Part IV

Student Guidance and Feedback

Chapter 5

Strategies of Students Performing Design Tasks

This chapter shows the most common strategies students use to create class designs. We demonstrate our approach of logging students' modelling activities while doing a software design task. We developed our own online modelling editor 'WebUML' and visualisation tool 'LogViz' for the logging and interpretation of the log files. As follow-up, students filled-out a brief questionnaire about their time spent and difficulties in performing the task. The results show that the students use different strategies for solving the tasks. We divided these strategies into four main categories: Depthless, Depth First, Breadth First and Ad Hoc. Our results show that the Depth First strategy supports students to deliver models with a better layout and richness (detail). The questionnaire shows that students find that choosing the appropriate UML elements is a difficult and time consuming task. We want to use our insights to improve our educational programs and tools. In the future we want to test WebUML and LogViz in larger educational contexts.

This chapter is based on the following publication: Dave R. Stikkolorum, Truong Ho-Quang, and Michel R.V. Chaudron. Revealing students' UML class diagram modelling strategies with WebUML and LogViz. In *Software Engineering and Advanced Applications (SEAA), 2015 41st Euromicro Conference on*, pages 275–279. IEEE, 2015

5.1 Introduction

Lecturers are familiar with the fact that students learn in different ways. These learning styles are categorised by authors in various ways, such as: being a 'do-er','dreamer', a 'thinker', a 'decision maker' [67]. Others use: being 'meaning-oriented', 'reproduction-oriented', 'application-oriented' or 'non-oriented' [154]. By acknowledging the fact that student groups consist of a mixture of representatives of these styles, lecturers can organise their education to serve this combination of different learning styles. Especially in classroom settings, that usually consist of students with different styles, this is of course difficult to achieve, while outside the lecture rooms students would choose strategies that fit their style better.

In software design courses students deal with problem solving challenges. A substantial number of students have difficulties with software design tasks. It is plausible that, based on their learning style, students will use different strategies to solve their software design problems.

By understanding students' difficulties in designing software, we can develop better teaching methods and tools. In order to understand what type of problems students have during modelling tasks we monitored students' modelling activities during an average UML class diagram assignment. To analyse students' activities while working in these tasks, we developed an online UML class diagram editor that can log the actions students perform during a modelling task. In addition, we developed a tool that is capable of visualising the different actions a student performs, which makes the analysis easier and enables us to discover patterns that are likely more difficult to see without the visualisation.

In this chapter we answer the following research questions:

- RQ1: *Can we identify strategies that students use for creating software designs?* We assume students approach their problem-solving strategies in different ways. We asked ourselves if it is possible to identify and categorise students' strategies.
- RQ2: *Can we learn which steps students find difficult in creating designs?* From experience in the classroom we learned there is a substantial part of students in a group that copes with difficulties in software design. We are interested in verifying/assessing whether it is possible to identify typical steps a student makes that are related to his/her difficulties.
- RQ3: *Do students that perform better on modelling tasks use different strategies than those who perform worse?* Is it possible to identify strategies that are more successful than others?

The contribution of this research is a description and/or classification of students' modelling behaviour revealed by our tools : WebUML (online UML class diagram editor) and LogViz (visual log analyser).

When we use the term 'model' in this chapter we mean that we explore modelling behaviour of just one view of the model: the structured view, the class diagram.

The remainder of this chapter is organised as follows: in Section 5.2 we explore related work. In Section 5.3 explains the method we used and includes the most important features of both tools. We show our results and discuss them in Sections 5.4, 5.5 and 5.6. We conclude and suggest future work in Section 5.7.

5.2 Related Work

Avouris et al. mention the value of logging students' learning activities and argue that logging is not enough [9]. They propose a combination of log info and video observations or snapshots in order to reveal patterns the log files do not show. In the same workshop Gibert-Darras et al. propose a set of design patterns for tracking students' problem solving performance [40]. They show patterns related to automated grading and the analysis of student answers. The patterns, found as result of answer analysis, do not reveal hidden learning strategies but can be used for monitoring students' learning. Blikstein reports on the use of a logging module for programming tasks [18] and identified student strategies that could help lecturers identify student problems in an early stage of the task.

