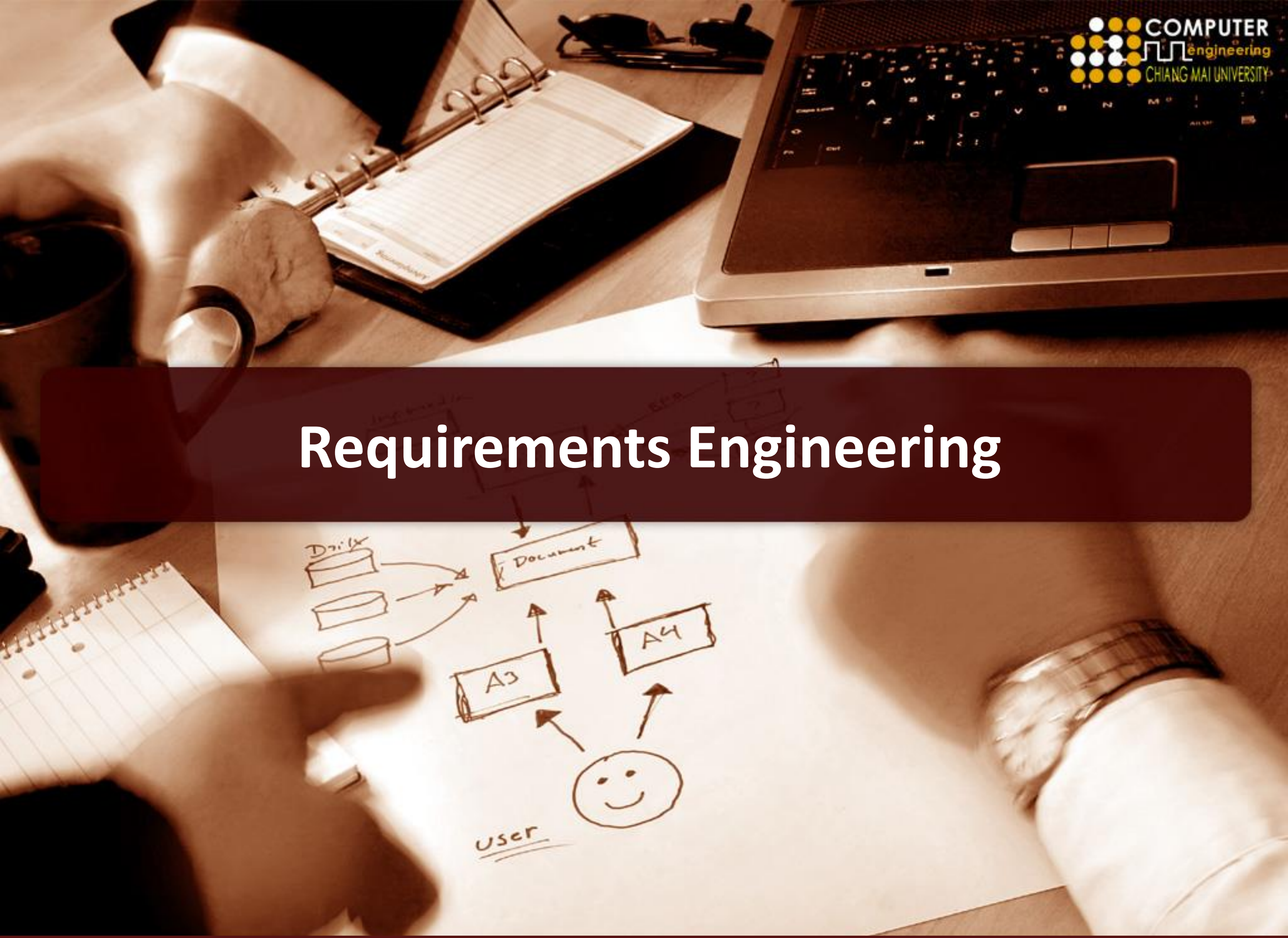


COMPUTER engineering CHIANG MAI UNIVERSITY

Requirements Engineering

Course Objective

- Students understand requirements engineering process and are able to apply the process to create a requirements document.

Course Info

- Course description
  - Basics of software engineering
  - Software development processes
  - Requirements engineering
  - Software design
  - Software construction and software testing
  - Software operation maintenance
  - Software quality
  - Advanced topics

Recap: Software Processes

- A process to define, implement and support software.
- Generally classified into two types
  - Plan-driven processes
  - Agile processes
- There are no right or wrong software processes. Only what works to deliver the expected results.

Overview

- Requirements
- Processes
- Activities
  - Elicitation
  - Specification
  - Validation
  - Management

** The contents presented in this slide are adapted from Ian Sommerville’s Software Engineering book.

Requirements Engineering

- Requirements originate from problems.
- The main purpose is to develop clear understanding of the software requirements.
- Why doing it?
- Customers/end users do not always know what is required. They do have not a good understanding of features and functions needed.

Requirements and Engineering

- What is it?
- Who does it?
- Why is it important?
- What are the steps?
- What is the work product?
- How to ensure it was done right?

-- Pressman, Software Engineering

Requirements Engineering

- Tasks and techniques that lead to understanding of the requirements.
- Major software engineering activities to communicate and model the requirements to meet user needs.
- Establish a solid base for design and construction.

Requirements

- States WHAT not HOW
- Service, function, feature, conditions, constraints, capabilities business rules representing user needs or system requirements in the following forms.
  - High-level abstract
  - Detailed mathematical functional specification
- Function of requirements
  - Used for a contract, either to bid or to be used.

Example: Requirements

- Which is a requirement?
  - A comfortable place to sit, or
  - An engine, or
  - A steering wheel
  - A maintainable steering capability

- Referenced from Agilebusiness.org, The DSDM Agile Project Framework - HANDBOOK - Requirements and User Stories, retrieved from https://www.agilebusiness.org/content/requirements-and-user-stories

- DSDM: Dynamic Systems Development Method

Example: Requirements

- “The Mentcare system shall generate monthly management reports showing the cost of drugs prescribed by each clinic during that month.”
  - User requirement
- “A report shall be created for each clinic and shall list the individual drug names, the total number of prescriptions, the number of doses prescribed and the total cost of the prescribed drugs.”
  - System requirement
- See Mentcare requirements document

Types of Requirements

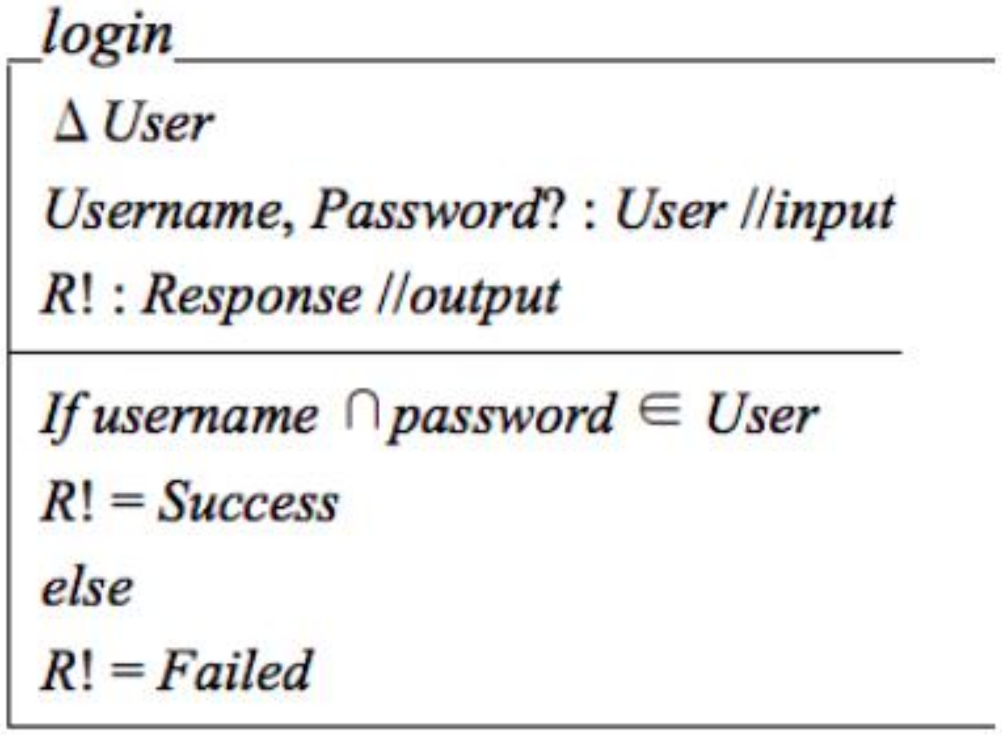
- User requirements
  - Natural language statements plus diagrams
  - Intended audiences are customers, end-users, architects, contract managers, etc.
- System requirements
  - Structured detailed description of system’s functions/features, services and operational constraints.
  - Audiences include end-users, technical persons on customer side, developers, architects, analysts, testers, etc.

