ViewPoint-oriented Software Development: Tool Support for Integrating Multiple Perspectives by Distributed Graph Transformation

Michael Goedicke[1], Bettina Enders[1], Torsten Meyer[1] and Gabriele Taentzer[2]

[1] Specification of Software Systems, Department of Mathematics and Computer Science, University of Essen, Germany
{goedicke, enders, tmeyer}@informatik.uni-essen.de
[2] Theoretical Computer Science / Formal Specification Group, Department of Computing, Technical University of Berlin, Germany
gabi@cs.tu-berlin.de

Abstract. Co-operative development of distributed software systems involves to address the multiple perspectives problem: many stakeholders with diverse domain knowledge and differing development strategies collaborate to construct heterogeneous development artifacts using different representation schemes. The ViewPoints framework has been developed for organizing multiple stakeholders, the development processes and notations they use, and the partial specifications they produce. In this contribution we present a tool environment supporting ViewPoint-oriented software development based on a formalization by distributed graph transformation.

Introduction and Related Work

In system design the various development stages are visited more than once and quite different notations and process models need to be integrated in order to satisfy the requirements of the different stakeholders' views and their processes. It is therefore highly desirable to provide flexible conceptual means and related tool support for representing the various cooperating stakeholders' views and process models. In this contribution we use the ViewPoints framework to represent such views and processes. In addition the ViewPoints framework involves to tolerate inconsistent information in related ViewPoints until it seems necessary or appropriate to check and (re)establish consistency -- at least in some parts of the system [1]. The ViewPoints framework has been used quite successfully and has been documented in the literature [2, 1, 4].

The question which is addressed here is how tool support can be constructed to effectively represent the loosely coupled approach: some local development within a ViewPoint is followed by interaction with related ViewPoints via consistency checks.

The approach of distributed graph transformation supports the idea of loosely coupled ViewPoints as outlined above quite naturally. It realizes the separation between

the independent development of single local ViewPoints and the configuration and connection of a set of related ViewPoints in a structured way.

Distributed graph transformation which is based on the double-pushout approach to algebraic graph transformation is introduced formally in [6]. Using AGG [7] as a computing platform an adequate level of tool support can easily be constructed. The manipulation of representation schemes is expressed as graph transformation rules and the interaction and cooperation of distributed ViewPoints is adequately formulated as distributed graph transformation rules. As a result we gain tool support for ViewPoints and a corresponding formal presentation [5, 4]. As such it provides the possibility for formal analysis and most importantly a great deal of flexibility for integrating new ViewPoints.

The ViewPoints framework was devised by A. Finkelstein et al. [2] to describe complex systems. An overview of other approaches related to multiple perspectives in software development can be found in [3]. In [1] a general overview wrt inconsistency management within the ViewPoints framework is given.

In the chapter *The ViewPoints Framework* we introduce briefly our approach to ViewPoint-oriented software development. Based upon this we present tool support for our approach in the chapter *The ViewPoint Tool*.

The ViewPoints Framework

A *ViewPoint* is defined to be a locally managed object or agent which encapsulates partial knowledge about the system and its domain. It contains partial knowledge of the design process [2]. The knowledge is specified in a particular, suitable representation scheme. An entire system is described by a set of related, distributable ViewPoints which are loosely coupled.

A single ViewPoint consists of five slots. The *style slot* contains a description of the scheme and notation which is used to describe the knowledge of the ViewPoint. The *domain slot* defines the area of concern addressed by the ViewPoint. *The specification slot* contains the actual specification of a particular part of the system which is described in the notation defined in the style slot. The fourth slot is called *work plan* and encapsulates the set of actions by which the specification can be built as well as a process model to guide application of these actions. Two classes of work plan actions are especially important: *In-ViewPoint check actions* and *Inter-ViewPoint check actions* are used for checking consistency within a single ViewPoint or between multiple ViewPoints, respectively. The last slot of a ViewPoint called *work record* contains the development history in terms of the actions given in the work plan slot.

A *ViewPoint template* is a kind of ViewPoint type. It is described as a ViewPoint in which only the style slot and the work plan slot are specified, i.e. the other slots are empty. When creating a new ViewPoint, the developer has the opportunity to use an existing ViewPoint template instead of designing the entire ViewPoint from scratch.

