|  |
| --- |
| Fontys university of applied sciences |
| Process report |
| Final version |
|  |
|  |
| **1/10/2016** |

|  |
| --- |
|  |

*Rosen Danev*

*Preslav Gerchev*

*Dimitar Vikentiev*

*Monica Stoica*

Table of Contents

[Introduction 2](#_Toc440215530)

[Rules and decisions 3](#_Toc440215531)

[Division of work 4](#_Toc440215532)

[Problems and solutions 6](#_Toc440215533)

[Individual part 8](#_Toc440215534)

[Dimitar 8](#_Toc440215535)

[Monica 8](#_Toc440215536)

[Rosen 8](#_Toc440215537)

[Preslav 8](#_Toc440215538)

# Introduction

Our group consists of four members: Rosen Danev, Preslav Gerchev, Dimitar Vikentiev and Monica Stoica, all part of class EI6S3.

# Rules and decisions

To begin withl, we decided that the user will have a limited working space. This working space will be designated by a non-resizable panel. When an element has to be drawn on the screen, the user selects the point and the system will calculate the center of the element (picture) and draw it on the scree. The elements will be represented by pictures while the pipeplines will be drawn as a lines.

To select a component, the user has to double click on it and a red rectangle will be drawn around it.

When drawing one of the elements on the screen we decided that the system will register the X and Y coordinates as the top-left corner of the figure. However, this does not apply for pipelines as well.

By default, all elements have a current flow and a capacity of 0.

Both adjustable and normal splitter will have a default value of 50% on both outputs.

To adjust the percentage of an adjustable splitter, the user double clicks on the drawn splitter and a track bar will be displayed.

The current and the maximum flow of any element cannot be negative numbers and the current flow must be a lower or an equal number to the maximum flow. Otherwise, the system will not register the changes. The flow will be registered as a double and not int because, for example, if we want to connect pipeline with a flow of 7 to a normal splitter, we will have the out coming flow of 3 and 3. Therefore, 1 part will be lost.

The pipeline flow will be shown on the middle of the line.

To make sure that there will not be compatibility problems when loading and existing network, the file will be saved in binary format (.XML

We did not implement an algorithm for overlapping lines because a line can be drawn by selecting multiple points on the screen (creating a path). Therefore, the user is responsible if the lines will overlap or not.

# Division of work

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nr | Task | Group | Dimitar | Monica | Preslav | Rosen |
|  | User Requirements Specifications |  |  |  |  |  |
|  | Design document |  |  |  |  |  |
|  | Process report |  |  |  |  |  |
|  | User interface |  |  |  |  |  |
|  | Class implementations |  |  |  |  |  |
|  | Serialization |  |  |  |  |  |
|  | Functionality |  |  |  |  |  |

# Problems and solutions

One of the problems that we encountered was deleting a line. So that a line will be deleted, it has to be selected first. We tried drawing the line in a rectangle but this would work only if the line is perfectly straight. So this wasn’t an option- we had to come up with a way of selecting a line regardless of its position.

What we came up with was using Graphics Path – given a set of points, it could detect whether a mouse click occurred inside. The rest was simply putting that pipeline into a variable and then drawing it with another colour to give the user feedback that he had selected a pipeline.

Another problem related to pipelines was text location – we had to show the current flow that was passing through the pipeline. However a pipeline could have 2(just start and end) or many (including points in between) points -so how could we give a good text location for the pipeline?

Our solution was to first check if there were any in between points-if so we would take the one in the middle of the list and return it as a point for drawing text. If there were no points in between we would simply create one based off the starting and the ending point.

Connecting those pipelines to components was also a problem – what if the user had clicked at the end of the pump and then at the end of the sink? The line would have been very awkward. Same goes for the rest of the components – the mergers and the splitters. So we came up with pre-fixed locations for connecting pipelines. For pumps it was just at the end and for sinks- just at the start. Creating those for splitter and merger was a bit harder since we had to create not 1 but 3 – at the end it all worked out pretty well.

**Changes**

Since handing in the last version of User requirements specifications and design documents we have made the following changes:

* Updated use cases number II, IV, VI,
* The user interface
* Class diagram and description of the members
* Updated the rules and decisions section
* Sequence diagrams

# Individual part

### Dimitar

### Monica

### Rosen

### Preslav