Tool Support for the Navigation in Graphical Models

Tobias Reinhard, Silvio Meier, Reinhard Stoiber, Christina Cramer, Martin Glinz
Department of Informatics
University of Zurich, Switzerland
{reinhard | smeier | stoiber | cramer | glinz} @ifi.uzh.ch

ABSTRACT

Graphical models are omnipresent in the software engineering field, but most current graphical modeling languages do not scale with the increasing size and complexity of today's systems. The navigation in the diagrams becomes a major problem especially if different aspects of the system are scattered over multiple, only loosely coupled diagrams.

In this paper we present the hierarchical navigation capabilities of the ADORA modeling tool. The user of this tool can freely control the level of detail in different parts of the model to reduce the size and complexity of the diagrams being displayed. Our fisheye visualization technique makes it possible to integrate all modeling aspects (structure, data, behavior, etc.) in one coherent model while keeping the size and complexity of the diagrams within reasonable limits.

Categories and Subject Descriptors: D.2.2 [Software Engineering]: Design Tools and Techniques; H.5.2 [Information Interfaces and Presentation]: Interaction styles

General Terms: Algorithms, Languages

Keywords: fisheye view, focus+context, graphical model, graphical user interface, hierarchical network, information visualization, navigation

1. INTRODUCTION

The graphical models that are used to describe current software systems grow in size and complexity with the systems they represent. Given the size and resolution of current display screens, only parts of large models can be shown at a time. Scaling down the whole model makes it possible to see the whole model at the expense of losing the details: the model elements, in particular the labels, become unreadable. The fact that a model diagram doesn't fit on the screen complicates the task of finding specific nodes or links because the user has to scale or pan frequently to see the elements that lie outside the currently visible area [6, 8]. The navigation problem is aggravated by the fact that navigation in graphical model is inherently difficult: in contrast to the straight sequential reading style of text, there is no common reading style for diagrams. So the reader has to develop his/her own inspection strategies [10].

ICSE'08, May 10–18, 2008, Leipzig, Germany.
ACM 978-1-60558-079-1/08/05.

One way to cope with the size and complexity is using hierarchical and aspectual decompositions of the model into manageable and understandable parts (in a way that follows the basic software engineering principles of information hiding and separation of concerns). Any such decomposition technique, whether based on the hierarchy or different aspects, can fully play to its strength only if it is integrated into the visualization and navigation capabilities of a tool.

However, most tools still rely on flat or practically flat models (i.e. they show all elements in one view) and support only panning and scaling for navigation. Tools that support navigation in hierarchical models visualize only one level of the hierarchy at a time (i.e. the composites and their parts are shown as separate diagrams). This "explosive zooming" [2] leads to frequent "context switches" by the user because each zoom step results in a completely new view.

Given this state of tools, it is not surprising that most modeling languages, including UML [9], rely on the principle of loosely coupled multi-diagram models as the primary means for separating concerns and decomposing large models into manageable parts. Only recently, hierarchical features have been added to UML in version 2.0. The main problem with this segregation is that it puts the intellectually demanding burden of integrating the different diagrams in one coherent model entirely on the user's shoulders [3]. The reader of the model has to switch back and forth between different diagrams which makes the already hard navigation task even harder. These switches occur frequently because the analysis and design process involves a constant interplay between different aspects of a system.

The basic idea of our approach is to solve this problem by reversing the underlying principles: we use an *integrated*, inherently hierarchical model instead of a loose collection of diagrams and a tool that *generates* abstractions and diagrams of manageable complexity by exploiting the hierarchy and filtering model elements. We have developed ADORA[1], which is comprised of (i) an integrated hierarchical *modeling language* [7] with hierarchical decomposition and views (structure, behavior, user interaction, etc.) and (ii) a *tool* that allows a user to navigate through the hierarchy and show or hide model elements according to the selected view(s).

In this paper, we present the current state of the ADORA tool. While we have previously published concepts for this tool [2, 14], we now can present the first full implementation, including new features such as a new algorithm for hierarchical zooming [11] and smart line routing [12].

