**Software Requirements Analysis**

“What does the customer want?”

**Lecture Objectives**

* To understand the importance of specifying requirements correctly
* To discuss the different representations of requirements in the analysis model
* To illustrate the modeling of data elements in the problem domain

**What Is Served?**

- Nasi Lemak
- Sambal
- Cucumber
- Peanuts?
- Egg?
- Ikan Bilis?
- Chicken?
- Sotong?

**Expectations**

- Standard: Nasi Lemak + Sambal + Cucumber + Peanuts + Ikan Bilis
- Great: All above + Egg + Chicken/Sotong
- Disappointing: Nasi Lemak + Sambal + Cucumber

**During Breakfast One Day...**

- Nasi Lemak!

**What We Are Doing**

- ABC Software!
The First Task

Analysis → Problem
Design → Models
Development → Solution
Testing

Analysis Principles

1. Information domain of a problem must be represented and understood
2. Required functions must be defined
3. Behaviour of software must be represented
4. Models that depict information, function, and behaviour must be partitioned in a layered or hierarchical fashion
5. Analysis process should move from essential info to implementation detail

Software Requirements Analysis

1. Developer and customer communicate to form the details of the software
2. Developer - interrogator, problem solver
3. Customer - required functions & performance
4. Problems:
   - Misinterpretation
   - Misinformation
   - Ambiguity

More guidelines

1. Understand the problem before creating the analysis model
2. Develop prototype
3. Record origin and reason of requirements
4. Use multiple views
5. Prioritize requirements
6. Eliminate ambiguity
7. Misinterpretation

Requirements Analysis Tasks

1. Problem Recognition: as perceived by the customer.
2. Evaluation & Synthesis: the analyst define data objects, evaluate the flow of information, define all software functions, understand system behavior, establish system interface characteristics and design constraints.
3. Modeling: models of data, information and control flow, and operational behaviors.
4. Specification: a model of the software is created and evaluated by both software engineers and customers.
5. Review: of software requirements specifications done by developer and customer.

The Analysis Model

- Data-Object Description
- Data Flow Diagram
- Entity-Relationship Diagram
- Control Specification (CSR/PIC)
- State-Transition Diagram
**Analysis Model Elements**

**Data Dictionary**
- Contains descriptions of all data objects used

**Entity-Relationship Diagram (ERD)**
- Describes relationships between data objects

**Data Flow Diagram (DFD)**
- Describes data flow & transformation

**State Transition Diagram (STD)**
- Describes system behaviour

---

**Entity Examples**

**External entity** – (anything that produces or consumes information)
- Thing (e.g. report or display)
- Occurrence or events (e.g. phone call)
- Role (e.g. salesperson)
- Organizational unit (e.g. accounting department)
- Place (e.g. warehouse)
- Structure (e.g. File)

---

**Data Modeling**

**Identification of ‘what’ elements from the problem domain that is relevant**
- Graphically represented by Entity-Relationship Diagram, which contains:
  - Entities
  - Attributes
  - Relationships

---

**Attributes**

**Properties of the entity**
- Name of an instance
- Description of an instance
- Reference to another entity

- One or more attributes must be defined as identifier - “key” to find an instance of the entity. (e.g. ID number of a student).
- Set of attributes may differ in different analysis

---

**Entities**

- Representation of items from the problem domain that is applicable to the system
- Has a set of attributes to describe it
- Drawn as a labeled rectangle in ERD

- **Customer**
- **Member**

---

**Relationships**

- Associations between instances of one or more entity types that is of interest
- Usually means that event has occurred, or there exists some natural linkage between entity instances

- Drawn as line between entities, labeled with verb phrases

- **Employee** is assigned to **Parking**
- **Place**

---
**Cardinality**

- Specification of the number of occurrences of one object that can be related to the number of occurrences of another object.
- Usually expressed as 'one' or 'many'.
- Possible relationships:
  - One-to-One
  - One-to-Many
  - Many-to-Many

**Many-to-Many Relationship**

```
Parent
+-----------------+------------------+
| is parent of    | is child of      |
|-----------------+------------------|
```

"Every parent has one or more children, and every child has one or more parent".

**One-to-One Relationship**

```
Member
+-----------------+------------------+
| is married to   | is spouse of     |
|-----------------+------------------|
```

"Every member has a spouse".

**Modality**

- Specifies whether the relationship is optional or mandatory.
- Modality is 0 if relationship is optional.
- Represented by dotted line in ERD.
- Modality is 1 if relationship is mandatory.
- Represented by straight line in ERD.

**One-to-Many Relationship**

```
Member
+-----------------+------------------+
| is married to   | is spouse of     |
|-----------------+------------------|
```

"Every member has one or more spouses".

**Optional relationships**

```
Member
+-----------------+------------------+
| is married to   | is spouse of     |
|-----------------+------------------|
```

"Every member may have a spouse".

```
Member
+-----------------+------------------+
| is married to   | is spouse of     |
|-----------------+------------------|
```

"Every member may have one or more spouses".
Example of Entity-Relationship Diagram

Customer places Order

placed by

Order contains

Product describes Order

is a

Order Item

References


“Software Engineering” by Ian Sommerville, Addison-Wesley, 2001