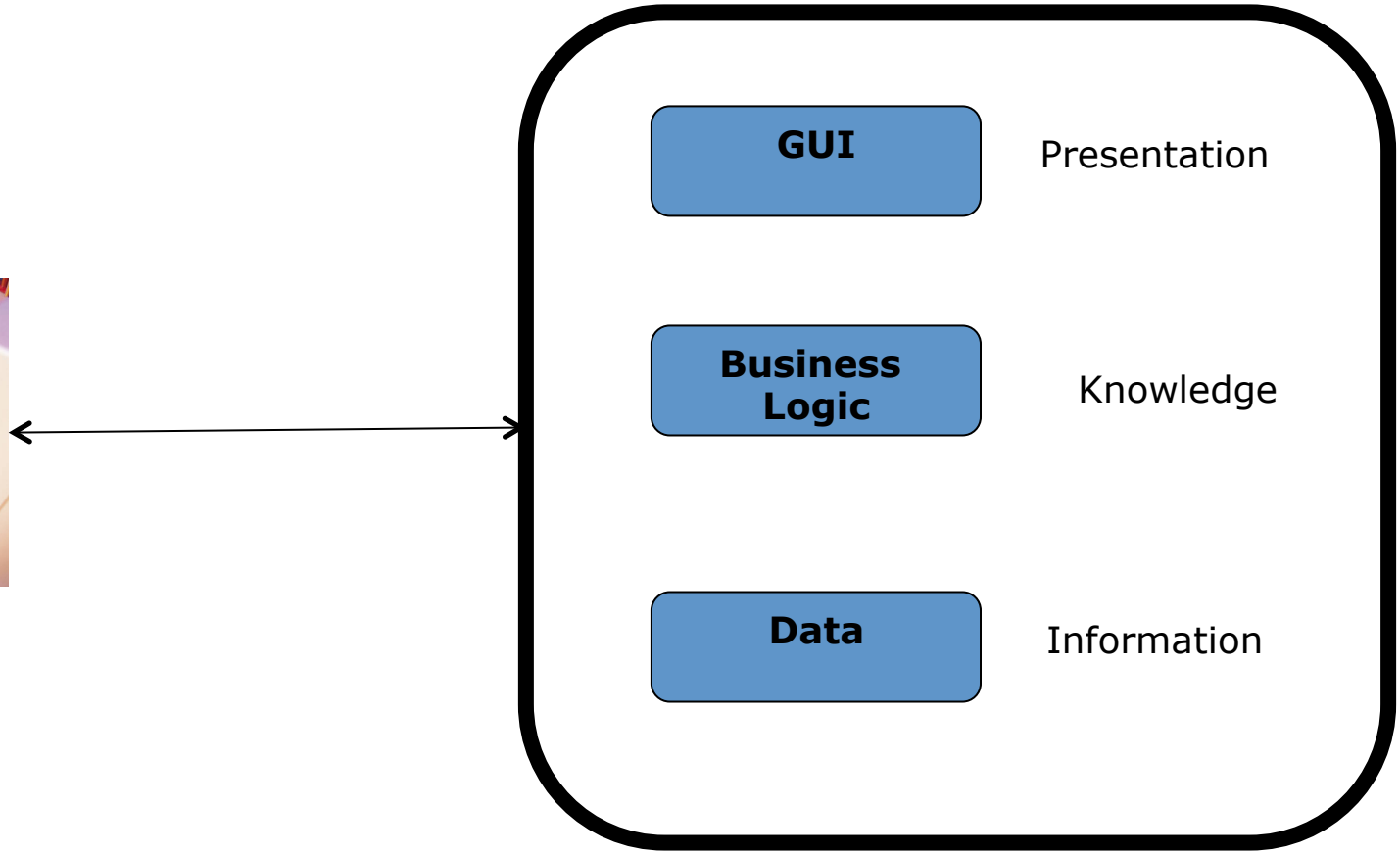
Software Engineering

- Software Engineering != Programming
  - Programming is actually a pretty small part.
- Determining what to build
  - Requirements (what tasks should the Software accomplish?)
  - Specifications (exact operating behavior)
- Determining how to build it (Design)
- Testing (functionality and experiential)
- Debugging (functional and performance)
- Maintaining the program (new APIs, platforms, minor features)
- More and more, these are not linear steps.
- Hopefully, software lifetime is mostly maintenance/ enhancement.

Focus

- We will spend most of our time on
  - Designing the right application
  - Writing correct, maintainable, and extensible code.
  - Building the right thing.
  - Integrating and connecting subsystems.

The notion of Software Engineering



Lectures

- Much of the course is learn-by-doing
  - Learn-by-doing works better if you "do" correctly.
- Lectures teach the concepts
  - Will make the programming/design easier
- Lectures provide the context, examples
  - Historic and future: the why
  - "War stories" — Avoid previous mistakes.
- Lectures are self-complete
  - But not detailed as text
  - No text book
    - But you can find more if needed on classical books or the Web
- You are expected to know the material
  - For future courses
  - For future employments

Some more Software Engineering Topics

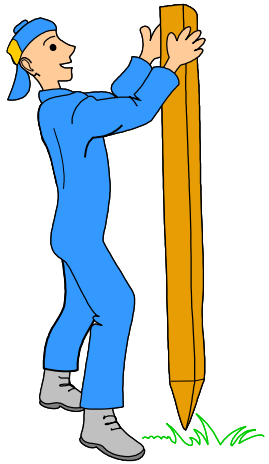
- Code
  - Interfaces
  - Collections
  - Io
  - Threads
  - Thread-design
- Testing
- Design-doc
- Requirements
- Patterns
- Specifications
- Networks
- Persistence
- Databases
- Embedded-lang
- Web

Introduction: Software is Complex

- Complex ≠ complicated
- Complex = composed of many simple parts
  related to one another
- Complicated = not well understood, or explained

Courtesy of Ivan Marsic

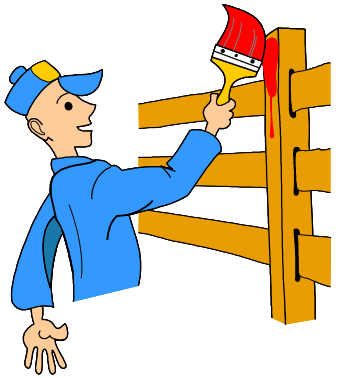
Complexity Example: Scheduling Fence Construction Tasks



Setting posts
[3 time units]

Cutting wood
[2 time units]

Nailing
[2 time units for unpainted; 3 time units otherwise]



Painting
[5 time units for uncut wood; 4 time units otherwise]

Setting posts < Nailing, Painting

Cutting < Nailing

...shortest possible completion time = ?

[⇒ "simple" problem, but hard to solve without a pen and paper]

Courtesy of Ivan Marsic

More Complexity

Suppose today is Tuesday, November 29

What day will be on January 3?

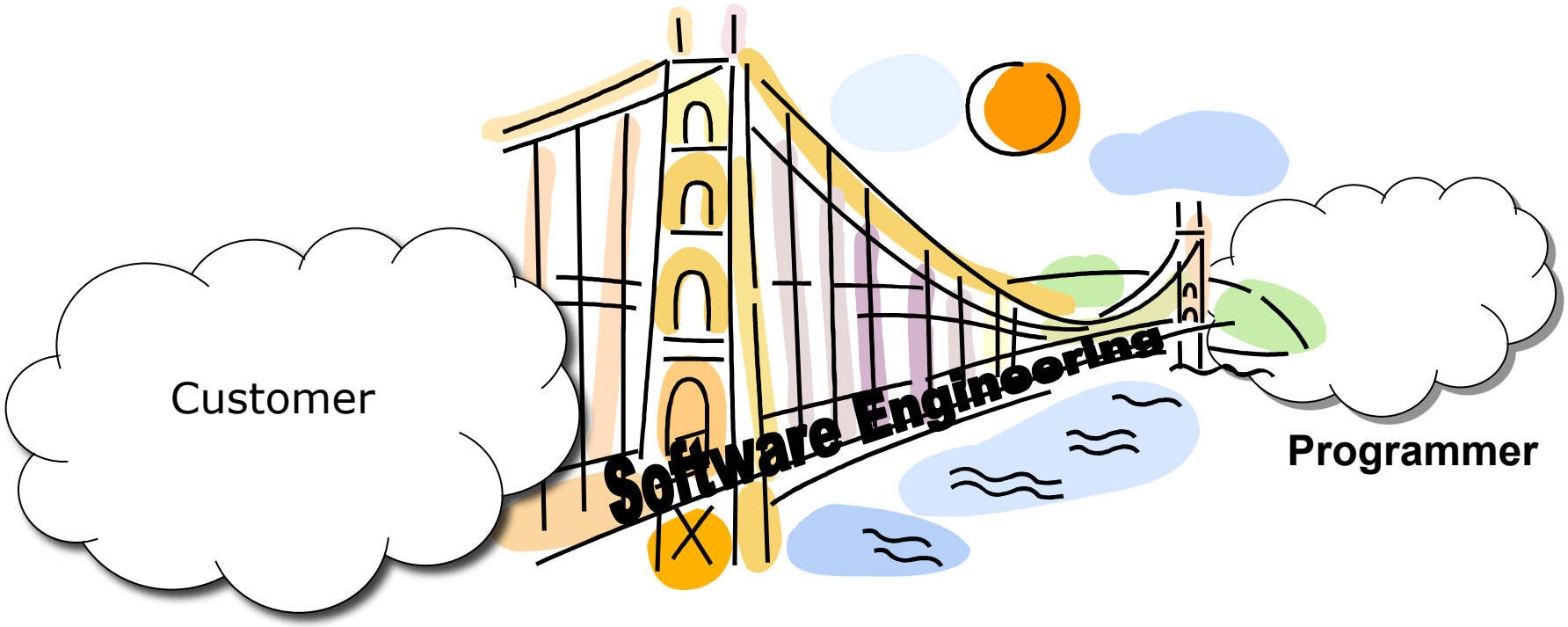
[To answer, we need to bring the day names and the day numbers into coordination, and for that we may need again a pen and paper]

Courtesy of Ivan Marsic

10

The Role of Software Engg. (1)

A bridge from customer needs to programming implementation



First law of software engineering

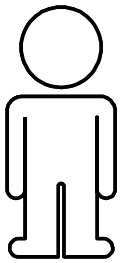
Software engineer is willing to learn the problem domain

(problem cannot be solved without understanding it first)

Courtesy of Ivan Marsic

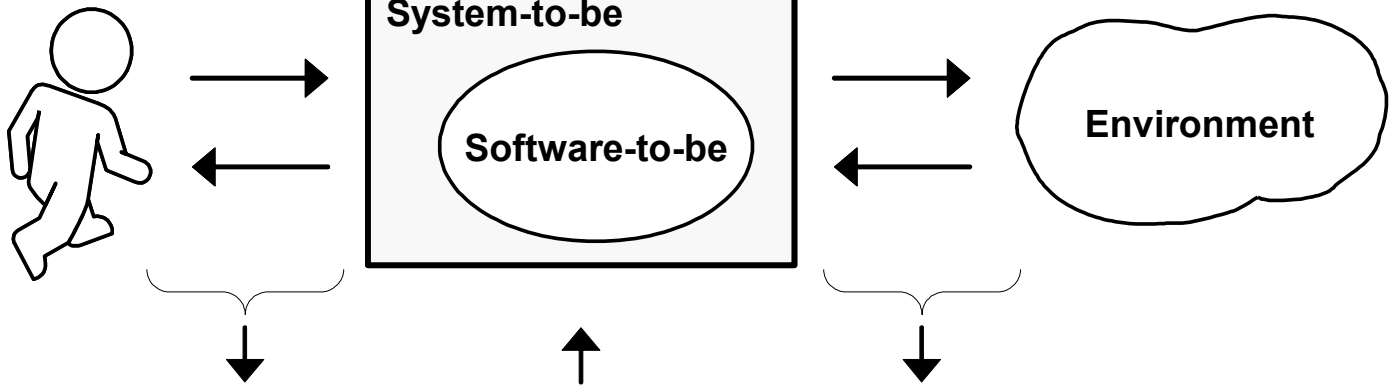
11

The Role of Software Engg. (2)



Customer:

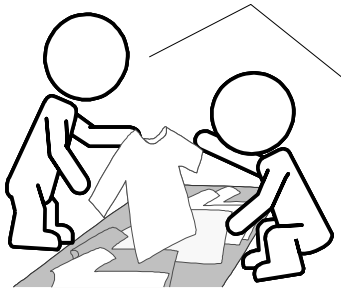
Requires a computer system to *achieve some business goals* by user interaction or interaction with the environment in a specified manner



Software Engineer's task:

To ***understand how*** the system-to-be needs to interact with the user or the environment so that customer's requirement is met and ***design*** the software-to-be

May be the same person



Programmer's task:

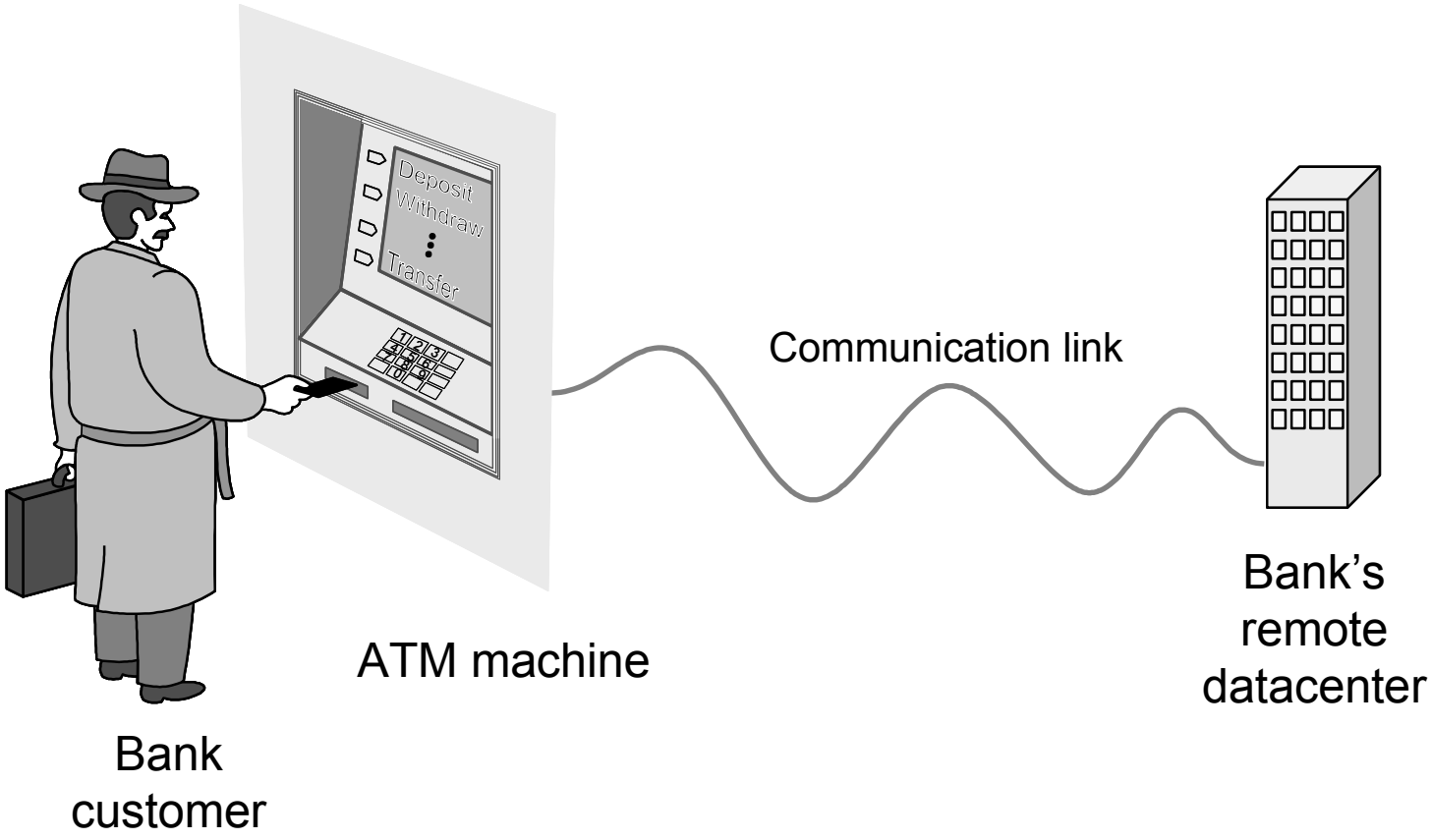
To ***implement*** the software-to-be designed by the software engineer

Courtesy of Ivan Marsic

12

Example: ATM Machine

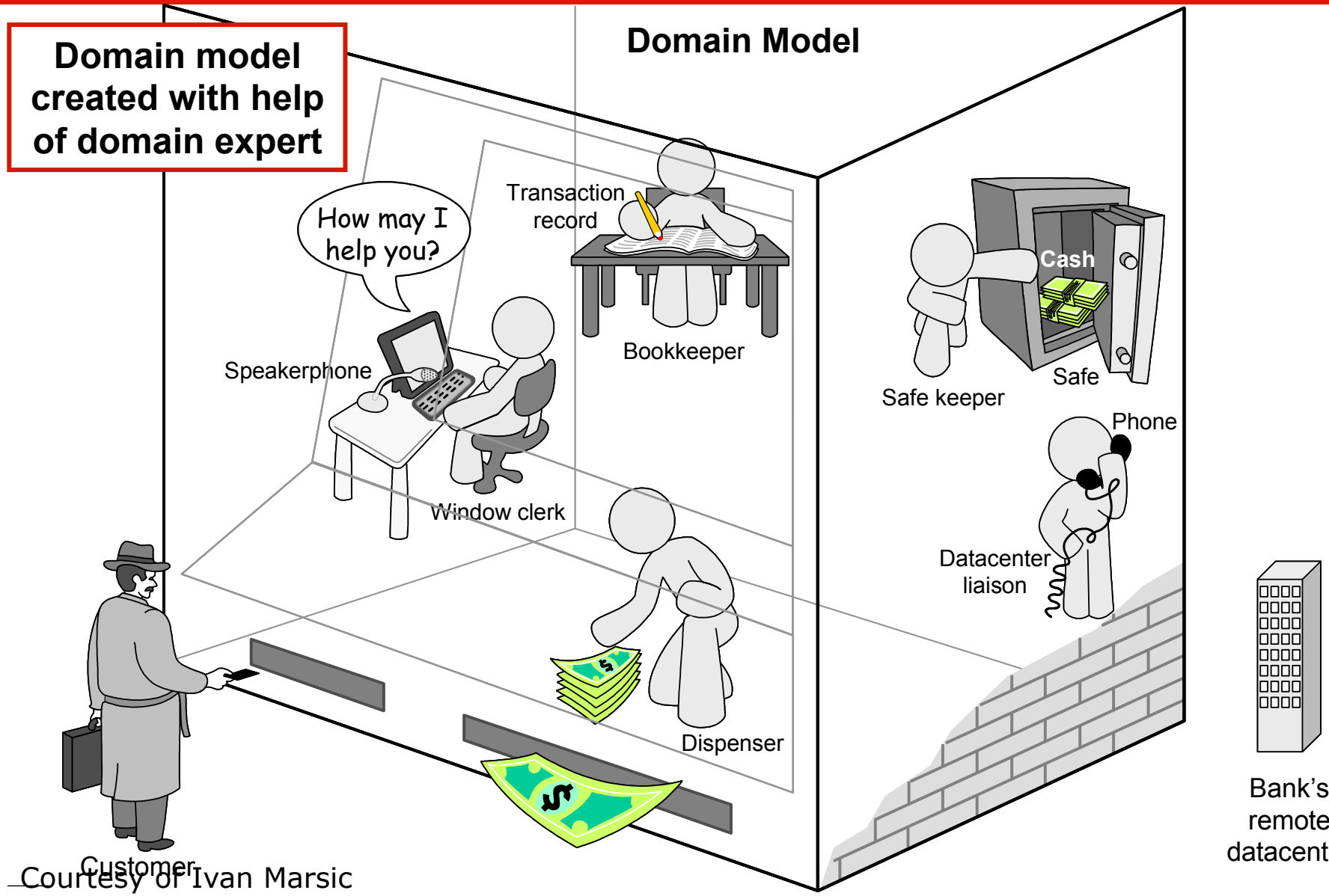
Understanding the money-machine problem:



Courtesy of Ivan Marsic

13

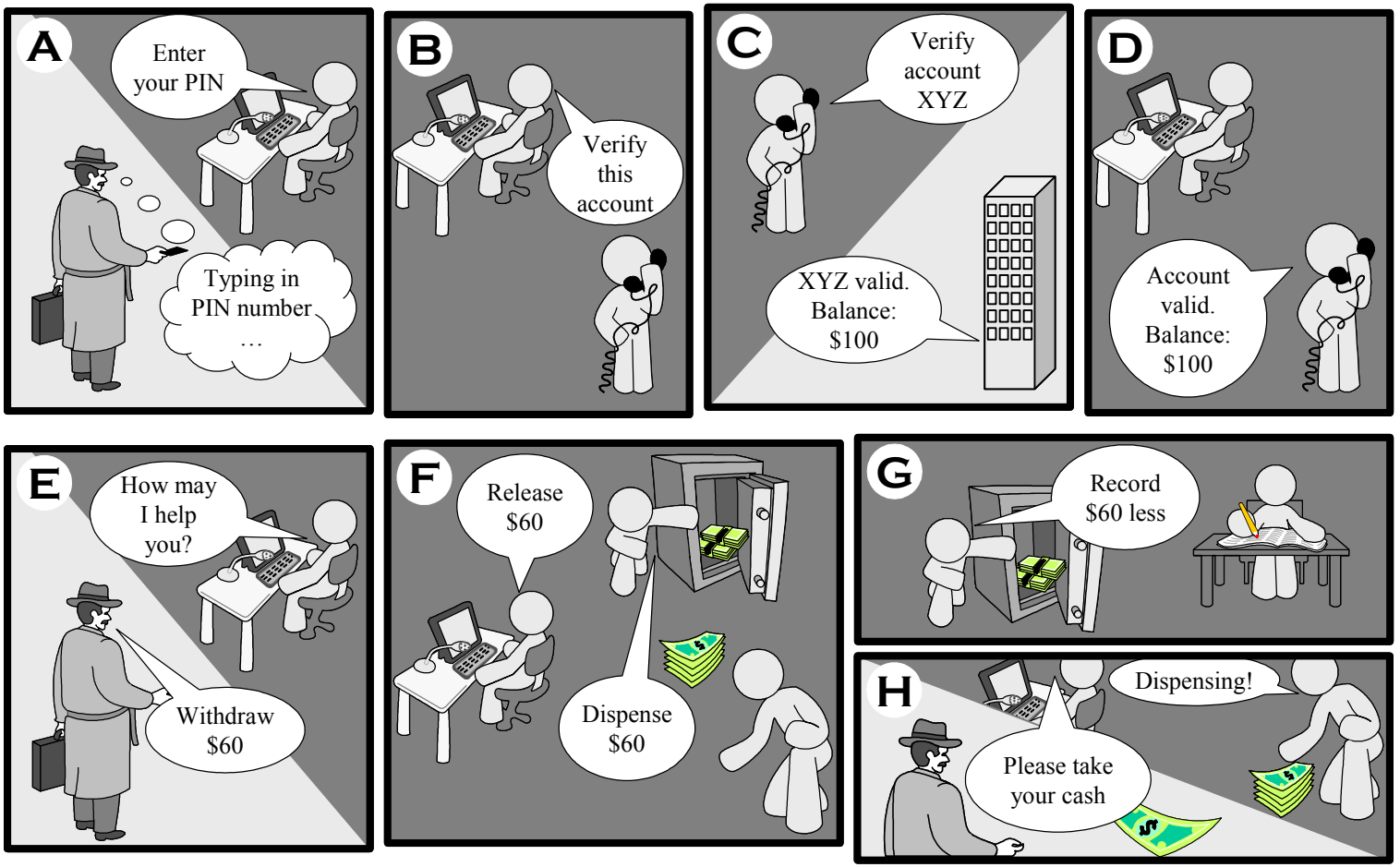
How ATM Machine Might Work



Courtesy of Ivan Marsic

14

Cartoon Strip: How ATM Machine Works



Courtesy of Ivan Marsic

15

Software Engineering Blueprints

- Specifying software problems and solutions is like cartoon strip writing
- Unfortunately, most of us are not artists, so we will use something less exciting:
  UML symbols
- However …

Courtesy of Ivan Marsic

16

Second Law of Software Engineering

- **Software should be written for people first**
  - (Computers run software, but hardware quickly becomes outdated)
  - Useful + good software lives long
  - To nurture software, people must be able to understand it

