Internal Report 2010-07 Month 2010

Universiteit Leiden

Opleiding Informatica

Visualizing the Level of Detail in UML Models

Robin van den Broek

BACHELOR THESIS

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

Visualizing the Level of Detail in UML Models

Robin van den Broek
Supervisors: Michel Chaudron and Ariadi Nugroho

Leiden Institute for Advanced Computer Science (LIACS)
Universiteit Leiden
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

UML models are used diversely in practice and provide a designer much freedom and flexibility when designing his models. This freedom and flexibility lead to diverse styles and rigors in modeling.

Level of Detail (LoD), which represents the amount of information that is used to specify models, is applied in many studies. Yet there is no tool or application that is able to visualize the level of detail software metrics on top of UML models. This paper describes our path to implementing a level of detail visualization and a small analysis using our implementation.

The contribution of this paper is that we explain how the LoD metrics can be visualized and show how this visualization can be helpful when analyzing the Level of Detail of UML models.

1. Introduction

This paper describes the contents of my bachelor project. During this project we have been searching for a way to visualize the level of detail metrics on top of UML models. The level of detail metrics are design metrics that are used to analyze UML models. As there is currently no tool that is able to do such a visualization we want to extend MetricView Evolution, which is able to visualize other software metrics, in such a way that it is able to visualize the level of detail metrics. By doing so we want to do an analysis and answer some research questions regarding the level of detail metrics and try to find some interesting facts about the level of detail that is used to specify a UML model.

2. Related Work

There have been some related studies in the field of visualizing design metrics on top of UML models. The most closely related works that studied the visualization of design metrics are from Maurice Termeer (MetricView) [4] and Martijn Wijns (MetricView Evolution) [5]. Both provided a way of visualizing several design metrics on top UML models in an intuitive way. Different from the aforementioned studies on visualizing design metrics, in this paper we want to study the visualization of the level of detail design metric on top of UML models and in this way try to find interesting facts about the level of detail of UML models.

3. Level of detail metrics

Design metrics are used to analyze UML models and try to give feedback on this model. One of these design metrics is the “Level of detail” [1]metric. The level of detail metric, further annotated as “LoD”, focuses on UML modeling and tries to investigate the relation between the quality of software models and the quality of the final implementation. In UML modeling the level of detail can be measured by quantifying the amount of information that is used to represent a modeling element. For example in modeling sequence diagrams one could think of message names that can be just an informal label, a label that represents a class method or a label that represents a class method including parameters. In modeling class diagrams there are many syntactical features that can be used to increase the level of detail as for example attributes, operations, association names, multiplicity, roles and so forth. UML models that are modeled with a low level of detail are providing just a few features to represent modeling elements such as for example classes without any attributes or operations and nameless associations. UML models that are modeled with a high level of detail are providing a lot of features to represent modeling elements and provide thus more detail about elements.

The level of detail metrics are defined at two types of UML diagrams: class diagrams and sequence diagrams. The following 4 metrics are defined for classes used in class diagrams:

- Attribute Signature Ratio ($CD_{AttrSigRatio}$). This metric measures the ratio of attributes that have a type definition compared to the total number of attributes.
- Operations with Parameters Ratio ($CD_{OpsWithParamRatio}$). This metric measures the ratio of operations that have parameters defined compared to the total number of operations.

- Association Label Ratio ($CD_{AssocLabelRatio}$). This metric measures the ratio of associations that have a label defined compared to the total number of associations for a class.
- Association Role Ratio ($CD_{AssocRoleRatio}$). This metric measures the ratio of associations that have a role defined at either end of the association compared to the total number of associations for a class.

These four metrics are ratio-based and therefore they can be normalized. By doing so we reduce the influence of class size. The above metrics can be categorized into two higher-level LoD measures:

- Attribute and Operation detailedness (CD_{aop}) which combines the Attribute Signature Ratio ($CD_{AttrSigRatio}$) and the Operations with Parameters Ratio ($CD_{OpsWithParamRatio}$) to give a value for the level of detail used to model attributes and operations.

$$CD_{aop}(x) = CD_{AttrSigRatio}(x) + CD_{OpsWithParamRatio}(x) \quad (1)$$

- Associations detailedness (CD_{asc}) which combines the Association Label Ratio ($CD_{AssocLabelRatio}$) and the Association Role Ratio ($CD_{AssocRoleRatio}$)to give a value for the level of detail used to model associations.

$$CD_{asc}(x) = CD_{AssocLabelRatio}(x) + CD_{AssocRoleRatio}(x) \quad (2)$$

As we can see in (1) and (2) the LoD measures of a class are determined based on the information that is related to that particular class (measured at class level), rather than on the LoD of the diagram in which a class appears. (1) and (2) lead us to the following definition for the class LoD:

$$Class\ LoD(x) = CD_{aop}(x) + CD_{asc}(x) \quad (3)$$

Five metrics are defined to measure LoD from sequence diagrams:

- Non Anonymous Object Ratio ($SD_{NonAnonymObjRatio}$). This metric measures the ratio of object that have a name defined compared to the total number of objects in a diagram.
- Non Dummy Object Ratio ($SD_{NonDummyObjRatio}$). This metric measures the ratio of objects that correspond to a modeled class compared to the total number of objects in a diagram.
- Non Dummy Message Ratio ($SD_{NonDummyMsgRatio}$). This metric measures the ratio of messages that correspond to a modeled class method compared to the total number of messages in a diagram.
- Return Message With Label Ratio ($SD_{ReturnMsgWithLabelRatio}$). This metric measures the ratio of return messages that do have a label defined compared to the total number of return messages in a diagram.
- Message With Parameter Ratio ($SD_{MsgWithParamRatio}$). This metric measures the ratio of messages that have parameters defined compared to the total number of messages in a diagram.

Just like the class diagram metrics, these metrics are based on ratio's and can therefore be normalized. The sequence diagram metrics can be categorized into the following two higher-level LoD measures:

- Sequence Diagram Object Detailedness (SD_{obj}) which covers the Non Anonymous Object Ratio ($SD_{NonAnonymObjRatio}$) and the Non Dummy Object Ratio ($CD_{NonDummyObjRatio}$) to give a value for the object detailedness for a sequence diagram.

$$SD_{obj}(x) = SD_{NonAnonymObjRatio}(x) + SD_{NonDummyObjRatio}(x) \quad (4)$$

- Sequence Diagram Message Detailedness (SD_{msg}) which covers Non Dummy Message Ratio ($SD_{NonDummyMsgRatio}$), Return Message with Label Ratio ($SD_{ReturnMsgWitLabelRatio}$) and Message with Parameter Ratio ($SD_{MsgWithParamRatio}$)to give a value for message detailedness in a sequence diagram.

$$SD_{msg}(x) = SD_{NonDummyMsgRatio} + SD_{ReturnMsgWithLabelRatio} + SD_{MsgWithParamRatio} \quad (5)$$

As we can see in (4) and (5) the LoD measures for a sequence diagram are based on the information that is related to the diagram. (4) and (5) lead us to definition for the sequence diagram LoD measure:

$$SequenceDiagram\ LoD = SD_{obj}(x) + SD_{msg}(x) \quad (6)$$

Another nice level of detail metric is the detailedness in which a class is modeled in all the sequence diagrams it appears in. As a class can be modeled in several sequence diagrams we can take the average sequence diagram level of detail to give an indication for this measure:

$$ClassLoD_{AvgSD}(x) = \frac{1}{n} \times \sum_{\substack{i \in Sequence\ Diagrams, \\ x \in i}}^{n} SequenceDiagramLoD\,(i) \quad (7)$$

3. MetricView Evolution

In 2004 a tool has been developed by Martijn Wijns called "MetricView Evolution"[1]. This tool is able to load a UML model, load software metrics and show a visualization of these software metrics on top of the loaded UML model. MetricView Evolution is able to load UML models created by Rational Rose and Rational XDE which are both modeling tools created by IBM. MetricView Evolution uses xsl translation to parse models from these tools and inserts the needed elements into an embedded MySQL database. After loading the model the metrics are calculated by extracting data from the database and calculate the metric values for all elements in the model using python scripts.

