Scenario-based Requirements Engineering and User-Interface Design

Outline

- Background
- Functions / tasks, goals, scenarios / use cases
- Requirements and object-oriented models
- A systematic design process
- Scenarios / use cases for interaction design
- Summary and Conclusion

What are requirements?

- User wishes / needs
- IEEE Standard:
  "A condition or capacity needed by a user to solve a problem or achieve an objective."
- "The \textit{system} shall be able to ..."
  - system to be built
  - composite system
- Example: "The ATM shall accept a cash card."
- Requirements modeling
Scenarios – Stories and narratives

- For representation of
  - cultural heritage
  - explanations of events
  - everyday knowledge
- Human understanding in terms of specific situations
- Human verbal interactions by exchanging stories

Use cases

- “particular cases of how the system is to be used”
- Use-Case Report (according to Unified Process):
  1. Brief Description
  2. Flow of Events
  3. Special Requirements
  4. Pre-conditions
  5. Post-conditions
  6. Extension Points
  7. Relationships
  8. Use-Case Diagrams
  9. Other Diagrams
Scenario-based Requirements Engineering and User-Interface Design

Use-case diagram

- UML graphical notation
- Ellipse: use case

  Name of use case

- Stick man: actor

  Name of actor

- Connecting line: association

Interaction design

- Design of interactions between human and computer
- Relation to requirements engineering
- Relation to task analysis
- No commitment to specific user interface
Widgets

- Interactive objects presented on the display
  - windows
  - buttons
  - scroll bars
- User interface elements
- Classification hierarchy of widgets

Outline

- Background
- Functions / tasks, goals, scenarios / use cases
- Requirements and object-oriented models
- A systematic design process
- Scenarios / use cases for interaction design
- Summary and Conclusion
Glossary

Scenarios: “sequences of actions aimed at accomplishing some task goal”
Goals: “partially specified states that the user considers as desirable”
Use cases: “particular cases of how the system is to be used”, “classes of scenarios”
Functions: “effects achieved by some entity”
Tasks: “pieces of work that a person or other agent has to (or wishes to) perform”

Functional requirements

- Describe required functionality not yet available
- Functional user requirements may be high-level statements of what the system should be able to do.
- Functional software/system requirements should describe the functions of the software/system to be built in detail (but not yet its design or implementation).
Rent Available Video:

A member of the video store identifies himself/herself to VSS (Video Store Software).

VSS shall check the identification.

If the identification is successful, VSS shall start a transaction and present a selection of video titles.

The member selects a video title that is available and indicates the intent to rent (a copy of) the video.

VSS shall book this rental on the account of the member and ask the clerk to hand out a video copy to the member.

The clerk hands out a copy of the video title and acknowledges this to VSS.

VSS shall again present a selection of video titles.

The member does not select further titles, but initiates the termination of the transaction.

VSS shall issue a good-bye message and terminate the transaction.
By-Function – Video Store Example

A member of the video store identifies himself/herself to VSS (Video Store Software).

VSS shall check the identification.
*By-Function:* Member Identification Check

... 

VSS shall book this rental on the account of the member and ask the clerk to hand out a video copy to the member.
*By-Function:* Video Rental Booking
  Video Handing-out Request

... 

Functional requ. – Video Store example

*Rent Available Video*  *By-Function*  *Video Rental Booking*

**Video Rental Booking:**

VSS shall book the rental of a copy of a video title on the account of the member, and reduce the number of available copies of the video title by 1.
Goal – Video Store example

Member Has Video for Rent By-Scenario Rent Available Video

Member Has Video for Rent:

A member of the video store has a copy of a video title from the store for rent.
Scenario-based Requirements Engineering and User-Interface Design

Outline

- Background
- Functions / tasks, goals, scenarios / use cases
- Requirements and object-oriented models
- A systematic design process
- Scenarios / use cases for interaction design
- Summary and Conclusion

Requirements and object-oriented models

Model

Domain | Requirements

Real world

Abstraction
Types of requirements

- «stereotype» Requirement
  - «stereotype» Envisioned Scenario
  - «stereotype» Functional Requirement
  - «stereotype» Quality Requirement
    - «stereotype» Constraint on System
      - «stereotype» Constraint on Process

Types of requ. – Constraints on system

- Performance
- Reliability
- Security
- Safety
- Portability
- Maintainability
- Reusability
- Interface
- Usability
Types of req. – Constraints on process

- Specific development process to follow
- Specific programming language for implementation
- Specific tools to be used
- Specific hardware to be used
- Political issues
- Time to market
- Terms of delivery
- Cost

Conflicts between Quality Requirements

- VSS example
  - VSS shall allow direct access to member data.
  - VSS shall protect member data from illegal access.
- Usability vs. Security
- Trade-off
- Common in complex systems
OOA model – ATM Example

