On the Description of Complex Systems with Multi-View Visual Languages

Juan de Lara
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Madrid (Spain)
jdelara@uam.es

1. Presentation

I'm an associate professor at the computer science department of the Universidad Autónoma in Madrid, where I teach automata theory, software engineering and modelling and simulation. My research interests are broadly in the areas of modelling and simulation, domain specific visual languages, and graph transformation. For my PhD thesis (2000) I worked in the design of OOCSMP, an object oriented continuous simulation language, and on a compiler able to generate Java applets from the models. The main idea was to integrate such generated programs in educative materials in the web. In this way, the language was designed to handle many visualization and multimedia elements [LA01], allowing visual interactive simulations. Later, I made a post-doc at the MSDL Lab, headed by Hans Vangheluwe, where we worked in the construction of AToM^3 [LV02], a meta-modelling tool which is able to generate customized modelling environments for Domain Specific Visual Languages (DSVL). The tool allows the manipulation of such DSVL by means of graph transformation techniques [EEPT06]. I've also been collaborating for some years with the group of Hartmut Ehrig at TU Berlin in the integration of graph transformation and meta-modelling techniques. Currently, I work together with Esther Guerra from the University Carlos III at Madrid in the description and manipulation of multi-view visual languages (such as UML), and in the extension of AToM^3 for these purposes. On the other hand, in the last few years I've been involved in several projects with the European Space Agency, where we have built simulators for the detection and prevention of electron discharges in waveguides [LPAG06].

2. Workshop Research Statement

2.1. Introduction

Visual Languages (VLs) are extensively used in many engineering activities for the specification and design of systems. As these become more complex, there is a need to break down specifications into smaller, more understandable parts that use the most appropriate notation. Thus, there are VLs (such as UML) made of a family of diagrams types used for the description of the different aspects of a system. We call such VLs Multi-View Visual Languages (MVVLs) [GDL05]. In these languages, the user has a need to build models (using the different diagram types) and check their consistency; of querying models to obtain partial models; and of transforming models into other formalisms for analysis. All these artefacts can be considered different kinds of views of the system. This necessity has been recently expressed by the OMG by defining a standard language for expressing Queries, Views and Transformations (QVT, see [QVT05]).

The (abstract and concrete) syntax of VLs can be described by means of meta-modelling. In the case of a MVVL, the syntax of the complete MVVL is defined with a meta-model (e.g. as in the case of UML). Each diagram type (or viewpoint) has its own meta-model too, which is a part of the complete MVVL meta-model. At the model level, the user builds different system views conformant to a viewpoint meta-model. System views are merged together in a unique model called *repository*. Triple Graph Grammars (TGGs) [S94] automatically derived from the meta-models perform the merging, allow incremental updates and relate the system views and the repository. They also provide syntactic consistency and

change propagation from a view to the others. In addition, it is possible to generate TGGs modelling different behavioural patterns for view management (e.g. cascading deletion vs. conservative deletion).

Figure 1 shows the structure that arises when defining a MVVL. Here we see the different diagram definitions, which may have some overlapping parts in the meta-model. In the figure, we have identified the overlapping elements for each pair of diagrams. The meta-model comprises three diagrams. Two of them overlap with the third, but not between them. Note how the structure can be formally described in categorical terms, for example each all the squares formed by two diagrams, their overlapping objects and the complete meta-model are *pullbacks*.

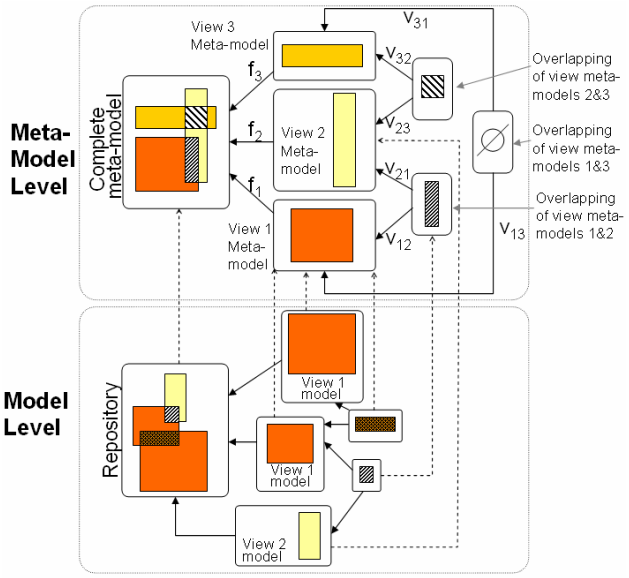

Figure 1: A MVVL at the Meta-model and Model Levels.

At the model level, in order to guarantee syntactic consistency, we build a unique model (called *repository*) by gluing all the system views built by the user. The resulting model is an instance of the complete VL meta-model. This gluing operation is performed by automatically generated TGG rules (derived from the meta-model information). Updating the repository (as a result of a view model modification) may leave some other view models in an inconsistent state. At this point, other automatically generated triple rules update the necessary view models. In this way, we have TGGs that propagate changes from the view models to the repository and the other way around. This is shown in Figure 2. This mechanism is similar to the model-view-controller (MVC) framework. In the figure, each model has in its upper-right corner the meta-model it is compliant with. Step 1 is usually performed by the user; steps 2, 3 and 4 are automatically performed by the tool.

