Studies into interactive didactic approaches for learning software design using UML

Stikkolorum, D.R.

Chapter 6

Uncovering Common Difficulties of Students Learning Software Design

This Chapter further investigates the prelimary study that was conducted in Chapter 5. It describes an online experiment with the aim to reveal common difficulties and modelling strategies of students during class diagram design.

To gain more insight in their difficulties while performing a software design task, the students were asked to register their arising questions using a form in WebUML (Chapter 4). To gain more insight in the overall class design approach we compared students that use Breadth First strategies with those that use Depth First strategies in terms of grading overall assignment performance and diagram layout. Based on statistical analysis and diagram observations we noticed i) students seem to introduce noise by misunderstanding the assignment text ii) students have difficulties in choosing the right abstractions iii) good layout seems to lead to a good overall grade iv) the difference in grade between the Breadth First and Depth First strategy groups is not significant, however comparing the number of element moves, as possible measure for efficiency, indicates significant difference. We suggest follow-up studies to investigate the results in more detail.

This chapter is based on the following publication: Dave R. Stikkolorum, Truong Ho-Quang, Bilal Karashneh, and Michel R.V. Chaudron. Uncovering students' common difficulties and strategies during a class diagram design process: an online experiment. In *Educators Symposium 2015, co-located with the ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems*, 2015

6.1 Introduction

Novice software designers, students or newly-qualified professionals, struggle with different problems during their training tasks or first software development assignments. Especially in the context of software design. Several studies show students' difficulties that are related to UML syntax [80] (or other modelling languages) and reasoning, in special abstraction skills [69][113][141]. Educators are all aware of the different learning styles [67][154] students have. In order to adjust study programs to be more suitable for different types of learning styles we need to understand why students make the mistakes they make.

While doing an assignment, lecturers are not always present to give feedback on the steps students take or questions they have. Furthermore, students will not always proactively ask a question the moment they have one. We assume there is a relationship between the questions they ask themselves during a modelling task and how the design process takes form. Recording students' questions would provide more insight into their design processes - insight in students' considerations while creating a design or what typical decisions they make or think they have to make.

In Chapter 5 we described how students used our online UML editor WebUML[1] during a class diagram design task. WebUML is capable of logging UML class design activities (such as creating elements, movements, deletion etc.). We introduced our way to categorise the approach into depth first (DF) and breadth first (BF) strategies. These two strategies are part of a set of four strategies (see Table 6.1) that explains students' overall approach of making a software design. BF and DF were the dominant approaches. Students seem to construct their design by building classes with detail first and then associate (DF) or having an overall class framework first and then add detail (BF).

In order to do more in-depth research on the design steps students take and the problems they face we extended WebUML with a 'register-your-question-button' to record questions and comments students have during their modelling task.

A part of the students' approach probably consists of how they organise their diagrams, the quality of the layout. In this chapter we want to follow up on our previous study on a large scale to show whether the strategies we identified are common. We used 98 student-pairs to achieve this. Our main research questions were:

- RQ1: Do students have typical questions that arise during the development process of a software design?

[1] https://webuml.drstikko.nl

Strategy name	Activity sequence
Depthless Strategy:	class → associate
Depth First Strategy:	class → add detail → associate
Breadth First Strategy:	class → associate → add detail
Ad Hoc Strategy:	no structured approach

Table 6.1: *Different strategy types*

- RQ2: Does the Breadth First or Depth First strategy leads to a better grade for a class diagram design task?
- RQ3: Does the layout of the diagram influences the grade of a student's work?

The remainder of this chapter is organised as follows: in Section 6.2 we explore related work, in Section 6.3 we explain the method we used. After showing the results in Section 6.4 we discuss them in Section 6.5. Validity threats are discussed in Section 6.6. We conclude and identify future work in Section 6.7.

6.2 Related Work

Leung and Bolloju [80] relate common mistakes novice modellers make to 3 quality categories: syntax, pragmatic and semantic. And suggest this knowledge could be used for training purposes. They don't relate the categories to the design process or grading. Although syntax is important, our study focuses more on the semantic category.

The visualisation of log files is close to the process of mining research. There are a number of tools that provide different ways of presenting event-based logging. The following two are most related to our visualiser.

Song et al. [125] introduced the tool Dotted Chart that displays the events of the instances of a process as coloured dots on its time-lines. However, the tool is limited in its ability to present different steps of processes (e.g., creation, movement, renaming of a particular activity) on the same chart.

