CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

# Using machine learning to evaluate the layout quality of UML class diagrams

Master's thesis in Computer science and engineering

GUSTAV BERGSTRÖM

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2021

Master's thesis 2021

# Using machine learning to evaluate the layout quality of UML class diagrams

GUSTAV BERGSTRÖM

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2021

Using machine learning to evaluate the layout quality of UML class diagrams
GUSTAV BERGSTRÖM



Supervisor: Michel Chaudron, Department of Computer Science and Engineering
Examiner: Jan-Philipp Steghöfer, Department of Computer Science and Engineering

Master's Thesis 2021
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2021

Using machine learning to evaluate the layout quality of UML class diagrams
GUSTAV BERGSTRÖM
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

# Abstract

The Unified Modeling Language (UML) is the standard way of visualizing the design of software systems using diagrams. It is important that the layout of a UML diagram is of high quality so that it is easy for a human to comprehend. Evaluation of layout quality is hard and is often done through time and resource consuming user studies.

This work uses a big data set of UML class diagrams to utilize the power of machine learning for creating an automatic evaluator of layout quality, and for finding the particular features of diagrams that have the highest impact on the layout quality.

Diagram features, inspired by layout aesthetics commonly mentioned in literature, that may affect layout quality were extracted using image processing. To establish a ground truth for the layout quality of the diagram data set, all diagrams were manually labeled with their perceived layout quality. The features and labels were provided to different machine learning algorithms to train on gaining information from.

The best performing machine learning approach was using a random forest, which gave a correlation of 0.66 and a relative absolute error of 72.47%. This indicates that it was able to successfully gain important information from the features about how to evaluate layout quality. The two most impactful features for evaluating layout quality were found to be the length of the longest line and orthogonal placement of rectangles.

The layout quality evaluator that was created can be useful for efficiently comparing different kinds of diagram layouts, for example layouts created by different algorithms to see which algorithm performs better. The findings regarding the most important layout aesthetics can be useful for indicating what aesthetics to prioritize when constructing layouts and layout algorithms.

Keywords: UML, machine learning, image processing, layout quality.

# Acknowledgements

I would like to thank my supervisor Michel Chaudron for his support throughout the entire work of this thesis. I would also like to thank Truong Ho-Quang and Rodi Jolak for their help and their dedication to this thesis. Finally, I would like to thank Yosser El Ahmar and Xavier Le Pallec for the help they provided.

Gustav Bergström, Gothenburg, April 2021

# Contents

# List of Figures

# List of Tables

# 1
# Introduction

This chapter introduces the topic and problem statement of this thesis. The purpose of the study and why it will make a significant contribution to the research field is described and the specific research questions are stated. The scope of the study is further framed by explaining the limitations that will be impactful. Finally, a brief overview of the rest of the report is given.

## 1.1 Background

The Unified Modeling Language (UML) is the standard way of visualizing the design of software systems using diagrams. There exist many different types of UML diagrams that all serve different purposes and have different areas of use. The most commonly used type is the class diagram, which shows the different classes of a system and how they are related. Classes are drawn as rectangles and relationships are drawn as lines between the rectangles, sometimes with an arrow indicating the direction of the relationship. An example of a simple class diagram is seen in Figure 1.1. The diagram shows three classes and the relationships between them.

One of the main purposes of UML diagrams is for people to get a quick overview and to easier understand the systems they represent. Therefore, it is important that they are easy for a human to comprehend and that they actually help with understanding the system. This is greatly affected by the quality of the diagram's layout, according to a study by Störrle [1]. In the study, participants were asked to perform certain tasks using diagrams that were classed as either "good" or "bad" layout. The participants performed significantly better using the "good" diagrams.

Layout quality can be both a subjective and an objective trait. It can be both about how aesthetically appealing a user finds the diagram and how easily the user can actually comprehend the diagram. Several specific layout aesthetics have been shown in research to affect the layout quality. An example is the number of crossing lines in a diagram: fewer crossing lines contributes to a higher quality.

Class diagrams are often created in one of two ways, either before or after the system is developed. Diagrams created before the development are said to be forward engineered and can serve as a guiding reference during the development. Forward engineered diagrams are usually drawn by humans, who tend to have an intuitive feeling for layout. Obtaining a high quality on forward engineered diagrams is

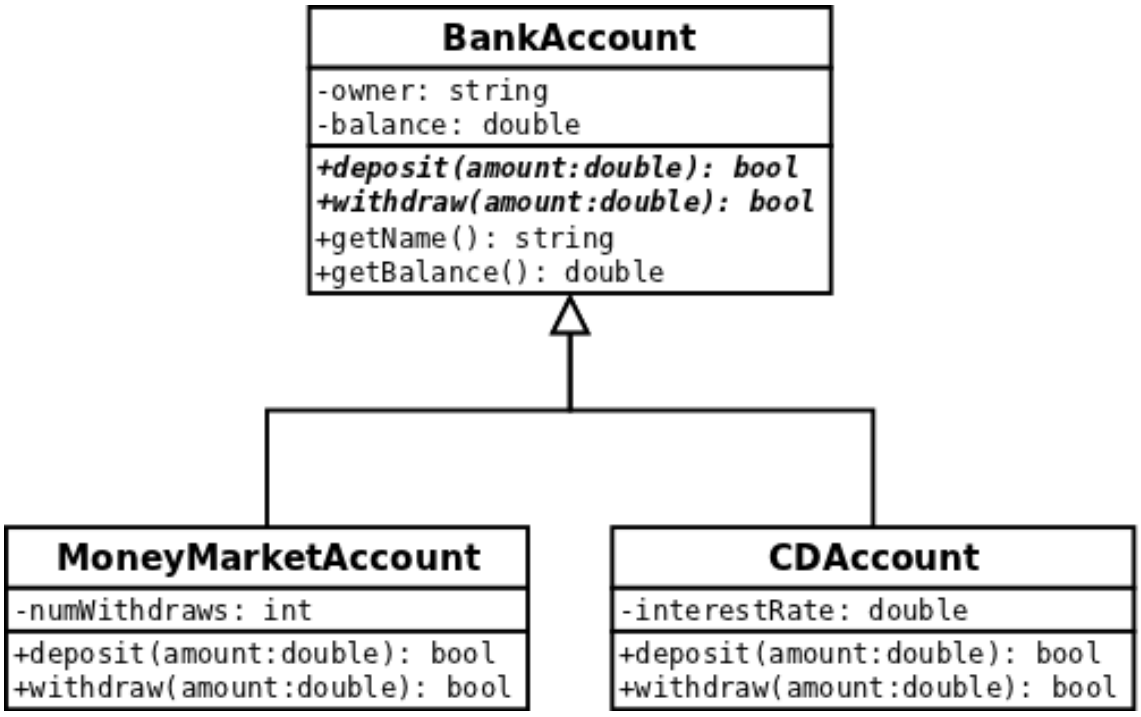


**Figure 1.1:** An example of a simple class diagram.

thereby usually not a problem. Diagrams created after the development are said to be reverse engineered and can serve as a documentation of the system. To save time, the diagrams are usually created using an automatic generator that generates a diagram given a code base. A wide range of algorithms for creating reverse engineered diagrams exist. It is important for such algorithms that the quality of the diagrams they produce is high.

Because of the importance of high layout quality, it is necessary to assess the quality somehow. Assessing individual diagrams can indicate if the layout quality of it is as good as desired, or if improvements need to be made. Assessment of diagrams produced by layout algorithms can furthermore help evaluating and compare the performance of the algorithms.

Recently, work has been done to collect a big data set of UML class diagrams [2]. The data set consists of thousands of diagrams and the idea behind this thesis is that the data set can be used to learn about layout quality.

## 1.2 Problem

To evaluate the layout quality of diagrams, time and resource consuming user studies are often conducted to see if the users find diagrams with one layout easier to comprehend than diagrams with other layouts. An automatic evaluator could give a quick indication of how good a layout is.

Furthermore, when constructing a diagram layout, there are many aesthetic criteria to consider. Some of them might even be conflicting so that when trying to optimize one of them, another might be suffering. An example of this is the two aesthetics saying that the number of line crossings and line bends should be minimized. Figure 1.2 illustrates how the two aesthetics conflict with each other. In the diagram to the left there is one line crossing. In the diagram to the right, the crossing has been removed, but instead a line bend is introduced. Therefore, it is important to know which aesthetics are most important for the layout quality, so that they can be focused on when constructing layouts. By creating a large data set of features representing aesthetics and labels representing the perceived quality, it can be found which aesthetics seem to be the most important ones for the quality.

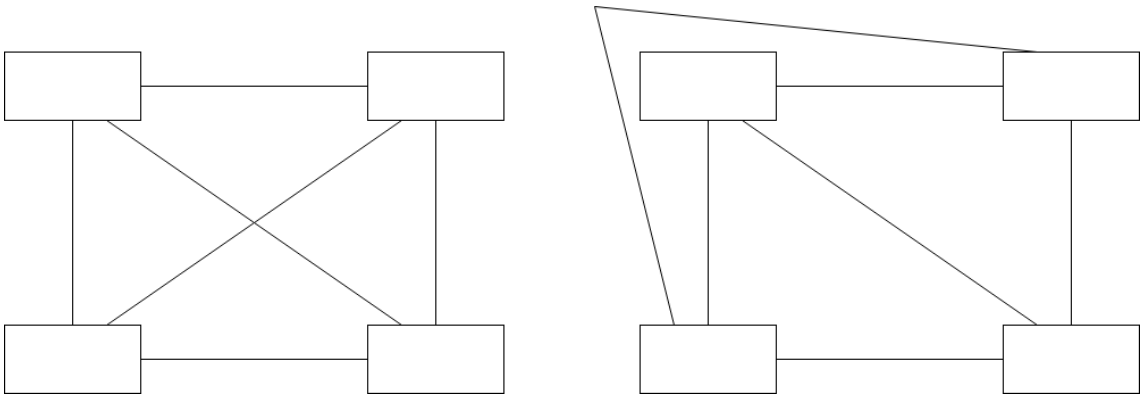

**Figure 1.2:** An example of conflicting aesthetics.

## 1.3 Purpose

The primary purpose of this study is to create an automatic evaluator of the layout quality of UML class diagrams using machine learning approaches. If the evaluator performs well, it can be a valuable tool for assessing the quality of class diagrams. The tool could also give suggestions about which aspects of a diagram are good and which could be improved. The study can also give an indication of which parts of the evaluator that work well and which parts that seem to be able to be improved. This information can be valuable for future research in the same area.

The basis for creating the evaluator will be the creation of a ground truth data set of diagrams, with their respective features and layout quality. This data set can be used to find information regarding which features seem to be the most important ones for the overall perceived layout quality. This information can be very useful when creating layouts. If some features seem to have a higher impact on the perceived quality than other, the emphasis can be on trying to improve those features in the layout creation. The data set can also be used to find benchmark data about class diagram aesthetics in general, for example averages and variances.

## 1.4 Research questions

The main research question of this study is:

- **RQ1:** How can machine learning be used for automatic evaluation of the layout quality of UML class diagrams?

To answer the main research question, the following sub-questions are investigated:

- **RQ1.1:** How well does the automatic evaluator perform evaluating the layout quality of previously unseen class diagrams?
- **RQ1.2:** Which features of class diagram layout are the most important for evaluating their layout quality?

## 1.5 Limitations

This study will only focus on UML class diagrams and not other types of UML diagrams. This is because class diagrams are the most commonly used UML diagrams and the data set that is used mostly contains class diagrams. Class diagrams are also the type of UML diagrams that are closest to mathematical graphs, which means research from that field can be used in this study.

The diagrams from the database have been collected by crawling open source repositories. Thus, all diagrams that are used for this project come specifically from open source projects where it has been decided to create a class diagram.

Since image processing will be used to extract features from the diagrams, not all imaginable features will be possible to consider. Due to the time frame of this project, as many features as possible that are convenient enough to extract will be considered. For example, the semantic meaning of classes and relations will be very difficult to find with the image processing that is used and no aesthetics regarding that will thereby be considered.

The ground truth regarding the quality of the diagrams will be the subjectively perceived quality by humans. Usually, the quality of diagrams is assessed by conducting tests on how well users can understand them and perform certain tasks with the help of them. Since this project uses a very big data set, conducting such time consuming user tests on every diagram is not feasible.

There exist many different machine learning algorithms that may be suitable for different use cases. For the scope of this project, algorithms provided by the machine learning software tool *WEKA*[1] that can perform regression will be used. Many machine learning algorithms have tweakable parameters that can be tuned to try to improve the performance of the algorithms. In this project, only default parameters provided by WEKA will be used. This is described further in Section 7.2.

[1] https://www.cs.waikato.ac.nz/ml/weka

## 1.6 Report overview

Chapter 2 gives a background on layout quality aesthetics and gives an overview of related work to this thesis. Chapter 3 presents the research methodology that is used and how it is adapted to this study. Chapter 4 presents the data set of diagrams that is used and how it is preprocessed to fit the needs of this study. Chapter 5 describes the extraction of features from the diagrams in the data set. The image processing approaches that are used to find important elements in the diagrams are described, as well as which features are extracted and how they are defined and calculated. Chapter 6 describes the process of manually labeling the diagrams in the data set with a perceived layout quality. Chapter 7 describes the machine learning approaches that are used to build the automatic evaluator. Chapter 8 presents the results of the study in terms of providing answers to the research questions. Chapter 9 discusses and analyses the found results, and frames possible validity threats. Chapter 10 gives a final conclusion on this work and suggests future work that can be done within the field.

# 2
# Background

This chapter brings up relevant background for this thesis. It starts with a summary of layout aesthetics and the research that has been done in that area. After that follows a presentation of related work that has been done in similar areas.

## 2.1 Layout aesthetics

Layout aesthetics are properties of a diagram layout that can have a relationship to the subjective perception of the diagram. Since UML class diagrams are closely related to mathematical graphs, aesthetics regarding graph layout are highly relevant. Research has been done both when it comes to general graph layout aesthetics, as well as aesthetics that are more specific to UML class diagrams.

According to Störrle [1], the layout of UML diagrams is governed by four levels of design principles.

- First, there are general principles that apply for all kinds of diagrams, for example that elements should be aligned and not obscure each other.
- Second, there are principles that apply to mathematical graphs, for example that the number of crossings and bends of lines should be minimized.
- Third, there are principles that apply mostly to UML diagrams, for example that similar elements should be grouped.
- The fourth principle level is support for better addressing the audience, for example by highlighting items with color or size.

Most research on UML diagrams focuses on principles from the second level. This is probably because these principles are to a higher extent quantifiable and measurable than principles from other levels. This is of high importance for this thesis, since the aesthetics need to be found with image processing, and then converted into numerical features for the machine learning part to work. As mentioned in Section 1.5, aesthetics that relate to the semantics of class diagrams will not be investigated.

Purchase [3] presents seven common aesthetic criteria for graph drawings and defines metrics to assess the presence of each one of them. In other research, Purchase et al. [4] [5] studies graph layout aesthetics with a focus on UML, where some additional aesthetics are presented. The aesthetics are evaluated empirically to find their relationships to user preferences. Ware et al. [6] studies the how eight different graph layout aesthetics affect the cognitive load of finding the shortest path between two nodes in a graph. Eichelberger [7] describes and orders 14 aesthetic for class

diagrams in a priority list. Eichelberger [8] extends on this in his PhD thesis, and presents a large number of aesthetics that are grouped into different categories. In later work, Eichelberger and Schmid [9] make an extensive summary of guidelines for the aesthetic quality of UML diagrams on different levels based on prior work. Sun and Wong [10] presents 14 criteria for UML class diagram layout, where some are more general and apply to all graph drawings and some more specifically target the semantics of the UML diagram. Coleman and Parker [11] presents a list of 19 graph layout aesthetics that were derived from literature and common sense.

| ID | Aesthetic |
|---|---|
| A1 | Line crossings |
| A2 | Line bends |
| A3 | Orthogonality |
| A4 | Line lengths |
| A5 | Diagram drawing size |
| A6 | Symmetry |
| A7 | Line angular distances |
| A8 | Class placement |
| A9 | Overlapping |
| A10 | Node sizes |

**Table 2.1:** Layout aesthetics.

Table 2.1 lists layout aesthetics that are prominent in research and can be useful for this project. The aesthetics are explained in the following sections.

## Line crossings

A line crossing is a point in the diagram where two lines intersect. If more than two lines intersect in the same point, all pairwise intersections are considered [3]. Minimizing the number of line crossings is one of the most commonly referenced aesthetics. Crossings make the lines harder to follow [10] and it is harder to see which classes are connected [8]. Crossings at small angles are more likely to cause visual confusion than crossings with an angle close to 90° [6].

## Line bends

A line bend is a point on a line that does not lie on a straight line between the two end points of the line [3]. Minimizing the number of line bends is also a very commonly referenced aesthetic. Straight lines are more continuous [10] and easier to follow for users [8].

## Orthogonality

The orthogonality of a line represents how far from an orthogonal angle it deviates [3]. To improve layout quality, lines should be drawn horizontally or vertically on an orthogonal grid [8] [10]. This is because horizontal and vertical orientations are

more likely to be perceived as figures than other orientations [9]. Just as with the lines, the nodes should also be placed on an orthogonal grid [3] [8] [10].

## Line lengths

The length of a line is the distance from the start to the end of the line, through all points on the line. Lines should not be too long or too short, since it makes grouping and separation hard [7] [10]. Line lengths should also be be kept as uniform as possible in a diagram [8].

## Diagram drawing size

The actual size of the diagram drawing should be minimized to support a homogenous node and line distribution and to reduce the need of scrolling [9]. Naturally, the drawing still has to fit all of the diagram elements and making it too small would probably create conflicts with other aesthetics. Some research only focuses on that it is the width of the drawing that should be minimized [11].

The aspect ratio of a drawing is the relationship between the height and the width. An optimal aspect ratio could be either minimized, which means that the diagram is quadratic [8], or fixed to a specific rectangular proportion [7].

## Symmetry

A lot of research suggests that increasing the symmetry of a diagram leads to an increased understandability of it. Symmetric areas are usually seen as a good figure [10] [9]. However, it is an aesthetic that can be hard to define as it is best considered perceptually rather than computationally [5]. Symmetry lines could be drawn arbitrarily in diagrams and there could be both global and local symmetry. All kinds of symmetry should preferably be maximized [8].

## Line angular distance

If there are multiple lines going from a node, the minimum angle between two lines should be maximized [3] [11]. The lines should be far apart so that it's easy to distinguish between them, which is of extra high importance if the resolution of the screen that the diagram is viewed on is low [8].

## Class placement

A lot of research on graph layout brings up the placement of nodes as an important aesthetic. Since classes in a UML diagram can be seen as nodes in a graph, the research is relevant for class diagrams as well.

Nodes should be distributed uniformly within the drawing area [8] [9] [11]. This helps providing a uniform appearance of the drawing which supports similarity and homogeneity [9]. Dishomogeneity leads to double observation, a discontinuity in the

visual perception process [8].

Nodes should not be too close together or too far apart [11]. Nodes that are connected to each other should be as close as possible to each other [8]. Nodes should also not be too close to edges that they are not connected to [7] [11]. Classes with a high degree should be placed near the center of the diagram [8].

### Overlapping

Nodes should not overlap other nodes [7] [8] [9] [10]. When nodes overlap, part of one node is not visible and the entire diagram cannot be read by the user. This is usually disliked by users [9].

Nodes and edges should not overlap each other [7] [8] [9] [10]. If a node overlaps an edge, it might look like the edge enters and exits the node instead of going past it. If an edge overlaps a node, it is not as problematic. This is usually tolerated by users, but should however be avoided [9].

Edges that are not intended to be joined should not overlap each other, which means that every edge should be readable as an individual [7] [8] [9]. Edge overlapping can be differentiated from edge crossing by defining overlapping as two edges having a path segment in common, rather than just a crossing point [9]. Edge overlapping can also be considered as edge crossing [10]. Edge overlapping is similar to node overlapping. At least a segment of one edge is not visible as an individual path, which means the entire diagram is not readable [9].

### Class sizes

The difference among sizes of the rectangles representing classes should be minimized [8]. The class sizes should also be as small as possible [8] [7].

## 2.2 Related work

No prior work is known that tries to create an automatic evaluator of layout quality of class diagrams. The big diagram data base that is used for this study is relatively new and unique. It is the creation of this data set that has opened up the opportunity for this study and the possibilities to do something similar before have been limited. However, there exists work that is related to different parts of this thesis. This includes evaluating layout quality and finding the most important layout aesthetics. Moreover, some similar work has been done that does not have the same purpose as this thesis, but uses similar methods. The related work on these topics is described in the following sections.

### 2.2.1 Evaluating layout quality

As seen in Section 2.1, work has been done to analyze individual layout aesthetics and how they contribute to the perceived quality of a diagram. However, no work is found where the goal is to use those aesthetics to somehow try to evaluate the overall quality. Störrle [1] classifies diagrams as "good" or "bad" to conduct his study, but mentions that not much emphasis is put on this classification. He simply classifies diagrams that conform to positive aesthetics and do not violate negative aesthetics as "good" diagrams.

### 2.2.2 Importance of aesthetics

When it comes to finding the importance of different layout aesthetics for the overall quality, some work has been done, which was also touched upon in Section 2.1. However, it has been proven hard to find significant results for many of the aesthetics.

Purchase et al. [12] performed a user study to validate the correlation between some aesthetics and the understandability of graphs. The participants had to carry out tasks to test how well they could understand graphs with different levels of conformance to the investigated aesthetics. The tasks had to be completed within a time limit and the measured variable was whether the participants could find the correct answers to the tasks or not. The study showed that minimizing line crossings (A1) and line bends (A2) had a significant correlation with the understandability of graphs, while the hypothesis that increasing local symmetry (A6) increases understandability is unconfirmed.

