RE Framework
Table of Contents

• RE framework
  – System context
  – Core activities
  – Requirements artefacts
  – Validation
  – Management

Table of Contents

• RE framework
  – **System context**
  – Core activities
  – Requirements artefacts
  – Validation
  – Management

System context

- Subject facet
- Usage facet
- IT system facet
- Development facet

Core activities
- Documentation
- Elicitation
- Negotiation

Requirements artefacts
- Goals
- Scenarios
- Solution oriented requirements

Validation

Management
System Context

“the part of the system environment relevant for defining, understanding, and interpreting the system requirements of a system to be developed”
System Context

- system boundary:
  - Which aspects pertain to the system to be developed and which aspects belong in the system context?

- context boundary:
  - Which aspects pertain to the system context (i.e., have a relation to the system to be developed) and which aspects are part of the irrelevant environment?
System Context

Defining system boundaries through interfaces:

- human-machine interface
- hardware interface
- software interface
System Context

Gray zone between system and system context:

- describes vaguely defined system boundaries during RE process
- refers to the system scope change during RE
System Context

- **Subject facet**: objects and events that are relevant for the system,
  - because the system must store or process information about these objects

- **Usage facet**: aspects concerning the usage of the system by people and
  - other systems

- **IT system facet**: aspects concerning the operational or technical environment in which the system is deployed

- **Development facet**: aspects that influence the development of the system
  - imposed by law, or by client and relate to the development process
But design changes the world...
But design changes the world...
Table of Contents

• RE framework
  - System context
  - Core activities
  - Requirements artefacts
  - Validation
  - Management

• performed iteratively / in parallel
Document important information elicited or developed when performing a core the RE activity
• *i.e.*, documentation, elicitation, negotiation, validation and/or management
Achieve progress in the content dimension by eliciting new requirements as well as detailed information about existing requirements

- Elicit (extract) all requirements at the level of detail for the system to be developed
Core activities

Documentation \(\leftrightarrow\) Elicitation

Negotiation

Achieve **agreement among all stakeholders** about the requirements

- has to deal with conflicts about requirements
Is there a “Requirements Lifecycle”

Source: Adapted from Pohl, CAISE 1993
Table of Contents

• RE framework
  – System context
  – Core activities
  – Requirements artefacts
  – Validation
  – Management

**Core activities**

- Documentation
- Elicitation
- Negotiation

**Requirements artefacts**

- Goals
- Scenarios

**Solution oriented requirements**

**System context**

- Subject facet
- Usage facet
- IT system facet
- Development facet

**Validation**

**Management**
Intention with regard to objectives, properties, or use of the system

Requirements artefacts

- Goals
- Scenarios
- Solution oriented requirements
Document sequences of interactions in which the system satisfies some goals or fails to satisfy them

Requirements artefacts

Goals

Scenarios

Solution oriented requirements
Specify requirements at the required level of detail, the desired properties and features of the system to be developed.
Requirements artefacts

Context

Design

Use Case Models
System Sequence Diagrams
Interaction Diagrams
Domain Models (simplified Class Diagrams)
Class Diagrams

System Description

Behavior

Structure

Program Code

import javax.crypto.
import java.io.
import java.util.

public class Home
    implements
protected Co
protected In
protected St
public stati

public HomeA
    KeyStore

    }
    try {
        input
    } catch

    LockCtrl
    LightCtr
    PhotoObs
Table of Contents

• **RE framework**
  – System context
  – Core activities
  – Requirements artefacts
  – **Validation**
  – Management

Core activities

Requirements artefacts

Solution oriented requirements

Goals

Scenarios

Validation

Validation techniques:
- Consideration of system context
- Execution of RE activities
- Created requirements artefacts
- Inspection
- Reviews
- Walkthroughs
- Perspective-based reading
- Prototyping

Management
Table of Contents

• RE framework
  – System context
  – Core activities
  – Requirements artefacts
  – Validation
  – Management

• Establishing requirements traceability
• Prioritising requirements
• Managing changes of requirements artefacts
Any questions
How does RE framework fit to the software development lifecycles (SDLCs)?
Lifecycle of Engineering Project

• Lifecycle models
  - Useful for comparing projects in general terms
  - Not enough detail for project planning