The metrics that have to be calculated can be configured by adding them to an xml-file which serves as metrics definition. Each definition should contain a reference to a python function, a name, a threshold that can be either upper or lower limit and a value for this threshold. When MetricView is done parsing

[1] MetricView Evolution is part of Martijn Wijns' master's thesis, based on the earlier developed tool "MetricView" by Maurice Termeer.

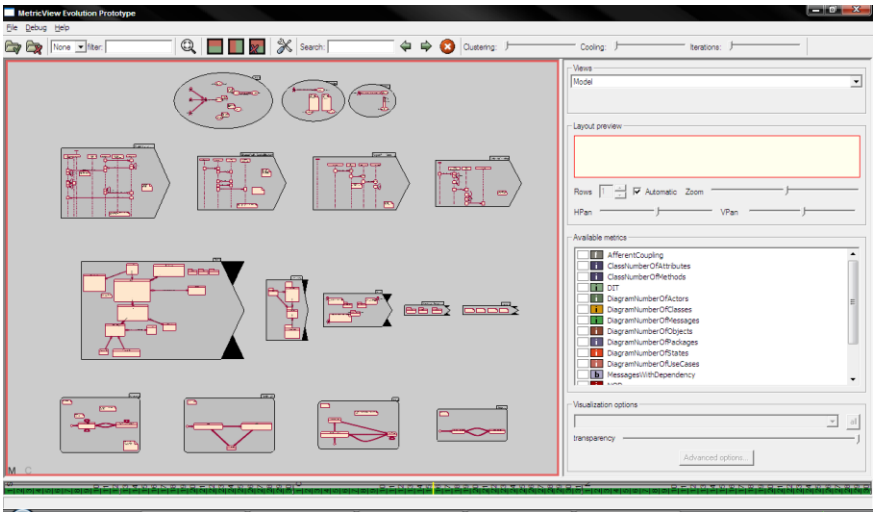


Figure 1: Screenshot MetricView Evolution as created by Martijn Wijns in 2004. The active window here is the Model Window which shows a "LangeDiagram" containing all Use Cases, Sequence Diagrams, Class Diagrams and Statemachines. There are no metrics selected as you can see in the "Available metrics" section.

and calculating the main view is presented. The user gets to choose between three windows: Model, Quality and Context. These three windows serve different purposes:

- Model Window. This window presents the entire UML model in one view. All use-case diagrams, class diagrams, sequence diagrams and statechart diagrams are combined into a so-called "LangeDiagram". This "LangeDiagram" does not only contain all these diagrams, but it is also able to show the relations between different elements (for example the relations between classes and corresponding objects in sequence diagrams). By doing so it is very easy to see which class is used in which diagrams and thus provides an intuitive way of browsing. [See Figure 1]. Next to being able of visualizing all diagrams one can also use zooming functionality to browse the model. Last but not least the "LangeDiagram" provides a way of presenting software metrics on top of the loaded UML model. By creating an extra layer on top of the model one can very easily see the values of all kind of software metrics (manually deciding which one a user would like to see).
- Quality Window. The quality window presents a so-called "Quality Tree". This tree consists of three different categories as children: maintainability, understandability and design smells. Each category has related design metrics as its children. As every design metric has a particular value (average value of the model) the categories can be rated and at the end there will remain one rating for the quality of a model.

MetricView Evolution provides an interactive 3D browsing ability to explore the model. Next to browsing a user is able to select different software metrics that have to be shown on top of the UML model. This is done by adding an extra layer on top of the model and representing each class with colors indicating good or bad performance for a certain metric. By doing so it is very intuitive to see which diagrams and elements are modeled and what the results for different software metrics for different elements are. The goal of visualizing the metrics in such a way is to see whether there is a way to improve the design of the model by improving classes and/or diagrams with low performance for certain metrics.

Next to being able to visualize the values of design metrics, MetricView Evolution is also able to visualize the evolution of a software model. By using a timeline which serves as a sort of revision history a user can see the differences between both the UML models themselves as well as the differences between the performance of the software metrics on the UML models. This provides a simple but clear overview when trying to see how software design models evolve during time.

4. Problem statement

The LoD metrics can be very interesting metrics to investigate when modeling software because they can give us information about the information that is used to represent a model. To do so we need to have an application or tool that can visualize the LoD metrics. MetricView Evolution is a tool that is capable of visualizing design metrics on top a UML model. However MetricView Evolution has limited software metrics that it can visualize which do not include the LoD metrics.

Also we want to see what we can learn from studying the LoD of UML models. For example what are the differences between different models in the level of detail used to model them and why are there any differences. To see what we can learn by studying the level of detail of UML models we selected a set of models which we can import into MetricView Evolution and we set up some research questions we would like to get answered. By visualizing these models we will try to answer the following research questions:

1. What are the differences and similarities in the use of level of detail for different models and what does this say about the type of UML model?
2. Are there any similarities or differences between
    a. class diagrams regarding the level of detail metrics?
    b. sequence diagrams regarding the level of detail metrics?
3. What can we say about the positioning of classes in a class diagram in relation to their LoD?
4. Do classes that have a high class LoD imply that the sequence diagrams in which they occur also have a high sequence diagram LoD?
5. If a class occurs in sequence diagrams does that imply that the class LoD of that class is higher than the class LoD of classes that do not occur in sequence diagrams?

5. Proposed solution

We will try to extend the current state of MetricView Evolution in such a way that it will be able to visualize the LoD metrics. The reason why we chose to improve MetricView Evolution is because of its intuitiveness and already existing capability of visualizing (other) software metrics on top of UML models. This way we can use the existing functionality and extend it to visualize the LoD metrics. To do a proper analysis to answer these research questions we need models that can be analyzed. We selected eight models which can be imported into our extended version of MetricView Evolution and analyze the LoD metrics for these models.

