2010-09 August 2010

Universiteit Leiden
Opleiding Informatica

How do we look at UML?

Ben Kwint

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Leiden University

Thesis BSc in Computer Science

How do we look at UML?

Author:
Ben Kwint
(0607932)

Supervisor:
Dr. Michel R. V. Chaudron
Werner Heijstek MSc.
Ariadi Nugroho MSc.

August 27, 2010

Abstract

In this paper we will discuss a Bachelor Thesis on an explorative eye-tracking experiment done at Leiden University. The goal of this experiment is to find out how people look at UML and whether their experience has any influence on this. We will also look at complexity of UML diagrams and what effect this has on the way people look at UML.

In this paper we will see that complexity is of great influence for understanding of UML diagrams, though experience does not seem to influence the way people look at UML at all.

Contents

List of Figures

List of Tables

1 Introduction

Most research on UML is done on the presentation or the results of UML. Meaning that not the comprehension, but the syntax is tested. For example Jonathan I Maletic [5] used an eye-tracking experiment to analyze the visualization of the UML.

Others analyze whether there is a relation between the UML models and the resulting product, an example of this kind of research is the research done by Ariadi Nugroho [1]. His research focuses on whether you can relate bugs in the end product with the actual UML models.

We wanted to see how people look at UML. Better is to say that we wanted to know when people thought they knew enough of the UML to be able to answer a set of questions.
Once analyzed this data might prove useful in helping UML users to improve their way of visually representing their models to the actual programmers that are going to work with it, thus improving the workflow and reducing the number of faults in the end product.

To do this we had to find a way to register the way people work with UML, especially how to look at the visual representation of UML. If you want to know where people look at and you want to know this in an exact manner you quickly come to eye-tracking. Therefore we used an eye-tracking experiment to get our data.

2 Eye-tracking

What is eye-tracking? Eye-tracking is a way to record the way someone looks at something, in our case a computer display.

Eye-tracking is not yet a really widely used experiment method in the field of Software Engineering. There are only a few researchers that work with eye-trackers and UML, so it is a new field to work in. For example Yann-Gaël Guéhéneuc [4] and Dr. Jonathan Maletic [5] work with Eye-tracking experiments with UML

For our experiment we used a Tobii eye-tracker. The advantage of an commercial eye-tracker over a home made eye-tracker is that everything is

build in and you get a set of default applications that make your work a lot easier. Next to this it is also a big advantage that the actual participant in the experiment does not see a couple of camera's, as they are hidden in the actual screen.

The eye-tracking device registers your eye-movements, which are stored as a set of x and y coordinates. These movements are then changed into **fixations** and **saccades**.

A point on your screen becomes a fixation after the user has been looking at the same place for a certain amount of time. The saccades are the movements of your eyes from one fixation to another.

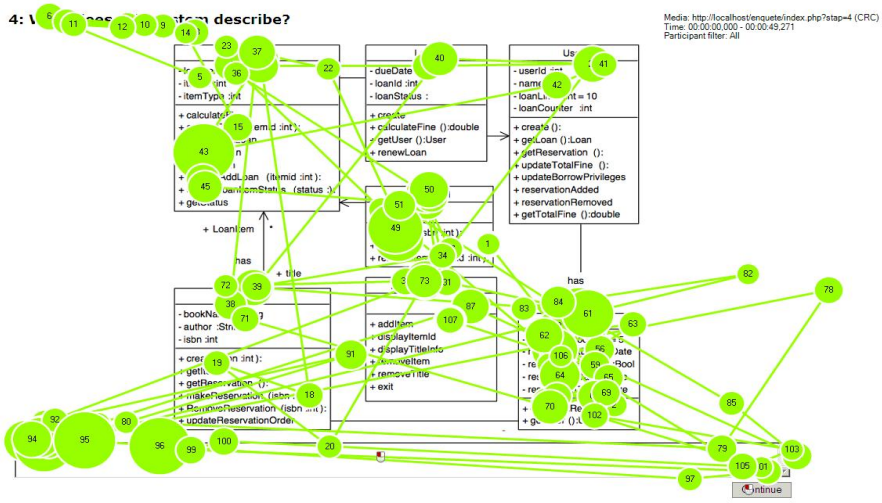

Figure 1: Gaze-plot on a class diagram

You could look at it as a graph with only 1 direction where every node has an incoming and outgoing branch, except for the start and end node, as you can see in **Figure 1**

Another way of representing this data is by means of a heatmap. These though they are interesting do not show you all the data. A heatmap (**Figure 2**) only shows where someone has been looking, it does not show any order.

Figure 2: Heatmap on a class diagram

Next to this information you can also define **A**rea's **O**f **I**nterest.

3 Area Of Interest

What is an Area Of Interest (**AOI**)? An area of interest is a predefined area on the screen that the Tobii eye-tracker is able to define. Because the Tobii eye-tracker generates a lot of information, we used the AOI functionality which can compact it into smaller, more usable data for the analysis. Once the experiment has been done you can see how many times the participant has looked at that part of the screen.

