Individual Project

Agnieszka Jastrzębska
Władysław Homenda
Lucjan Stapp
Target:

1. Improvement of software development skill
2. Introduction to industrial method of building application in practical way
<table>
<thead>
<tr>
<th>Date</th>
<th>Day</th>
<th>Lab</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2016-02-25</td>
<td>Thursday</td>
<td>Lab 1</td>
<td>Start, theme’s offer, preliminary choice</td>
<td>till Monday 8:00 AM final decision by e-mail</td>
</tr>
<tr>
<td>2016-03-03</td>
<td>Thursday</td>
<td>Lab 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2016-03-10</td>
<td>Thursday</td>
<td>Lab 3</td>
<td>requirement specification</td>
<td>⇒ technical</td>
</tr>
<tr>
<td>2016-03-17</td>
<td>Thursday</td>
<td>Lab 4</td>
<td>requirement specification - second term</td>
<td>for LS group Thursday 24</td>
</tr>
<tr>
<td>2016-03-24</td>
<td>Thursday</td>
<td>Lab 5</td>
<td>requirement specification - second term – LS group</td>
<td></td>
</tr>
<tr>
<td>2016-04-07</td>
<td>Thursday</td>
<td>Lab 6</td>
<td>technical project</td>
<td></td>
</tr>
<tr>
<td>2016-04-14</td>
<td>Thursday</td>
<td>Lab 7</td>
<td>technical project - second term</td>
<td></td>
</tr>
<tr>
<td>2016-04-21</td>
<td>Thursday</td>
<td>Lab 8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2016-04-28</td>
<td>Thursday</td>
<td>Lab 9</td>
<td>code of modules</td>
<td></td>
</tr>
<tr>
<td>2016-05-05</td>
<td>Thursday</td>
<td>Lab 10</td>
<td>version 0.98</td>
<td></td>
</tr>
<tr>
<td>2016-05-12</td>
<td>Thursday</td>
<td>Lab 11</td>
<td>version 0.99</td>
<td></td>
</tr>
<tr>
<td>2016-05-19</td>
<td>Thursday</td>
<td>Lab 12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2016-06-02</td>
<td>Thursday</td>
<td>Lab 13</td>
<td>version 1.0</td>
<td>⇒testing</td>
</tr>
<tr>
<td>2016-06-09</td>
<td>Thursday</td>
<td>Lab 14</td>
<td>test report</td>
<td></td>
</tr>
<tr>
<td>2016-06-16</td>
<td>Thursday</td>
<td>Lab 15</td>
<td>acceptance, final grades</td>
<td></td>
</tr>
</tbody>
</table>
Individual Project

Rules of classification:

- Active attendance on all red terms
  - We will accept ONE absence/ non preparation ⇒ extra term (week later if possible)
  - Each next absence / non preparation -5 points
- Business analysis to 15 points
- Technical analysis to 15 points
- Application to 55 points
- Tests to 15 points

To pass >= 51 points
Individual Project

The application should be written in
1. C# - IDE Visual Studio
2. JAVA - IDE Eclipse

The application **should work** in our department (MaIS) laboratory:
- after installing on network disk;
- no virtual machines;
- if additional library will be needed, special permission is required.
Each document contains:

1. Title page:
   1. Document title;
   2. Author (s);
2. Information about document:
   1. Document metrics;
   2. History of changes;
3. Table of contents (for documents > 5 pages);
4. Summary („Summary for management“);
5. Appropriate content;
6. Ending part;
7. **Good principle:** page enumeration: page $XX/YY$. 

---

**Document form (1/3)**
## Document metric

<table>
<thead>
<tr>
<th><strong>Document metric</strong></th>
<th><strong>Project name</strong></th>
<th><strong>Company:</strong></th>
<th><strong>WUT</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Project</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Name:</strong></td>
<td>Document name</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Topics:</strong></td>
<td>Document topics</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Author:</strong></td>
<td>John Smith, George Wilson</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>File:</strong></td>
<td>File name</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Version no:</strong></td>
<td>XX.YY</td>
<td>Status: Working / final</td>
<td>Opening date: 2015-02-01</td>
</tr>
<tr>
<td><strong>Summary:</strong></td>
<td>The main purpose of the document (one sentence)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Authorized by</strong></td>
<td></td>
<td>Last modification date: 2015-02-26</td>
<td></td>
</tr>
</tbody>
</table>
## History of changes

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Who</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>2015-02-01</td>
<td>John Smith</td>
<td>Definition of the main purpose of the document</td>
</tr>
<tr>
<td>02</td>
<td>2015-02-14</td>
<td>John Smith</td>
<td>Version 01</td>
</tr>
<tr>
<td>03</td>
<td>2015-02-26</td>
<td>George Wilson</td>
<td>Modification of § 2</td>
</tr>
</tbody>
</table>
Models for software development

**Programming process** – a kit of activities, methods and transformation, used by computer scientists to develop and maintain software and other allied products connected with software such as:

- plan and schedule,
- project,
- code,
- test sets,
- user textbooks, etc.