Courtesy of Ivan Marsic

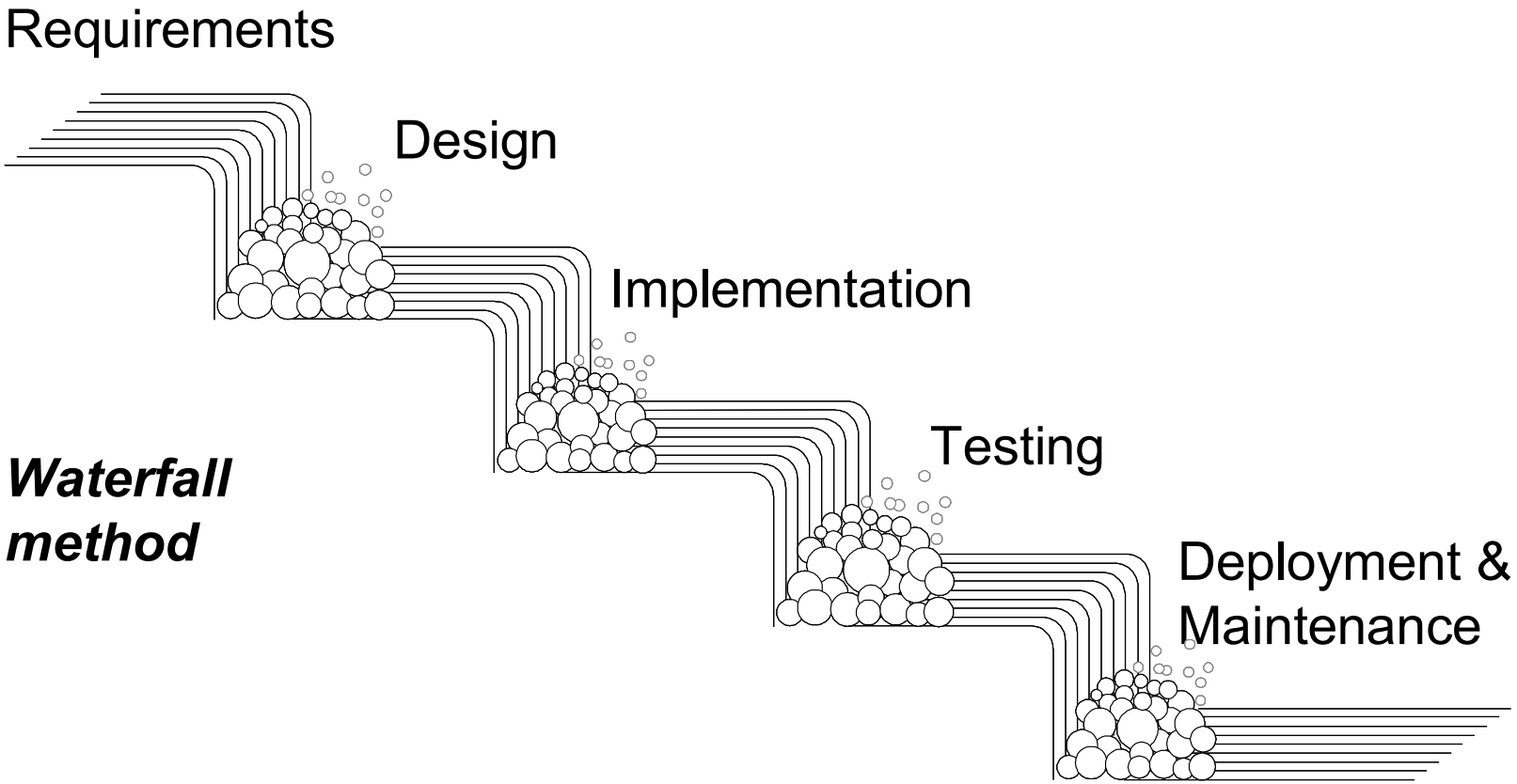
Software Development Methods

- Method = work strategy
  - The Feynman Problem-Solving Algorithm:
    (i) Write down the problem (ii) think very hard, and (iii) write down the answer.
- Waterfall
  - Unidirectional, finish this step before moving to the next
- Iterative + Incremental
  - Develop increment of functionality, repeat in a feedback loop
- Agile
  - User feedback essential; feedback loops on several levels of granularity

Courtesy of Ivan Marsic

18

Waterfall Method



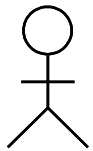
Unidirectional, no way back
finish this step before moving to the next

Courtesy of Ivan Marsic

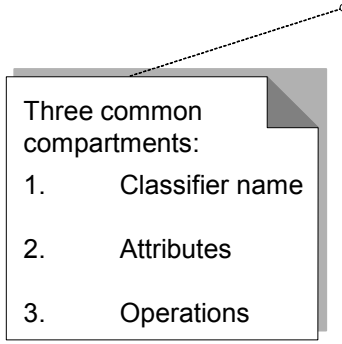
19

UML - Language of Symbols

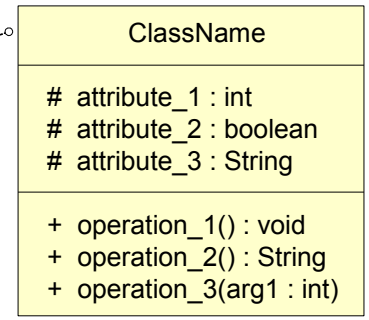
UML = Unified Modeling Language



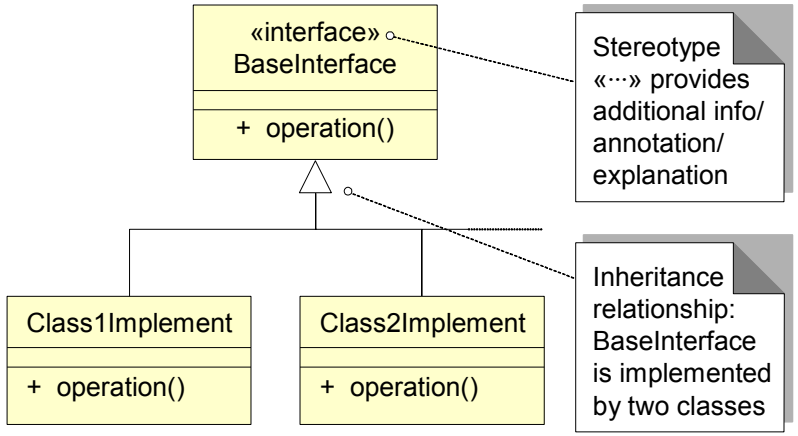
Actor



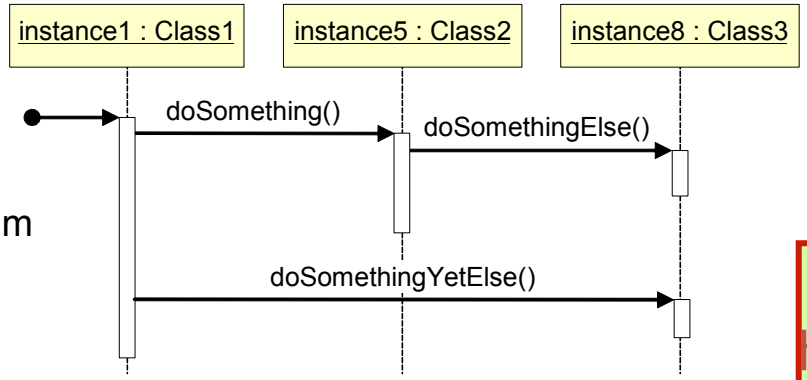
Comment



Software Class



Software Interface Implementation



Interaction Diagram

Online information:
http://www.uml.org

Courtesy of Ivan Marsic

20

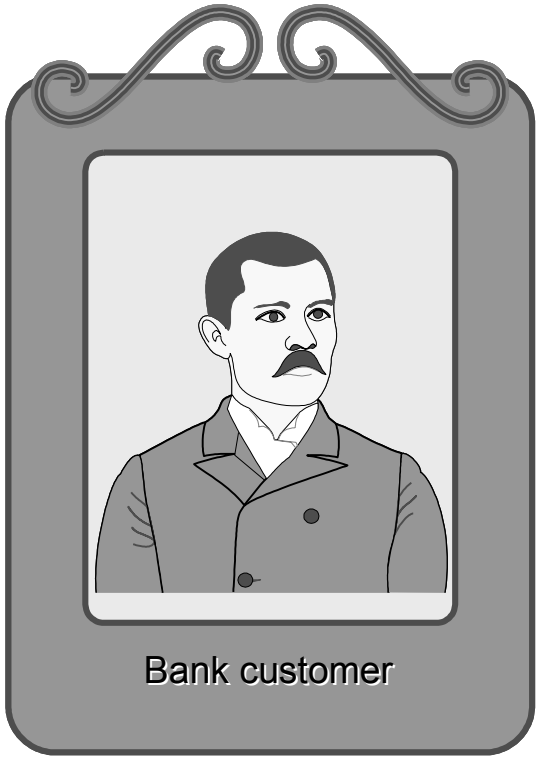
Understanding the Problem Domain

- System to be developed
- Actors
  - Agents external to the system
- Concepts/ Objects
  - Agents working inside the system
- Use Cases
  - Scenarios for using the system

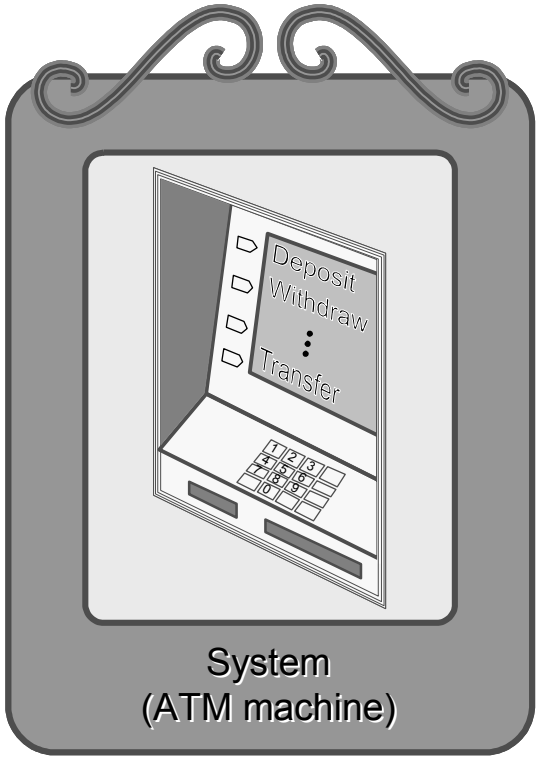
Courtesy of Ivan Marsic

21

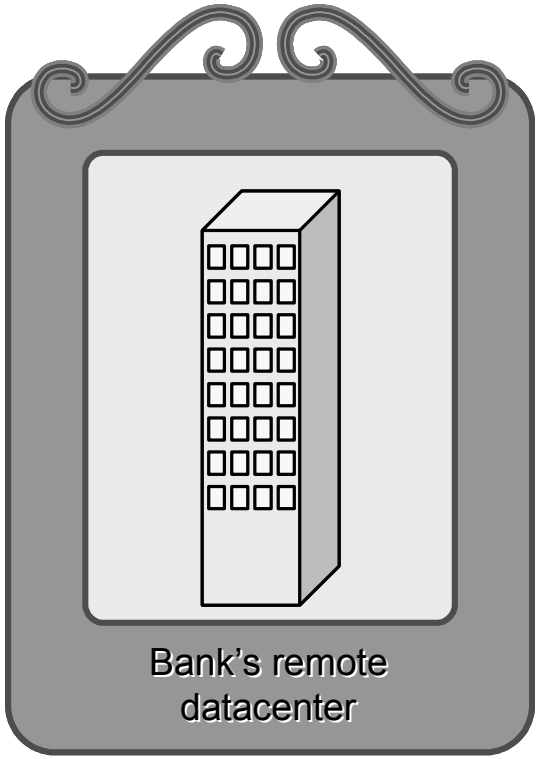
ATM: Gallery of Players



Bank customer



System
(ATM machine)



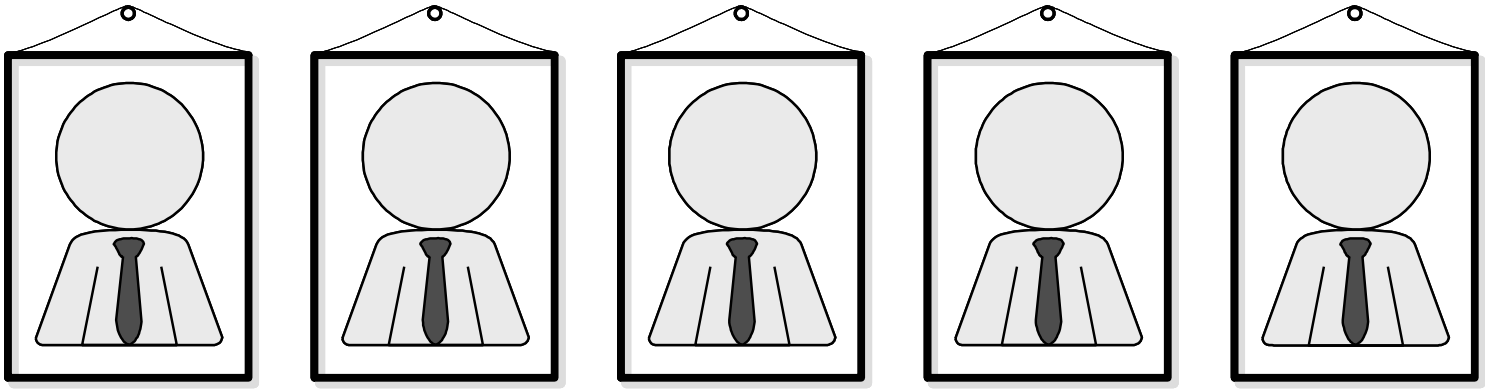
Bank's remote
datacenter

Actors (Easy to identify because they are visible!)