In our case we defined several AOI's. Every class in the class diagram has a class AOI and a class name AOI. The same is true for sequence diagrams, there every class has an AOI and every message as well. With multiple choice questions we also defined the answers as AOI's.
We defined these areas in the diagrams as AOI's because we wanted to know how people would move over a UML diagram. We wanted to be able to follow the trail of movements from one UML element to the next in the model. This way we could analyze how people looked at the diagram and we could get the view order, or whether there is a central point, etc.

4 Experiment design

When we started to design the experiment we had to ask ourselves how we would ask the questions. Because we wanted to find out how someone looked at a fragment of UML. To be able to do this you have to design a set of questions that allows you to view the view pattern on the entire UML diagram. We wanted to know the way of looking at it when someone needed to understand the entire diagram, not just one part. Our question thus have to be non leading, meaning that we did not want to give a hint in the question on where you could possibly find the answer to the question.

We considered using multiple choice questions, at first we thought we should not use them, because they often are leading. In some cases though we could use multiple choice, for example if you ask the question "What is the role of Class diagram X in the model" you can, without being leading to a certain answer, give a set of multiple choice answers and the user still has to consider the entire diagram before he can make a decision.

We also used open questions, these force the participant to consider the entire diagram, thus providing us with the data we wanted.
In the end we did two experiments. The first experiment was a try with three questions because we did not know how long it would take people to answer these questions and we wanted to see how people would respond to the eye-tracker. The second experiment was the actual experiment, this had different questions and more questions. Which took about fifteen minutes per participant to do.

The first experiment did not give enough information because it only used three questions and all were of the same kind, this was only done to see how we would have to set up our final experiment.

As for the questions, we decided to take four different diagrams. 2 of the diagrams used in this experiment came from an earlier experiment done by Christian F.J. Lange and Michel R. V. Chaudron [2]. We used four different diagrams because using only one would mean someone's memory would play a big role while doing the experiment, after three questions someone does not have to see the entire diagram anymore to be able to answer the question. Four diagrams gave us enough opportunity to create enough questions to gather the needed data. More diagrams would have meant the experiment would take too long.

Two diagrams would have programming questions and two would have questions on design. The design questions would be split up into three or

four questions in three categories.

- Global understanding of what the system should do.
- What the role is of a certain part of the diagram.
- And how you should add/edit a certain functionality.

We used the above stated categories of questions because all these questions force the participant to take a look at the entire diagram before they formulate an answer. For the third category this does not seem obvious, but if you ask someone to add a functionality that has influence on the entire system someone should first consider all the possible problems that can occur, thus forcing the participant to look at the entire diagram.

5 Problems

During the preparations for our experiment we ran into some problems which might be interesting to discuss because they would apply to any eye-tracking experiment. An interesting book on these problems is Eye Tracking Methodology [3].

5.1 Question presentation

There are many ways you can show your UML fragments to the designer/programmer. This led us to the question *"How to we let the subject look at the fragments of UML?"*. We came up with the following presentation methods that we could use.

5.1.1 Question and diagrams on one screen

The idea with this method is that we show both the UML diagrams and the question in one screen. This allows us to really follow the flow between UML diagrams and the question at hand. So we would be able to easily see where someone would be looking at when he has read a certain part of the question. Allowing us to see what part of a UML diagram someone needs to be able to understand that part. Because we want to know what information someone needs and how someone looks at UML this method of questioning is therefore interesting for our experiment. In our test and final experiment we used this method of questioning.

5.1.2 Separating the question from the diagrams

This method can be seen in separate ways, in which the order of viewing is important.

- Diagram to question.
- Question to diagram.
- Switching screens.

The problem with the first two cases is that you add an extra factor to the experiment, which is memory. The last case is basically the same as showing everything on one screen. The only difference is that the user has to do something to make the switch between the diagrams and the question. This only adds more difficulty when trying to analyze the data. We therefore decided this is not interesting for our experiment. The first method of questioning proposed is more interesting, because it better fits our needs.

5.2 Question formulation

When we finally had some UML fragments to work with we started thinking about questions. This though led to the next problem.

We needed to come up with questions that would stimulate viewing the entire UML diagram and not just parts of it.

We had a UML fragment consisting of a Sequence and a Class diagram. In this case one of the two diagrams showed an inconsistency. We wanted the subject to find this inconsistency but we did not want to lead him to the answer by asking what was wrong about the diagrams.

6 Participants

For our experiment we asked professional UML users to participate in our experiment. As said before we did 2 experiments, in our first experiment we asked students and teachers at Leiden University to participate. We did this to be able to make an estimation on the time needed by experienced users would need to complete the experiment. In the second experiment we only asked experienced users to participate.

