Lecture 03: Software Requirements

EGCI340: SOFTWARE DESIGN (WEEK 03)

Review

1) What kind of the project that is suitable for Prototyping Model?
2) Which model can not be used for the complex or large size of the project?
3) What are the key assumptions of Agile process?

Outline

- Types of Requirements
  - User Requirements
  - System Requirements
- Functional and Non-Functional Requirements
- Requirement and Design
- Requirement Specifications
- Software Requirement Standard and Document Template

What is a requirement?

Requirement:

- A high-level abstract statement of a service or
- A high-level of a system constraint to a detailed mathematical functional specification

Requirements may serve for

- A bid for a contract (open to interpretation)
- The contract itself (must be defined in detail)

Types of requirements

User requirements

- Statements in natural language plus diagrams of the application's services
- Its operational constraints
- Written for customers

System requirements

- A structured document or descriptions of:
  - System's functions, Services, Operational constraints
- Defines what should be implemented
  - Derived from contract between client and contractor

User Requirements

- Describe functional and non-functional requirements
  - Understand by system users who do not have technical knowledge

- User requirements are defined using natural language, tables and diagrams
  - Understand by all users

Problems with Natural Language

Lack of clarity

- Precision is difficult without making the document difficult to read

Requirement's confusion

- Functional and non-functional requirements tend to be mixed-up

Requirement's integration

- Several different requirements may be expressed together

System Requirements

- More detailed specifications of system functions, services and constraints than user requirements
- Use for designing the system Illustrating the system models
- Incorporated into the system contract

Definitions and Specifications [1]

User Requirement Definition

1. LIBSYS shall keep track of all data required by copyright licensing agencies in the UK and elsewhere

System Requirements Specification

1. On making a request for a document from LIBSYS, the requestor shall be presented with a form that records details of the user and the request made.
2. LIBSYS request forms shall be stored on the system for five years from the date of request.
3. All LIBSYS request forms must b indexed by user, by the name of the material requested and by the supplier of the request.
4. LIBSYS shall maintain a log of all requests that have been made to the system.
5. For material where authors lending rights apply, loan details shall be sent monthly to copyright licensing agencies that have registered with LIBSYS

Functional and Non-Functional Requirements

Functional requirements

- Statements of services the system should provide
- How the system should react to particular inputs and,
- How the system should behave in particular situations

Non-functional requirements

- Constraints on the services or functions offered by the system
  - Such as timing constraints
- Constraints on the development process, standards, etc.

Functional Requirements

- Describe functionality or system services
- Depend on the type of softwares, expected users and the type of systems where the software is used
- High-level statements of what the system should do
- Describe the system services in detail

Examples of Functional Requirements

- The user shall be able to search
- The system shall provide appropriate viewers
- Every order shall be allocated a unique identifier (ORDER_ID)
- The user shall be able to copy to the account's permanent storage area

Requirements Imprecision (not clear)

- Problems arise when requirements are not precisely stated
- Ambiguous requirements may be interpreted in different ways by developers and users

- For example: consider the term ‘appropriate viewers’
  - User intention
    - Special purpose viewer for each document type
  - Developer interpretation
    - Provide a text viewer that shows the contents of the document

Requirements Completeness and Consistency

Complete

- They should include descriptions of all facilities required

Consistent

- There should be no conflicts or contradictions in the descriptions of the system facilities

In practice, it is impossible to produce a complete and consistent requirements document

Non-Functional Requirements

System properties and constraints

- e.g. reliability, response time and storage requirements

Constraints are:

- I/O device capability
- System representations and, etc.

Non-Functional Classifications

Product requirements

- Requirements which specify that the delivered product must behave in a particular way
- e.g. execution speed, reliability, etc.

Organizational requirements

- Requirements which are a consequence of organizational policies and procedures
- e.g. process standards used, implementation requirements, etc.

External requirements

- Requirements which arise from factors which are external to the system and its development process
- e.g. interoperability requirements, legislative requirements, etc.

