05506017
Software Engineering

Chapter 6 :
Requirement Engineering

ผศ.ดร.วรางคณา กิ้มปาน

ภาควิชาวิทยาการคอมพิวเตอร์ สจล.

Software Requirements

- A **requirement** specifies the business functions that the user will be able to perform using the system-to-be in different “situations” or “contexts”, and the kind of experience the user will have during this work
- Other concerns, such as how the system will manage the resources (computing, network, …), how the system will manage and protect user’s data, etc.

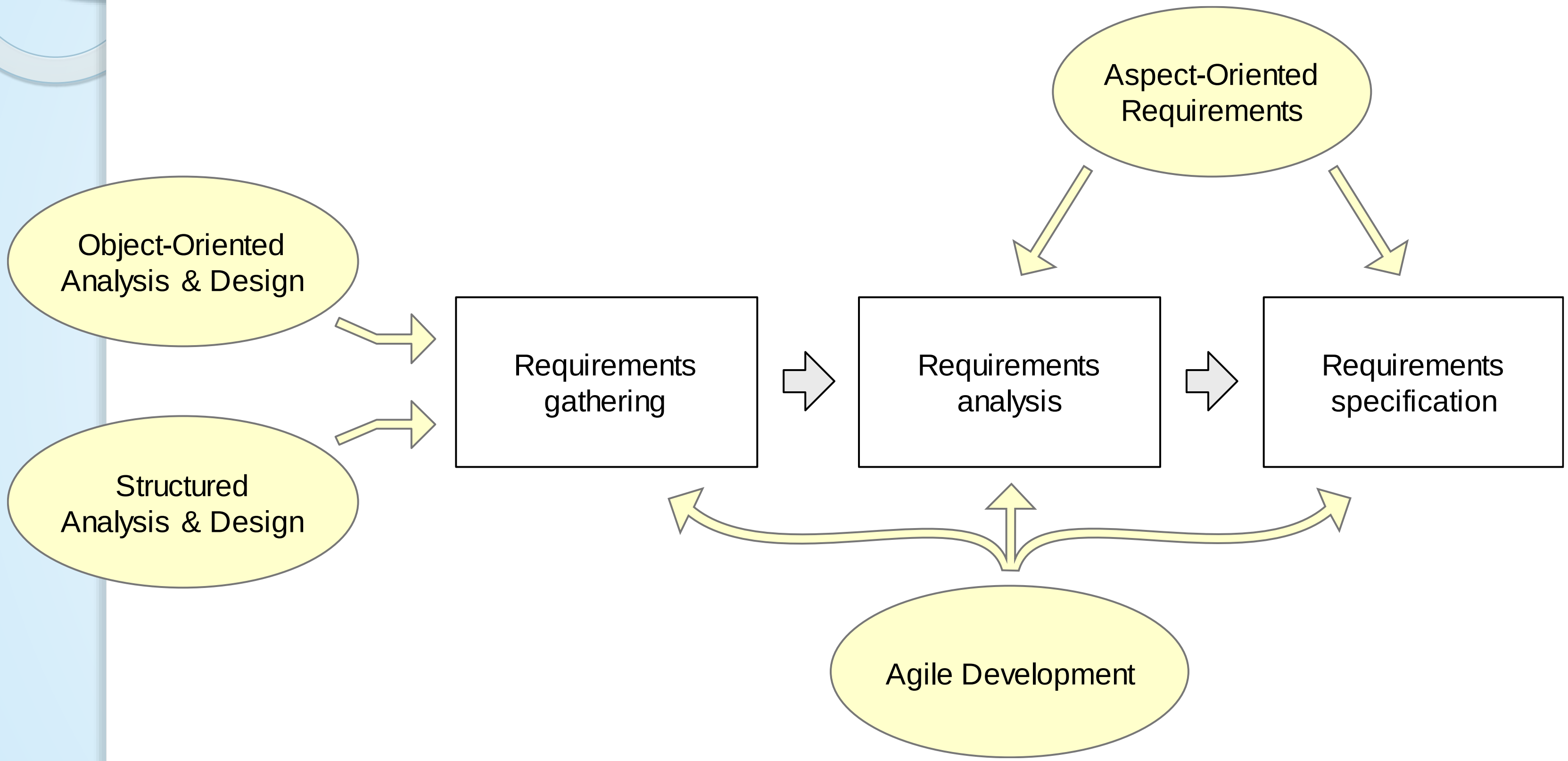
Software Requirements (2)

- User requirements will often be high-level, vague and incomplete. They are more like high-level goals, or business goals, rather than software requirements needed by the developer
- When trying to achieve a given high-level goal, we will need to consider what matters, what are the important parameters, so that we can derive the detailed technical requirements