Ko et al. conducted a study in which they used the combination of a logging file and visual interpretation tool to analyse the behaviour of software developers during a maintenance task [66]. They successfully identified different strategies the developers used.

Westera et al. found predictors in the logs of their serious games in the field of consultancy for learning [160].

Claes, Pingerra et al. [27][103] logged students' events during business modelling sessions. They used visual analysis and found different styles and related them to model quality.

Agarwal and Sinha conducted an empirical study [1] on the usability of UML. Novice developers rated UML low. Some difficulties with class diagrams were identified.

We are not aware of cases in the specific task of software modelling where students'

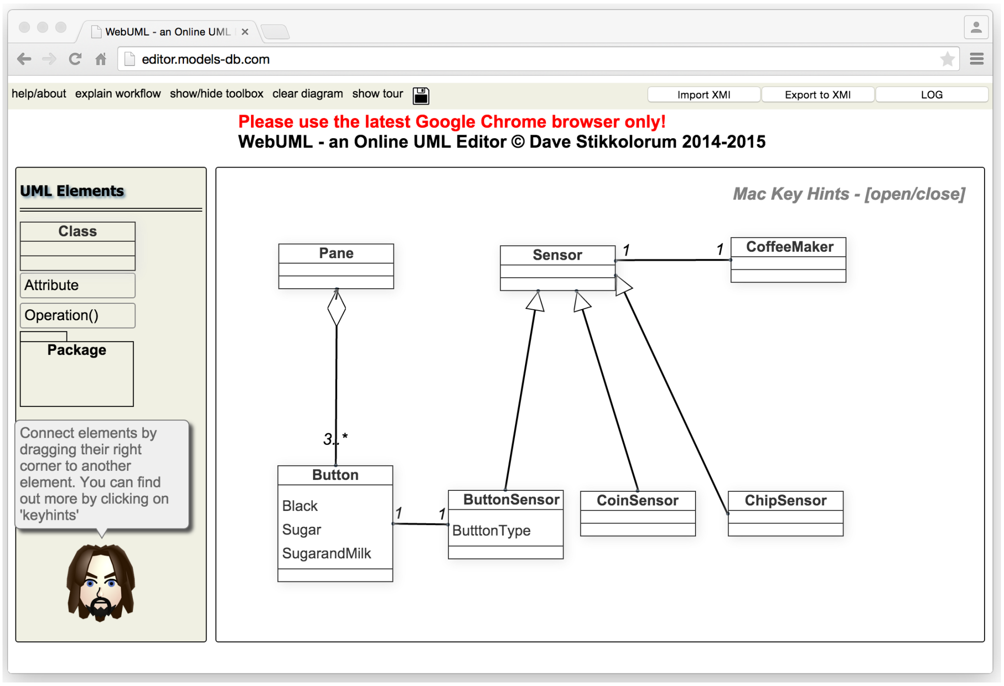


Figure 5.1: *Student's work in progress using the online editor*

logs were analysed to find task strategies.

5.3 Method

In this section we explain the method we used. We show the participant group, explain the modelling tool and the log format it delivers. We show the features of the visualisation tool that is capable of filtering the modelling tool's log files. We conclude with explaining the follow-up questionnaire that we conducted.

5.3.1 Participants

We observed 20 bachelor students, studying software engineering, during their 2nd semester course (Analysis and Design). The experiment was conducted in the third week of the course. The students were not experienced with UML or software design. They just learned the theoretical basics of class diagrams. Prior to the course the students followed a course on Java programming.

a Coffee Machine A coffee machine can make different types of coffee (one at a time): black, with sugar, with sugar and milk. The different types of coffee are poured into a cup. The front of the machine contains a pane with buttons. The buttons are used for selecting the desired drink. One can pay with a chip card or coins. The chipcard is read by a chip sensor and the coins by a coin sensor.

