Towards Flexible Object and Class Modeling Tools: An Experience Report

Andreas Kästner[1], Martin Gogolla[1], and Bran Selic[2]

[1] University of Bremen, Bremen, Germany, {andreask|gogolla}@cs.uni-bremen.de
[2] Malina Software Corp., Ottawa, Canada, selic@acm.org

Abstract. Recently, we have proposed an approach for class modeling starting from imperfect object models. The underlying process puts emphasis on the free expression of ideas of the developer through allowing to formulate incomplete and even inconsistent object diagrams. From these object diagrams, class diagrams are deduced. This paper reports on experiences with a supporting tool grouped into two categories. One category involves students at a university and how they get along with the general concept and the tool. The other category shows how researchers work with our proposed ideas.

Keywords: Experience report, Evaluation, UML object diagram, UML class diagram, Incremental transformation by example, Tool support

1 Introduction

To allow for more flexibility in modeling, it would be beneficial to have modeling tools which are not very restrictive with their syntax and in their development processes. To come closer to diagrams informally sketched on paper with a pencil, we proposed a lenient modeling approach [5,8]. Starting with object diagrams that can leave parts empty, we developed an automatic transformation to class diagrams as a plugin for the USE tool [3,4]. Both the object diagrams as well as the class diagrams use a flexible syntax which comes close to a drawing tool while still following an internal meta-model. Developers are given the option to incrementally develop the object diagram while receiving feedback from the class diagram. Similar approaches for developing specifications of system states in a very flexible way include [6,9,10].

This work is still in development and we try to get as much feedback as possible from potential users to incorporate their ideas. For this reason, we did two studies with two different potential user groups. First, we used the tool in a modeling course at university. Since one of the reasons of our work is to reduce the barrier to get started with modeling, it would be interesting to see how people with little modeling experience use our tool. Secondly, the approach should also help experts to sketch their ideas in an informal manner. For this reason, we developed a specific task for a group of people who already have some

experience in the modeling field. The task included a text in natural language and the experts were asked to use the tool to create an object diagram based on the described circumstances.

The main goal of these two studies is to gather ideas and see how different people work with the current state of our tool. This feedback can, and at the time of this writing was already, used for immediate improvements that focus on the needs of potential users. More formal and extensive studies that also test these improvements can then be done in the future.

The structure of the rest of this paper is as follows. Section 2 describes our previous work on this topic. Section 3 describes how the students worked with the tool. Section 4 explains the approach that the experts used to solve their task. Section 5 explains what was learned from our experiments and how it will influence future work. Finally, a conclusion is given in Sect. 6.

2 Context: From (Imperfect) Object Diagrams to (Imperfect) Class Diagrams

The basic idea of our approach is to consider UML object diagrams and extract as much information as possible during an automatic transformation to UML class diagrams. Or formulated more generally: we aim at a transformation from the instance level to the generic level of a model. However, since instances are just specific scenarios, the general case can not be perfectly described. To fill the gap, we introduced the concept of special markers to highlight where information is still missing <?> or even conflicting with other information <!>. We also allow incomplete input on the object side. To highlight optional parts that can be refined further, e.g. attributes or role names, we use the <+> marker.

An important part of this work is to allow for leniency during the development. The syntax should not force the user to do specific things, but more resemble drawing diagrams on paper. Because of this approach, there can be inconsistencies and missing information in the diagrams. In Fig. 1 for example, not every role name or association name is written in the "Input object diagram". The "Output class diagram" however shows as much information as possible.

A much more in-depth description of our work can be found in [8]. While that paper explains the concept in general, this contribution focuses on potential users and possible directions in which our research may go.

3 O2C in Practice: How Students Worked with the Concept

In this section, we present two examples created by two different students of the modeling course "Design of Information Systems"[1]. The students had the task to model a system in a context they could choose themselves. To help them achieve that goal, we introduced them to our object to class concept. Our idea