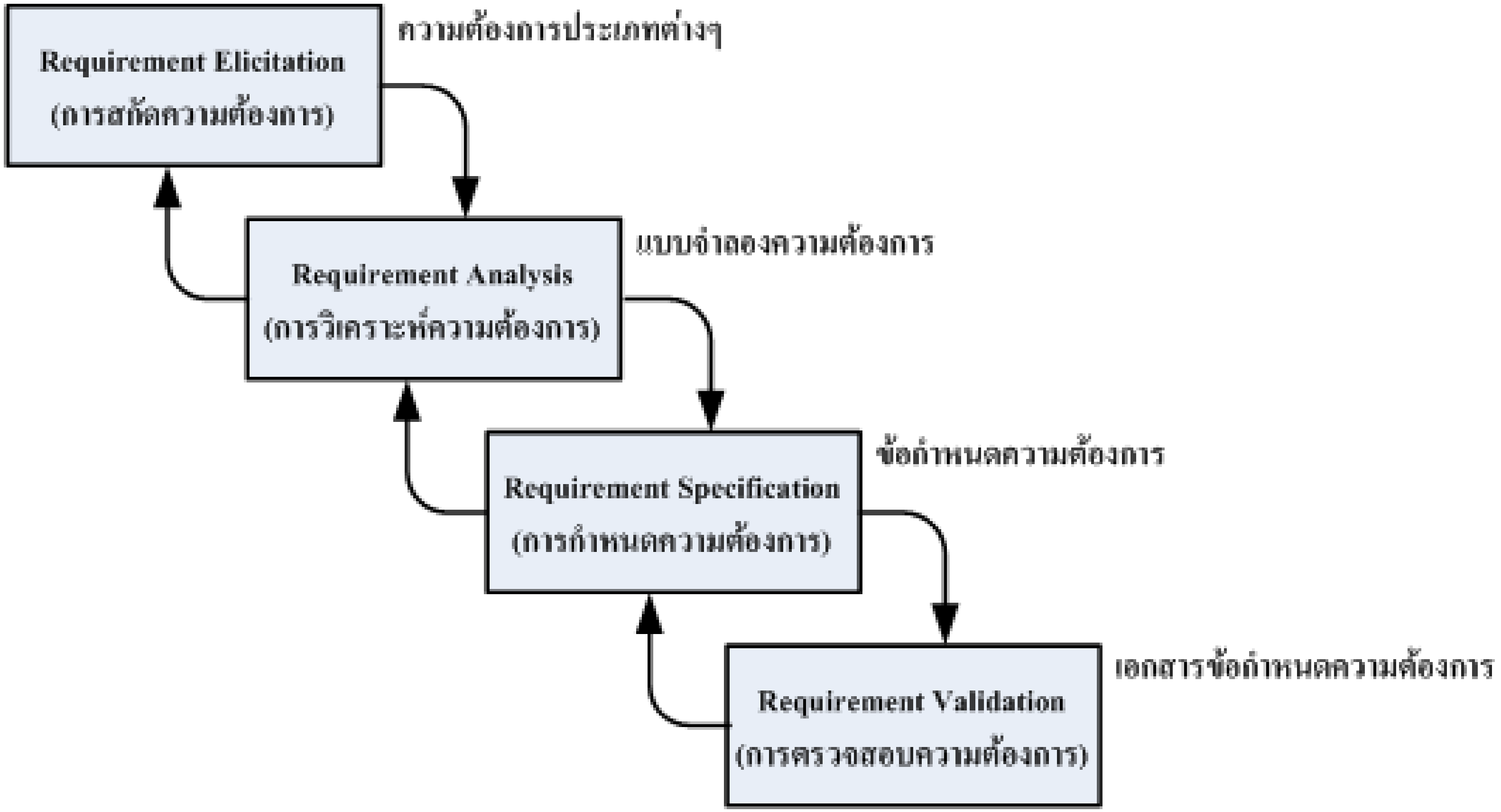
Software Requirements (3)

- Only based on deeper understanding of detailed issues, we can identify important "scenarios" or "situations" and identify what parameters should be considered in each situation
- Then using these parameters, we decide what the system should do, or how to respond to this situation (i.e., inputs)

Requirements Process


Aspect-Oriented Requirements
Object-Oriented Analysis & Design
Structured Analysis & Design
Requirements gathering
Requirements analysis
Requirements specification
Agile Development