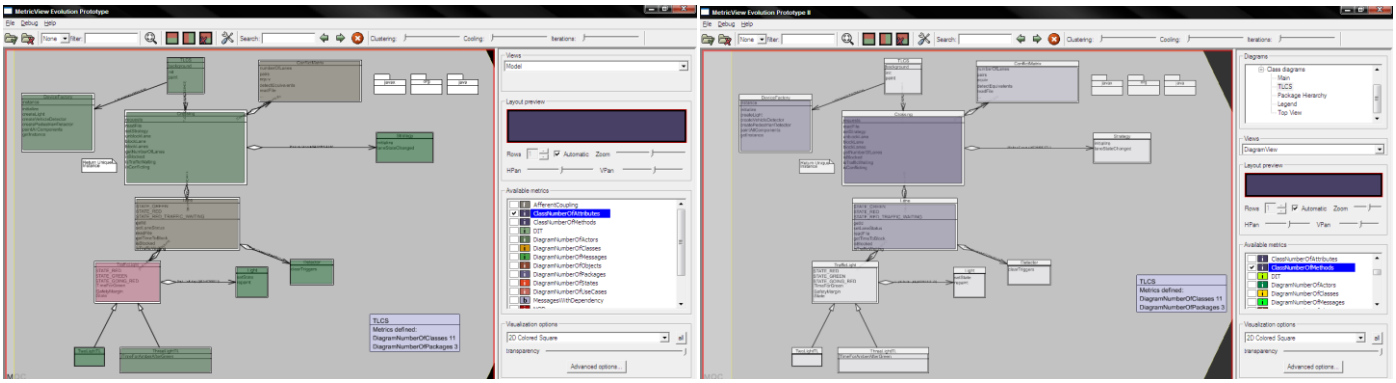


Figure 2: Visualization of a model for one metric. The figure on the left shows the old state of visualization. It shows a visualization of metric values that is based on color offsets. The figure on the right shows the new state of visualization in which the visualization of metric values is based on transparency.

6. Implementation

As mentioned before the implementation is done by extending the existing tool "MetricView Evolution". When extending MetricView Evolution it is important to take in account the current architecture and try to extend the tool using this specific architecture for the sake of extensibility. The LoD metrics consist of several sub-metrics as stated in section 3. All of these metrics have to be implemented in a similar way as the current available metrics are implemented. Implementation is done using python functions and XML declarations which are read by MetricView Evolution when calculating metrics for a certain model.

After implementing the level of detail metrics they have to be tested by using some example models with which we can verify that the implemented calculations are correct. Also the visualization of the metrics have to be studied to see whether the results are clear and we can get information out of the visualization.

In the old state of MetricView Evolution a visualization of a software metric is done by using a color. Let's say for example that our visualized metric has the standard color blue. When a model element has a high value for this metric it's color will be blue. However when a model element has a low value for this metric it's color become a color offset and turn into another color like red or green. This way of representing an element's performance for a metric is obvious when one wants to visualize the performance of a model for one metric (See Figure 2). However if one wants to see the performance of a model for more than one metric this view gets a little confusing. A user might not be able distinguish the different metrics and thus gets confused (See Figure 3).

To improve the visualization of a design metric we changed the way a metric is visualized. Instead of working with a color offset which leads to different colors we chose to work with color transparency as an indicator for high or low values. High values get a high transparency rate and low values get a low transparency rate. By using transparency each metric has the same color representation for high and low values and thus reduces the confusion that was created by the old way of visualizing metrics. Also by using transparency we provide a way in which a user can see very fast what elements have a high or low value for a certain metric by spotting the transparency rate for elements (See Figure 2 and Figure 3).

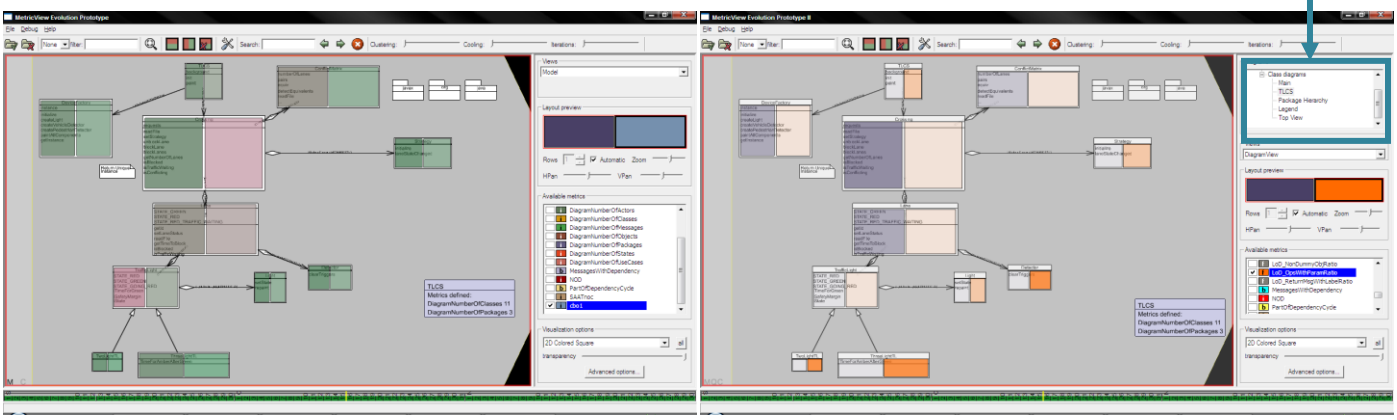


Figure 3: Visualization of a model for two metrics. The figure on the left shows the old state of visualization. It shows a visualization of a model for two metrics. The use of color offsets can confuse the user because colors can have a different meaning for different metrics. The figure on the right shows the new state of visualization. It also shows a visualization of a model for two metrics. The new state of visualization makes it easy to spot high and low values for both metrics (high and low transparency levels respectively).

Another issue we found was a performance issue. When we tried to load big models that contain many diagrams the browsing functionality failed by being way to slow. The main cause for this problem is the "Model View". The model view presents all class diagrams, sequence diagrams, use-case diagrams and statechart diagrams and thus has to re-render every diagram on a refresh. This process is quite intense for a processor. To get rid of this issue we created a "Diagram View" which only visualizes one diagram at a time so the rendering will not be slowed down by the rendering other diagrams. Implementing the "Diagram View" also implied to implement a "Diagram Explorer" in which a user can browse the diagrams of a model. An example can be seen in Figure 3 on the top right corner where "Diagrams" is stated and a tree-browser functionality is offered in the list box.

After solving these issues we wanted to load several models to analyze the LoD metrics. However this brought us to the next issue. MetricView was only able to load models created by Rational Rose and Rational XDE. Both Rational Rose as Rational XDE are modeling tools that have become a little outdated and are no longer used a lot for modeling. So we were not able to find a lot of models created by Rational Rose nor by Rational XDE. To solve this issue we decided to write a parser for the more recent modeling tool Rational RSA. The parser has to translate the original RSA model into MetricView Evolution's database. To do so we created this parser as much the same as the existing parsers (for consistency's purpose).

Summarizing this implementation section we can state that we made the following improvements and extensions:

- We implemented a way of visualizing the LoD metrics using MetricView Evolution
- We improved the way metric values are visualized by using transparency instead of color offsets. This way every metric has its own color and high or low values for metrics can be visualized by showing high or low transparency values for that color.

- We improved the performance of MetricView Evolution by adding a “Diagram View” which provides visualization of a single diagram and by adding the “Diagram Explorer” to the user interface.
- We added functionality to load models created by Rational RSA by creating a RSA parser that translates the model into a database.