[1] http://www.db.informatik.uni-bremen.de/teaching/courses/ss2018_eis/

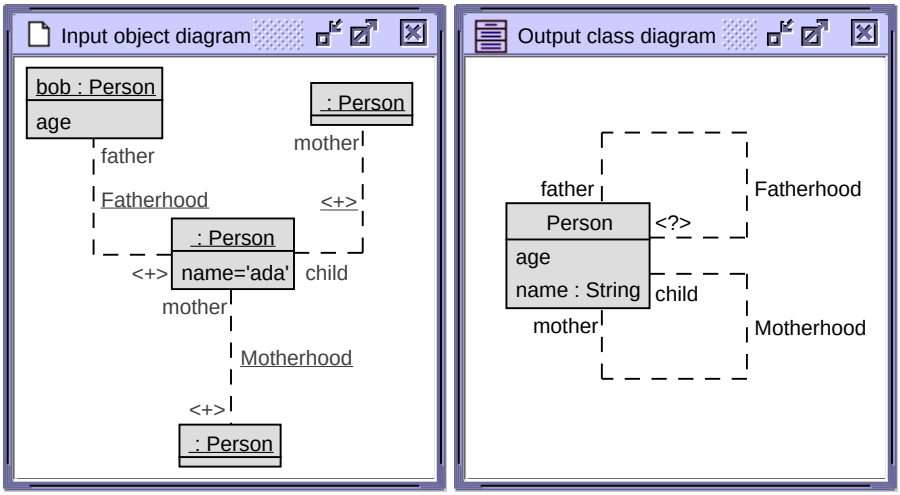

Fig. 1. An imperfect object diagram and the resulting imperfect class diagram.

that modeling may be easier when you start by looking at concrete instances was thus tested in a practical setting. The complete created diagrams are listed in [7].

3.1 Driving School Example

One student modeled a system for driving schools. The idea is to have a general approach, suitable for cars, planes, and boats. He created two specific scenarios in the form of object diagrams. The first one depicts an aviation academy and the second one depicts a driving school for cars. After the transformation, the two resulting class diagrams do not have a lot in common. Out of the 7 classes in the first diagram and the 6 classes in the second diagram, only three classes are identical. While the attributes of those classes match perfectly, in the one diagram, there is an association between them, while there is none in the other one. The student continued to merge the two automatically created diagrams into one final class diagram.

Another observation is that the student tried to fill out as much information as possible in the object diagram, while he could have left some parts empty.

The student also wrote a bit of criticism. While the general approach was deemed “a useful method for approaching a universal model”, the student did not like the usability of the tool. The main points of negative criticism involve the layout of the diagram and a missing feature for automatic saving.

3.2 Parachuting Example

Another student modeled a system for a drop zone in the parachuting context. The model revolves around groups of people who enter an aircraft to jump with

the help of a parachute. He created two scenarios in the form of two object diagrams. The focus of the first one lies in the administrative side of the drop zone. It includes aircrafts, their manufacturers and persons working for the drop zone. The second object diagram revolves around an actual jumping process, it includes groups of participating persons. The transformation of these two object diagrams results in two class diagrams. Like in the previous example, the student tried to merge these two diagrams into one large diagram.

After using the plugin, the student also gave some criticism. He said that the way the plugin shows incomplete parts in the form of dashed lines can be hard to see for larger diagrams. He proposed to use different colors, like red, instead. He also said that even for small models, like the one in this example, there are already large object diagrams necessary. It would be preferable to allow multiple object diagrams (for sub-scenarios) as input and merge them all into one class diagram during the transformation. He criticized that multiple objects can have the same identifier, for example after cloning an object, and that the plugin allowed that without error messages. Finally, there was some minor criticism that is specific to the implementation and not to the concept in general. He said that newly created objects are hard to find in the GUI because they can be covered by other objects. He said the button to delete attributes should only be available when an attribute is selected and that there should be a possibility to delete multiple objects at once.

4 O2C in Practice: How Experts Worked with the Concept

In this section we explain the task that was given to modeling experts and the results.

4.1 The Task

The idea was to give a modeling task to experts and see how they handle it with the help of the O2C plugin. A general idea of our procedure can be seen in Fig. 2.

The complete diagrams are listed in [7]. First, a class diagram was developed (Fig. 3). This diagram shows a simplified version of a traveling scenario where routes exist between buildings. From this class diagram, an object diagram was instantiated in which some of the traveling opportunities for the MODELS 2018 conference are listed. Then we described the object diagram in natural language as listed below:

> MODELS 2018 takes place at the “IT University of Copenhagen”. This building is in the town of “Copenhagen” in the country of “Denmark”.
>
> There are many ways to get to the conference. For example, you can take a plane to get from the building “Toronto Pearson International Airport” in the town of “Toronto” in the country of “Canada”

