

User REquirements Specification



Version #2

Nikola Nikushev

Bilger Yahov

Lyubomir Dimov

Tao Hua

Table of Contents

[Introduction 2](#_Toc436398449)

[Functional requirements (use-cases) 2](#_Toc436398450)

[1. Create a component 2](#_Toc436398451)

[2. Remove a component 2](#_Toc436398452)

[3. Add a pipeline 3](#_Toc436398453)

[4. Remove a pipeline 3](#_Toc436398454)

[5. Editing a component 3](#_Toc436398455)

[6. Clearing the grid 4](#_Toc436398456)

[7. Saving a grid 4](#_Toc436398457)

[8. New file 5](#_Toc436398458)

[9. File open 5](#_Toc436398459)

[10. Undo last change 5](#_Toc436398460)

[User interface 6](#_Toc436398461)

[Non-functional requirements 6](#_Toc436398462)

# Introduction

User Requirements Specification document introduces you to the requirements which our application will meet and the features it will possess. Every software application which is to be developed needs to meet two kinds of requirements – functional and non-functional.

We described the functional requirements in a way of using use-cases. Use-cases can be very helpful describing the flow of actions between the user and the system. Below you can see 10 use-cases. They are described in a structured and neat way. Each use-case has its own extensions below the Main Success Scenario.

Non-functional requirements of an application have to deal with its quality aspects. We briefly emphasized on most common non-functional requirements. We gave brief examples about accessibility, efficiency, maintainability, usability and Reusability.

# Functional requirements (use-cases)

## Create a component

Goal level: Sea level

Pre: No

Actor: User

Main Success Scenario:

1. User clicks on a component tool from the toolbox
2. System highlights the tool
3. User clicks on a spot on the grid
4. System places the component on the desired position on the grid.

Extensions:

3a. if the newly created component overlaps an old one.

* 3a.1 the system shows a message informing the user.
* The user can choose either to place the component on a valid place or to release the element.(look at use case “Releasing a component”)

4a. If the component has flow capacity

* The system shows the flow of the component

## Remove a component

Goal level: Sea level

Pre: There is at least one component on the work space (grid)

Actor: User

Main Success Scenario:

1. User selects removing a component tool from the toolbox.
2. System highlights the tool
3. User clicks on a component on the grid
4. System deletes the element.

Extensions:

3a.There are connected pipelines to the component

* The user is shown a message to confirm that when deleted the pipelines connected to the component will be deleted.

1. If the user clicks on “confirm”
   * System will delete the component and the connected pipelines.
2. If the user clicks on “cancel”
   * System closes the message box

3b. The user clicks on a pipeline from the grid

* See “Removing a pipeline”

## Add a pipeline

Goal level: Sea level

Pre: There is at least two components on the grid

Actor: User

Main Success Scenario:

1. User clicks on the pipeline tool from the toolbox.
2. System highlights the tool.
3. User clicks on a output of a component
4. User clicks on the grid to make a path for the pipeline
5. System draws the pipeline on each click on the grid
6. User clicks on an input of a component
7. System draws the last part of the pipeline
8. System updates the flow of the input component

Extensions:

3a. if the output is not free

* The system shows a message informing the user that this is not a valid output.

4a. if the user clicks on an element from the toolbox the operation is aborted.

6a. if the input is not free

* The system shows a message informing the user that this is not a valid input

## Remove a pipeline

Goal level: Sea level

Pre: There is at least two components on the grid which are connected with a pipeline

Actor: User

Main Success Scenario:

1. User clicks on delete component tool from the toolbox
2. System highlights the tool.
3. User clicks on a pipeline from the grid
4. The system deletes the pipeline
5. The system updates the flow of the input component

Extensions:

3a. if the user clicks on the form

* Nothing happens

3b. if the user clicks on a component

* Go to “Remove a component”

## Editing a component

Goal level: Sea level

Pre: There is a component on the grid

Actor: User

Main Success Scenario:

1. The user clicks on the editing properties tool from the toolbox.
2. The system highlights the tool.
3. User clicks a component that is not a pipeline from the grid.
4. The system highlights the component.
5. The system enables the appropriate properties of the component under the toolbox.
6. The user edits the component properties
7. The user clicks on “Update” button
8. The system updates the component

Extensions:

1a. If the user selects a pipeline

* Nothing happens

5a. If the user has specified invalid settings

* A message is shown that the user has specified invalid settings for the component
* The system does not apply the changes

## Clearing the grid

Goal level: Sea level

Pre: There are some components on the grid

Actor: User

Main Success Scenario:

1. User opens menu

2. User clicks on clear

3. User is shown a confirmation dialog to confirm that they wish to clear the grid.

4. User presses “Yes”.

3. System clears the grid

Extensions:

4a.1 User presses “No”.

4a.1 System aborts the operation.

## Saving a grid

Goal level: Sea level

Pre: There are some components on the grid

Actor: User

Main Success Scenario:

1. User opens menu

2. User clicks on save file

3. System shows save file dialog

4. User specifies file name and location

5. System saves the grid

Extensions:

4.а User does not specify name or location.

4.а.1 The system shows an error message informing the user

## New file

Goal level: Sea level

Actor: User

Main Success Scenario:

1. User presses the button “new file”
2. System shows the new grid

Extensions:

1.а If there are not saved changes

* System shows “Save file” dialog – See use-case “Saving a grid”
* If the user clicks cancel, the dialog closes and the system discards the changes.

## File open

Goal level: Sea level

Actor: User

Main Success Scenario

1. User presses the button to open file

2. System displays file explorer

3. User selects the needed file

4. User presses “Ok” button

5. System opens the selected file

Extensions:

5a: It is unable to open selected file (wrong format or file was damaged)

5a.1: System displays message that file cannot be opened

5b. if there are unsaved changes

* System shows “Save file” dialog – See use-case “Saving a grid”
* If the user clicks cancel, the dialog closes and the system discards the changes.

## Undo last change

Goal level: Sea level

Actor: User

Pre: At least one modification had been made

Main Success Scenario:

1. User clicks on undo button

2. System depict the network by one action backwards.

Extensions:

1a. if there are no changes

* The undo button is disabled

Note: The user can click on the button a maximum of 5 times, or until no changes are found and the button is disabled.

# User interface

# Non-functional requirements

Of course when using an application the things that can bother us or make us happy are not always related to the product’s functionality. What about Accessibility, Efficiency, Maintainability, Reusability, Usability?

1. Everything that you will ever need creating flow diagrams or systems, you can easily navigate to through the toolbox. The toolbox is placed on the left side of the working space. One point for accessibility!
2. You do not want to lose your time drawing lines? “Flowly” does it for you! Just pick a component from the toolbox and WISELY place it on the grid! One point for efficiency!
3. You hardly ever can get an error while working with “Flowly”. Everything on the software is properly set-up! If you get an error the System will not give the possibility for breakdown! One point for maintainability!
4. Do you know that you can safe and load/open flow systems? Yes you can with “Flowly”. You can work on your project whenever you want! One point for Reusability!
5. Summing it all our application is easy to use, it has friendly User Interface, it is error-prone and really efficient to use! One point for usability!
6. The application is based on windows forms and in matter of usability it runs over windows 7, 8 and above.