Example: Non-Functional Requirements

- Product requirement
  - The user interface for LIBSYS shall be implemented as simple HTML without frames or Java applets

- External requirement
  - The system shall not disclose any personal information about customers apart from their name and reference number to the operators of the system

Goals and Requirements

Non-functional requirements may be very difficult to state precisely and imprecise requirements may be difficult to verify

Goal

- A general intention of the user such as ease of use

Verifiable non-functional requirement

- A statement using some measure that can be objectively tested

Goals are helpful to developers as they convey the intentions of the system users

Example

System goal

- The system should be easy to use by experienced controllers
- The system should be organized in such a way that user errors are minimized

A verifiable non-functional requirement

- Experienced controllers shall be able to use all the system functions after a total of two hours training.
- After this training, the average number of errors made by experienced users shall not exceed two per day

Requirements Measurement[1]

Property	Measure
Speed	Processed transactions/second User/Event response time Screen refresh time
Size	K bytes Number of RAM chips
Ease of use	Training time Number of help frames
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

Requirements Interaction

Conflicts between different non-functional requirements are common in complex systems

For example: a spacecraft system

- To minimize weight, the number of separate chips in the system should be minimized
- To minimize power consumption, lower power chips should be used
- However, using low power chips may mean that more chips have to be used

Which one is the most critical requirement?

Requirements and Design

In principle:

- Requirements should state *what the system should do*
- The design should describe *how to do this*

Guidelines for Writing Requirements

IEEE standard format

- Use language in a consistent way
  - “shall” for mandatory requirements, “should” for desirable requirements
- Use text highlighting to identify key parts of the requirement
- Avoid the use of computer jargon

Problems with Natural Language (NL) Specification

Ambiguity

- Readers and writers of the requirement must interpret the same words in the same way
- NL is naturally ambiguous

Over-flexibility

- The same thing can interpreted in the different ways in the specification

Types of Writing Specification [1]

Notation	Description
Structured natural language	This approach depends on defining standard forms or templates to express the requirements specification.
Design description languages	This approach uses a language like a programming language but with more abstract features to specify the requirements by defining an operational model of the system. This approach is not now widely used although it can be useful for interface specifications.
Graphical notations	A graphical language, supplemented by text annotations is used to define the functional requirements for the system. An early example of such a graphical language was SADT (Ross, 1977) (Schoman and Ross, 1977). Now, use-case descriptions (Jacobsen, et al., 1993) and sequence diagrams are commonly used (Stevens and Pooley, 1999).
Mathematical specifications	These are notations based on mathematical concepts such as finite-state machines or sets. These unambiguous specifications reduce the arguments between customer and contractor about system functionality. However, most customers don't understand formal specifications and are reluctant to accept it as a system contract.

Structured Language Specifications

- Limited by a *predefined template* for requirements
- All requirements are written in a *standard* way
- The terminology used in the description may be limited
- The advantage :
  - The most of the expressiveness of natural language is maintained

Form-Based Specifications

- Definition of the function or entity
- Description of inputs and where they come from
- Description of outputs and where they go to
- Indication of other entities required
- Pre and post conditions (if appropriate)
- The side effects (if any) of the function

Form-Based Specifications (Cont.) [1]
(Structured Natural Language)

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

Action: CompDose is zero if the sugar level is stable or falling or if the level is increasing but the rate of increase is decreasing. If the level is increasing and the rate of increase is increasing, then CompDose is computed by dividing the difference between the current sugar level and the previous level by 4 and rounding the result. If the result, is rounded to zero then CompDose is set to the minimum dose that can be delivered.

Requires	Two previous readings so that the rate of change of sugar level can be computed.
Pre-condition	The insulin reservoir contains at least the maximum allowed single dose of insulin.
Post-condition	r0 is replaced by r1 then r1 is replaced by r2
Side effects	None

Tabular Specification [1] (Design Description Langauge)

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

Graphical Models

Graphical models are useful

- When you need to show how state changes
- Where you need to describe a sequence of actions