Requirement Engineering



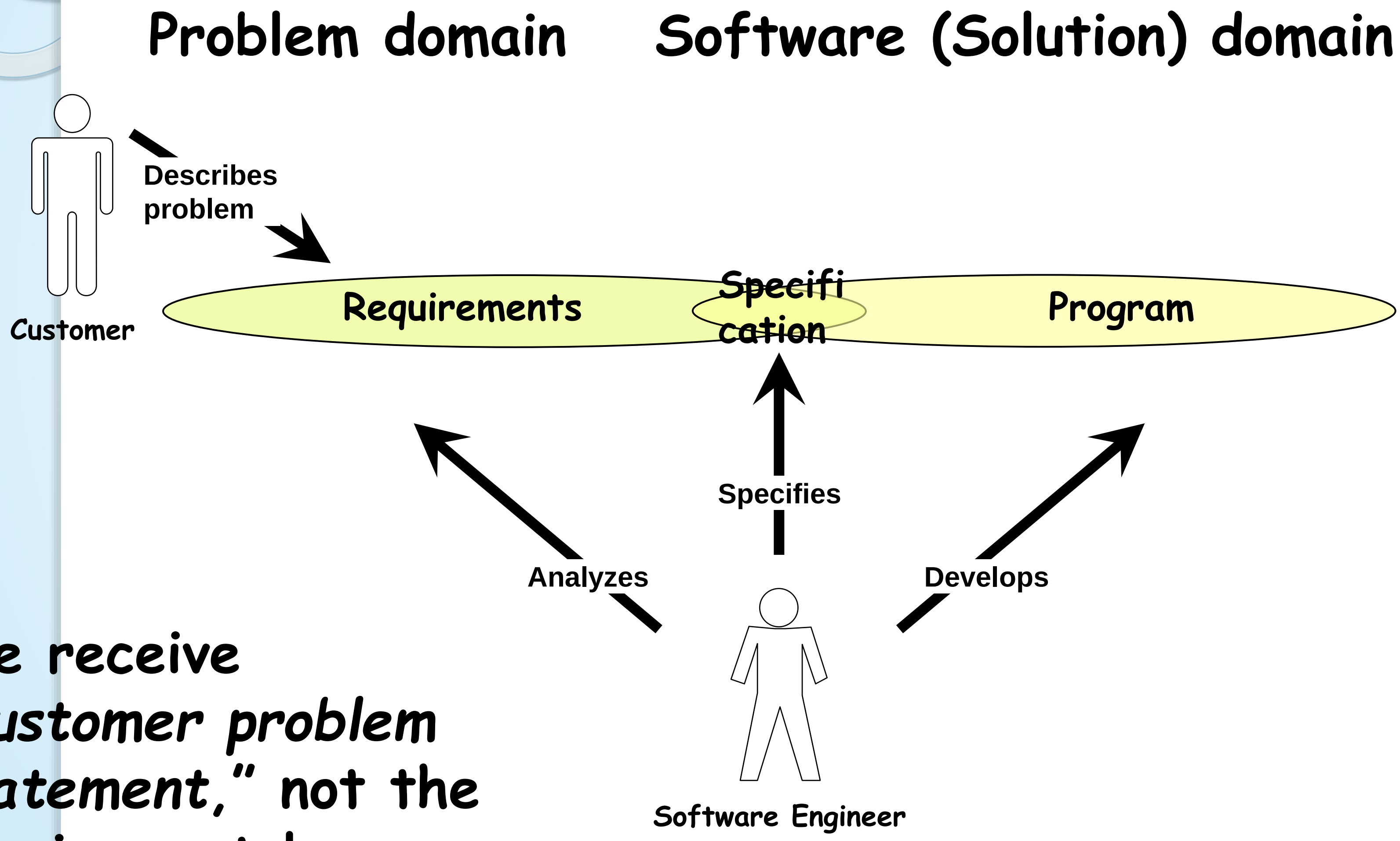
Requirements Engineering Components

1. Requirements gathering
    - ❑ also known as "requirements elicitation" helps the customer to define what is required: what is to be accomplished, how the system will fit into the needs of the business, and how the system will be used on a day-to-day basis
2. Requirements analysis
    - ❑ refining and modifying the gathered requirements
3. Requirements specification
    - ❑ documenting the system requirements in a semiformal or formal manner to ensure clarity, consistency, and completeness

Practical Requirements Engineering

- Test your idea in practice and use the result in further work, iterating through these creative and evaluative steps until a solution is reached
  - No one can know all the constraints for a solution before they go through the solving experience
- Define the criteria for measuring the success ("acceptance tests")
- Avoid *random* trial-and-error by relying on domain knowledge (from publications or customer expertise)

Requirements and Specification



Problem domain

Software (Solution) domain

Customer

Describes problem

Requirements

Specification

Program

Analyzes

Specifies

Develops

Software Engineer

We receive "*customer problem statement*," not the requirements!

Requirements Derivation

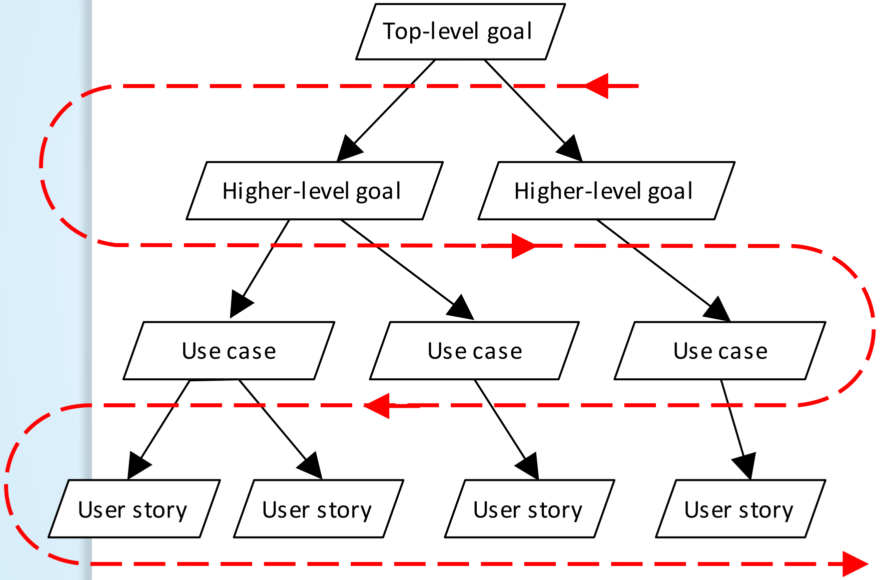
- *Detecting* that a problem exists is different from *defining* the problem and its causes, and solution constraints. Depending on the cause, the solution will be different.
- Requirements are determined by:
  - Judgment about customer's business goals and obstacles that currently are hindering their achievement
  - Conditions on solutions imposed by real-world constraints:
    - Physical
    - Social/Cultural
    - Legal
    - Financial
    - …
  - Threats created by adversaries

Requirements Derivation (2)

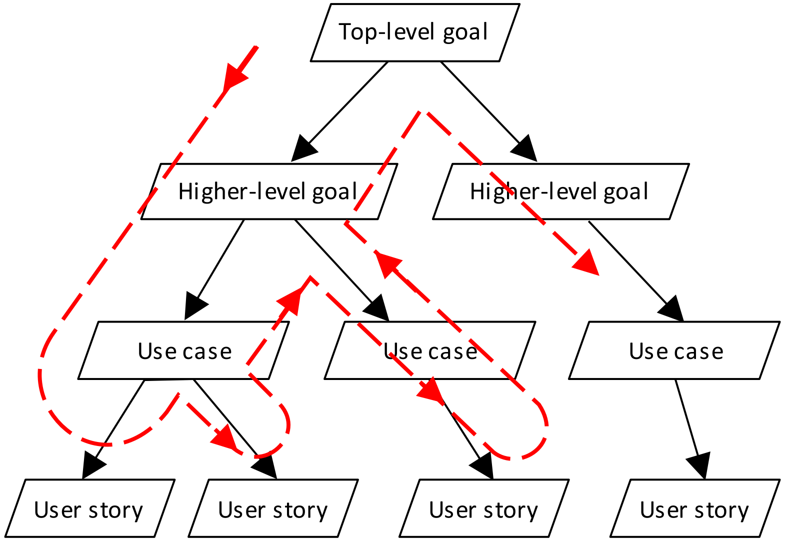
- → Requirements are **not** simply desires!
- Requirements are desires *adjusted to real-world constraints and threats*