Claes et al. [26] followed up on the research in [125] and introduced a way to improve the visualisation. The tool PPMChart visualises modelling operations of one modeller

(one person) in the construction of a single process model as different coloured and shaped dots in a horizontal time-lines chart. The authors concluded that this approach of visualisation would provide audiences with different views at different levels of abstraction on the process modelling operations. However, the tool is not able to present multiple logging files concurrently or to detect modellers' patterns automatically.

Störrle addresses the relation between the understandability of a model and how well a design is organised. Making a good layout seems to be correlated with cognitive load. He mentions novice modellers benefit more than experts from good layout. The size of the layout seems to stand out. The larger, the more difficult the design is to understand. [144] [145]

Our approach of registering questions is arguably a 'think-aloud' [82] kind of approach. We are not aware of other research that uses that approach.

6.3 Method

In this section we describe our experimental approach. First we show the overall framework. Then we discuss the student participants and the class design task they had to model. Furthermore we explain the tools we used for collecting the data and performing the analysis. Subsequently, we discuss our approach in grading the students' models and analysing the results. More information is found in our technical report [130]

6.3.1 Overall Framework

Figure 6.1 shows the overall framework that was used for our online experiment.

Data collection was done online through a class design assignment performed by students from Uganda. The students were asked to submit their solution to a model task by using the pre-discussed web-based UML editor (WebUML). Besides the model file, students' modelling activities and questions during the modelling session were logged. After submitting their assignments the students' diagrams were graded by three experts in terms of overall task performance and layout.

The data analysis phase consisted of: i) an analysis of the registered student questions ii) an analysis of students' modelling strategies with a focus on the two emerged approaches Breadth First and Depth First. We developed a string pattern matching method to automatically detect the two approaches in the students' logging files iii)

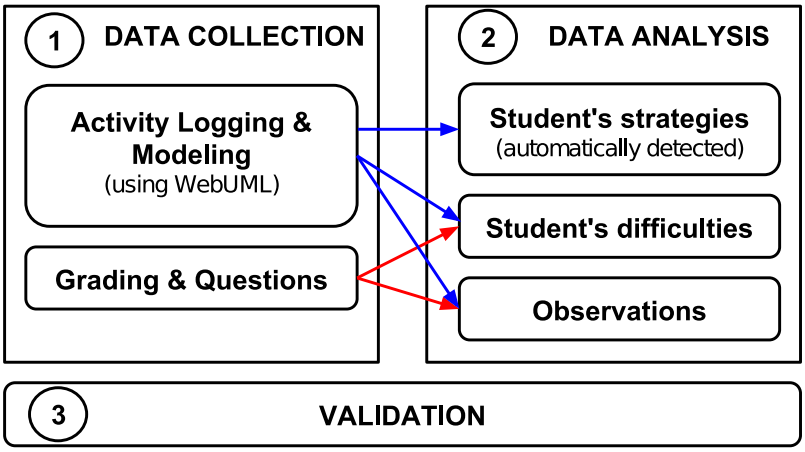

Figure 6.1: *Experimental framework*

observation of students' diagram solutions.

6.3.2 Participants

120 student-pairs were invited to perform a design task with our online class diagram editor. The students involved were 3rd year Software Engineering students following a Bachelor of Science degree program. They already followed courses in software design principles, UML and programming. The task was graded but not used as part of the course grade. The native language of the students is English. We did not choose to work with pairs on purpose. This is the students' university's approach for practical assignments.

6.3.3 Task

The students were asked to make a class design of a game that was presented online in a pdf-file. The text was a short (153 words) one paragraph description and was written in English:

The Modelling Task – a Tank Game[2] *In this task you will design a game with use of the UML class diagram. You do not need to use packages in this assignment. The description of the game is as follows: A player (user) controls a certain tank. This tank is a Panzer Tank, a*

[2] text: B. Karasneh

Centurion Tank or a Sherman Tank. They fire bullets and Tank shells. Bullets can be Metal, Silver or Gold bullets.

A tank moves around a world (level). The aim is to destroy all other tanks in the world. After a world has been completed the tank advances to the next world. A list of all the worlds visited is kept.

An entire game consists of 8 levels. A world contains a maximum of 20 tanks that compete for victory. Each tank remembers which tanks it has destroyed in the past. The score for each level is kept by a scoreboard that gets notified by the individual tanks each time an opponent is shot. The players control their tanks through an interface allowing for steering, driving (reverse / forward), switching ammo and firing.