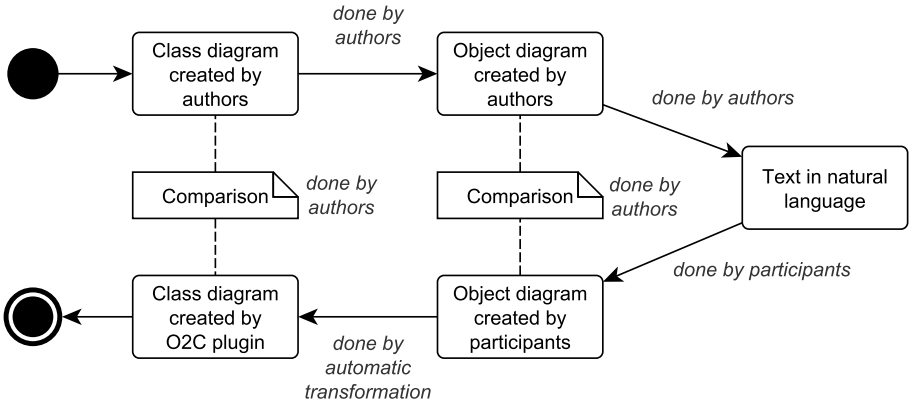

Fig. 2. The procedure of creating and executing the experiment.

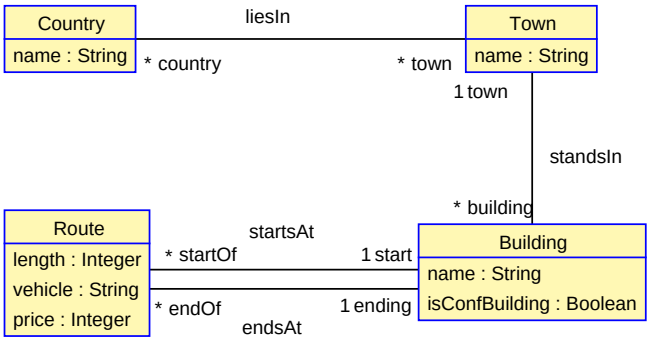

Fig. 3. The author intended model behind the task.

to the building “Copenhagen Airport”. This route is 6200km long and costs $500. You can also reach “Copenhagen” by train. If you are in the country “Germany”, you can start at the building “Hamburg Railway Station” in the town of “Hamburg” and take a direct train to the building “Copenhagen Railway Station”. This Route is 300km long and costs $100.

There are two official conference hotel buildings, the “Imperial Copenhagen” and the “Wakeup Copenhagen”.

The route from the “Copenhagen Railway Station” to the “Imperial Copenhagen” is 2km long and if you walk, it is also free. The route from the “Copenhagen Railway Station” to the “Wakeup Copenhagen” has the same length and you can walk as well. The route from the “Copenhagen Airport” to the “Imperial Copenhagen” is 10km long and costs

> $50 if you take a taxi. The route from the "Copenhagen Airport" to the "Wakeup Copenhagen" is 9km long and costs $45 if you take a taxi.
>
> Finally, to reach the conference building from one of these hotels, you can use public transport. The route from the "Imperial Copenhagen" to the "IT University of Copenhagen" is 3km long and costs 3$ with the bus. The route from the "Wakeup Copenhagen" to the "IT University of Copenhagen" is 3km long and costs $3 with the metro.

The participants had to read this text and create an object diagram, based on this text, with the help of our O2C plugin. After that, they also had to answer some questions concerning the usability of the plugin, based on the "System Usability Scale"[1],[2]. The full instructions were and are available online[2].

To evaluate what the participants have done, we compared our original object diagram with the diagram that the participants have created. We also compared our original class diagram with the class diagram that gets created by using our automatic transformation, as shown in Fig. 2.

4.2 First Solution

The first thing to notice when looking at this solution is that the participant used 5 classes instead of 4. The solution includes an additional "Conference" class instead of an attribute "isConfBuilding" in the "Building" class. The participant utilized the possibilities of the tool to leave role names empty but filled in every association name. Instead of using identifiers for the objects, this participant used anonymous objects. There are also some name differences for the same concept, like "cost" instead of "price".

4.3 Second Solution

This solution also includes 5 classes instead of 4. The 5th class is also a "Conference" class. There are also no "name" attributes used, this information is moved to the object identifier. The information which vehicle is used is also moved to the object identifier. In some cases, that leads to duplicate identifiers like two "Taxi" objects. The units "$" and "km" are handled differently. While they are not mentioned in the original object diagram, they are listed by this solution. That also leads to the attributes "distance" and "cost" to be of the type "String" instead of "Integer" after the O2C transformation. This participant completely labeled every association and every role name.

4.4 Third Solution

The first notable difference is that, in this solution, there are 3 associations between "Route" and "Building" in the class diagram instead of 2. The reason for that is actually a spelling mistake in the "startBuilding" role name. The

[2] `https://goo.gl/forms/fiQUwmZUhZsUXRhS2`

transformation identifies a new association this way that was not intended. This solution makes use of the functionality to leave parts empty but leaves so many parts empty that there is no association name for some of the associations. This solution also includes the units of length and cost in the attributes, which leads to "String" attributes after the transformation.

