NÕUETE ESITAMINE
(REQUIREMENTS SPECIFICATION)
Eelmisel korral

• Mis on süsteem?
  – Tarkvaramahukas süsteem;
  – Sotsiotehniline süsteem;
  – Kontrollsüsteem;
  – Infosüsteem
• Nõuete analüüsi väljakutsed
• Mis on nõue?
• Nõuete esitamise tasemed
• Nõuete tüübid
• Mis on nõuete analüüs?
• Nõude kolm põhiomadust
• Nõuete liigid
  – Funktsionaalsed nõuded
  – Mittefunksionaalsed nõuded ehk kvaliteedinõuded
• Nõuete esitamise viisid
  – Loomulik keel
Nõuete esitamise viisid

• Natural language
• Structured natural language, for example user stories and scenarios
• Graphical notations, for example, use cases, activity diagrams, and goal models
• Mathematical specifications, for example, in the Z language
Natural language specification

• Requirements are written as natural language sentences supplemented by diagrams and tables.
• Used for writing requirements because it is expressive, intuitive and universal. This means that the requirements can be understood by users and customers.
Problems with natural language

• Lack of clarity
  – Precision is difficult without making the document difficult to read.

• Requirements confusion
  – Functional and non-functional requirements tend to be mixed-up.

• Requirements amalgamation
  – Several different requirements may be expressed together.
Structured specifications

• An approach to writing requirements where the freedom of the requirements writer is limited, and requirements are written in a standard way
You have understood

- **Why** the system will be built, and
- For **whom** it is useful.

You have explained this in the Wiki.

- **what** is going to be built and have broken the functionality down into functional requirements

After reading the requirements,
I understand **who will be the users** of the application and **what will the user be able to do with the system**.
Example: Home Access Control

Objective: Design an electronic system for:

- Home access control
  - Locks and lighting operation
- Intrusion detection and warning
## Example NL Requirements

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Priority</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REQ1</td>
<td>5</td>
<td>The system shall keep the door locked at all times, unless commanded otherwise by authorized user. When the lock is disarmed, a countdown shall be initiated at the end of which the lock shall be automatically armed (if still disarmed).</td>
</tr>
<tr>
<td>REQ2</td>
<td>2</td>
<td>The system shall lock the door when commanded by pressing a dedicated button.</td>
</tr>
<tr>
<td>REQ3</td>
<td>5</td>
<td>The system shall, given a valid key code, unlock the door and activate other devices.</td>
</tr>
<tr>
<td>REQ4</td>
<td>4</td>
<td>The system should allow mistakes while entering the key code. However, to resist “dictionary attacks,” the number of allowed failed attempts shall be small, say three, after which the system will block and the alarm bell shall be sounded.</td>
</tr>
<tr>
<td>REQ5</td>
<td>2</td>
<td>The system shall maintain a history log of all attempted accesses for later review.</td>
</tr>
<tr>
<td>REQ6</td>
<td>2</td>
<td>The system should allow adding new authorized persons at runtime or removing existing ones.</td>
</tr>
<tr>
<td>REQ7</td>
<td>2</td>
<td>The system shall allow configuring the preferences for device activation when the user provides a valid key code, as well as when a burglary attempt is detected.</td>
</tr>
<tr>
<td>REQ8</td>
<td>1</td>
<td>The system should allow searching the history log by specifying one or more of these parameters: the time frame, the actor role, the door location, or the event type (unlock, lock, power failure, etc.). This function shall be available over the Web by pointing a browser to a specified URL.</td>
</tr>
<tr>
<td>REQ9</td>
<td>1</td>
<td>The system should allow filing inquiries about “suspicious” accesses. This function shall be available over the Web.</td>
</tr>
<tr>
<td>Identifier</td>
<td>Priority</td>
<td>Requirement</td>
</tr>
<tr>
<td>------------</td>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>REQ1</td>
<td>5</td>
<td>The system shall keep the door locked at all times, unless commanded otherwise by authorized user. When the lock is disarmed, a countdown shall be initiated at the end of which the lock shall be automatically armed (if still disarmed).</td>
</tr>
<tr>
<td>REQ2</td>
<td>2</td>
<td>The system shall lock the door when commanded by pressing a dedicated button.</td>
</tr>
<tr>
<td>REQ3</td>
<td>5</td>
<td>The system shall, given a valid key code, unlock the door and activate other devices.</td>
</tr>
<tr>
<td>REQ4</td>
<td>4</td>
<td>The system should allow mistakes while entering the key code. However, to resist &quot;dictionary attacks,&quot; the number of allowed failed attempts shall be small, say three, after which the system will block and the alarm bell shall be sounded.</td>
</tr>
<tr>
<td>REQ5</td>
<td>2</td>
<td>The system shall maintain a history log of all attempted accesses for later review.</td>
</tr>
<tr>
<td>REQ6</td>
<td>2</td>
<td>The system should allow adding new authorized persons at runtime or removing existing ones.</td>
</tr>
<tr>
<td>REQ7</td>
<td>2</td>
<td>The system shall allow configuring the preferences for device activation when the user provides a valid key code, as well as when a burglary attempt is detected.</td>
</tr>
<tr>
<td>REQ8</td>
<td>1</td>
<td>The system should allow searching the history log by specifying one or more of these parameters: the time frame, the actor role, the door location, or the event type (unlock, lock, power failure, etc.). This function shall be available over the Web by pointing a browser to a specified URL.</td>
</tr>
<tr>
<td>REQ9</td>
<td>1</td>
<td>The system should allow filing inquiries about “suspicious” accesses. This function shall be available over the Web.</td>
</tr>
</tbody>
</table>