Participant	age	gender	background
1	40-50	male	Quality Assurance
2	30-40	male	IT Consultant
3	40-50	male	entrepreneur
4	30-40	male	Resource manager
5	20-30	male	System AnalYst
6	50-60	male	IT Architect
7	20-30	male	System Analyzer
8	40-50	male	IT Consultant
9	40-50	male	IT Consultant
10	40-50	male	Software manager
11	40-50	male	IT Consultant

Table 1: Participants

7 Definitions

Before we state our findings we have to give some definitions.

7.1 Understanding

For this experiment we have to define what understanding means. In our case we assume that understanding means the following:

A participant understands a UML diagram when the participant thinks he or she has gathered enough information from the diagram to answer the question, whether or not the answer is correct.

7.2 Complexity of a diagram

To answer some of our research questions we have to define what the complexity is of UML diagrams. For this we used some of the definitions of SDMetrics [6].

Variable	Description
NumbAttr	The number of child attributes per class
NumOps	The number of child operation per class
Getters	The number of getters per class
NOC	Number of children classes per class (inheritance)
Dep_Out	Number of relations/dependancies on which the class depends
Dep_In	Number of relations/dependancies which depend on the class

Table 2: Variables used to calculate complexity

We used the variables stated in **Table 2**.

These variables have a rating between 0 and 1, the average of them gives the complexity of the class.

We calculate the rating by taking the number count of every variable and divide it by doing $rating = count/MAX(count)$. Taking the average of the ratings per class gives us the complexity.

Because most of our classes are quite small we decided not too add many variables because they would turn out to be zero almost constantly.

If one of the used variables would be zero for every class the variable would be omitted.

7.3 Participants experience

For our experiment we needed to know how experienced someone is in the field of software engineering. For this we had every participant fill in a form before he or she started the experiment. This form asked some general questions but also asked the participant to fill in a self evaluation of how much experience someone had with five different methods/tasks that are used in Software engineering.

Abbriviation	**Full name**
UML	Unified Modeling Language
SM	Software Metrics
DDS	Designing Software Systems
RSC	Reviewing Source Code
RSD	Reviewing Software Designs

Table 3: Variables used to calculate experience

We asked them to evaluate themselves (a grade between zero and five) on the methods/tasks given in **Table 3**. For each participant we calculated their experience by dividing the sum of all methods/tasks by the number of methods/actions.

8 Research questions

For this research we have formulated eight different questions we want to answer, which we will rey do in the rest of this paper and by that we will try to set up an answer to our main question **"How do we look at UML?"**

1. Is there a center point in a UML diagram?
2. Is there a relation between complexity and the number of fixations of a class?
3. Is there a shorter fixation time on less complex classes?
4. Is there a relation between the average view time and the number of fixations on an area of interest?
5. Does experience shorten your view time or the number of fixation you need to understand UML?
6. Is it possible to determine someone's experience by evaluating his/her view pattern?
7. Do people follow given relations in a UML diagram?
8. Is there a relation between coverage and experience when looking at a diagram?

The answers to these questions will be given by presenting the data and drawing a conclusion from them, from these we will formulate our answer to the main research question.

8.1 Is there a center point in a UML diagram?

To be able to answer this question we had to find a way to determine which of the elements in the UML diagram are seen as the most important part of the diagram.

To determine which of these elements are most important we did the following.

We used four variables to give a rating of importance to every part of the diagram. We used these variables:

- Average view time.
- Fixation count.
- The number of incoming saccades on an AOI.
- Number of times viewed when not focused on another AOI. (For example from the question or answer area.)

These variables were rated between zero and five and the average determines how important it is.

The UML elements all have a prefix assigned to them, in this case there are 2 prefixes **SC_** and **C_** which mean **S**equence **c**lass (class in a sequence diagram) and **C**lass which is given in the class diagram.

Every class in a class diagram and every class in a sequence diagram is given a rating between zero and five. This gives the importance for each element in the diagram. The more important an element is the more its central in the diagram, because of the variables we used to determine importance.

UML diagram 1		**UML diagram 2**		**UML diagram 3**		**UML diagram 4**	
C_Passenger	0.55	SC_LoanItem	0.4	C_Class4	2.26	SC_Class3	1.08
C_Customer	1.46	SC_Title	0.63	C_Class2	2.51	C_Class1	1.83
C_AirlineCompany	1.78	SC_ItemControl	0.66	C_Class3	3.14	SC_Class2	1.95
C_City	2.08	SC_LibraryTerminal	1.41	SC_Class4	3.22	SC_Class1	2.39
C_Airport	2.23	C_Title	2.59	SC_Class1	3.82	C_Class4	2.09
C_Booking	2.47	C_ItemControl	2.62	SC_Class3	4.34	C_Class3	2.93
C_StopoverInfo	2.67	C_LibraryTerminal	2.99	C_Class1	4.83	C_Class2	5
C_Flight	5	C_Reservation	3.02				
		C_Loan	3.86				
		C_LoanItem	4.75				
		C_User	4.84				

Table 4: Ratings per UML diagram and UML element

In **Table 4** you can see these ratings per UML diagram. The higher the grade is the more important the element is and thus how central it is in the UML diagram.