[1] ADORA is an acronym for **A**nalysis and **D**escription **O**f **R**equirements and **A**rchitecture

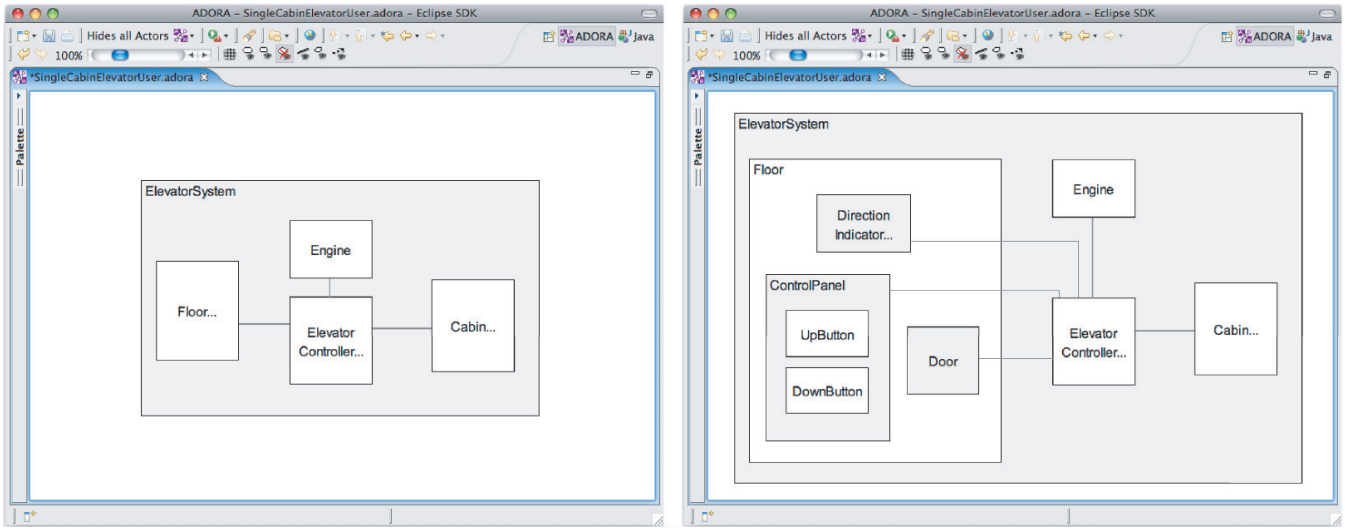

Figure 1: a. The screenshot on the left shows an abstract view of an elevator system. b. The situation after successively zooming-in *Floor* and then *ControlPanel* is shown in the right screenshot.

2. FISHEYE VIEWS

Our visualization concept is based on fisheye views [6] which show local detail and global context together in a single view and therefore release the user from the task of mentally integrating detail and context.

The use of fisheye views for the visualization of graphical models has two advantages: Showing only an abstract view of the model parts which are not in the user's current focus of interest reduces the amount and complexity of the information displayed while still showing all the details in the current focus of interest. On the other hand, the integration of a fisheye visualization in a modeling tool avoids the frequent context switches that occur with an explosive zoom, because the surrounding context is always shown.

Fig. 1 shows the basic idea of our fisheye zoom approach: The figure on the left shows an abstract view of a hierarchical model of an elevator system: only the top level components *Floor*, *Engine*, *ElevatorController* and *Cabin* are visible. The ellipsis after the name of a component indicates that the component has an inner structure that is currently hidden. By successively **zooming-in** the components *Floor* and then *ControlPanel*, we get the figure on the right, which shows all the details in the focal point (the *ControlPanel* component) together with the global structure of the model. Conversely, by **zooming-out** the components we get a more abstract view of the system.

Switching between abstract and detailed representations of a node also has implications on the links between the nodes. The three links between *ElevatorController* and one of the internal components of *Floor* in Fig. 1b are represented by a single abstract link, visualized as a bold line, between *ElevatorController* and *Floor* in the more abstract view shown in Fig. 1a. Thus, the user can easily construct an abstract overview of every component's parts and their relationships.

The concept of abstract links and the layout changes that occur with every zoom operation result in the need to adjust the links accordingly. We have previously developed a line routing algorithm [12] that automatically re-routes the lines in the diagram after a layout change.

2.1 Fisheye Zoom Algorithm

The layout of a diagram has to be adapted accordingly if a node is zoomed-in or zoomed-out. This is the task of the zoom algorithm which rearranges the layout by moving and resizing nodes. The biggest challenge for the zoom algorithm lies in adjusting the diagram layout but preserving the original layout as far as possible: after zooming, the user should still recognize the diagram. This is important because the user builds a mental map [8] for the navigation in the model. Additionally, a lot of the value of a graphical model is encoded in the so called secondary notation [10] (i.e., the layout and other perceptual clues like adjacency of nodes, clustering, and the use of white space) that should be preserved by the zoom algorithm.