’shall’: mandatory (?)

’should’: optional (?)
### Example NL Requirements

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Priority</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REQ1</td>
<td>5</td>
<td>The system shall keep the door locked at all times, unless commanded otherwise by authorized user. When the lock is disarmed, a countdown shall be initiated at the end of which the lock shall be automatically armed (if still disarmed).</td>
</tr>
<tr>
<td>REQ2</td>
<td>2</td>
<td>The system shall lock the door when commanded by pressing a dedicated button.</td>
</tr>
<tr>
<td>REQ3</td>
<td>5</td>
<td>The system shall, given a valid key code, unlock the door and activate other devices.</td>
</tr>
<tr>
<td>REQ4</td>
<td>4</td>
<td>The system should allow mistakes while entering the key code. However, to resist “dictionary attacks,” the number of allowed failed attempts shall be small, say three, after which the system will block and the alarm bell shall be sounded.</td>
</tr>
<tr>
<td>REQ5</td>
<td>2</td>
<td>The system shall maintain a history log of all attempted accesses for later review.</td>
</tr>
<tr>
<td>REQ6</td>
<td>2</td>
<td>The system should allow adding new authorized persons at runtime or removing existing ones.</td>
</tr>
<tr>
<td>REQ7</td>
<td>2</td>
<td>The system shall allow configuring the preferences for device activation when the user provides a valid key code, as well as when a burglary attempt is detected.</td>
</tr>
<tr>
<td>REQ8</td>
<td>1</td>
<td>The system should allow searching the history log by specifying one or more of these parameters: the time frame, the actor role, the door location, or the event type (unlock, lock, power failure, etc.). This function shall be available over the Web by pointing a browser to a specified URL.</td>
</tr>
<tr>
<td>REQ9</td>
<td>1</td>
<td>The system should allow filing inquiries about “suspicious” accesses. This function shall be available over the Web.</td>
</tr>
</tbody>
</table>

**Compound** REQ: How to test it?
### Example NL Requirements

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Priority</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REQ1</td>
<td>5</td>
<td>The system shall keep the door locked at all times, unless commanded otherwise by authorized user. When the lock is disarmed, a countdown shall be initiated at the end of which the lock shall be automatically armed (if still disarmed).</td>
</tr>
<tr>
<td>REQ2</td>
<td>2</td>
<td>The system shall lock the door when commanded by pressing a dedicated button.</td>
</tr>
<tr>
<td>REQ3</td>
<td>5</td>
<td>The system shall, given a valid key code, unlock the door and activate other devices.</td>
</tr>
<tr>
<td>REQ4</td>
<td>4</td>
<td>The system should allow mistakes while entering the key code. However, to resist &quot;dictionary attacks,&quot; the number of allowed failed attempts shall be small, say three, after which the system will block and the alarm bell shall be sounded.</td>
</tr>
<tr>
<td>REQ5</td>
<td>2</td>
<td>The system shall maintain a history log of all attempted accesses for later review.</td>
</tr>
<tr>
<td>REQ6</td>
<td>2</td>
<td>The system should allow adding new authorized persons at runtime or removing existing ones.</td>
</tr>
<tr>
<td>REQ7</td>
<td>2</td>
<td>The system shall allow configuring the preferences for device activation when the user provides a valid key code, as well as when a burglary attempt is detected.</td>
</tr>
<tr>
<td>REQ8</td>
<td>1</td>
<td>The system should allow searching the history log by specifying one or more of these parameters: the time frame, the actor role, the door location, or the event type (unlock, lock, power failure, etc.). This function shall be available over the Web by pointing a browser to a specified URL.</td>
</tr>
<tr>
<td>REQ9</td>
<td>1</td>
<td>The system should allow filing inquiries about “suspicious” accesses. This function shall be available over the Web.</td>
</tr>
</tbody>
</table>