The ViewPoints framework is independent from any particular development method and actively encourages multiple representations. Software development methods and techniques are defined as sets of ViewPoint templates which encapsulate

the notations provided as well as the rules how they are used. Integration of methods and views is realized by such rules referring to multiple ViewPoint templates.

A more detailed description of the ViewPoints framework and its formalization by distributed graph transformation is given in [4]. In the next section we now present a brief overview of tool support.

The *ViewPoint Tool*

The *ViewPoint Tool* comprises three main components: the *ViewPoint manager*, the *template editor* and *the ViewPoint editor*. While the *ViewPoint manager* serves as a central tool to coordinate all activities within using the ViewPoints framework, the *template editor* allows to design ViewPoint templates – i.e. styles combined with work plan actions – and the *ViewPoint editor* allows to work with specifications and work records of actual ViewPoints. All ViewPoints used in the *ViewPoint editor* have to be instantiated from existing ViewPoint templates developed by the *template editor*.

The *ViewPoint manager* serves to organize all developed ViewPoint templates and all actual ViewPoints (cf. Figure 1). It is used as a starting point to enter the *template editor* and the *ViewPoint editor*. First ViewPoint templates can be created which then can be developed further within the *template editor*. Then actual ViewPoints can be instantiated from a template which are usable within the *ViewPoint editor*.

The *template editor* allows to edit the work plan slot and the style slot of a ViewPoint template. All actions of the ViewPoint template's work plan have to be modeled as graph transformation rules (cf. Figure 2). The *ViewPoint editor* allows to edit a ViewPoint instantiated from a ViewPoint template developed by the *template editor*. All actions defined in the corresponding template's work plan can be applied to build an actual specification. Figure 3 depicts a *ViewPoint editor* window, the actual specification is displayed on the left and all work plan actions are listed on the right.

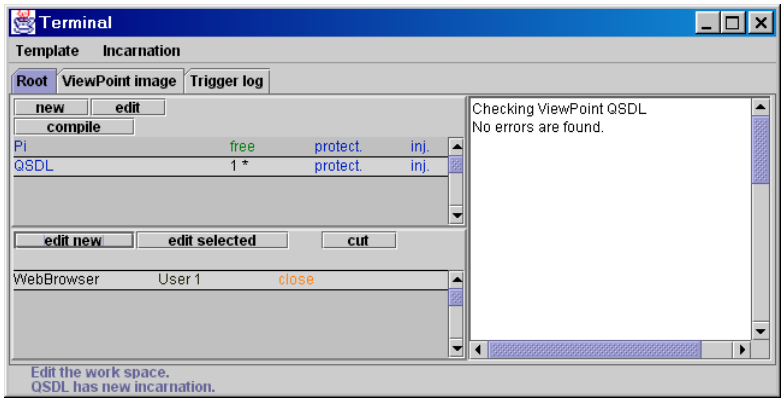

Fig. 1. *ViewPoint manager* window.

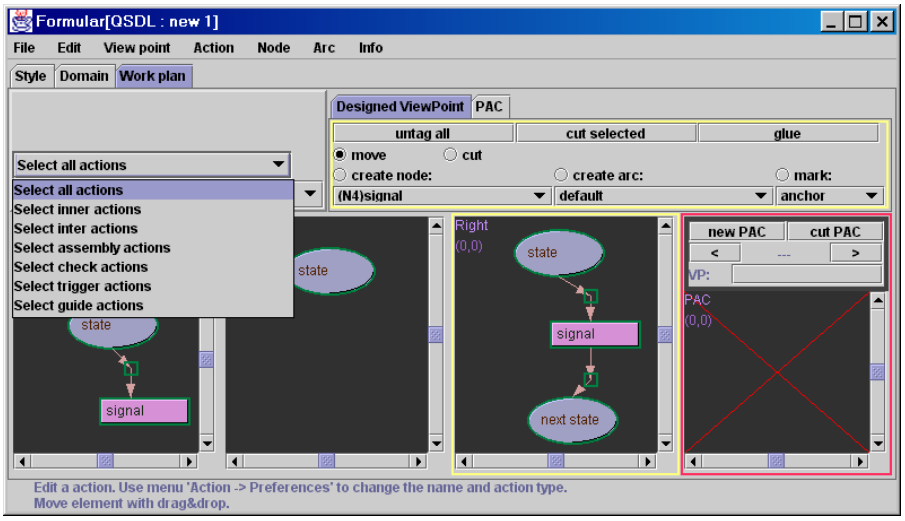

Fig. 2. The work plan window of the *template editor*.