If we look at the position of the UML elements within the diagrams (**Section: 10**) we can see that the highest rated elements are in the center of the complete diagram (class and sequence) or centered in the class diagram. So we could say that if you place important classes central in your diagram they tend to stand out more than the rest of the diagram.

8.2 Is there a relation between complexity of a class and the number of fixations?

For answering this question we used the method of complexity calculation using the definitions from SDMetrics [6]. This method gave us a way to rate every class model with a certain complexity, allowing us to make the needed analysis.

Class	Complexity	Nr of fixations
City	0.18	108
AirlineCompany	0.18	95
Passenger	0.23	25
Customer	0.38	62
StopoverInfo	0.4	163
Airport	0.43	120
Flight	0.7	333
Booking	0.73	127

Table 5: Complexity & fixation diagram 1

Class	Complexity	Nr of fixations
ItemControl	0.38	213
Title	0.48	198
Reservation	0.55	200
Loan	0.55	254
LibraryTerminal	0.57	226
User	0.73	274
LoanItem	0.92	353

Table 6: Complexity & fixation diagram 2

Class	Complexity	Nr of fixations
Class1	0.47	123
Class4	0.5	50
Class3	0.6	65
Class2	0.7	65

Table 7: Complexity & fixation diagram 3

Class	Complexity	Nr of fixations
Class3	0.33	49
Class1	0.44	38
Class4	0.44	82
Class2	1	172

Table 8: Complexity & fixation diagram 4

Table 9: Complexity

			fixations	fix time
Kendall's τ	complexity	*Correlation Coefficient*	.388[1]	.348[1]
		Sig. (2-tailed)	.010	.021
		N	23	23
	fixations	*Correlation Coefficient*		.889[2]
		Sig. (2-tailed)		.000
		N		23

[1] Correlation is significant at $p \leq 0.05$
[2] Correlation is significant at $p \leq 0.01$

If we look at **Table 9** we can see that our statistical analysis shows that there is a relation between complexity and the number of fixations. If one variable increases and the other decreases, the correlation coefficients are negative according to Kendall's τ definition. If one increases and the other increases, the correlation coefficients are positive. Our analysis shows that if the complexity increases, the number of fixations also increases, thus complexity increases the number of fixations.

8.3 Is there a shorter fixation time on less complex classes?

For this question we used the same way of giving a complexity rating to classes as in the previous question. Here we only compare the complexity with the fixation time to see whether there is a relation between fixation time and complexity.

As we can see in **Table 10 - 12** on average the fixation time is higher when the complexity gets higher. Though one of the examples also shows us that it is not true.

Class	Complexity	Fixation time
City	0.18	33685
AirlineCompany	0.18	38754
Passenger	0.23	12194
Customer	0.38	29332
StopoverInfo	0.4	60479
Airport	0.43	42869
Flight	0.7	147004
Booking	0.73	63492

Table 10: Complexity & fixation time diagram 1

Class	Complexity	Fixation time
ItemControl	0.38	90441
Title	0.48	87242
Reservation	0.55	89939
Loan	0.55	94154
LibraryTerminal	0.57	99608
User	0.73	108764
LoanItem	0.92	139658

Table 11: Complexity & fixation time diagram 2

Class	Complexity	Fixation time
Class1	0.47	37136
Class4	0.5	15706
Class3	0.6	24882
Class2	0.7	19681

Table 12: Complexity & fixation time diagram 3

Class	Complexity	Fixation time
Class3	0.33	17154
Class1	0.44	10804
Class4	0.44	18841
Class2	1	58907

Table 13: Complexity & fixation time diagram 4

If we look at **Table 9** in **Section: 8.2** we also analysed whether the complexity is in relation to the fixation time.

According to our analysis this is indeed the case. When complexity increases. the fixation time also increases. If we look at **Table 10-12** one of the four cases shows us that the fixation time gets lower when the complexity gets higher, but in all other cases on average when the complexity gets higher the fixation time also gets higher.

We should say that the higher the complexity the higher the fixation time. Which sounds trivial, but during the experiment the fixation time on complex classes is also higher when the class is not necessary for answering the question.

8.4 Is there a relation between the average view time and the number of fixations on an area of interest?

When someone is scanning a UML model to understand what is describes you would expect that this person look at places in the diagram very often but not that long. So you would expect a high fixation count and a low fixation time. By asking ourselves the question *"Is there a relation between the average view time and the number of fixation on an AOI?"* we try to prove that the assumption given earlier is true.

To see if there is a relation between the average fixation time and the number of fixations, we have done the following analysis on the data.

We took the average fixation time on every AOI and the number of times there was a fixation on this AOI. We have divided the AOI's into 2 groups, low fixation count and high fixation count.

Low and high fixation count is determined by taking the average of the fixation count and every AOI smaller or equal to it is in the low group **(L)** , everything higher is in the high group **(H)**.