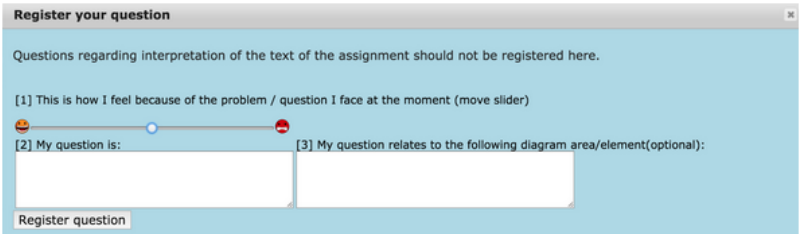

Figure 6.2: *Form used to register students' questions*

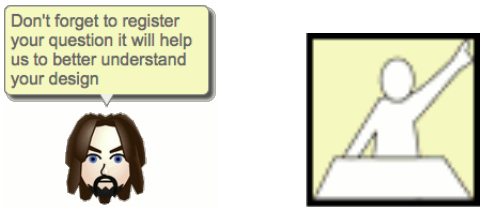

Figure 6.3: *Reminder and question button in WebUML*

The students were asked to press a button whenever they ran into difficulties or had a question or remark. With the web-form that popped up they were able to i) explain how frustrated they were at the moment that the difficulty arose and ii) record the question. Figure 6.2 shows this web-form.

The web-form was not meant for getting answers from a lecturer or assistant instantly, but only for registering the student's questions and/or difficulties. The online editor reminded the student every 5 minutes in a non-intrusive way: a message on the left side of the screen appeared and the question icon was highlighted (Figure 6.3). The experiment was part of a regular practical class of two hours, but they were allowed to

use more time. The students were asked to upload their work when they were finished.

6.3.4 Instrumentation

For the experiment we used three of our own tools. They were meant for creating the class designs, logging the activities and analysing the log files.

Modelling & Logging tool: WebUml is an online class diagram editor. WebUML is capable of logging students' activities in a comma separated file (explained in [136]) and saves the last version of the diagram in a xmi file and a png picture file. The files are compressed in a zip file and uploaded with an upload button. WebUML was designed to address a larger number of students independent of location or time. For this experiment we extended the log capabilities with the addition to register student questions. Students can register questions by filling in a form (explained in Subsection 6.3.3)

Analysis tools: For visual analysis we use LogViz[3]. LogViz is capable of displaying the designers' activities in WebUML over time. We can compare different log files (in this case different student-pairs) and measure times between activities. The tool is able to auto-classify the log files by the students' design strategies (Depth First, Breadth First, Depthless and Ad-hoc).

For gathering statistics, such as the number of creations we use StatLog[4]. StatLog reads WebUML's log files, counts all design activity occurrences in the logs and saves a table with the data that was found.

The registered student questions that were recorded in the log files were filtered out using general command line tools and regular expressions and then combined into one file.

6.3.5 Assignment Evaluation

The students' work was evaluated in three ways: i) every model was graded for overall task performance ii) every model was graded for layout quality iii) every activity log was automatically labelled with a strategy.

Grading: the grading was done during 4 grading sessions by 3 experts having more than 6 years of experience in software design and education. The experts considered

[3] LogViz - https://gitlab.com/truonghoquang/LogVisualizer
[4] StatLog - https://gitlab.com/stikkolorum/StatLog

two aspects to grade: i) task grade, how well does the diagram reflect the problem of the assignment? ii) the layout, how well is the diagram organised? For both of the grades a rubric was used (see Table 6.2).

