REQUIREMENTS ENGINEERING

REQUIREMENTS, USER STORIES AND USE CASES

Coordinator: Dr Kuldar Taveter
Lecturer: Dr Marinos Georgiadis
Mentors: Dr Marinos Georgiadis, Rafał Włodarski
OUTLINE

- Requirements
- Elicitation Techniques
- User Stories
- Use Cases
REQUIREMENTS ENGINEERING (RE)

- Decomposed to:
  - Requirements elicitation;
  - Requirements analysis;
  - Requirements specification
  - Requirements validation;
  - Requirements management.

- In practice, RE is an iterative activity.
Requirements elicitation – the process and techniques used by systems analysts to identify or extract system problems and solution requirements from the user community.

Requirement – something that the information system must do or a property that it must have.
- **Functional requirement** - something the information system must do
  - E.g., Create Prescription
  - CRUD

- **Nonfunctional requirement** - a property or quality the system must have
  - Performance (e.g., Print prescription in 10 seconds)
  - Security
  - Usability
  - Reliability

- **NFRs may be:**
  - Solution-wide or impacting a group of functional requirements: e.g.
    - All functionality must carry the company logo
    - All functionality must respond within 2 seconds to requests
  - Related to one particular functional requirement
• Business requirements
  • Define the business problems to be solved or the business opportunities to be addressed by the software product. In general, the business requirements define why the software product is being developed. Business requirements are typically stated in terms of the objectives of the customer or organization requesting the development of the software.
  • Product Vision

• User Requirements
  • Statements in natural language plus diagrams of the services the system provides and its operational constraints. Written for customers.

• System requirements
  • Detailed descriptions of the system’s functions, services and operational constraints.
  • Defines what should be implemented so may be part of a contract between client and contractor.
  • For each iteration
The starting point for software product development is a ‘product vision’.

Product visions can be simple statements that define the essence of the product to be developed.

The product vision should answer three fundamental questions:

- What is the product to be developed?
- Who are the target customers and users?
- Why should customers buy this product?
PRODUCT VISION TEMPLATE

- FOR (target customer)
- WHO (statement of the need or opportunity)
- The (PRODUCT NAME) is a (product category)
- THAT (key benefit, compelling reason to buy)
- UNLIKE (primary competitive alternative)
- OUR PRODUCT (statement of primary differentiation)
FOR teachers and educators WHO need a way to help students use web-based learning resources and applications, THE iLearn system is an open learning environment THAT allows the set of resources used by classes and students to be easily configured for these students and classes by teachers themselves. UNLIKE Virtual Learning Environments, such as Moodle, the focus of iLearn is the learning process rather than the administration and management of materials, assessments and coursework. OUR product enables teachers to create subject and age-specific environments for their students using any web-based resources, such as videos, simulations and written materials that are appropriate.
EXAMPLES OF REQUIREMENTS

- **User requirement**
  - “Customer makes a call
  - Receptionist records date and time of the appointment”
  OR
  - “As a Receptionist, I want be able to record an appointment, so that I know when a customer will come.”

- **System requirement**
  - “Receptionist selects to see appointment form on her device’s screen
  - System shows appointment form
  - Receptionist enters customer’s ID
  - System checks if customer is already in the database
  - System returns relevant message
  - If the customer does not exist in the database, Receptionist selects the New Customer form, otherwise…”
ELICITATION TECHNIQUES

- Sampling of existing documentation, forms, and databases.
- Observation of the work environment.
- Questionnaires.
- Interviews.
- Prototyping.
- Meetings.
Sampling – process of collecting a representative sample of documents, forms, and records.

- Organization chart
- Memos and other documents that describe the problem
- Standard operating procedures for current system
- Completed forms
- Manual and computerized screens and reports
- Samples of databases (entities, attributes, etc.)
- Flowcharts and other system documentation
- And more
WHY TO SAMPLE COMPLETED RATHER THAN BLANK FORMS

- Can determine type of data going into each blank
- Can determine size of data going into each blank
- Can see data relationships
Interview - a fact-finding technique whereby the systems analysts collect information from individuals through face-to-face interaction.