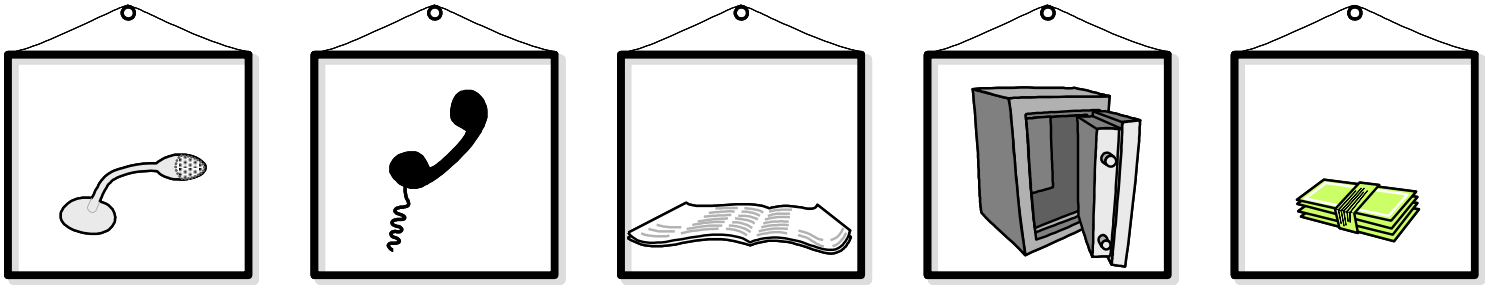
Courtesy of Ivan Marsic

22

Gallery of Workers + Things



Window clerk | Datacenter liaison | Bookkeeper | Safe keeper | Dispenser



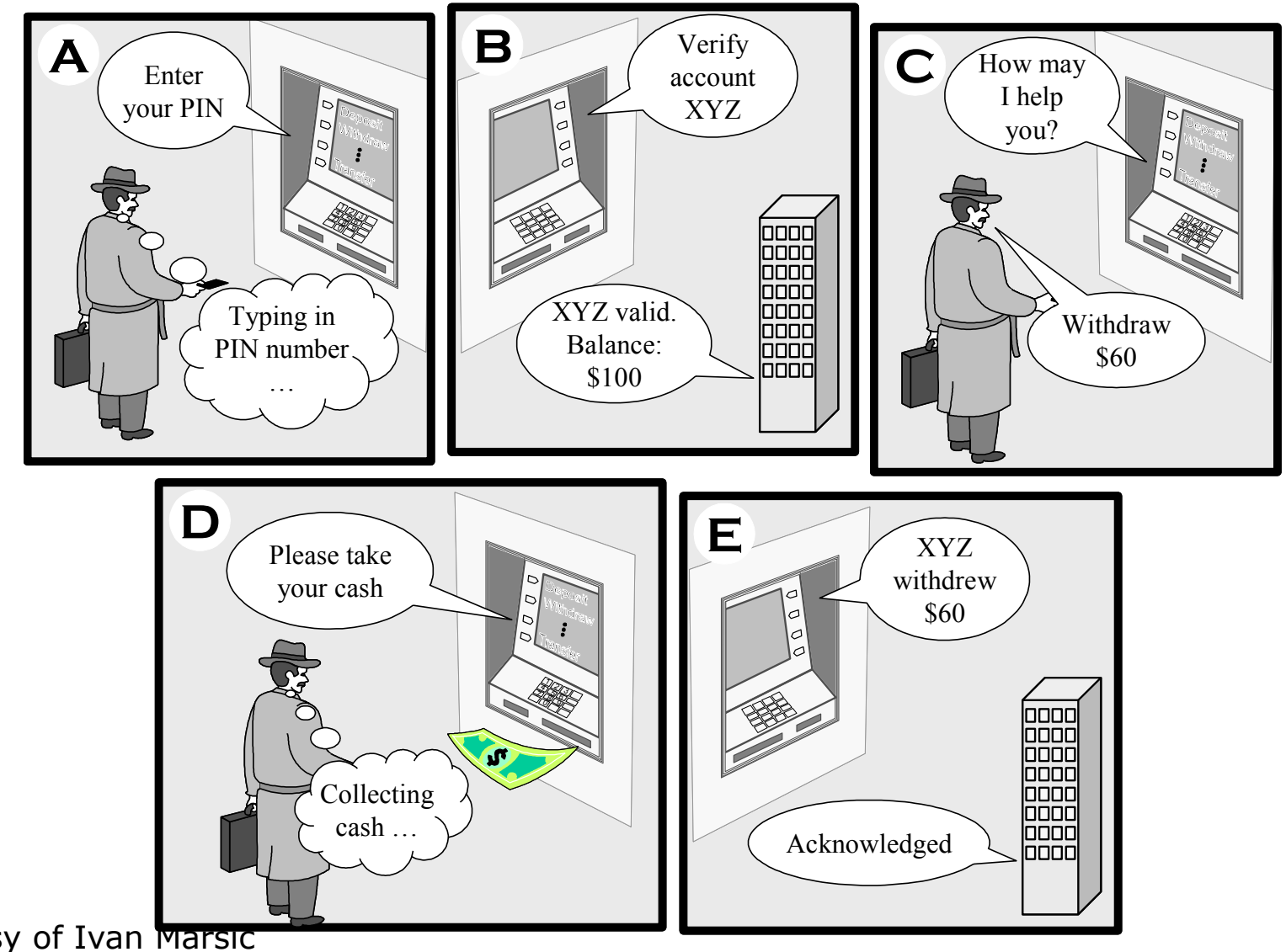
Speakerphone | Telephone | Transaction record | Safe | Cash

Concepts (Hard to identify because they are invisible/imaginary!)

Courtesy of Ivan Marsic

23

Use Case: Withdraw Cash

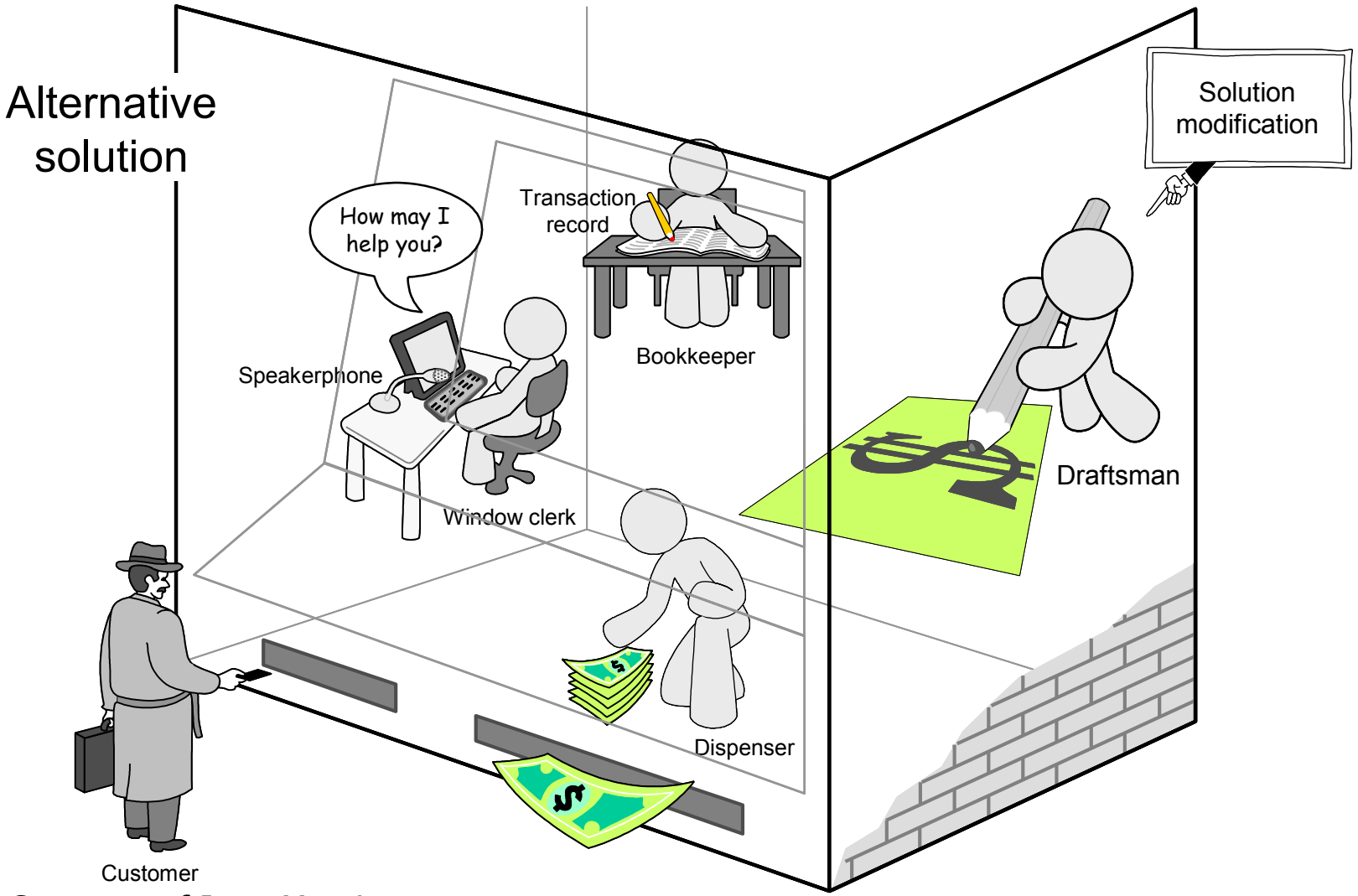


Courtesy of Ivan Marsic

24

How ATM Machine Works (2)

Domain Model (2)

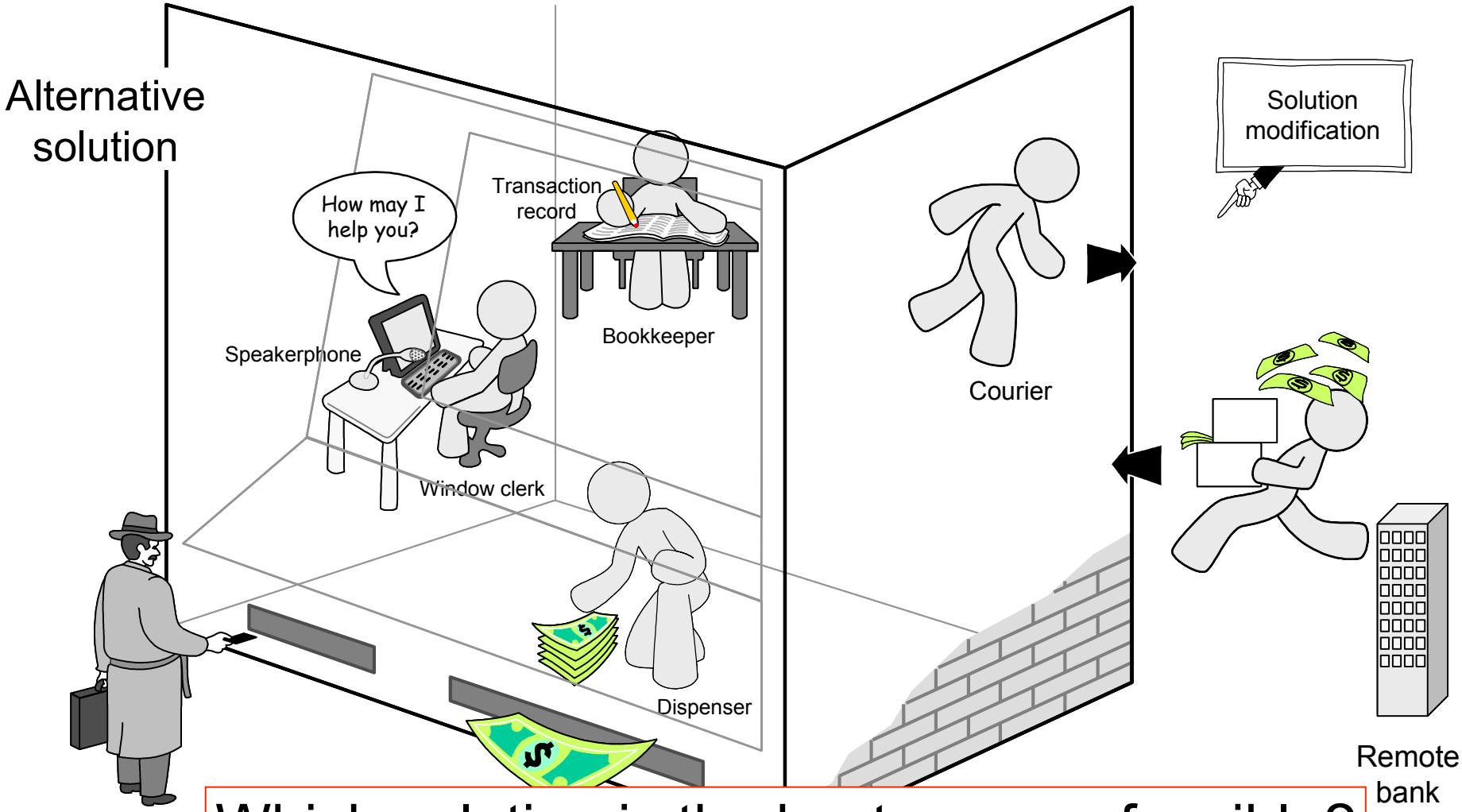


Courtesy of Ivan Marsic

25

How ATM Machine Works (3)

