Improving Modeling with Layered UML Diagrams

Harald Störrle
Department of Informatics and Mathematical Modeling, Technical University of Denmark, Richard Petersens Plads, 2800 Lyngby, Denmark

Keywords: Model based Development, Model Structuring, Diagram Presentation.

Abstract: Layered diagrams are diagrams whose elements are organized into sets of layers. Layered diagrams are routinely used in many branches of engineering, except Software Engineering. In this paper, we propose to add layered diagrams to UML modeling tools, and elaborate the concept by exploring usage scenarios. We validate the concept by implementation, lab assessments, and field testing. We conclude that layers enhance and complement conventional diagrams and model structuring techniques, are easy to add to existing modeling infrastructure, and are easy to apply by modelers.

1 INTRODUCTION

In a previous survey (Störrle, 2007), we have found that models in industry can grow to substantial sizes. We have also found that there are some very large diagrams, that is, diagrams representing hundreds or thousands of model elements. Often, these are also very important diagrams in the sense that they are widely used and play a central role in the organizations that created them; frequently, posters created from these diagrams can be found on many office walls. For such big diagrams, organizing and structuring the underlying models and their presentation as diagrams becomes a major challenge (Störrle, 2010). Today, we have only few practical and widely available methods to structure diagrams, and they have significant shortcomings. As a remedy, we propose to add layering features to UML modeling tools, emulating graphic design tools like Adobe Illustrator or Photoshop.

A *layer* groups a set of *presentation elements* to express some commonality of the model elements visualized by the corresponding presentation elements. As a shorthand, we will often refer to the model elements of a layer, meaning the model elements presented by the presentation elements placed on a layer. A *layered diagram* is a diagram with an ordered set of layers. There are contexts where "model" and "diagram" are synonymous, but this is not the case for most UML modeling tools. Layers as we discuss them in this paper have no impact to the underlying models, they are only used to structure *diagrams*.

The placement of elements on layers is not part of the model as such. Thus, the constraints or relationships between the model elements as implied by the modeling language (e.g., the UML meta model) do not constrain the placement of corresponding presentation elements on layers. Thus, a layer does not have to be a correct or complete UML diagram by itself—only the combination of all layers together has to satisfy the same language-induced constraints as a conventional diagram. Fig. 1 contrasts the concepts used in layered diagrams with those used in conventional diagrams. Obviously, the conceptual differences are small, which helps with the implementation of layered diagrams.

Each set of layers may be shown or hidden independently, so that n layers allow 2^n combinations of visible layers. Also, layers may be selectively locked so that they stay visible, but are not affected by manipulations. Finally, layers may be used to select the input to semantic model operations such as union, intersection, or difference by simply computing the set of model elements presented by a set of layers. Elements may also be placed on layers dynamically by filters, such that there are layers to collect model elements based on their type, the presence of an error or annotation, their time of creation, and so on.

2 RELATED WORK

Today, very few UML modeling tools offer layering; the only two exceptions we know of are Visu-

Störrle H..
Improving Modeling with Layered UML Diagrams.
DOI: 10.5220/0004337302060209
In *Proceedings of the 1st International Conference on Model-Driven Engineering and Software Development* (MODELSWARD-2013), pages 206-209
ISBN: 978-989-8565-42-6

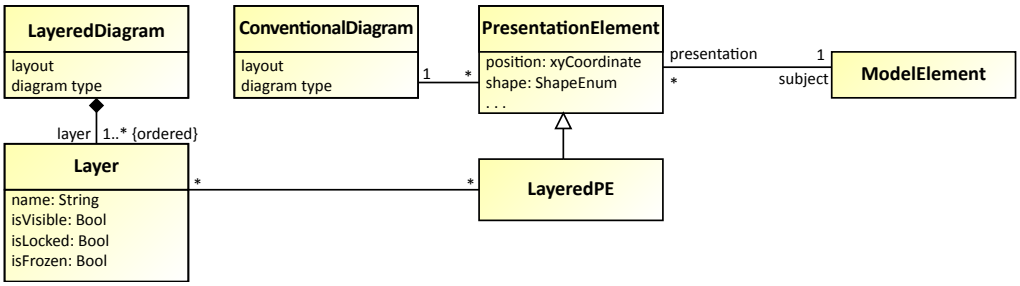

Figure 1: Comparing conventional diagrams with layered diagrams from a conceptual perspective.

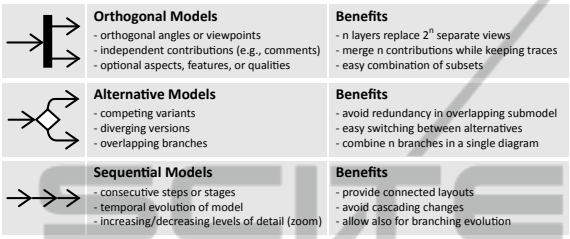

Figure 2: Some scenarios where using layers can improve UML modeling practice.

