Requirements Engineering

Software 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
### 6.6 Map Series Tool

<table>
<thead>
<tr>
<th>Use Case Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Summary</strong></td>
</tr>
<tr>
<td><strong>Actors</strong></td>
</tr>
<tr>
<td><strong>Pre-Conditions</strong></td>
</tr>
<tr>
<td><strong>Post-Conditions</strong></td>
</tr>
<tr>
<td><strong>Priority</strong></td>
</tr>
</tbody>
</table>

#### Scenario

1. **User starts the tool.**
   System displays a list of map series that the user can select from. Default map series will be 'Landscape 1:7920'. Can be set at any scale.

2. **User selects map series on form.**
   System then determines if any boundary features are selected.
   - **Features Selected:**
     1. If features are selected, it asks the user if they want to generate a map series for the selected feature. Only one feature can be used at a time.
   - **No Features Selected:**
     1. If no features are selected, or user opts to select the feature manually, the system prompts the user to select the district and compartment of interest from pull downs. It then zooms to that location, generates the map sheet boundaries, draws them with the map sheet names.

3. **User can select individual sheets on screen, or select to print just an index map, or the entire series.**
   System starts generating and printing maps based on the selected sheets

4. **User collects maps from printer**

#### Notes

#### Deployment

Tool in ArcMap and in ArcGIS Server
Communication

How do we get there?

6.6 Map Series Tool

<table>
<thead>
<tr>
<th>Use Case Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Summary</strong></td>
</tr>
<tr>
<td><strong>Actors</strong></td>
</tr>
<tr>
<td><strong>Pre-Conditions</strong></td>
</tr>
<tr>
<td><strong>Post-Conditions</strong></td>
</tr>
<tr>
<td><strong>Priority</strong></td>
</tr>
</tbody>
</table>

**Scenario**

User starts the tool.
System displays a list of map series that the user can select from. Default map series will be ‘Landscape 1:7920’. Can be set at any scale.
User selects map series on form.
System then determines if any boundary features are selected.
Features Selected:
  i. If features are selected, it asks the user if they want to generate a map series for the selected feature. Only one feature can used at a time.
No Features Selected:
  i. If no features are selected, or user opts to select the feature manually, the system prompts the user to select the district and compartment of interest from pull downs. It then zooms to that location, generates the map sheet boundaries, draws them with the map sheet names.

3) User can select individual sheets on screen, or select to print just an index map, or the entire series.
System starts generating and printing maps based on the selected sheets
4) User collects maps from printer

**Notes**

**Deployment**
Tool in ArcMap and in ArcGIS Server
“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
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
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.
Constraints

• *Global* requirements frequently include *safety* and *security* requirements

The product shall pose no danger, risk, or injury to its users.
## Contract Style

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>The system will support client inquiries from four access points: in person, paper-based mail, voice communication, and electronic communication (Internet, dial-up, and LAN/WAN).</td>
<td>Four access points are how; we should focus instead on who needs access from where.</td>
</tr>
<tr>
<td>The telephone system must be able to support an 800 number system.</td>
<td>An 800 number? Can’t use 888 or 877? Again, what’s missing is who needs what kind of access from where.</td>
</tr>
<tr>
<td>The telephone system must be able to handle 97,000 calls per year and must allow for a growth rate of 15 percent annually. Of these calls it is estimated that 19 percent will be responded to in an automated manner and 81 percent will be routed to call center staff for response. Fifty percent of the calls can be processed without reference to the electronic copy of the paper file, and approximately 50 percent will require access to the system files.</td>
<td>Valuable statistics; this one is actually pretty good.</td>
</tr>
</tbody>
</table>
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!
Contract Style

• Provide a *contract* between sponsors and developers

• Can run to *hundreds of pages*

• Abstract all requirements, with little context
WE INTERVIEWED HUNDREDS OF USERS AND TURNED ALL OF THEIR SUGGESTIONS INTO FEATURES.
Contract Style

love it

hate it
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.
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”…
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

**Use-case:** Access camera surveillance—display camera views (ACS-DCV).

**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.”

**Priority:** Moderate priority, to be implemented after basic functions.

**When available:** Third increment.

**Frequency of use:** Infrequent.
Use-Case Template for Surveillance

Use-case: Access camera surveillance—display camera views (ACS-DCV).

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."
Use-Case Template for Surveillance

Use-case: Access camera surveillance—display camera views (ACS-DCV).

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.”

Priority: Moderate priority, to be implemented after basic functions.

When available: Third increment.

Frequency of use: Infrequent.
What we expect

1. A set of requirements
   contract style • ≤4 pages • safety and security are musts

2. A set of use cases
   Pressman style • ~10–20 pages

3. A GUI design
   covering all “must-have” and most “may-have” use cases

4. Architectural models and data models
   covering all “must-have” and most “may-have” use cases

5. An executable prototype
   covering all “must-have” use cases
What we expect

1. A set of requirements
   contract style • \(\leq 4\) pages

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>The system will support client inquiries from four access points: in person, paper-based mail, voice communication, and electronic communication (Internet, dial-up, and LAN/WAN).</td>
<td>Four access points are how; we should focus</td>
</tr>
<tr>
<td>The telephone system must be able to support a number system.</td>
<td>on how they are supported.</td>
</tr>
</tbody>
</table>

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!
What we expect

2. A set of *use cases*
   Pressman style • ~10–20 pages

---

**Use-Case Template for Surveillance**

Use-case: Access camera surveillance—display camera views (ACS-DCV).

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 and password.
3. The system displays the current camera views.

*Exceptions:*

1. ID or password 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
What we expect

3. A GUI design
covering *all* “must-have” and *most* “may-have” use cases

![Diagram of a GUI design](image-url)
What we expect

4. Architectural models and data models covering *all* “must-have” and *most* “may-have” use cases.
What we expect

5. An executable *prototype* covering all “must-have” use cases
What we expect

We will *calibrate* all contracts to result in similar effort across all projects
Summary

What we expect

1. A set of requirements
   contract style • ≤4 pages • safety and security are musts

2. A set of use cases
   Pressman style • ~10–20 pages

3. A GUI design
   covering all “must-have” and most “may-have” use cases

4. Architectural models and data models
   covering all “must-have” and most “may-have” use cases

5. An executable prototype
   covering all “must-have” use cases