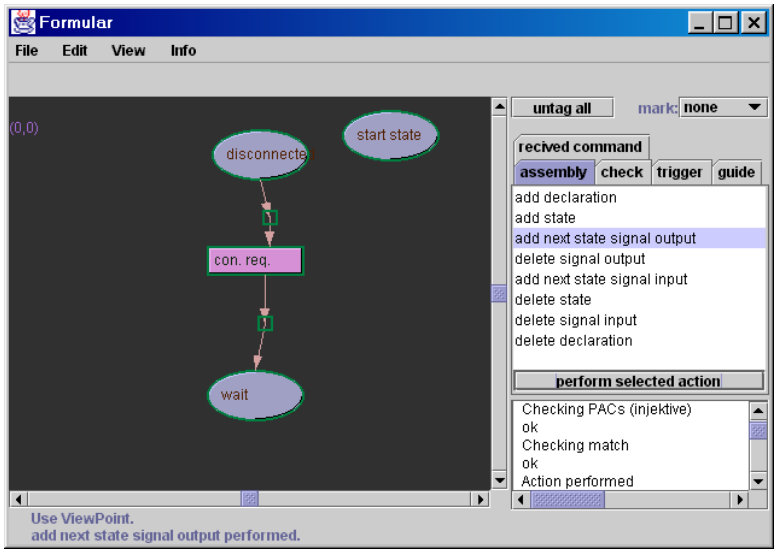

Fig. 3. The *ViewPoint editor*.

A more detailed description of the ViewPoint Tool – including distribution aspects and inconsistency management – is given in [7].

Conclusion and Further Work

In this contribution we have sketched a brief introduction to the ViewPoints framework and the *ViewPoint Tool* environment. Various case studies modeling non-trivial applications (e.g., integration of architecture design and performance evaluation views [5]) have shown that our implementation meets all the requirements for supporting the multiple perspectives problem in a flexible ViewPoint environmet. The present version of the *ViewPoint Tool* is based on the local version of AGG [9]. Currently we are working on integrating the features of a prototype AGG version realizing distributed graph transformation.

References

1. Easterbrook, S. and Nuseibeh, B., "Using ViewPoints for Inconsistency Management", *BCS/IEE Software Engineering Journal*, pp. 31-43, 1996.
2. Finkelstein, A., Kramer, J., Nuseibeh, B., Finkelstein, L., and Goedicke, M., "Viewpoints: A Framework for Integrating Multiple Perspectives in System Development", *Int. Journal of Software Engineering & Knowledge Engineering*, vol. 2(1), 1992.
3. Finkelstein, A. and Sommerville, I., "The Viewpoints FAQ", *Software Engineering Journal*, vol. 11 (1), pp. 2-4, 1996.
4. Goedicke, M., Meyer, T., and Taentzer, G., "ViewPoint-oriented Software Development by Distributed Graph Transformation: Towards a Basis for Living with Inconsistencies", *Proc. 4th IEEE International Symposium on Requirements Engineering*, Limerick, Ireland, 1999.
5. Goedicke, M., Enders, B., Meyer, T. and Taentzer, G., "Tool Support for ViewPoint-oriented Software Development", *Proc. International Workshop and Symposium AGTIVE*, Kerkrade, The Netherlands, 1999, Lecture Notes on Computer Science, Springer, to appear.
6. Taentzer, G., Fischer, I., , Koch, M., and Volle, V., "Distributed Graph Transformation with Application to Visual Design of Distributed Systems", in Rozenberg, G. (ed.), *Graph Grammar Handbook 3: Concurrency & Distribution*, World Scientific,1999.
7. Taentzer, G., Ermel, C., and Rudolf, C., "AGG-Approach: Language and Tool Environment", in Rozenberg, G. (ed.), *Graph Grammar Handbook 2: Specification & Programming*, World Scientific, 1999.