4.5 Results of the System Usability Scale

The System Usability Scale (SUS) is a simple, but well proven and valid method to measure usability [1]. One reason the SUS was used, is the easy comparability of the score. The SUS score results in a single number, which can be compared to thousands of other evaluations that were done before. The result of the calculation is a value between 0 and 100. In a meta-analysis [11], data was collected from over 5000 users across 500 different evaluations. It was found out, that an average score is 68 and a score of 80.3 puts the result into the top 10%. The average score for this evaluation was 75.

It has to be noted though that the participation rate was rather low with only four answers (the three solutions plus one student answer). Even though the results of the SUS are robust with a small number of participants [12], four is probably still too low and the results are just listed for completeness. However, the result of 75 comes really close to the result from our previous evaluation, which was 75.7 [8].

5 Lessons Learned

The results from our experiments lead to many topics of discussion. Many of those were only discovered through those experiments and were not clear to us before. The topics range from general discussions about modeling to minor implementation details of the tool.

5.1 General Discussion Topics

Newly identified future tasks One of the requested features involves *getting help from the already existing class diagram* while creating the object diagram. One could, for example, imagine partially labeled links during the creation process, that get their labels from the class side.

One interesting point of discussion is whether *multiple object diagrams*, explaining different scenarios, should be *merged into a single class diagram* or if there should be one class diagram for each object diagram. The conflicting ideas here are, on the one hand, a class diagram should cover a lot of cases. On the other hand, concrete scenarios seem to be a way some people think, so why shouldn't there be a class diagram for different cases? The downside, of course, would be that some classes would be in multiple diagrams, but that could be prevented with an appropriate class package architecture.

Something that a lot of participants tended to do was to *fill out every possible label for every object and link*. Our approach, however, allows leaving redundant parts empty. It might be necessary to better communicate this approach to users of the tool.

Another interesting topic is how to *handle the object identifiers*. Right now, these identifiers have no influence on the transformation and therefore the resulting class diagram. This means that right now, it is absolutely allowed to have duplicate object identifiers without any form of error message. This is already not a perfect approach, but once we allow object-valued attributes, it will lead to errors. One could imagine that in the future, the tool detects similar names when doing the merge and asks the modeler if they are meant to be the same.

One thing that became apparent during the experiment with experts, was that there are always *multiple possibilities to model* a given textual specification. For example, some participants used an additional "Conference" class, while in the prepared example, there was an attribute "isConferenceBuilding" that had roughly the same semantic. With future concepts like association classes, there will be similar occurrences where the same concept can be modeled in different ways.

Another limitation of UML came into focus during the experiment. When a numerical attribute also has a unit (e.g. price=$12), would that be an attribute of type string? A monetary data item is more than just numbers, so that would probably the best solution.

Known future tasks Some of the proposed ideas by participants were already known to us but were not yet implemented. This includes *support for further concepts* like association classes, part-whole relationships, and n-ary associations. It would also be good to improve the functionality to allow for object-valued attributes and enumeration-valued attributes.

The *order of attributes* also has to be addressed in the future. The order on the object side (e.g. first name and last name next to each other) should be retained on the class side. Right now, the order of the object attributes does not influence the order of the attributes in the classes. Depending on the input, this is a non-trivial problem, because there may also be conflicts within the order.

5.2 Layout Issues

There are some issues that do not clearly fit into the categories of either general approaches or tool specific problems. The most important one would be layout. Of course diagram layout is strongly connected to the specific tool, but it is also a very interesting research topic with many possible approaches.

One participant noticed that it helps to *hide parts of the diagram* when trying to *develop a good layout*. This could be the possibility to hide things like role names or association names, but also objects. To not overwhelm users with too many options, there could be an order like only showing association names and only later showing the role names. Another approach could be to not fully hide

parts of the diagram, but use a different color to grey-out parts and remove them from the main focus. A more radical approach to hide parts would be to work with multiple object diagrams for different contexts. This way complete diagrams would be hidden and the user can focus on one specific context at a time.

Right now we use *dashed lines* to highlight incomplete objects or links. A participant made the remark that it would be more obvious to *use colors* like red instead. That was actually done by us in a previous version of the tool but then removed in favor of the dashed lines. This shows that not every implementation detail is favored by all users. Different users with different expectations, expect different implementations. Also, in the interest of not discriminating against individuals who do not perceive the full range of colors (i.e., color blind people), the use of color as a primary differentiator is discouraged. Furthermore, support for mainstream black and white printouts is most common.