7. Analysis

This chapter provides the analysis we did by using our extended version of MetricView Evolution. To do this analysis we used eight models to import into MetricView Evolution. Before answering the research questions we first describe some statistics about the models we used to answer these research questions. The models we used contained several diagrams which are mainly class diagrams. Four out of eight models also contain sequence diagrams so we can analyze both the class level of detail and the sequence diagram level of detail for these four models.

Table 1: Model characteristics

Model	# of class diagrams	average # of classes per diagram	# of sequence diagrams
GLL	2	16,00	21
AMF	11	12,72	0
DM	4	8,50	0
WNL	37	9,00	0
BHN	38	2,39	104
PTSM	35	5,11	172
AMDL	15	5,00	8
HAM	10	6,70	0

To differentiate between different models regarding for both class LoD and sequence diagram LoD both measures the composed metrics are given. For the class LoD the CD_{aop} and CD_{asc} are specified which stand for attribute and operation detailedness and association detailedness respectively. For the sequence diagram LoD the SD_{obj} and SD_{msg} are specified which stand for object detailedness and message detailedness respectively (See section 3. Level of detail metrics).

The class LoD and sequence diagram LoD are rated on a scale which can be very low, low, moderate, high or very high. The assignment of a certain rating for a model has been done using MetricView Evolution. The model has been analyzed using the “Model View” of MetricView Evolution. Then for classes the metrics Class LoD, CD_{aop} and CD_{asc} have been selected to visualize, for sequence diagrams the metrics sequence diagram LoD, SD_{obj} and SD_{msg} have been selected to visualize. According to this visualization of the LoD metrics for each model a rating has been assigned for the visualized metric. So for example if in a model the class LoD shows very much color (low transparency which indicates low value) and all red borders (indicating low values) then the rating low will be assigned as class LoD. When for example there are about as much classes that show much color (indicating low value) as there are classes that show little color (indicating high value) then the rating moderate will be assigned as class LoD.

Table 2: Class level of detail ratings.

Model	Class LoD	$\mathbf{CD_{aop}}$	$\mathbf{CD_{asc}}$
GLL	moderate	very low	high
AMF	low	moderate	low
DM	low	low	moderate
WNL	low	very low	moderate
BHN	low	low	low
PTSM	low	low	low
AMDL	moderate	moderate	moderate
HAM	moderate	high	low

Table 3: Sequence diagram level of detail ratings.

Model	Sequence Diagram LoD	$\mathbf{SD_{obj}}$	$\mathbf{SD_{msg}}$
GLL	low	high	very low
BHN	moderate	moderate	moderate
PTSM	high	high	moderate
AMDL	moderate	low	high

Table 2 presents the class LoD for all models. It also shows how the class LoD is constructed from the lower-level CD_{aop} and CD_{asc} LoD measures. Table 3 presents the sequence diagram LoD for all models. It also shows how the sequence diagram LoD is constructed from the lower-level SD_{obj} and SD_{msg} LoD measures

7.1 RQ1: What are the differences and similarities in the used level of detail for different models?

Table 2 presents the class level of detail for all models. The class level of detail for these models varies between low and moderate. This indicates that all of the models have not been modeled very detailed in the class diagrams which makes them similar in this way. Next to the class level of detail this table also shows the specification of this level of detail into attribute and operation detailedness and association detailedness. These values vary between very low and high and thus provide more differences between the models. So we can say that the models are similar in the way that all the models have a low class level of detail and that the models differ in the way the class level of detail is constructed from the attribute and operation detailedness and the association detailedness.

Table 3 presents the sequence diagram level of detail for the models that contain sequence diagrams. The sequence diagram level of detail average rating for all models is higher than the average for the class level of detail which implies that if there are any sequence diagrams available they are modeled more detailed than the class diagrams. The sequence diagram level of detail differs more than the class level of detail both in the object detailedness as in the message detailedness the models and there are no real similarities in the sequence diagram level of detail.

If we look at the relation between the class level of detail and the sequence diagram level of detail there seems to be a connection between the attribute and operation detailedness for the class level of detail

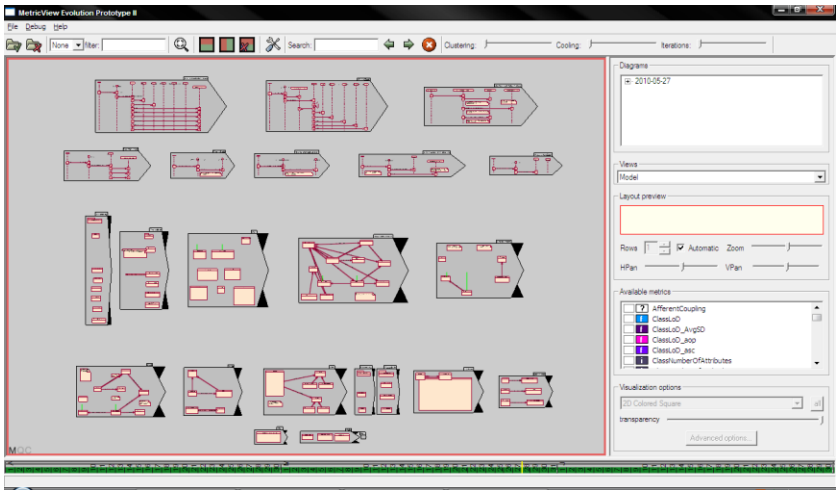


Figure 4. Example of a model that provides some large class diagrams and more smaller class diagrams.

and the message detailedness for the sequence diagram level of detail. The higher the rating for the attribute and operation detailedness the higher the rating for message detailedness is (GLL gives very low and very low, BHN gives low and moderate, PTSM gives low and moderate, AMDL gives moderate and high for CD_{aop} and SD_{msg} respectively). Though we think that this is not that surprising because when classes contain operations that can be used as a message in the sequence diagrams it is likely that there will be a reference between these two elements.

7.2 RQ2: Are there any similarities or differences between: 1.) class diagrams regarding the level of detail metrics and 2.) sequence diagrams regarding the level of detail metrics?

To answer this research question we scanned all models and tried to find any similarities or differences that are noticeable.

Similarities and differences in class diagrams

Most of the models we have seen offer a common view. They provide some large class diagrams (size in terms of number of classes per diagram)and then more small to very small class diagrams. When we look at the class LoD for these diagrams we can say that larger diagrams generally offer more class LoD than the smaller diagrams. So the larger class diagrams are modeled with more detail and seem to be more important than the smaller diagrams. The differences we could find is that some models are modeled more detailed than other models and thus provide a larger spread of the class LoD. The differences across models is probably due to the different type of models (analysis, architecture, design).

Similarities and differences in sequence diagrams

When analyzing the sequence diagrams we could only look at four models because there are four out of eight models containing sequence diagrams. For the sequence diagrams we generally observed the same results: the models contain some large sequence diagrams (size in terms of number of objects and messages) and small sequence diagrams. The difference between the larger and smaller sequence diagrams is that the sequence diagram LoD tends to be higher in larger sequence diagrams than in smaller sequence diagrams. The differences in the sequence diagram LoD are probably due to the larger sequence diagrams covering more important behavior than the smaller sequence diagrams.