Purchase [13] performed a similar study, with the difference that some additional aesthetics were investigated and that both the taken time and the correctness of the answers to the tasks were measured. The study showed that the effect of minimizing line crossings (A1) was significant for both time and correctness, the effect of minimizing line bends (A2) was significant for correctness but only approaches significance for time, and the effect of increasing symmetry (A6) was significant for time but not for correctness. However, the effect of increasing orthogonality (A3), as well as maximizing the minimum angle between edges leaving the same node (A7), was non-significant for both correctness and time.

Purchase et al. [4] performed another kind of user study to test the user preference of diagrams with different values of different aesthetics. The participants were given pairs of diagrams: one with a high value of a certain aesthetic and one with a low value of the same aesthetic. They then had to choose which of the diagrams they preferred. The data was analyzed by calculating a percentage preference for each of the aesthetics. The percentage preference was 93% for fewer line crossings (A1), 91% for fewer line bends (A2), 73% for narrower diagrams (A5), and 61% for increased orthogonality (A3). All results were statistically significant.

Purchase et al. [5] also performed a user study where participants were given a text specification and an example diagram modelling the specification. The exam-

ple diagram had a medium high value of all of the investigated aesthetics. The participants were then presented with diagrams with higher and lower values of the aesthetics than the example diagram, where some diagrams modelled the same specification as the given one and some did not. They were to answer if the presented diagrams modelled the given specification or not, and both the accuracy and time of the answers was measured. After the study, the authors concluded that only the aesthetic of minimizing line bends (A2) seemed to matter, although only a little. None of the other investigated aesthetics, including line length variation (A4), orthogonality (A3), symmetry (A6), node distribution (A8) and having short but not too short lines (A4), had a significant impact.

Ware et al. [6] performed a user study where the time needed for participants to perceive the shortest path between two specified nodes in a graph was measured. In the graphs, the values of a number of aesthetics were varied. The study showed that the continuity of the shortest path (related to A2) and line crossings on the shortest path (A1) were significant. Other aesthetics, including total number of line crossings in the graph (A1), line crossing angles on the shortest path (A1), average line length on the shortest path (A4) and total line length on the shortest path (A4), were not significant.

As mentioned in Section 2.1, Eichelberger [7] orders 14 aesthetics for class diagrams in a priority list. The importance of the aesthetics were validated through different discussions in software engineering courses, evaluations of CASE tools and the author's own work on a domain-specific layout algorithm. However, no evaluation by making user studies was done. In the list, general constraints on nodes, including distances between nodes (A8), avoiding overlapping (A9) and minimizing class sizes (A10), is on third place. Avoiding line crossings (A1) is on fifth place. General edge constraints, including line lengths (A4) and line bends (A2) as well as not placing nodes too close to edges (A8), is on sixth place. Graph drawing constraints, including aspect ratio (A5), drawing size (A5), symmetry (A6), line angles (A7) and, again, line bends (A2), is on fourteenth place. Many of the other aesthetics on the list relates to semantics in diagrams.

### 2.2.3 Similar work

On the topic of using image processing to find features of UML diagrams, some similar work has been done. Karasneh and Chaudron [14][15] have developed a system that reads an image of a UML diagram and extracts the semantic meaning, i.e. the classes and relationships, of it. From this information it creates an XMI file of the UML model.

Moreover, Ho-Quang et al. [16] have developed an automatic classifier of UML class diagrams, which was done using similar methods as this thesis aims to do. It uses image processing to find features in images, and then applies machine learning to train a model that can distinguish if an image is a class diagram or not. The difference from this thesis is that the classifier has to recognize images that both

contain and do not contain class diagrams, which makes the image processing that is required a bit different. Furthermore, the machine learning approach differs in this thesis since a binary classifier is not what is desired, but rather an evaluator that can estimate the layout quality as a numerical value of a diagram.

# 3
# Research methodology

The chosen research methodology for this thesis is design science research. The development of an automated evaluator that utilizes machine learning very much resembles the artifact development that is central in design science research. This chapter describes the methodology in theory and how it is applied in this work.

## 3.1 Design science research

Design science research is a research methodology that is based on design science, which is the science of artifacts. As opposed to natural things, artifacts are created by human beings and could for example be machines or organizations [17]. The two major activities of design science are designing and investigating artifacts to improve something in a problem context [18].

Johannesson and Perjons [19] present a method framework for design science research, which consists of the following five activities:

**1. Explicate problem**
This activity is about analyzing and formulating a practical problem. The problem should be shown to be significant not only for a local practice, but be of general interest.

**2. Define requirements**
In this activity an artifact is proposed as a solution to the explicated problem. The problem is transformed into requirements of the artifact.

**3. Design and develop artifact**
In this activity an artifact that fulfills the defined requirements is created.

**4. Demonstrate artifact**
In this activity the feasibility of the developed artifact is illustrated in a real-life case.

**5. Evaluate artifact**
In this final activity the artifact is evaluated to determine how well it fulfills its requirement and how well it solves the problem.

Dresch et al. [17] reviews literature on design science research methods and con-

cludes that the following four steps are the most prominent and are present in almost all literature:

1. **Problem definition**
2. **Suggestions for possible solutions**
3. **Development**
4. **Evaluation**

These steps are evidently very similar to the method framework presented by Johannesson and Perjons, with the exception of the demonstration activity from the framework not being included.

## 3.2 Applying design science research

The design science research methodology used in this thesis is derived from the methods mentioned in Section 3.1 and is boiled down to the following four steps, visualized in Figure 3.1:

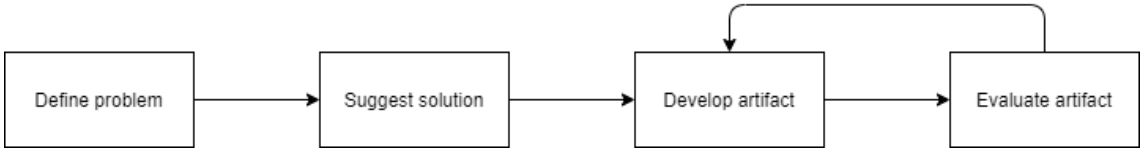


**Figure 3.1:** Design science research methodology in this thesis.

**1. Define problem**
The problem that this thesis addresses is that evaluation of layout quality is hard and time consuming, which is described in more detail in Section 1.2.

**2. Suggest solution**
The suggested solution for this problem is an automatic evaluator of the layout quality of UML class diagrams. The solution should take an image of a class diagram as input and give an estimation of the layout quality as output.

Figure 3.2 shows a breakdown of the suggested artifact, which on the highest level consists of two parts: A data set of class diagram representations that can be used to train and evaluate a machine learning model, and a machine learning approach with as high performance as possible with regards to evaluating layout quality. The data set of class diagram representations requires two parts: Features representing those diagrams, and labels that indicate their layout quality. To extract features that represent the class diagrams, four parts are required: Image processing to extract elements of interest from the images, creating or selecting features that give a good representation of the diagrams with respect to the layout aesthetics described in Section 2.1, as well as the definitions of the actual calculation of those features using information extracted from the image processing, and finally a data set of class diagram images that the features should be extracted from. Obtaining the image

data set requires a filtering strategy to only include diagrams that are presumably appropriate for a machine learning approach to gain valuable information from. As the class diagram data set is not already labeled with layout quality, a strategy is needed to manually obtain such labels, as well as a data set to apply this labeling on, the same data set as the features are extracted from. To come up with high performing machine learning approach, a set of different approaches is needed so that those can be compared and the one that seems most feasible can be chosen.

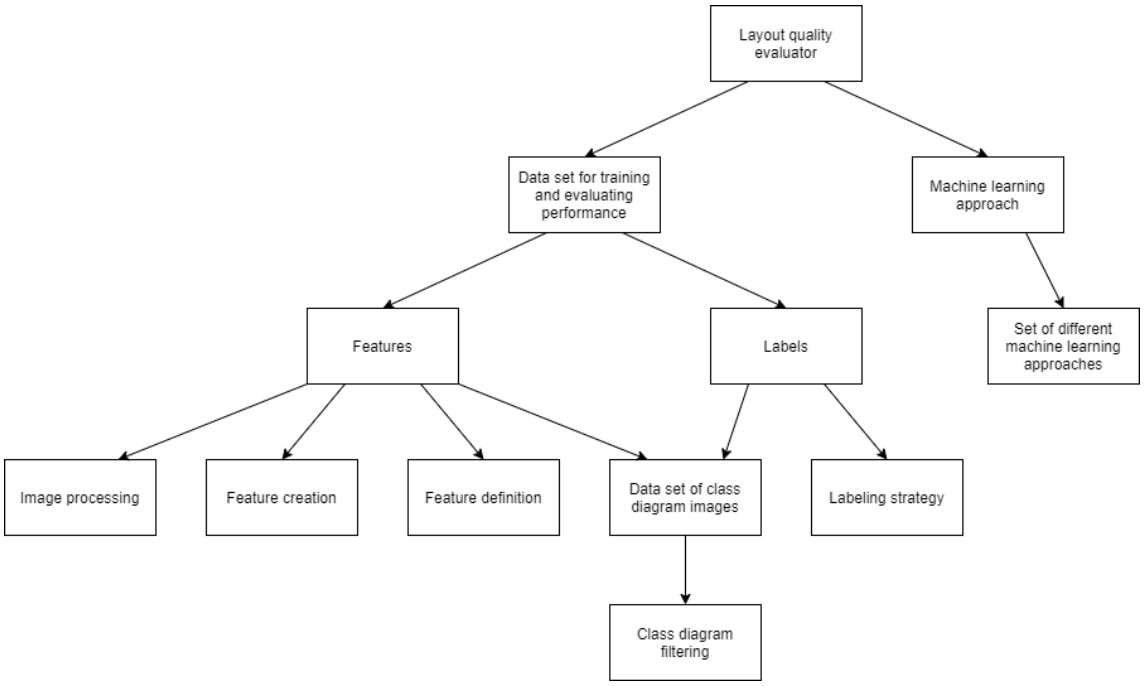


**Figure 3.2:** Breakdown of the suggested artifact.

The suggested solution can be seen as a set of the smallest building blocks of this artifact. Those six building blocks are represented as leaves in the tree seen in Figure 3.2 and are as follows:

- Image processing
- Feature creation
- Feature definition
- Class diagram filtering
- Labeling strategy
- Set of different machine learning approaches

**3. Develop artifact**

Developing the artifact involves developing all of the six artifact building blocks mentioned above, which is described in the following parts of this report respectively: *Image processing* in Section 5.1, *Feature creation* in Section 5.2, *Feature definition* in Section 5.3, *Class diagram filtering* in Section 4.2, *Labeling strategy* in Chapter 6 and *Machine learning approaches* in Chapter 7.

**4. Evaluate artifact**
The artifact is evaluated by analyzing its performance based on metrics described in Section 7.4. As the evaluation is done using cross validation, which is described in Section 7.2, this step can also be considered to include the demonstration activity that Johannesson and Perjons [19] suggest. The test folds of the cross validation contain real life diagrams, of which the artifact tries to evaluate the layout quality.

In Figure 3.1 there is also an arrow going from *Evaluate artifact* to *Develop artifact*, which represents an iterative workflow that is often used in design science research. During the evaluation of the artifact, potential flaws and areas of improvement are detected. The development step is revisited with the goal of improving the artifact, which is then evaluated and compared to the previous version. In this thesis, some iterative improvements were performed but due the time constrain of a master thesis the number of iterations were limited.

## 3.3 Methodology overview

Figure 3.3 shows an overview of the methodology, including inputs to the artifact development, the development of the artifact building blocks, intermediate activities that are required to put together the artifact, and the evaluation of the artifact. It also shows how the research questions are answered. The enumerated parts of the figure are described below.

1. The diagram database that is described in Section 4.1.
2. A filtering strategy is developed to decide which diagrams from the data set (1) that are suitable to use in this project. This is described in Section 4.2.
3. The diagrams from the data set (1) are filtered manually according to the developed filtering strategy (2), which results in a filtered diagram data set (4).
4. A subset of the entire data set (1) that contains the diagrams for which features and labels will be found.
5. A labeling strategy for how to manually label diagrams is developed. This is described in Chapter 6.
6. The filtered diagram data set (4) is labeled using the labeling strategy (5) that was developed. This results in a set of labels for the diagrams (7).
7. A data set containing labels that represent the layout quality of the diagrams in the data set (4).
8. The image processing software from previous similar work that is described in Section 5.1.2.
9. The image processing is developed by examining the image processing software (8) to understand how it works and see what parts of it that are usable for this project. This is also described in Section 5.1.2.
10. The filtered diagram data set is processed using the image processing (9) that was developed, which results in a set of processed diagrams (11), including the diagrams' found elements.

11. The output of the image processing (10), which is the diagrams represented by the rectangles and lines that were found in them.
12. The layout aesthetics found in literature that are described in Section 2.1.
13. Features are created by examining the layout aesthetics (12) to come up with feature representations of them that can be extracted from diagrams processed with image processing (9). This is described in Section 5.2.
14. Ways of numerically calculating the created features (13) are defined. This is described in Section 5.3.
15. Features from the feature creation (13) of the processed diagrams are computed using the defined feature calculations (14). This results in a set of features for the diagrams.
16. A data set containing features that represent different layout aesthetics (12) for the diagrams in the data set (4).
17. Features (16) and labels (7) are combined into a data set (18) that can be used by machine learning.
18. The ultimate data set of features and labels that is used by machine learning.
19. Feature selection is performed to answer **RQ1.2** (20) and as an input to the machine learning approaches (21). This is described in Section 7.1.
20. The output of the feature selection (19) is used to answer **RQ1.2**.
21. A set of machine learning approaches is developed, including a general approach described in Section 7.2, different machine learning algorithms described in Section 7.3, and selected features from the feature selection (19).
22. The machine learning approaches (21) are used to train and evaluate machine learning models on the data set of features and labels (18). The best model is used as the artifact (23) of this design science research.
23. The best performing machine learning model is chosen as the artifact which will be evaluated.
24. The artifact is evaluated by examining the evaluation of the best performing model, which was already done (22). The arrow going back to the development section illustrates an iterative process of trying to improve the artifact. The dashed extension of the line shows that it is only the development of the filtering (2) that is iterated upon for the scope of this project. Finally, the evaluation of the artifact is used to answer **RQ1.1** (25).
25. The evaluation of the artifact (24) is used to answer **RQ1.1**.

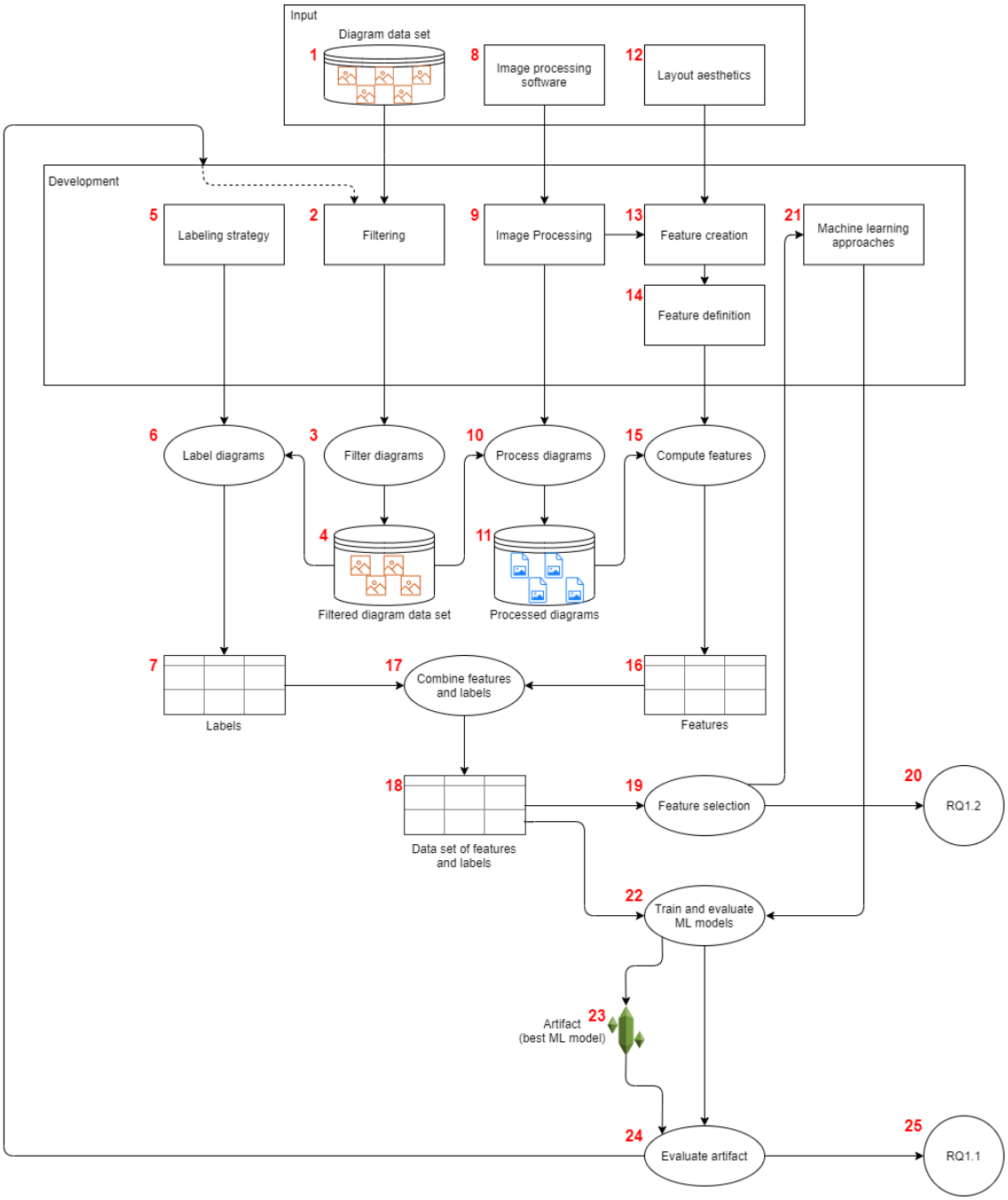


**Figure 3.3:** Overview of the methodology.

# 4
# Diagram data set

To be able to train and evaluate a machine learning model, a large data set of class diagrams is required. Such a data set was compiled in previous research and provided for this thesis. This chapter describes that data set, as well as how it was filtered to only include diagrams suitable for machine learning training and evaluation.

## 4.1 Initial data set

The data set of diagram images that is used for this project is the result of previous research by Hebig et al. [2] The data set consists of thousands of UML diagrams, that are mostly class diagrams, and it was created by scraping open source GitHub repositories for images. The images were then classified as being either UML diagrams or not, using the work done by Ho-Quang et al. [16]

For this project, a subset of the whole data set was given, consisting of a few thousands of class diagram images. The diagrams are of quite different character, which is beneficial when creating an evaluator that is supposed to be general. Examples of this can be seen in Figure 4.1. The diagram on top has blue and gray classes and only vertical and horizontal lines, while the diagram below has yellow classes and diagonal lines, sometimes with arrows and labels.

There are also some diagrams that are rather trivial for an analysis of the layout quality, for example diagrams with very few classes. An example of this can be seen in Figure 4.2, which shows a diagram with only two classes. In such a case, not much can be done to improve or worsen the layout quality.