Decomposition of Business Goals

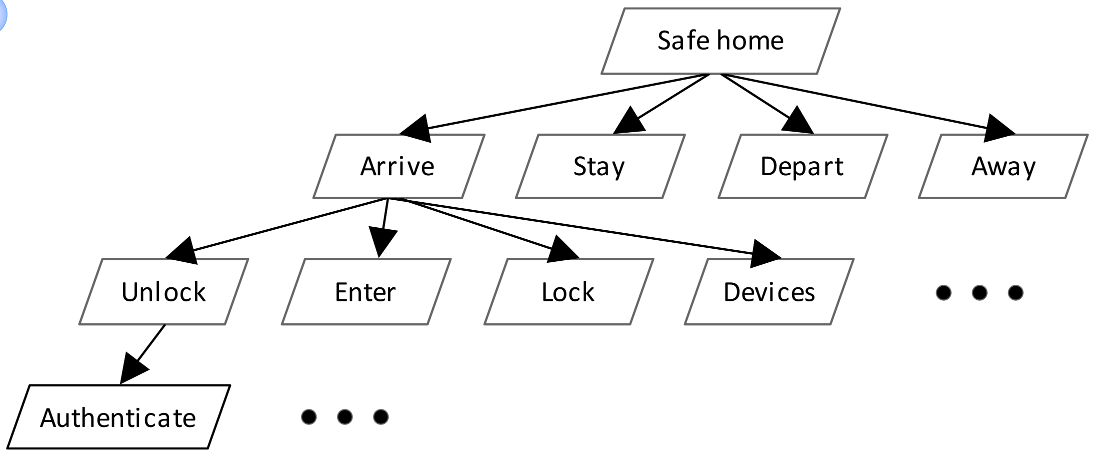
Breadth-first refinement



Depth-first refinement



Examples:
Safe Home Access /Restaurant Automation



Safe home

Arrive

Stay

Depart

Away

Unlock

Enter

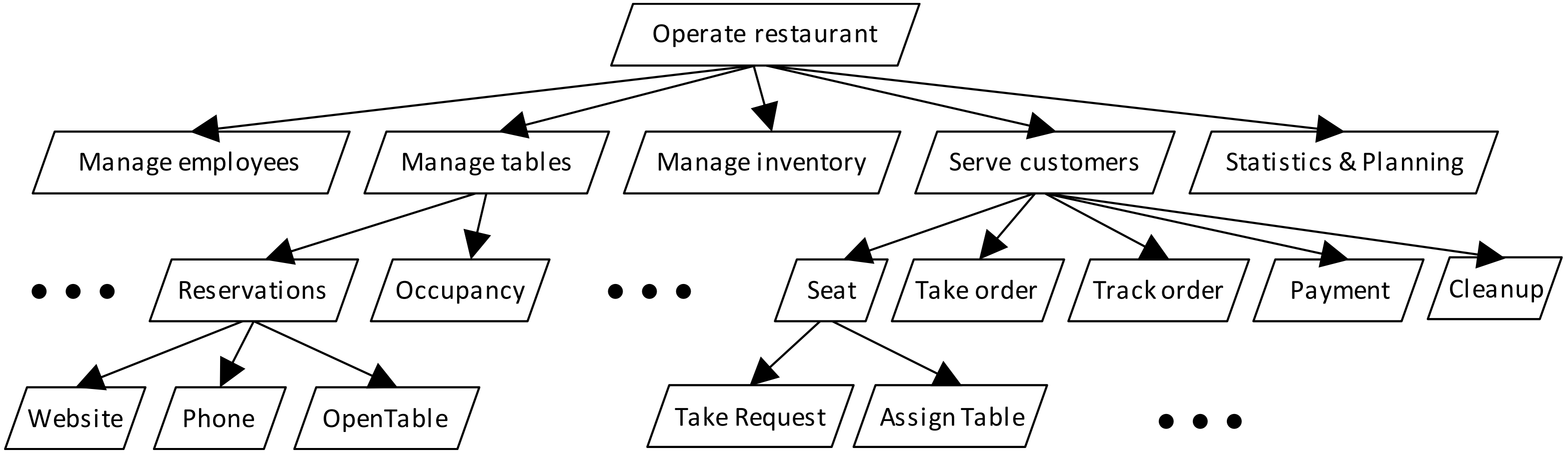
Lock

Devices

• • •

Authenticate

• • •



Operate restaurant

Manage employees

Manage tables

Manage inventory

Serve customers

Statistics & Planning

• • •

Reservations

Occupancy

• • •

Seat

Take order

Track order

Payment

Cleanup

Website

Phone

OpenTable

Take Request

Assign Table

• • •

Problem Analysis Examples

- Traffic Monitoring:

  Problem: User is delayed to work and needs help

  - Cause A: Traffic is unpredictable (predictably high if repeated delays?!)
  - Cause B: User is unfamiliar with the route
  - Cause C: User has a habit of starting late

Problem Example: Safe Home Access

Problem detected:

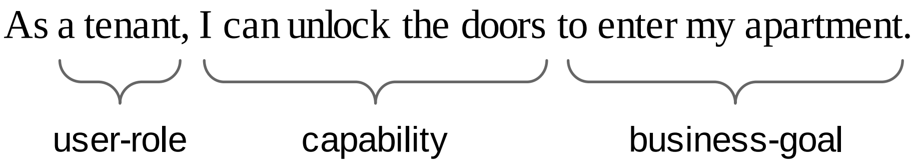
inconvenient physical keys or unwanted intrusion
(plus: operating household devices and minimizing living expenses)