5.3 Topics Regarding the Implementation of the Tool

There was also a lot of feedback regarding the implementation of the tool itself that has little to do with the general concept. We categorized these ideas into the standard software improvement issues "usability improvements" and "bugs". We list these topics in a short way, without much discussion, since the other lessons learned from this chapter are more important and require more space.

Usability improvements: (a) The possibility to clone links or at least use existing labels in the same context as suggestions to improve the input process, (b) Aligning objects and classes, a functionality that is possible in the basic USE tool, but not in our plugin, (c) The possibility to edit multiple objects or links at the same time (e.g. selecting multiple objects and when they share an attribute, making it possible to edit all attributes at once), (d) Creating empty link "elements" that can exist on their own and only later connecting them to adjacent objects, (e) More key bindings (e.g. ctrl+c ctrl+v for creating clones and ctrl+x/del for removing elements), (f) The possibility to load .soil files through the GUI, (g) Automatic saving of the object diagram to lower the chance of losing progress, (h) Making it more clear where newly created objects are positioned by somehow highlighting them, and (i) The possibility to delete multiple objects at once.

Bugs: (a) A bug leads to the problem that once a link is destroyed, it always gets destroyed three times, which can also be seen in the soil script, (b) On the object side, it is only possible to save the position of the objects in the layout file. It would be preferable to also save the position of links and their labels. On the class side, the layout saving functionality does not work at all and has to be corrected, (c) The undo/redo function does not work as expected, and (d) The instruction readme file that comes with the plugin is unclear about how loading data from SOIL files works.

6 Conclusion

Our ongoing work on this topic was supported by an evaluation. Both the answers from the students, as well as the answers from the experts led to new ideas about what areas should be improved first. A major category that became again apparent to us is the need for good layout possibilities. Layout is an important part of understanding diagrams and needs to be given more attention in future work.

Usability is a crucial topic. However, as builders of an academic prototype, it is not our primary focus. We see our task as giving triggers to commercial or open source tool builders once the theoretical work is done. Until then, the features may change frequently and thus we will focus on getting the features done first and polish the usability later.

Acknowledgements

We would like to thank Nisha Desai, Khanh-Hoang Doan, Julian Stoick and Moritz Weinig for their fruitful contributions to our work.

References

1. Brooke, J.: SUS-A Quick and Dirty Usability Scale. Usability Evaluation in Industry (1996) 189–194
2. Finstad, K.: The System Usability Scale and Non-Native English Speakers. Journal of Usability Studies **1**(4) (2006) 185–188
3. Gogolla, M., Büttner, F., Richters, M.: USE: A UML-Based Specification Environment for Validating UML and OCL. Science of Computer Programming **69** (2007) 27–34
4. Gogolla, M., Hilken, F., Doan, K.H.: Achieving Model Quality through Model Validation, Verification and Exploration. Journal on Computer Languages, Systems and Structures, Elsevier, NL (2017) Online 2017-12-02.
5. Gogolla, M., Hilken, F., Kästner, A.: Some Narrow and Broad Challenges in MDD. In Seidl, M., Zschaler, S., eds.: Software Technologies: Applications and Foundations, Cham, Springer International Publishing (2018) 172–177
6. GSD Lab at University of Waterloo, MODELS group at IT University of Copenhagen: Clafer - Lightweight Modeling Language. `http://www.clafer.org`
7. Kästner, A., Gogolla, M.: Additional Material: Towards Flexible Object and Class Modeling Tools. `http://www.db.informatik.uni-bremen.de/publications/intern/o2c-casestudy-addon.pdf` (2018)
8. Kästner, A., Gogolla, M., Selic, B.: From (Imperfect) Object Diagrams to (Imperfect) Class Diagrams. Accepted for publication: MODELS 2018 (2018)
9. López-Fernández, J.J., Cuadrado, J.S., Guerra, E., de Lara, J.: Example-driven meta-model development. Software & Systems Modeling **14**(4) (2015) 1323–1347
10. Salay, R., Chechik, M., Famelis, M., Gorzny, J.: A Methodology for Verifying Refinements of Partial Models. Journal of Object Technology **14**(3) (2015)
11. Sauro, J.: A practical guide to the system usability scale: Background, benchmarks & best practices. Measuring Usability LLC (2011)
12. Tullis, T.S., Stetson, J.N.: A comparison of questionnaires for assessing website usability. In: Usability professional association conference. Volume 1. (2004)