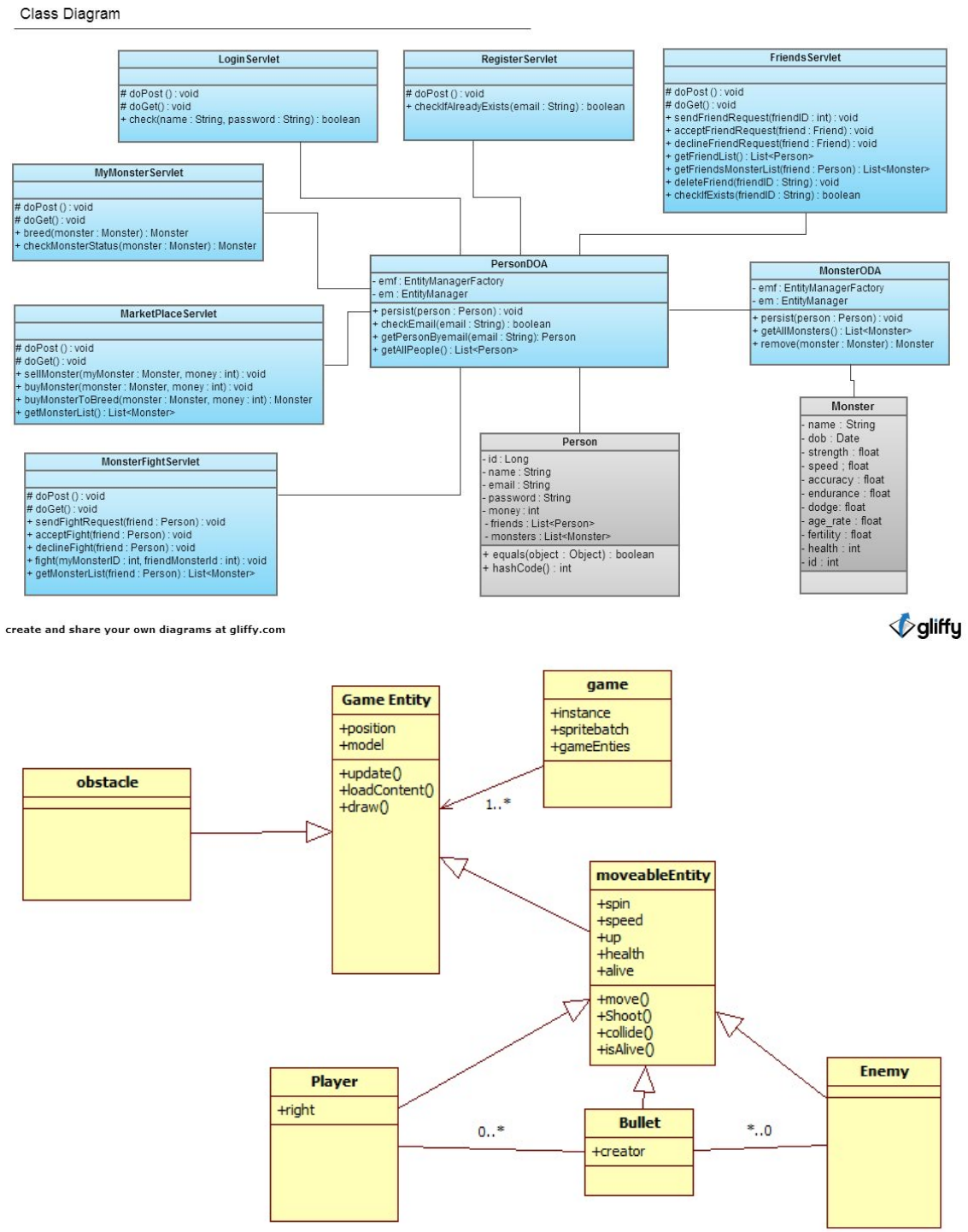


**Figure 4.1:** An example of two diagrams with different characteristics.

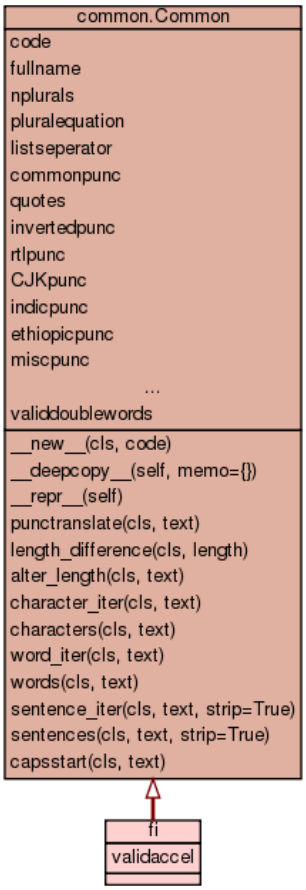


**Figure 4.2:** An example of a diagram with a trivial layout.

## 4.2 Filtering the data set

As shown in Section 4.1, there exist trivial diagrams in the data set that are not interesting regarding layout quality. The data set also contains rare occasions of diagrams that are not actually class diagrams, and diagrams that do not conform to the UML specification to represent classes as boxes. Furthermore, there are some completely useless diagrams in the data set that are not possible to process and extract features from. Some examples are hand drawn diagrams, screenshots of diagrams inside a software, cropped off diagrams, duplicates, and diagrams with backgrounds or overlays. The data set needed to be filtered to remove such undesired diagrams. This was done iteratively as flaws were detected in the initial filtering. The following sections describe the iterations of this process.

### 4.2.1 First filtering

To only include interesting diagrams, a lower bound on the amount of classes was needed. As seen in Figure 4.2, the layout is trivial for a diagram with only two classes. Even in the case with a diagram with three classes the layout is rather trivial. Even if all classes have relationships to each other, they can simply be laid out as a triangle. However, when a diagram reaches four classes, interesting decisions need to be done regarding the layout, as seen in Figure 1.2. Therefore, it was decided to include diagrams with four or more classes.

Diagrams were excluded from the data set if they violated at least one of the following six rules:

- The diagram should be of the type class diagram.
- Classes should be represented as rectangular boxes in the diagram.
- The image should display the entire diagram and only contain diagram elements.
- The diagram should contain four or more classes.
- The diagram should contain three or more relations.
- No duplicates should be included.

After the filtering, a total of 654 diagrams remained to be used in the final data set.

### 4.2.2 Second filtering

The validation of the image processing, which is described in Section 5.1.3, showed that it is much worse at finding lines than rectangles. An examination of the calculations of the defined features, which are described in Section 5.3, showed that in some diagrams there were no lines found at all. This led to that many of the features that were based on lines had incorrect values. This gives a lot of misinformation in the data that a machine learning model is supposed to learn from, which is likely to affect its performance.

To try to somewhat mitigate this flaw, a second iteration of the filtering was performed with the following rule added:

- The image processing should find at least one line in the diagram.

This new rule filtered out an additional 42 diagrams, which left a total of 612 diagrams in the data set.

# 5
# Feature extraction

This chapter describes the process of extracting layout aesthetics based features that can be used in the machine learning. The feature extraction includes three major parts: Processing the images to find the elements in the diagrams, selecting features that represent relevant layout aesthetics, and finally defining how to calculate those features given the output of the image processing.

## 5.1 Image processing

The diagrams in the data set are just image files, without any information of the layout of the diagrams. To be able to extract layout quality features from the diagrams, image processing is necessary to find the classes and relationships and where they are positioned. Since classes are represented as rectangles and relationships are represented as lines, those are the shapes that are searched for in the images.

The image processing method that was used is based on the work of Ho-Quang et al. [16]. This section briefly describes the algorithms that the method uses and how they are put together to find rectangles and lines, as well as a validation of how well the method works with the class diagram data set that is used in this thesis.

### 5.1.1 Image processing algorithms

Four common image processing algorithms are used by the method for finding elements in the diagram images. Those are *Canny edge detection*, *Hough Transformation*, *Suzuki 85* and *Ramer-Douglas-Peucker*, and the basics of them are described in the following sections.

#### Canny edge detection

Canny edge detection [20] is an algorithm for detecting edges in images. First, the algorithm removes noise in the image by blurring it using a Gaussian filter [21]. The blurred image is then used to find the edge gradient and direction for each pixel in the image. Then, the algorithm removes any pixels that are not part of an edge. This happens if the pixel is not a local maximum in the direction of the gradient. Finally, the edges are thresholded to only keep the ones with a high intensity gradient or that are connected to such an edge. This removes all shorter lines that are considered noise.

#### Hough Transformation (HT)

Hough Transformation (HT) [22] is a procedure for detecting straight lines in images. It is applied on an image where edges are already detected. The edges are represented as a direction and distance from the center of the image. Edges that have the same direction and distance are considered to form a line.

#### Suzuki 85 (S85)

Suzuki 85 (S85) [23] is a border following algorithm that is used to find contours in an image. The pixels in the image are scanned and when a pixel that satisfies a condition for being a starting point of a border is found, that border is followed and all pixels on it are marked.

#### Ramer-Douglas-Peucker (RDP)

The Ramer–Douglas–Peucker (RDP) algorithm [24] is used to approximate a polygon with few points from a curve with arbitrarily many points. The algorithm calls itself recursively and removes the points that are least important for representing the curve. RDP is a suitable algorithm for finding shapes. For example, a curve that is approximated with four points could potentially be a rectangle.

### 5.1.2 Image processing method

Ho-Quang et al. [16] uses the algorithms described in Section 5.1.1 to extract shapes and lines from diagrams. *Canny edge detection* is used initially to detect edges in the images that the other algorithms can use. *S85* is used to find contours, from which *RDP* tries to find polygons. The contours that are found are also are split into line segments that together with the lines found by *HT* are used to find additional rectangles as well as lines representing relations in the diagram.

Figure 5.1 shows an overview of the method, which includes the following parts:

(A) The original image.
(B) Lines found by HT.
(C) Contours found by S85. Those are used by RDP to find shapes.
(D) Horizontal and vertical lines from (B) and (C) that are on the same axes and represent the same line are joined together into a single line. Rectangles that are not found by RDP are extracted from these lines by finding parallel horizontal lines that intersect the same two vertical lines on each end.
(E) The rest of the lines, including lines that are not horizontal or vertical, are joined and used to find connecting lines between shapes.
(F) The extracted elements that remain after all lines that are within shapes have been removed.

This image processing method was built to handle different kinds of diagrams with different shapes, while only rectangles and lines between those are of interest for the class diagrams used in this work. Therefore a minor modification to the method was done to ignore other shapes than rectangles slightly improve the rectangle extraction.

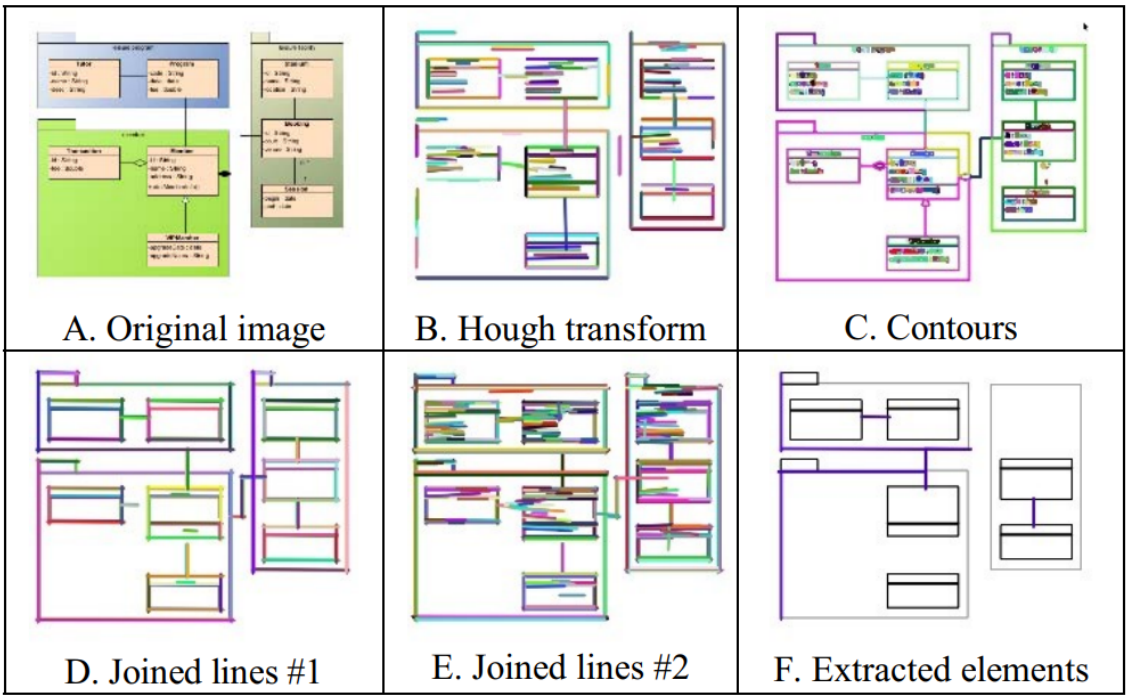


**Figure 5.1:** Image processing overview. Adapted from [16].

### 5.1.3 Validation

To validate the image processing, 10 random diagrams from the data set were chosen. The output of the image processing for those diagrams was examined manually to check how well it finds rectangles and lines. Table 5.1 shows the results of the validation in terms of how many of the rectangles and lines that are found, as well as how many rectangles and lines that are not there, but are falsely found. It is clear that the image processing is better at finding rectangles than it is at finding lines. 85 of the 105 rectangles are found while only 25 of the 107 lines are found. 17 rectangles and 23 lines are falsely found.

Figure 5.2 shows the image processing validation for diagram 4. The original diagram is to the left and the output of the image processing is to the right. For this diagram, the image processing works fairly well. All rectangles except one, as well as all solid lines, are found, and no rectangles or lines are falsely found. However, the dotted lines are not found.

Figure 5.3 shows the image processing validation for diagram 8. The original diagram is on top and the output of the image processing is below. For this diagram, the bigger rectangles are found, but the thinner ones are either not found or mistaken as lines. Some of the bigger rectangles are also found twice and no lines are correctly found.

Figure 5.4 shows the image processing validation for diagram 9. The original diagram is on top and the output of the image processing is below. It can be seen that

| Diagram | Actual rectangles | Found rectangles | False rectangles | Actual lines | Found lines | False lines |
|---|---|---|---|---|---|---|
| 1 | 16 | 10 | 3 | 24 | 3 | 2 |
| 2 | 11 | 10 | 3 | 9 | 6 | 5 |
| 3 | 8 | 7 | 3 | 7 | 4 | 0 |
| 4 | 7 | 6 | 0 | 10 | 6 | 0 |
| 5 | 8 | 7 | 3 | 8 | 3 | 1 |
| 6 | 22 | 19 | 1 | 20 | 2 | 8 |
| 7 | 7 | 5 | 1 | 9 | 0 | 4 |
| 8 | 10 | 6 | 3 | 6 | 0 | 3 |
| 9 | 11 | 11 | 0 | 10 | 0 | 0 |
| 10 | 5 | 4 | 0 | 4 | 1 | 0 |
| Total | 105 | 85 | 17 | 107 | 25 | 23 |

**Table 5.1:** Image processing validation results.

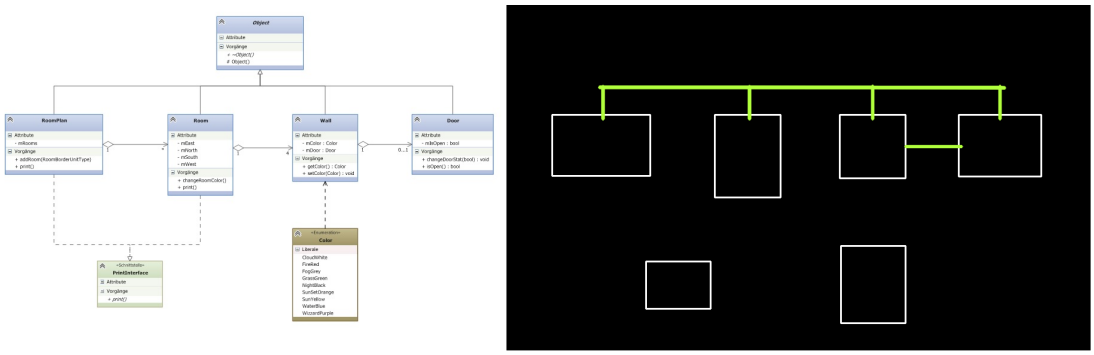

**Figure 5.2:** Image processing validation for diagram 4.

for some reason no lines are found in this diagram, even though the lines are very clearly visible in the original diagram. However, all rectangles are found, and no rectangles are falsely found.

Appendix A includes images of the validation results for the rest of the diagrams.

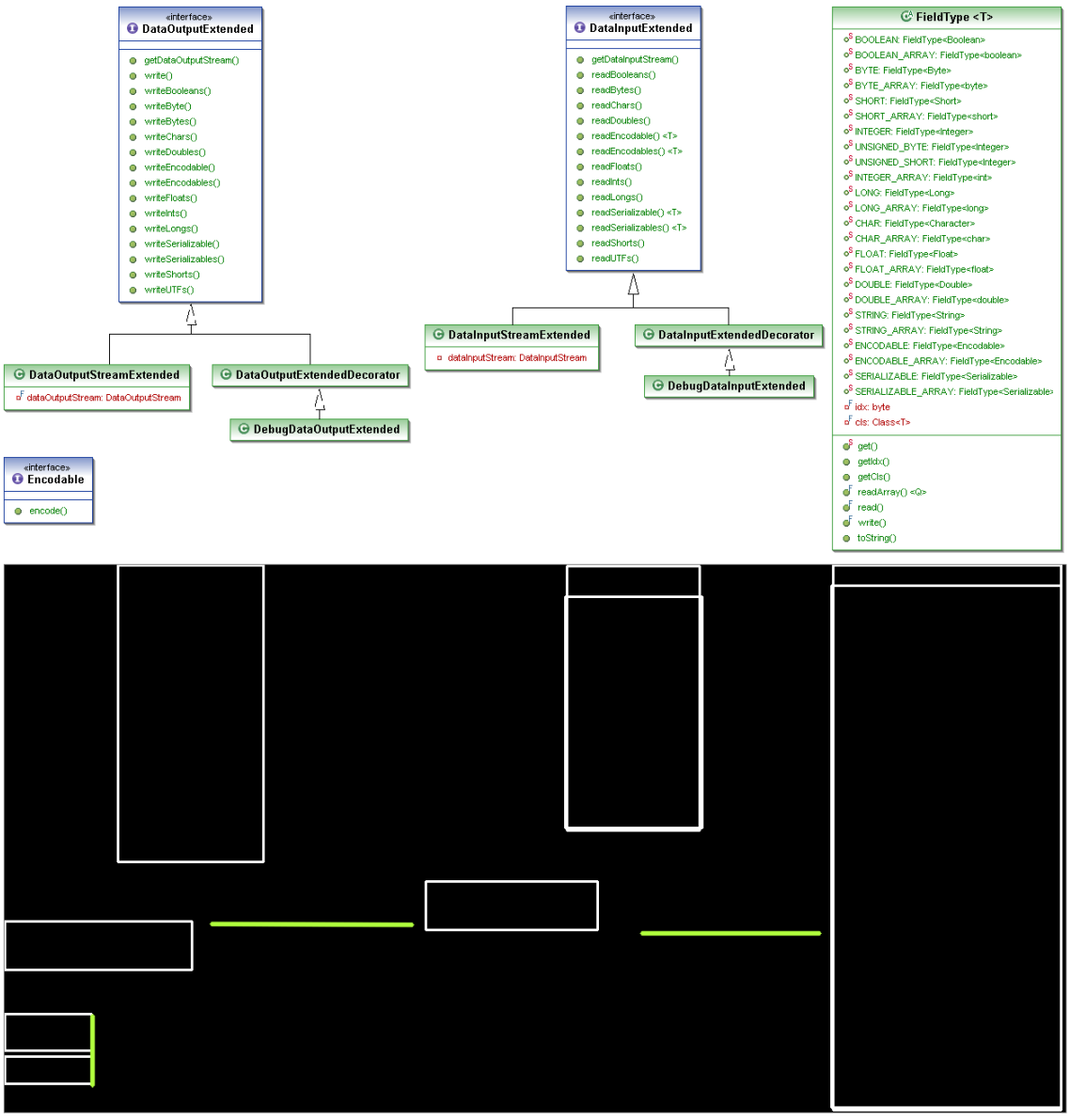


**Figure 5.3:** Image processing validation for diagram 8.

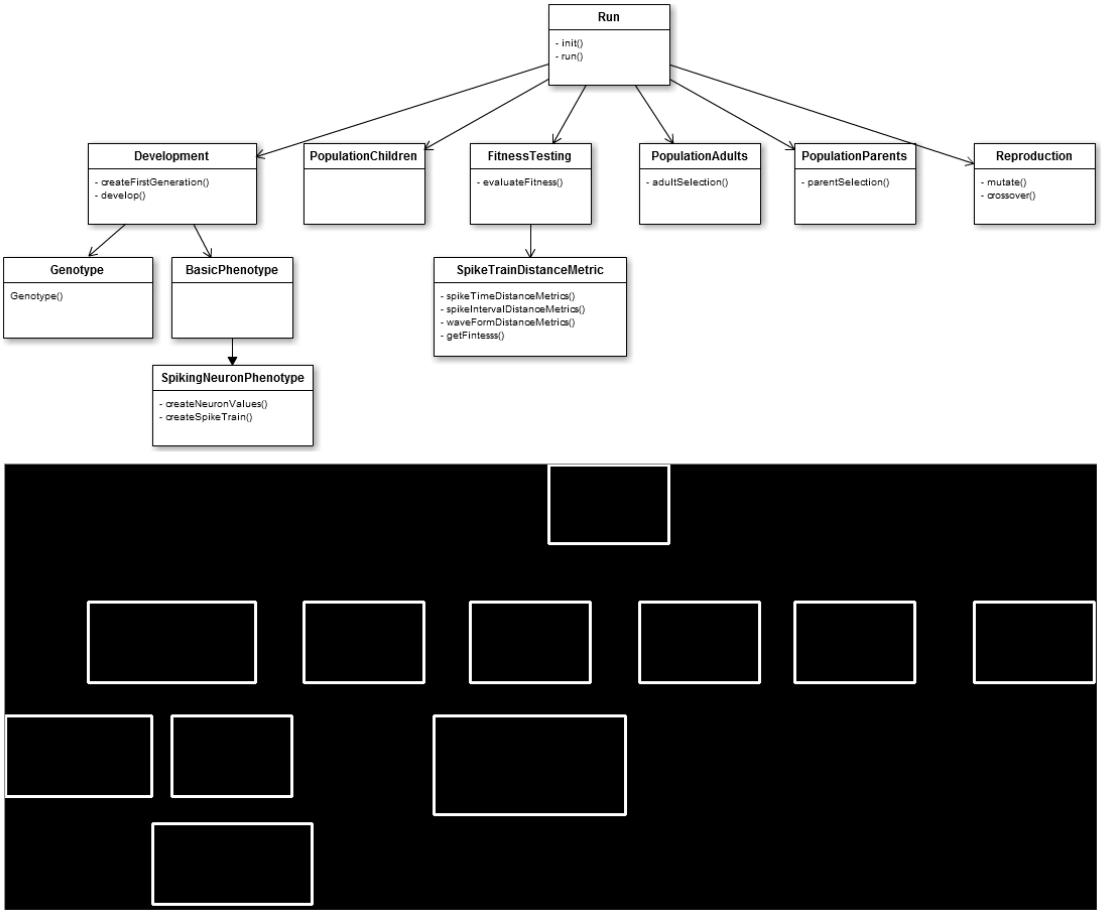


**Figure 5.4:** Image processing validation for diagram 9.

## 5.2 Feature creation

The features that appeared most important in literature and that were feasible to find with the image processing were selected for extraction. As stated in Section 5.1.2, rectangles and lines were found in the diagram images and many of the selected features are thereby ones that rely on those elements.

Below follows descriptions of possible features in relation to the aesthetics discussed in Section 2.1, and motivations for why they were selected or not.