Formal Specification Language

- Z notation, originally proposed in 1977 by Jean-Raymond Abrial, is a formal specification language used for modeling computing systems.
- The purpose is to provide the clear specification of computer programs.
- See a glossary of Z notation.

Z Notation

Z Schema for the login method for class user



```
login
  Δ User
  Username, Password? : User //input
  R! : Response //output
  ──────────────────────────
  If username ∩ password ∈ User
  R! = Success
  else
  R! = Failed
```

Referenced from

– Bakri, S. H., Harun, H., Alzoubi, A., Ibrahim, R., Jamaludin, Z., ChePa, N., & Bakar, M. S. A. (2013, August). The formal specification for the inventory system using Z language. In *4th International Conference on Computing and Informatics, ICOCI* (pp. 28-30).

Stakeholders

- Individual or Organization
- Owner, user or anyone directly or indirectly involve with the system
- Include
  - “End users, end user organizations, supporters, developers, producers, trainers, maintainers, disposers, acquirers, customers, operators, supplier organizations and regulatory bodies.”

- Referenced from SEBOK wiki

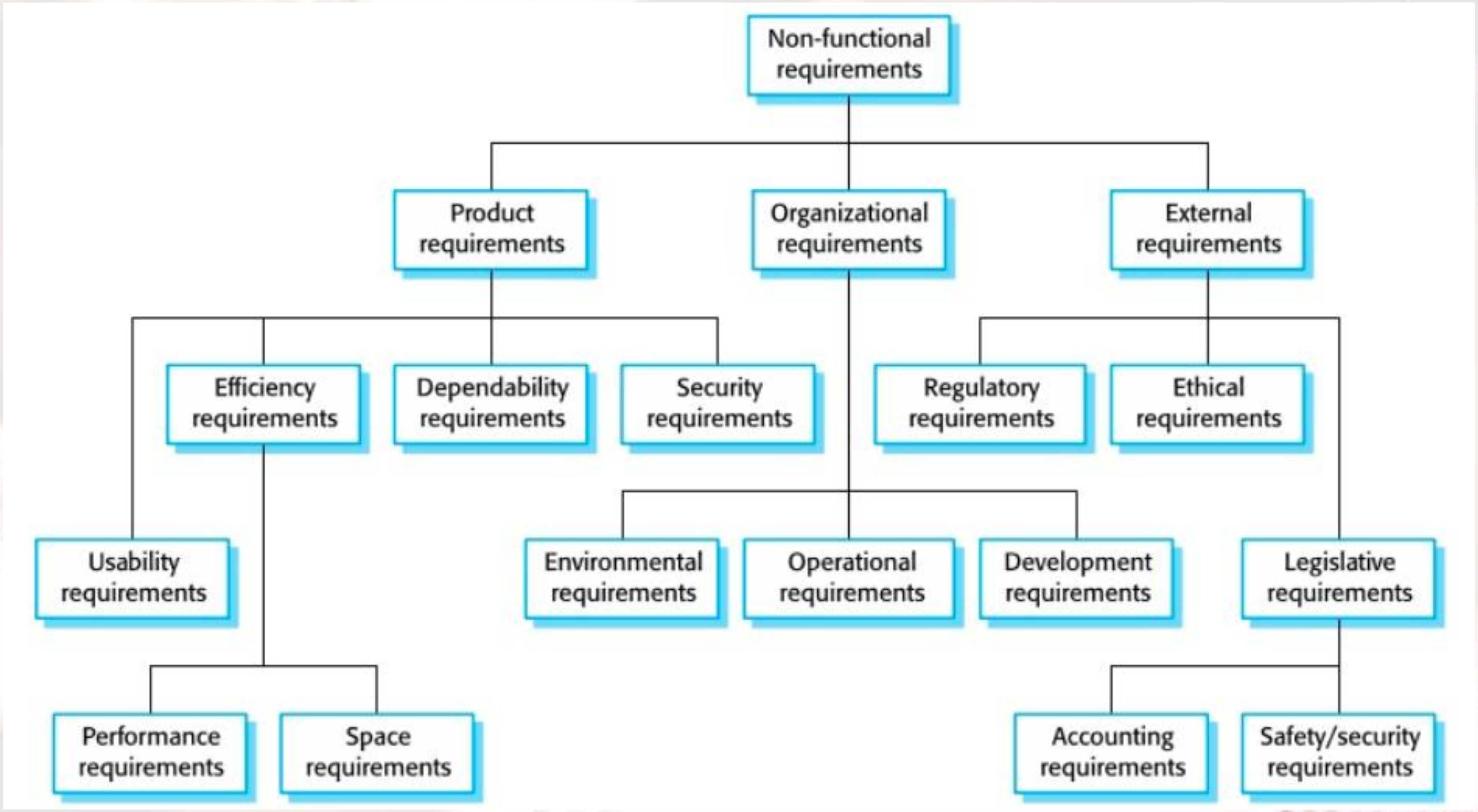
Example: Stakeholders

- List stakeholders for CMU Marathon’s Website.

Functional vs. Non-Functional

- A requirement can be functional or non-functional.
- Functional requirements: FR
  - What is required
- Non-Functional requirements: NFR
  - How well it behave

NFR



- Diagram taken from Sommerville's SE Book

Example: FR vs. NFR

- The system shall provide two payment options: Online bank deposit and Credit card.
- Users of the Mentcare system shall authenticate themselves using their health authority identity card.
- A user shall be able to search the appointments lists for all clinics.
- The system shall implement patient privacy provisions as set out in HStan-03-2006-priv.
- The system shall generate each day, for each clinic, a list of patients who are expected to attend appointments that day.

Example: FR vs. NFR (Contd.)

- A message indicating successful payment must be issued within 5 seconds after submitting the payment.
- Each staff member using the system shall be uniquely identified by his or her 8-digit employee number.
- The payment system must be available 24 hours per day, everyday.
- The Mentcare system shall be available to all clinics during normal working hours (Mon–Fri, 0830–17.30). Downtime within normal working hours shall not exceed five seconds in any one day.

Metrics for Specifying NFR

- Speed
  - Processed transactions/second
  - User/event response time
  - Screen refresh time
- Size
  - Mbytes
  - Number of ROM chips
- Ease of use
  - Training time
  - Number of help frames

Metrics for Specifying NFR

- Reliability
  - Mean time to failure
  - Probability of unavailability
  - Rate of failure occurrence
  - Availability
- Robustness
  - Time to restart after failure
  - Percentage of events causing failure
  - Probability of data corruption on failure
- Portability
  - Percentage of target dependent statements
  - Number of target systems

Other Types of Requirements

- Architecture
- Business
- User
- Function
- Quality of Service
- Implementation

- Referenced from Wikipedia.org, Requirement, retrieved from https://en.wikipedia.org/wiki/Requirement

Characteristics of Requirements

- Unambiguous
- Cohesive
- Complete
- Consistent
- Verifiable (testable)
- Traceable
- Feasible

- Examples can be found from http://www.informit.com/articles/article.aspx?p=1152528&seqNum=4

Requirements and Agile Methods

- Why Incremental requirements engineering?
- Requirements
  - User stories
  - Backlog
- Terminology
  - Theme – Main features
  - Epic – Major components / contain stories
  - Story – Smaller work / task / Building blocks
  - Version – Software release
  - Sprint – Iteration
- Referenced from
  - Atlassian Agile Coach, "Epics, Stories, Versions, and Sprint", retrieved from https://www.atlassian.com/agile/project-management/epics-stories-themes
  - Craft Blog, "How to write epic user stories", retrieved from https://blog.craft.io/2017/04/23/writing-epic-user-stories/

Requirements and Agile Methods

- Why Incremental requirements engineering?
- Requirements
  - User stories
  - Backlog
- Terminology
  - Theme – Main features
  - Epic – Major components / contain stories
  - Story – Smaller work / task / Building blocks
  - Version – Software release
  - Sprint – Iteration
- Referenced from
  - Atlassian Agile Coach, “Epics, Stories, Versions, and Sprint”, retrieved from https://www.atlassian.com/agile/project-management/epics-stories-themes
  - Craft Blog, “How to write epic user stories”, retrieved from https://blog.craft.io/2017/04/23/writing-epic-user-stories/

Example: Agile Requirements

- Theme
  - Buying process
- Epic
  - Checkout
  - Shopping cart
- User story
  - As an online shopper, I want to pay using my PayPal, so I don't have to enter my credit card number.