Table 6.2: *Class Diagram Rubric for Grading Design Modelling*

Grade	Judgement, criteria description
1	The student does not succeed to produce a UML diagram related to the task. He/she is not able to identify the important concepts from the problem domain (or only a small number of them) and name them in the solution/diagram. The diagram is poor and not/poorly related to problem description with a lot of errors: high number of wrong uses of UML elements mostly no detail in the form of attributes or operations.
2	The student is not able to capture the majority of the task using the UML notation. Most of the concepts from the problem domain are not identified. The detail, in the form of attributes or operations, linked to the problem domain is low. Some elements of the diagram link to the assignment, but too much errors are made: misplaced operation / attributes non cohesive classes few operation or attributes are used.
3	The student is able to understand the assignment task and to use UML notions to partly solve the problem. The student does not succeed to identify the most important concepts. A number of logical mistakes could have been made. Most of the problem is captured (not completely clear) with some errors: missing labels on associations missing a couple important classes / operations / attributes Logical mistakes that could have been made: wrong use of different types of relationships wrong (non logical) association of classes.
4	The student captures the assignment requirements well and is able to use UML notations in order to solve the problem. Almost all important concepts from the problem are identified. Some (trivial) mistakes have been made: Just one or two important classes / operations / attributes are missing Design could have been somewhat better (e.g. structure, detail) if the richness of the UML (e.g. inheritance) was used.
5	Student efficiently and effectively used the richness of UML to solve the assignment. The problem is clearly captured from the description. concepts from the domain / task are identified and properly named The elements of the problem are represented by cohesive, separate classes (supports modularity) with a single responsibility In the problem domain needed attributes and operations are present Multiplicity is used when appropriate Naming is well done (consistent and according to UML standard) Aggregation / Composition / Inheritance is well used No unnecessary relationships (high coupling) are included.

The rubric consisted of a 5 point scale and the experts agreed on the rubric before the experiment started. In advance of the actual grading, a set of possible ideal solutions was discussed for calibration. The grading was done in two steps: first the assessors graded all diagrams separately, then they discussed the differences in grading and gave the diagram the final marks (task, layout) after consensus. Grading was done in batches of 25 class diagrams.

Student's strategy: students' strategies were automatically classified using a string pattern matching approach. Extracted from the log files, the creation activities of class diagram elements were constructed as a string of the 4 letters: C (represents CLASS), O (represents OPERATION), A (represents ATTRIBUTE) and R (represents

CCCCCCCAOCCCCOAOOOOCRRRAORRRRRRRRRRRRRRCRCRRRCRCROOAOAA

Figure 6.4: *String fetched from the log to determine a student's strategy*

associations/relationships between classes). Figure 6.4 shows an example of such a creation string. As an example, we show the regular expressions of the two most used strategies in Table 6.3. A log is labelled as the strategy that matches the longest string in the creation string.

Strategy	Activity order	Regular Expression
Breadth First	class → associate → add detail	[C]+[R|C]+[O|A]+
Depth First	class → add detail → associate	[C]+[O|A]+[R|C]+

Table 6.3: *Regular expressions used to fetch a strategy from the log file*

6.4 Results

In this section we present the results of our online experiment. First we explain the overall response and the questions students asked. Then we present the statistics of the class design log files combined with the experts' grades and modelling strategies that were identified. Finally, we summarise the observations that were made during the grading process of the class diagram designs.

6.4.1 Recorded Log Files - Overall Response

We recorded useful log files of 98 student-pairs. Although 100+ student-pairs participated in the experiment, some files were corrupted or incomplete.

6.4.2 Registered Questions

Out of the total response (N=98) 31 questions from 24 different student-pairs were registered during the experiment. From 7 student-pairs we received 2 questions. 1 pair asked 3 questions. The others asked 1 question. We divided the questions into the following 5 categories (questions can fit in multiple categories):

Category	Occurrences	Example question from respondents
Task Comprehension	7	How many users are allowed to play at a given time?
Tool Usage	16	How do I draw associations?
Tool Feedback	9	Why doesn't the tool support adjusting of the class in the event when the operation name is too long to fit in the fixed size?
UML/OO Comprehension	3	Could the different types of tanks be modeled as specializations of the tank class or as an attribute in the Tank class?
Notation/Syntax	7	How do you represent inheritance?

Table 6.4: *Categories of questions students asked*

- Task Comprehension: how well the student understood the task
- Tool Usage: Questions about the usage of the online tool.
- Tool Feedback: Remarks about or suggestions for improvement of the tool.
- UML/OO comprehension: Related to the UML and/or object orientation comprehension of the student.
- UML Syntax/Notation: questions about graphical representations of UML or other elements the student wanted to draw.

Each question was rated in sense of relevancy to the task on a 1 to 5 scale (no relevance - high relevance). For 1 question it was impossible to determine the relevance because of the unclear wording of the question. Both category overview and relevance rates are shown in tables 6.4 and 6.5. Most questions (16) were related on how to perform certain actions in the tool (Tool Usage). The lowest number was related to the comprehension of OO concepts and/or UML.

On the 'feeling' indicator 22 student-pairs responded neutral, 7 student-pairs felt bad/angry while recording questions, 2 pairs felt happy. Remarkable is that questions about UML syntax only links to neutral or positive feelings. They don't relate to angry feelings.