#### A1 Line crossings

The number of crossing lines should be minimized and the angles of the line crossings should be maximized. It can easily be found which of the lines cross each other and at what angle, to extract the features *Line crossings* and *Crossing angles*.

#### A2 Line bends

The number of line bends should be minimized. The lines that are extracted in the image processing are divided into straight segments. For example, if a line has one bend, it is represented as two straight line segments that connect at the point of the line bend. Thanks to this representation, the feature *Line bends* can easily be extracted.

#### A3 Orthogonality

Lines should be drawn horizontally or vertically and rectangles should be placed on an orthogonal grid. The angle of the lines and their deviation from orthogonality can be used to extract the feature *Line angles*. It can also used to classify lines as orthogonal or not orthogonal and extract the feature *Line orthogonality*. The positions of the found rectangles can be checked to see how well they conform to an orthogonal grid, which gives the feature *Rectangle orthogonality*.

#### A4 Line lengths

Lines should not be too long or too short and the line lengths should be as uniform as possible. The length of the lines can easily be found and can be used to extract the features *Line length*, *Line length variation*, *Longest line* and *Shortest line*.

#### A5 Diagram drawing size

The size of a diagram should be minimized while still fitting all of the diagram elements and without conflicting other aesthetics by squeezing elements too close together. To address this aesthetic, the feature *Rectangle coverage* is extracted by finding the sizes of the rectangles. The feature measures how much of the total diagram area that is covered by rectangles. The aspect ratio of the diagram could also have an impact on the layout quality and the feature *Aspect ratio* can be extracted using the height and width of the diagram.

### A6 Symmetry

Symmetry in diagrams should be maximized to increase the understandability. According to Purchase et al. [5], a computational algorithm to evaluate the symmetry of a diagram would be complex, and also only a very rough model of how humans perceive symmetry. It was therefore chosen to not extract any features regarding symmetry.

### A7 Line angular distances

If multiple lines are going from a rectangle, the minimum angle between two of those lines should be maximized. The image processing software that was used does not relate the lines that it finds with rectangles. Thereby, no feature was extracted for this aesthetic.

### A8 Class placement

Rectangles should be distributed uniformly within the diagram and they should not be too close or too far apart. The positions of the rectangles can be used to extract the features *Rectangle distribution* and *Rectangle proximity*. Other aspects of this aesthetic include that connected rectangles should be close to each other, rectangles should not be too close to lines that they are not connected to, and rectangles with a high degree should be placed near the center of the diagram. All of these require connections between diagram elements, which, as mentioned, is not found by the image processing.

### A9 Overlapping

Rectangles and lines should not overlap each other. Since the image processing is built to only find standalone diagram elements, it gets confused by overlapping elements and can't find those. This makes it hard to extract a feature for this aesthetic.

### A10 Node sizes

Rectangles should be as small as possible and the difference among their sizes should be minimized. By finding the rectangle sizes the features *Rectangle size* and *Rectangle size variation* can be extracted.

Table 5.2 summarizes the features that were created and which aesthetic they relate to.

| ID | Feature name | Aesthetic |
|---|---|---|
| F1 | Line crossings | A1 |
| F2 | Crossing angles | A1 |
| F3 | Line bends | A2 |
| F4 | Line angles | A3 |
| F5 | Line orthogonality | A3 |
| F6 | Rectangle orthogonality | A3 |
| F7 | Line length | A4 |
| F8 | Line length variation | A4 |
| F9 | Longest line | A4 |
| F10 | Shortest line | A4 |
| F11 | Rectangle coverage | A5 |
| F12 | Aspect ratio | A5 |
| F13 | Rectangle distribution | A8 |
| F14 | Rectangle proximity | A8 |
| F15 | Rectangle size | A10 |
| F16 | Rectangle size variation | A10 |

**Table 5.2:** Created features.

## 5.3 Feature definitions

To be able to analyze the features and perform machine learning, a numerical value has to be calculated for each of the selected features. These values are based on the elements found by the image processing software. How each feature was defined and calculated is described below.

### F1 Line crossings

The number of line crossings in a diagram can be found by counting the number of intersections between the found lines. However, simply using the number of line crossings as a feature would favor diagrams with fewer relationships, as they would more likely have fewer line crossings. Instead, a relative value was desired, which led to dividing the number of line crossings with the maximum possible number of line crossings. This would occur if every line in the diagram would cross every other line. Then there would be $\frac{\#lines(\#lines-1)}{2}$ crossings. Equation 5.1 shows the definition.

$$\text{F1} = \frac{\#crossings * 2}{\#lines(\#lines - 1)} \tag{5.1}$$

### F2 Crossing angles

For each line crossing the crossing angle is calculated, which is a value between 0 and 90. The average crossing angle is then calculated and used as the feature. Equation

5.2 shows the definition.

$$\mathrm{F2} = \frac{\sum_{i=1}^{\#\mathrm{crossings}} \mathrm{angle}(\mathrm{crossing}_i)}{\#\mathrm{crossings}} \tag{5.2}$$

### F3 Line Bends

As mentioned in Section 5.2, lines are represented as straight segments and the number of bends on a line is one less than its number of segments. The average number of bends per line is calculated and used as the feature. Equation 5.3 shows the definition.

$$\mathrm{F3} = \frac{\sum_{i=1}^{\#\mathrm{lines}} \mathrm{bends}(\mathrm{line}_i)}{\#\mathrm{lines}} \tag{5.3}$$

### F4 Line angles

The deviation angle from orthogonality, i.e. how far from being horizontal or vertical, is calculated for each line. If a line is more than 45° degrees from being horizontal, it is less than 45° from being vertical and vice versa, which makes the deviation from orthogonality a value between 0 and 45. The average line angle is calculated and used as the feature. Equation 5.4 shows the definition.

$$\mathrm{F4} = \frac{\sum_{i=1}^{\#\mathrm{lines}} \mathrm{angle}(\mathrm{line}_i)}{\#\mathrm{lines}} \tag{5.4}$$

### F5 Line orthogonality

A line is orthogonal if it is exactly horizontal or vertical. A margin of error of 1° is used to allow for small deviations due to the quality of the image and the image processing. The number of orthogonal lines is divided by the total number of lines to get a ratio between 0 and 1 as the feature. Equation 5.5 shows the definition.

$$\mathrm{F5} = \frac{\sum_{i=1}^{\#\mathrm{lines}} \begin{cases} 1 & \text{if } \mathrm{angle}(\mathrm{line}_i) < 1° \\ 0 & \text{otherwise} \end{cases}}{\#\mathrm{lines}} \tag{5.5}$$

### F6 Rectangle orthogonality

The orthogonality of rectangles is calculated by counting the number of distinct rectangle positions on the x- and y-axis respectively. For both axes, a set of occupied positions is kept and all rectangles are looped through and examined to populate this set. If the currently examined rectangle has a position that is not is the set, it's position is added to the set. If it has, or is within a small error margin from, a position that is in the set, no position is added to the set. The ratio between the number of distinct rectangle positions and the total number of rectangles is

used to calculate feature, and the calculated values for the axes are added together. Equation 5.6 shows the definition.

$$\begin{aligned} \text{F6} = &\left(1 - \begin{cases} 0 & \text{if \#xPositions} = 1 \\ \frac{\text{\#xPositions}}{\#rectangles} & \text{otherwise} \end{cases}\right) \\ &+ \left(1 - \begin{cases} 0 & \text{if \#yPositions} = 1 \\ \frac{\text{\#yPositions}}{\#rectangles} & \text{otherwise} \end{cases}\right) \end{aligned} \tag{5.6}$$

**F7 Line length**

This feature is the average length of all found lines. Equation 5.7 shows the definition.

$$\text{F7} = \text{avg}(\{\text{length}(\text{line}_i) | 1 \leq i \leq \#\text{lines}\}) \tag{5.7}$$

**F8 Line length variation**

This feature measures how much the line lengths vary by calculating the standard deviation of the lengths. Equation 5.8 shows the definition.

$$\text{F8} = \text{stdev}(\{\text{length}(\text{line}_i) | 1 \leq i \leq \#\text{lines}\}) \tag{5.8}$$

**F9 Longest line**

This feature is the length of the longest found line. Equation 5.9 shows the definition.

$$\text{F9} = \max_{i=1}^{\#\text{lines}} \text{length}(\text{line}_i) \tag{5.9}$$

**F10 Shortest line**

This feature is the length of the shortest found line. Equation 5.10 shows the definition.

$$\text{F10} = \min_{i=1}^{\#\text{lines}} \text{length}(\text{line}_i) \tag{5.10}$$

**F11 Rectangle coverage**

This feature measures how much of the total diagram area that is covered by rectangles. This is done by adding together all rectangle areas. The total rectangle area is divided by the total area of the diagram to get a ratio between 0 and 1 as the feature. Equation 5.11 shows the definition.

$$\text{F11} = \frac{\sum_{i=1}^{\#\text{rectangles}} \text{size}(\text{rectangle}_i)}{\text{width} * \text{height}} \tag{5.11}$$

### F12 Aspect ratio

This feature is calculated by dividing the diagram image width with the image height. Equation 5.12 shows the definition.

$$\text{F12} = \frac{\text{width}}{\text{height}} \tag{5.12}$$

### F13 Rectangle distribution

For this feature, the image is divided into four quadrant sections of equal size. For each rectangle, the area of each of the four section that it covers is calculated. These rectangle areas are summed up for the four sections respectively, which gives each one of them a value for how much of it is covered by rectangles. The variance of these values is finally used as the feature. Equation 5.13 shows the definition.

$$\text{F13} = \text{var}(\{\text{rectangleCoverage}(\text{quadrant}_i) | 1 \leq i \leq 4\}) \tag{5.13}$$

### F14 Rectangle proximity

For each rectangle, the distance to the closest of the other rectangles is calculated. This means each rectangle has a value for how close to another rectangle it is. The average of this proximity value over all rectangles is used as the feature. Equation 5.14 shows the definition.

$$\text{F14} = \frac{\sum_{i=1}^{\#\text{rectangles}} \text{shortestDistanceToOtherRectangle}(\text{rectangle}_i)}{\#\text{rectangles}} \tag{5.14}$$

### F15 Rectangle size

This feature is the average area of all found rectangles. Equation 5.15 shows the definition.

$$\text{F15} = \frac{\sum_{i=1}^{\#\text{rectangles}} \text{size}(\text{rectangle}_i)}{\#\text{rectangles}} \tag{5.15}$$

### F16 Rectangle size variation

This feature measures how much the rectangle sizes vary by calculating the standard deviation of the sizes. Equation 5.16 shows the definition.

$$\text{F16} = \text{stdev}(\{\text{size}(\text{rectangle}_i) | 1 \leq i \leq \#\text{rectangles}\}) \tag{5.16}$$

# 6
# Labeling

The diagram data set that is used contains only images of diagrams, and has no information about the layout quality of the diagrams. To be able to train and evaluate a supervised machine learning model, the diagrams have to be labeled with a layout quality. This was done manually by experts in the area, and this chapter describes the steps of this process.

## 6.1 Labeling strategy

The labeling strategy consists of three parts. First, people with sufficient expertise to do the labeling have to be found. Second, a scale that the labelers can use to assess the diagrams has to be selected. Finally, an efficient way of labeling a big set of diagrams is required.

### 6.1.1 Finding experts

The people doing the manual labeling should have some expertise in the area of UML diagrams. They should have experience from working with UML diagrams and should have studied the topic at a university level. Professors, PhD students and master students with the required expertise were asked to participate in the labeling.

### 6.1.2 Likert scale

The Likert scale [25] was chosen as the labeling scale used for labeling diagrams. The scale origins from psychology and is used to measure peoples' attitude towards some statement. It is a bipolar scale and measures positive and negative, and sometimes also neutral, responses. Different numbers of options can be used but a typical Likert scale has five levels and presents the following five options:

1. Strongly disapprove
2. Disapprove
3. Undecided
4. Approve
5. Strongly approve

Naturally, a Likert scale is an ordinal scale, but if the distance between the options is the same and the scale is symmetric around a middle point it can also be treated as an interval scale. This is of interest regarding machine learning, as many regression algorithms assume an interval scale for the dependent variable.

For the labeling of layout quality of class diagrams in this thesis, a five level Likert scale was used. The scale is symmetric and intended to have equal distances between the five options that follow below:

1. Very bad
2. Bad
3. Ok
4. Good
5. Very good

### 6.1.3 Labeling tool

To streamline the labeling, a labeling software tool was developed and set up as a web page. Figure 6.1 shows what the tool looks like. To the left, the diagram that is supposed to be labeled is shown. At the bottom right, six buttons are shown. One for each of the five label options, as well as an option to skip the diagram. When one of those six buttons are clicked, a new diagram is shown. At the top right, there is an optional choice to mark which issues made an impact on the decision and to make a comment on the decision. This was added during the labeling strategy validation after discussions mentioned in Section 6.2.2.

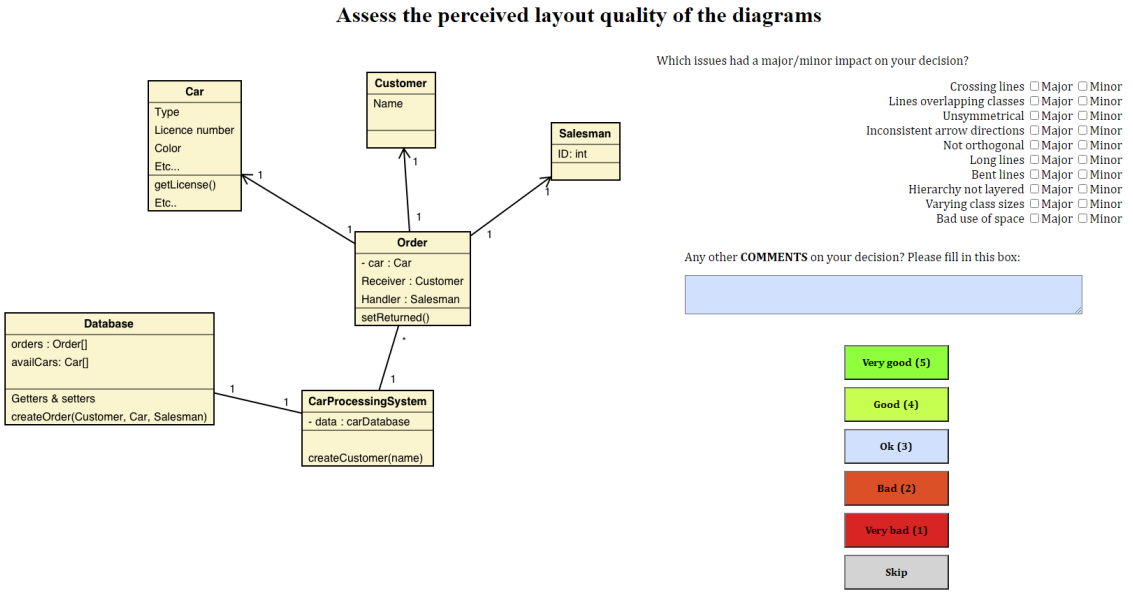


**Figure 6.1:** Screenshot of the labeling website.

## 6.2 Labeling strategy validation

To let the labelers try out the labeling strategy and to evaluate the feasibility of it, an iterative validation was performed. A group of four people who were to participate in the labeling were given 30 random diagrams from the data set to label. The interrater agreement was measured using intraclass correlation, which is described further in Section 6.2.1, and diagrams where the labels differed significantly were discussed to understand what caused those differences. The discussions led to slight variations in the way that the strategy was presented for the next round of the validation. The iterative validation rounds are described in Section 6.2.2.

### 6.2.1 Intraclass correlation (ICC)

To measure the interrater agreement between the different labelers during the validation, ICC was used. It is used to calculate a value between 0 and 1 that describes how strongly the labelers agree with each other. A higher value indicates a higher agreement.

Cicchetti [26] gives the following guidelines on how to interpret the ICC measure:

- $0.00 \leq \text{ICC} < 0.40$: Poor
- $0.40 \leq \text{ICC} < 0.60$: Fair
- $0.60 \leq \text{ICC} < 0.75$: Good
- $0.75 \leq \text{ICC} \leq 1.00$: Excellent

Koo and Li [27] give different, slightly stricter guidelines:

- $0.00 \leq \text{ICC} < 0.50$: Poor
- $0.50 \leq \text{ICC} < 0.75$: Moderate
- $0.75 \leq \text{ICC} < 0.90$: Good
- $0.90 \leq \text{ICC} \leq 1.00$: Excellent

### 6.2.2 Iterations

At each iterative round, a set of instructions on how to perform the labeling was given. Each labeler labeled 30 diagrams and the ICC was measured. Differences in labeling were discussed and the reasons for the differences were used to clarify the instructions for the next round.

#### Round 1

The following instructions were given to the labelers:

*"Please assess the diagrams with your perceived layout quality of it between (1) Very bad and (5) Very good. Consider how aesthetically appealing you find the diagram layout and how well you think it would help understanding the system it represents."*

There were some quite big differences between the labels for some of the diagrams. The ICC was 0.345, which is considered poor by both Cicchetti [26] and Koo and

Li [27]. A big reason for label differences was discovered to be the interpretation of semantic content of the diagrams. Some labelers took this into account when assessing the quality while others didn't. For example, diagrams with a bad class hierarchy were in some cases assessed with a low quality. To prevent this, it was agreed that the instructions needed to be clearer that it is only the layout of the diagrams that should be assessed. There was also a wish to enter reasons for the assessments, to visualize labeling differences and simplify the discussions for the next iteration.

### Round 2

The following sentence was added to the instructions:

*"Assess only the layout of the diagrams without considering the semantic content of them."*

In addition to the changes to the instructions, the labeling tool was modified to allow entering reasons for the assessments. This can be seen at the top right of Figure 6.1. Common issues that were found in the diagrams of iteration 1 are listed and the labeler can express which issues had a minor or a major impact on their decision. A free text comment box was also added, where other thoughts could be added on top of the defined issues.

This time the interrater agreement was much higher, with an ICC of 0.469, which is considered fair by Cicchetti [26], but still poor by Koo and Li [27]. The diagrams with the most varying labels seemed to be very simple ones, with few classes and relations. Such diagrams were in some cases not given a high quality even if there was really no room for improvement. Figure 6.2 shows an example of such a diagram. This diagram was labeled with a 5 by three of the labelers and a 3 by the fourth labeler, with the motivation that it was too simple to get a higher quality rating. It was agreed that diagrams can have a very good layout quality even if the complexity is low and that this should be added to the instructions for clarification.

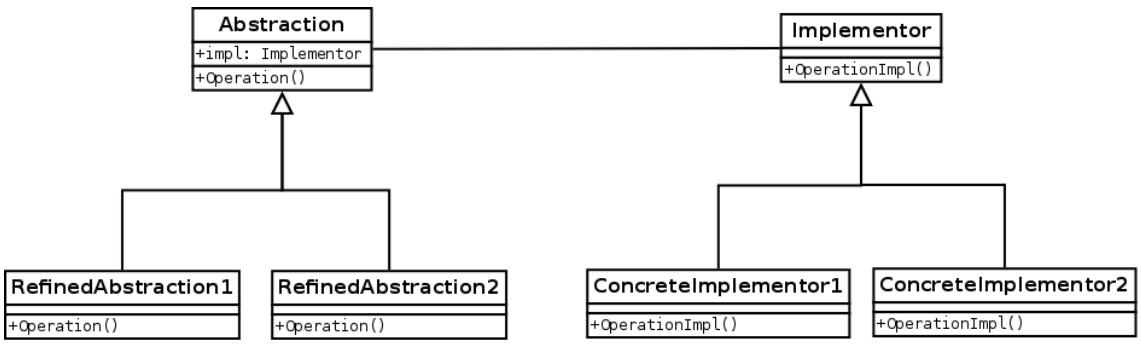


**Figure 6.2:** Example of a simple diagram with conflicting labels.

**Round 3**

The following sentences were added to the instructions:

*"Think about how many different issues you find with the diagrams that could be improved. A very small diagram without any issues could for example have a good layout even though the layout is very simple."*

There was now a slightly higher interrater agreement, with an ICC of 0.498, which is considered fair by Cicchetti [26], and very close to moderate by Koo and Li [27]. It was decided that this level of agreement was acceptable, since there is no absolute truth of what is a right or wrong layout quality assessment. Perceived quality is complex and it must be accepted to differ between different people.

## 6.3 Final labeling

Since ratings can differ between raters it was decided to let two people rate each diagram and take the average of those ratings as the label.

After the iterative validation, one more rater was able to join the labeling group to help speed up the process. The person labeled test set from iteration 3 and an ICC was calculated for all labelers together. The ICC was 0.500, which is almost unchanged from before this labeler was added. Therefore it was decided to let new labeler start labeling diagrams from the final data set.

## 6.4 Label distribution

Figure 6.3 shows the distribution of labels. The distribution seems to be roughly normally distributed, but slightly skewed towards the higher end. This is reasonable since diagrams are desired to have a high layout quality and this is probably strove for when creating the diagrams. This normal-like distribution is suitable for machine learning, and the fact that there is a fair amount of diagrams rated both 1 and 5 allows machine learning algorithms to learn from both very good and very bad diagrams, which is beneficial.