- Find facts
- Verify facts
- Clarify facts
- Generate enthusiasm
- Get the end-user involved
- Identify requirements
- Solicit ideas and opinions

The personal interview is generally recognized as the most important and most often used fact-finding technique.
Unstructured interview – conducted with only a general goal or subject in mind and with few, if any, specific questions. The interviewer counts on the interviewee to provide a framework and direct the conversation.

Structured interview – interviewer has a specific set of questions to ask of the interviewee.

Open-ended question – question that allows the interviewee to respond in any way. (what, why)

Closed-ended question – a question that restricts answers to either specific choices or short, direct responses. (Is.., Who, When)
Advantages

- Give analyst opportunity to motivate interviewee to respond freely and openly
- Allow analyst to probe for more feedback
- Permit analyst to adapt or reword questions for each individual
- Can observe nonverbal communication

Disadvantages

- Time-consuming
- Success highly dependent on analyst’s human relations skills
- May be impractical due to location of interviewees
PROCEDURE TO CONDUCT AN INTERVIEW

1. Select Interviewees
   • End users
   • Learn about individual prior to the interview

2. Prepare for the Interview
   • interview guide

3. Conduct the Interview

4. Follow Up on the Interview
   • Memo that summarizes the interview
**Sample Interview Guide**

**Interviewee:** Jeff Bentley, Accounts Receivable Manager  
**Date:** January 19, 2003  
**Time:** 1:30 P.M.  
**Place:** Room 223, Admin. Bldg.  
**Subject:** Current Credit-Checking Policy

<table>
<thead>
<tr>
<th>Time Allocated</th>
<th>Interviewer Question or Objective</th>
<th>Interviewee Response</th>
</tr>
</thead>
</table>
| 1 to 2 min.    | **Objective**  
Open the interview:  
- Introduce ourselves  
- Thank Mr. Bentley for his valuable time.  
- State the purpose of the interview — to obtain an understanding of the existing credit-checking policies. | |
| 5 min.         | **Question 1**  
What conditions determine whether a customer’s order is approved for credit?  
Follow-up | |
| 5 min.         | **Question 2**  
What are the possible decisions or actions that might be taken once these conditions have been evaluated?  
Follow-up | |
| 3 min.         | **Question 3**  
How are customers notified when credit is not approved for their order?  
Follow-up | |
### Question 4
After a new order is approved for credit and placed in the file containing orders that can be filled, a customer might request that a modification be made to the order. Would the order have to go through credit approval again if the new total order cost exceeds the original cost?

**Follow-up**

### Question 5
Who are the individuals who perform the credit checks?

**Follow-up**

### Question 6
May I have permission to talk to those individuals to learn specifically how they carry out the credit-checking process?

**Follow-up**
If so: When would be an appropriate time to meet with each of them?

### Objective
Conclude the interview.
- Thank Mr. Bently for his cooperation and assure him that he will be receiving a copy of what transpired during the interview.

<table>
<thead>
<tr>
<th>Time Allotted</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>21 minutes</td>
<td>Time allotted for questions and objectives</td>
</tr>
<tr>
<td>9 minutes</td>
<td>Time allotted for follow-up questions and redirection</td>
</tr>
<tr>
<td>30 minutes</td>
<td>Time allotted for interview (1:30 p.m. - 2:00 p.m.)</td>
</tr>
</tbody>
</table>

**General Comments and Notes:**
INTERVIEW TIPS

- Ask if you can record the interview
  - Make sure the recorder is visible
  - Say that they can turn it off at any time.
  - In any case, take notes!
- **Try to cover the most relevant “W” questions**
  - *Who* would be the typical users/roles of the system?
  - *Why* will each user type use the system?, What problem it solves?
  - *What* should the system do for each user?
  - *When* (in which circumstances) will each user use the system?
  - *What* information do you exchange with other users?
- **Follow up interesting leads**
  - e.g., “Could we pursue what you just said a little further?”
- **Ask open-ended questions towards the end**
  - e.g., “Is there anything else you would like to add?”
    - e.g., “What else would you like to add?”
MEETINGS (GROUP)