From the literature on fisheye visualization and especially from previous experience with the ADORA tool [2] we have learned that the stability of the layout is of special importance: A sequence of zoom operations followed by a sequence of inverse zoom operations should result in the original layout. Additionally, the zoom algorithm should have the following properties: no overlapping of nodes, having multiple points of interest zoomed-in at the same time, permitting model editing, and running in real-time.

We have developed a zoom algorithm that builds upon the basic ideas of the Continuous Zoom approach [1]: we construct an interval structure for the diagram by projecting the node boundaries on the X- and Y-axes and apply interval scaling to that structure. Fig. 2 shows the visualization of the interval structure in our ADORA tool.

The length of the intervals that are projections of a zoomed node are scaled up or down to adjust the whole structure and subsequently the positions of the nodes. Zooming-in a node increases the size of the intervals and therefore moves the other nodes further away, while zooming-out decreases the size of the intervals and moves the remaining nodes closer together. Our zoom algorithm guarantees the stability of the layout in all situations (i.e., the layout of a specific view of the hierarchy is always exactly the same irrespective of the sequence of zoom operations that led to it). A detailed description of our zoom algorithm can be found in [11].

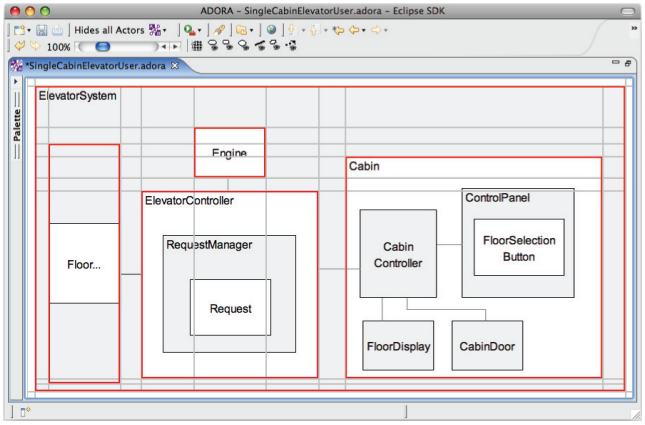

Figure 2: The interval structure after *Floor* has been zoomed out

2.2 Layout Adjustments

A major advantage of our algorithm is that it can not only be used for the fisheye zooming operations, but also for adjustments of the layout in general. The capability of the zoom algorithm to adjust the layout of a diagram is useful in the following situations [11]:

1. **Navigation in the hierarchy:** The zoom algorithm can, as just described, be used to browse the hierarchy by zooming-in nodes to see the details and zooming-out to get a more abstract view.

2. **Smart editing:** Editing graphical models is often a tedious task. Adding and deleting nodes requires a lot of manual work: existing nodes must be moved to provide the required space or to remove the empty space resulting from deletions [14]. Our zoom algorithm can be used to automatically expand a diagram when space is required for the insertion of a node and to contract the layout when empty space becomes available.

   If there isn't enough empty space to insert a new node, the node is inserted with the maximum size it can occupy without moving any existing node. The zoom algorithm is then used to expand the node to its intended size, thus creating the required space automatically. Removing a node is done the other way round: the zoom algorithm is used to shrink the node to be removed to the size of a point.

3. **Filtering nodes:** Apart from zooming-out nodes, the presented algorithm offers the possibility to hide individual nodes to reduce the size and complexity of a diagram. Nodes are filtered by applying the technique to remove nodes described above, but without actually removing the nodes from the model.

3. TOWARDS A SINGLE INTEGRATED MODEL

The technique to hide individual nodes can be used to generate different views (that show model elements of certain types only) from a single, integrated model of a system [14], instead of forcing modelers to create views themselves by drawing a multitude of diagrams of different types, as required in UML, for example.

The combination of fisheye zoom with the view generation mechanism described above in the ADORA tool makes the use of a comprehensive integrated modeling language such as ADORA [7] feasible.

The basic modeling elements in the ADORA language are *abstract, prototypical objects* that are used instead of classes. These objects are recursively decomposed into other objects, thus leading to a *systematic hierarchical decomposition* in a straightforward, easily understandable way. The visual notation of nested shapes is used to represent compositions, which makes their semantics intuitively clear. An ADORA model integrates all modeling facets (structure, data, behavior and user interaction) in *a single, coherent model*.