By dividing the AOI's in these groups you get the table shown in **Table 14**. This table shows the average fixation time per question and the averages in the groups **L** and **H**.

Question	Average	L	H
1	433.53	466.4	397
2	290	283.17	306.4
3	397.05	381.08	431.67
4	445.07	468.29	424.75
5	363.2	370.74	338.43
6	303.04	296.19	318.71
7	363.12	352	413.17
8	395.49	414.62	355.86
9	306.09	292.4	335.43

Table 14: Average fixation time, average low/high fixation time

Based on the data in **Table 14** we could conclude that there is a relationship between the fixation count and fixation time. Though the data also shows that this is not strictly always the case. We could safely conclude that on average (5 out of 9 cases) the fixation time will be lower when there are less fixations on a certain AOI.

8.5 Does experience shorten your view time or the number of fixations you need to understand UML?

To be able to answer this question we did a statistiscal analysis on the data we had. We wanted to know whether any of methods/tasks that determines a part of the experience has any correlation with the fixation time or the number of fixations.

			avg View	**avg Fixation**
Kendall's τ	exP UML	*Correlation Coefficient*	-.073	-.122
		Sig. (2-tailed)	.782	.643
		N	11	11
	exp SM	*Correlation Coefficient*	.022	.174
		Sig. (2-tailed)	.933	.497
		N	11	11
	exp DSS	*Correlation Coefficient*	.127	.192
		Sig. (2-tailed)	.637	.478
		N	11	11
	exp RSC	*Correlation Coefficient*	-.130	-.131
		Sig. (2-tailed)	.616	.616
		N	11	11
	exp RSD	*Correlation Coefficient*	-.170	-.318
		Sig. (2-tailed)	.518	.228
		N	11	11
	avg Exp	*Correlation Coefficient*	-.094	-.076
		Sig. (2-tailed)	.693	.751
		N	11	11
	avg View	*Correlation Coefficient*		.771[1]
		Sig. (2-tailed)		.001
		N		11

[1] Correlation is significant at the 0.01 level (2-tailed).

Table 15: Correlation Matrix for Experience and View Time and # Fixations

If you look at **Table 8.5** we fail to find any relation between the fixation time or the number of fixations with any of the methods/tasks. What you would expect is that the fixation time and the number of fixations do correlate with each other. This is of course also visible in our dataset, this also strengthens our statement given in **Section: 8.4**.

If we look at our data we do not find any relation between experience and fixation time or the number of fixations. This seems rather conclusive from our dataset, but it is certainly a point for further research, because we only used small, simple UML diagrams. If you would use bigger diagrams you might be able to find a difference between the participants.

8.6 Is it possible to determine someone's experience by evaluating his/her view pattern?

For this question we evaluated our data and rated every user per question on his average view time, average fixation count and the number of relations this person has followed.

As assumption we took that more experienced users will follow more relations and have less fixations and use less time to look at a UML diagram. After doing this analysis we took the average of the three parameters and ordered all the participants on their achieved level, this though gave us a wrong image, so we removed the relations from the equation. After doing this most participants are rated quite good.

	Exerience	Correctness (%)	View time	Fixation count	Average
P3	5	0,47	0,3	0,23	0,33
P5	4	0,73	0,13	0,25	0,37
P10	4,2	0,73	0,25	0,29	0,42
P7	4,8	0,56	0,47	0,42	0,48
P4	2,8	0,82	0,33	0,36	0,5
P11	3,6	0,64	0,53	0,47	0,55
P9	5	0,82	0,49	0,38	0,56
P1	3,8	0,82	0,4	0,48	0,56
P6	4,6	0,73	0,66	0,6	0,66
P2	5	0,64	0,84	0,61	0,7
P8	4,8	0,73	0,68	0,73	0,71

Table 16: Level rating

Rating someone by fixation time, fixation count and followed relations does not give you a good image of the participants experience. Though if you take out the relations from the equation and add the percentage of correct answers (as has been done in **Table 16**), the rating becomes much better and you are able to rate most users quite good in the order of their experience. Though there are still some people that do not follow the pattern.

8.7 Do people follow given relations in a UML diagram?

The question whether people follow the given relations in a diagram might sound a bit trivial, but it is really interesting to answer.

As a designer and user of UML you might think that people tend to follow the given relation in a diagram. But as we were watching the replay videos of our experiment we started to question whether it was that trivial.

Question	Followed relations	Branches without relation
1	140	97
2	109	39
3	139	64
4	189	180
5	129	96
6	48	40
7	122	56
8	71	73
9	94	302

Table 17: Followed branches, separated on relation or not.

To be able to determine the followed relations in a UML diagram we first had to define all the relations that are in the UML diagram. We used a database that contained all the AOI data and added the relations in it. This way we where able to see the saccades in the data. We could see when someone moved from AOI **X** to AOI **Y**. By combining this data with the relations in the database we where able to determine, as you can see in **Table 17**, how many times someone followed a relation and when someone did not follow a relation.