- Used for summarization and feedback
  - E.g., meet with stakeholders towards the end of each stage:
    - to discuss the results of the information gathering stage
    - to conclude on a set of requirements
    - to agree on a design etc.
  - Use the meeting to confirm what has been learned, talk about findings

- Every meeting should have a clear objective

- Plan the meeting carefully:
  - Schedule the meeting and arrange for facilities
  - Prepare an agenda and distribute it well in advance
  - Keep track of time and agenda during the meeting
  - Follow up with a summary to be distributed to meeting participants
A COMBINED STRATEGY

1. Learn from existing **documents**, forms, reports, and files.
2. If appropriate, **observe** the system in action.
3. Given all the facts that already collected, design and distribute **questionnaires** to clear up things that aren’t fully understood.
4. Conduct **interviews** (or group work meetings).
5. Build discovery **prototypes** for any functional requirements that are not understood or for requirements that need to be validated.
6. Follow up (e.g., with meetings) to verify facts.
User stories are finer-grain narratives that set out in a more detailed and structured way a single thing that a user wants from a software system.

Standard format:

- As a <role | persona> I <want | need> to <do something> so that <reason>

  - As a teacher, I need to be able to report who is attending a class trip so that the school maintains the required health and safety records.

- Product backlog includes user stories.

- User stories should focus on a clearly defined system feature or aspect of a feature that can be implemented within a single iteration.
Acceptance criteria: define the boundaries of a user story, and they are used to confirm when a story is completed and working as intended.

For example:
“As a conference attendee, I want to be able to register online, so I can register quickly and cut down on paperwork”, the acceptance criteria could include:
- A user cannot submit a form without completing all the mandatory fields.
- Information from the form is stored in the registrations database.
- Protection against spam is working.
- Payment can be made via credit card.
- An acknowledgment email is sent to the user after submitting the form.
It makes sense in any product development to spend time trying to understand the potential users and customers of your product.

A range of techniques have been developed for understanding the ways that people work and use software.

- These include user interviews, surveys, observation, etc.
- Some of these techniques are expensive and unrealistic for small companies.

Informal user analysis and discussions, which simply involve asking users about their work, the software that they use, and its strengths and weaknesses are inexpensive and very valuable.
Personas are ‘potential users’ where you create a character portrait of a type of user that you think might use your product.

- For example, if your product is aimed at managing appointments for dentists, you might create a dentist persona, a receptionist persona and a patient persona.

- Personas of different types of user help you imagine what these users may want to do with your software and how it might be used.

- You should describe their background and why they might want to use your product.

- You should also say something about their educational background and technical skills.

- These help you assess whether or not a software feature is likely to be useful, understandable and usable by typical product users.
Jack, a primary school teacher
Jack, age 32, is a primary school (elementary school) teacher in Ullapool, a large coastal village in Scottish Highlands. He teaches children from ages 9-12. He was born in a fishing community north of Ullapool, where his father runs a marine fuels supply business and his mother is a community nurse. He has a degree in English from Glasgow University and retrained as a teacher after several years working as a web content author for a large leisure group.

Jack’s experience as a web developer means that he is confident in all aspects of digital technology. He passionately believes that the effective use of digital technologies, blended with face to face teaching, can enhance the learning experience for children. He is particularly interested in using the iLearn system for project-based teaching, where students work together across subject areas on a challenging topic.
Emma, a history teacher
Emma, age 41, is a history teacher in a secondary school (high school) in Edinburgh. She teaches students from ages 12 to 18. She was born in Cardiff in Wales where both her father and her mother were teachers. After completing a degree in history from Newcastle University, she moved to Edinburgh to be with her partner and trained as a teacher. She has two children, aged 6 and 8, who both attend the local primary school. She likes to get home as early as she can to see her children, so often does lesson preparation, administration and marking from home.