Using the ADORA tool, the modeler can edit and/or generate diagrams of reasonable content and size by zooming and filtering: by zooming s/he chooses the desired foci and by filtering s/he creates the view(s) that s/he is currently interested in. Fig. 3 shows an example. The modeler has set two foci, *Floor* and *Cabin*, by successively zooming in these objects and their components. In the screenshot on the left side of Fig. 3, the modeler has selected three views in combination: structure (the rectangles), user interaction (the ovals) and external actors (the hexagon), thus showing how the actor *User* interacts with the elevator system. In the screenshot on the right side of the figure, the modeler has decided that s/he just wants a structural overview where user interaction is hidden. The modeler can easily generate such views in the tool by clicking buttons in the symbol bar at the top of the window that hide or show the model elements of the corresponding type(s). The size and the layout of the diagram are automatically adjusted when the modeler selects another view.

4. IMPLEMENTATION

The ADORA tool is implemented as a set of plugins for the Eclipse Framework [4] and uses the Graphical Editing Framework (GEF) [5]. The zoom algorithm has been implemented as a separate pure Java plugin and can therefore be used by any other Java based graphical modeling tool. The ADORA tool is available under an open source license from http://www.ifi.uzh.ch/rerg/research/projects/adora/.

5. RELATED WORK

Furnas implemented the Fisheye Lens [6] concept in computer graphics by the means of a semantic zoom which shows or hides a point in the structure depending on its "degree of interest". The original Fisheye Lens formulation has no explicit control over the layout and can therefore not be used directly for graphical models. The Graphical Fisheye Views by Sarkar and Brown [13] use geometric transformations to graphically distort the layout. Graphical Fisheye Views are not particularly suitable for graphical models because the distortion reduces the readability (especially of text labels).

The logical fisheye concepts SHriMP of Storey and Müller [15] and Continuous Zoom by Bartram et al. [1] rely on a fixed screen size and are therefore inherently global (i.e. the zooming of one node changes the size of all nodes in the model). However, the concept of scaling all nodes down so that the model always fits on the screen cannot be used for large graphical models because the scaled down nodes become too small. An earlier approach of a fisheye visualization technique in our research group [2] uses translation

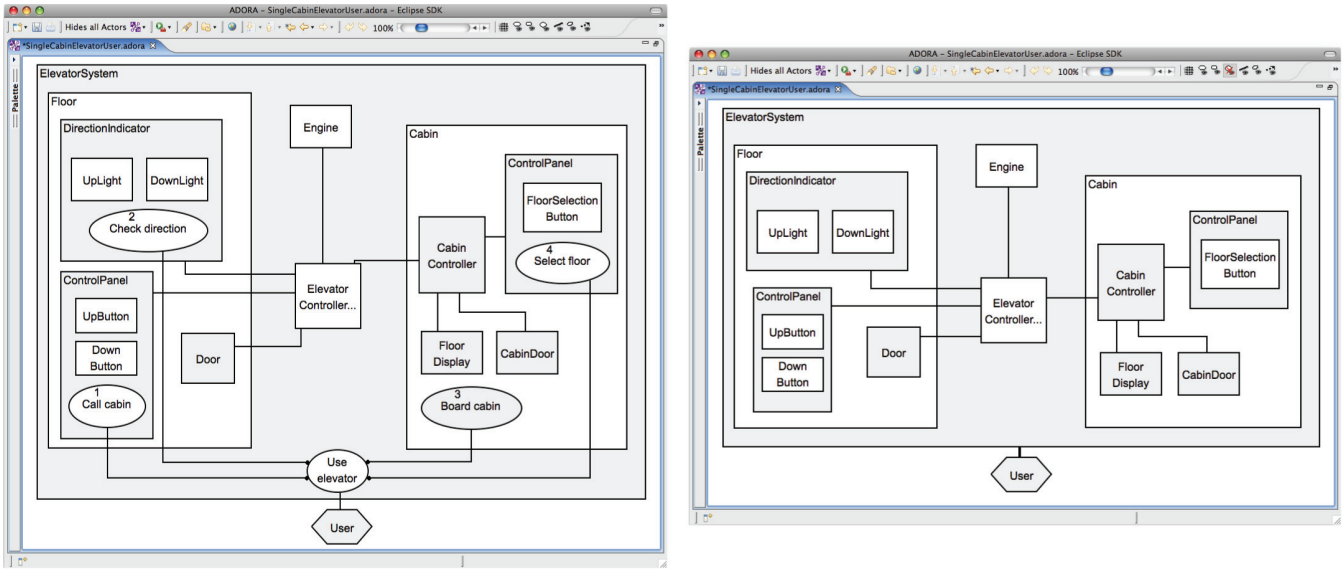

Figure 3: The left hand side shows the scenarios (ovals) embedded into the hierarchical structure. The right hand side shows the same model with this user view hidden.