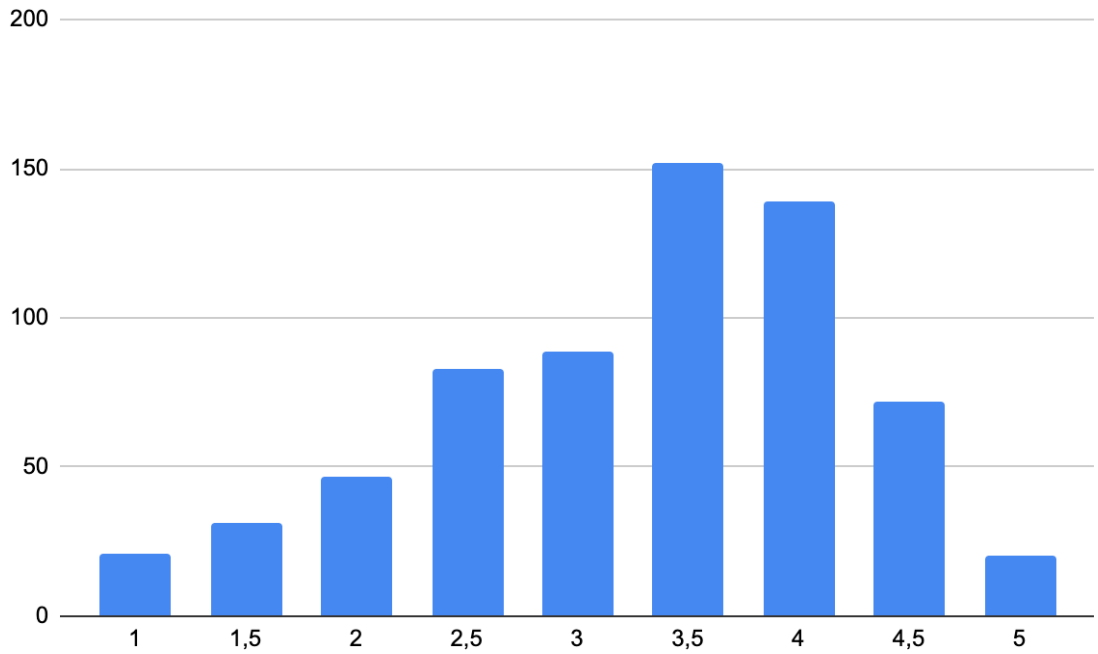


**Figure 6.3:** Distribution of layout quality labels of the entire diagram data set.

# 7
# Machine Learning

This chapter describes the machine learning approaches, algorithms and evaluation metrics that were used. The software tool *WEKA*[1] was used for the machine learning parts of this thesis. WEKA provides implementations of machine learning and feature selection algorithms as well as tools for preprocessing and visualizing data.

## 7.1 Feature selection

Feature selection is an approach that is commonly used in machine learning to limit the features that are used to a subset of all the features. The purpose of this is to select the features that have a significant impact on the dependent variable, and leave out the others out. This may improve the performance of machine learning algorithms as the insignificant features may provide complicating noise. The direct results of the feature selection is used to answer **RQ1.2**, by showing which layout aesthetics inspired features have the highest impact on the layout quality. The results are also used to try to improve the machine learning performance for **RQ1.1**. Two different feature selection algorithms are used, one that ranks all the features and one that finds the most suitable subset of features.

#### Ranking

The Pearson's correlation between each feature and the dependent variable is measured. The features are then ranked based on the absolute correlation as correlation can be negative and has a value between -1 and 1. To select a subset of features from this ranking, a threshold must be chosen. In this case it was chosen to select all features with an absolute correlation of 0.2 and above.

#### Subset

Subsets are evaluated based on the predictive ability of each feature and the degree of redundancy between them. Ideally a subset should contain features that have high correlation with the dependent variable but low intercorrelation [28]. No internal ranking among the features in the selected subset is given.

[1] https://www.cs.waikato.ac.nz/ml/weka

## 7.2 Machine learning approach

The different machine learning algorithms were trained and evaluated by using a 10-fold cross-validation. This means that the data set is split randomly into ten folds. One of the folds are held out at a time as a validation set and the other nine folds are used as a training set. When all ten folds have served as a validation set, the average of the ten validation results are taken as the final result.

The different calculated features have very different orders of magnitude. For example, *Rectangle coverage* is limited between 0 and 1 while *Rectangle sizes* can have very high values. This might complicate the balance between features for the machine learning algorithms. To mitigate this, all features are normalized to a value between 0 and 1.

As the layout quality, which is the dependent variable, is seen as an interval scale, a regression approach was taken for a machine learning model to make predictions on the same scale. The set of machine learning algorithms that were evaluated therefore consisted of algorithms provided by Weka that support regression. Many of the algorithms provide tweakable parameters that affect some parts of the algorithms. To not make the evaluation of different algorithms too extensive, it was chosen to only use the default parameters provided by WEKA. The different algorithms and their parameters are described in Section 7.3.

## 7.3 Machine learning algorithms

Twelve different machine learning algorithms provided by WEKA were evaluated. In this section, those are briefly described, including their tweakable parameters and the default values that WEKA provides for those.

#### LinearRegression

*LinearRegression* performs least-squares multiple linear regression [29]. This algorithm includes feature selection and the method for doing this can be changed. It also has a mechanism for detecting and eliminating colinear features, which can be enabled or disabled. Finally, there is a ridge parameter that can reduce overfitting. Table 7.1 shows WEKA's default values for the parameters of LinearRegression.

| Parameter | Value |
|---|---|
| Feature selection method | M5 |
| Eliminate colinear features | enabled |
| Ridge | $10^{-8}$ |

**Table 7.1:** Parameter values for LinearRegression.

### SimpleLinearRegression

*SimpleLinearRegression* performs linear regression by choosing only one feature, the on that gives the smallest squared error [29]. It has no tweakable parameters.

### SMOreg

*SMOreg* implements the sequential minimal optimization algorithm for learning a support vector regression model [29]. It has a complexity parameter $C$ and it can perform a transformation of the data. The transformation is set to *Normalize* by default, which has no effect as the data is already normalized which is mentioned in Section 7.2. SMOreg also uses a kernel function and an optimization function. Table 7.2 shows WEKA's default values for the parameters of SMOreg.

| Parameter | Value |
|---|---|
| C | 1.0 |
| Data transformation | Normalize |
| Kernel function | PolyKernel |
| Optimization function | RegSMOImproved |

**Table 7.2:** Parameter values for SMOReg.

### GaussianProcesses

*GaussianProcesses* implements the Bayesian Gaussian process technique for non-linear regression [29]. Like SMOreg, GaussianProcesses uses a kernel function and can perform a data transformation. It also has a noise regularization parameter for controlling the closeness of fit. Table 7.3 shows WEKA's default values for the parameters of GaussianProcesses.

| Parameter | Value |
|---|---|
| Kernel function | PolyKernel |
| Transition | Normalize |
| Noise | 1.0 |

**Table 7.3:** Parameter values for GaussianProcesses.

### MultilayerPerceptron

*MultilayerPerceptron* is a neural network that is trained using back propagation [29]. It has an input layer, a number of hidden layers and an output layer. It has many different tweakable parameters, including learning rate, momentum, and number of training epochs. It can also use a validation set for determining when the training performance is getting worse. This is not used by default, and the tweakable validation set size is thereby set to 0. Finally, there is a parameter to define the number of hidden layers and the number of nodes in each hidden layer.

The default is one hidden layer with a number of nodes equal to the average of the number of features and the number of class values. Table 7.4 shows WEKA's default values for the parameters of MultilayerPerceptron.

| Parameter | Value |
|---|---|
| Learning rate | 0.3 |
| Momentum | 0.2 |
| Training epochs | 500 |
| Validation set size | 0 |
| Hidden layers | $\frac{\#\text{features}+\#\text{classes}}{2}$ |

**Table 7.4:** Parameter values for MultilayerPerceptron.

### DecisionTable

*DecisionTable* builds and uses a simple decision table by evaluating feature subsets [29]. The feature subset search method can be altered as well as the evaluation measure. There is also an option to use the nearest-neighbor method *IBk* instead of the table's global majority as the method for determining the class for instances not covered by a decision table entry. Table 7.5 shows WEKA's default values for the parameters of DecisionTable.

| Parameter | Value |
|---|---|
| Search method | BestFirst |
| Evaluation measure | RMSE |
| Use IBk | disabled |

**Table 7.5:** Parameter values for DecisionTable.

### M5Rules

*M5Rules* obtains regression rules using separate-and-conquer [29]. In each iteration a model tree is built using the M5 algorithm and the best leaf is made into a rule. It has a tweakable parameter to control the minimum number of instances to allow at a leaf node. It also provides options to generate unpruned trees, to generate a regression tree instead of a model tree, and to use unsmoothed predictions. Table 7.6 shows WEKA's default values for the parameters of M5Rules.

| Parameter | Value |
|---|---|
| Min instances | 4.0 |
| Unpruned | disabled |
| Regression tree | disabled |
| Unsmoothed | disabled |

**Table 7.6:** Parameter values for M5Rules.

### RandomTree

*RandomTree* constructs a tree that considers a given number of random features at each node, performing no pruning [29]. RandomTree has many tweakable parameters that affect the tree construction, including the number of features at each node, the maximum depth of the tree, the minimum total weight of the instances in a leaf, and the minimum proportion of the variance on all the data that needs to be present at a node to perform splitting. The number of folds of the data that is used for backfitting can also be set. The default value for this is 0, which means no backfitting is performed. There are also options to allow unclassified instances and to break ties randomly when several attributes look equally good. Table 7.7 shows WEKA's default values for the parameters of RandomTree.

| **Parameter** | **Value** |
|---|---|
| Features | $\text{int}(\log_2(\#\text{predictors}) + 1)$ |
| Max depth | unlimited |
| Min weight | 1.0 |
| Min variance proportion | 0.001 |
| Backfitting folds | 0 |
| Allow unclassified instances | disabled |
| Break ties randomly | disabled |

**Table 7.7:** Parameter values for RandomTree.

### RandomForest

*RandomForest* constructs a random forest by bagging ensembles of random trees [29]. Naturally, it shares some parameters with RandomTree, including number of features at each node, maximum tree depth and breaking ties randomly. Moreover, the size of each bag as a percentage of the training set size, the number of execution slots to use for constructing the ensemble, and the number of iterations to be performed, can be tweaked. Finally, there are options to calculate the out-of-bag error and to compute feature importance. Table 7.8 shows WEKA's default values for the parameters of RandomForest.

| **Parameter** | **Value** |
|---|---|
| Features | $\text{int}(\log_2(\#\text{predictors}) + 1)$ |
| Max depth | unlimited |
| Break ties randomly | disabled |
| Bag size | 100% |
| Execution slots | 1 |
| Iterations | 100 |
| Calculate out-of-bag | disabled |
| Compute feature importance | disabled |

**Table 7.8:** Parameter values for RandomForest.

### REPTree

*REPTree* builds a regression tree using information gain/variance [29]. It also shares some parameters with RandomTree, including maximum tree depth, minimum total weight of the instances in a leaf, and variance proportion needed for splittning. REPTree can prune the tree using reduced-error pruning and the number of folds of the data used for pruning can be set. The data that is not used for pruning is used for growing the rules. Finally, the initial class value count can also be specified. Table 7.9 shows WEKA's default values for the parameters of REPTree.

| | |
|---|---|
| Max depth | unlimited |
| Min weight | 2.0 |
| Min variance proportion | 0.001 |
| Pruning | enabled |
| Pruning folds | 3 |
| Initial count | 0.0 |

**Table 7.9:** Parameter values for REPTree.

### M5P

*M5P* implements the M5' model tree algorithm, which is an improvement to the M5 algorithm [29]. It has the same four tweakable parameters as M5Rules. Those are the minimum number of instances to allow at a leaf node as well as the three options to generate unpruned trees, to generate a regression tree instead of a model tree, and to use unsmoothed predictions. Table 7.10 shows WEKA's default values for the parameters of M5P.

| **Parameter** | **Value** |
|---|---|
| Min instances | 4.0 |
| Unpruned | disabled |
| Regression tree | disabled |
| Unsmoothed | disabled |

**Table 7.10:** Parameter values for M5P.

### DecisionStump

*DecisionStump* builds one-level binary decision trees and does regression based on mean squared error [29]. It has no tweakable parameters.

## 7.4 Evaluation metrics

To be able to evaluate different machine learning approaches, some evaluation metrics are required. This section describes the metrics provided by WEKA that were chosen for evaluating.

### Correlation coefficient

The correlation coefficient measures the correlation between the predicted and actual values, in this case predicted and labeled layout quality of class diagrams. This is a value between -1 and 1, where negative values represent a negative correlation while positive values represent a positive correlation. The absolute value of the correlation coefficient indicates how much of the variance in the data that is explained by the evaluated model. For example, a correlation coefficient of 0.5 indicates that 50% of the variance is explained by the model. There exist different ways of calculating correlation and WEKA uses Pearson's correlation coefficient for this. The correlation between variables $X$ and $Y$ is calculated using Equation 7.1, where $n$ is the sample size, $X_i, Y_i$ are the sample points, and $\overline{X}$ and $\overline{Y}$ are the sample means for $X$ and $Y$ respectively.

$$r_{XY} = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \overline{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \overline{Y})^2}} \tag{7.1}$$

The following interpretation guidelines for correlation are given by Evans [30]:

- Less than 0.20: very weak
- 0.20 to 0.39: weak
- 0.40 to 0.59: moderate
- 0.60 to 0.79: strong
- 0.80 or greater: very strong

### Mean absolute error (MAE)

The mean absolute error (MAE) is a measure of error that uses the absolute errors between predictions and actual values. The mean of those errors for all diagrams in the validation set is taken to get the MAE. Naturally, the MAE should be as low as possible. If the MAE is 0, it means that all predictions were exactly correct. MAE is calculated using Equation 7.2, where $n$ is the sample size and $Y_i$ is the true value and $X_i$ is the prediction value for sample point $i$.

$$\text{MAE} = \frac{\sum_{i=1}^{n}|Y_i - X_i|}{n} \tag{7.2}$$

### Relative absolute error (RAE)

The relative absolute error (RAE) measures how well the machine learning approach performs with respect to absolute error in comparison to a trivial approach that simply predicts the mean value of the data set for each data point. An RAE below 100% indicates that the approach performs better than the trivial approach, while an RAE above 100% is worse than the trivial approach. The RAE should in other words be as low as possible. RAE is calculated using Equation 7.3, where $n$ is the sample size, $Y_i$ is the true value and $X_i$ is the prediction value for sample point $i$,

and $\overline{Y}$ is the sample mean for the true values. As seen in the equation, RAE is proportional to MAE.

$$\text{RAE} = \frac{\sum_{i=1}^{n} |Y_i - X_i|}{\sum_{i=1}^{n} |Y_i - \overline{Y}|} \tag{7.3}$$

# 8
# Results

This chapter presents the results that were found in this study with respect to the research sub-questions stated in Section 1.4:

- **RQ1.1:** How well does the automatic evaluator perform evaluating the layout quality of previously unseen diagrams?
- **RQ1.2:** Which features of class diagram layout are the most important for evaluating their layout quality?

The chapter is structured in iterations representing the iterations of the artifact development described in Section 3.2. Two iterations were performed, where the diagram filtering was altered between the iterations, which is described in Section 4.2. For each iteration, the results with respect to both sub-questions are presented. **RQ1.2** is answered using the feature selection algorithms described in Section 7.1 and **RQ1.1** is answered using the machine learning approaches described in Section 7.2 and Section 7.3. The results for **RQ1.2** are presented first, as they have an impact on the method for obtaining results for **RQ1.1**. Finally, the results for the final artifact, which is the best performing machine learning model, are presented.

## 8.1 First diagram filtering

This section presents the results with respect to **RQ1.2** and **RQ1.1** using the diagram data set obtained by performing the first diagram filtering described in Section 4.2.1.

### 8.1.1 Feature selection

Feature selection was performed using both the ranking and the subset feature selection algorithms described in Section 7.1.

Table 8.1 shows the results for the ranking algorithm. The correlation between the labels and all the features respectively are presented, sorted by absolute correlation.

Table 8.2 shows the results for the subset algorithm. The best found subset of features is presented, in no particular order. It shows that the best found subset consists of four of the 16 features.

| ID | Feature | Correlation | Absolute correlation |
|---|---|---|---|
| F9 | Longest line | -0.49 | 0.49 |
| F7 | Line length | -0.37 | 0.37 |
| F6 | Rectangle orthogonality | 0.37 | 0.37 |
| F8 | Line length variation | -0.32 | 0.32 |
| F2 | Crossing angles | 0.26 | 0.26 |
| F3 | Line bends | -0.24 | 0.24 |
| F14 | Rectangle proximity | -0.21 | 0.21 |
| F15 | Rectangle size | -0.19 | 0.19 |
| F4 | Line angles | -0.18 | 0.18 |
| F1 | Line crossings | -0.17 | 0.17 |
| F16 | Rectangle size variation | -0.14 | 0.14 |
| F13 | Rectangle distribution | 0.10 | 0.10 |
| F11 | Rectangle coverage | 0.08 | 0.08 |
| F12 | Aspect ratio | 0.05 | 0.05 |
| F10 | Shortest line | -0.03 | 0.03 |
| F5 | Line orthogonality | -0.03 | 0.03 |

**Table 8.1:** Feature selection for the first diagram filtering using correlation based ranking.

| ID | Feature |
|---|---|
| F16 | Rectangle size variation |
| F9 | Longest line |
| F2 | Crossing angles |
| F6 | Rectangle orthogonality |

**Table 8.2:** Feature selection for the first diagram filtering using the subset algorithm.

### 8.1.2 Machine learning approaches

The machine learning algorithms presented in Section 7.3 were trained and evaluated using the general approaches described in Section 7.2 on three different feature sets:

1. All features
2. Result of ranking feature selection algorithm
3. Result of Subset feature selection algorithm

Table 8.3 shows the evaluation results for the different machine learning algorithms when trained and evaluated using the three different feature sets. The table presents the three evaluation metrics *Correlation*, *MAE* and *RAE* that are described in Section 7.4. It is sorted by MAE.

The best performing machine learning approach for the first diagram filtering was the *RandomForest* algorithm with no feature selection. It achieved a correlation of 0.66, an MAE of 0.56 and an RAE of 72.47%.

| Algorithm | Feature selection | Correlation | MAE | RAE |
|---|---|---|---|---|
| RandomForest | None | 0.66 | 0.56 | 72.47% |
| M5P | None | 0.62 | 0.60 | 76.57% |
| RandomForest | Ranking | 0.61 | 0.60 | 76.67% |
| M5P | Ranking | 0.60 | 0.60 | 76.74% |
| M5Rules | Ranking | 0.60 | 0.60 | 76.84% |
| M5Rules | None | 0.61 | 0.60 | 77.03% |
| RandomForest | Subset | 0.60 | 0.60 | 77.64% |
| M5P | Subset | 0.60 | 0.61 | 77.82% |
| SMOreg | None | 0.60 | 0.61 | 78.23% |
| LinearRegression | None | 0.61 | 0.61 | 78.31% |
| M5Rules | Subset | 0.59 | 0.61 | 78.63% |
| GaussianProcesses | None | 0.60 | 0.61 | 78.70% |
| SMOreg | Ranking | 0.57 | 0.62 | 79.09% |
| DecisionTable | None | 0.57 | 0.62 | 79.16% |
| DecisionTable | Ranking | 0.57 | 0.62 | 79.16% |
| DecisionTable | Subset | 0.57 | 0.62 | 79.16% |
| SMOreg | Subset | 0.57 | 0.62 | 79.28% |
| LinearRegression | Ranking | 0.58 | 0.62 | 79.76% |
| LinearRegression | Subset | 0.57 | 0.62 | 79.92% |
| GaussianProcesses | Ranking | 0.57 | 0.62 | 80.19% |
| GaussianProcesses | Subset | 0.57 | 0.62 | 80.21% |
| REPTree | Ranking | 0.56 | 0.63 | 81.44% |
| REPTree | Subset | 0.55 | 0.64 | 81.93% |
| MultilayerPerceptron | Subset | 0.56 | 0.64 | 82.15% |
| REPTree | None | 0.54 | 0.64 | 82.64% |
| SimpleLinearRegression | None | 0.48 | 0.66 | 85.17% |
| SimpleLinearRegression | Ranking | 0.48 | 0.66 | 85.17% |
| SimpleLinearRegression | Subset | 0.48 | 0.66 | 85.17% |
| MultilayerPerceptron | Ranking | 0.50 | 0.67 | 85.50% |
| DecisionStump | None | 0.49 | 0.67 | 85.61% |
| DecisionStump | Ranking | 0.49 | 0.67 | 85.61% |
| DecisionStump | Subset | 0.49 | 0.67 | 85.61% |
| MultilayerPerceptron | None | 0.48 | 0.70 | 90.34% |
| RandomTree | None | 0.48 | 0.72 | 93.10% |
| RandomTree | Ranking | 0.44 | 0.79 | 100.97% |
| RandomTree | Subset | 0.40 | 0.79 | 101.36% |

**Table 8.3:** Machine learning approach evaluation for the first diagram filtering.

## 8.2 Second diagram filtering

This sections presents the results with respect to **RQ1.2** and **RQ1.1** using the diagram data set obtained by performing the second diagram filtering described in Section 4.2.2.

### 8.2.1 Feature selection

Feature selection was performed using both the ranking and the subset feature selection algorithms described in Section 7.1.

Table 8.4 shows the results for the ranking algorithm. The correlation between the labels and all the features respectively are presented, sorted by absolute correlation.