A User Story

- A requirements written from a goal of end-user.
- Format
  - As a <stakeholder type>
  - I want <feature>
  - So that <some business value / reason>
- Example
  - As a runner,
  - I want to register and pay online.
  - So that I can reserve my place at once without having to go to different places in order to register, pay and complete the registration process.

User Story (contd.)

- Information
  - Primary stakeholder
  - Effect of the story on the stakeholder
  - Business value to the stakeholder derived from the effect
  - Acceptance criteria (for testing)

Version

- Specified by Product owner
- Delivered at the end of each iteration
- Example
  - Version 1: login, logout, profile
  - Version 2: View and print order documents
  - Version 3: Inventory

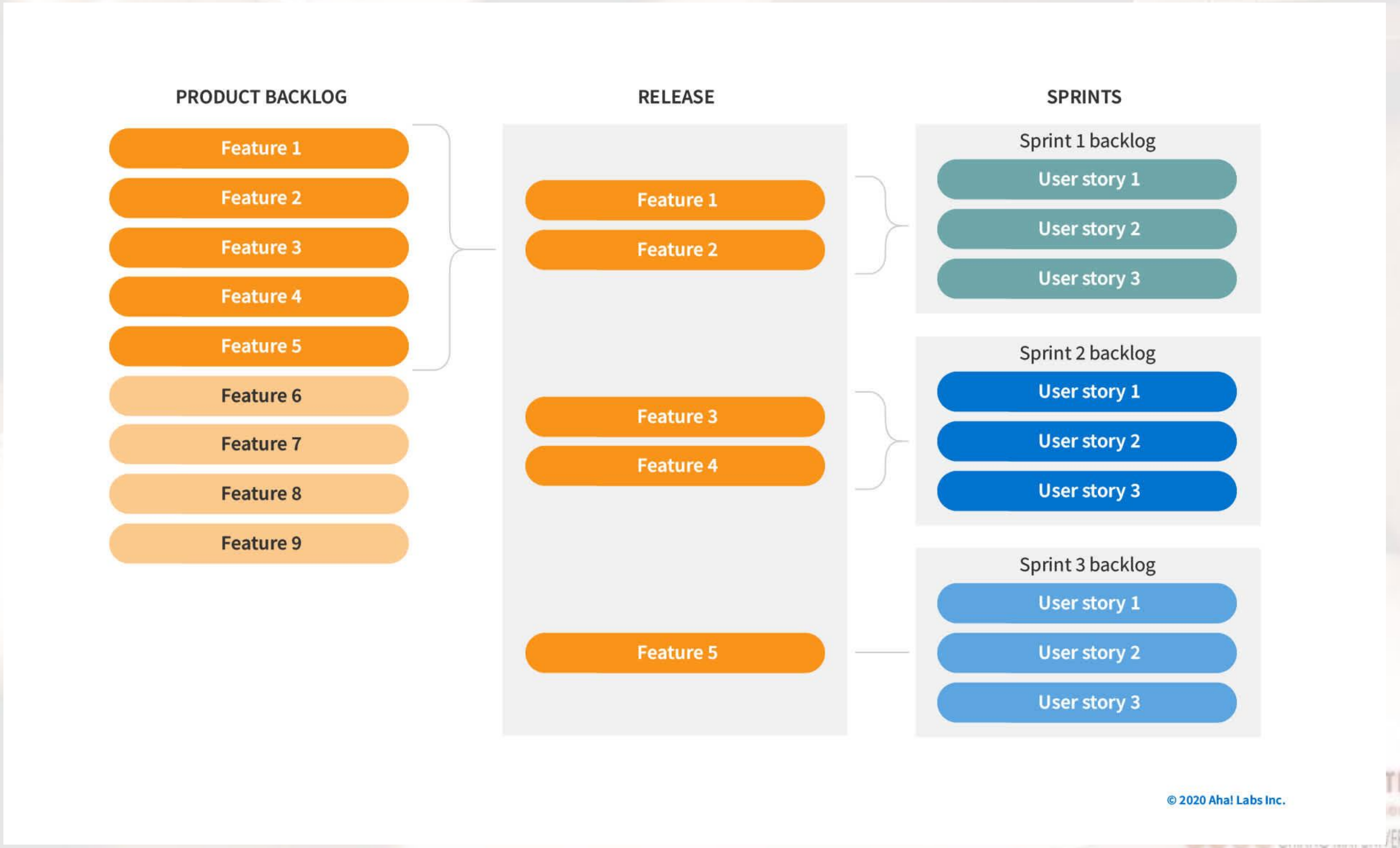
COMPUTER engineering CHIANG MAI UNIVERSITY

Product Backlog

- A wish list
- A scope of final product
- A list of prioritized features that can grow over a series of sprint.
- The product owner gives the team the list at the sprint planning meeting.
- The team expands each product backlog into sprint backlog tasks.

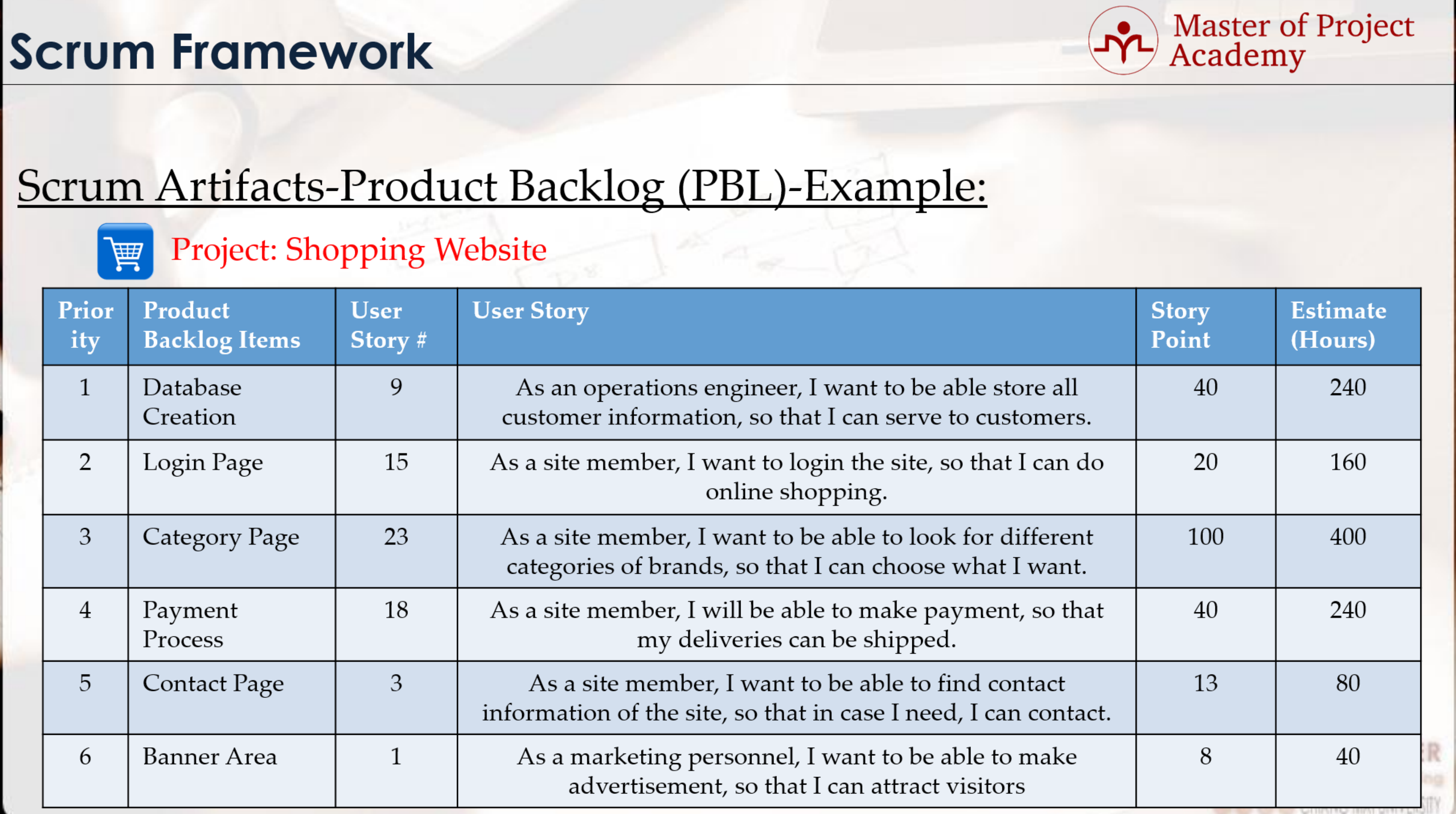
Product Backlog vs. Sprint Backlog

- (Referenced from aha.io)