Figure 5.2: *Coffee Machine Modelling Assignment*

5.3.2 The Task / Assignment

We asked each student to use our tool 'WebUML' and to model a class diagram of a coffee machine. The short (76 words) case text was digitally offered to the students (Figure 5.2 / Appendix B). The students had to extract the details from the text for delivering a basic class diagram using classes, attributes, operations, association, aggregation and inheritance. There was no time constraint.

5.3.3 Online Modelling Editor

Because we wanted to have a simplistic UML editor with a minimum subset of the UML and the possibility to embed it in different online education software environments, such as serious games or gamified courses (e.g. specific MOOCs), we developed our own modelling tool : WebUML[1] (Figure 5.1, see also Chapter 4). At the moment of writing WebUML is only targeted at UML class diagrams equipped with the most used relationships, attributes, operations, labelling and multiplicity elements. The editor supports XMI import and export. The editor is capable of logging students' activities while doing modelling activities. Specifically for UML class diagrams we categorised the following activities for logging:

- CREATE - creation of UML elements, such as Class, Operation, Attribute, Association etc.
- MOVE - the movement of already created elements to another location in the diagram.
- SET - set value of attributes of elements e.g. names of classes, attributes etc.

[1]WebUML - `https://webuml.drstikko.nl`

- REMOVE - removal of an element from the diagram.

All items are logged with a unix timestamp[2]. For simplicity reasons we chose to save the log file in a comma separated file. For example CREATE :

```
<TIME>,CREATE,<UML_ELEMENT_TYPE>,<UML_ELEMENT_ID>, <UML_ELEMENT_NAME>,<CONTAINER>

Log fragment: 1423143012689, CREATE, CLASS, cl_1_1, Class, Diagram
```

5.3.4 Visualisation Tool

The visualisation tool, LogViz[3] is capable of reading one or more WebUML log files. It can filter on activity and UML element level. Figure 5.3 shows the main screen of the tool. Area 1 contains the controlling buttons of the tool; area 2 consists of the log's visualisation properties such as activity colours, symbols of UML elements, etc; area 3 is the most essential component of the visualiser where graphs and activity points are shown. Logging activities and UML elements are represented by different colours and symbols.

5.3.5 Follow-up Questionnaire

Our editor (WebUML) only records students' direct usage of the tool, not what students do when they are not using it (e.g. thinking about the modelling task, discussing). In order to anticipate on missing this data, a brief questionnaire was conducted. The questionnaire consists of 11 questions. We asked students to estimate the total time they spent on the assignment, time spent on different (sub)activities, and steps that they found difficult. Four of the questions are aimed at tool-feature development. We did not include them in the research that this chapter describes.

5.4 Results

In this section we present the different results: i) the statistics derived from the online modelling editor's log files; ii) the identified strategies of the students' modelling

[2]Unix Timestamp - https://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap04.html#tag_04_16