**'Compound' REQ: How test it?**

For **REQ3**, the customer may suggest these test cases:

- Test with the valid key of a current tenant on his or her apartment (pass)
- Test with the valid key of a current tenant on someone else’s apartment (fail)
- Test with an invalid key on any apartment (fail)
- Test with the key of a removed tenant on his or her previous apartment (fail)
- Test with the valid key of a just-added tenant on his or her apartment (pass)
# Example User Stories

<table>
<thead>
<tr>
<th>Identifier</th>
<th>User Story</th>
</tr>
</thead>
<tbody>
<tr>
<td>ST-1</td>
<td>As an authorized person (tenant or landlord), I can keep the doors locked at all times. (<em>so I am safe</em>)</td>
</tr>
<tr>
<td>ST-2</td>
<td>As an authorized person (tenant or landlord), I want the lock be automatically locked after a defined period of time.</td>
</tr>
<tr>
<td>ST-3</td>
<td>As an authorized person (tenant or landlord), I can lock the doors on demand.</td>
</tr>
<tr>
<td>ST-4</td>
<td>As an authorized person (tenant or landlord), I can unlock the doors. (<em>Test: Allow a small number of mistakes, say three.</em>)</td>
</tr>
<tr>
<td>ST-5</td>
<td>As a landlord, I can at runtime manage authorized persons.</td>
</tr>
<tr>
<td>ST-6</td>
<td>As an authorized person (tenant or landlord), I can view past accesses.</td>
</tr>
<tr>
<td>ST-7</td>
<td>As a tenant, I can configure the preferences for activation of various devices.</td>
</tr>
<tr>
<td>ST-8</td>
<td>As a tenant, I can file complaint about “suspicious” accesses.</td>
</tr>
</tbody>
</table>

Note: ‘Why’ part is missing in the examples above.
User Stories

As a tenant, I can unlock the doors to enter my apartment.

- user-role (benefactor)
- action (functionality)
- business-value (goal)

• Similar to NL requirements, but focus on the user benefits, instead on system characteristics (alone).

• Unfortunately, third element (business-value, goal) is often ommitted

• Preferred tool in agile methods.
NL Requirements vs. User Stories

• Traditional requirement – “shall” statements:
  • “The system shall provide a user configurable interface for all user and system manager functions”
  • “The user interface shall be configurable in the areas of:
    • Screen layout
    • Font
    • Background and text color

• Corresponding “User Story”:
  • “As a system user or system manager, …
  • … I want to be able to configure the user interface for screen layout, font, background color, and text color, …
  • … so that I can use the system in the most efficient manner”
Kasutuslood (user stories)

• Üldkuju: As a user playing some role, I must be able to perform some activities [in order to achieve some goal]
User Story

• <Actor/Role>  As a user
  ...
• <Action>     I want to narrow down people search results by location
  ...
• <Goal>       so I can find the right person more quickly

Acceptance test:

Given I am on the search screen
And ’Paula’ is on the same indexed page with ’Tartu’
When I search for ’Paula’
Then I see ’Tartu’ in the location section of the search results
Agile Models

**Basic Philosophy**
- 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!
- Respond to the customer
  - Rather than focusing on the contract

**Weaknesses**
- Relies on programmer’s memory
  - Code can be hard to maintain
- Relies on oral communication
  - Mis-interpretation possible
- Assumes single customer representative
  - Multiple viewpoints not possible
- Only short term planning
  - No longer term vision

---

**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
- **Write test cases before code**
- **The program code is the design doc**
  - Can also use CRC cards (Class-Responsibility-Collaboration)
- **Continuous Integration**
  - Integrate and test several times a day

*Source: Adapted from Nawrocki et al, RE ’02*
Extreme Programming

- Collect User stories
- Code
- Integrate
- Test
- Release
- Plan
- Write test cases

Each cycle: approx 2-3 weeks
Scrum

The Agile: Scrum Framework at a glance

Inputs from Executives, Team, Stakeholders, Customers, Users

- Product Owner
- The Team
- Sprint Backlog
- Sprint Planning Meeting
- Task Breakout
- Burndown/up Charts
- Scrum Master
- Daily Scrum Meeting
- 1-4 Week Sprint
- Sprint end date and team deliverable do not change
- Sprint Review
- Finished Work
- Sprint Retrospective

Ranked list of what is required: features, stories, ...