Product Backlog: Example

- (Referenced from masterofproject.com)



Scrum Framework

Master of Project Academy

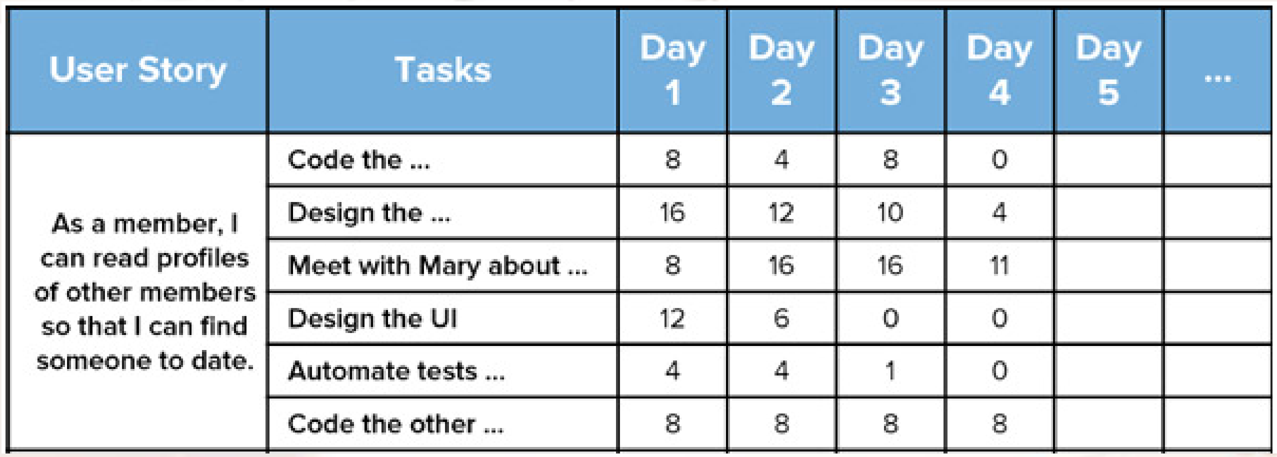
Scrum Artifacts-Product Backlog (PBL)-Example:

Project: Shopping Website

Priority	Product Backlog Items	User Story #	User Story	Story Point	Estimate (Hours)
1	Database Creation	9	As an operations engineer, I want to be able store all customer information, so that I can serve to customers.	40	240
2	Login Page	15	As a site member, I want to login the site, so that I can do online shopping.	20	160
3	Category Page	23	As a site member, I want to be able to look for different categories of brands, so that I can choose what I want.	100	400
4	Payment Process	18	As a site member, I will be able to make payment, so that my deliveries can be shipped.	40	240
5	Contact Page	3	As a site member, I want to be able to find contact information of the site, so that in case I need, I can contact.	13	80
6	Banner Area	1	As a marketing personnel, I want to be able to make advertisement, so that I can attract visitors	8	40

Sprint Backlog

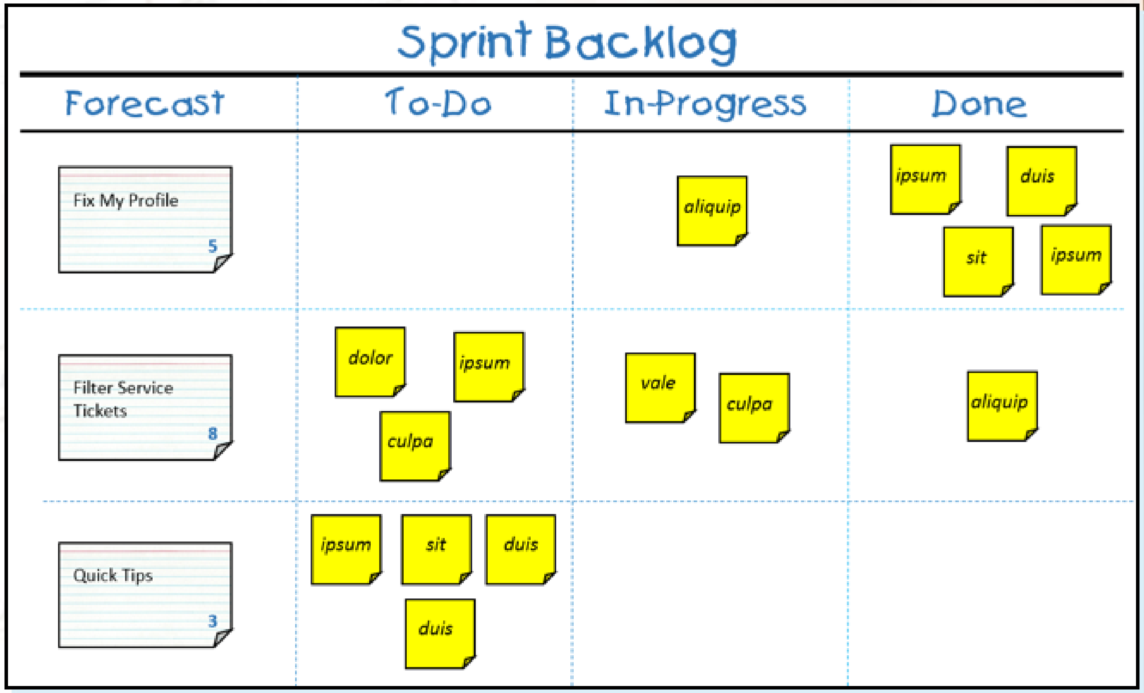
- Sprint backlog
  - List of tasks, commonly maintained as a spreadsheet, necessary to complete each user story along with estimation (how many hours)
  - A number of product backlog items (user stories) is selected foff the top of product backlog during the sprint planning meeting
  - Referenced from Mountain Goat Software, “Sprint Backlog”, retrieved from https://www.mountaingoatsoftware.com/agile/scrum/scrum-tools/sprint-backlog



User Story	Tasks	Day 1	Day 2	Day 3	Day 4	Day 5	...
As a member, I can read profiles of other members so that I can find someone to date.	Code the ...	8	4	8	0		
	Design the ...	16	12	10	4		
	Meet with Mary about ...	8	16	16	11		
	Design the UI	12	6	0	0		
	Automate tests ...	4	4	1	0		
	Code the other ...	8	8	8	8		

Sprint Backlog

- Sprint backlog
  - Created by and for developers.
  - Highly visible, real-time picture of the work during the Sprint in order to achieve the Sprint Goal.
  - Updated through -out the Sprint.
  - Should contain enough detail for progress inspection during daily scrum. (scrum.org)



Sample Product Backlog

Backlog item	Estimate
Allow a guest to make a reservation	3 (story points)
As a guest, I want to cancel a reservation.	5
As a guest, I want to change the dates of a reservation.	3
As a hotel employee, I can run RevPAR reports (revenue-per-available-room)	8
Improve exception handling	8
...	30
...	50