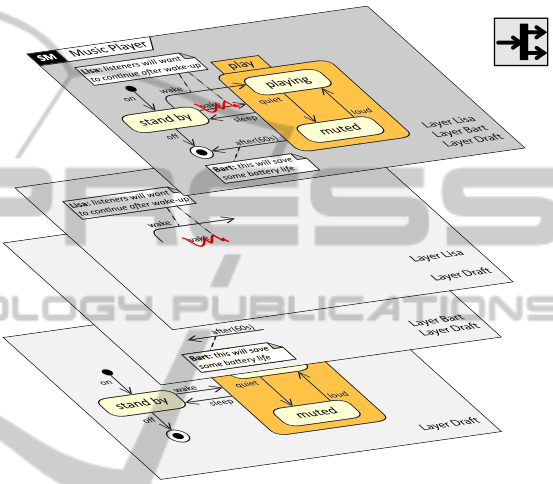

Figure 3: Layers for orthogonal aspects.

alParadigm (www.visual-paradigm.com) and Trinity (Peltonen et al., 2010). Without layers, modelers could use multiple independent diagrams for each aspect. Clearly, the number of views may grow exponentially with the number of independent aspects – not so with layers. Alternatively, modelers might use annotations or color coding to deal with orthogonal, alternative, or sequential views. Again, this works for small numbers of views, but is limited by the built-in limitations of the human visual apparatus. Using animated and 3d-models (see (Gogolla et al., 1999; McIntosh, 2009)) is limited by the challenging nature of navigating even small 3d models.

3 LAYER USAGE SCENARIOS

We describe three scenarios to highlight the shortcomings of existing model structuring mechanisms and point out how layers can help address the issue; Fig. 2 provides an overview. All three scenarios are straightforward and common in large scale domain modeling.

Orthogonal Models. Consider the case of model dimensions, i.e., models that capture orthogonal angles, points of view, or dimensions, such as independent product aspects, features, or qualities. Consider a model under review by several people. Assume that the reviewers document their remarks as model annotations "in place". At some point, the remarks are joined together yielding a common model with all review remarks in it. Simply merging different models has several disadvantages: (1) the resulting diagrams can become quite crowded and confusing when there are many comments on one issue; (2) extra effort is required to ensure that every comment can be traced to the reviewer responsible for it; (3) the model might become inconsistent and difficult to understand when reviewers propose competing solutions to an issue.

When using layers, however, each review may use an individual layer (or set of layers) for his or her contributions. This way, we may selectively switch between different reviewers' viewpoints, but still can see several reviewers' comments simultaneously, if required. Also, it is always clear who the author of a comments is, by simply looking at the layer on which the contribution resides. Fig. 3 shows a small example of using layers in this scenario.

Alternative Models. Consider the case of modeling alternatives, i.e., models that capture competing solutions to a given problem, diverging variants of designs, or branching versions. For instance, a model may document variants of business process im-

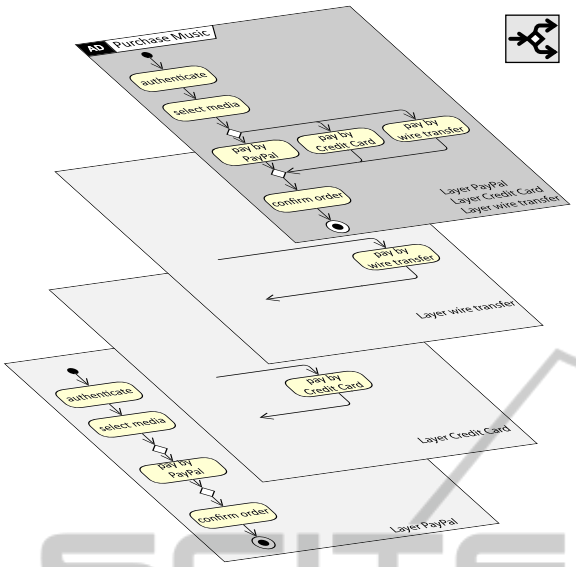

Figure 4: Layers for alternatives.

plementations deployed to different customers (see Fig. 4). Assume that the alternatives have commonalities as well as differences.

Of course one could simply join all alternatives into a single model, but this will make it difficult to distinguish variants. Also, only those parts where all variants overlap (i.e., the intersection of all alternatives) may be factored out, so that model parts may have to be stored redundantly, which may lead to inconsistencies and model clones. When using layers, however, each alternative may use a separate layer, see Fig. 4. Discarding an alternative amounts to discarding a layer with all the model elements it refers to. There is also no need to store overlapping model fragments redundantly, since the modeler may choose to show all layers that contribute elements to an alternative.

Sequential Models. Consider the case of model sequences, i.e., models consisting of subsequent stages to document an evolution of refinement or elaboration, such as the visualization of a projects's release schedule. Now, assume that a sequence of use case diagrams is used to represent a series of increments planned to be produced over time (see Fig. 5 for an example). Each increment contains a set of actors and use cases that adds to the previous elements.