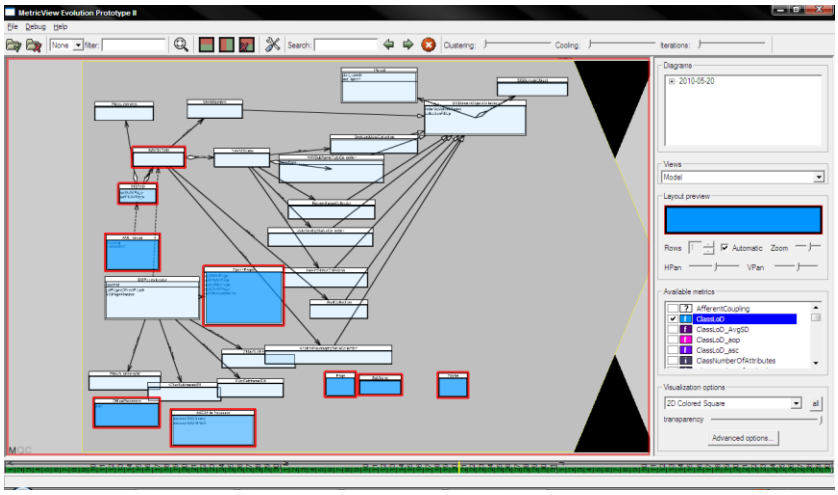


Figure 5: Example of a class diagram with the class LoD metric selected. The classes with a red border around it imply a low value for the selected metric (in this case the class LoD). Most of these classes with a border are positioned around the borders of the diagram.

7.3 RQ3: What can we say about the positioning of classes in a class diagram in relation to their LoD?

To answer this research question we have to analyze models that have class diagrams with sufficient variance of class LoD. As we can see in Table 2 there are only three models that provide such a variance: GLL, AMDL and HAM. We found these three models to provide enough variance in LoD because when analyzing these models using MetricView Evolution there are enough differences between class LoD in class diagrams such that we can distinguish high values from moderate and low values. This way we will be able to analyze the position of classes in relation to their LoD.

For all of these three models we observed that classes that have a low level of detail are likely to be positioned near to the borders of a class diagram [see Figure 5].

7.4 RQ4: Do classes that have a high level of detail imply that the sequence diagrams in which they occurs also have a high level of detail?

To answer this research question we have to analyze the classes that have high LoD values. To analyze whether the sequence diagrams in which these classes occur have a high sequence diagram LoD we use the $ClassLoD_{AvgSD}$ metric. This metric measures the average sequence diagram LoD of all sequence diagrams in which a class occurs.

For the class diagrams we studied there seemed to be a connection between the LoD of a class and the average LoD of the sequence diagrams in which that class appears. Many of the class diagrams showed that when a class has a high class LoD it is very likely to have a high average sequence diagram LoD. This implies that the sequence diagrams in which the class occurs are modeled with a high level of detail [see Figure 6]. Note that a low average sequence diagram LoD can be due to the class not appearing in any sequence diagrams.

So we can say that when a class has a high class LoD it is very likely that the class average sequence diagram level of detail is also high. This implies that the sequence diagrams where the class occurs in tend to have a high sequence diagram LoD.

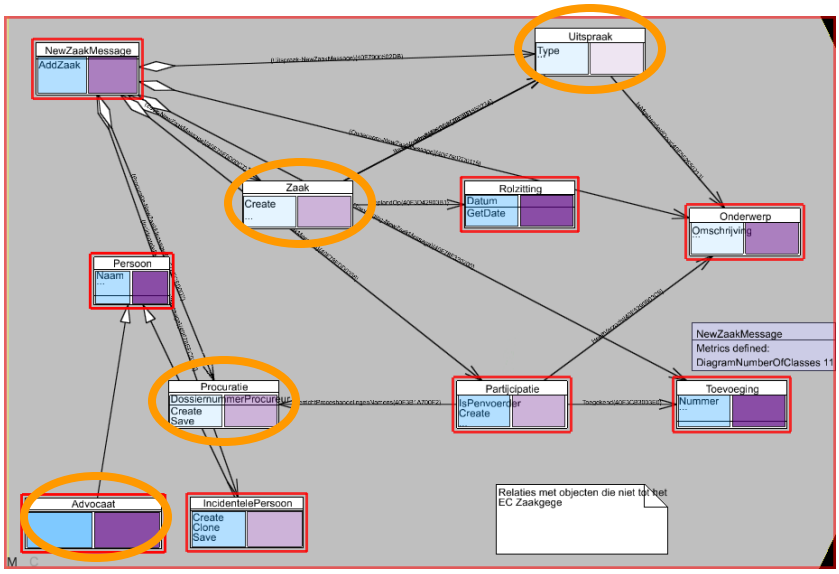


Figure 6: Visualization of the two metrics class LoD (light blue) and average sequence diagram LoD (purple). This figure provides a class diagram that shows a connection between the class LoD and the average sequence diagram LoD. If a class has a certain level of transparency for the blue color, the transparency for the purple color tends to be almost the same (see for example the elements with the orange circles)which indicates their relation. High level of transparency meaning high level of detail for both class level of detail as for average sequence diagram level of detail.

7.5 RQ5: If a class occurs in sequence diagrams does that imply that the class LoD for that class is higher than the class LoD for classes that do not occur in sequence diagrams?

To answer this research question we have to compare the class LoD of classes that are used in sequence diagrams and that of classes that are not used in sequence diagrams. To do so we categorize all classes into two groups:

1 Classes that occur in sequence diagrams further annotated as “SD and CD”.
2 Classes that do not occur in sequence diagrams further annotated as “CD only”.

Table 4. Representation of gathered data. For each model we divided the classes into one of the two groups and then calculated the average class level of detail for this group.

Model	**SD and CD**		**CD only**	
	Nr of classes	*Average Class LoD*	*Nr of classes*	*Average Class LoD*
GLL	13	1,474	19	1,332
BHN	85	0,464	6	1,638
PTSM	100	0,765	72	0,603
AMDL	35	2,212	40	1,349

Table 4 shows the quantitative results. For each model it shows the number of classes and the average class LoD of that group. For three out of four of these models the average class LoD for classes that occur in sequence diagrams is higher than the average class LoD for classes that do not occur in sequence diagrams. The only exception is the BHN model which has a different distribution in the number of classes between the two groups (85 in “SD and CD” and 6 in “CD only”, where the other models are distributed more evenly).

To see whether the difference in average LoD is significant we ran a statistical analysis using SPSS. Because the BHN model shows different characteristics than the rest of the models and the BHN model also influences the results of the descriptive statistics (see the differences between the mean values of Table 5 and Table 6)**Fout! Verwijzingsbron niet gevonden.** they might influence the test results as well. Therefore we ran the statistical analysis twice, first time including the BHN model and second time without the BHN model.

