Integrating a Zoomable User Interfaces Concept into a Visual Language Meta-tool Environment

Na Liu[1], John Hosking[1] and John Grundy[1, 2]
Department of Computer Science[1] and Department of Electrical and Computer Engineering[2], University of Auckland, Private Bag 92019, Auckland, New Zealand
{karen, john, john-g}@cs.auckland.ac.nz

1. Introduction

From our experiences developing and evaluating a wide range of visual language environments, and those of others, we have found that visual languages often have a major problem with lack of screen real estate [7, 3, 10]. Multiple views are often used to solve the problem by allowing large systems to be modelled by breaking them into smaller parts [3, 5]. However this approach has limitations and thus we have begun to explore a complementary approach - "Zoomable User Interfaces", or ZUIs. We have added some prototype ZUI facilities to Pounamu, a meta-tool we have been developing for building multiple view visual language environments [3]. We motivate the need for ZUIs in visual language tools, illustrate our ZUI extensions to Pounamu, and discuss our experiences to date in building and using these facilities.

2. Motivation

Pounamu is a Meta-CASE tool that is used to specify and realise a wide range of domain-oriented, visual language environments. Pounamu provides tools enabling users to specify a meta-model that defines the constructs (information structures and associated semantics) of a visual programming language, and graphical notations that are used to visually represent the syntax of the language in multiple views. The specified visual language tool can be used for modelling during and after its specification with highly dynamic tool specification and target tool updates.

Pounamu provides multi-view support both for its own meta-tool facilities and for the target visual language modelling views defined. Consistency is supported between views of the same and different types via the shared information repository layer. Figure 1 shows a Pounamu-defined Unified Modelling Language (UML) diagramming tool in use, with two UML views shown.

Very often visual modelling of systems with such design views produces large, complex views that can become very hard to understand or navigate [3, 5, 7]. Splitting the visual design models into different views is important to manage the complexity of the visualisations. However, users have to repeatedly context switch when moving between multiple views in order to get an overall idea of contextually associated modelling elements [6, 1]. Also, our users of Pounamu modelling views have found that they often want to zoom in on parts of a view or zoom out to get a "bigger picture", as well as have the tool handle multiple view consistency.

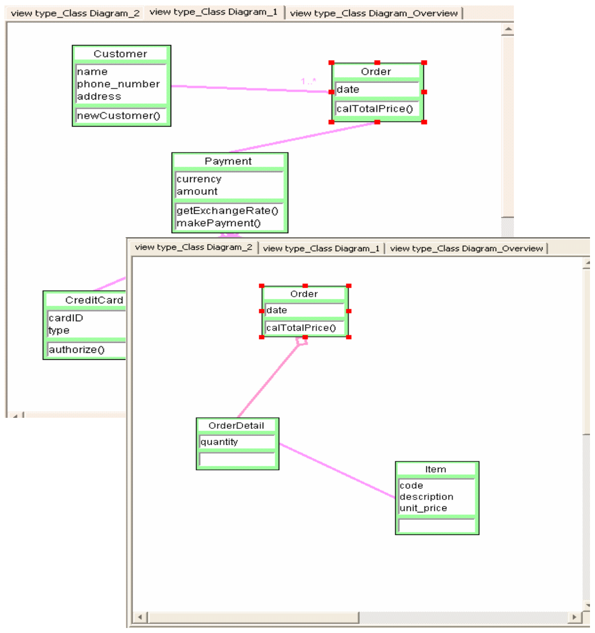

Figure 1. Examples of multiple UML design views.

A number of researchers have begun to experiment with zooming capabilities integrated into tools, both visual language and form-based. Bederson el at [1] made use of scene graphs for implementing two-dimensional graphical applications [1]. They also used similar approaches for complex map and tree visualisation [4, 7]. Mugridge et al [6] applied zoomable user interfaces to the Naked Objects framework for form-based interface construction. Other applications have included zoomable web browsers [2] and form-based user interfaces [8]. In these applications Zoomable User Interface facilitates provide a virtually unlimited screen space for views. They provide support for zooming in and out of views, splitting views into focal points and allowing users to construct new focus sub-groups interactively. Toolkits to realise these zoomable user interfaces include Jazz [2], and ZVTM [10]. All of these applications are bespoke i.e. ZUIs have been used for domain-specific applications.

We wanted to experiment with the suitability of ZUIs in a visual meta-tool environment to enhance large visual program diagram management and navigation. We added to Pounamu the following ZUIs: a Radar view (whole diagram zooming facilities), Zoomable view (part diagram and individual element zooming), Split view (Zoomable selection element zooming) and Focal view. All of these enhanced views support zooming-in and zooming-out of the Pounamu modelling elements in zoomable panels, in addition with the consistent editing among the modelling elements in different views. The ZUIs support different levels of design overview with fewer details. They supplement to the existing multi-view User Interfaces in Pounamu that support modelling a whole system in parts allowing easier control over each smaller view.