vectors to move the nodes in a similar way as SHriMP does, but does not globally scale all nodes. However, this approach cannot guarantee the stability of the layout in all situations.

6. CONCLUSIONS

We have presented the capabilities of the ADORA tool for navigating and visualizing integrated hierarchical models. The user can use a fisheye zoom to navigate in the hierarchy and to reduce the complexity of the diagrams by hiding currently irrelevant details. A view generation mechanism that shows different aspects of the system in one diagram offers an additional dimension to reduce the complexity.

A problem that remains is the fact that even with our fisheye view concept the models frequently grow beyond the size of the screen. Therefore we provide additionally scroll bars and a separate operation for linear view scaling when a model grows beyond the size of the available display area.

We have shown our concepts in the context of the ADORA language because the filtering mechanism plays to its strength with an integrated model only. In principle, however, the presented concepts can be used for any graphical modeling language that supports a hierarchical decomposition, e.g. hierarchical UML diagrams such as component diagrams.

7. REFERENCES

[1] L. Bartram, A. Ho, J. Dill, and F. Henigman. The Continuous Zoom: A Constrained Fisheye Technique for Viewing and Navigating Large Information Spaces. In *UIST '95: Proceedings of the 8th Annual ACM Symposium on User Interface and Software Technology*, pages 207–215, 1995.

[2] S. Berner, S. Joos, M. Glinz, and M. Arnold. A Visualization Concept for Hierarchical Object Models. In *Proceedings of the 13th IEEE International Conference on Automated Software Engineering (ASE'98)*, pages 225–228, 1998.

[3] D. Dori. Why Significant UML Change Is Unlikely. *Comm. of the ACM*, 45(11):82–85, November 2002.

[4] Eclipse. http://www.eclipse.org.

[5] Eclipse Graphical Editing Framework (GEF). http://www.eclipse.org/gef.

[6] G. W. Furnas. Generalized Fisheye Views. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 16–23, 1986.

[7] M. Glinz, S. Berner, and S. Joos. Object-oriented modeling with ADORA. *Information Systems*, 27(6):425–444, 2002.

[8] K. Misue, P. Eades, W. Lai, and K. Sugiyama. Layout Adjustment and the Mental Map. *Journal of Visual Languages and Computing*, 6(2):183–210, 1995.

[9] OMG. Unified Modeling Language: Superstructure, version 2.1.1. OMG document formal/2007-02-05, 2007.

[10] M. Petre. Why Looking Isn't Always Seeing: Readership Skills and Graphical Programming. *Comm. of the ACM*, 38(6):33–44, June 1995.

[11] T. Reinhard, S. Meier, and M. Glinz. An Improved Fisheye Zoom Algorithm for Visualizing and Editing Hierarchical Models. In *Proceedings of the Second International Workshop on Requirements Engineering Visualization (REV'07)*, 2007.

[12] T. Reinhard, C. Seybold, S. Meier, M. Glinz, and N. Merlo-Schett. Human-Friendly Line Routing for Hierarchical Diagrams. In *Proceedings of the 21st IEEE International Conference on Automated Software Engineering (ASE'06)*, pages 273–276, 2006.

[13] M. Sarkar and M. H. Brown. Graphical Fisheye Views. *Comm. of the ACM*, 37(2):73–83, December 1994.

[14] C. Seybold, M. Glinz, S. Meier, and N. Merlo-Schett. An Effective Layout Adaptation Technique for a Graphical Modeling Tool. In *Proceedings of the 25th International Conference on Software Engineering (ICSE'03)*, pages 826 – 827, 2003.

[15] M.-A. D. Storey and H. A. Müller. Graph Layout Adjustment Strategies. In *GD '95: Proceedings of the Symposium on Graph Drawing*, pages 487–499, 1996.