For better understands and usage the above process one use **models**.
Base idea of **waterfall model**: project process is realized sequentially. The next phase starts after completion and termination the previous one.

Typical solution for “good engineering practice”, hence the other name of the model – **classical model**.

We assume that results of previous phases have to be revised and corrected.
Waterfall Model

- Requirements Specification = Business analysis
- Technical Project
- Implementation
- Testing
- Launch
- Maintenance
Waterfall Model

One can distinguish following phases in waterfall model:

- Need’s analysis;
- System specification;
- Project preparation;
- Programming;
- Integration;
- Testing;
Waterfall Model

- Need analysis
- System specification
- Project preparation
- Programming
- Integration
- Testing

analysis phase
Artifacts

There are four basic sets of artifacts:

1. **Requirement set** (or requirement specification or business analysis):
   defines, what the system under consideration should do – in detailed way.

2. **Project set** (or technical project):
   defines, how the system should be built.

3. **Implementation set**:
   includes module code and defines how the system is integrated from previously built components.

4. **Deployment set**:
   delivers all needed information about configuration of the prepared system.
Artifacts

Requirement set (requirement specification)
The purpose this document is to create a model of the system's functional requirements that is independent of implementation constraints. All information, which defined functionality of the system (what system should do) are included in this set.
Artifacts

Requirement set (requirement specification)
Also the model of functional and nonfunctional use cases, the domain model, analyze model and other elements such as mockup, GUI prototype (style guide) and limitations of the law make up the requirement set.

Requirements should be as detailed as possible but independent of technical details.
Artifacts

Project set (technical project)

All the data that describes how the system should be built is included in this set. Especially information like decisions concerning methods with which the system is constructed and the usage of previously prepared components. Also, budget and time limitations as well as quality criteria are present in the technical project.

The project and test models in addition to other elements describing the system's nature are also necessary - especially prototypes.
Artifacts

Implementation set

All data concerning the components of the software being built is included in the implementation set. In particular the following elements are included:

- Source code in different programming languages,
- Configuration files,
- Data files,
- Programming components,
- Instructions describing how system should be integrated in productive environment.
Artifacts

Deployment set

The information on when and how the software being built will be installed and launched on the production environment is included in this set.
**Business analysis**

**Business Analysis** is the set of tasks, knowledge, and techniques required to identify business needs and determine solutions to business problems.

Solutions often include a systems development component, but may also consist of process improvement or organizational change.

**Business Analysis** includes the requirement specification.
In engineering, a **requirement** is a singular documented need of what a particular product or service should be or do


Requirement = **what** computer system should do and **how**.
Business analysis

Business analyst (= requirement analysts) wants to make sure that he/she define the application in a way that meets the end-users’ needs. Essentially, he/she wants to define the right application.

This means that he/she must document the right requirements through listening carefully to ‘customer’ feedback, and by delivering a complete set of clear requirements to the technical architects and coders who will write the program.
Business analysis

If a business analyst has limited tools or skills to help her/him elicit the right requirements, then the chances are fairly high that she/he will end up documenting requirements that will not be used or that will need to be re-written – resulting in rework as discussed above.

The time wasted to document unnecessary requirements not only impacts the business analyst, it also impacts the rest of the development cycle.
Business analysis

Experts estimate that 10% to 40% of the features in new software applications are unnecessary or go unused. Being able to reduce the amount of these extra features by even one-third can result in significant savings.

Based on http://en.wikipedia.org/
Business analysis

Object-oriented modeling is a common approach to modeling applications, systems, and business domains.

Object-oriented modeling typically divides into two aspects of work: the modeling of dynamic behaviors like business processes and use cases, and the modeling of static structures like classes and components.

In business analysis one can limited to dynamic behavior of the application, static structure is typically in technical analysis.
**Business analysis**

**User story**: the modern tool to describe future customers needs.

A user story is one or more sentences in the everyday or business language of the end user that captures what a user does or needs to do as part of his/her job function.
**User story**: the modern tool to describe future customers needs.

It captures the 'who', 'what' and 'why' of a requirement in a simple, concise way, often limited in detail by what can be hand-written on a **small paper** notecard.

User stories are written by or for the business user as that user's primary way to influence the functionality of the system being developed.
User Story: the basis of requirements

- Elements of User Story
  - As... (concrete type of user)
  - I want... (problems to be solved)
  - because... (desired results).

- Definition of user satisfaction criteria
  - Typically as set of acceptance tests (Definition of Done)
**User Story:** the basis of requirements:

Examples:

As a [customer], I want to get [a new popup window to appear with the explanation of how to complete the registration form], so that I can [register to the conference without forgetting fields].