3. A Zoomable User Interfaces Example

Consider a user wanting to model a complex UML design with several UML views, with some views getting very large and complex. Taking a prototype "Customer-Order" web system design example, two UML class diagram views are modelled using the Pounamu specified UML tool and are shown in Figure 1. The two diagrams are displayed in two separate Pounamu views with the modelling element instance "Order" referencing the same entity in both views. If the views have more elements added, they become harder to display and navigate.

Our prototype Radar views and Zoomable views support overviews of a conventional Pounamu view with different scaling factors. The Split and Focal views form so-called "bi-focal" display views that display a user's dynamic selections of scenes from the Radar and Zoomable views respectively. This combination of zoomable views allows the user to select any set of elements in the Radar view and the Zoomable view, and the corresponding selected elements are displayed in the Split view and Focal view respectively. When shapes are selected in the Radar view or Zoomable view, the ZUI system checks whether there exists proper relationships between the selected shapes. If relationships exist, they are consistently added into the Split or Focal view together with the selected shapes.

The ZUI views support:

- Zooming-in and zooming-out of displayed objects
- Selection of a single or a particular group of objects
- Relocating single objects into ZUI views
- Panning objects into any position within the zoomable canvas that they belong to
- Scaling of an object into a bigger or smaller size
- Consistent editing of objects between ZUI views and the existing multi-views

The two separate class diagrams from different views from Figure 1 have been combined into an overall view and displayed as zoomable views in Figure 2 (1). The Radar views and Zoomable views have been populated automatically in these prototype ZUIs from the shared meta-model instances. The positions of the objects in the Radar view have been rearranged by the user in the zoomable panel shown in Figure 2 (2) for a better layout of the overview (no overlapping, no distortion of shapes). The Zoomable view is synchronized with the objects' location changes, its shapes repositioned if moved in the Radar view and vice-versa. Apart from zooming-in and zooming-out, Radar views and Zoomable views also support dynamic selections of part of the display. In Figure 2 (2) the user has selected three elements in the Radar view, and the selected elements are displayed in the Split view. The elements in the Split view can be zoomed-in and zoomed-out, panned and scaled.

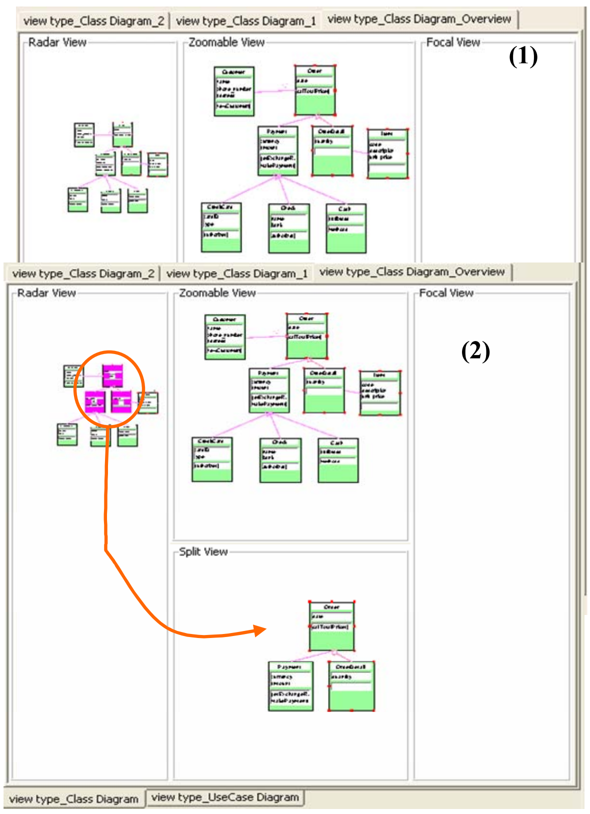
view type_Class Diagram_2
view type_Class Diagram_1
view type_Class Diagram_Overview
Radar View
Zoomable View
Focal View
(1)
(2)
Split View
view type_Class Diagram
view type_UseCase Diagram

Figure 2. (1) Radar and Zoomable views and (2) Split view from Radar selections.

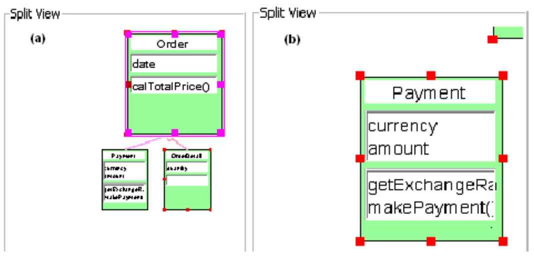

Figure 3. Scaling of selected view shapes.