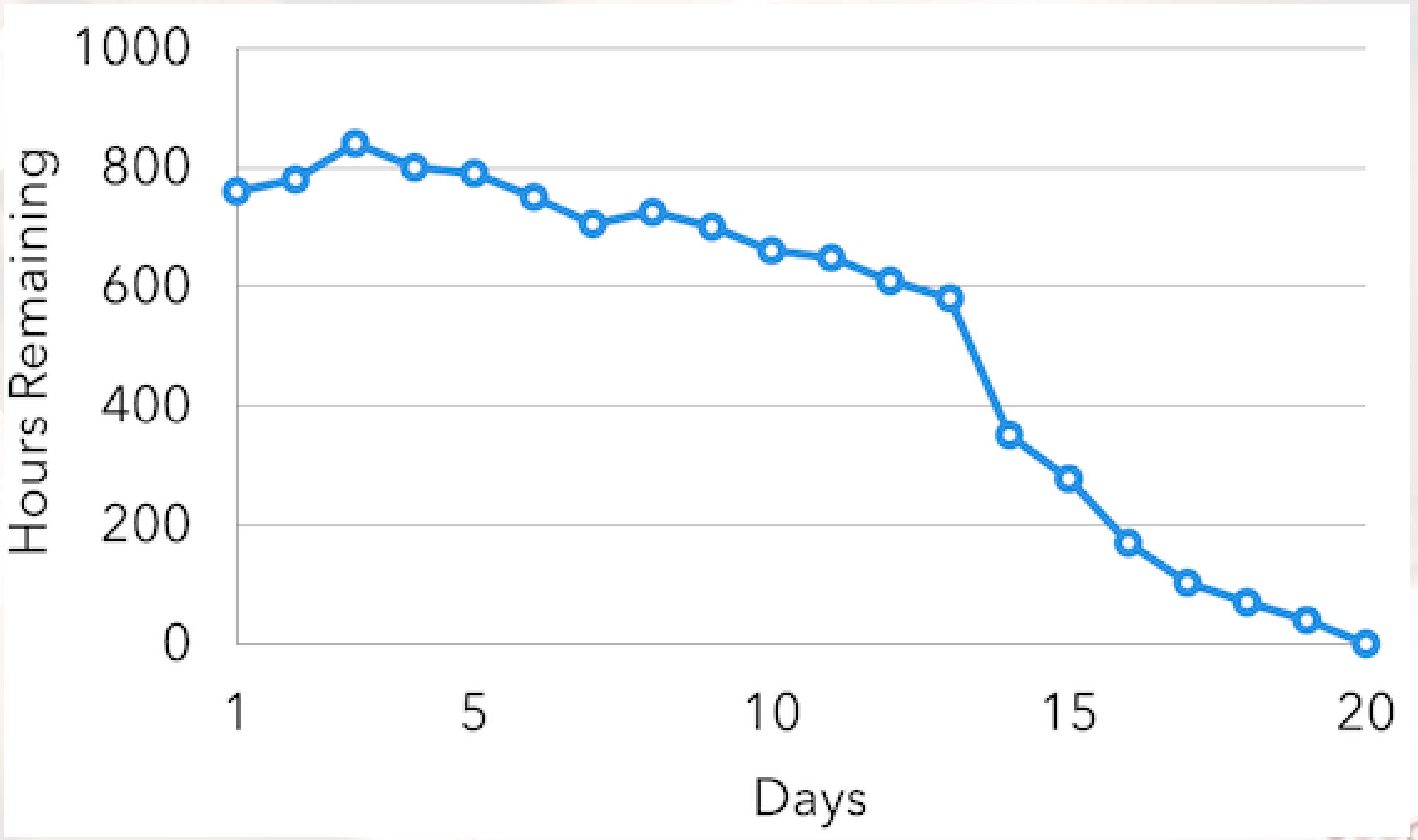
- (Referenced from slidetodoc.com)

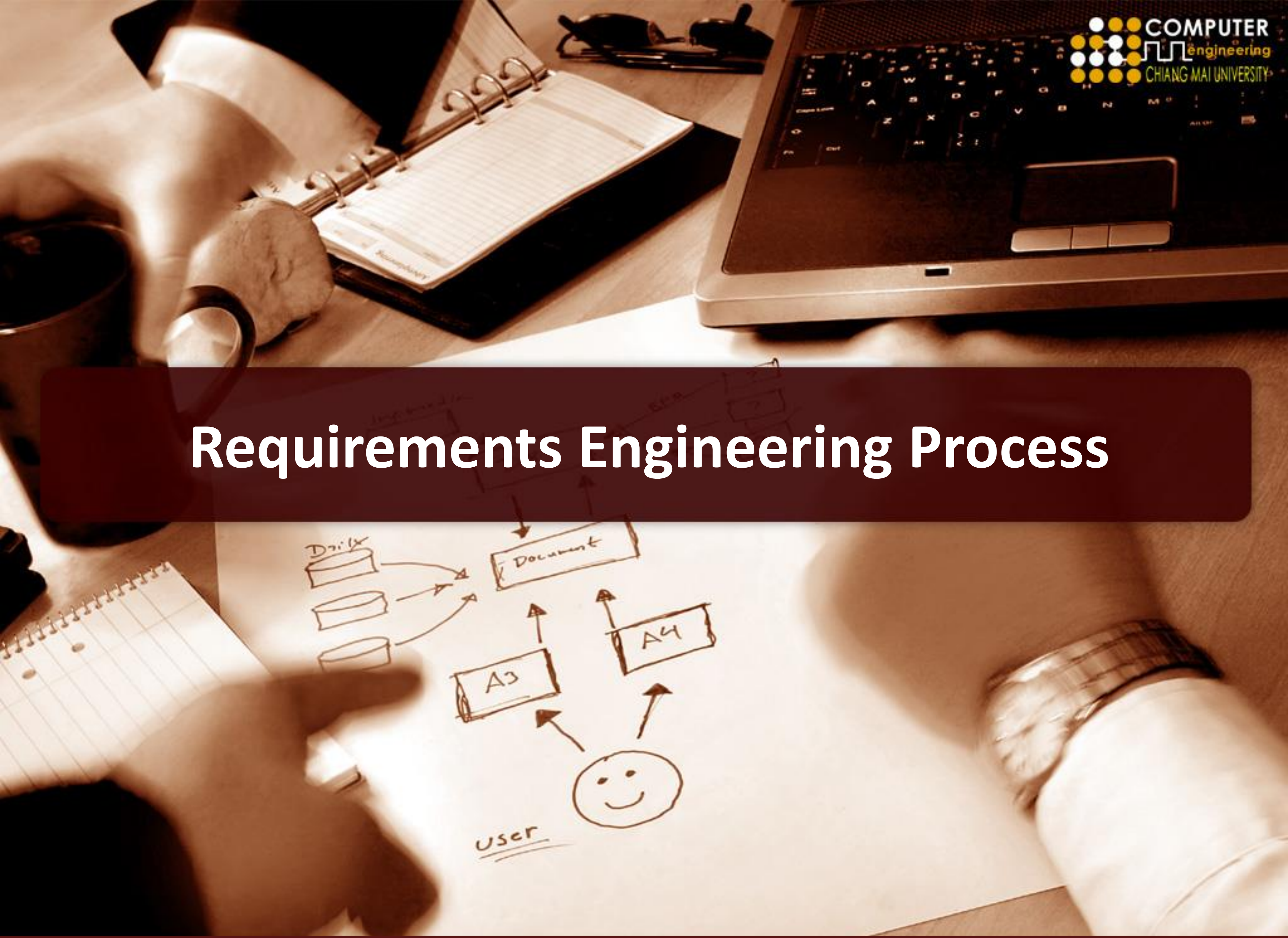
Product Backlog vs. Sprint Backlog

- Product Backlog
  - A list of prioritized and estimated features in form of user stories
- Sprint Backlog
  - A list of tasks decomposed tasks from user stories to be completed during the sprint
  - Time estimate for each task

Burndown Chart

- What can you see from this chart?