• Examples:
  - Sequential models: Waterfall, V model
  - Rapid Prototyping
  - Phased Models: Incremental, Evolutionary
  - Iterative Models: Spiral
  - Agile Models: eXtreme Programming, Scrum, Kanban
Waterfall Model

- View of development:
  - A sequential process of stepwise refinement
  - Largely a high level management view
  - All requirements are refined up front
  - All the steps are done once in the strict order
Waterfall Model

• Pros:
  - Easy to understand
  - Easy to monitor timewise
  - The outcome is crystal clear

• Cons:
  - Static view of requirements - ignores volatility
  - Lack of user involvement once specification is written
  - Unrealistic separation of specification from design
  - Doesn’t accommodate prototyping, reuse, etc.
  - Long delivery time
The V-Model is a software development process model that represents the phases of software development as a V-shaped diagram. It consists of the following phases:

- **Requirements Phase**:
  - System Requirements
  - Software Requirements

- **Design Phase**:
  - Preliminary Design
  - Detailed Design

- **Implementation Phase**:
  - Code and Debug
  - Component Test

- **Integration Phase**:
  - Software Integration
  - System Integration

- **Testing Phase**:
  - Unit Test
  - Component Test
  - Acceptance Test

The model emphasizes the iterative and concurrent nature of software development, where requirements and design are analyzed and designed at the same time, followed by implementation and integration phases. The model is often used to ensure that requirements are properly translated into designs, which are then implemented in a way that satisfies the requirements.
V-Model

Pros:
- Simple and easy to use
- Each phase has specific deliverables
- Works well for small projects where requirements are easily understood

Cons:
- Little flexibility and adjusting scope is difficult and expensive
- No clear path for problems found during testing phases
- Doesn’t accommodate prototyping, reuse, etc.
- Long delivery time
Agile Models

- Develops software iteratively
- Delivers multiple ‘software increments’
- Reduce communication barriers
  - Programmer interacts with customer
- Reduce document-heavy approach
  - Documentation is expensive and of limited use
- Have faith in the people
  - Don’t need fancy process models to tell them what to do!
- Is driven by customer descriptions of what is required (scenarios)
  - Rather than focusing on the contract
- Follow the manifesto of 12 principles

Source: Adapted from Nawrocki et al, RE’02

Agile Models

Pros:
- Recognizes that plans are short-lived
- Adapts as changes occur
- Improved quality by finding and fixing defects quickly and identifying expectation mismatches early

Cons:
- Code can be hard to maintain
  - Focuses on working with software and lacks documentation efficiency
- Relies on oral communication
  - Mis-interpretation possible
- Assumes single customer representative
  - Multiple viewpoints not possible
- Only short term planning
  - No longer term vision
Agile Models

Agile methodology includes

● Agile Scrum
● Extreme Programming (XP)
● Lean Software Development
● Kanban
● Feature-driven development (FDD)
● Rapid application development (RAD)
● Scumban
● Dynamic systems development method (DSDM)
Scrum

- Short sprints with small deliverables
- Specific roles assignment
  - product owner, scrum master, and development/scrum team
Extreme Programming

https://www.geeksforgeeks.org/difference-between-scrum-and-xp/
Extreme Programming

- Instead of a requirements spec, use:
  - User story cards
  - On-site customer representative
- Pair Programming
- Small releases
  - E.g. every two or three weeks
- Planning game
  - Select and estimate user story cards at the beginning of each release
- Emphasise strong engineering practices
- Pair programming
- Test-driven development (TDD)
- The program code is the design doc
Where is RE in these lifecycles?
Usage of SDLCs

- Agile/Scrum: 33%
- Kanban: 13%
- Waterfall: 10%
- Water/Scrum/Fall: 5%
- Lean: 4%

Why learn RE if it is not much relevant in Agile SDLCs?

• Not all the systems developed by Agile/Scrum
• Documenting changes in the system is also a kind of requirements management ➔ new features defined context for the future features
• RE is not only about documentation
Scrum example

- **Product Owner is a requirements engineering expert**
  - elicits requirements (a.k.a. user stories)
  - manages the scope (system boundary)
  - documents (for effective team communication)
  - manages ‘specification’ (a.k.a. product backlog) by communicating them, prioritising, defining acceptance criteria

- **Product Backlog is a dynamic set of requirements**
  - Backlog refinement = requirements validation

- **Scrum team members**
  - use RE modelling techniques to communicate the RE implementation/understanding
Any questions