As you can see in **Table 17**, you can clearly see that the answer to the question is yes. But our experiment is not fully conclusive. The data might say yes but if you look at the UML we have used in our experiment we can not say yes. All our diagrams were ordered and only used straight lines to give relations. For further research it might be nice to ask this question in general, but use diagrams which are differently ordered to get a better understanding on this question.

8.8 Is there a relation between coverage and experience when looking at a diagram?

When we looked at our data we wanted to see whether coverage (the amount of the diagram seen by the user) has any relation with the experience level of the user.
We evaluated this in three ways, but in the end used only one of the methods for our answer, because it did not differ that much.
We evaluated it in the following 3 ways:

No mask (Figure 3a)

First we looked at the diagrams themselves, we used those as the marker for our coverage measurement. This gave us all rather low values of coverage.

Block size mask (Figure 3b)

Next we tried covering the entire area which contains the diagram with one big block, this gave us rather high values of coverage.

Only mask the diagrams (Figure 3c)

Next we only covered up the insides of our diagrams, this means we got about the average of the above two methods. For our conclusion we will use this method of evaluation.

(a) No masking (b) Block masking (c) Diagram masking

Figure 3: Coverage methods

To determine the percentage of the screen that was covered by the eyes of the user we build an application, see below, that would take in two bitmap

files, one with a view pattern construction in red and one with a black overlay as told above.
In **Table 18** you can see that most experienced users have a lower (on average) coverage, except for participant 3.

Question		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
	Lvl	3.8	5	5	2.8	4	4.6	4.8	4.8	5	4.2	3.6
1		39.9	19.35	53.06	36.71	33.93	16.02	36.25	8.89	20.06	14.81	25.52
2		11.98	21.64	15.1	16.23	30.87	12.2	24.73	12.74	34.64	27.69	18.04
3		2.63	4.28	11.8	15.01	11.77	8.93	1.27	5.09	5.06	18.66	18.29
4		30.48	9.73	80.78	26.54	25.57	32.66	42.07	30.16	28.89	19.92	43.84
5		33.07	9.54	39.61	32.98	35.54	25	10.5	14.4	23.84	20.37	20
6		9.97	7.46	12.7	11.57	10.48	12.26	5.7	4.97	5.68	17.23	6.19
7		27.74	23.16	28.11	29.56	32.34	45.26	15.96	33.83	12.05	18.59	15.72
8		57.65	41.12	61.57	63.11	83.36	66.84	24.89	35.96	50.1	51.67	26.06
9		39.82	17.42	95.08	26.13	44.01	52.4	48.52	54.18	38.37	61.03	10.73
	Avg:	**28.14**	**17.08**	**44.2**	**28.65**	**34.21**	**30.17**	**23.32**	**22.25**	**24.3**	**27.77**	**20.49**

Table 18: Coverage per question per participant.

Coverage analysis code

```
1  #include <iostream>
2  #include "/cximage/ximage.h"
3  using namespace std;
4
5  // determine the coverage of an image
6  bool coverage( CxImage* img, CxImage* mask ){
7    double covered = 0, total = 0;
8    RGBQUAD c, m;
9    DWORD width = img->GetWidth(), height = img->GetHeight(), i, j;
10   for( i = 0; i < width; i++ )
11     for( j = 0; j < height; j++ ){
12       // get the color of pixel i and j on both images
13       c = img->GetPixelColor(i,j,true);
14       m = mask->GetPixelColor(i,j,true);
15       if( m.rgbBlue == 0 && m.rgbGreen == 0 && m.rgbRed == 0 ){
16         total++; //number of points masked ++
17         if( c.rgbBlue == 0 && c.rgbGreen == 0 && c.rgbRed == 255 )
18           covered++; // if there is a red color, it has been masked
19       }
20     }
21   if( total > 0 ){
22     cout << (100*(covered/total))<< "% of the image has been seen." << endl;
23     return true;
24   }
25   return false;
26 }
27
28 int main( int argc, char* argv[] ){
29   if( argc == 3 ){ // did the user give 2 file names
30     FILE* in1 = fopen( argv[1], "r" );  // first check whether both images
31     FILE* in2 = fopen( argv[2], "r" );  // given exists
32     if( in1 == NULL || in2 == NULL )
33       cout << "On or both of the files given don't exist." << endl;
34     else if( !coverage( new CxImage( in1, CXIMAGE_FORMAT_BMP),
35                         new CxImage( in2, CXIMAGE_FORMAT_BMP) ) )
36       cout << "No masking has been given." << endl;
37   }
38   else
39     cout << "   Give 2 image files as parameters." << endl
40          << "1: This should be the original image, with the gazeplot." << endl
41          << "2: This should be the image with the masking of the AOI." << endl;
42   return 0;
43 }
```

On average the experienced users have a view coverage of 26.89% and the less experience users have a coverage of 27.85%. If we do not take our exception into the calculations the difference get bigger to 23.42%. So there is a difference, but it is not that big that you could conclude that there is a relation between experience and coverage.