Table 5. Descriptive statistics including the BHN model

Class LoD	Group	Statistic	Value
	SD and CD	Mean	0.9121
		Median	1.0000
		Minimum	0.0000
		Maximum	3.0000
		Std. Deviation	0.8752
	CD only	Mean	0.9740
		Median	1.0000
		Minimum	0.0000
		Maximum	2.3333
		Std Deviation	0.7412

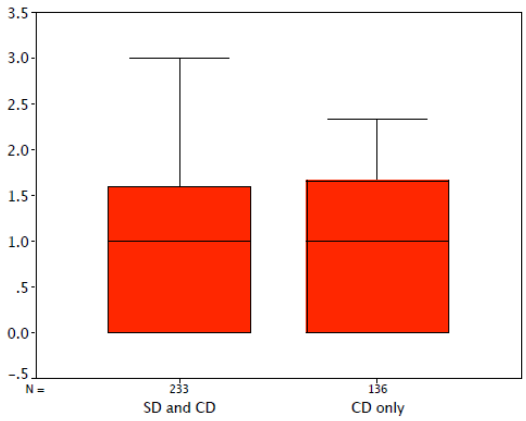


Figure 7: Box plot for descriptive statistics including BHN model

Table 6. Descriptive statistics excluding the BHN model

Class LoD	Group	Statistic	Value
	SD and CD	Mean	1.1695
		Median	1.0000
		Minimum	0.0000
		Maximum	3.0000
		Std. Deviation	0.9392
	CD only	Mean	0.9362
		Median	1.0000
		Minimum	0.0000
		Maximum	2.0000
		Std Deviation	1.1695

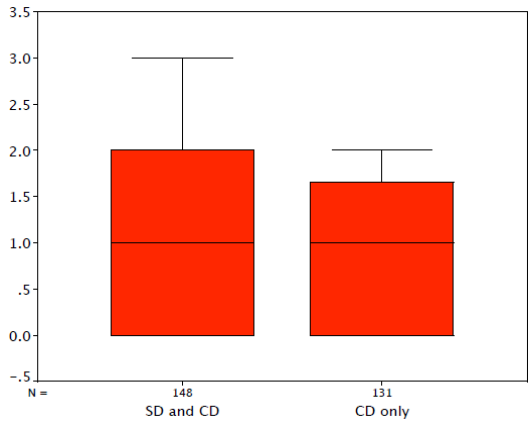


Figure 8: Box plot for descriptive statistics excluding BHN model

The results for both descriptive statistics are quite different and this implies that the BHN model is a disturbing factor in this analysis. Because the BHN model can be seen as an extreme case in this analysis we can discard it and see if the analysis without the BHN model leads us to any significant difference for this statistic. To do so we ran a Mann-Whitney Test.

Table 7. Mann-Whitney Test – Ranks of the measured class level of detail variable (BHN model excluded)

Class LoD	Group	N	Mean Rank	Sum of Ranks
	SD and CD	148	148.53	21982.00
	CD only	131	130.37	17078.00
	Total	279		

Table 8. Mann-Whitney Test – The significance of differences in the class level of detail variable (BHN model excluded)

	Class LoD
Mann-Whitney U	8432.000
Wilcoxon W	17078.000
Z	-1.939
Asymp. Sig. (2-tailed)	0.053

Table 7 provides the comparisons of mean ranks for the two groups. As we can see the mean ranks for the SD and CD group is about 18 points higher than the mean rank for the CD only group which indicates that the SD and CD group have higher values for the class level of detail.

Table 8 provides us a significant difference of 0.053. Despite the fact that this is 0.003 higher than the accepted 0.05 significance level we believe that this indicates that there is a significant difference between the two groups for the class LoD statistic. Sequence diagrams are used to model scenarios and functionality. If a class occurs in the sequence diagrams it is more likely to be an important class because it is used to model behavior as well as for structural analysis and thus tends to have a higher class LoD than classes that do not occur in any sequence diagrams and are used for structural analysis only.

Both these quantitative and statistical analysis lead us to the conclusion that the class level of detail for a class is higher when it is used in sequence diagrams.

8. Model analysis

This section gives an informal analysis of the LoD used in the models that are described in section 7. We will present some screenshots that are made using MetricView Evolution and we will describe the contents of these screenshots.

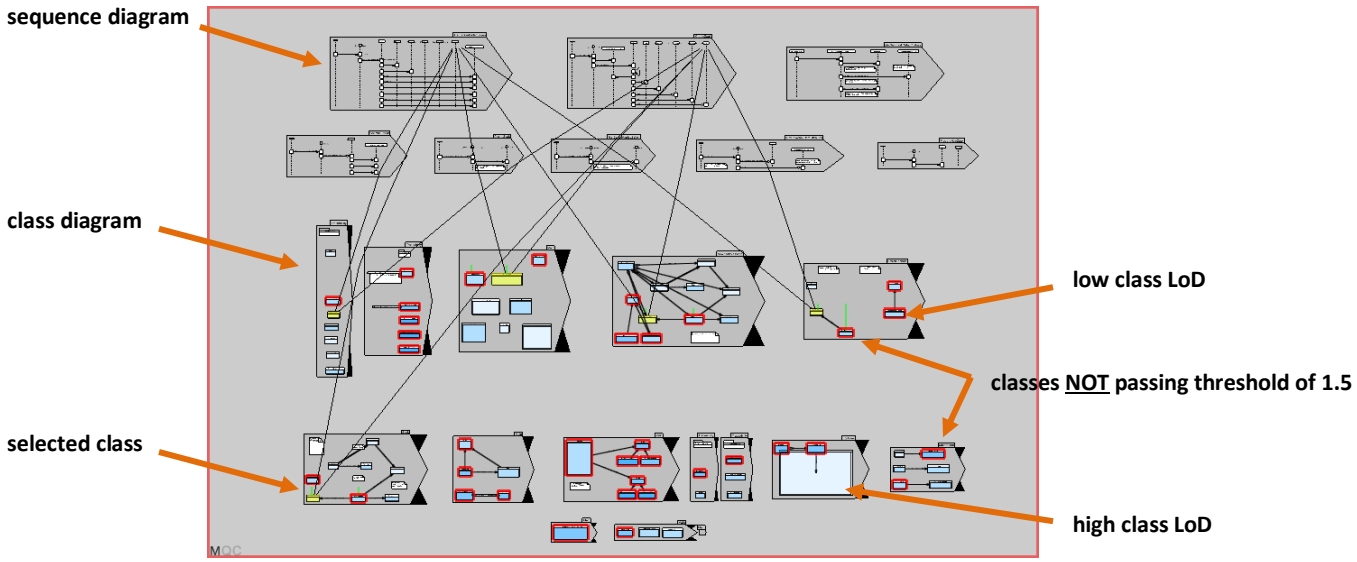


Figure 9. Screenshot (1) of a model that was imported by MetricView Evolution.

Figure 9 is a screenshot that was taken from MetricView Evolution when using the Model Window. The selected metric is the class LoD metric which is represented by the color light blue. Classes that have very little color (high transparency) represent a high class LoD value and classes that have much color (low transparency) represent a low class LoD. Next to the light blue color we can also see classes that have are surrounded with a red border. These classes have a class LoD value that does not pass the threshold of 1.5 for the class LoD metric. Note that the maximum value a class can have for the class LoD is 4 and the minimum value a class can have is 0 (see section 3). As we can see in this model there is a selected class (the yellow colored class, which happens to occur in more than one class diagram). This model view also represents the relations between a class and it's corresponding object occurrences in the sequence diagrams (visualized by the black lines across the model). This model offers both sequence diagrams and class diagrams. What we can see in the class diagrams is that there are certain differences in the class LoD values between the several classes. We can also see that the larger class diagrams generally offer higher LoD values than the smaller class diagrams (we can see this by the coloring of classes for visualizing high and low values).

