# Assignment 4

## Implementing requirements

It is high time you implemented the requirements. During this lesson, we will focus on one requirement only. You will implement it in two-person groups. So, divide yourself into small groups. If you are one-person group, you will get separate instruction when it is relevant.

All the groups choose **ONE AND THE SAME REQUIREMENT ITEM**! Make sure that this requirement ITEM is not too time-consuming and make sure that everybody understands what the requirement means. In the former phase, you probably divided one requirement into lower-level requirements. You may very well choose the lower-level requirement in this phase, and probably, you will have to further divide it into tasks, if required.

It is important that everybody has the same understanding of the scope and risks of the requirement. All the misunderstanding and other problems should be resolved at this stage. So, before starting working in groups, do the following:

- Estimate the effort required for implementing the requirement item. You already did it during Assignment 2. If not, then do it now. If yes, check it and revise it if necessary.
- *Risk identification and practice*. Mira's simple suggestion for a risk management practice is to be found at the end of this document.

After you have agreed upon the effort and risks, start working in your individual groups. **NO GROUP SHOULD COMMUNICATE WITH ANOTHER GROUP during this time!!** While implementing the requirement, use the following practices:

- *Design practice*. Mira's simple suggestion for a design practice is to be found at the end of this document.
- *Test-first programming.* Design test cases first before writing code for them. The test cases may either be implemented in Java or written on a piece of paper.
- *Pair programming practice*. Make sure that you change roles every x minutes and that you follow the pair programming rule of conduct as defined on lecture slides.
- *Refactoring*. If you write on a new piece of code then I advise you to end your programming with refactoring. If you, on the other hand, continue working on the existing code, then I advise you to check whether refactoring is needed first.

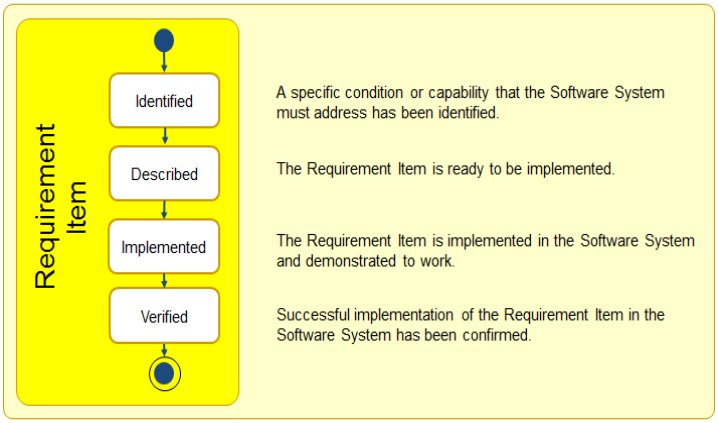
**Commented [MMK1]:** Agree upon time intervall. No longer than 20 minutes!

While doing your work, follow the status states of the requirement item as defined by SEMAT and shown below. After all the groups have coded, they should meet and compare their work and exchange their experiences. Together, agree upon the following:

- Your experience with pair programming
  - List good sides
    - Anything that you felt was fantastic?

    - List bad sides
        - Anything that did not work?
    - If you are one-person group, then you make literature study to find out good and bad sides of pair-programming using at least two references. At least two bad sides and at least three good sides should be listed.
- Your experience with test-first programming
    - List good sides
    - List bad sides
- Your experience with refactoring
    - List good sides
    - List bad sides
- Your experience with designing the code first before coding
    - List good sides
    - List bad sides
- Your ability to estimate the effort required for implementing the requirement
    - Compare your estimated effort in Assignment 2 to the real one. What was the difference?
- Document all the results of your work.

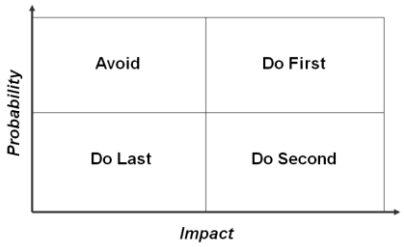
## Requirements Item



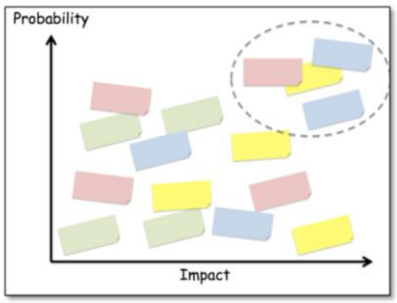

## Risk identification and analysis practice

- Identify risks that might hinder you from achieving your implementation goal with respect to time, financial constraints or other.
- Use collaborative agile risk management game. By following the round-the-table routine, each team member should write a risk on a postit note and place it within one of the square areas. If a team member does not have any idea for a new risk, then he does not agree with the placement of the already identified risk, she/he is allowed to move it to the square that she/he believes is more appropriate for the risk at hand. If you are one-person group, you simulate that you are at least two people with different views of the risks.

**Commented [MMK2]:** When doing it, check how much effort you estimated for the requirement. Is it enough? Here, you might find a risk.




**Collaborative agile risk mngt game**




## Design practice

- Divide the requirement into tasks
- Agree together on the goal of the task/requirement
- Identify data that needs to be managed
- Identify major steps to be processed
- Using your own graphical method,
    - Identify classes/objects and their responsibilities
    - Define how they are going to communicate with one another
- Using your own graphical or textual method, identify the flow of execution. For this, you may use pseudo code, textual description of the steps, such as for example:
    - Initialize NumberOfStudents to Zero
    - Input Age
    - If Age > 56
        - Print “Sorry, you are too old, we cannot accept you to the course”
    - Else
        - Print “Du är välkommen att studera på KTH”.
    - Continue with the next program steps here.