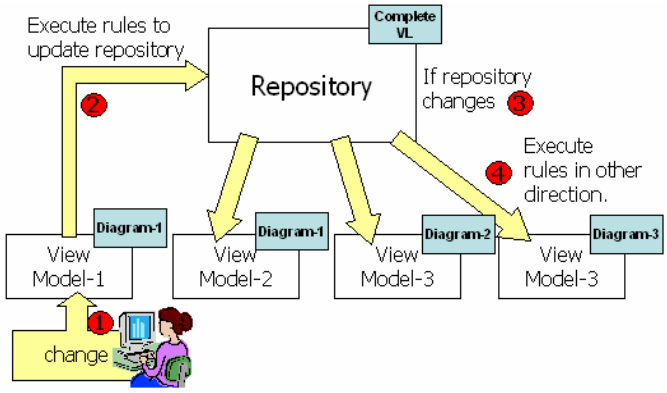

Figure 2: Change Propagation

For static semantics checking, the MVVL designer should provide additional rules. The system views (or the repository) can be translated into other formalisms for dynamic semantics checking, analysis and

simulation. We call the target model *semantic view*. The translation is defined by the MVVL designer by means of TGGs that establish correspondences between the elements in the source model and its semantic view. Thus, the results of the analysis can be back annotated and shown in the base model, likely more intuitive for the user.

The user may also need to extract information from the system, which may be scattered through the different views. The information to be extracted may even not be conformant with the meta-model of any of the diagrams. Following a visual approach, we have proposed *graph query patterns* [GL06] as a declarative visual query language to obtain derived views from a base model (another view or the repository). Starting from the patterns, a TGG is automatically generated to build the derived view and maintain it consistent with respect to changes in the base model (i.e. derived views are incremental). If the query is predefined by the MVVL designer (and later used by a specific kind of user) we call it *audience-oriented view*.

2.2. Questions to be investigated during the Workshop

There are many open questions that need further evaluation and investigation. Some of them are listed below:

- One can ask whether a multi-view system as described below can be considered a multi-formalism system (in the sense of [VLM02]). In the case of a multi-view system, one uses the most appropriate diagram type for each aspect to be described. However, all the notations are related through a common meta-model, which assumes well-known relations and constraints between concepts in the different diagrams. In the case of a multi-formalism system, the different notations to be used need not be related *a priori*. Therefore, one may ask which mechanisms are necessary to build *consistent* multi-formalism systems. Can we use the same approach as in MVVLs? What is needed in addition?.

- Other interesting issue is the comparison of graph-based and textual-based (like OCL [WK03]) techniques for expressing and obtaining queries on models, in the context of Domain Specific Visual Languages (DSVLs). Structural constraints may be more understandable with graph patterns, which in addition can use the concrete syntax of the DSVL. On the other hand, OCL may be able to express for example cycles, which is difficult to do with patterns in a general way. Rensink in [R04] showed that a nesting of positive and negative patterns of arbitrary depth is necessary to make patterns equivalent in expressive power to first order logic. What are the advantages of each approach? Which kind of queries can be more easily expressed with each one?.

- For *semantic views*, a mechanism for back-annotation of analysis results is needed. In this way, the analysis process can be hidden, and the user gets the results back in the context of the original language. Moreover, one may have different semantic domains, with different analysis methods, which should be selected depending on the property that has to be proved. Moreover, one may want the latter to be expressed using a user-friendly notation. Is there a general mechanism to do this?.

3. Success Conditions

I will consider the workshop a success if a) I can get may suitcase back ☺ and b) if some of the questions proposed (or other similar ones in related areas) is investigated, and either solved partially, or results in further interesting questions.

References

[EEPT06] Ehrig, H., Ehrig, K., Prange, U. and Taentzer, G. 2006. "*Fundamentals of Algebraic Graph Transformation*". Monographs in Theoretical Computer Science. Springer.

[GDL05] Guerra, E., Díaz, P., de Lara, J. 2005. "*A Formal Approach to the Generation of Visual Language Environments Supporting Multiple Views*". Proc. IEEE VL/HCC. pp.: 284-286. Dallas.

[GL06] Guerra, E., de Lara, J. "*Graph Transformation vs. OCL for View Definition*". Proc. 1st Algebraic Foundatios for OCL and Applications. WAFOCA'06, Valencia (Spain).

[LA01] de Lara, J., Alfonseca, M. 2001. "*Constructing Simulation-Based Web Documents*". IEEE Multimedia, Special issue on Web Engineering. January-March. 2001. pp 42-49

[LV02] de Lara, J., Vangheluwe, H. 2002. "*AToM^3: A Tool for Multi-Formalism Modelling and Meta-Modelling*". In Proc. ETAPS/FASE'02. LNCS 2306, pp.: 174 - 188. Springer-Verlag. See also the AToM3 home page: http://atom3.cs.mcgill.ca.

[LPAG06] de Lara, J., Pérez, F., Alfonseca, A., Galán, L., Montero, I., Román, E., Raboso, D. 2006. "*Multipactor Prediction for On-Board Spacecraft RF Equipment with the MEST Software Tool*". IEEE Transactions on Plasma Science, April 2006.

[QVT05] QVT specification by OMG: http://www.omg.org/docs/ptc/05-11-01.pdf

[R04] Rensink, A. 2004. "*Representing First-Order Logic Using Graphs*". Proc. ICGT'04, LNCS 3256, pp.: 319-335.

[S94] Schürr, A. 1994. "*Specification of Graph Translators with Triple Graph Grammars*". In LNCS 903, pp.: 151-163. Springer.

[VLM02] Vangheluwe, V., de Lara, J., Mosterman, P. 2002. "*An Introduction to Multi-Paradigm Modelling and Simulation*". Tutorial. En AI, Simulation and Planning – AIS'2002. Pp.: 9-20. Lisbon.

[WK03] Warmer, J., Kleppe, A. 2003. "*The Object Constraint Language: Getting Your Models Ready for MDA, 2nd Edition.*" Pearson Education. Boston, MA.