Domain Model (3)



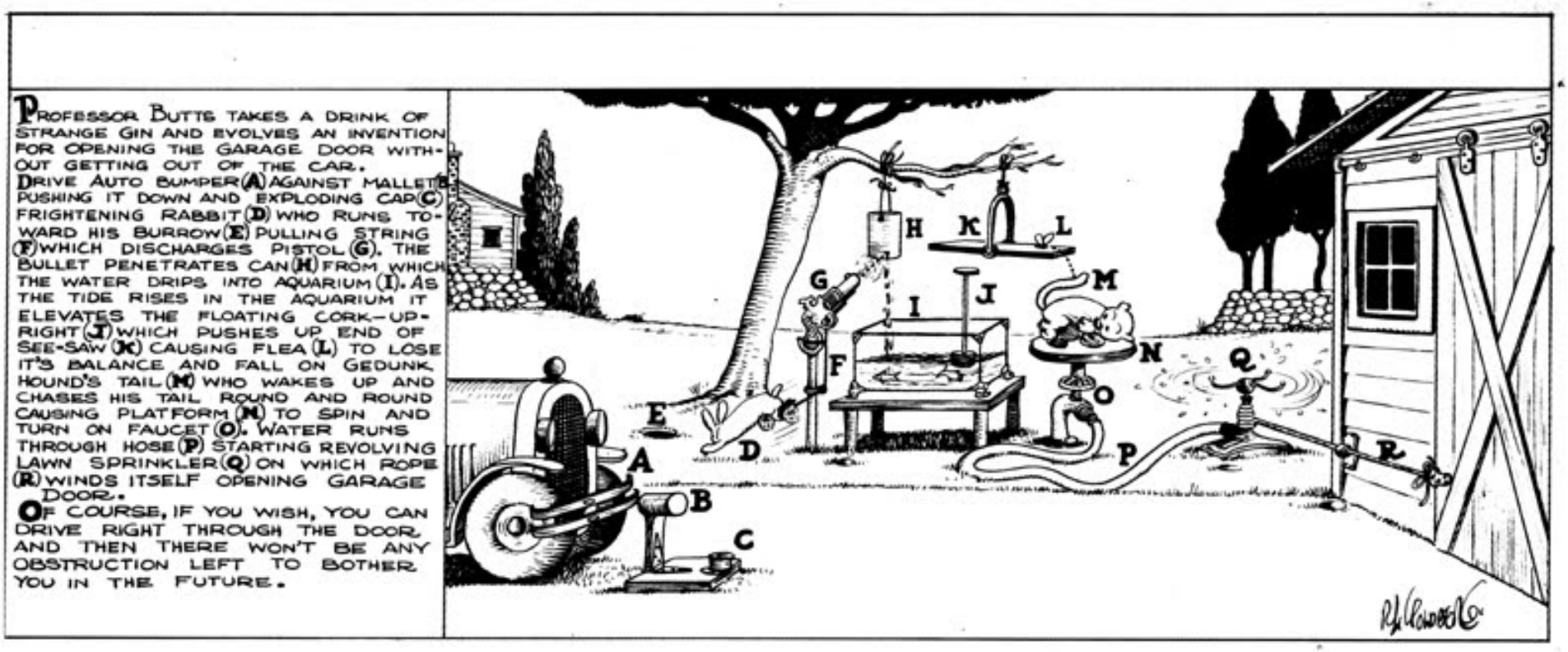
Which solution is the best or even feasible?

Courtesy of Ivan Marsic

26

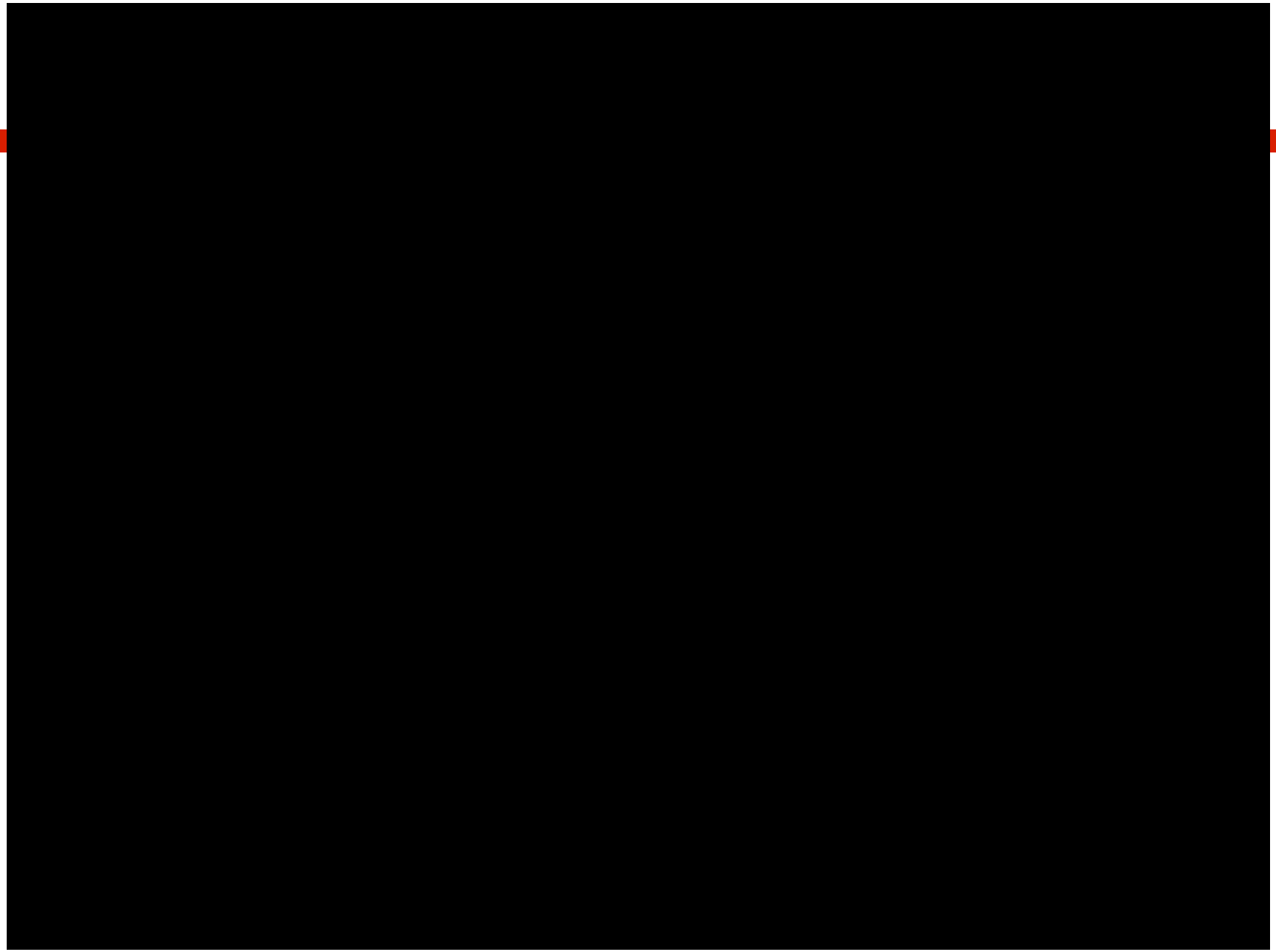
Rube Goldberg Design

Garage door opener

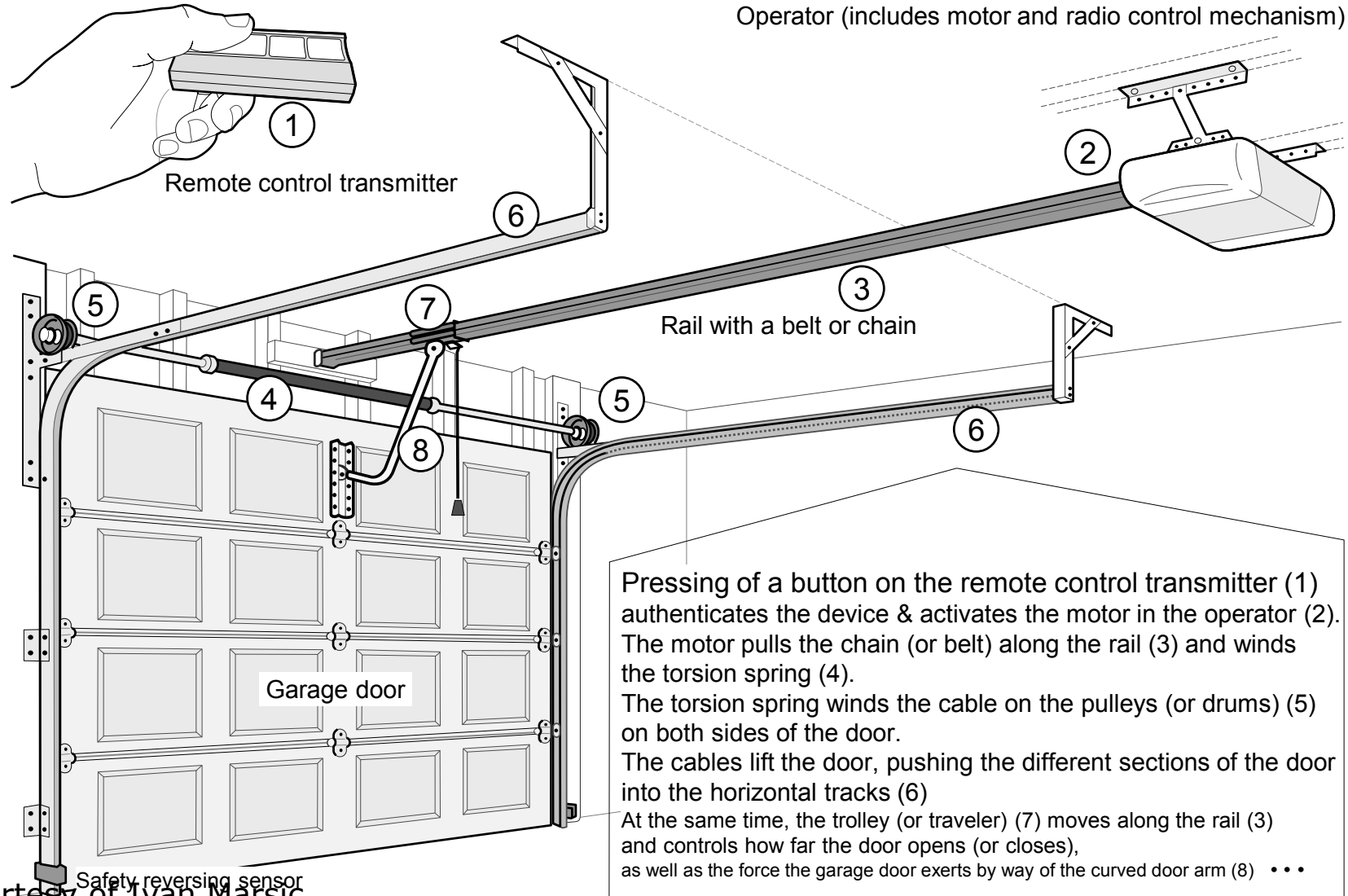


Courtesy of Ivan Marsic

27



Actual Design



Pressing of a button on the remote control transmitter (1) authenticates the device & activates the motor in the operator (2).
The motor pulls the chain (or belt) along the rail (3) and winds the torsion spring (4).
The torsion spring winds the cable on the pulleys (or drums) (5) on both sides of the door.
The cables lift the door, pushing the different sections of the door into the horizontal tracks (6)
At the same time, the trolley (or traveler) (7) moves along the rail (3) and controls how far the door opens (or closes),
as well as the force the garage door exerts by way of the curved door arm (8) • • •