Table 8.5 shows the results for the subset algorithm. The best found subset of features is presented, in no particular order. It shows that the best found subset consists of six of the 16 features.

| ID | Feature | Correlation | Absolute correlation |
|---|---|---|---|
| F9 | Longest line | -0.49 | 0.49 |
| F7 | Line length | -0.36 | 0.36 |
| F6 | Rectangle orthogonality | 0.36 | 0.36 |
| F8 | Line length variation | -0.30 | 0.30 |
| F2 | Crossing angles | 0.25 | 0.25 |
| F3 | Line bends | -0.22 | 0.22 |
| F14 | Rectangle proximity | -0.20 | 0.20 |
| F15 | Rectangle size | -0.20 | 0.20 |
| F4 | Line angles | -0.17 | 0.17 |
| F1 | Line crossings | -0.16 | 0.16 |
| F16 | Rectangle size variation | -0.12 | 0.12 |
| F13 | Rectangle distribution | 0.09 | 0.09 |
| F11 | Rectangle coverage | 0.08 | 0.08 |
| F12 | Aspect ratio | 0.05 | 0.05 |
| F5 | Line orthogonality | 0.03 | 0.03 |
| F10 | Shortest line | 0.00 | 0.00 |

**Table 8.4:** Feature selection for the second diagram filtering using correlation based ranking.

| ID | Feature |
|---|---|
| F16 | Rectangle size variation |
| F7 | Line length |
| F9 | Longest line |
| F8 | Line length variation |
| F2 | Crossing angles |
| F6 | Rectangle orthogonality |

**Table 8.5:** Feature selection for the first diagram filtering using the subset algorithm.

### 8.2.2 Machine learning approaches

The machine learning algorithms presented in Section 7.3 were trained and evaluated using the general approaches described in Section 7.2 on three different feature sets:

1. All features
2. Result of ranking feature selection algorithm
3. Result of Subset feature selection algorithm

Table 8.6 shows the evaluation results for the different machine learning algorithms when trained and evaluated using the three different feature sets. The table presents the three evaluation metrics *Correlation*, *MAE* and *RAE* that are described in Section 7.4. It is sorted by MAE.

The best performing machine learning approach for the second diagram filtering was the *RandomForest* algorithm with no feature selection. It achieved a correlation of 0.65, an MAE of 0.57 and an RAE of 73.63%.

| Algorithm | Feature selection | Correlation | MAE | RAE |
|---|---|---|---|---|
| RandomForest | None | 0.65 | 0.57 | 73.63% |
| M5P | None | 0.64 | 0.58 | 74.89% |
| RandomForest | Ranking | 0.63 | 0.59 | 75.68% |
| M5Rules | None | 0.63 | 0.59 | 75.70% |
| RandomForest | Subset | 0.62 | 0.59 | 76.27% |
| M5P | Ranking | 0.61 | 0.59 | 76.31% |
| M5P | Subset | 0.61 | 0.60 | 76.76% |
| M5Rules | Ranking | 0.61 | 0.60 | 76.88% |
| M5Rules | Subset | 0.60 | 0.60 | 77.37% |
| LinearRegression | None | 0.61 | 0.61 | 77.93% |
| GaussianProcesses | None | 0.60 | 0.61 | 78.05% |
| SMOreg | None | 0.59 | 0.61 | 78.38% |
| SMOreg | Subset | 0.57 | 0.62 | 79.35% |
| REPTree | None | 0.58 | 0.62 | 79.36% |
| LinearRegression | Subset | 0.58 | 0.62 | 79.36% |
| GaussianProcesses | Subset | 0.57 | 0.62 | 79.60% |
| SMOreg | Ranking | 0.57 | 0.62 | 79.69% |
| LinearRegression | Ranking | 0.57 | 0.62 | 80.25% |
| GaussianProcesses | Ranking | 0.56 | 0.63 | 80.63% |
| REPTree | Ranking | 0.57 | 0.63 | 80.98% |
| REPTree | Subset | 0.56 | 0.63 | 81.78% |
| DecisionTable | Ranking | 0.53 | 0.64 | 82.59% |
| DecisionTable | Subset | 0.53 | 0.64 | 82.59% |
| DecisionTable | None | 0.51 | 0.65 | 83.16% |
| DecisionStump | None | 0.50 | 0.66 | 84.78% |
| DecisionStump | Ranking | 0.50 | 0.66 | 84.78% |
| DecisionStump | Subset | 0.50 | 0.66 | 84.78% |
| SimpleLinReg | None | 0.48 | 0.66 | 85.16% |
| SimpleLinReg | Ranking | 0.48 | 0.66 | 85.16% |
| SimpleLinReg | Subset | 0.48 | 0.66 | 85.16% |
| MultilayerPerceptron | Subset | 0.48 | 0.69 | 88.48% |
| MultilayerPerceptron | Ranking | 0.46 | 0.69 | 88.89% |
| RandomTree | None | 0.49 | 0.71 | 92.05% |
| MultilayerPerceptron | None | 0.44 | 0.77 | 99.20% |
| RandomTree | Ranking | 0.46 | 0.78 | 99.94% |
| RandomTree | Subset | 0.38 | 0.80 | 103.45% |

**Table 8.6:** Machine learning approach evaluation for the second diagram filtering.

## 8.3 Final results

This section presents the final results for the research questions. For **RQ1.2**, the best performing machine learning approach for evaluating layout quality is presented. For **RQ1.1**, an attempt to rank the features by importance based on the feature selection that was performed is presented.

### 8.3.1 Layout quality evaluator performance

The best performing machine learning approach for the two iterations respectively have very similar performance. They have the same correlation MAE when rounded to two decimals. The one using the first diagram filtering has a slightly lower RAE, which also means it has a lower MAE, which gives it the advantage. This makes the best approach overall the following:

| Algorithm | Feature selection | Filtering | Correlation | MAE | RAE |
|---|---|---|---|---|---|
| RandomForest | None | First | 0.66 | 0.56 | 72.47% |

Figure 8.1 shows the predictions made by this model. The X-axis represents the labeled layout quality and the Y-axis represents the predicted layout quality. Each cross represents a diagram in the data set and the size of the crosses represents the size of the prediction errors.

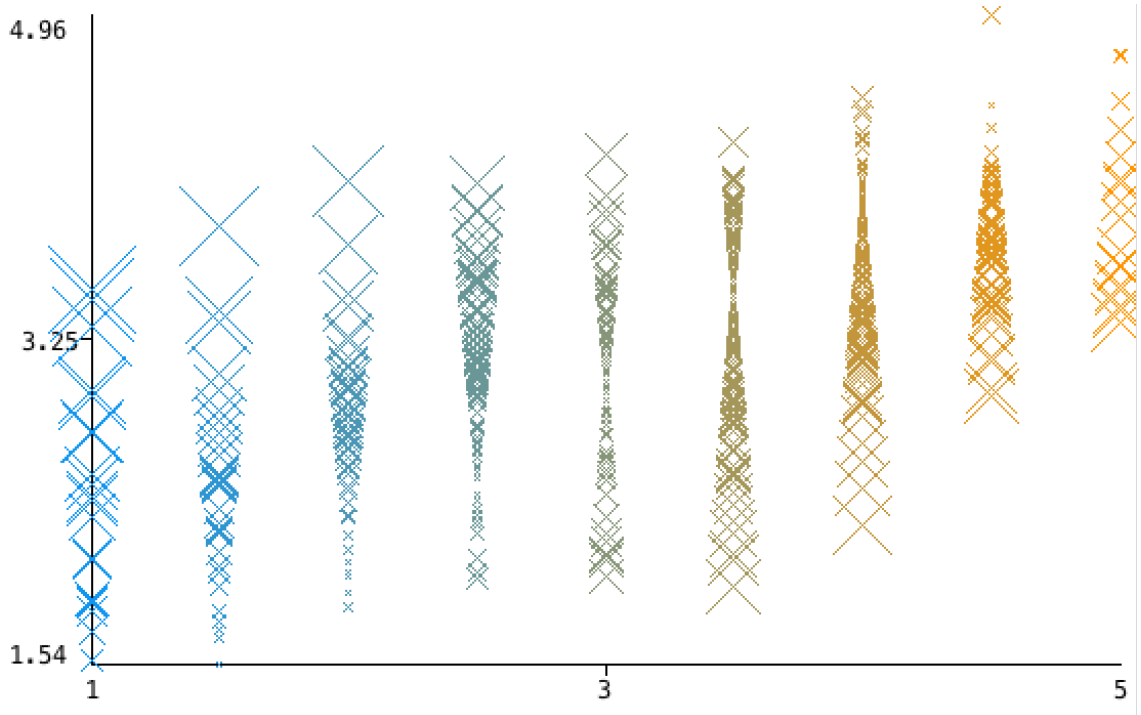


**Figure 8.1:** Predictions of the layout quality of diagrams made by the best performing machine learning approach.

Table 8.7 visualizes the predictions in a slightly different way, using a confusion matrix. A confusion matrix is usually used for visualizing predictions in classification problems and shows the distributions of predictions for each of the classes. As regression approaches, and not classification, were used in this study, the layout quality predictions are rounded to the nearest .5, which gives nine classes between 1.0 and 5.0. The rows in the table represent the actual classes that were labeled and each column shows how many diagrams that were classified into the corresponding class. We can for example see that 49 diagrams that were labeled with a layout quality of 3.5 were predicted to have a layout quality of 3.0. Larger quantities are highlighted with darker colors.

| | | Predicted layout quality | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | **1.0** | **1.5** | **2.0** | **2.5** | **3.0** | **3.5** | **4.0** | **4.5** | **5.0** |
| Actual layout quality | **1.0** | 0 | 2 | 7 | 6 | 4 | 2 | 0 | 0 | 0 |
| | **1.5** | 0 | 4 | 9 | 11 | 4 | 2 | 1 | 0 | 0 |
| | **2.0** | 0 | 0 | 8 | 13 | 19 | 6 | 1 | 0 | 0 |
| | **2.5** | 0 | 0 | 5 | 19 | 32 | 19 | 8 | 0 | 0 |
| | **3.0** | 0 | 0 | 5 | 15 | 35 | 28 | 6 | 0 | 0 |
| | **3.5** | 0 | 0 | 3 | 14 | 49 | 61 | 25 | 0 | 0 |
| | **4.0** | 0 | 0 | 0 | 4 | 20 | 53 | 56 | 6 | 0 |
| | **4.5** | 0 | 0 | 0 | 0 | 4 | 26 | 37 | 4 | 1 |
| | **5.0** | 0 | 0 | 0 | 0 | 1 | 8 | 7 | 4 | 0 |

**Table 8.7:** Confusion matrix.

### 8.3.2 Most important features

Table 8.8 shows a ranking of the features by a calculated score. The correlation based feature selection ranks the features according to their correlation with the labeled layout quality. A joint ranking for the two diagram filtering iterations is done by adding the results shown in Tables 8.1 and 8.4 per feature. This ranking gives the 16 features a score between 1 and 16, where the feature with the score 1 is the one with the highest correlation. This score is presented in the *Ranking* column in Table 8.8. For the subset feature selection, the features are put into three different groups. The first group contains features that are selected in both diagram filtering iterations, the second group contains features that are selected in one of the iterations, and the third group contains features that are selected in neither of the iterations. The selected features for the two iterations are shown in Tables 8.2 and 8.5 respectively. The three groups are given a score so that both feature selection algorithms are weighted equally towards the total score. For example, the first group contains four features. Thereby, the features in that group are given the score $\frac{1+2+3+4}{4} = 2.5$. The table is sorted by the total score, which is the sum of the scores for the two feature selection algorithms.

| ID | Feature | Aesthetic | Ranking | Subset | Score |
|---|---|---|---|---|---|
| F9 | Longest line | A4 | 1 | 2.5 | 3.5 |
| F6 | Rectangle orthogonality | A3 | 3 | 2.5 | 5.5 |
| F7 | Line length | A4 | 2 | 5.5 | 7.5 |
| F2 | Crossing angles | A1 | 5 | 2.5 | 7.5 |
| F8 | Line length variation | A4 | 4 | 5.5 | 9.5 |
| F16 | Rectangle size variation | A10 | 11 | 2.5 | 13.5 |
| F3 | Line bends | A2 | 6 | 11.5 | 17.5 |
| F14 | Rectangle proximity | A8 | 7 | 11.5 | 18.5 |
| F15 | Rectangle size | A10 | 8 | 11.5 | 19.5 |
| F4 | Line angles | A3 | 9 | 11.5 | 20.5 |
| F1 | Line crossings | A1 | 10 | 11.5 | 21.5 |
| F13 | Rectangle distribution | A8 | 12 | 11.5 | 23.5 |
| F11 | Rectangle coverage | A5 | 13 | 11.5 | 24.5 |
| F12 | Aspect ratio | A5 | 14 | 11.5 | 25.5 |
| F5 | Line orthogonality | A3 | 15 | 11.5 | 26.5 |
| F10 | Shortest line | A4 | 16 | 11.5 | 27.5 |

**Table 8.8:** Ranking of features.

# 9
# Discussion

In this chapter, the found results are discussed and compared to results of prior work. Potential threats to the validity of this study, and how the threats are mitigated, are also discussed.

## 9.1 Discussion of results

The results that are presented in Chapter 8 are discussed in relation to the research questions stated in Section 1.4. The usability and flaws of the results are discussed and, where applicable, they are compared to previous related work.

### 9.1.1 RQ1: Evaluator performance

The best performing machine learning approach (Section 8.3.1) has an MAE (described in Section 7.4) of 0.56. This means that the predicted layout quality of a previously unseen diagram differs by 0.56 on average from the labeled layout quality of the diagram. This is proportional to an RAE (described in Section 7.3) of 72.47% for this data set, meaning that the absolute error of a prediction of a diagram is on average 72.47% of the absolute error between the mean labeled layout quality of the data set and the labeled layout quality of the diagram. In other words, the performance of the evaluator is significantly better than simply predicting the sample mean for every diagram. This means that the machine learning approach was able to extract valuable information from the features that helps predicting layout quality. The correlation coefficient was 0.66, indicating that 66% of the variance in layout quality can be explained by the trained model. This is a strong correlation according to Evans' guidelines [30], which are presented in Section 7.4.

Even though the results seem promising, they are not perfect. As seen in Section 5.1.3, the image processing has some flaws that can lead to incorrect calculation of features, which in turn can mislead the machine learning algorithms. Below, the three diagrams where the prediction error was the highest are shown. The left part of the figures shows the original diagram image, and the right part shows the representing elements that were extracted with the image processing. Rectangles are drawn in white and lines are drawn in green.

Figure 9.1 shows the diagram that had the highest prediction error. It was labeled with a layout quality of 1.0 and the machine learning model predicted a quality of

3.5, which gives a prediction error of 2.5. It can be seen that many of the lines were not found. Many of the undetected lines are long and have many bends, which are probable reasons for the layout quality being labeled low. These are features that are negatively correlated with layout quality, which is seen in Tables 8.1 and 8.4. Since the machine learning algorithm doesn't get the correct information for these features, it is misled to predict a higher layout quality than it should.

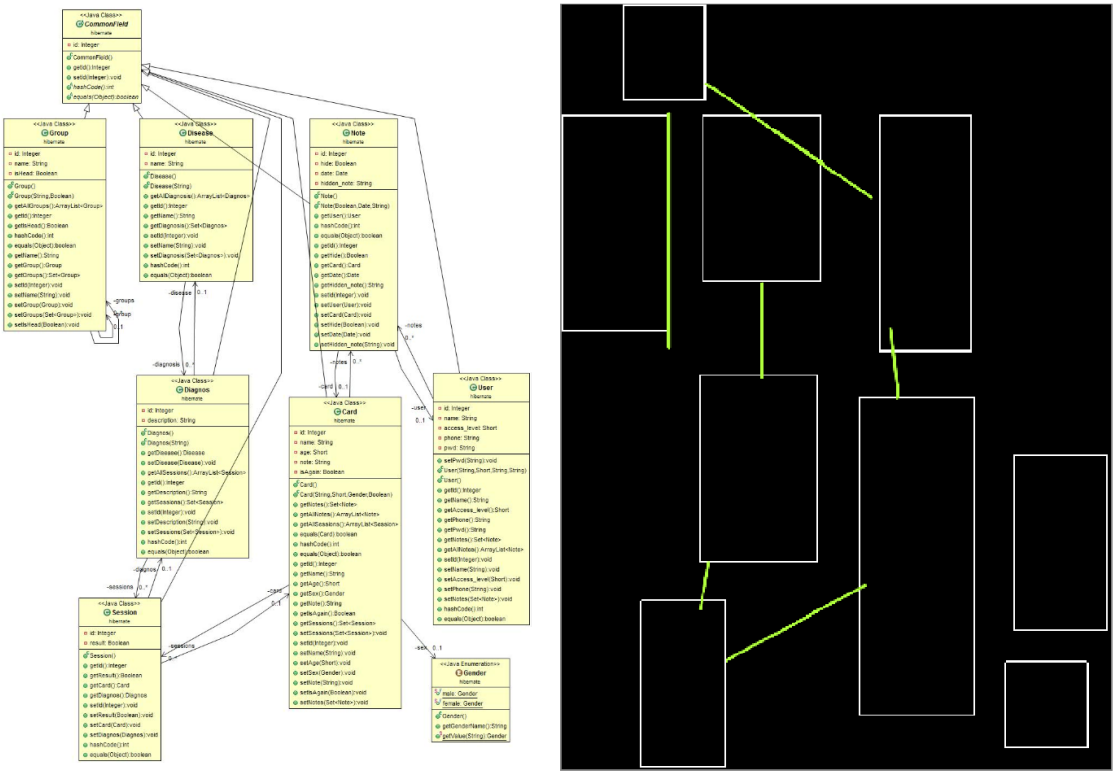

**Figure 9.1:** The diagram with the highest prediction error.

Figure 9.2 shows the diagram that had the second highest prediction error. It was labeled with a layout quality of 1.0 and the machine learning model predicted a quality of 3.4, which gives a prediction error of 2.4. In this case, the difficulty for the image processing to find lines is even more clear. Only one line is found, and that one line is not correctly detected. Many of the lines are crossing each other, which is a feature that is negatively correlated with layout quality, seen in Table 8.1 and 8.4. Another possible reason for the low layout quality label is that many lines are overlapping rectangles. This is something that a lot of literature does not encourage, which is seen in Section 2.1. Due to the image processing limitations described in Section 5.2, no feature regarding this was created, which means the machine learning algorithms don't get this information and can't take it into account when making predictions.

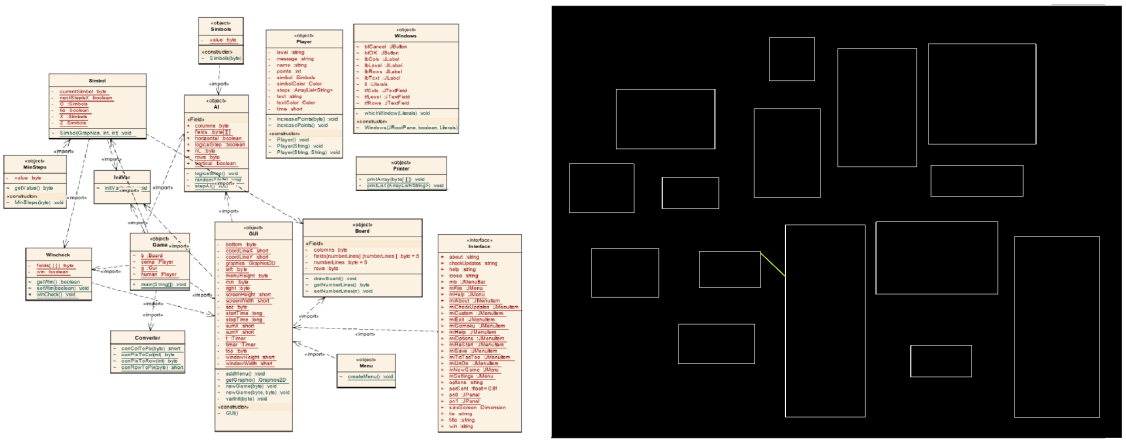

**Figure 9.2:** The diagram with the second highest prediction error.

Figure 9.3 shows the diagram that had the third highest prediction error. It was labeled with a layout quality of 1.5 and the machine learning model predicted a quality of 3.8, which gives a prediction error of 2.3. Here the image processing finds no lines at all and some of the rectangles are undetected as well. There are lots of long lines, line crossings and line bends that are not detected. This makes the machine learning algorithm predict a too high layout quality for the same reasons as in the two previously discussed diagrams.

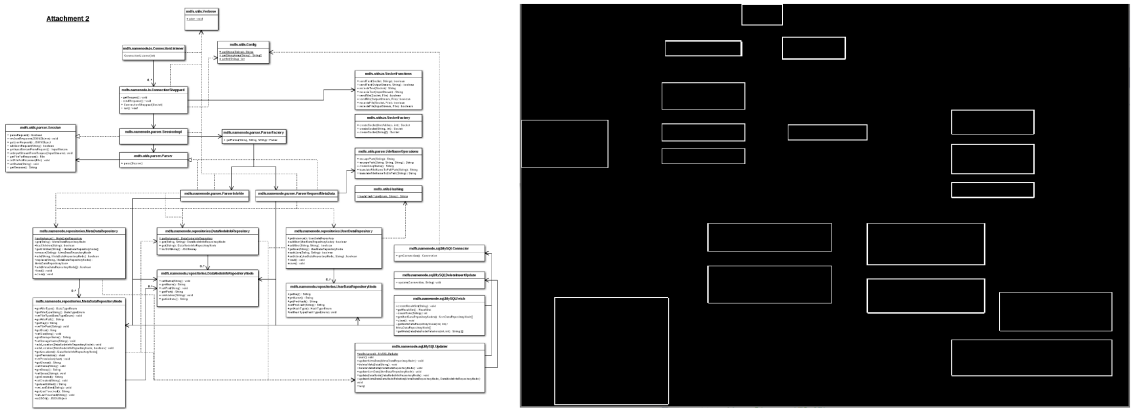

**Figure 9.3:** The diagram with the third highest prediction error.

A common denominator for the three diagrams with the highest prediction errors is that they all are labeled with a low layout quality, but predicted to have a medium quality. As discussed, a likely reason for this is the lack of detected lines. Many of the features regarding lines, including *Longest line*, *Line length*, *Line length variation*, *Line bends* and *Line crossings* are negatively correlated with layout quality, which is seen in Tables 8.1 and 8.4. This makes the machine learning algorithms falsely think that it is beneficial to have no lines at all or a few short lines. In addition, Tables 8.1, 8.2, 8.4 and 8.5 show that some of those features are among the most important ones when it comes to predicting layout quality, which makes the effect extra large.