Analysis of the Causes:

- User forgets to lock the door or turn off the devices
- User loses the physical key
- Inability to track the history of accesses
- Inability to remotely operate the lock and devices
- Intruder gains access

System Requirements: based on the selected causes

Requirements as User Stories

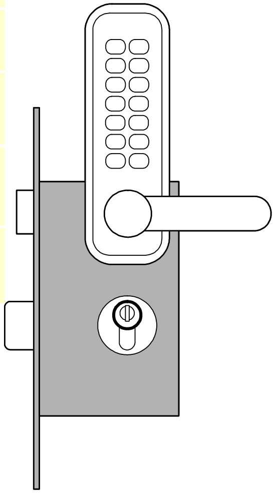

As a tenant, I can unlock the doors to enter my apartment.

- Preferred tool in **agile methods**.
- Stated in terms of user's goals and capabilities instead of system features
- Written by the customer or user, not by the developer.
- The development effort to implement a story is estimated immediately
- Acceptance tests are written when the story is identified
- Requirements identified only for the next iteration, not for the whole project

Example : User Story Requirements

Safe Home Access

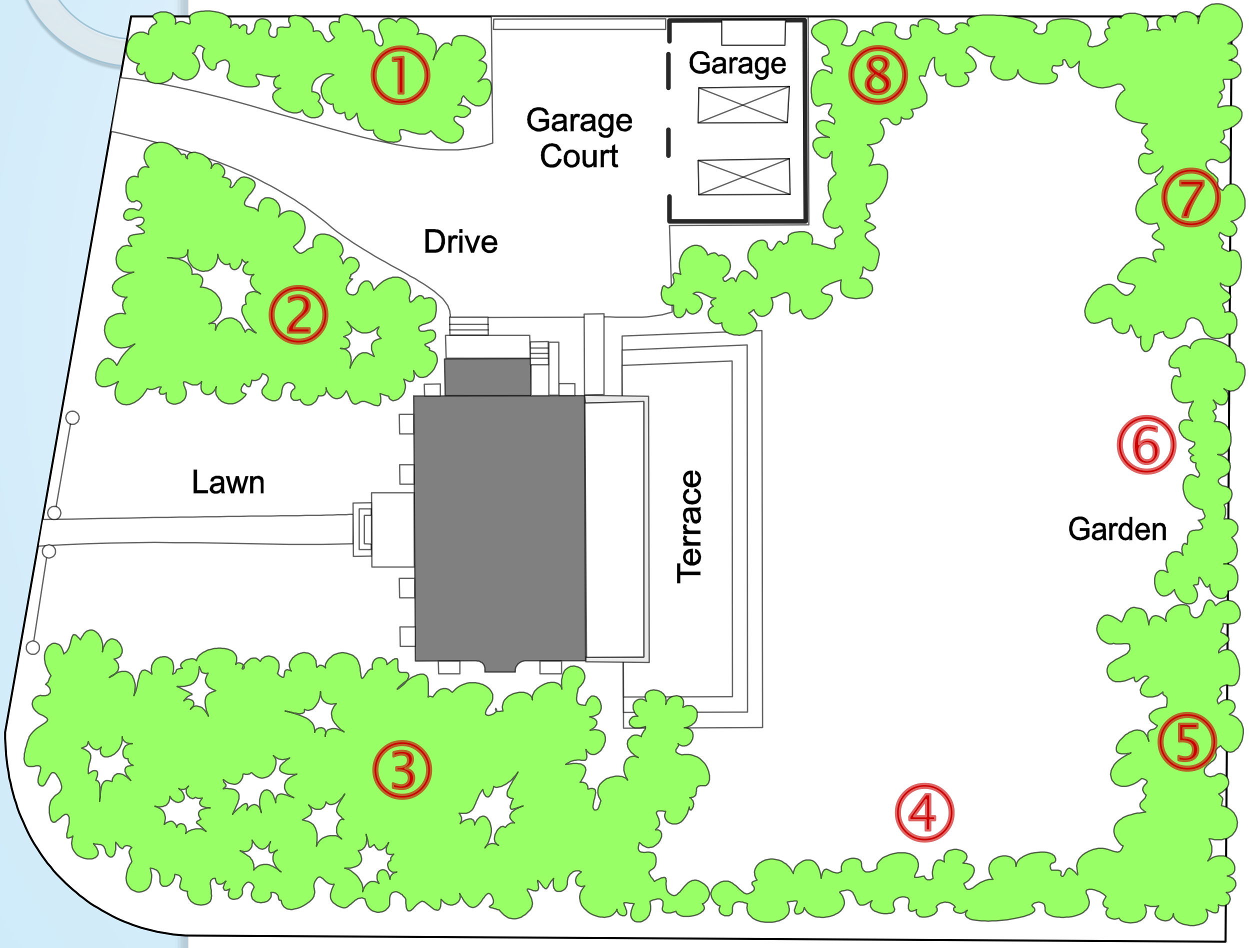
Identifier	User Story	Size
REQ-1	As a user, I can be sure that the doors by default will be locked.	**4 points**
REQ-2	As a user, I will be able to unlock the doors using a valid key.	**7 points**
REQ-3	An intruder will not be able to unlock the doors by guessing a valid key; the system will block when it detects a "dictionary attack."	**7 points**
REQ-4	As a user, I can be sure that the doors will be automatically locked at all times.	**6 pts**
REQ-5	The door keypad will be backlit when dark for visibility.	**3 pts**
REQ-6	Anyone will be able to lock the doors on demand.	**3 pts**
REQ-7	As a user, I will be able to manage additional user accounts.	**10 pts**
REQ-8	As a user, I will be able to view the history of accesses to my home.	**6 pts**
REQ-9	As a user, I will be able to configure the preferences for how my household devices will be activated on my arrival.	**6 pts**