Emma uses social media and the usual productivity applications to prepare her lessons, but is not particularly interested in digital technologies. She hates the virtual learning environment that is currently used in her school and avoids using it if she can. She believes that face-to-face teaching is most effective. She might use the iLearn system for administration and access to historic films and documents. However, she is not interested in a blended digital/face-to-face approach to teaching.
The main benefit of personas is that they help you and other development team members empathize with potential users of the software.

Personas help because they are a tool that allows developers to ‘step into the user’s shoes’.

Personas can help you check your ideas to make sure that you are not including product features that aren’t really needed.
Personas should be based on an understanding of the potential product users, their jobs, their background and their aspirations.

You should **study and survey potential users** to understand what they want and how they might use the product.

From this data, you can then abstract the essential information about the different types of product user and use this as a basis for creating personas.
A scenario is a narrative that describes how a user, or a group of users, might use your system.

There is no need to include everything in a scenario – the scenario isn’t a system specification.

It is simply a description of a situation where a user is using your product’s features to do something that they want to do.

Scenario descriptions may vary in length from two to three paragraphs up to a page of text.
Fishing in Ullapool
Jack is a primary school teacher in Ullapool, teaching P6 pupils. He has decided that a class project should be focused around the fishing industry in the area, looking at the history, development and economic impact of fishing.

As part of this, students are asked to gather and share reminiscences from relatives, use newspaper archives and collect old photographs related to fishing and fishing communities in the area. Pupils use an iLearn wiki to gather together fishing stories and SCRAN (a history archive site) to access newspaper archives and photographs. However, Jack also needs a photo-sharing site as he wants students to take and comment on each others’ photos and to upload scans of old photographs that they may have in their families. He needs to be able to moderate posts with photos before they are shared, because pre-teen children can’t understand copyright and privacy issues.

Jack sends an email to a primary school teachers’ group to see if anyone can recommend an appropriate system. Two teachers reply and both suggest that he uses KidsTakePics, a photo-sharing site that allows teachers to check and moderate content. As KidsTakePics is not integrated with the iLearn authentication service, he sets up a teacher and a class account with KidsTakePics.

He uses the iLearn setup service to add KidsTakePics to the services seen by the students in his class so that, when they log in, they can immediately use the system to upload photos from their phones and class computers.
SCENARIO ELEMENTS

• A brief statement of the overall objective.
  • In Jack’s scenario, this is to support a class project on the fishing industry.

• References to the personas involved (Jack) so that you can get information about the capabilities and motivation of that user (role).

• Information about what is involved in doing the activity.
  • E.g.: gathering reminiscences from relatives, accessing newspaper archives, etc.

• An explanation of problems that can’t be readily addressed using the existing system.
  • Young children don’t understand issues such as copyright and privacy, so photo sharing requires a site that a teacher can moderate to make sure that published images are legal and acceptable.

• A description of one way that the identified problem might be addressed.
  • In Jack’s scenario, the preferred approach is to use an external tool designed for school students.
Emma is teaching the history of the First World War to a class of 14 year olds (S3). A group of S3 students are visiting the historic World War One battlefields in northern France. She wants to set up a ‘battlefields group’ where the students who are attending the trip can share their research about the places they are visiting as well as their pictures and thoughts about the visit.

From home, she logs onto the iLearn system using her Google account credentials. Emma has two iLearn accounts – her teacher account and a parent account associated with the local primary school. The system recognises that she is a multiple account owner and asks her to select the account to be used. She chooses the teacher account and the system generates her personal welcome screen. As well as her selected applications, this also shows management apps that help teachers create and manage student groups.

Emma selects the ‘group management’ app, which recognizes her role and school from her identity information and creates a new group. The system prompts for the class year (S3) and subject (history) and automatically populates the new group with all S3 students who are studying history. She selects those students going on the trip and adds her teacher colleagues, Jamie and Claire, to the group.
EMMA’S SCENARIO: USING ILEARN FOR ADMINISTRATION

She names the group and confirms that it should be created. The app sets up an icon on her iLearn screen to represent the group, creates an email alias for the group and asks Emma if she wishes to share the group. She shares access with everyone in the group, which means that they also see the icon on their screen. To avoid getting too many emails from students, restricts sharing of the email alias to Jamie and Claire.