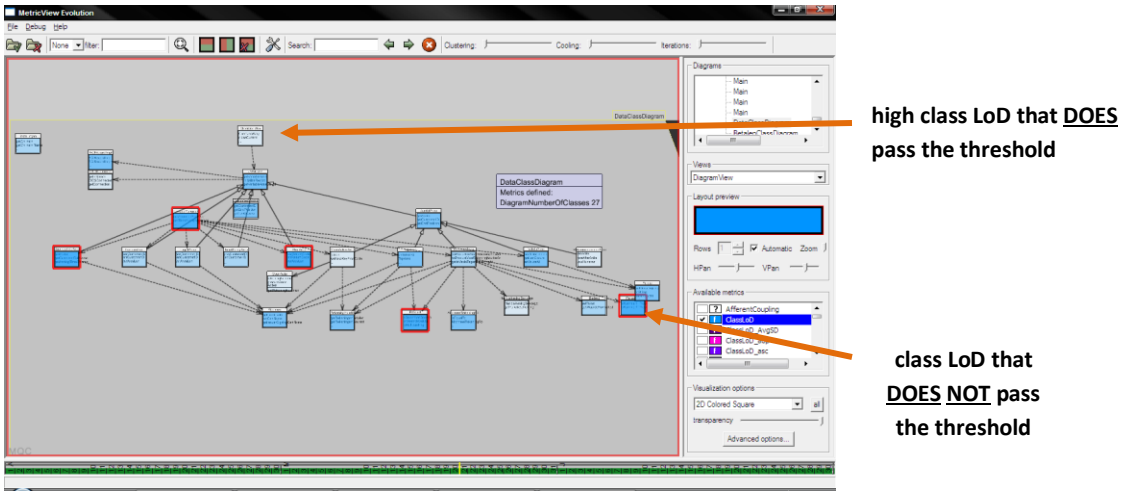


Figure 10. Screenshot (2) of a model that was imported by MetricView Evolution.

Figure 10 presents a screenshot for one class diagram. The selected metric is the class LoD metric which is represented by the color light blue. This specific class diagram offers a nice spread of the class LoD values as we can see in the differences in transparency that is used for the classes (high transparency implying high class LoD values, low transparency implying low class LoD values). This class diagram offers a layered structure which might indicate different layers in a model.

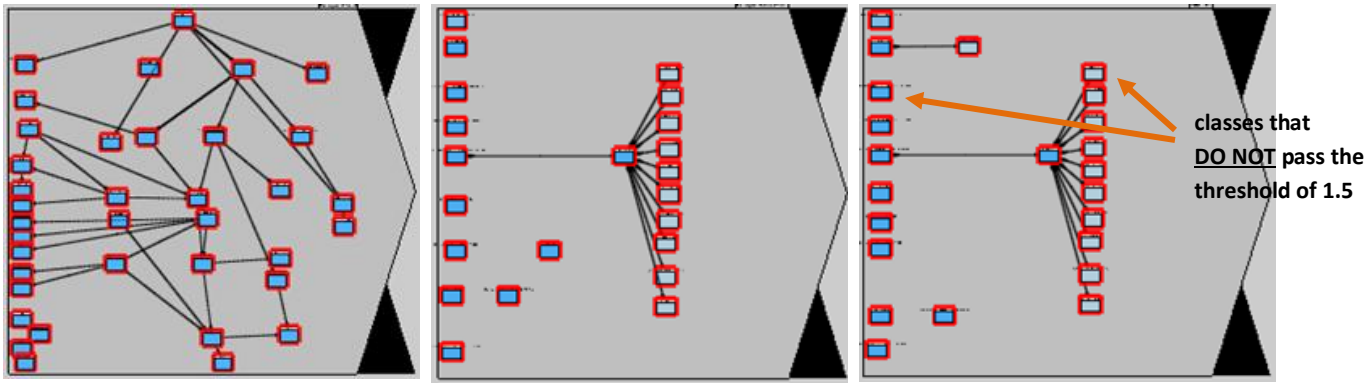


Figure 11. Three screenshot of a model that was imported by MetricView Evolution

Figure 11 presents three screenshots of class diagrams of a model that was imported by MetricView Evolution. The selected metric is the class LoD metric which is represented by the color light blue and we can immediately see that for these three class diagrams all classes do not pass the threshold for the class LoD metric of 1.5 (indicated by the red borders). Though the class LoD values being low there are some small differences in the two rightmost class diagrams where the classes on the right side of the model have a higher class LoD value than the classes on the left side of the model.

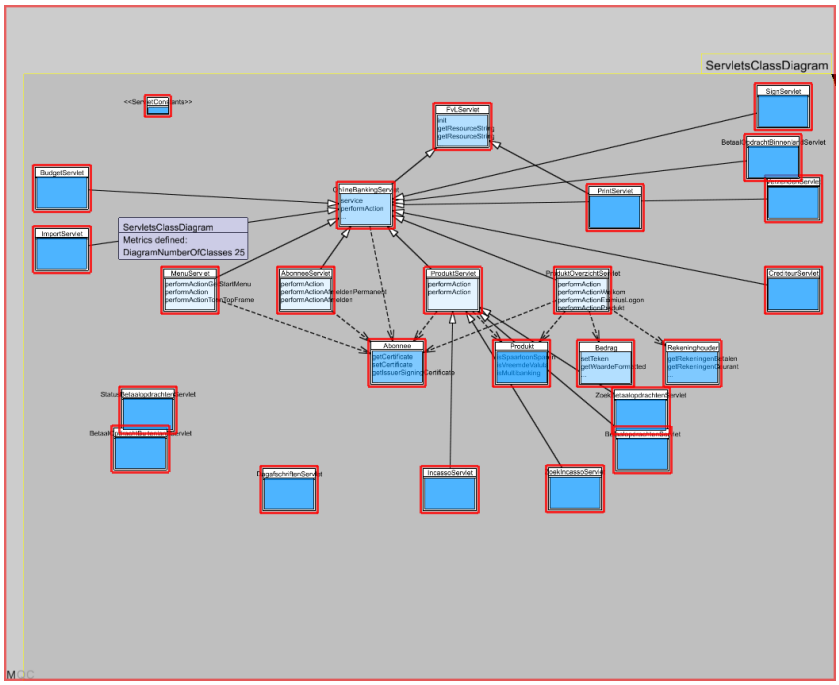


Figure 12. Screenshot (3) of a model that was imported by MetricView Evolution.

Figure 12 presents a screenshot of a class diagram. The currently selected metric is the class LoD metric which is represented by the color light blue. We can observe that all the classes in this class diagram do not pass the class LoD threshold of 1.5 (note that every class is surrounded with a red border which indicates not passing the threshold for a certain metric, which in this case is the class LoD). Even though the class LoD values of these classes are low and do not pass the threshold of 1.5 we can observe that there are differences between the classes in the class LoD values (note the differences in transparency) that probably range from 0 till nearly 1.5.