- ❑ Note no priorities for user stories
  - Story priority is given by its order of appearance on the to-do list
- ❑ Estimated size points (last column) will be described later
- ❑ Compare to IEEE-830 style requirements
  - https://en.wikipedia.org/wiki/Software_requirements_specification

Sizing the Problem

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

Sizing the Problem (2)

Step 3: Estimate the size of the total work

Total size = Σ points-for-section ***i*** (*i* = 1..N)

Step 4: Estimate speed of work (velocity)

Step 5: Estimate the work duration

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

Sizing the Problem (3)

Assumptions:

- Relative size estimates are accurate
- That's why parts should be small & similar-size!

Advantages:

- Velocity estimate may need to be adjusted (based on observed progress)
- However, the total duration can be recomputed quickly
  - Provided that the relative size estimates of parts are accurate
    —accuracy easier achieved if the parts are small and similar-size

Unfortunately:

Unlike hedges, software is mostly **invisible** and **does not exist** when project is started
➔ The initial estimate hugely depends on experience and imagination

Example User Story Requirements

Restaurant Automation

Identifier	User Story	Size
REQ-1	As a host, I can take a seating request including customer party information, place into the seating queue, and have a table assigned or waiting time estimated.	10 points
REQ-2	As a waiter, I can input customer's order.	5 points
REQ-3	As a waiter, I can add special instructions to an order at the customer's request.	2 points
REQ-4	As a waiter, I can notify the chef of the order without walking to the kitchen.	2 pts
REQ-5	As a waiter, I can view customer's bill and enter their payment information.	7 pts
REQ-6	As a waiter, I will be notified when an order has been completed.	2 pts
REQ-7	As a chef, I can see the queue of orders waiting to be prepared.	3 pts
REQ-8	As a chef, I can mark orders as "In Preparation" and "Complete".	2 pts
REQ-9	As a chef, I can modify the menu to make certain dishes available or unavailable if supplies are limited.	6 pts
REQ-10	As a chef, I can adjust and update the supply inventory to let the manager know of any lacking ingredients.	7 pts

- After requirements analysis:

Identifier	User Story	Size
REQ-2a	As a waiter, I can input orders at different times for customers at the same table.	7 points
REQ-2b	As a waiter, I can input different courses at different times for the same customer.	5 pts
REQ-2c	As a waiter, I can input side plates after the main course is served.	2 pts

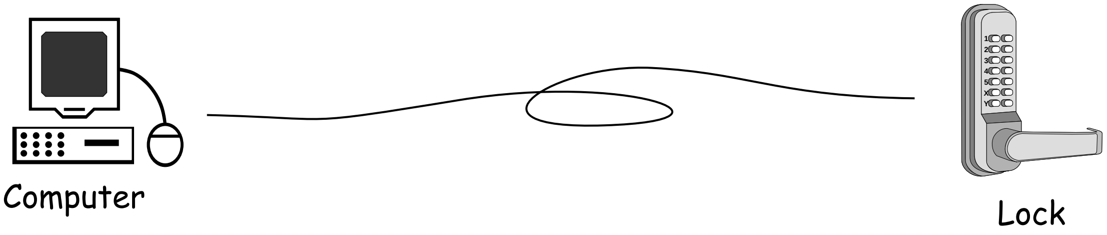
Requirements Analysis

- Requirement REQ-3 states that intruders will not be able to succeed with a “dictionary attack,” but many details need to be considered and many parameters determined (“business policies”)
  - What distinguishes user’s mistakes from “dictionary attacks”
    - The number of allowed failed attempts, relative to a predefined threshold value
      - The threshold shall be small, say three ← business policy!
  - How is the mechanical lock related to the “blocked” state?
    - Can the user use the mechanical key when the system is “blocked”?

Requirements Analysis (2)

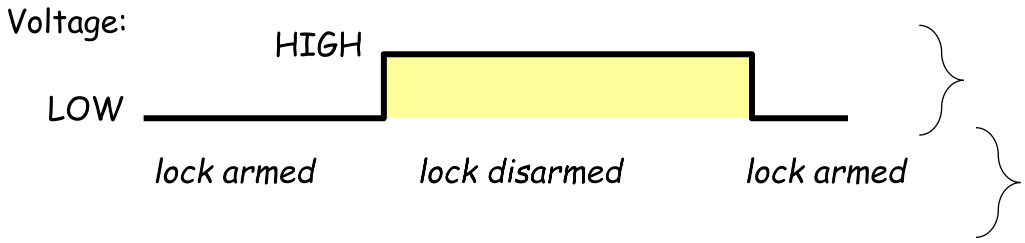
- Requirement REQ-5 states that the keypad should be backlit when dark
  - Is it cost-effective to detect darkness vs. keep it always lit?
- Etc.
- Requirements analysis should not be exhaustive, but should neither be avoided.

Problem Domain:
How Electronic Lock Works



We may need separate descriptions/models of door vs. lock.

Door state is what the user cares about; lock is one way of achieving it.



Physical domain (lock) description; used in specification

Semantic meaning defined by user's goals; used in requirements

The behavior of the system-to-be determined not only by user's actions
but also by the context ("situation").