Transaction

date-time

Cashier transaction

Remote transaction

Cash notes

on hand
dispensed

Customer

name

address

password

Cashier Station

ATM

Cash Card

bank code
card code
serial number

Entry Station

out of order

Institute of Computer Technology

OOA model adapted – ATM Example

Transaction

date-time

Cashier transaction

Remote transaction

Cash notes

on hand
dispensed

Customer

name

address

password

Cashier Station

ATM

Cash Card

bank code
card code
serial number

Entry Station

out of order

Institute of Computer Technology
OOA model – UML sequence diagram

- Represents a scenario
- Interaction of instances
- Activation
- System border

OOA model – Video Store example

- Video Title
- Video
- Member
- Clerk
- VSS
- Customer
Requirements vs. requirements representation

- Reuse of requirements representation only
- Distinction between descriptive and model-based
  - Descriptive: need described
  - Model-based: abstraction of what the system should look like

Outline

- Background
- Functions / tasks, goals, scenarios / use cases
- Requirements and object-oriented models
- A systematic design process
- Scenarios / use cases for interaction design
- Summary and Conclusion
Systematic process

Idea: navigation in the metamodel graph
Excerpt:

Goals
  By-Scenario
Scenarios
  By-Function
Functional Requirements
Scenario-based Requirements Engineering and User-Interface
Design

Systematic process

**Idea:** navigation in the metamodel graph

**Excerpt:**

- **Goals**
  - By-Scenario
- **Scenarios**
  - By-Function
- **Functional Requirements**

**What is known already?**

**Old system or system to be built?**

---

Systematic process – Given goals

1. If some goal is known from the old system, then figure out whether this is still a goal in the new system that will include the system to be built.
   
   E.g., Meeting a Friendly Person, Customer Has Cash.

2. If some goal is known for the new system, then try to link it to one or more scenarios for the new system that are already known.
   
   E.g., Customer Has Receipt – Get Cash from ATM.

3. If some goal that is known for the new system cannot be linked to any scenario for the new system, then develop one or more such scenarios and link them to the goal.
   
   E.g., Customer Has Cash – Get Cash from ATM.
Systematic process – Given scenarios

1. If some scenario is known from the old system, then determine the goals that are achieved through it.
   E.g., Get Cash from Human Cashier – Customer Has Cash.

2. If some scenario is known from the old system, then try to develop an analogous scenario for the new system.
   E.g., Get Cash from Human Cashier – Get Cash from ATM.

3. If some scenario is known for the new system, then try to link it to one or more goals and, each action contained in it to one or more functions for the new system that are already known.
   E.g., Get Cash from ATM – Customer Has Cash – Cash Provision.

Systematic process – Given scenarios (cont.)

4. If some scenario that is known for the new system cannot be linked to any goal for the new system, then determine one or more goals and link them to the scenario.
   E.g., Get Cash from ATM – Customer Has Cash.

5. If one or more actions contained in some scenario that is known for the new system cannot be linked to any function for the new system, then develop one or more such functions and link them to the actions of this scenario.
   E.g., Get Cash from ATM – Receipt Provision.
Systematic process – Given functional req.

1. If some function is known from the old system, then figure out whether this is still a required function in the new system that will include the system to be built. E.g., finger prints – Cash Card Acceptance, Cash Provision.

2. If some function is known for the new system, then try to link it to one or more actions contained in scenarios for the new system that are already known. E.g., Check Amount – Get Cash from ATM.

3. If some function that is known for the new system cannot be linked to any action contained in any scenario for the new system, then develop one or more such scenarios and link one or more actions contained in them to the function. E.g., money transfer between accounts.

Systematic process (cont.)

- Partial sequences of steps selected according to what is known – agenda
- Both model-driven and data-driven
- Successful termination – agenda empty
- Improvement of
  - Completeness
  - Non-redundancy
  - Understandability
- But no guarantee
Scenario-based Requirements Engineering and User-Interface Design

Institute of Computer Technology
Connect with Goals

Each scenario should be connected with at least one goal that can be achieved through its execution.

The underlying idea is to model scenarios that are both necessary and useful. In addition, a scenario may be better understood when some corresponding goal is known.

In the case that a scenario cannot be connected appropriately with any goal that is already represented, either some goals may yet be missing in the representation or the scenario may not serve any goal of the user.

Please try to follow the instructions in sequence, where the one highlighted through a white background is the current one.