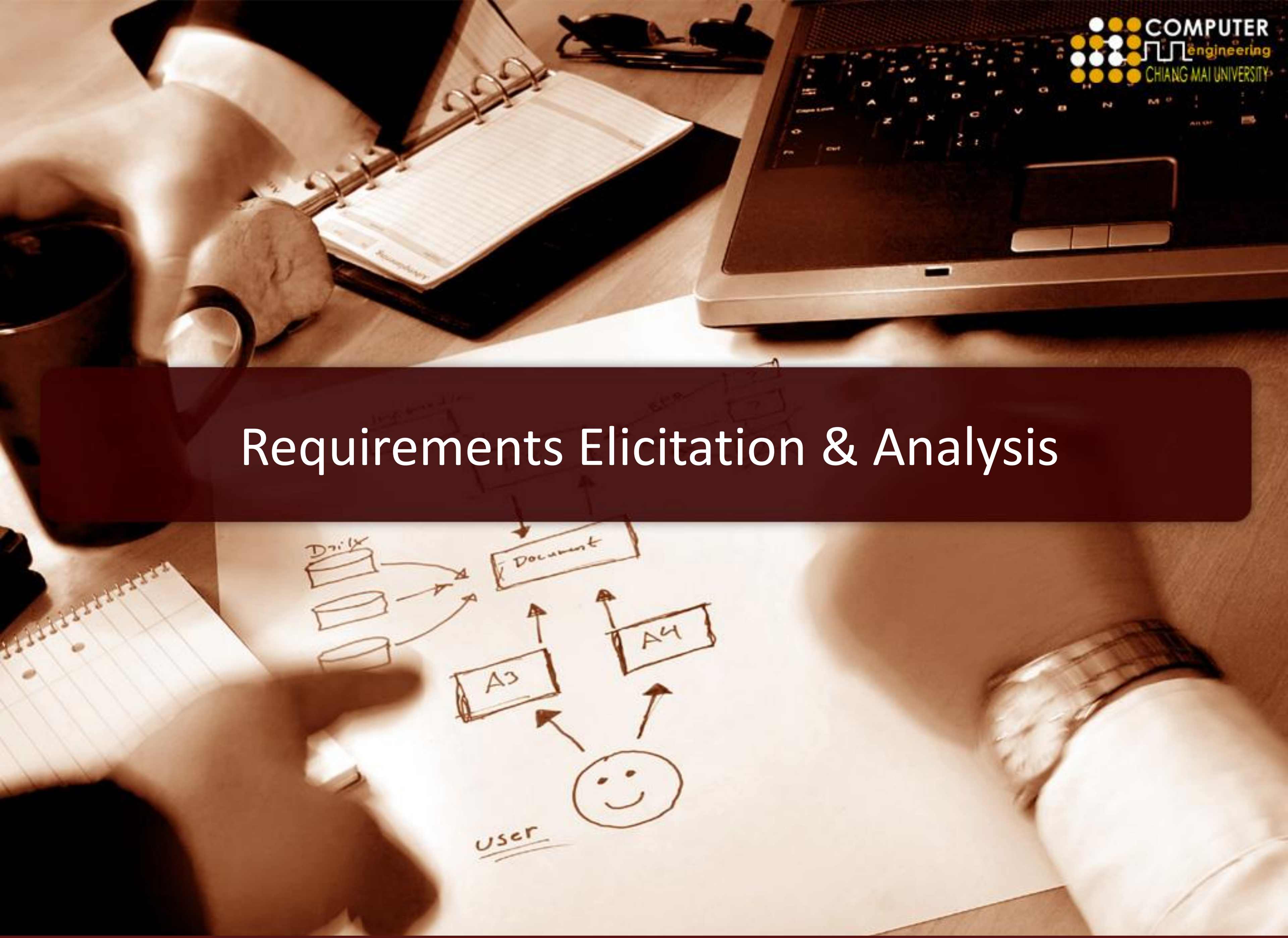




Requirements Engineering Process

Tasks and Activities

- Inception (Feasibility study/ Proposal)
- Elicitation (,Elaboration/refine) and Analysis
- Negotiation
- Specification
- Validation
- Management



Requirements Elicitation & Analysis

Elicitation and Analysis

- Involves requirements gathering and requirements collection
- Staff works with stakeholders (end-users, managers, engineers, etc.) to discover system services, performance, hardware/software constrains, etc.
- Stages include:
  - Discovery
  - Classification and organization
  - Prioritization and negotiation
  - Specification

RE: Practices and Problems

- Wagner and colleges (2018) surveyed Agile projects for RE practices and problems in North America, South America and Europe over 92 organizations. The followings were studied.
  - Elicitation and specification
  - Change and alignment with test
  - Improvement
  - Common problems
- Fernández, Daniel Méndez, et al. "On Evidence-based Risk Management in Requirements Engineering." International Conference on Software Quality. Springer, Cham, 2018.

Survey: Elicitation Techniques and Documentation

- Elicitation
  - Most frequently used techniques
    - Interview
    - Prototyping
    - Facilitated meeting (workshop)
    - Scenarios
    - Observation
  - Documentation
    - Business process model/Data model
    - Structured requirements list
    - Use case model

Survey: Changes in Requirements

- More than 50% update requirements document/backlog
- 15% regularly change the requirements specification
- Although, the majority go with impact analysis between requirements to analyze the effect of changes, and also on the code, one-fifth did not manage changes.
- Requirements are used to define acceptance criteria for testing.

Survey: Common Problems in Agile RE

- Requirements are too abstract and ambiguous
- Unclear/unmeasurable NFR
- Communication within team and between dev and customer
- Changing business processes and or requirements
- Incomplete/hidden requirements
- Inconsistent requirements

Survey: Common Problems in Agile RE

- Missing traceability
- Not enough access to business information
- Weak knowledge of application domain
- Not enough time
- Incompatibility between innovation and need for formal acceptance of requirements
- Technically unfeasible requirements

Elicitation and Analysis Process

- Discovery
- Classification and organization
- Prioritization and negotiation
- Specification

Requirements Discovery

- Gathering information about the system from a range of stakeholders and extract requirements from this information
- Techniques
  - Interview – overall understanding
    - Closed interview
    - Opened interview
    - Mix
  - Ethnography – understand existing process
    - Observation how people work

Requirements Discovery Technique

- Stories and Scenarios
  - Real life examples of how users use a system
  - Usually include:
    - Starting situation
    - Normal flow of events
    - What can go wrong
    - Concurrent activities
    - Finishing state

COMPUTER engineering CHIANG MAI UNIVERSITY

Requirements Specification

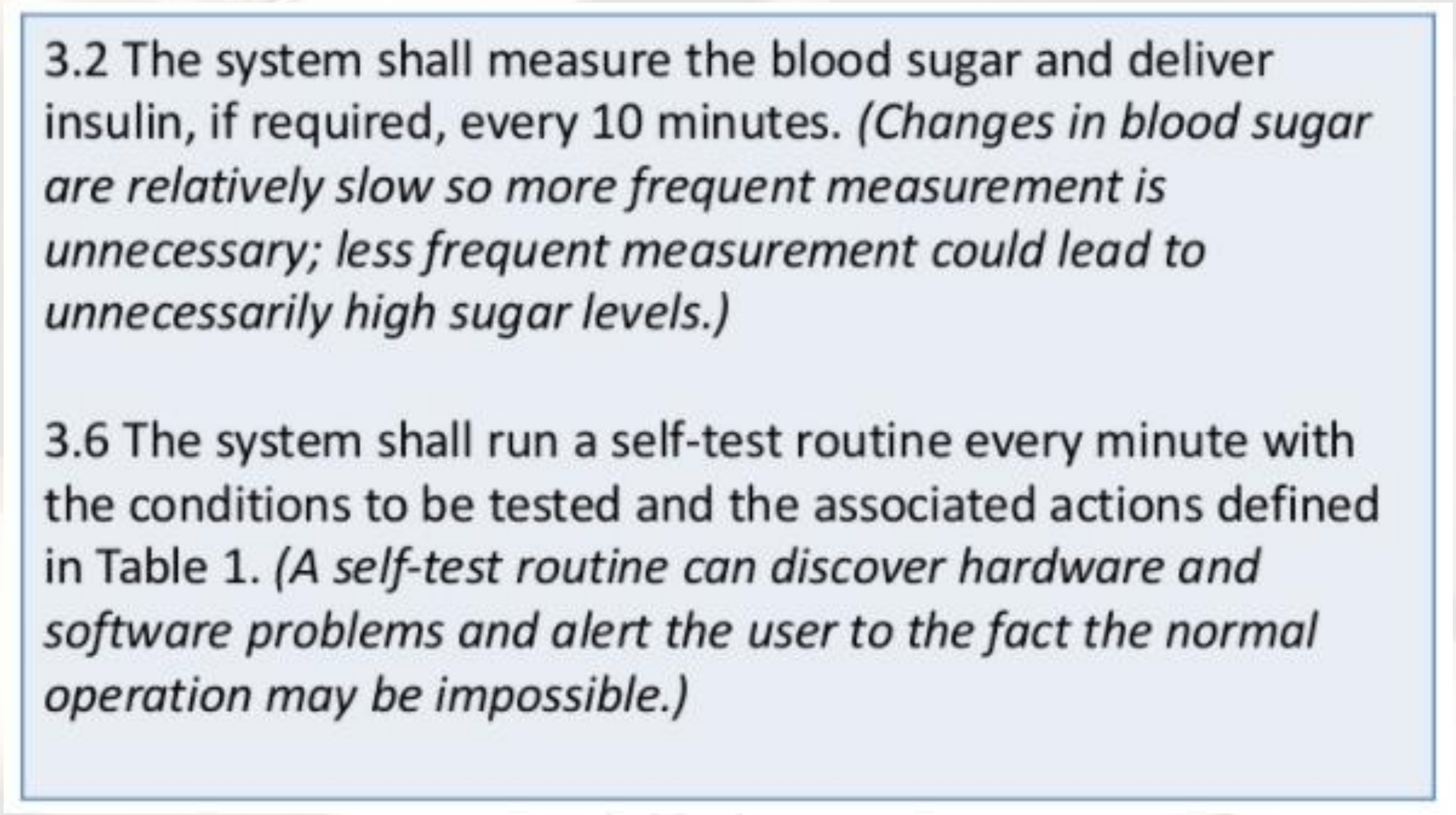
Requirements Specification

- Process of creating a requirements document
- Have to be readable by non-technical background persons
- How to write:
  - Natural language
  - Structured natural language (i.e. Template)
  - Design description languages (i.e. Pseudocode)
  - Graphical notations (i.e. UML)
  - Mathematical specifications (i.e. FSM)

Requirements and Design

- They are inseparable and do overlap
- Requirements → WHAT
- Design → HOW
- Think of the situations where overlapping occurs.