E.g., what in case of power failure?
- By default armed
- By default disarmed (e.g., fire exit)

Analyst's Task: Three Descriptions

The ***requirement***

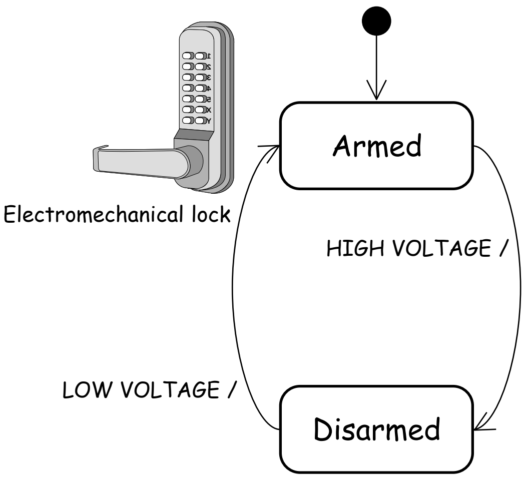
What user wants:

When valid keycode entered & Unlock pressed, open the lock;

Automatically lock after a period of time.

The ***problem domain***

How problem domain behaves:



The ***specification***

What software-to-be will do (at interface):

If entered number matches one of stored numbers & Button-1 pressed, put HIGH voltage on Output-port-1;

Start a timer countdown;

When the timer expires, put LOW voltage on Output-port-1.

Concern:

It is not obvious that this is the only or even "correct" solution to the requirement-posed problem.

Problem Frames tell us what each description should contain and how to verify the concern.

From Requirements to Business Policies

- Not only refinement of customer requirements, but also feasibility and how realistic
- Needs to identify the points where **business policies** need to be applied.

Explicit identification of **business policies** is important for two reasons:

1. Making the need for BP explicit allows involving other stakeholders, particularly the customer, in decision making about the BP solutions to adopt

2. Helps to anticipate potential future changes in the policies, so mechanisms can be implemented in the code that localize the future changes and allow quick substitution of implemented business policies

These issues too important to be left to the programmer to make ad-hoc decisions and hard-code them.

Types of Requirements

- Functional Requirements
- Non-functional requirements (or quality requirements)
  - FURPS+
  - Functionality (security), Usability, Reliability, Performance , Supportability
- User interface requirements

Tools for Requirements Eng.

- Tools, such as user stories and use cases, used for:
  - Determining what exactly the user needs ("requirements analysis")
  - Writing a description of what system will do ("requirements specification")
  - Difficult to use the same tool for different tasks (analysis vs. specification)

Acceptance Tests

- Means of assessing that the *project success criteria*
- The requirements are met as expected
- Conducted by the customer throughout the project, not only at the end
- An acceptance test describes whether the system will pass or fail the test, given specific input values
- We cannot ever guarantee 100% coverage of all usage scenarios, but *systematic approach* can increase the *expected* degree of coverage

Acceptance Tests (2)

- Each *requirement* describes for a given "situation" (i.e., system *inputs*), the *output* or behavior the system will produce
  - The "output" represents the user's need or business goal
- An **acceptance test** specifies a set of scenarios for determining whether the (part of the) system meets the customer requirements
- An **acceptance test** ***case*** specifies, for a given "situation" or "context" (defined by current system inputs), the output or behavior the system will produce in response

Example User Stories

Identifier	User Story	Size
REQ-1	As an authorized user, I will be able to keep the doors by default always locked.	**4 points**
REQ-2	As an authorized user, I will be able to unlock the doors using a valid key.	**7 points**
REQ-3	An intruder will not be able to unlock the doors by guessing a valid key; the system will block upon a “dictionary attack.”	**7 points**
REQ-4	The door will be automatically locked after being open for a defined period of time.	**6 pts**
REQ-5	As a user, I will have the door keypad backlit when dark for visibility.	**3 pts**
REQ-6	Anyone will be able to lock the doors on demand.	**2 pts**
REQ-7	As a landlord, I will be able at runtime to manage user authorization status.	**10 pts**
REQ-8	As an authorized user, I will be able to view the history of accesses and investigate “suspicious” accesses.	**6 pts**
REQ-9	As an authorized user, I will be able to configure the preferences for activation of household devices.	**6 pts**