Team selects starting at top as much as it can commit to deliver by end of Sprint

AGILE FOR ALL
Potentially Shippable Product Increment

- Product Backlog
- Sprint Backlog
- Sprint Planning
- Daily Scrum
- 1–4 Week Sprints
- Sprint Review
- Sprint Retrospective
- Potentially Shippable Product Increment
Product Backlog

- Product Backlog
- Sprint Backlog
- Sprint Planning
- Daily Scrum
- 1–4 Week Sprints
- Sprint Review
- Sprint Retrospective
- Potentially Shippable Product Increment
## Product Backlog näide

<table>
<thead>
<tr>
<th>As a</th>
<th>I want to</th>
<th>So that (I can)</th>
<th>Business Value</th>
<th>Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>HR Manager</td>
<td>Publish new vacancies</td>
<td>Find candidates</td>
<td>80</td>
<td>20</td>
</tr>
<tr>
<td>Job Hunter</td>
<td>Apply for a job</td>
<td>Quickly apply for a job</td>
<td>80</td>
<td>40</td>
</tr>
<tr>
<td>HR Manager</td>
<td>Triage applicants</td>
<td>Politely eliminate unpromising candidates</td>
<td>50</td>
<td>8</td>
</tr>
<tr>
<td>Googlebot</td>
<td>effectively find and index all postings</td>
<td>Ensure that internet searchers can find job postings on this site</td>
<td>50</td>
<td>13</td>
</tr>
<tr>
<td>System Admin</td>
<td>quickly recognize and analyze system</td>
<td>ensure rapid resolution of technical problems</td>
<td>30</td>
<td>20</td>
</tr>
</tbody>
</table>
Sprint Backlog

- Product Backlog
- Sprint Planning
- Daily Scrum
- 1-4 Week Sprints
- Sprint Review
- Sprint Retrospective
- Potentially Shippable Product Increment
### Sprint Backlog näide

<table>
<thead>
<tr>
<th>User Story</th>
<th>Tasks</th>
<th>Day 1</th>
<th>Day 2</th>
<th>Day 3</th>
<th>Day 4</th>
<th>Day 5</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>As a member, I can read profiles of other members so that I can find someone to date.</td>
<td>Code the ...</td>
<td>8</td>
<td>4</td>
<td>8</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>As a member, I can read profiles of other members so that I can find someone to date.</td>
<td>Design the ...</td>
<td>16</td>
<td>12</td>
<td>10</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>As a member, I can read profiles of other members so that I can find someone to date.</td>
<td>Meet with Mary about ...</td>
<td>8</td>
<td>16</td>
<td>16</td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>As a member, I can read profiles of other members so that I can find someone to date.</td>
<td>Design the UI</td>
<td>12</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>As a member, I can read profiles of other members so that I can find someone to date.</td>
<td>Automate tests ...</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>As a member, I can read profiles of other members so that I can find someone to date.</td>
<td>Code the other ...</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>As a member, I can update my billing information.</td>
<td>Update security tests</td>
<td>6</td>
<td>6</td>
<td>4</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>As a member, I can update my billing information.</td>
<td>Design a solution to ...</td>
<td>12</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>As a member, I can update my billing information.</td>
<td>Write test plan</td>
<td>8</td>
<td>8</td>
<td>4</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>As a member, I can update my billing information.</td>
<td>Automate tests ...</td>
<td>12</td>
<td>12</td>
<td>10</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>As a member, I can update my billing information.</td>
<td>Code the ...</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Nõuete valideerimise tehnikad

• Requirements reviews
  – Systematic manual analysis of the requirements.

• Prototyping
  – Using an executable model of the system to check requirements.

• Test-case generation
  – Developing tests for requirements to check testability.
Prototüüpimine (Wireframe'i näide)
Prototüüpimise plussid

• Lahendus mõeldakse detailides läbi
• Lõppkasutaja saab „proovida“ funktsionaalsust enne realisatsiooni
• Tellijal ja täitjal ühine nägemus lõpptulemusest
• Tellija ja täitja saavad täpsemalt kokku leppida projekti skoobis ning vaheetulemites
• Saab eraldada, mis on lihtsam ja mis keerulisem funktsionaalsus
• Selgemalt saab eraldada, mis on muudatus, mis on puudujääk ja mis täitja viga
• Selgem ülevaade kui palju projektist valmis on
HARJUTUS (MAX 3 PUNKTI)