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

Table 5.1: *Overview of measured modelling activities*

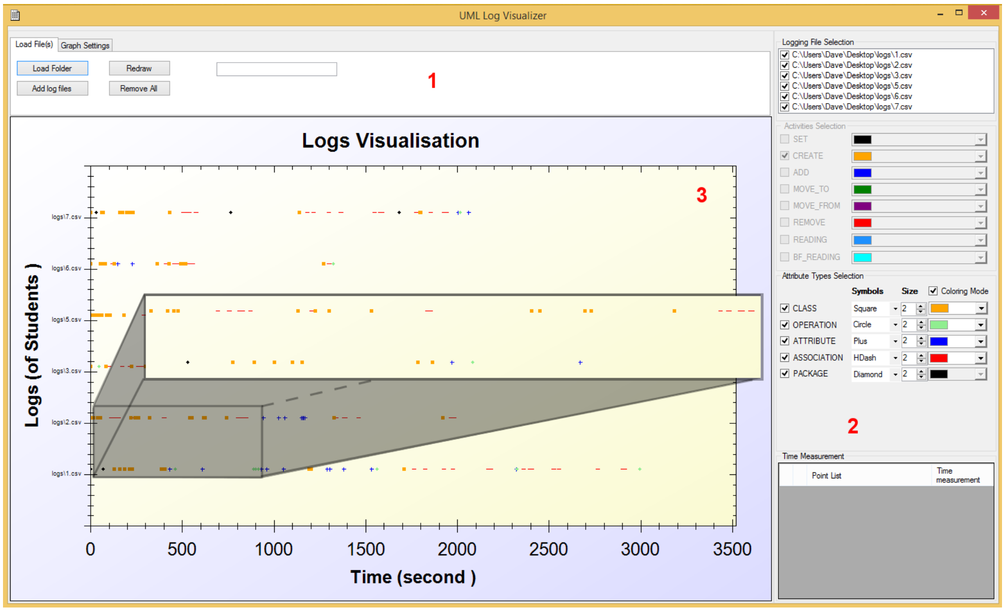
Student	Time (secs)	CREATE	MOVE	SET	ADD	REMOVE
1	395	11	2	11	0	0
2	1709	7	15	15	0	0
3	3580	29	115	40	5	4
4	2111	41	17	51	12	6
5	1688	50	73	78	23	6
6	703	30	2	14	0	0
7	1334	33	16	17	2	1
8	5579	93	150	43	28	2
9	225	9	18	0	0	0
10	2099	40	110	20	3	9
11	2299	59	210	55	41	10
12	3003	51	100	28	20	2
13	5607	96	270	71	40	6
14	1603	70	24	57	37	6
15	3123	28	40	21	14	6
16	1400	46	96	18	2	13
17	11357	59	360	16	4	12
18	689	22	54	12	2	3
19	2618	73	80	34	0	15
20	2087	48	49	26	7	3

task we analysed using the visualisation tool and iii) the response on the follow-up questionnaire.

5.4.1 Statistics

Table 5.1 shows the measurements of the different activities we logged. Each row in the table represents the log of the assignment task of one student. The numbers in columns 'CREATE' to 'REMOVE' represent the amount of occurrences of this particular activity during the performance of the assignment. The columns Time and MOVE show a large spread: Time (M = 2660.37, SD = 2515.05), MOVE (M = 90.05, SD = 95.09).

5.4.2 Visualisation



Logging File Selection

- [✓] C:\Users\Dave\Desktop\logs\1.csv
- [✓] C:\Users\Dave\Desktop\logs\2.csv
- [✓] C:\Users\Dave\Desktop\logs\3.csv
- [✓] C:\Users\Dave\Desktop\logs\5.csv
- [✓] C:\Users\Dave\Desktop\logs\6.csv
- [✓] C:\Users\Dave\Desktop\logs\7.csv

Activities Selection

☐	SET	⬛
✓	CREATE	🟧
☐	ADD	🟦
☐	MOVE_TO	🟩
☐	MOVE_FROM	🟪
☐	REMOVE	🟥
☐	READING	🟦
☐	BF_READING	🟦

Attribute Types Selection

		Symbols	Size	✓ Coloring Mode
✓	CLASS	Square	2	🟧
✓	OPERATION	Circle	2	🟩
✓	ATTRIBUTE	Plus	2	🟦
✓	ASSOCIATION	HDash	2	🟥
✓	PACKAGE	Diamond	2	⬛

Figure 5.3: *LogViz visualisation tool*

Since the task was a 'creating' task we filtered the logs' CREATE activities. Figure 5.3 shows a 'zoom in' of two student logs. As explained in Section 5.3.3, actions are logged in time. Figure 5.3 shows that the upper student associates (horizontal dash) classes early in the task, while the other first identifies attributes and associates later.

We examined all the log files by analysing the students' creation steps (create class, create attribute, create operation, create association) and identifying creation patterns by looking at the, in time, clustered creation actions. We identified 4 different main strategies (Table 5.2) and variations on these strategies. We indexed them 1 - 4 and are shown in Table 5.3. The last column in that table shows the amount of occurrences in the observed student group.

Explanation of the main strategies:

1. **Depth-less Strategy** - students don't add detail in the form of operations and/or attributes to come to a complete diagram.

2. **Depth First Strategy** - students, after they create classes, first add detail in the form of operations and/or attributes, then associate the classes.