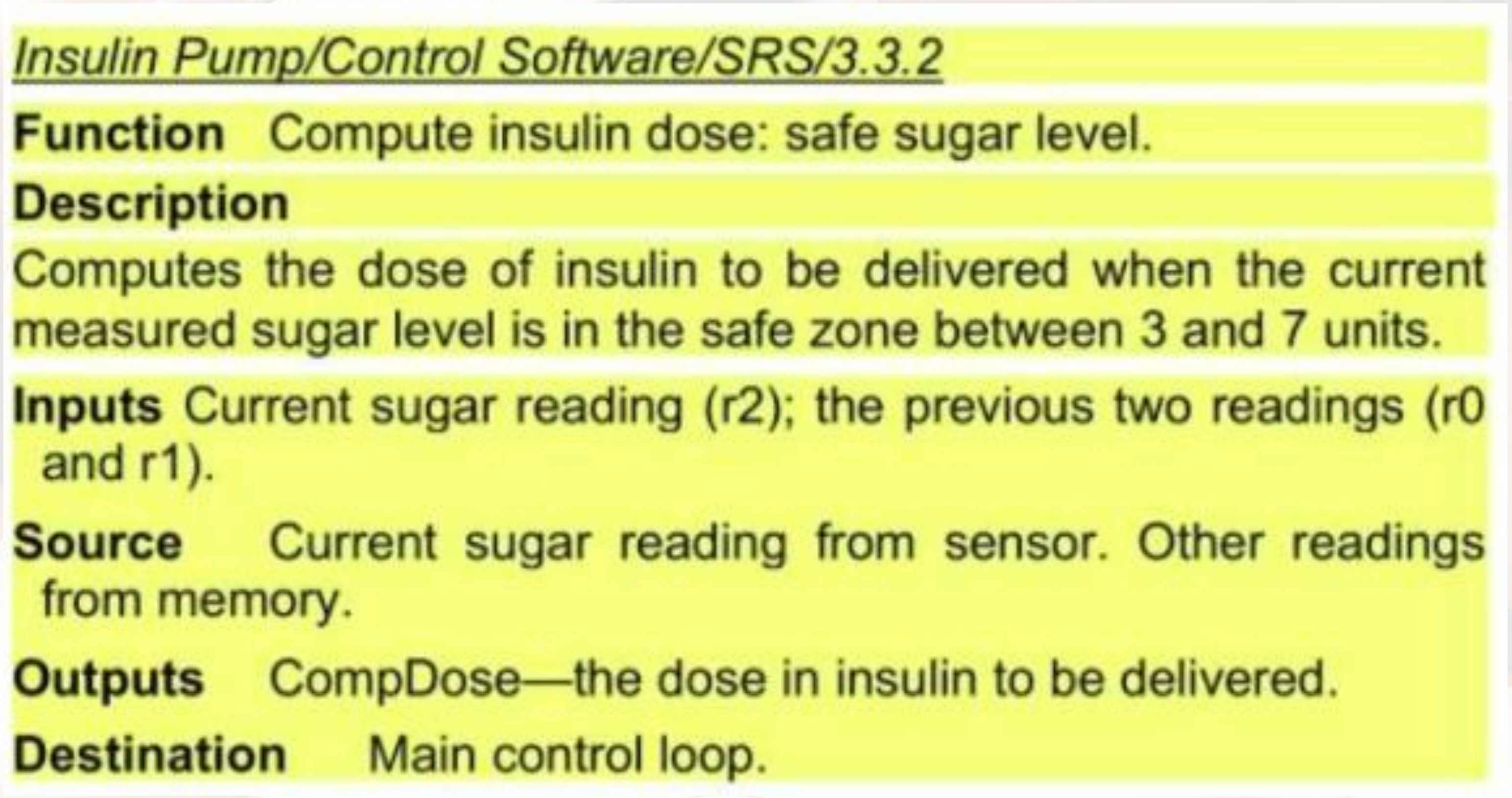
Example: Natural Language


3.2 The system shall measure the blood sugar and deliver insulin, if required, every 10 minutes. *(Changes in blood sugar are relatively slow so more frequent measurement is unnecessary; less frequent measurement could lead to unnecessarily high sugar levels.)*

3.6 The system shall run a self-test routine every minute with the conditions to be tested and the associated actions defined in Table 1. *(A self-test routine can discover hardware and software problems and alert the user to the fact the normal operation may be impossible.)*

- This example is taken from Sommerville's SE Book

Example: Structured NL



Insulin Pump/Control Software/SRS/3.3.2

Function Compute insulin dose: safe sugar level.

Description

Computes the dose of insulin to be delivered when the current measured sugar level is in the safe zone between 3 and 7 units.

Inputs Current sugar reading (r2); the previous two readings (r0 and r1).

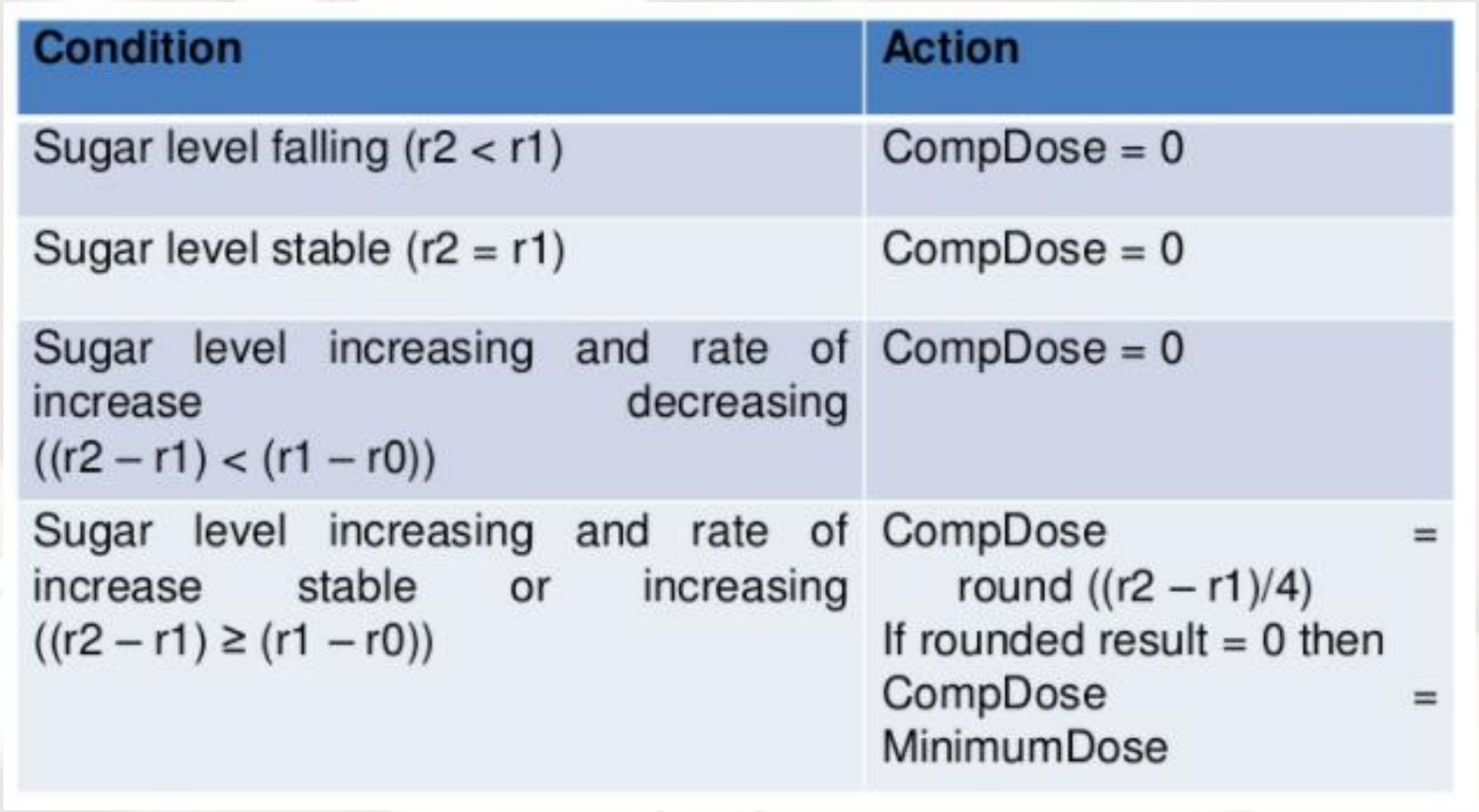
Source Current sugar reading from sensor. Other readings from memory.