To try to mitigate for these faulty predictions, all diagrams without any detected lines were filtered out for a second iteration of machine learning, as described in Section 4.2.2. However, this had no positive impact on the results. As seen in Section 8.3.1, the best performing machine learning approach used the first diagram filtering, which includes diagrams without any detected lines. A possible reason for this is that it was not enough to only filter out the diagrams with no detected lines. There are still diagrams with only one line where many lines are undetected, for example the diagram in Figure 9.2. Another possible reason is that the data set gets smaller when diagrams are filtered out. As always when it comes to machine learning, an as large as possible data set is desired. A general rule of thumb that sometimes is used within machine learning, is to have at least ten times more data points than the number of features. Including diagrams without any detected lines, 654 diagrams were used, and filtering those diagrams out, 612 diagrams remained. 16 features were calculated, which means both data sets well exceed the guideline and they are both probably sufficiently large to not have a large negative impact on the machine learning performance.

The illustrations of the machine learning predictions seen in Figure 8.1 and Table 8.7 show that the evaluator tends to make it's predictions quite close to the mean layout quality. There are not a lot of extreme, i.e. very high or very low, predictions. This makes sense, given the distribution of layout quality labels presented in Figure 6.3. However, there are quite a lot less extreme predictions than labels. A natural explanation for this is the label distribution itself. Since there are not very many diagrams with extreme labels, there is less information for a machine learner to gain about such diagrams. Another possible reason for the too few extremely low predictions could be that the reason for a diagram to get labeled with a low quality is that there is a lot of chaos in the diagram. There could for example be overlapping rectangles and lines, which is hard to find with image processing. This means that the machine learner might not be getting one of the most important pieces of information needed for knowing that the layout quality is bad.

Another interesting thing to note from the results is that in almost no cases, neither of the feature selection algorithms improved the machine learning performance. As seen in Section 8.3.1, the best performing machine learning approach did not use any feature selection. An explanation for this can be that many of the machine learning algorithms already use feature selection internally, which makes the preparatory feature selection unnecessary. It could also be that the features that were excluded with the feature selection actually provided a small amount of useful information, rather than being destructive noise.

### 9.1.2 RQ2: Most important aesthetics

To find the most important aesthetics for indicating layout quality, the results of the feature selection, which is described in Section 7.1, was used.

As seen in Table 8.8 in Section 8.3.2, many of the features that were found among

the most important ones in this study, including *Longest line*, *Line length* and *Line length variation* are related to aesthetic A4, *Line lengths*. This is a strong indication that line lengths are definitely important when it comes to layout quality. It should however be noted that as shown in Section 5.1.3, the image processing had difficulties detecting lines, which might make these results less valid. This is discussed further in Section 9.2.

Previous related work regarding the importance of different aesthetics in relation to layout quality was presented in Section 2.2.2. Below, the features that are used in this study are compared to the findings for related aesthetics in literature.

### Longest line (F9)

Eichelberger [7] ranked general line constraints, including line lengths, on the sixth place out of 14 aesthetics. Purchase et al. [5] did not find a significant effect for having short but not too short lines with regards to the understandability of graphs.

### Rectangle orthogonality (F6)

Purchase et al. [4] found a 61% preference for more orthogonal diagrams. Purchase [13] and Purchase et al. [5] did not find a significant effect for orthogonality with regards to the understandability graphs.

### Line length (F7)

Eichelberger [7] ranked general line constraints, including line lengths, on the sixth place out of 14 aesthetics. Purchase et al. [5] did not find a significant effect for having short but not too short lines with regards to the understandability of graphs. Ware et al. [6] did not find significant effects for average line length and total line length on the shortest path between two nodes in a graph with regards to the time needed to perceive the shortest path.

### Crossing angles (F2)

Ware et al. [6] did not find a significant effect for line crossing angles on the shortest path between two nodes in a graph with regards to the time needed to perceive the shortest path.

### Line length variation (F8)

Purchase et al. [5] did not find a significant effect for having uniform line lengths with regards to the understandability of graphs.

### Rectangle size variation (F16)

No previous work related to the importance of rectangle size variation was found.

### Line bends (F3)

Purchase et al. [12] found a significant effect and Purchase et al. [5] found a small, however significant, effect for minimizing line bends with regards to the understandability of graphs. Purchase [13] found a significant effect for minimizing line bends with regards to the correctness when carrying out graph understandability tasks. The effect also approaches significance with regards to the time needed to solve the tasks. Purchase et al. [4] found a 91% preference for fewer line bends. Ware et al. [6] found a significant effect for increasing the continuity of the shortest path between two nodes in a graph with regards to the time needed to perceive the shortest path. Eichelberger [7] ranked general line constraints, including line bends, on the sixth place out of 14 aesthetics. He also ranked graph drawing constraints, also including line bends, on the fourteenth place.

### Rectangle proximity (F14)

Eichelberger [7] ranked general node constraints, including distances between nodes, on the third place out of 14 aesthetics.

### Rectangle size (F15)

Eichelberger [7] ranked general node constraints, including minimizing class sizes, on the third place out of 14 aesthetics.

### Line angles (F4)

Purchase et al. [4] found a 61% preference for more orthogonal diagrams. Purchase [13] and Purchase et al. [5] did not find a significant effect for orthogonality with regards to the understandability graphs.

### Line crossings (F1)

Purchase et al. [12] found a significant effect for minimizing line crossings with regards to the understandability of graphs. Purchase [13] found a significant effect for minimizing line crossings with regards to both the time needed and the correctness when carrying out graph understandability tasks. Purchase et al. [4] found a 93% preference for fewer line crossings. Ware et al. [6] found a significant effect for minimizing line crossings on the shortest path between two nodes in a graph with regards to the time needed to perceive the shortest path. However, they did not find a significant effect for minimizing the total number of line crossings in the graph. Eichelberger [7] ranked avoiding line crossings on the fifth place out of 14 aesthetics.

### Rectangle distribution (F13)

Purchase et al. [5] did not find a significant effect for node distribution with regards to the understandability graphs.

### Rectangle coverage (F11)

No previous work related to the importance of rectangle coverage was found.

### Aspect ratio (F12)

Purchase et al. [4] found a 73% preference for narrower diagrams. Eichelberger [7] ranked graph drawing constraints, including aspect ratio, on the fourteenth place out of 14 aesthetics.

### Line orthogonality (F5)

Purchase et al. [4] found a 61% preference for more orthogonal diagrams. Purchase [13] and Purchase et al. [5] did not find a significant effect for orthogonality with regards to the understandability graphs.

### Shortest line (F10)

Eichelberger [7] ranked general line constraints, including line lengths, on the sixth place out of 14 aesthetics. Purchase et al. [5] did not find a significant effect for having short but not too short lines with regards to the understandability of graphs.

Table 9.1 shows a summary of the presence in literature of evaluation of layout aesthetics that are related to the features. [12], [13], [5] and [6] investigate whether certain aesthetics have a significant effect on the quality of the layout. Significant effects are denoted with a *y* in those columns in the table, effects that are approaching significance are denoted with an *a*, and non-significant effects are denoted with an *n*. [13] measures both correctness and time for the tasks in their experiment, which is why this column has two entrys. The first one represents significance for correctness and the second one represents significance for time. [4] measures people's preference of diagrams with a high degree of certain aesthetics over diagrams with a low degree of the aesthetics. The preference is represented by a percentage. [7] ranks 14 groups of aesthetics by their importance. *Line bends* occurs in the groups on both sixth and fourteenth place, which is why this row has two entrys.

| ID | Feature | [12] sig | [13] sig | [4] pref | [5] sig | [6] sig | [7] rank |
|---|---|---|---|---|---|---|---|
| F9 | Longest line | - | - | - | n | - | 6 |
| F6 | Rectangle orthogonality | - | n/n | 61% | n | - | - |
| F7 | Line length | - | - | - | n | n | 6 |
| F2 | Crossing angles | - | - | - | - | n | - |
| F8 | Line length variation | - | - | - | n | - | - |
| F16 | Rectangle size variation | - | - | - | - | - | - |
| F3 | Line bends | y | y/a | 91% | y | y | 6/14 |
| F14 | Rectangle proximity | - | - | - | - | - | 3 |
| F15 | Rectangle size | - | - | - | - | - | 3 |
| F4 | Line angles | - | n/n | 61% | n | - | - |
| F1 | Line crossings | y | y/y | 93% | - | y* | 5 |
| F13 | Rectangle distribution | - | - | - | n | - | - |
| F11 | Rectangle coverage | - | - | - | n | - | - |
| F12 | Aspect ratio | - | - | 73% | - | - | 14 |
| F5 | Line orthogonality | - | n/n | 61% | n | - | - |
| F10 | Shortest line | - | - | - | n | - | 6 |

* Significant on shortest path, but not the total number of line crossings.

**Table 9.1:** Evidence for feature importance in literature.

## 9.2 Validity threats

This section discusses the threats to validity of this research and what was done in order to mitigate those threats.

### Diagram data set

All diagrams from the data base that was used are from open source projects. There is a chance that such projects might not be representative of software projects in general. However, the criteria for a good layout should still be the same, independent of the nature of the diagram.

As described in Section 4.2, diagrams from the initial data set that were not desired to be used in this work were filtered out. This filtering was done manually by going over all diagrams one by one, and there could be a risk that mistakes were made during the filtering. To minimize this risk, the filtering was performed systematically and carefully by defining and applying clear filtering rules.

### Feature extraction

There is a risk that the calculated features don't accurately represent the aesthetics that they are supposed to represent. As seen in Section 5.1.3, the image processing is not perfect, which affects how the features are calculated. If, for example, the longest line in a diagram is not found, the *Longest line* feature will not be accurate.

Moreover, the actual calculation of the features might not represent the features in a perfect way. For some features, like *Longest line* and *Shortest line*, the calculation is straightforward. However, for features with a more complex calculation, like *Rectangle orthogonality*, it is possible that the calculations are not perfectly representative for the features. While these risks are indeed validity threats, it is likely that they have a negative impact, rather than a positive, on the results. Incorrectly calculated features probably creates noise that makes it harder for the machine learning algorithms to find correlations between the features and layout quality. Therefore, the results gained can be considered valid from this aspect.

There is also a possibility that there are other features than the extracted ones that are better indicators of layout quality. There could be semantic issues that need to be considered when constructing layouts, which is proposed by for example Purchase et al. in [5]. A limitation of this research stated in Section 1.5 was to not consider such semantics. Moreover, it was not feasible to extract any feature for some aesthetics, for example *Symmetry*, as described in Section 5.2. It is proposed as future work in Section 10.2 to investigate more features.

### Labeling

The labeling of the diagrams was done by asking people about their subjective perception of the layout quality of the diagrams, which means the labels are no absolute truth for layout quality. To mitigate this risk, each diagram was labeled by two different persons and the average of those was used as the label for the diagram. The data set is also considered large enough to balance potential differences between labelers out.

There is also a risk that the labels don't represent layout quality in terms of how easy it is to understand and work with the diagrams. This is a risk that had to be taken in order to label such a big data set that was used. To get more representative labels in this aspect, extensive user experiments would have to be carried out on each diagram, which was not possible in the time frame of this work.

The introduction of layout flaws options in the second round of labeling validation, described in Section 6.2.2, might have biased labelers to correlate the labeled quality with certain aesthetics. For example, when a labeler sees *crossing lines* as a layout flaw, they might look for crossing lines in the diagram and rate it higher if there are few crossing lines. This might increase the correlation between crossing lines and the labels. This was still considered a good tradeoff to take as it greatly simplified the discussions about the labeling strategy.

### Machine learning

The Likert scale that was used for labeling of diagrams is naturally an ordinal scale, while regression algorithms require an interval scale. As argued for in Section 6.1.2, the used scale can be considered an interval scale since it is symmetric around a

middle point and the distance between the options is intended to be the same.

As stated in Section 1.5, only machine learning algorithms provided by *WEKA* and their default configurations were investigated. There is a possibility that other machine learning approaches could give better performing models. However, this does not mean that the found results are invalid, only that they could potentially be improved.

# 10

# Conclusion

This chapter concludes the results and contributions of this work and suggests future work that can be done to improve or extend it.

## 10.1 Results and contributions

The primary goal of this work, which is stated in **RQ 1.1**, was to create an automatic layout quality evaluator for UML class diagram and to find out how well it performs. The results presented in Section 8.3.1 show that the evaluator is able to gain valuable information from diagrams and it clearly performs better than just predicting the sample mean layout quality. In quantitative terms, it performs with a correlation of 0.66 and an MAE of 0.56, which for this data corresponds to an RAE of 72.47%.

The developed evaluator can be implemented in a software tool where diagrams can be given as input and return a predicted layout quality as output. This can be useful for both evaluating individual diagrams without any real context, but maybe more importantly to evaluate the output of automatic diagram layout algorithms to compare different algorithms or versions of algorithms to see which one can create diagrams with the highest layout quality.

In addition to the primary goal, a secondary goal, stated in **RQ1.2**, was to find which layout aesthetics are most important for evaluating layout quality. Table 8.1 shows that there are features that are found to have a significant correlation with layout quality. Learning which features that are most important for evaluating layout quality of a diagram can provide guidelines for what aesthetics are most important to prioritize when constructing layouts and layout algorithms.

Finally, this study produced a data set of images together with a ground truth consisting of manually produced and validated labels describing the quality of the respective diagram, as well as a set of features extracted via image processing. This data set can be used for further data analysis regarding layout aesthetics and how these correlate with layout quality.

## 10.2 Future work

This section discusses directions in which the research described in this thesis can be improved, extended, or used by other research.

### Improve image processing

The image processing used for finding graphical elements in diagrams has some weaknesses that are likely to affect the results, which is seen in Section 5.1.3 and discussed in Section 9.1.1). The image processing method already uses well-known and established algorithms that might be difficult to improve. However, some improvements to how these algorithms and the outputs that they produce are used could possibly be made that could yield improved results.

### Improve machine learning

Another direction for improvement is adding more features. It may be possible to come up with additional features that capture valuable information for estimating layout quality. As seen in Section 5.2, some layout aesthetics were hard to turn into features, for example *Symmetry* and *Overlapping*. More advanced image processing algorithms that can extract more diagram elements more accurately could enable the creation of additional relevant features.

A limitation of this work stated in Section 1.5 is that only machine learning approaches and associated tailorable parameters that are provided by *WEKA* were used. To increase the likelihood of finding a fitting machine learning approach for this problem, more approaches need to be evaluated. This includes trying out more algorithms and more systematically optimize the parameters of these algorithms. An example could be to make an extensive investigation of neural networks, which can be tailored in a large number of ways. Another approach that could be interesting to investigate is using classification algorithms instead of regression. The diagrams could then be labeled with classes such as *good*, *bad* and perhaps *medium*, and classifiers could be trained to classify diagrams into those classes.

An alternative angle that could provide insight into how well the machine learning of this problem could perform is to only include diagrams where the image processing works well. This could for example be done by defining a threshold for the proportion of diagram elements that should be found in a diagram for it to be included in the data set. The diagrams where too few elements are found would be removed from the set.

### Improve labeling

As discussed in Section 9.2, the labeling of diagrams gives no absolute truth for their layout quality and there were some differences between the labelers regarding how some diagrams were labeled. This was mitigated by having each diagrams labeled by two labelers and use the average as the final label. Naturally, having

more labelers label each diagram would likely bring the labels closer to an imagined truth, and it would be interesting to see if this would affect the results. As suggested in Section 9.2, extensive usability experiments on the diagrams could maybe provide more reliable labels, but would require large amounts of time. It could be interesting to perform such experiments on a subset of the diagrams and look for correlations between the results of those and the labels produced in this work.

### Future Directions

This work focuses specifically on the evaluation of the layout of UML class diagrams. It would be interesting to apply the same approach to other types of diagrams. As different diagram types have different kinds of elements and are structured differently, it might be challenging to find a general approach that works well for all diagram types. At the same time, there are many commonalities in guidelines for the aesthetics of diagram layouts, such as minimizing crossing lines, which apply to multiple types of diagram. Perhaps some context dependent tailoring of this work would make it usable for other diagram types as well.

The results of this work can be used for developing new automatic layout algorithms. The aesthetics that were found most important could possibly be given more weight in these algorithms, and the layout quality evaluator can be used to evaluate the diagrams that are produced by the algorithms. This could open up the possibility for reinforcement learning for layout algorithms.

Another area where the evaluator could be useful is in the overall assessment of UML models. This topic has recently gotten increasing attention [31], [32], [33], but is not entirely solved. These types of assessments focus on the quality of the design, by looking at the quality of the decomposition and the conformance or violation of design principles, such as coupling. Such assessments, could be complemented by assessing the quality of the layout.

From the perspective of learning how to create diagrams with good layout, it would be more useful to get specific feedback on which aspect of a diagram could be improved, rather than only a grade which is the feedback produced by the current evaluator. Possibly, more specific feedback could be automatically created in addition to the current grade on a five point scale. This could be found by looking at which layout features of a diagram have a value that stands out compared to the average value of that feature for a collection of good diagrams.

# Bibliography


[1] H. Störrle, "On the impact of layout quality to understanding UML diagrams", in *2011 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, Sep. 2011, pp. 135–142.

[2] R. Hebig, T. H. Quang, M. R. V. Chaudron, G. Robles, and M. A. Fernandez, "The Quest for Open Source Projects That Use UML: Mining GitHub", in *Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems*, ser. MODELS '16, Saint-malo, France: ACM, 2016, pp. 173–183.

[3] H. C. Purchase, "Metrics for Graph Drawing Aesthetics", *Journal of Visual Languages & Computing*, vol. 13, no. 5, pp. 501–516, 2002.

[4] H. C. Purchase, J.-A. Allder, and D. Carrington, "User Preference of Graph Layout Aesthetics: A UML Study", in *Graph Drawing. GD 2000. Lecture Notes in Computer Science*, J. Marks, Ed., vol. 1984, Berlin, Heidelberg: Springer, 2001.

[5] H. C. Purchase, M. McGill, L. Colpoys, and D. Carrington, "Graph Drawing Aesthetics and the Comprehension of UML Class Diagrams: An Empirical Study", in *Proceedings of the 2001 Asia-Pacific Symposium on Information Visualisation - Volume 9*, ser. APVis '01, Sydney, Australia: Australian Computer Society, Inc., 2001, pp. 129–137.

[6] C. Ware, H. Purchase, L. Colpoys, and M. McGill, "Cognitive Measurements of Graph Aesthetics", *Information Visualization*, vol. 1, no. 2, pp. 103–110, 2002.

[7] H. Eichelberger, "Aesthetics of class diagrams", in *Proceedings First International Workshop on Visualizing Software for Understanding and Analysis*, Jun. 2002, pp. 23–31.

[8] H. Eichelberger, "Aesthetics and automatic layout of UML class diagrams", doctoralthesis, Universität Würzburg, 2005.

[9] H. Eichelberger and K. Schmid, "Guidelines on the aesthetic quality of UML class diagrams", *Information and Software Technology*, vol. 51, no. 12, pp. 1686 –1698, 2009.

[10] D. Sun and K. Wong, "On evaluating the layout of UML class diagrams for program comprehension", in *13th International Workshop on Program Comprehension (IWPC'05)*, May 2005, pp. 317–326.

[11] M. K. Coleman and D. S. Parker, "Aesthetics-based Graph Layout for Human Consumption", *Software: Practice and Experience*, vol. 26, no. 12, pp. 1415–1438, 1996.

[12] H. C. Purchase, R. F. Cohen, and M. James, "Validating graph drawing aesthetics", in *Graph Drawing*, F. J. Brandenburg, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 435–446.

[13] H. Purchase, "Which aesthetic has the greatest effect on human understanding?", in *Graph Drawing*, G. DiBattista, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 248–261.

[14] B. Karasneh and M. R. V. Chaudron, "Extracting UML models from images", in *2013 5th International Conference on Computer Science and Information Technology*, Mar. 2013, pp. 169–178.

[15] ——, "Img2UML: A System for Extracting UML Models from Images", in *2013 39th Euromicro Conference on Software Engineering and Advanced Applications*, Sep. 2013, pp. 134–137.

[16] T. Ho-Quang, M. R. V. Chaudron, I. Samúelsson, J. Hjaltason, B. Karasneh, and H. Osman, "Automatic Classification of UML Class Diagrams from Images", in *2014 21st Asia-Pacific Software Engineering Conference*, vol. 1, Dec. 2014, pp. 399–406.

[17] A. Dresch, D. Lacerda, and J. A. V. Antunes Jr, *Design Science Research: A Method for Science and Technology Advancement*. Sep. 2014.

[18] R. Wieringa, *Design Science Methodology for Information Systems and Software Engineering*. Jan. 2014, pp. 1–332.

[19] P. Johannesson and E. Perjons, *An Introduction to Design Science*. Jul. 2014, pp. 1–197.

[20] J. Canny, "A Computational Approach to Edge Detection", in *Readings in Computer Vision*, M. A. Fischler and O. Firschein, Eds., San Francisco (CA): Morgan Kaufmann, 1987, pp. 184 –203.

[21] P. J. Burt, "Fast filter transform for image processing", *Computer Graphics and Image Processing*, vol. 16, no. 1, pp. 20 –51, 1981.

[22] R. O. Duda and P. E. Hart, "Use of the Hough Transformation to Detect Lines and Curves in Pictures", *Commun. ACM*, vol. 15, no. 1, pp. 11–15, Jan. 1972.

