Generating Eclipse Editor Plug-Ins Using Tiger

Enrico Biermann[1], Karsten Ehrig[2], Claudia Ermel[1], and Gabriele Taentzer[3]

[1] Institut für Softwaretechnik und Theoretische Informatik, TU Berlin, Germany
[2] Department of Computer Science, University of Leicester, UK
[3] Fachbereich Mathematik und Informatik, Universität Marburg, Germany
tigerprj@cs.tu-berlin.de
http://tfs.cs.tu-berlin.de/tigerprj

Abstract. We present TIGER, a visual environment to design visual language (VL) specifications based on meta models, graph grammars and layout definitions. A VL specification serves as basis to generate a visual editor for VL diagrams as ECLIPSE plug-in.

Introduction. Domain specific modeling languages are of growing importance for software and system development. Meta tools are needed to support the rapid development of domain-specific visual editors. A visual language (VL) definition based on a meta model in combination with a rule-based specification of editor commands is used in TIGER *(Transformation-based Generation of Environments)* to generate a corresponding visual editor.

TIGER combines the advantages of precise VL specification techniques using graph transformation concepts with sophisticated graphical editor development features offered by the Eclipse Graphical Editing Framework (GEF) [1]. Using graph transformation at the abstract syntax level, an editor command is modeled in a rule-based way. The application of such syntax rules to the underlying syntax graph of a diagram is performed by the graph transformation engine AGG [2]. TIGER extends AGG by means for concrete syntax definition. From the VL definition, Java source code is generated, implementing an ECLIPSE visual editor plug-in based on GEF. Thus, the generated editors appear in a timely fashion, conforming to the ECLIPSE standard for graphical tool environments.

Tiger Architecture and User Interface. The two major parts of the TIGER environment are the *Designer* [3] and the *Generator* [4]. A VL is specified by the *Designer* providing the following four parts: the abstract syntax, the concrete syntax of the corresponding abstract elements, the start graph, and the syntax rules which define the VL editing operations. After defining the VL in the *Designer* the *Generator* is evoked to generate an editor as ECLIPSE plug-in based on GEF.

The TIGER user interface makes extensive use of the standard elements provided by the ECLIPSE workbench paradigm. Fig. 1 shows a few sample designer views and editors arranged in the Tiger perspective defining a VL for automata: the tree view 1 shows the hierarchical structure of a VL alphabet, a visual editor 2 is used to define the layout for a symbol type, and a properties view 3 allows to change values for graphical layout properties of the selected ellipse

A. Schürr, M. Nagl, and A. Zündorf (Eds.): AGTIVE 2007, LNCS 5088, pp. 583–584, 2008.

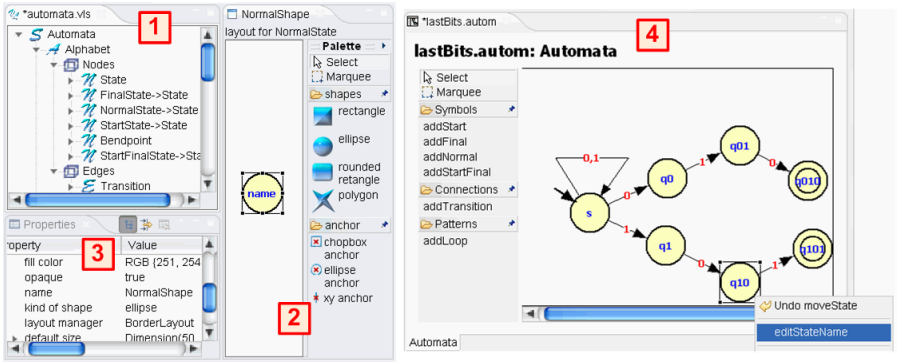

Fig. 1. The TIGER perspective in ECLIPSE and the generated automata plug-in

figure. Screenshot 4 shows the generated editor plug-in for automata. The editor palette contains VL-specific creation operations, grouped into categories *Symbols* (for creating symbols), *Connections* (for creating connections between two symbols) and *Patterns* (for modifying patterns consisting of more than one symbol). Dialogs are used for the definition of input parameter and for edit operations which can be evoked in the context menu of a selected symbol.

Note that graph transformation-based editors like TIGER, in contrast to related meta-model-based editors like GMF [5] and AToM^3 [6] or MetaEdit+ [7], do not only offer basic editor commands, but also support complex editing commands which insert or manipulate larger model parts consisting of a number of elements. With complex editing commands, model optimizations, such as model refactoring, as well as model simulation may be performed.

References

1. Eclipse Consortium: Eclipse Graphical Editing Framework (GEF) – Version 3.2 (2006), http://www.eclipse.org/gef
2. Taentzer, G.: AGG: A Graph Transformation Environment for Modeling and Validation of Software. In: Pfaltz, J.L., Nagl, M., Böhlen, B. (eds.) AGTIVE 2003. LNCS, vol. 3062, pp. 446–456. Springer, Heidelberg (2004)
3. Ermel, C., Ehrig, K., Taentzer, G., Weiss, E.: Object Oriented and Rule-based Design of Visual Languages using TIGER. In: GraBaTs 2006, vol. 1. EC-EASST (2006)
4. Ehrig, K., Ermel, C., Hänsgen, S., Taentzer, G.: Generation of visual editors as Eclipse plug-ins. In: Proc. ASE 2005, pp. 134–143. IEEE Computer Society, Los Alamitos (2005)
5. Eclipse Consortium: Eclipse Graphical Modeling Framework (GMF)(2007), http://www.eclipse.org/gmf
6. de Lara, J., Vangheluwe, H., Alfonseca, M.: Meta-Modelling and Graph Grammars for Multi-Paradigm Modelling in AToM^3. Software and System Modeling 3(3), 194–209 (2004)
7. Tolvanen, J., Rossi, M.: MetaEdit+: Defining and Using Domain-Specific Modeling Languages and Code Generators. In: Proc. Object-oriented programming, systems, languages, and applications (OOPSLA 2003), pp. 92–93. ACM Press, New York (2003)