The group management app then asks Emma if she wishes to set up a group web page, wiki and blog. Emma confirms that a web page should be created and she types some text to be included on that page.

She then accesses flickr using the icon on her screen, logs in and creates a private group to share trip photos that students and teachers have taken. She uploads some of her own photos from previous trips and emails an invitation to join the photo-sharing group to the Battlefield email list. Emma uploads material from her own laptop that she has written about the trip to iLearn and shares this with the ‘Battlefields Group’. This action adds her documents to the web page and generates an alert to group members that new material is available.
Scenarios should always be written from the user’s perspective and based on identified personas or real users.

Your starting point for scenario writing should be the personas that you have created. You should normally try (more than 1 member) to imagine several scenarios from each persona.

Ideally, scenarios should be general and should not include implementation information.
USER INVOLVEMENT

- It is easy for anyone to read and understand scenarios, so it is possible to get users involved in their development.

- The best approach is to develop an imaginary scenario based on our understanding of how the system might be used, then ask users to explain what you have got wrong.

- They might ask about things they did not understand and suggest how the scenario could be extended and made more realistic.

- Experience shows that users are not good at writing scenarios.
  
  • The scenarios that they usually create are based on how they work at the moment. They are far too detailed and the users can’t easily generalize their experience.
Scenario statement: From home, she logs onto the iLearn system using her Google account credentials.

User stories:

As a teacher, I want to be able to log in to my iLearn account from home using my Google credentials so that I don’t have to remember another login id and password.

As a teacher, I want to access the apps that I use for class management and administration.

As a teacher and parent, I want to be able to select the appropriate iLearn account so that I don’t have to have separate credentials for each account.
Stories can be used to describe features in your product that should be implemented.

Each feature can have a set of associated stories that describe how that feature is used.
USER STORIES DESCRIBING THE FEATURE “GROUPS”

As a teacher, I want to be able to send email to all group members using a single email address.

As a teacher, I want to be able to share uploaded information with other group members.

As a teacher, I want the iLearn system to automatically set up sharing mechanisms such as wikis, blogs and web sites.

As a teacher, I want to be able to create a group of students and teachers so that I can share information with that group.

As a teacher, I want the system to make it easy for me to select the students and teachers to be added to a group.
As you can express all of the functionality described in a scenario as user stories, do you really need scenarios?

Scenarios are more natural and are helpful for the following reasons:

- Scenarios read more naturally because they describe what a user of a system is actually doing with that system. People often find it easier to relate to this specific information rather than the statement of wants or needs set out in a set of user stories.

- If you are interviewing real users or are checking a scenario with real users, they don’t talk in the stylized way that is used in user stories. People relate better to the more natural narrative in scenarios.

- Scenarios often provide more context - information about what the user is trying to do and their normal ways of working. You can do this in user stories, but it means that they are no longer simple statements about the use of a system feature.
Definition: A use case is **all the ways** of using a system to achieve a particular goal for a particular user.

Taken together the set of all the use cases gives you all of the useful ways to use the system.

**Principles at the heart of any successful application of use cases:**

1. Understand the big picture
2. Focus on value
3. Build the system in slices
4. Deliver the system in increments
Without an understanding of the system as a whole you will find it impossible to make the correct decisions about:

- what to include in the system
- What to leave out of it
- what it will cost
- what benefit it will provide

A use-case diagram is a simple way of presenting an overview of a system’s requirements. You can see

- all the ways the system can be used
- who starts the interaction
- any other parties involved.
The use case diagram shows the system boundary, the use cases internal to the system, and the actors external to the system,
The basic flow is needed if the use case is ever to be successfully completed; this must be implemented first.

The alternatives though are optional. They can be added to the basic flow as and when they are needed.

So, you focus on what is most important/valuable.
Each UC scenario is a good candidate slice. It is defined by part of the use-case narrative and one or more of the accompanying test cases.