[23] S. Suzuki and K. Abe, "Topological structural analysis of digitized binary images by border following", *Computer Vision, Graphics, and Image Processing*, vol. 30, no. 1, pp. 32 –46, 1985.

[24] U. Ramer, "An iterative procedure for the polygonal approximation of plane curves", *Computer Graphics and Image Processing*, vol. 1, no. 3, pp. 244 –256, 1972.

[25] R. Likert, "A technique for the measurement of attitudes", *Archives of Psychology*, vol. 22, no. 140, pp. 5 –55, 1932.

[26] D. V. Cicchetti, "Guidelines, criteria, and rules of thumb for evaluating normed and standardized assessment instruments in psychology", *Psychological Assessment*, vol. 6, no. 4, pp. 284 –290, 1994.

[27] T. K. Koo and M. Y. Li, "A Guideline of Selecting and Reporting Intraclass Correlation Coefficients for Reliability Research", *Journal of Chiropractic Medicine*, vol. 15, no. 2, pp. 155 –163, 2016.

[28] M. A. Hall, "Correlation-based Feature Subset Selection for Machine Learning", *Thesis submitted in partial fulfillment of the requirements of the degree of Doctor of Philosophy at the University of Waikato*, 1998.

[29] E. Frank, M. A. Hall, and C. J. Pal, *The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques"*, 4th ed., Morgan Kaufmann, 2016.

[30] J. D. Evans, *Straightforward statistics for the behavioral sciences.* Thomson Brooks/Cole Publishing Co, 1996.

[31] D. R. Stikkolorum, P. van der Putten, C. Sperandio, and M. Chaudron, "Towards Automated Grading of UML Class Diagrams with Machine Learning", in *Proceedings of the 31st Benelux Conference on Artificial Intelligence (BNAIC 2019) and the 28th Belgian Dutch Conference on Machine Learning (Benelearn 2019), Brussels, Belgium, November 6-8, 2019*, K. Beuls, B. Bogaerts, G. Bontempi, P. Geurts, N. Harley, B. Lebichot, T. Lenaerts, G. Louppe, and P. V. Eecke, Eds., ser. CEUR Workshop Proceedings, vol. 2491, CEUR-WS.org, 2019.

[32] Y. Boubekeur, G. Mussbacher, and S. McIntosh, "Automatic assessment of students' software models using a simple heuristic and machine learning", in *MoDELS '20: ACM/IEEE 23rd International Conference on Model Driven Engineering Languages and Systems, Virtual Event, Canada, 18-23 October, 2020, Companion Proceedings*, E. Guerra and L. Iovino, Eds., ACM, 2020, 20:1–20:10.

[33] W. Bian, O. Alam, and J. Kienzle, "Is automated grading of models effective?: assessing automated grading of class diagrams", in *MoDELS '20: ACM/IEEE 23rd International Conference on Model Driven Engineering Languages and Systems, Virtual Event, Canada, 18-23 October, 2020*, E. Syriani, H. A. Sahraoui, J. de Lara, and S. Abrahão, Eds., ACM, 2020, pp. 365–376.

# A

# Appendix 1 - Image processing validation

In Section 5.1.3, images showing the image processing validation results for some of the validation diagrams are presented. Here, the images for the rest of the diagrams are found.

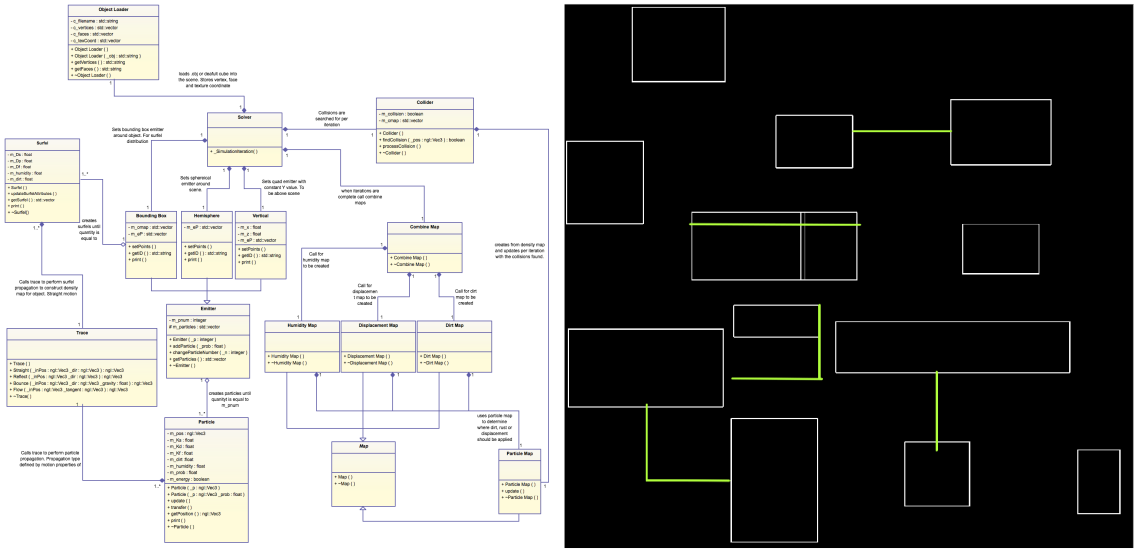

**Figure A.1:** Image processing validation for diagram 1.

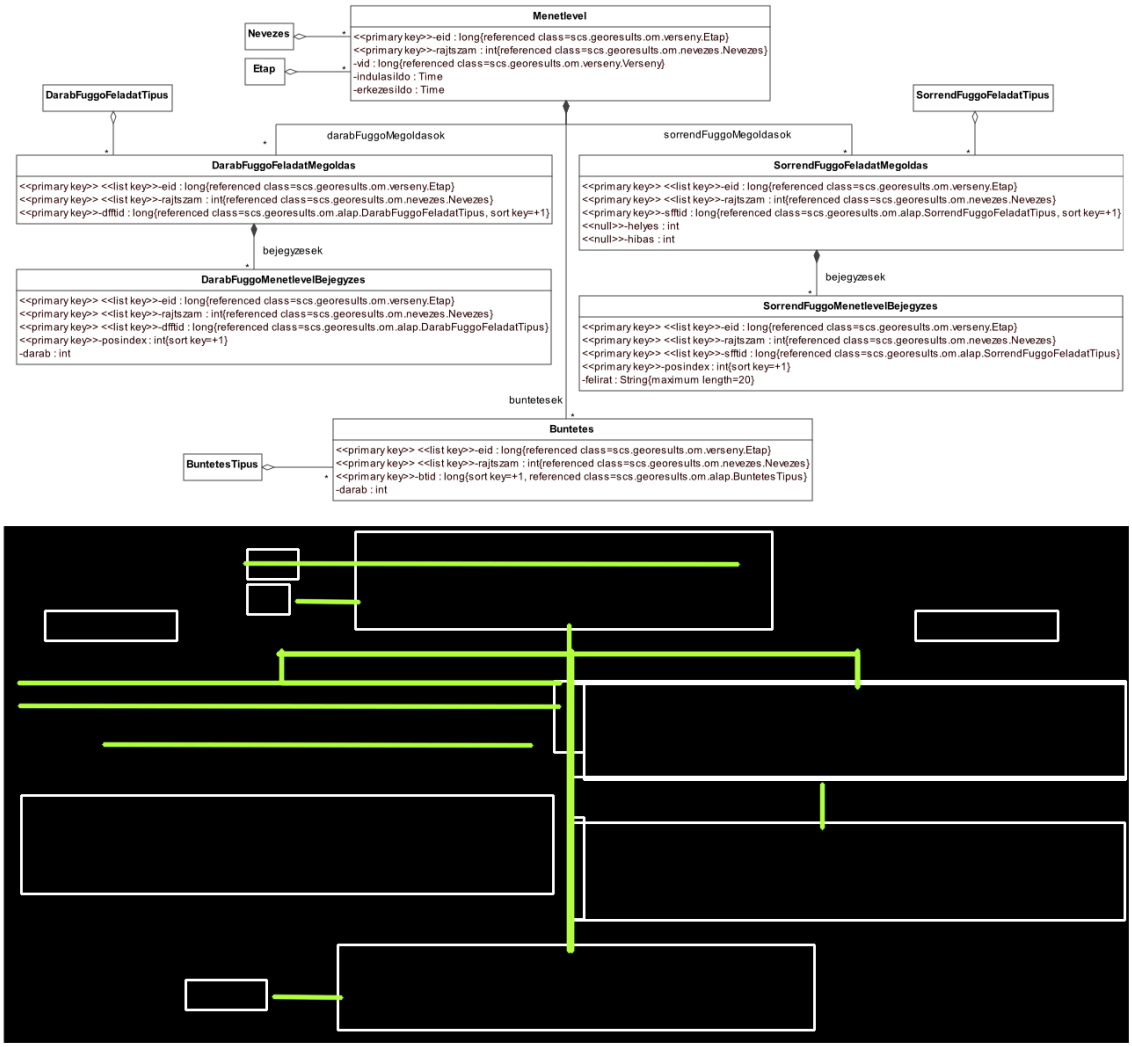


**Figure A.2:** Image processing validation for diagram 2.

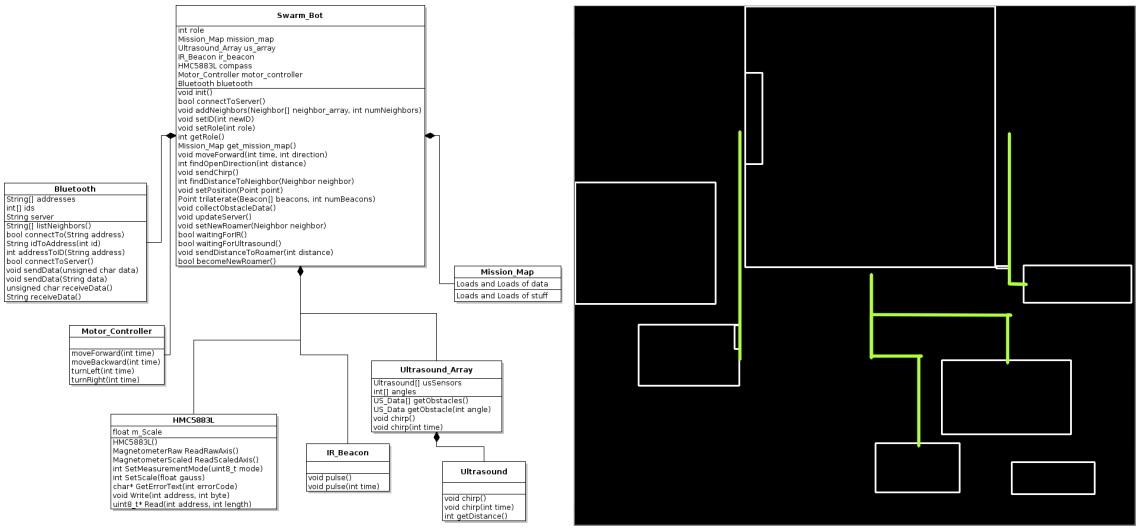


**Figure A.3:** Image processing validation for diagram 3.

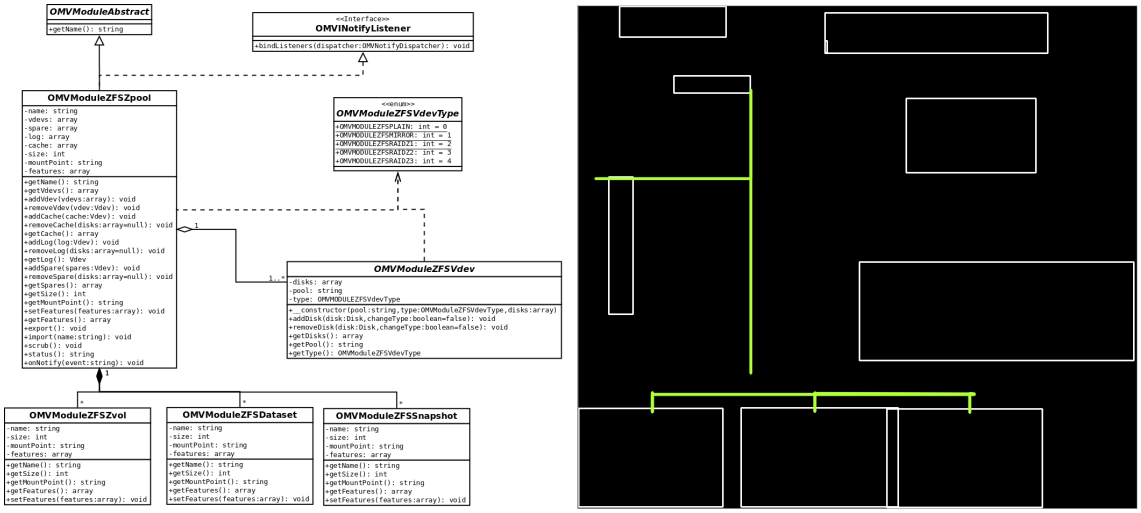


**Figure A.4:** Image processing validation for diagram 5.

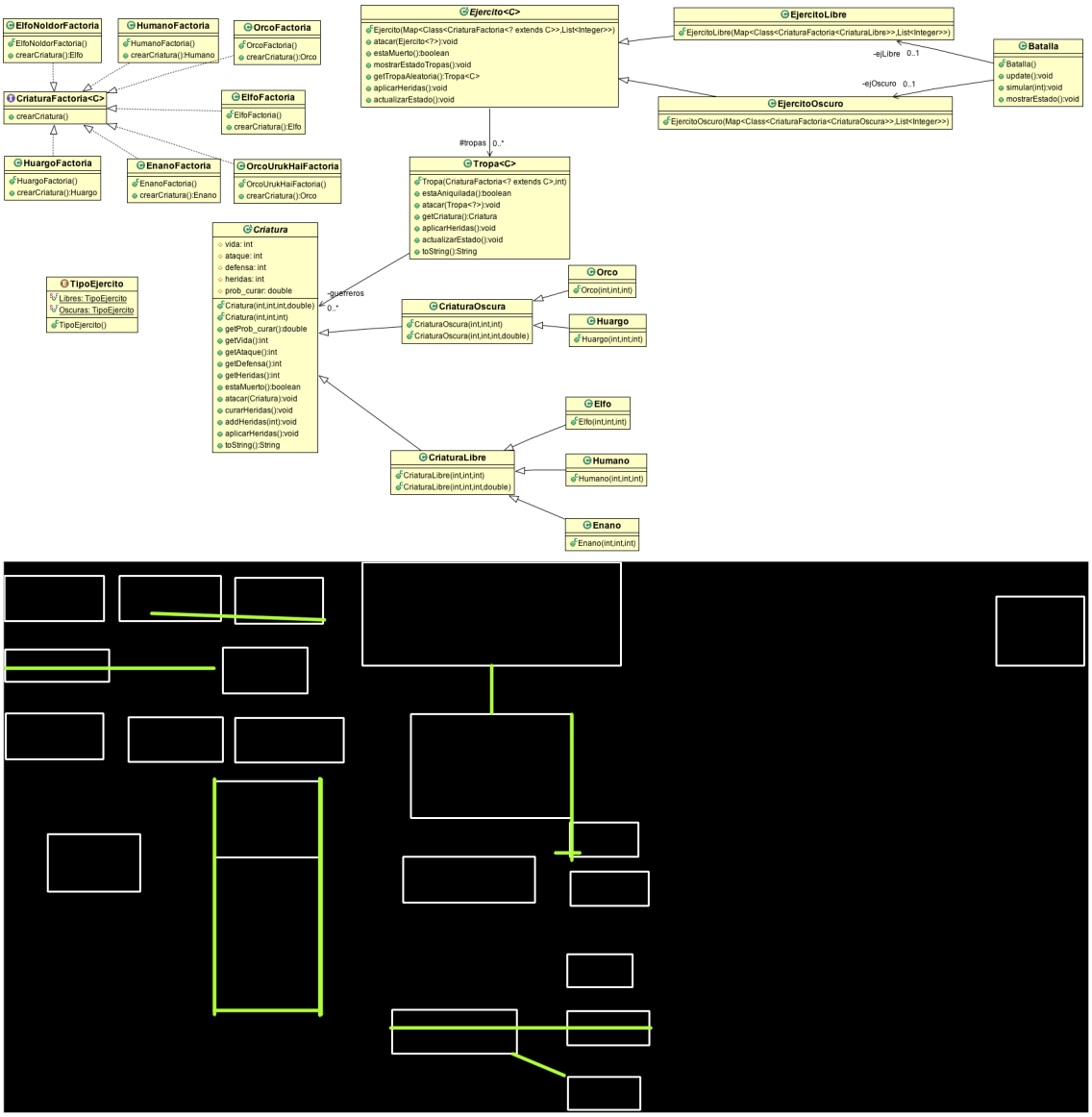


**Figure A.5:** Image processing validation for diagram 6.

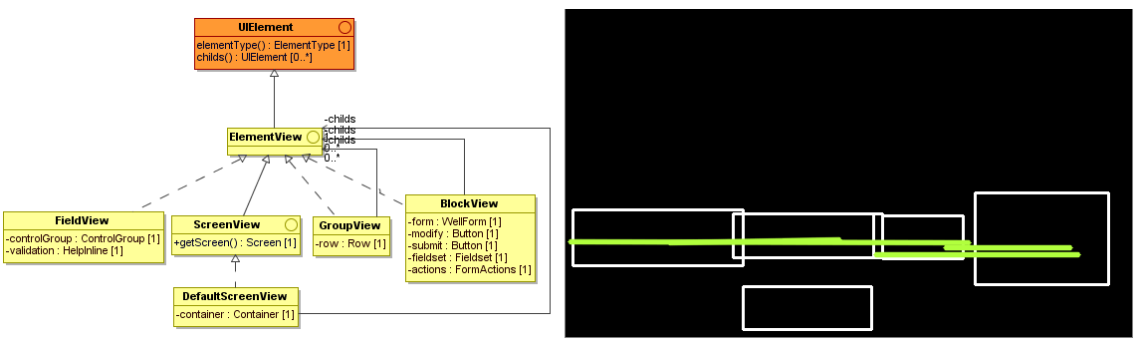


**Figure A.6:** Image processing validation for diagram 7.

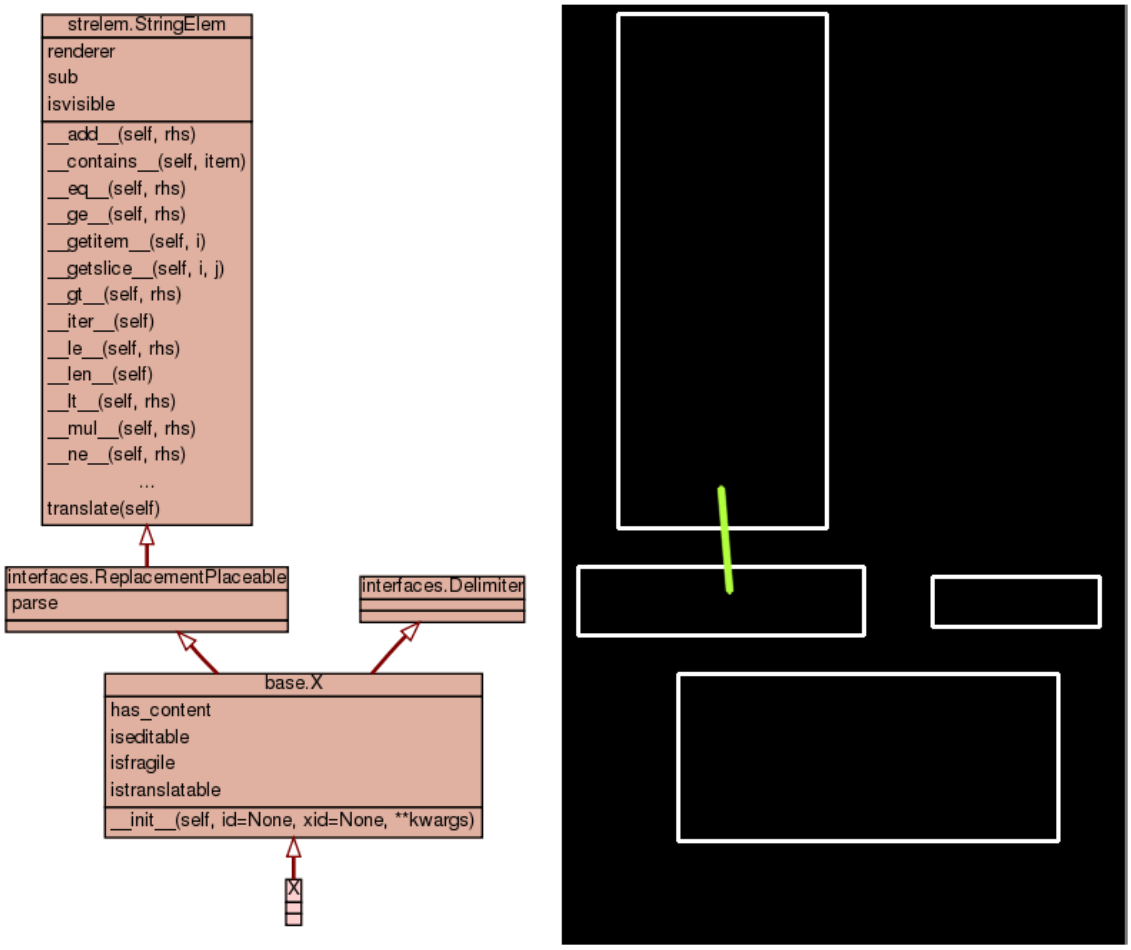


**Figure A.7:** Image processing validation for diagram 10.