Courtesy of Ivan Marsic

29

Software Measurement

- What to measure?
  - Project (developer's work),

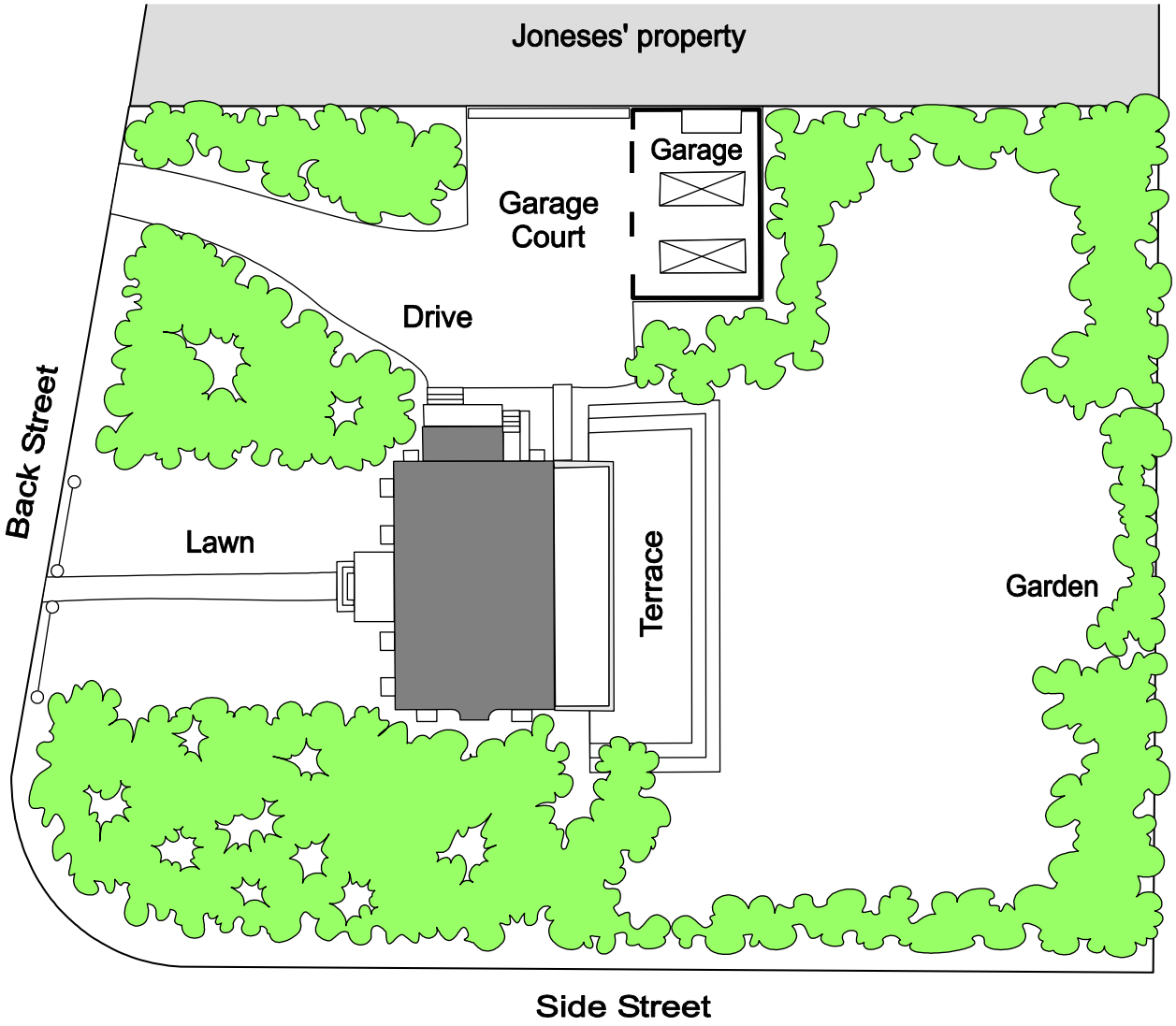
    for budgeting and scheduling
  - Product,

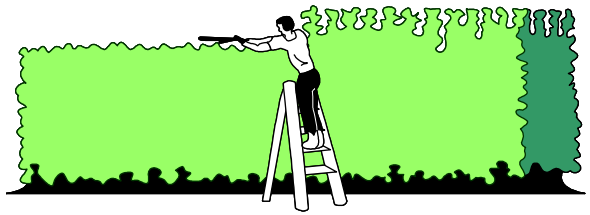
    for quality assessment

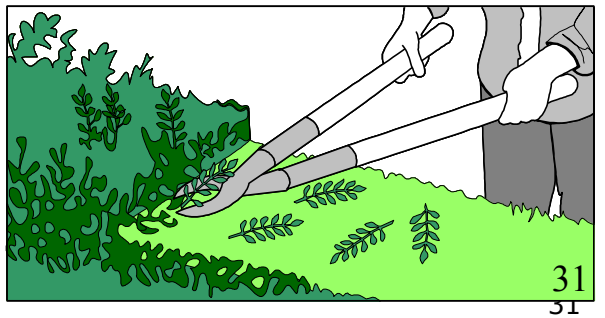
Courtesy of Ivan Marsic

30

Formal hedge pruning



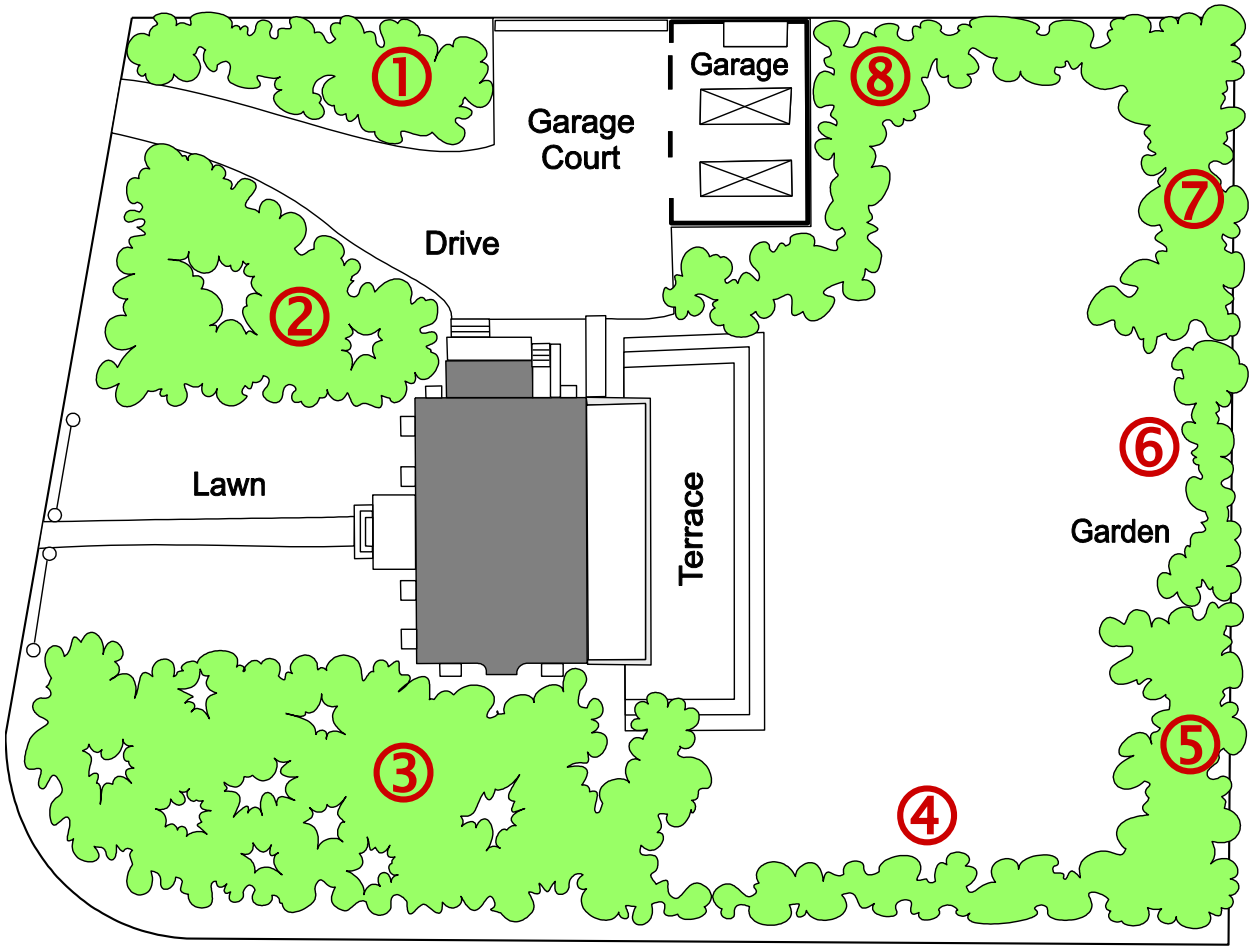




Courtesy of Ivan Marsic

Sizing the Problem (1)

Step 1: Divide the problem into *small* & *similar* parts



Step 2: Estimate *relative* sizes of all parts

Size(①) = 4

Size(②) = 7

Size(③) = 10

Size(④) = 3

Size(⑤) = 4

Size(⑥) = 2

Size(⑦) = 4

Size(⑧) = 7

Courtesy of Ivan Marsic

32

Sizing the Problem (2)

- Step 3: Estimate the size of the total work

$$\text{Total size} = \sum \text{points-for-section } \boldsymbol{i} \quad (i = 1..\text{N})$$

- Step 4: Estimate speed of work (velocity)

- Step 5: Estimate the work duration

$$\text{Travel duration} = \frac{\text{Path size}}{\text{Travel velocity}}$$

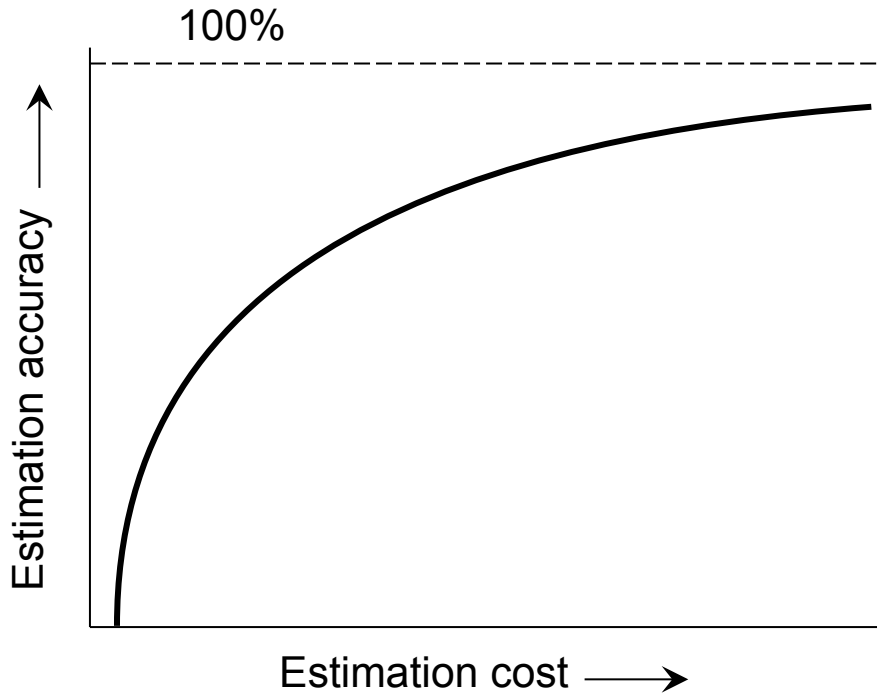
Courtesy of Ivan Marsic

Sizing the Problem (3)

- Advantages:
  - Velocity estimate may need to be adjusted (based on observed progress)
  - However, the total duration can be computed quickly (provided that the *relative* size estimates of parts are accurate – easier to achieve if the parts are **small** and **similar-size**)

