Requirements Engineering

Andreas Zeller • Saarland University

Waterfall Model

(1968)

Communication

project initiation
requirements gathering

Planning

estimating
scheduling
tracking

Modeling

analysis
design

Construction

code
test

Deployment

delivery
support
feedback

Based on the Book by Pressman: “Software Engineering – a Practitioner’s Approach”, as well as Wikipedia
Communication

How do we get there?

“Requirement”
Standard Glossary of Software Engineering Terminology

1. A condition or capability needed by a user to solve a problem or achieve an objective.

2. A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents.

3. A documented representation of a condition or capability as in (1) or (2).

A Software Crisis

Denver International Airport (DIA)
Construction started in 1989 • 53 sq miles
• Planned: 1.7 bio USD costs, opening 1993
Glass’ Law

Requirement deficiencies are the prime source of project failures.

“Requirements Analysis”

Standard Glossary of Software Engineering Terminology

- The process of studying user needs to arrive at a definition of system, hardware, or software requirements.
- The process of studying and refining system, hardware, or software requirements.

Analysis vs Design

- Analysis = what the software should do
  - Software functionality
  - Software properties
- Design = how it should do it
Up-front RE

• “We must know [exactly] what to build before we can build it”
• classical engineering viewpoint
• leads to waterfall process
• … but is this realistic for today’s systems?

In our Course

• Gather Requirements with few (≤ 3) iterations
• Gather UI Design with several (≥ 3) iterations

Topics in Requirements Analysis

• Identify Stakeholders
• Elicit Requirements
• Identify Requirements
• Prototypes
Stakeholders

- Persons or organizations who…
  - have a valid interest in the system
  - are affected by the system

Stakeholders

- anyone who operates the system
  (normal and maintenance operators)
- anyone who benefits from the system
  (functional, political, financial and social beneficiaries)
- anyone involved in purchasing or procuring the system

Stakeholders

- organizations which regulate aspects of the system
  (financial, safety, and other regulators)
- organizations responsible for systems which interface with the system under design
- people or organizations opposed to the system
  (negative stakeholders)
Elicit Requirements

- Interviews are the best way to elicit requirements
- Explore requirements systematically
- Sounds simple – but is the hardest part!

Why is Elicitation hard?

- Problems of scope
  What is the boundary of the system? • What details are actually required?

- Problems of understanding
  Users do not know what they want • don’t know what is needed • have a poor understanding of their computing environment • don’t have a full understanding of their domain • omit “obvious” stuff • are ambiguous

- Problems of volatility
  Requirements change over time

Identify Requirements

- Types of requirements
  Functional requirements • Nonfunctional requirements • Constraints

- Contract-style requirements

- Use cases (user stories)
Types of Requirements

Suppose we want to set up a system that tracks who has had how much coffee.

Functional Requirements

• An action the product must take to be useful

The product shall allow to track individual payments of coffee servings

Nonfunctional Requirements

• A property or quality the product must have

The product shall be accessible in multiple languages (such as German and English)
Constraints

- Global requirements – on the project or the product

The product shall be available before March 1st.

Contract Style

Classify product features as

- Must-have features
  “The product must conform to accessibility guidelines”

- May-have features
  “The product may eventually be voice-controlled”

- Must-not-have features
  “The product supports only one language”

Be explicit about must-not-have features!

From “Use cases: requirements in context” By Daryl Kulak, Eamonn Guiney
Contract Style

- Provide a contract between sponsors and developers
- Can run to hundreds of pages
- Abstract all requirements, with little context

Use Case

- An actor is something that can act – a person, a system, or an organization
- A scenario is a specific sequence of actions and interactions between actors (where at least one actor is a system)
- A use case is a collection of related scenarios – successful and failing ones
- Useful for clients as well as for developers

Strengths
- Provides a checklist of requirements.
- Provides a contract between the project sponsor(s) and developers.
- For a large system can provide a high level description.

Weaknesses
- Such lists can run to hundreds of pages. It is virtually impossible to read such
Actors and Goals

• What are the boundaries of the system? Is it the software, hardware and software, also the user, or a whole organization?

• Who are the primary actors – i.e., the stakeholders?

• What are the goals of these actors?

• Describe how the system fulfills these goals (including all exceptions)

Example: SafeHome

Initial Scenario

Use case: display camera views
Actor: homeowner

If I’m at a remote location, I can use any PC with appropriate browser software to log on to the SafeHome Web site. I enter my user ID and two levels of passwords and, once I’m validated, I have access to all the functionality. To access a specific camera view, I select “surveillance” and then “select a camera”. Alternatively, I can look at thumbnail snapshots from all cameras by selecting “all cameras”. Once I choose a camera, I select “view”…
Refined Scenario

Use case: display camera views
Actor: homeowner

1. The homeowner logs on to the Web Site
2. The homeowner enters his/her user ID
3. The homeowner enters two passwords
4. The system displays all major function buttons
5. The homeowner selects “surveillance” button
6. The homeowner selects “Pick a camera”…

Alternative Interactions

- Can the actor take some other action at this point?
- Is it possible that the actor encounters some error condition? If so, which one?
- Is it possible that some other behavior is encountered? If so, which one?

Exploring alternatives is the key to successful requirements analysis!
Use-Case Template for Surveillance

Primary actor: Homeowner.

Goal in context: To view output of camera placed throughout the house from any remote location via the Internet.

Preconditions: System must be fully configured; appropriate user ID and passwords must be obtained.

Trigger: The homeowner decides to take a look inside the house while away.

Scenario:
1. The homeowner logs onto the SafeHome Products Web site.
2. The homeowner enters his or her user ID.
3. The homeowner enters two passwords (each at least eight characters in length).
4. The system displays all major function buttons.
5. The homeowner selects “surveillance” from the major function buttons.
6. The homeowner selects “pick a camera.”
7. The system displays the floor plan of the house.
8. The homeowner selects a camera icon from the floor plan.
9. The homeowner selects the “view” button.
10. The system displays a viewing window that is identified by the camera ID.
11. The system displays video output within the viewing window at one frame per second.

Exceptions:
1. ID or passwords are incorrect or not recognized—see use-case: “validate ID and passwords.”
2. Surveillance function not configured for this system—system displays appropriate error message; see use-case: “configure surveillance function.”
3. Homeowner selects “view thumbnail snapshots for all cameras”—see use-case: “view thumbnail snapshots for all cameras.”
4. A floor plan is not available or has not been configured—display appropriate error message and see use-case: “configure floor plan.”
5. An alarm condition is encountered—see use-case: “alarm condition encountered.”
Suppose we want to set up a system that tracks who has had how much coffee.
Horizontal Prototype

Prototypes

Top Layer (GUI)
Bottom Layer

Vertical Prototype

Top Layer (GUI)
Bottom Layer
Prototypes

- A horizontal prototype tests a particular layer (typically the GUI) of the system
- A vertical prototype tests a particular functionality across all layers
- Resist pressure to turn a prototype into a final result!

What we expect

- A set of requirements
  contract style • 5–10 pages
- A set of use cases
  Pressman style • 20–40 pages
- A GUI design
  covering most of the use cases
- Architectural models and data models
  covering most of the use cases
- An executable prototype
  covering 5–95% of the use cases (negotiable)

All numbers are negotiable depending on project

Summary