Views support zooming of the whole view content, as shown by the smaller diagram in the Radar view, which can be zoomed in or out using the mouse. Views also support scaling of selected items. For example, in Figure 3 (a) the user has selected a UML diagram element "Order" and scaled this up in the Split view to provide them a clearer and bigger view. Figure 3 (b) shows the Payment object zoomed-in in the Split view. This allows the user to focus in on selected view content but maintains the context of the zoomed information.

Consistency between ZUI views and conventional non-zoomable Pounamu views is supported. Editing consistency between a selected object is managed among all the ZUI views and multi-views. When any view object is modified in a view or standard Pounamu property sheet, the changes are synchronized among all the other views, property sheets and its tree node representation.

Although the examples we have used illustrate a UML diagramming tool, we stress that our ZUI views can be used by any tool generated by our Pounamu meta tool.

4. Discussion

The ZUIs we have added to Pounamu supplement its existing multi-view interface. They do not affect any existing features and behave like add-ins with an enhanced display capability. The users can switch between the ZUIs and the existing multi-views in an uninhibited manner, and may choose not to use the ZUIs at all as they have been designed as a plug-in..

Pounamu is implemented using conventional Swing GUI components for views, property sheets and view components. We used the Jazz [3] framework to add our new ZUI capability to Pounamu views. Jazz uses the Java2D rendering engine. It supports the rendering of lightweight Swing components in a scene graph and provides panning, zooming, selecting, scaling and rotating functions on Swing widgets through virtual cameras. The Pounamau ZUI views have been implemented via specialised Pounamu modeller view classes at back-end and a set of change event listeners at front-end to manage consistency and integration. These ZUI components and event handlers are plugged into the Pounamu "ModellerPanel" component and initialised.

Apart from various UML diagram types and models, we have experimented with ZUIs applied to several complex Web services composition models. Since web services models can become very large and complex, they provided a good test case for the ZUIs implementation. Users found it relatively easy to navigate around and use the zoomable views in Pounamu. Our experiences to date indicate that having ZUIs in Pounamu for high-level design overviews and complex view navigation is a very promising extension to the target visual language tools.

Our current prototype has problems with nested Swing component zooming due to limitations of the Jazz framework we used. Repainting of ZUIs can be a problem and some errors are caused when mouse actions are passed to the inner-most Swing widgets. Jazz applications are also very memory intensive, which we found placed an unreasonable limit on the number of elements that could be added to our zoomable views. We are planning more formal user evaluations on a ZUI-enabled Pounamu application to further validate our ZUI concepts.

5. Summary

We have introduced Zoomable User Interfaces into Pounamu, a visual language meta-tool, providing enhanced view navigation and management features. The Zoomable User Interfaces provide the user with support for flexible and dynamic design overviews and view element focus, and help address problems of lack of screen space and display capability in complex views.

6. References

1. Bederson, B., Meyer, J. and L. Good. Jazz: An Extensible Zoomable User Interface Graphics Toolkit in Java, in *Proceedings of 2000 ACM Conference on User Interface and Software Technology*, ACM Press, 171-180.
2. Bederson, B. and Meyer, J. Implementing a Zooming User Interface: Experience Building Pad++, *Software - Practice and Experience*, vol 28, no. 10, August 1998, 1101-1135.
3. Grundy, J.C., Hosking, J.G. and Mugridge, W.B., Inconsistency management for multiple view software development environments, *IEEE Transactions on Software Engineering*, Vol. 24, No. 11, November 1998, 960-981.
4. Hornbaek, K., Bederson, B. and Plaisant, C. Navigation patterns and usability of zoomable user interfaces with and without an overview, *ACM Transactions on Computer-Human Interaction*, vol. 9, no. 4, December 2002, 362-389.
5. Meyers, S., Difficulties in Integrating Multiview Editing Environments, IEEE Software, vol. 8, no. 1, 49-57, Jan 1991.
6. Mugridge,W.B. Nataraj, M. and Singh, D. Emerging User Interfaces through First-Class Viewers, In *Proceedings of the 2003 New Zealand Conference on Computer-Human Interaction*, Dunedin, New Zealand, July 3-4 2003.
7. Plaisant, C., Grosjean, J., and Bederson, B.B., SpaceTree: Supporting Exploration in Large Node Link Tree, Design Evolution and Empirical Evaluation, In *Proceedings of the 2002 IEEE Symposium on Information Visualization*, Boston, October 2002, pp. 57 –64.
8. Pook, S., Lecolinet, E., Vaysseix, G. and Barillot, E.. Context and interaction in zoomable user interfaces. In *Proceedings of the 2000 ACM Conference on Advanced Visual Interfaces*, Palermo, Italy, May 2000, ACM Press, pp. 227-231.
9. Zhu, N, Grundy, J, Hosking, J, Pounamu: a meta-tool for multi-view visual language environment construction, in *Proc IEEE VL/HCC'04*, Rome, Italy, September 2004.
10. ZVTM, *Zoomable Visual Transformation Machine*, http://zvtm.sourceforge.net/.