Courtesy of Ivan Marsic

Exponential Cost of Estimation

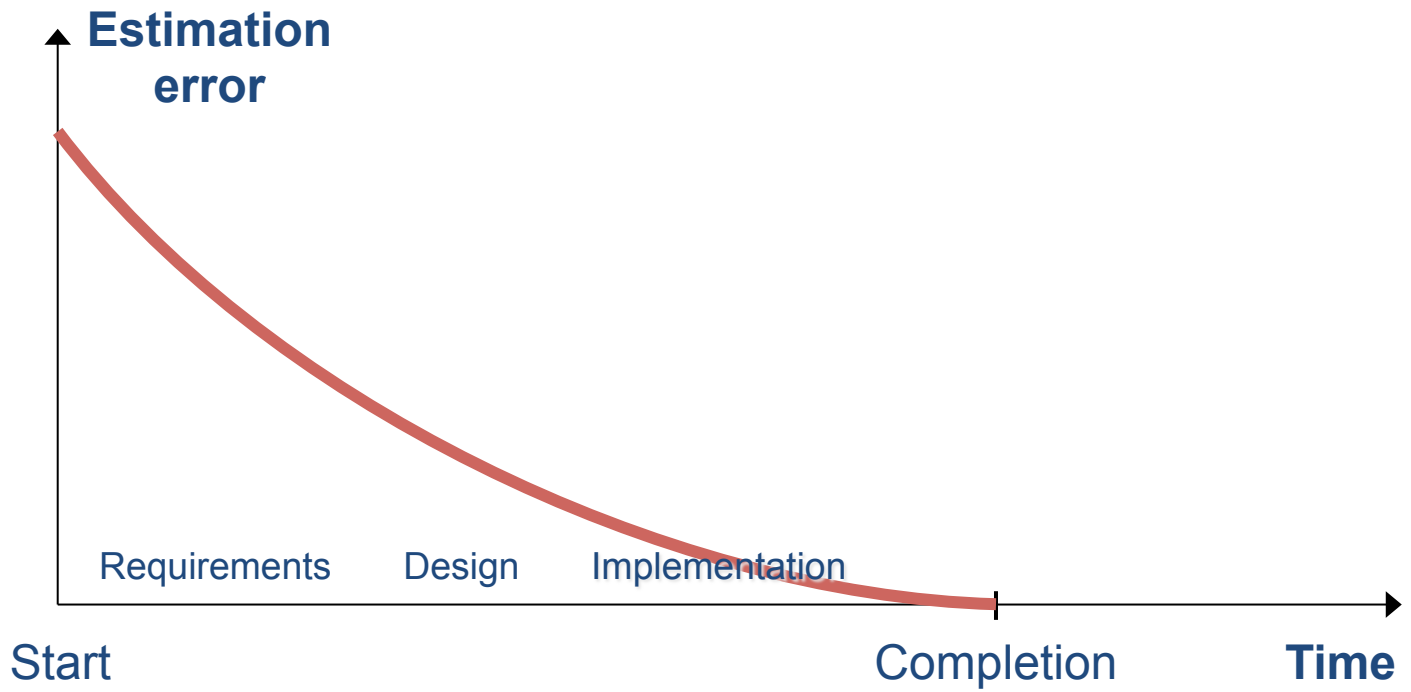


- Improving accuracy of estimation beyond a certain point requires huge cost and effort (known as the law of diminishing returns)
- In the beginning of the curve, a modest effort investment yields huge gains in accuracy

Courtesy of Ivan Marsic

35

Estimation Error Over Time

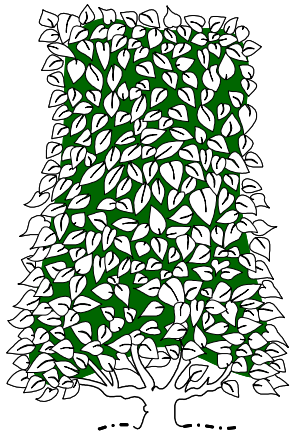


The *cone of uncertainty* starts high and narrows down to zero as the project approaches completion.

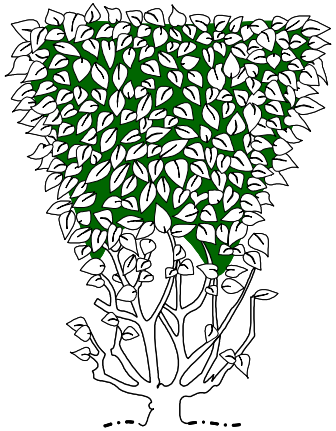
Courtesy of Ivan Marsic

36

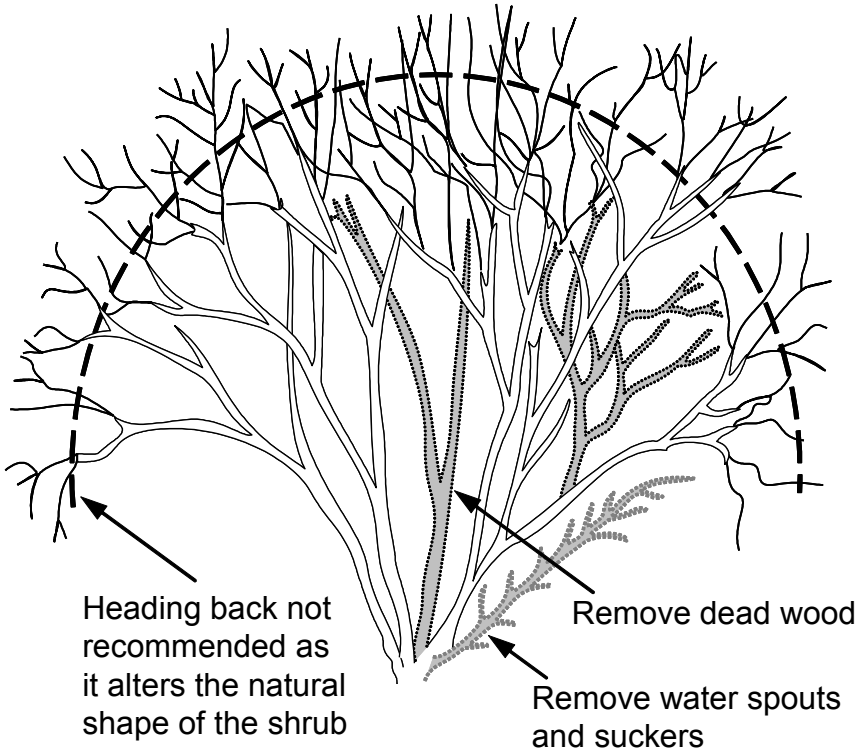
Measuring Quality of Work

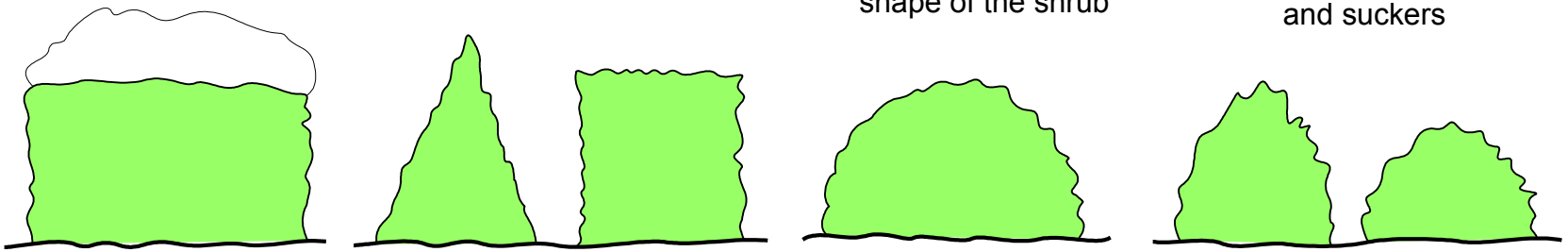


Good Shape
(Low branches get sun)



Poor Shape
(Low branches
shaded from sun)





Snow accumulates on broad flat tops

Straight lines require more frequent trimming

Peaked and rounded tops hinder snow accumulation

Rounded forms, which follow nature's tendency, require less trimming

Courtesy of Ivan Marsic

37

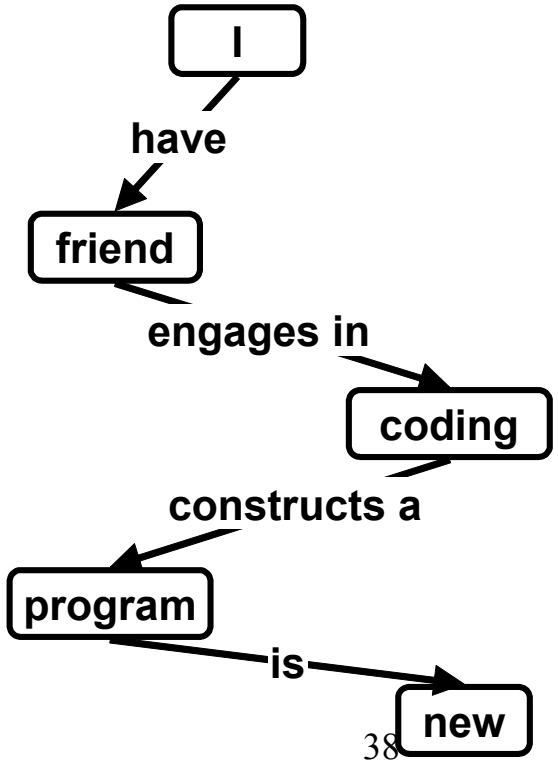
Concept Maps

Useful tool for problem domain description

SENTENCE: "My friend is coding a new program"

translated into propositions

Proposition	Concept	Relation	Concept
1.	I	have	friend
2.	friend	engages in	coding
3.	coding	constructs a	program
4.	program	is	new