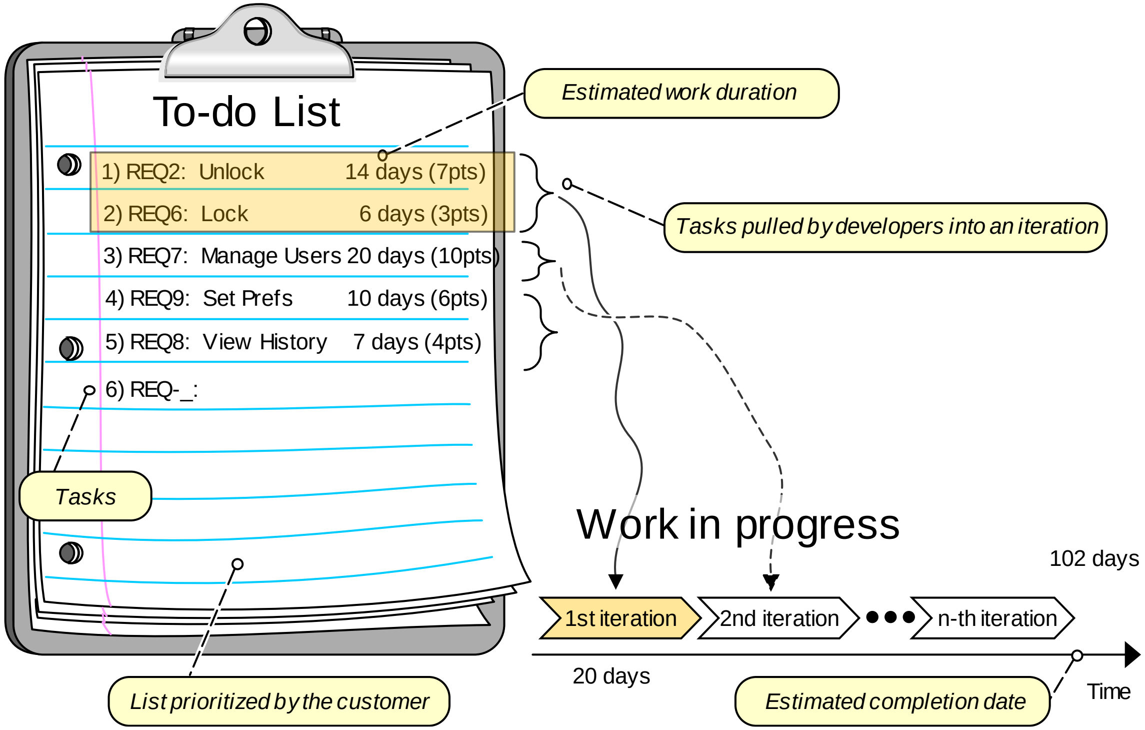
Agile Estimation of Project Effort

- ❑ Instead of assigning priorities, the customer creates an ordered list of user stories ➔ TO-DO LIST
- ❑ Developers simply remove the top list items and work on them in the next iteration ➔ IN-PROGRESS LIST

Requirements and estimated effort

2 days of work per 1 point

REQ1)	Default Locked	4pts →	8 days
REQ2)	Unlock	7pts →	14 days
REQ3)	Prevent Attack	7pts →	14 days
REQ4)	Autolock	6pts →	12 days
REQ5)	Backlit	3pts →	6 days
REQ6)	Lock	3pts →	6 days
REQ7)	Manage Users	10pts →	20 days
REQ8)	View History	6pts →	12 days
REQ9)	Set Preferences	6pts →	12 days
			102 days



To-do List

1) REQ2: Unlock 14 days (7pts)

2) REQ6: Lock 6 days (3pts)

3) REQ7: Manage Users 20 days (10pts)

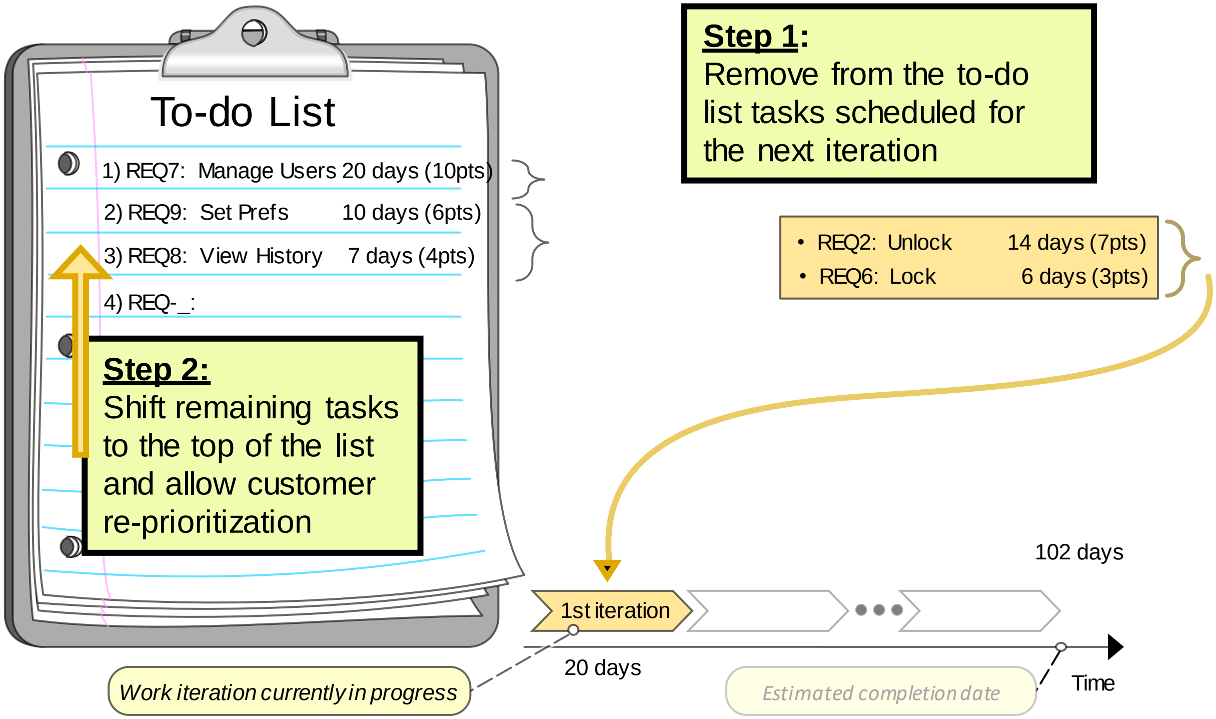
4) REQ9: Set Prefs 10 days (6pts)

5) REQ8: View History 7 days (4pts)

6) REQ-_:

Tradeoff between Customer Flexibility and Developer Stability

- Items pulled by developers into an iteration are not subject to further customer prioritization
- Developers have a **steady goal** until the end of the current iteration
- Customer has **flexibility** to change priorities in response to changing market forces



To-do List

1) REQ7: Manage Users 20 days (10pts)
2) REQ9: Set Prefs 10 days (6pts)
3) REQ8: View History 7 days (4pts)
4) REQ-_:

Step 1:
Remove from the to-do list tasks scheduled for the next iteration

- REQ2: Unlock 14 days (7pts)
- REQ6: Lock 6 days (3pts)

Step 2:
Shift remaining tasks to the top of the list and allow customer re-prioritization

1st iteration
102 days
20 days
Time
Work iteration currently in progress
Estimated completion date