Example: Graphical Model -- Sequence Diagrams

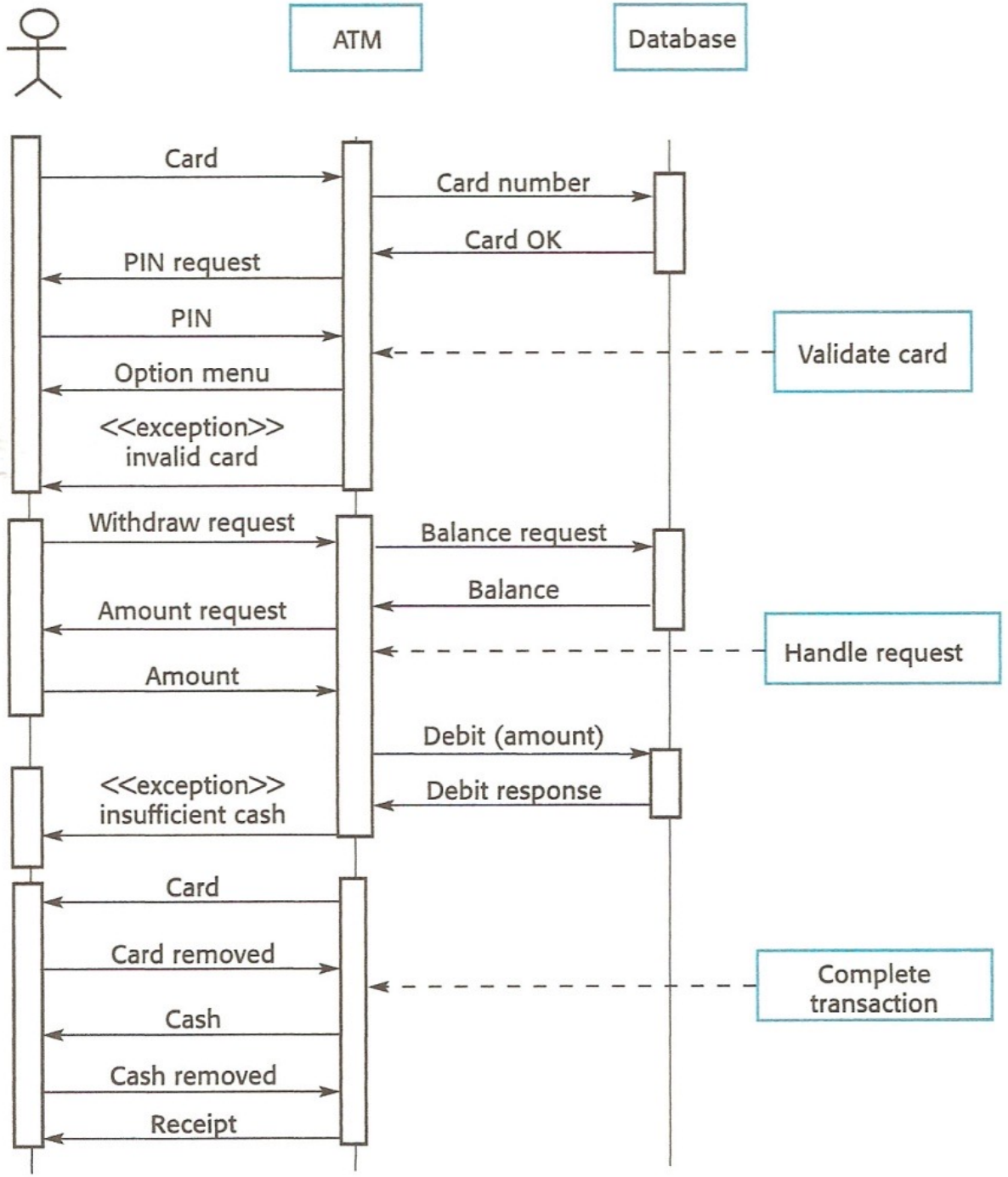
These show the sequence of events that take place during some user interaction with a system

You read them from top to bottom to see the order of the actions that take place

Cash withdrawal from an ATM

- Validate card
- Handle request
- Complete transaction

Sequence Diagram of ATM Withdrawal



The Requirement's Document

The requirements document is the official statement of what is required of the system developers

Include :

- A definition of user requirements
- A specification of the system requirements

It is NOT a design document.

- Describe about *WHAT* the system should do rather than *HOW* it should do it

IEEE Requirement Standard

Defines a generic structure for a requirement’s document

- Introduction
- General description
- Specific requirements
- Appendices
- Index

Requirements Document Structure

1) Preface
2) Introduction
3) Glossary
4) General description
5) System architecture
6) User requirement specification
7) System requirement specification
8) System models
9) Appendices
10) Index

Reference

1. Ian Sommerville, Software Engineering 10th Edition, Pearson, April 2015

Wait! It’s not Done Yet

Any Questions?

:O)

Thank you

Assignment 1 (Part1) 30%

For odd student id, Create user and system requirements for AirVisual

For even student id, Create user and system requirements for Grab (Food)

Your SRS must contain at least

- 1 user requirements
- 2 functional requirements according to your use requirements
- 1 non-functional requirement according to your use requirements

Your Requirement must follow Ex-User-Requirements_student.docx

Assignment 1 (cont.)

- Try the software and find more information on the internet about your assigned software/system
- Write user and system requirement that can serve the software/system
- See Ex-User-Requirements.pdf as your example