Relevance index	Occurrence	Example question from respondents
1	2	How do you connect many arrows together for inheritance?
2	8	It is hard to delete an attribute once its written
3	7	Is a scoreboard an attribute?
4	9	Where should we note our assumptions?
5	4	If scoreboard falls under class, what attributes can it take in this case?

Table 6.5: *Relevancy of recorded questions*

6.4.3 Statistics of the Logs and Evaluation Data

The log files consist of the recorded user data: time spent (in minutes) and the different modelling activities (in frequencies) such as creating UML elements (classes, attributes etc.). The dataset was extended with a number for the grade and a number for the layout evaluation. Table 6.6 shows the statistics summary of the logged data and evaluation grades. The total time row contains some extreme values. This is due to some students uploaded their work late in the evening or the next morning or due to technical errors. If we discard these cases we find a range from 11 - 400 minutes with mean = 92.66, sd = 83.06 and N=87.

Statistic	N	Mean	St. Dev.	Min	Max
grade	98	3.06	0.84	1	4
layout	98	3.27	0.65	2	5
totaltime	98	1,215,771	5,267,869	11	23,819,063
creates	98	70.54	28.71	16	168
sets	98	42.36	17.07	3	96
adds	98	25.59	15.18	3	87
moves	98	77.08	63.44	7	364
removes	98	11.10	11.05	0	71
readings	98	3.09	5.18	0	36
modellings	98	2.91	5.15	0	36
comments	98	0.32	0.65	0	3

Table 6.6: *Descriptives of all logged variables and evaluation*

6.4.4 Modelling Strategies

The extracted creation strings from the logs that were mentioned in Subsections 6.3.4 and 6.3.5 were explored using the regular expressions and manual analysis. We identified the two major groups: Depth First (N=43) and Breadth First (N=45). Also, a number of student approaches could be described as: 'Ad Hoc' (N=5) - they don't use a clearly observable pattern - and a 'Both' (N=2) group that seems to switch between Depth First and Breadth First. Some logs were labelled with 'U' (N=3). In this case there was no complete pattern recorded or just missing.

6.4.5 Group Comparison

To compare the performance of the student-pairs (the grade) grouped by the strategy they use (Depth First, Breadth First) we performed a Wilcoxon rank sum test[5] . We only compared the groups Breadth First (N=45) and Depth First(N=43). Although the mean of BF (grade=3.13) is higher than DF (grade=2.88) the Wilcoxon test did not show a significant difference between the two strategy groups (W = 1122, p = 0.1735).

If we compare the student-pairs in terms of the amount of moves grouped by strategy the Wilcoxon test does indicate a significant difference between BF and DF strategy student-pairs (W = 1354.5, p = 0.0013, meanBF = 92.38, meanDF = 59.26)

6.4.6 Correlation

At first glance no remarkable correlations were found in the dataset of the experiment. However, excluding very poor models did yield an interesting correlation. During the grading process we came across several class diagrams that were poor in the sense of a low number of classes and did not use richness of the UML (such as inheritance). These kind of diagrams most of the time scored high (≥3) on layout. There cannot be a lot of things wrong with the layout of poor diagrams, except from the alignment. The examination of the subset (N=66) that excluded such cases resulted in a correlation coefficient of 0.32 (p=0.009) between layout and grade. Which can be seen as a moderate positive correlation.

[5] https://stat.ethz.ch/R-manual/R-devel/library/stats/html/wilcox.test.html

6.4.7 Student Diagram Observations

During the process of grading the diagrams we discussed the common mistakes or additions students seem to make. In this subsection we summarise our observations.

Mistakes or Unidentified Elements *Reflexive associations (self associations)*: the task description should have triggered the students to use a reflexive association or with the help of an intermediate class. It seemed common not to notice this.

Wrong use of UML elements: a typical mistake for students is to use the wrong element for a certain purpose, such as using aggregation or composition when inheritance was meant.

Misplaced operations or attributes: students seem to have difficulties to identify the responsibility of a class and only save this class for this purpose (cohesion).

Forgotten elements: information that appears in the task is not represented in the solution (such as classes, operations, attributes etc.).

Concept as attribute instead of class: students have difficulties deciding between classes and attributes (abstraction).

'Loose' classes: a number of diagrams consisted of classes that did not have any relation with another class.

Addition of Elements Sometimes students felt the need to include non standard notation in the attribute fields, such as code notation or numbers instead of using the multiplicity element. Although it was not needed to include types (such as Int or String) a number of students added this to an attribute or operation. They also tend to include an 'id' as part of the class attributes. Although expected, from experience in classrooms, students seemed not to use too many associations per class (high coupling).