3. **Breadth First Strategy** - students, after they create classes, first associate the classes, then add detail in the form of operations and/or attributes.

4. **Ad Hoc Strategy** - students don't use any form of structured approach to solve a class diagram task.

Table 5.2: *Class design strategy types*

1	Depthless Strategy	class → associate
2	Depth First Strategy	class → add detail → associate
3	Breadth First Strategy	class → associate → add detail
4	Ad Hoc Strategy	no structured approach

Table 5.3: *Different strategies students use while making a class diagram*

Strategy Index	Strategy Description	Occurrences
1	Create all classes → associate classes	3
1A	Loop (create classes→ associate classes)	1
2	Create classes → add operations and attributes → associate classes	
2A	Loop (Create classes → add operations and attributes → associate classes)	4
2B	Loop (Create classes → add operations and attributes) → associate classes	4
2C	Loop (Create classes) → Loop (add operations and attributes) → associate classes	2
3	Create classes → associate classes → add operations and attributes	1
3A	Loop (Create classes → associate classes → add operations and attributes)	1
3B	Loop (Create classes → associate classes) → Loop (operations and attributes)	3
4	No clear strategy: mix of classes, attributes, operations in random-like order	1

5.4.3 Follow-up Questionnaire

From the survey that was conducted after the assignment we received 8 responses. Although not all students responded, we show their answers in Table 5.4. It shows that the time spent on the task varies among the students (from 30 - 280 minutes). Students spent their time in three main activities: i) understanding the assignment and the related domains; ii) making modelling decisions; iii) usage of the tool. The responses indicate understanding and making decisions are the most time-consuming tasks. 5 out of 8 respondents spent more than 50 percent of the assignment time on understanding the assignment or making decisions on their models.

Half of the students reported doing the task continuously, while the other half inserted breaks for thinking over the case or looking for supporting information in books or online.

Most students found it difficult to decide class, attribute and association type (5 out of 8). 5 out of 8 observed students mentioned that they changed their diagrams layout to make them more reasonable.

5.5 Discussion

In this section we discuss the research questions we listed in Section 5.1. We do this based on our interpretation of the results presented in Section 5.4

5.5.1 RQ1: Can we learn strategies that students use for creating software designs?

From the visualisations we clearly found three types of strategies (summarised in Table 5.2) that students use for developing their diagrams. We also identified a fourth type which one could name a rest group (Ad Hoc). We could not determine a logic strategy in that case. We only identified one person for this category. The most interesting is the difference between the Depth First and Breadth First strategies where students differ in the moment they add attributes and/or operations in their class diagrams.

Table 5.4: *Summary of the reponses on the follow-up questionnaire*

ID	U (%)	M (%)	D (%)	O (%)	Time	Cont.	Difficulties	Revs	Revision Type
1	25	25	50	0	30	1		3	Normally I re-check my designs to be aligned.
3	29	33	33	5	60	1	Attributes and operations.	3	Organise classes.
4	54	18	29	0	280	0	How to split each element up into either a class or have it as an attribute.	3	Association should not be dependency
7	50	33	17	0	60	1	Whether to use packages or not, and about what to include in the model.	20	Mostly from scratch.
8	33	33	33	0	60	1	The payment and make coffee part were difficult to make.	2	Basically on design to make it more reasonable.
9	30	50	20	0	60	0		3	Moving things around trying to make the model cleaner.
10	20	50	30	0	50	0	How to connect the diagram between the classes. If there should be attributes/methods involved or not.	20	Re-making whole diagram, new classes, add/remove attributes and methods
16	10	60	30	0	30	0	Deciding what UML associations to use was difficult for me.	3	UML associations and naming classes.