Select the most useful use case to find the most useful thing that the system does.

To find the most central slice you will need to shed all the less important ways of achieving the goal and handling problems. You can do this by focusing on the scenario described by the basic flow. The first slice might just be the basic flow and one test case.

Iteration 1: At least two slices from two core use cases, preferably the main flows from the two use cases plus their test cases. More on slide 50.
The slices are more than just requirements and test cases

- The slice goes through
  - Requirements
  - Design
  - Implementation
  - Test cases and test scripts used to execute the test cases and the test results generated.

- Traceability
▪ Incremental development of a release of a system
• Start by finding actors and use cases
• Select and prioritize the parts (slices) of the use cases to be developed.
• Detail the use-cases slices and more importantly their tests
• Design – Use Case Realization
• Implement software for a slice.
• Execute the tests, tracking progress in terms of verified, working software, and
• Feed back the results in order to handle change and better support the team.
ITERATION 1 - DELIVERABLES

- Product vision
- Definition of Functional and Non-Functional Requirements at a higher level – User Stories – Features (Group your user stories)
- Project plan
  - List of all project tasks including user stories (backlog)
  - Define in detail the tasks of the second iteration and their deliverables
- At least two slices from two core use cases, preferably the main flows from the two use cases plus their test cases (detailed user stories including their acceptance criteria are another option, but we still want use specifications to be developed).
- Links where necessary between user requirements (user stories), system requirements (use case specifications or detailed user stories), non-functional requirements and tasks in issue tracker
• Other possible deliverables
  • a Use Case diagram, including at least the two use cases for which their detailed description will be specified,
  • a sequence diagram and an activity diagram for each of the aforementioned use cases
  • a class diagram and database design may be parts of this iteration.
  • However, apart from the Use Case diagram, the rest of the diagrams and database design may be planned for the next iteration or may be refined during that if started in the first iteration.
  • For creating UML diagrams
    • You may use an external UML tool and add links in your Wiki to these diagrams or just upload them to your Wiki.
    • You may use PlantUML or a similar tool
    • If possible, make the necessary links in the issue tracker

• Collaboration infrastructure
  • VCS, Wiki, Issue Tracker
THANK YOU
SOME EXTRA MATERIAL Follows
When looking for actors, ask the following questions:

- Who or what provides inputs to the system?
- Who or what receives outputs from the system?
- Are interfaces required to other systems?
- Are there events that are automatically triggered at a predetermined time?
- Who will maintain information in the system?

Actors should be named with a noun or noun phrase
In most projects, the most important use cases are developed first.

**Use-case ranking and priority matrix** – a tool used to evaluate use cases and determine their priority.

- Evaluates use cases on 1-5 scale against six criteria.
  1. Significant impact on the architectural design.
  2. Easy to implement but contains significant functionality.
  3. Includes risky, time-critical, or complex functions.
  4. Involves significant research or new or risky technology.
  5. Includes primary business functions.
  6. Will increase revenue or decrease costs.
# Sample Use-Case Ranking and Priority Matrix

<table>
<thead>
<tr>
<th>Use-Case Name</th>
<th>Ranking Criteria, 1 to 5</th>
<th>Total Score</th>
<th>Priority</th>
<th>Build Cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Submit Subscription Order</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Place New Order</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Make Product Inquiry</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Establish New Member Subscription Program</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>Generate Daily 10-30-60-Day Default Agreement Report</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Revise Order</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>
Use-case dependency diagram – graphical depiction of the dependencies among use cases.

- Provides the following benefits:
  - Graphical depiction of the system’s events and their states enhances understanding of system functionality.
  - Helps identify **missing** use cases.
  - Helps facilitate project management by depicting **which use cases are more critical**.
SAMPLE USE-CASE
DEPENDENCY DIAGRAM

Establish New Member Subscription Program

Submit Subscription Order

Place New Order

Revise Order

Generate Daily 10-30-60 Day Default Agreement Report

Make Product Inquiry

Depends on

Depends on

Depends on

Depends on
REFERENCES