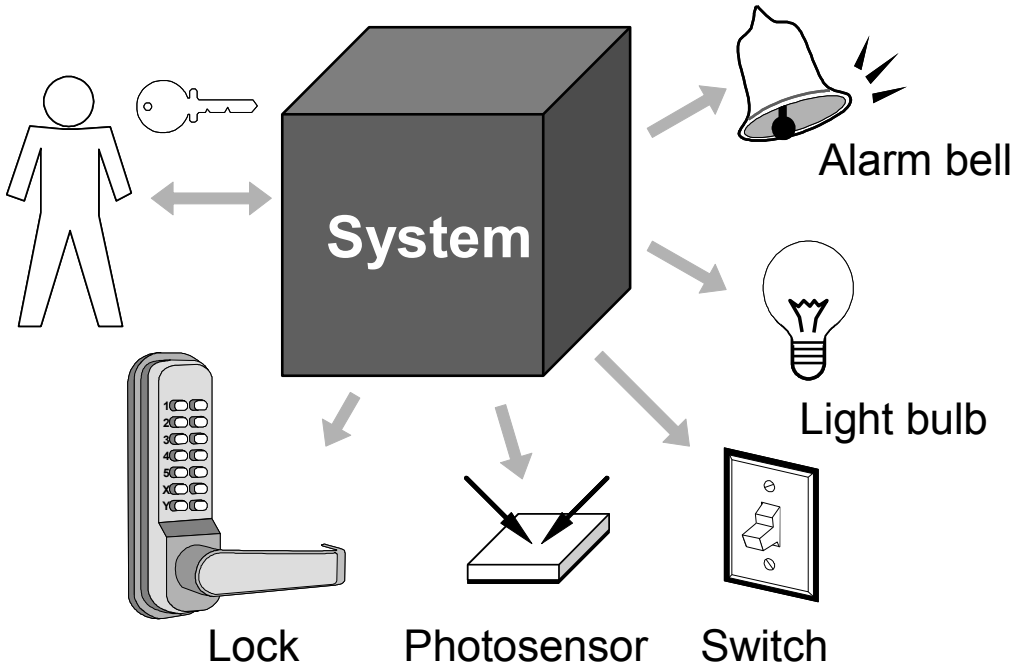


Search the Web for Concept Maps

Courtesy of Ivan Marsic

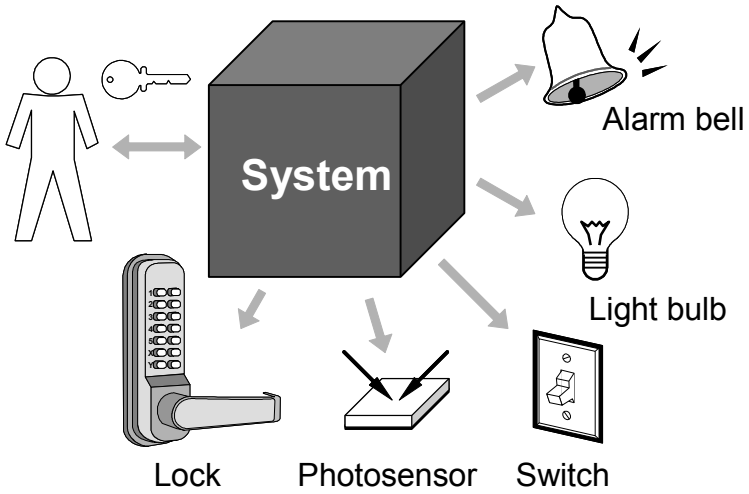
Case Study: Home Access Control

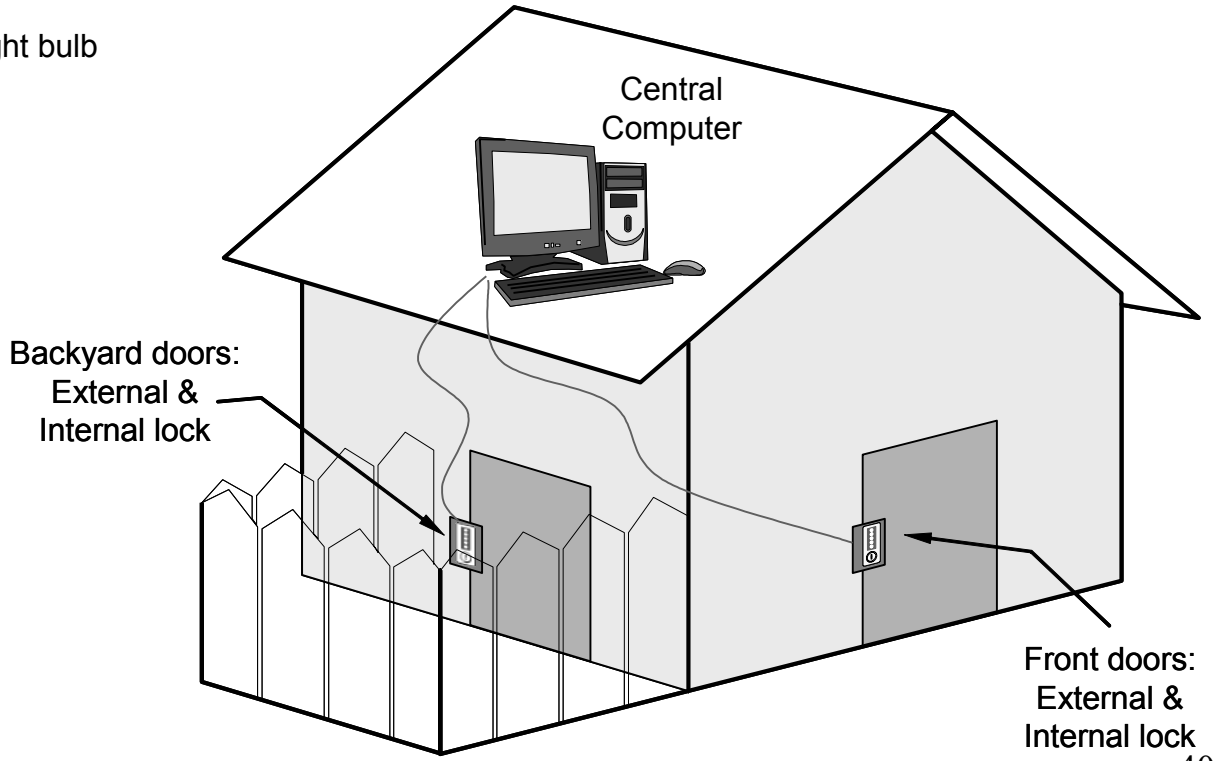
- Objective: Design an electronic system for:
  - Home access control
    - Locks and lighting operation
  - Intrusion detection and warning



Courtesy of Ivan Marsic

Case Study – More Details



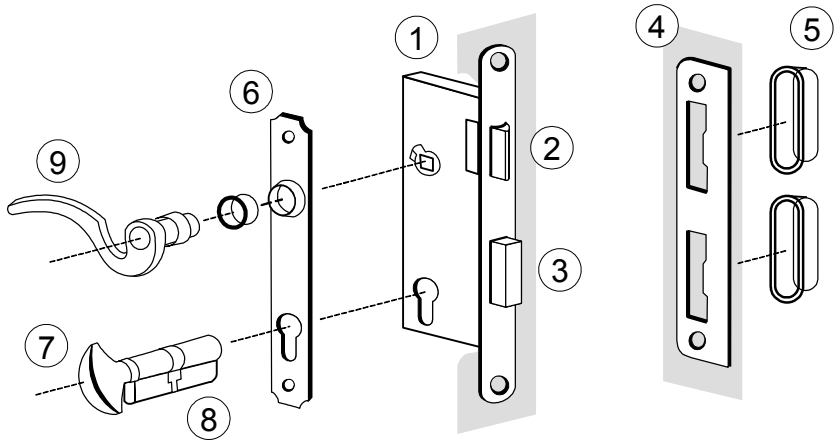


Courtesy of Ivan Marsic

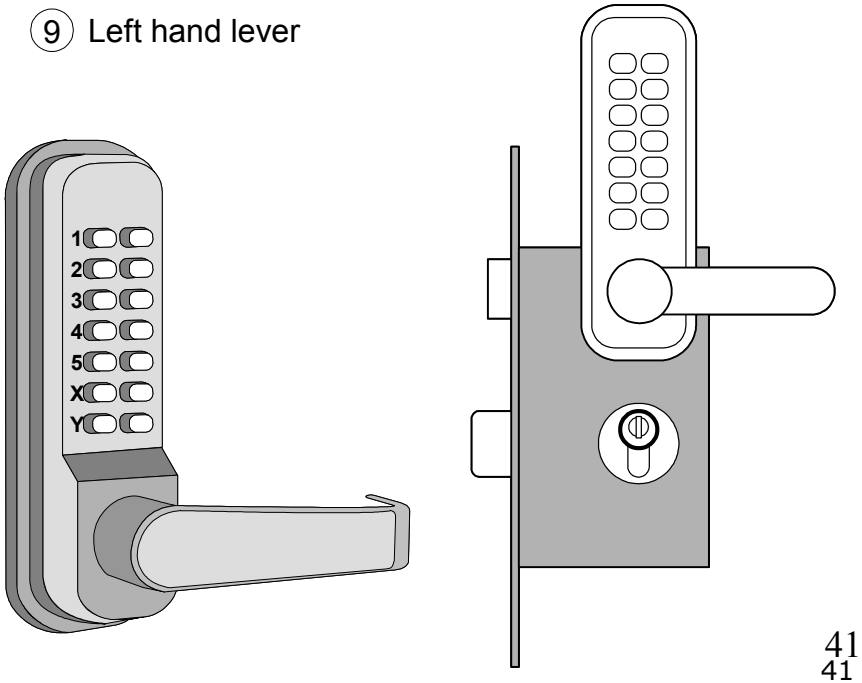
40

Know Your Problem

Mortise Lock Parts



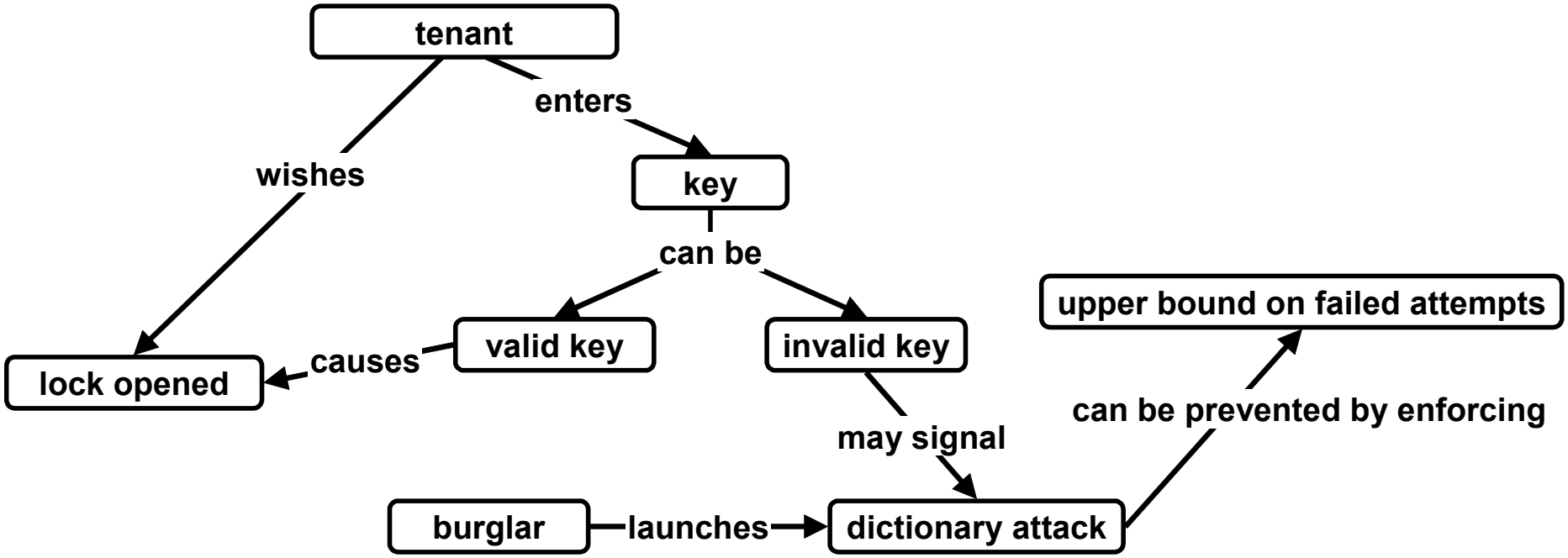
1. Lock case
2. Latch bolt
3. Dead bolt
4. Strike plate
5. Strike box
6. Protective plate
7. Thumb-turn
8. Lock cylinder
9. Left hand lever



Courtesy of Ivan Marsic

41

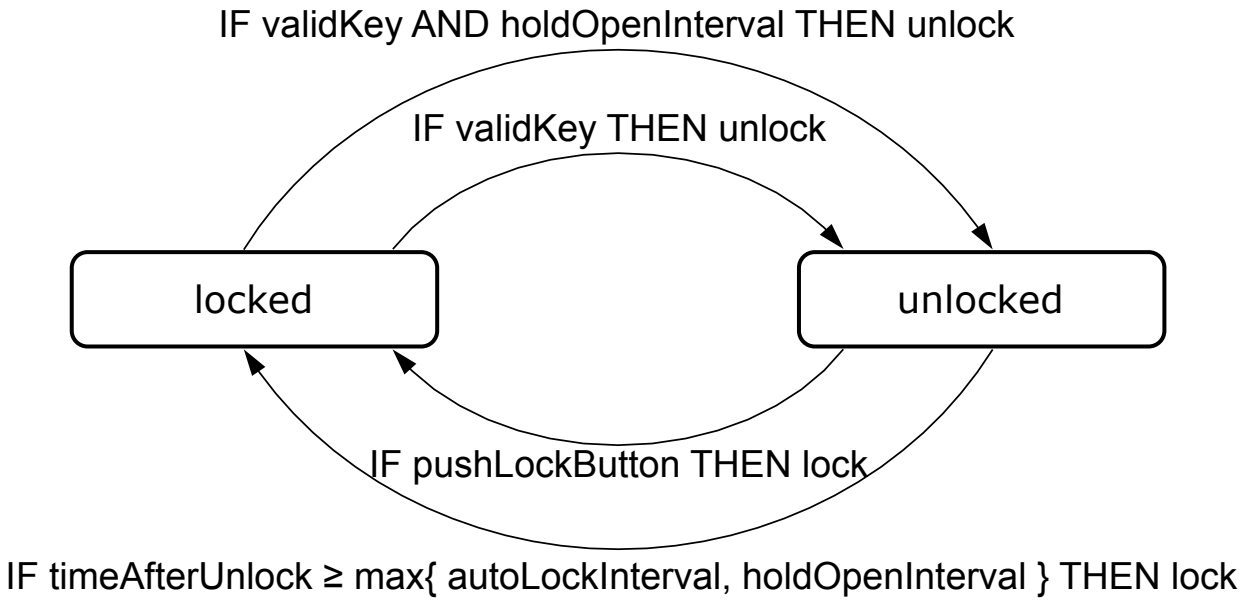
Concept Map for Home Access Control



Courtesy of Ivan Marsic

42

States and Transition Rules



… what seemed a simple problem, now is becoming complex

Courtesy of Ivan Marsic

43

Questions?