9 Conclusion

During the experiment we found out that many things could improve. But we have also seen a lot of interesting things. We started out with the question "How do people look at UML?" better would be to say "How do different people look at UML?". In our case different people are people with different backgrounds and different experiences in the field of Software Engineering.

We have seen that complexity influences the amount of time someone spends on looking at UML (**8.3**) before he or she thinks they understand it. This is also true for the number of fixations on class diagrams (**8.2**). In the end though experience does not seem to influence this that much (**8.5**), which you would expect to see, but it is stated that if the fixation count increases, the average fixation time also increases (**8.4**).

We have also seen that there is a central point, in either the entire UML diagram or in the center of a class diagram (**8.1**). This though is certainly a point for further research.

If we look back at experience, it is mostly possible to split a group of participants on their view pattern into experience groups using only fixation count and view time (**8.6**). We say that it is mostly possible because in our reseach people were forced to give an answer, in this case if someone does not know the correct answer he or she will guess something.

Relations seem to be important as well, in our case we could see that most transition between parts of diagrams followed the relations already given in the diagram (**8.7**). Though we also say that this is a good point to do some more research on.

Though you might expect that experienced participants filter out certain parts of the diagrams once they know what they are looking for. Our coverage data (**8.8**) showed us that experienced participants look at every part of the diagram as well as the less experienced participants. The difference was about 1 percent which is absolutely not significant.

To give an answer to our research question. All participants look at UML almost the same way though complexity has influence on the way we look at

it. Complexity increases the time and the eye movements needed to understand UML.

In our introduction we stated that we might be able to help with improving the visualization of UML diagrams. As we say in **Section 8.1** there is a central point within UML diagrams. If you can build your UML in such a way that this central point is actually centralized you might improve understanding. We have also seen that if you order UML diagrams in a strict manner, in our case into a grid with strait lines/relations between them, people tend to follow these relations. We have not tested crossing relation lines in UML diagrams thus we can not with certainty say that this does not happen when the UML diagram is unordered but we would advice using this method of visualisation.

10 Further research

Because this research project was based on the question "How do people look at UML?" some questions in this experiment could be done as a project on their own.

Most promising for this would be the question: "Do people follow given relations in a diagram?". In our experiment that is true, but we only used ordered diagrams. It would be nice to see whether people also follow these relations when a diagram had a lot of classes that are connected but not well ordered.

The question whether there is a central point in UML diagrams is a good question to work on. We had structured UML with a nice layout. Making this unordered might even push people even more on finding a central diagram. And what happens if a central diagram is really in the center of the UML diagram, does this make it easier to understand UML?

Another sub-question we used was "Does experience shorten your fixation time." discussed in **Section: 8.5**. It might be interesting to see wether you can find a relation between complexity and fixation time/fixation count. We had a small set of participants with simple UML diagrams which you can find in **Section: 10**.

References

[1] Michel R.V. Chaudron Ariadi Nugroho. *An Empirical Analysis of Level of Detail in UML Models and its Relation with Defect Density.* ACM/IEEE, 2008.

[2] A. T. Duchowski. *Eye Tracking Methodology.* Springer, 2007.

[3] Yann-Gaël Guëhénéuc Gerardo Cepeda Porras. *An empirical study on the efficiency of different design pattern representations in UML class diagrams.* Springer, 2010.

[4] Christian F.J. Lange Michel R.V. Chaudron. *Effects of Defects in UML Models - An Experimental Investigation.* 2005.

[5] Jonathan I. Maletic Shehnaaz Yusuf, Huzefa Kagdi. *Assessing the Comprehension of UML Class diagrams via Eye Tracking.* IEEE, 2007.

[6] Jürgen Wüst, 2009. http://www.sdmetrics.com/down/SDMetricsManual.pdf.

Appendix: UML diagrams used in the experiment

Here we will present the UML diagrams we have used in our experiment.