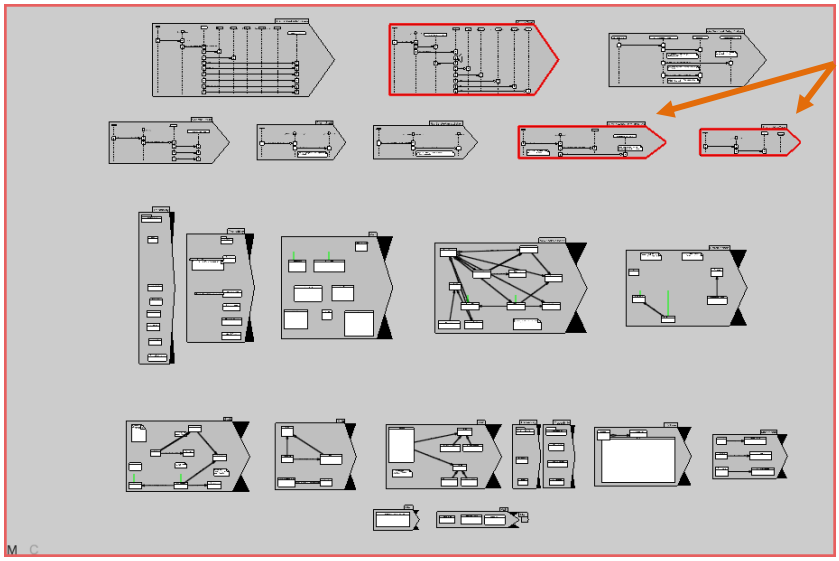
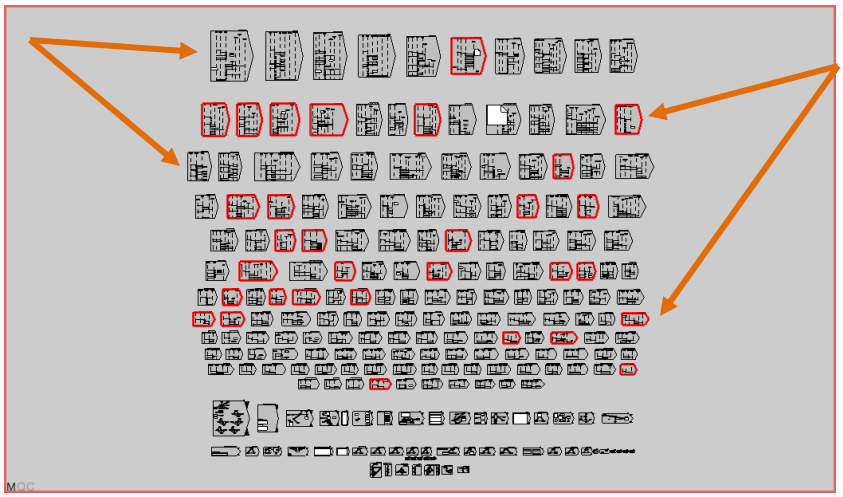


Figure 13. Screenshot (4) of a model that was imported by MetricView Evolution.

Figure 13 presents a screenshot that was taken from the same model that is used in Figure 9. However in this model the selected metric is the sequence diagram LoD metric. Visualization for metrics that are at diagram level are done in a different way than for metrics that are at class level: there is no use of coloring transparency but only use of red borders indicating that a sequence diagram does not pass the threshold for a certain metric. In this case the metric is the sequence diagram LoD and thus there are three classes that do not pass the sequence diagram LoD threshold of 1.5. Note that the minimum value for the sequence diagram LoD is 0 and the maximum value for the sequence diagram LoD is 5 (see section 3).

sequence diagram LoD that DO pass the threshold of 1.5



sequence diagram LoD that DO NOT pass the threshold of 1.5

Figure 14. Screenshot (5) of a model that was imported by MetricView Evolution.

Figure 14 presents a screenshot of a very large model containing both a lot of class diagrams as sequence diagrams. The selected metric for this model is the sequence diagram LoD metric. As we can see for this model nearly 70% of all sequence diagrams are visualized without a red border. These sequence diagrams have a sequence diagram LoD value that passes the sequence diagram LoD threshold of 1.5.

9. Conclusion

In this paper we showed how we are able to visualize the level of detail metrics on top UML models using our extended version of MetricView Evolution. Our implementation of MetricView Evolution has been very helpful doing this analysis as it is very easy to visualize the level of detail metrics in a UML model. It gives a fast and clear overview of the spread in which level of detail is used and it also indicates parts of the system that might need attention. MetricView Evolution has also been very helpful in finding relations between classes and sequence diagrams by the given "LangeDiagram" which connects classes to the corresponding objects in sequence diagrams.

Doing the analysis using MetricView Evolution we observed the following facts:

- We observed that there seems to be a relation between the CD_{aop} and the SD_{msg} LoD measures. A higher CD_{aop} leads to a higher SD_{msg} value as well.
- The models that we analyzed had the common property in the way the class LoD is used in larger and smaller class diagrams (size in terms of number of classes). The models generally had the same property for the sequence diagrams LoD, which also is higher for larger sequence diagrams than for smaller sequence diagrams (size in terms of number of objects and messages).
- We observed that classes that have a low class LoD tend to be positioned near the borders of a class diagram.
- We observed that if a class has a high class LoD it is very likely that the average sequence diagram LoD is high which implies that the sequence diagrams in which the class occurs tend to have a high sequence diagram LoD.
- Doing a statistical analysis we proved that there is a significant difference between the classes that occur in sequence diagrams and the classes that do not occur in sequence diagrams in relation to their class LoD. If a class occurs in sequence diagrams it generally has a higher class LoD.

Note that the above described analysis has some limitations with respect to the number of models that are used to do the analysis on. If we had more models to do the analysis on our results would be more reliable but in our case it was about showing that we can use MetricView Evolution to do an analysis about the level of detail design metrics.

10. Future work

The level of detail metrics are interesting to investigate and there is some future work left in this field of study. First of all there are some features in MetricView Evolution that can be improved such as the visualization of the sequence diagrams imported from files created by Rational Software Architect. The current visualization does not suffice because the order and position in which messages and objects are displayed is incorrect according to the original RSA file. Another improvement is to implement features to export the level of detail metrics to either a csv file or an excel sheet. By doing so it is easy to create statistics and investigate the level of detail metrics in a different way. Another improvement which can add substantial benefits is the implementation of a built-in statistical analysis. This feature can give reports about an entire model without a user even looking at the model.

References

1. NUGROHO, A. AND CHAUDRON, M. R. V. Level of Detail in UML Models and its Relation with Defect Density in Source Code – *An Empirical Assessment of an Industrial Java System (2009)*.
2. SDMETRICS. The UML design quality metrics tool, http://www.sdmetrics.com
3. METRICVIEW EVOLUTION. The UML model visualization tool with software metrics, http://www.win.tue.nl/empanada/metricview/
4. TERMEER, M. MetricView: Visualization of Software Metrics on UML Architectures. (2005)
5. WIJNS M. MetricView Evolution – *Monitoring Architectural Quality (2006)*