ID=Diagram ID - U=Understanding - M=Making Decisions - D=Drawing
O=Other - Time = Time Spent - Revs=Number of Revisions

5.5.2 RQ2: Can we learn which steps students find difficult in creating designs?

From the log statistics we have the impression students use a great amount of MOVE actions. This could indicate they have difficulties in making a layout. Layout should help them to get 'the big picture'. This could be due to the fact that students don't have experience in software design and have to 'rethink' their designs over and over again. There is a large spread in time used to complete the designs. This could be due to the fact that some students have to think longer about decisions that they make about their designs. It could also mean that they went for a cup of coffee during the assignment. As said before, there was no time constraint or class setting that forced them to work continuously on the task. We found out from the logs of the modelling tool that it is difficult to accurately measure the exact time students spent on a particular activity. On the other hand, looking at the gaps in the visualisations between activities, one will find several gaps where a student 'does nothing'. Large gaps of, for example, 1000 seconds are difficult to interpret. They could mean anything, but gaps of 60 to 200 seconds could indicate thinking time. Assuming that, it is remarkable most of the times this is followed by a class MOVE action. The follow-up questionnaire has shed some light on our understanding about the gaps and what students found difficult about their task. For example students mention: *'It is hard to identify design elements from the assignment text'* and *'How to split each element up into either a class or have it as an attribute?'*.

Students spent most of their time in understanding the assignment and making decisions on what the model should look like. Some students reported that they needed time to get themselves familiar with the modelling tool. This resulted in a lot of intuitive sketching and deletion at first, then delaying for quite a long time before actually start drawing. They don't appear in Table 5.4 because we don't have their logs.

5.5.3 RQ3: Do students that perform better on modelling tasks use different strategies than those who perform worse?

To answer this question we only looked at the Depth First and Breadth First group. We evaluated the diagrams of the Depth First group and the Breadth First group. We took into account that the students are novices and that they looked at the layout (good readability) and richness (e.g. applying aggregation in stead of an 'empty' association for explaining a relationship better) of the diagrams. From this evaluation the Depth First group performed best. This could be explained with the idea it helps students to associate classes when classes have more detail (operations/attributes). When leaving these elements out of the classes first, the Breadth First approach, they will have more difficulties with associations.

5.5.4 Other Observations

From the results it seems a common trend to add attributes and operations at the same time.

There were two students (16 and 9) that did the task at once after spending time researching on the assignment and sketching on papers. Their comments where:

- "Two days, first 20-30 minutes was spent reading up on the material and think of a possible outline. Created the diagram the next day"
- "First I made a sketch on paper and practised a little bit with the given tool in order to get familiar with the tool environment. After that I did the assignment at once."

Their logging files do not completely cover the process.

5.6 Threats to Validity

We are aware of the difficulty in generalising the interpretations we presented in Section 5.5. A statistical approach would help here, but the amount of subjects is not ideal for a statistical analysis. Our first aim was to find possible strategies.

The time recorded in WebUML does not cover the 'real' total time, because of the fact students can spend time before they start modelling or when the tool is not used. We tried to capture this difference by asking them in the follow-up questionnaire.

5.7 Conclusions and Future Work

In the experiment reported in this chapter we revealed typical strategies that students use for a software design task. We developed our online UML editor: WebUML. WebUML is designed to log the different activities students perform during their tasks. Together with the editor we developed LogViz, a visual log analyser, in order to visualise WebUML's logs.

This chapter showed that, by visualising the students' activities, we were able to identify typical strategies students use to make class designs: Depth Less, Depth First, Breadth First, Ad Hoc. Two approaches stood out: the Depth First approach and the Breadth First approach. (RQ1)

From our logs students seem to use a lot of time to get an organised layout, which is likely to be a typical problem for novices. Some of the students take a long time to finish their assignment. On the one hand this could be related to all the moves. On the other hand they mention themselves it is hard to transfer from text to UML elements or to decide between attribute or class. (RQ2).

From a statistical point of view we could not prove if one strategy leads to better designs than the other. We did however find indications that a Depth First strategy could lead to a better layout, and therefore a class diagram that is better to understand. (RQ3)

Our tools open up new ways of studying the process of class design. Having the possibility to analyse students' modelling behaviour enables us to develop better educational programs and tools. Recognising certain strategies could help us to give better feedback during a task in stead of post-task.

In future research we will increase the amount of subjects to also statistically investigate the strategies and find possible correlations between model quality metrics and strategy actions. We want to test our tools as part of other educational suites, i.e. e-learning or MOOC-like environments.