1. Select the scenario instance "Starting A New Game".
2. Expand its portion content by pressing the push button "+".
3. Create a tuple of the association by selecting "Insert".
4. Select in the popup window named "Create Instance Connection" those goals from the list to be connected with the scenario. Close this window by pressing the push button "Connect", which will connect the scenario with these goals.
5. Perform Instruction.
Supporting the Primary Tasks

- Presenting choice of conceptual entities to be modeled
- Providing example descriptions of conceptual entities
- Guidance in linking entities according to the method
Supporting the Secondary Tasks

- Active guidance through step-by-step instructions and monitoring
- Immediate feedback at each point
- Letting system perform actions on one’s behalf

Outline

- Background
- Functions / tasks, goals, scenarios / use cases
- Requirements and object-oriented models
- A systematic design process
  - Scenarios / use cases for interaction design
  - Summary and Conclusion
Interaction scenarios with attached task descriptions

- Scenarios as a prerequisite
- Example scenario: Get Cash from ATM
- Attached tasks:
  - Entering PIN code
  - Cash provision
  - ...
- Tasks of user and machine

1. Put tasks into predefined categories

   Based on a hierarchy of task categories

   ![Task hierarchy diagram]

   - Task
   - System Task
   - User Task
   - Complementary Task
   - Output Task
   - Internal System Task
2. Aggregate complementary tasks

- Task of user and complementary task of machine
- Together make up an interaction through the UI

```
<table>
<thead>
<tr>
<th>Interaction Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complementary Task</td>
</tr>
<tr>
<td>User Task</td>
</tr>
</tbody>
</table>
```

3. Compose and decompose tasks

Find right granularity

- aggregate
- decompose

```
<table>
<thead>
<tr>
<th>Aggregate Interaction Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interaction Task</td>
</tr>
</tbody>
</table>
```
4. Classify tasks according to the kind of interaction

Based on defined hierarchy of Interaction Tasks

- Interaction Task
  - Task of Selecting an Action
  - Aggregate Interaction Task
  - I/O Task

5. Map to widget classes

<table>
<thead>
<tr>
<th>Task</th>
<th>Widget</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task of selecting an action</td>
<td>Widget of selecting an action</td>
</tr>
<tr>
<td>Aggregate interaction task</td>
<td>Container</td>
</tr>
<tr>
<td>I/O Task</td>
<td>Control</td>
</tr>
</tbody>
</table>
Essential use cases

- Larry Constantine
- Essential modeling
- Abstract usage of system
- ATM example:

<table>
<thead>
<tr>
<th>User</th>
<th>ATM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identify self</td>
<td>Verify identity</td>
</tr>
<tr>
<td>Choose</td>
<td>Dispense cash</td>
</tr>
<tr>
<td>Take cash</td>
<td></td>
</tr>
</tbody>
</table>

Concrete vs. abstract – ATM Example

- John inserts his cash card with the magnetic stripe up into the slot of the ATM machine of the bank ...
- The customer initiates a transaction by inserting a cash card.
- Insert card
- Identify self
- Request

There is a whole spectrum!
Concrete vs. abstract – Discussion

- Being concrete is one of the main points of scenarios, in contrast to abstract specifications!
- Scenarios may talk about more general uses through specific examples.
- How concrete is best (for which purpose)?
- Too much detail may contain built-in, premature assumptions (about a UI).
- Abstract descriptions leave choices open in the UI design.

Content model

- Larry Constantine
- Abstract interface contents
- Abstract components to be supplied by the user interface, placeholders for the actual visual components in the implemented interface

- Video Titles (list of titles)
- Member Identification
- Members (list of members)
Process of content modeling

- Larry Constantine
- Examining (validated) use case narratives
- For each user step, what will have to be supplied for user to complete step?
- External view
- Language of user and application domain
- What information will be needed by and from the user?
- What functions will be needed by the user?

Essential use case – Video Store Example

<table>
<thead>
<tr>
<th>User</th>
<th>Clerk</th>
<th>VSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identify self</td>
<td></td>
<td>Verify identity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Offer video title choices</td>
</tr>
<tr>
<td>Choose video title</td>
<td></td>
<td></td>
</tr>
<tr>
<td>for rent</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hand out copy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Interaction context

- Larry Constantine
- Different tasks carried out in different spaces / contexts
- Distinct interaction contexts in UI

Context navigation map

- Larry Constantine
- Navigational relationships among interaction contexts
Implementation model

- Larry Constantine
- Layout of the UI and interaction between user and system
- How to embody each abstract interaction context in the UI as an actual interaction context? E.g., as a screen, window, dialogue box, etc.
- How to realize each abstract component as some actual visual component on the UI? E.g., as a tool, command button, text box, etc.

Outline

- Background
- Functions / tasks, goals, scenarios / use cases
- Requirements and object-oriented models
- A systematic design process
- Scenarios / use cases for interaction design
- Summary and Conclusion
Summary and Conclusion

- Goals, scenarios and functions / tasks can be combined.
- This combination serves as a basis for a systematic approach.
- Scenarios / use cases may help both in requirements engineering and in interaction design.
- In this sense, scenario-based requirements engineering can facilitate interaction design.

Literature

Selected work of this tutorial presenter