Grading While grading, some criteria seemed to be more important than others. For example, missing a clear relationship between two main classes was considered as a major design flaw, while the misdirection of relationships or missing a label are not considered as a big problem. At the moment our rubric does not contain this distinction

per level although the experts used it unconsciously.

During our discussions we realized we could add some more layout criteria for layout quality. From the experiment experience we propose: symmetry, the spread-out of the elements and diagram size. We did not take these into account while grading layout.

6.5 Discussion

In this section we discuss the results by answering the research questions.

6.5.1 Do students have typical questions that arise during the development process of a software design?

It is not in every student's nature to ask questions and it is often considered as something to avoid. Because of the number of questions (31 registered by 24 student pairs), we could not do any statistical analysis. We did identify typical categories: Task Comprehension, Tool Usage, Tool Feedback, UML/OO comprehension, UML Syntax/Notation. We observed that most of the questions were related to the tool. Tool use is discussed widely and often considered difficult in use [2]. For some students this likely seems to distract them from their tasks. We assumed our tool is very easy to use. The students were able to use it without any training other than a little practice in advance.

Students occasionally seem to seek for extra information that is not in the text, but at the same time it is not needed for the solution. 5 out of 7 task related questions were considered to be of low (2) relevance.

Surprisingly the number of questions related to UML/OO comprehension was low (3). This could be explained by the fact that the students had prior knowledge and were already in their 3rd academic year.

Based on the relevance number, most student-pairs seem to be capable of asking relevant (index 3-5) questions. Which points out, that doing this by the means of an online tool is a good approach.

Derived from our diagram observations, and supported by the students questions, choosing between attributes or classes (which addresses OO comprehension) seems to be a general difficulty for students.

6.5.2 Does the Breadth First or Depth First strategy leads to a better grade for a class design task?

From the statistical analysis we cannot conclude one of the approaches leads to a better grade. There is no significant difference. We are not yet convinced that both strategies can lead to diagrams of the same quality. Deeper investigation must be done. We could rerun the experiment with a different grade scale and see if it has a better spread. At the moment in the 5 point scale grades set the grades 1 and 5 were not represented that much which may has lead to the results we have now. Another option is to run the experiment with professionals and investigate if there is a bigger share of BF or DF strategies.

According to the statistical test there was a significant difference between the two strategies in terms of number of movements. If we interpret this as a measure for efficiency it suggests the DF could be more efficient than the BF approach.

In case both strategies can result in comparable quality models it still can be the case certain strategies cause certain difficulties. In our dataset we have not enough questions recorded to perform such an analysis.

6.5.3 Does the layout of the diagram influences the grade of a student's work?

Based on the moderate correlation of 0.32 we advise students to pay attention to create a nice layout in their overall design approach. 32% of a grade can explained by the layout. We assume a good layout not only helps the student to understand his/her own model, but also provides the lecturer better insights.

6.6 Threats to Validity

Although we evaluated the model and layout according to a rubric, they are still graded manually which introduces a bias. On one hand we tried to reduce this bias to determine the grade through discussion. On the other hand this same discussion could have lead to a milder evaluation.

To be sure the automated process of labelling the strategy worked the results were checked by hand by two experts.

We are aware of the fact we used pairs of students. The results might not represent individual students.

6.7 Conclusion and Future Work

In this chapter we presented our approach to uncover students difficulties and strategies while making a class design. We recorded the solutions, design activities and questions students have in a log file. A small number of students registered their questions from which a majority was relevant to their assignment. Based on the questions and diagram observations students introduce noise by adding unneeded elements and have difficulties in choosing between attributes and classes as representation of certain concepts from the assignment text.

In general the student-pairs clearly seem to use a DF or BF strategy but do not significantly differ in terms of grades. Both strategies could profile students and different profiles could lead to different difficulties during modelling. Reruns of the experiment in different settings could gain more insight. Also, collecting more student questions that are related to certain strategies could help us to explain our questions.

Comparing the amount of movements between BF and DF, the results suggest DF to be more efficient than the BF strategy.

Based on the moderate correlation we found between grade and layout, we assume that paying attention to a nice layout helps students to perform better on their modelling assignments.

In future research we continue to explore students' strategies and difficulties. Based on the results of the experiment in this chapter and future research we aim to develop educational programs in the field of software design that fit students' profiles better.