As a [user closing the application], I want to be [prompted to save if I have made any change in my data since the last save], so I am sure [I do not lost my data]
User story - properties:

One may use the INVEST technique:

- **I**ndependent
- **N**egotiable
- **V**aluable
- **E**stimable
- **S**ized appropriately
- **T**estable

Bill Wake, INVEST in Good stories, and SMART Tasks
Business analysis

User Stories
Provide a small-scale and easy-to-use presentation of information. Are generally formulated in the everyday language of the user and contain little detail, thus remaining open to interpretation. They should help the reader understand what the software should accomplish. Must be accompanied by acceptance testing procedures (acceptance criteria) for clarification of behavior where stories appear ambiguous.

Use Cases
Describe a sequence of interactions, and may be worded in terms of a formal model. A use case is intended to provide sufficient detail for it to be understood on its own. A use case has been described as “a generalized description of a set of interactions between the system and one or more actors, where an actor is either a user or another system”. May be delivered in a stand-alone document.
Business analysis

For us, business analysis is a document presented:

- All requirements for application
- The main user paths
- The basic sources of needed data

Useful tool: use-case model or user stories

The client will only obtain what was described in the requirement specification (mostly in the scope of the functionality of the build system).
Obscurity in requirements:
- conceptual (e.g. Month report what means month?)
- dictionary
  - Correspondence: 2 meanings
- Business jargon / techspeach
  - „contra–entry in red”,
  - DoW: DoW pilot = stewardess or Description of Work
- Informatics techspeach
Business analysis

Non functional requirements – **how** the system should work

- Usability
- Reliability
- Performance
- Security and safety
-Effectiveness
Business analysis

FURPS division

- Functionality
- Usability
- Reliability
- Performance
- Security and Safety
Business analysis

FURPS

- **Usability**: The capability of the software to be understood, learned, used and be attractive to the user when used under specified conditions. [ISO 9126] (usually understood as user friendliness).

- **Reliability**:
  - Typical measures:
    - MTBF - Mean Time Between Failure;
    - Maximal number of hours in a month, when a system can be out of action for maintenance reasons (important for systems working in 7*24 mode).

- **Performance**:
  - Typical measures:
    - Number of transactions in a time unit.
    - Number of users working in parallel with given average reaction time.

- **Security**:
  - Authorization (logins and passwords),
  - Access rights,
  - Firewalls.
Business analysis

FURPS

Importance of non functional requirements.
Typically non functional requirements are not articulated in an use case model.

Hence we add **supplementary specification**.

**Even though** it is called **supplementary**, this specification is absolutely necessary in the requirements description of a system.
Business analyses

Typical business analysis contains:

- Notion description,
  - Use case model,
  Or
  - Set of user stories,
  Or
  - System description in natural language
- Supplementary specification.
Technical project

- System architecture,
- Full description of used algorithms,
- User style guide – if needed,
- Class project.
A cellular automaton consists of a regular grid of **cells**, each in one of a finite number of **states**. The grid can be in any finite number of dimensions. For each cell, a set of cells called its **neighborhood** is defined relative to the specified cell. An initial state is selected by assigning a state for each cell.
Cellular automaton

- A new *generation* is created according to some fixed *rules* that determines the new state of each cell in terms of the current state of the cell and the states of the cells in its neighborhood.

- Typically, the rule for updating the state of cells is the same for each cell and does not change over time, and is applied to the whole grid simultaneously.
Cellular automaton

- There are many rules for updating the states.
- Some of them are listed below:
  - Wireworld
  - Langton's ant
  - Brian's Brain
  - “Life” – many versions
Project

Cellular automaton

In net one can find many working programs for cellular automata e.g.

- *Mirek celebration*,
- *Alan Hensel's Life applet*,
- *Caffeine*

Please **DO NOT copy** them as your own.
Project

- Your application should:
  - Have to work in 24 points neighborhood
  - Have possibility to introduce rules which determine the new state for each cell
Project

Your application should:

- Have possibility to introduce rules which determine the new state for each cell

but for 24 points neighborhood there is max. $2^{24} (16\ 777\ 216)$ rules

HENCE

we need more **smart** ideas to introduce rules:

e.g.

If there is at least k living cells in neighborhood than...

or

if there is 3 dead cells in first row than ...
You proposed your rules in general, but rules can be introduced by your applications user.

Sometimes rules can be contrary between themselves, your application should propose an algorithm to solve this problem.

With application you should prepare a few working examples:
- set of rules
- starting state
Project

- Your application should:
  - Have possibility to change—dynamically—the rule which was previously introduced to the system
  - Have possibility to observed grid states step by step or after n steps (where n is introduced by user)
    - If this new view is beyond the screen, there is a possibility to observe the whole system of cells
  
- Number of colors— as you wish; may be black and white or multicolored.
**Decision**

- Today – till the end of class (5:00 PM)

- or

- till Monday 8:00 AM by e-mail to your supervisor

**Priorities:** according to declaration.

1. 1/3 of students will realized project under supervising of prof. W. Homenda

2. 1/3 of students will realized project under supervising of dr L. Stapp

3. 1/3 of students will realized project under supervising of Ms. A Jastrzębska