Showing all increments simultaneously in one diagram will make it difficult to see the progress over time and the individual contribution per increment. Using an individual diagram for each stage results in a large number of diagrams that are hard to maintain: changes to the i-th diagram of a sequence of n dia-

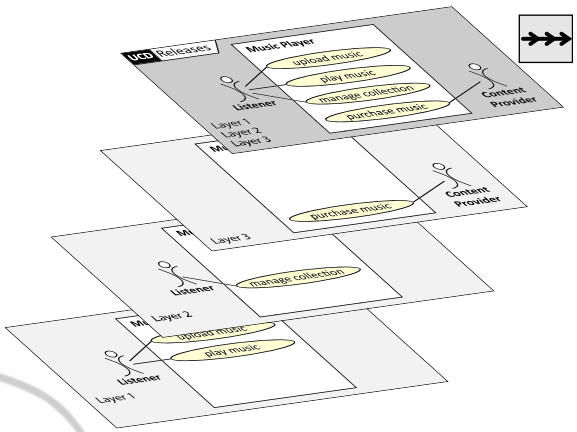

Figure 5: Layers for sequential stages.

grams will have to be reflected in all diagrams from $i+1$ to n, or the model's audience will be confused about changing layouts. Using layers, on the other hand, each increment can be placed on its own layer. Thus, they can easily be shown in isolation, or as part of a set. Changing the elements of the i-th layer is immediately visible in all layers above i.

4 VALIDATION

We validated our proposal in three ways. First, we implemented it to explore any inherent hidden technical difficulties, which proved to be more or less straightforward.

Second, we studied the outcome in controlled environments with expert modelers from academia and industry, and interviewed the participants and logging problems and shortcomings to be amended. We asked graduate students to model the GUI of a calendar application using AIDE (without layering). The resulting UML StateMachine consisted of 178 States (nested up to 9 levels deep) and 218 Transitions, presented in a single diagram. The facilities for navigating large diagrams that were available at this point (zooming and locator-maps) were clearly inadequate for diagrams of this size. Next, we added layering support to AIDE, and asked students to restructure these models using layers. We logged problems and shortcomings, resulting in a list of bugs, none of which indicated conceptual problems with layers. This case study clearly showed that layers can be used in modeling StateMachines with great benefit. In particular, navigating a large diagram and focusing on specific parts or aspects becomes very easy. Structuring the model presentation by placing model elements on up to 20 layers greatly helped, and handling so

many layers posed no significant problem. Defining the "right" layers, however, requires a certain amount of thought. We speculate that the exact nature of layers is specific to the application domain and model purpose. We also presented the improved AIDE to 22 expert modelers from industry and academia and collected their opinions in order to establish the validity of our earlier findings in a larger and more diverse populations. The feedback was unanimously positive: all experts immediately saw the potential benefit of layers in their respective fields and could well imagine using such a feature in previous and current projects. Some of them readily proposed more examples of usage scenarios.

Third, we field tested our approach by using it in the classroom in a UML modeling course. No issues related to layering appeared. Some student teams used layering in unexpected ways, e.g., to separate different diagram regions for splitting and merging work. Students also exploited the layers in their presentations. Note that the students had only been informed of the existence of layers, without any form of training in their usage; the students invented these usage scenarios ad hoc. We conclude that, on the one hand, that the adoption of layers might be supported by creating a stack of appropriately named default layers. On the other hand, more focus ought to be put on supporting collaboration through layers.

5 CONCLUSIONS

In this paper we propose layers to structure large diagrams. We illustrate the benefits of layers using three realistic and common usage scenarios. We validate our proposal by implementation, case study analysis, and field test. We find that layers complement conventional diagrams by providing *dependent* views, which may drastically reduce the number of diagrams in some situations. In contrast to existing approaches like 3d and animated UML diagrams, layered diagrams offer three benefits: (1) the concepts are commonly used in many fields and easy to understand and apply; (2) there are many useful application scenarios for layered diagrams; (3) layers are easy to implement in modeling tools.

While we have already explored many application scenarios, there seem to be even more (e.g., version control, group collaboration, or pseudo-animation). More extensive qualitative research is needed to better understand the extent and limits of layers in UML modeling. Also, a more thorough investigation into the benefit of layers is clearly needed.

REFERENCES

Gogolla, M., Radfelder, O., and Richters, M. (1999). Towards three-dimensional representation and animation of UML diagrams. In *Proc. 2nd Intl. Conf. Unified Modeling Language*, pages 489–502. Springer Verlag.

Gorton, I., Cuesta, C. E., and Babar, M. A., editors (2010). *Proc. 4th Eur. Conf. Sw. Architecture (ECSA'10): Companion Volume*. ACM.

McIntosh, P. M. (2009). *X3D-UML: User-Centred Design, Implementation and Evaluation of 3D UML Using X3D*. PhD thesis, RMIT University.

Peltonen, J., Felin, M., and Vartiala, M. (2010). From a Freeform Graphics Tool to a Repository Based Modeling Tool. In (Gorton et al., 2010), pages 277–284.

Störrle, H. (2007). Large Scale Modeling Efforts: A Survey on Challenges and Best Practices. In Hasselbring, W., editor, *Proc. IASTED Intl. Conf. Software Engineering*, pages 382–389. Acta Press.

Störrle, H. (2010). Structuring very large domain models: experiences from industrial MDSD projects. In (Gorton et al., 2010), pages 49–54.