Outputs CompDose—the dose in insulin to be delivered.

Destination Main control loop.

- This example is taken from Sommerville's SE Book

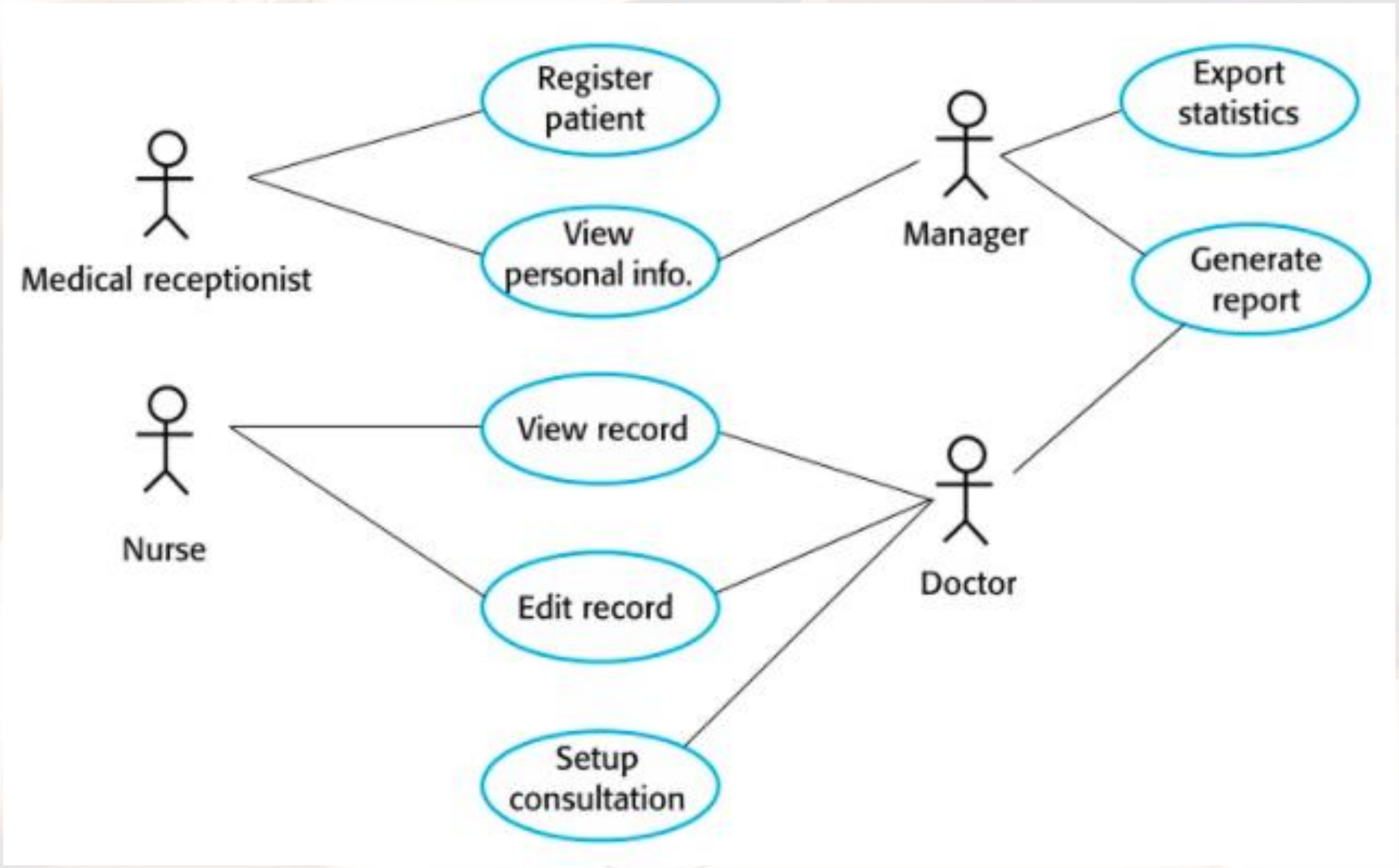
Example: Tabular Specification



Condition	Action
Sugar level falling (r2 < r1)	CompDose = 0
Sugar level stable (r2 = r1)	CompDose = 0
Sugar level increasing and rate of increase decreasing ((r2 – r1) < (r1 – r0))	CompDose = 0
Sugar level increasing and rate of increase stable or increasing ((r2 – r1) ≥ (r1 – r0))	CompDose = round ((r2 – r1)/4) If rounded result = 0 then CompDose = MinimumDose

- This example is taken from Sommerville's SE Book

Example: Use Case



- This example is taken from Sommerville's SE Book

Software Requirements Specification

- An official statement of system services
- Include both user requirements and system requirements
- Intended audiences include:
  - System customers
  - Managers
  - System engineers
  - System test engineers
  - System maintenance engineers

Requirements Specification

- IEEE 830-1998
  - Recommended practices for SRS
- Considerations
  - Functionality
  - External interfaces
  - Performance
  - Attributes
  - Design constraints
- Avoid placing design or project requirements into SRS

SRS Contents

1. Introduction

   1.1 Purpose
   - Purpose and Intended audience of SRS

   1.2 Scope
   - Identify product names (e.g. Report generator)
   - What will product(s) do and not do
   - Describe application, benefits, objectives and goals of software

   1.3 Definitions, acronyms, abbreviation

   1.4 References

   1.5 Overview of SRS

SRS Contents

2. Overall Description
   - 2.1 Product perspective
   - 2.2 Product functions
   - 2.3 User classes and characteristics
   - 2.4 Design/implementation constraints
   - 2.5 Assumptions and dependencies
   - 2.6 Apportioning of requirements

SRS Contents

3. Specific requirements

- Identify requirements that may be delayed to the future version.

SRS Contents

4. System Features

- System feature A
  - Description and priority
  - Action/result
  - Functional requirements
- System feature B

SRS Contents

5. Other Nonfunctional Requirements
   - Performance requirements
   - Safety requirements
   - Security requirements
   - Software quality attributes
   - Project documentation
   - User documentation

SRS Contents

6. Other Requirements
   - Terminology
   - Glossary
   - Definitions list

Requirements Validation

Requirements Validation

- Checking whether the requirements align with what the customers need.
- Cost of fixing a requirement error is a lot higher than fixing an implementation error
- What to check:
  - Validity
  - Consistency
  - Completeness
  - Realism
  - Verifiability

Validation Techniques

- Requirements Reviews
  - Manual analysis
- Prototyping
  - Executable system model
- Test-case generation
  - Developing tests for requirements

Requirements Review

- Regular reviews should be held with client and staff attending the reviews.
- Perform while creating the requirements definition
- Formal reviews
  - Complete documents
- Informal reviews

Review Checks

- Verifiability
  - Testable?
- Comprehensibility
  - Understood?
- Traceability
  - Clearly stated?
- Adaptability
  - Can be changed without a large impact?

Practices in Requirements Review

- Few tools available for validation of requirements
- Done through review meeting such as Sprint review in Scrum at the end of the Sprint
- In Scrum, customers and team together check requirements against organizational standards and knowledge, and user requirements to to find conflicts and errors.

- Mokhtar, Siti Osnita, et al. "Issues and Challenges of Requirements Review in the Industry." Indian Journal of Science and Technology 10.3 (2017).
- Darwish, Nagy Ramadan, and Salwa Megahed. "Requirements Engineering in Scrum Framework." Requirements Engineering 149.8 (2016): 24-29.

Requirements Management

Requirements Management

- A process to manage changes in requirements throughout development
- Record of individual requirements and their changes to study impact of requirements changes.
- A formal process for making change proposals is needed

Requirements Management Activities

- Uniquely identify requirements
- Establish a change management process
- Define traceability policy
- Acquire supporting tools

Managing Requirements Change

- Analyze problem from the change proposal and specify change
- Analyze the effect of the change using traceability information and system knowledge to determine cost
- Modify requirements document, and / or system design and implementation

Activity

- Each group elicits additional requirements from the software prototype used in class today.
- Write the requirement sentences for the system and group them into appropriate categories.
- Write them on Mango.

Summary

Summary

- Requirements define WHAT while designs define HOW
- FR and NFR have to be identified and documented
- Requirements engineering is an iterative process
- Requirements engineering (RE) process consists of different activities: Elicitation and Analysis, Specification, Validation and Management
- A range of techniques such as interview, ethnography, stories and scenarios can be used to elicit discussion and requirements.

End of Lecture Questions

1. What is the difference between FR and NFR?
2. What are characteristics of good requirements?
3. Give an example of a user story for software from your course project.
4. What are the main activities in a Requirements Engineering (RE) process?
5. Why is RE important?