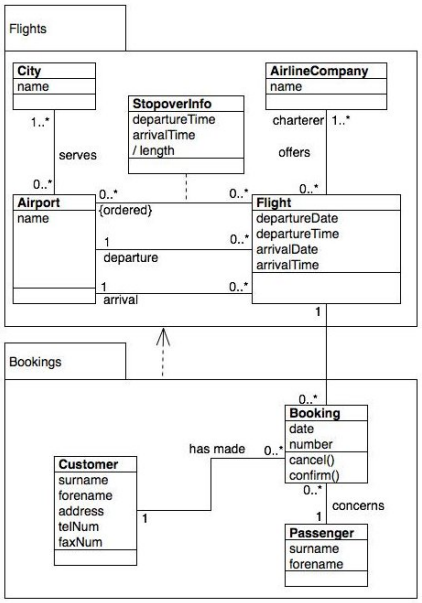

Figure 4: UML diagram used in question one, two and three

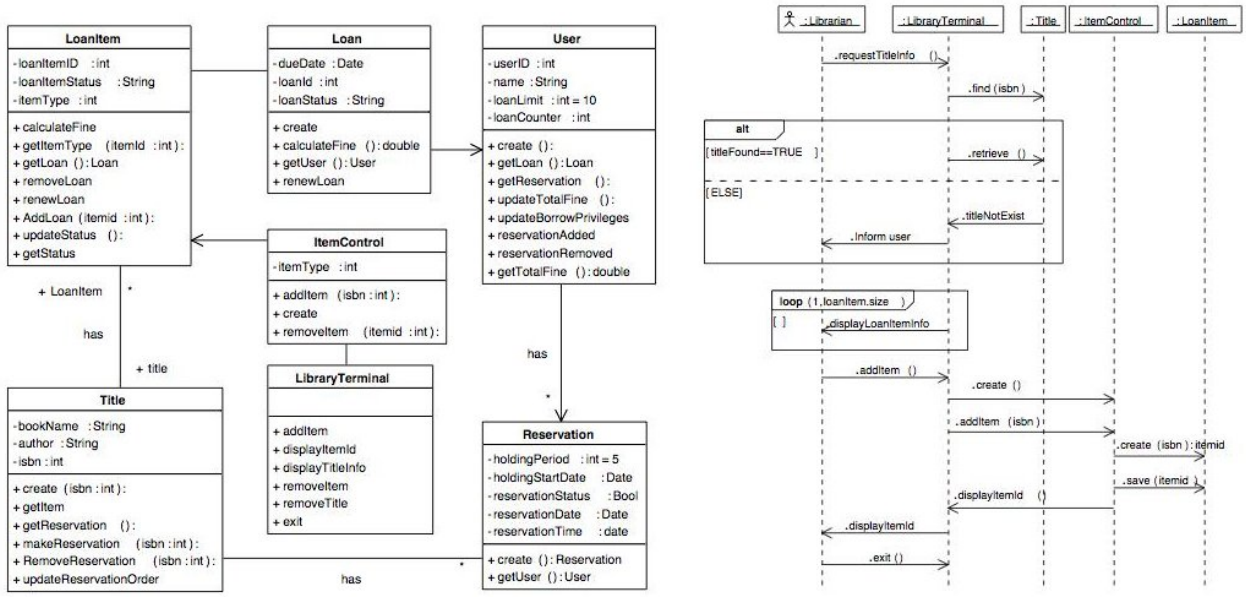

Figure 5: UML diagram used in question four, five, six and seven

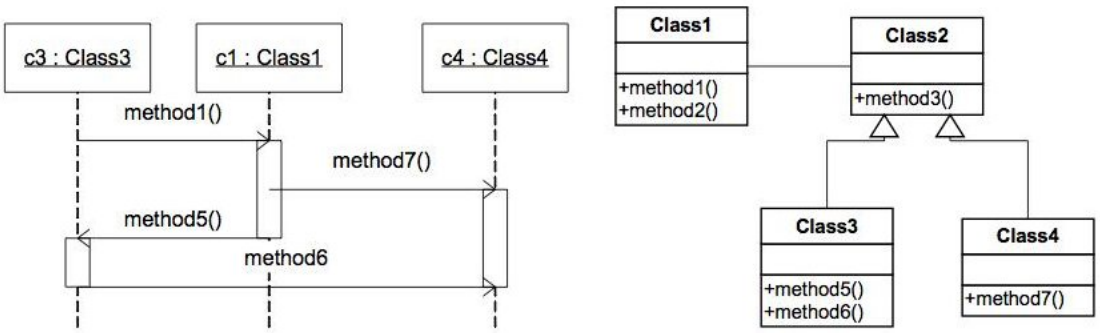

Figure 6: UML diagram used in question eight

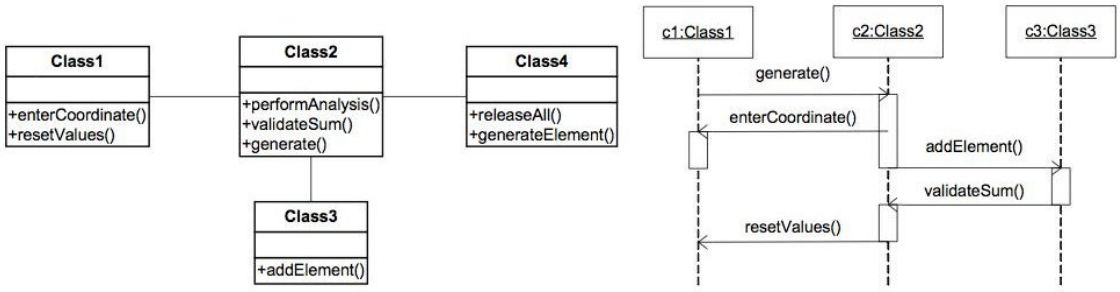

Figure 7: UML diagram used in question nine