Scenario integration via the transformation and manipulation of higher-order graphs*

Hongzhi Liang

School of Computing, Queen's University, Canada
liang@cs.queensu.ca

Abstract. The integration of different models, such as scenarios, is an important component of the requirements engineer's work. If manually performed, the integration operation is error-prone and time consuming. Thus, an integrated computer-aided environment would be desirable. In the paper, we propose a framework based on mathematical category theory machinery of algebraic operations with higher-order graphs that provides a formalization and a generic pattern for scenario integration. In order to evaluate the proposed framework, we have instantiated the framework and are currently developing an experimental tool.

1 Introduction

As applications become larger and more complex, multiple teams may parallelly involve in different development phases. For instance, during the requirement phase, partial behavior models are created independently by different modelers and then they are merged in order to describe a more comprehensive behavior. Thus, the ability to integrate, e.g., merge two or more models, becomes important in such scenarios. Although integration could be manually performed, the operation is likely error-prone and time consuming. Therefore, an integrated computer-aided environment for the integration operation would be very useful. Our research goal, hence, is aimed to develop an integrated environment where model management (MMt) tasks, for instance model integration, could be performed in an intelligent way.

A fundamental problem of model integration is that different models representing different views of the same universe can essentially overlap in different ways. A proper integration has to take this into account, otherwise the result will *implicitly* contain duplications and redundancies. The question, therefore, is what should be specified in addition to the set of models so that their integration would merge all the information contained in the views without loss and duplication. Moreover, we need to have a generic pattern for model overlap not dependant on peculiarities of a particular modeling language. Ideas from the category theory will be used to build such a generic pattern for the entire view/model integration operation.

* I would like to thank my supervisor Juergen Dingel and research fellow Zinovy Diskin for their contributions to this work.

2 Research Plan

Our research approach can be separated into two phases. The first phase is the development of the theoretical framework, where we address the problem of integration by first giving a precise specifications of *what* the operations to be performed are, and then proceeding to *how* they can be implemented in an efficient way. The second phase is the implementation phase where we instantiate the framework. An experimental tool that integrates UML sequence diagrams [6] will be created to help us evaluate the proposed theoretical framework. The current status of this research is that we have finished the theoretical framework and one of the components of the tool. We will briefly elaborate the theoretical framework and the implementation.

2.1 Theoretical framework

In [2], we presented an extendable framework that provides a formalization and a generic pattern for integrating models, such as UML2 sequence diagrams or similar notations such as ITU Message Sequence Chart [4]. The framework is based on mathematical category theory machinery of algebraic operations with higher-order graphs. Our theoretical foundation can address, for instance, the fundamental problem of model overlapping by using a proper integration that takes this problem into account and merges all the information contained in the views without loss and duplication. A distinctive feature of our approach is the use of *derived* elements, where information that is explicitly specified in one view can be implicit or derived in another view.

In particular, to integrate sequence diagrams, we first showed that the basic structure of sequence diagrams can be formalized as a chain of higher-order graph mappings: base graph $\longleftarrow$ collaboration graph $\longleftarrow$ occurrence graph. Base and collaboration graphs provide the structural basis of the sequence diagrams in question, which define class and message types, and objects and message channels, respectively. Occurrence graphs are just a partial order of event and message occurrences of the sequence diagrams. Based on the formalization, a procedure of sequence diagram integration has been developed and applied to a non-trivial example in [2]. A generic pattern for model integration is then summarized as follows:

1. Formalization. We fix some universe $\mathcal{U}$ of higher-order graph-based structures like scenario graphs. We will call objects of this universe *$\mathcal{U}$-graphs* or just *graphs*. Models to be integrated are presented as $\mathcal{U}$-graphs, $\mathbf{G} = \{G_1...G_m\}$, which we call *views*.

2. Specifying view correspondences. Correspondences between models are specified by another family of graphs, $\mathbf{R} = \{R_1...R_n\}$. As we have already mentioned, the latter may contain new information not captured by views. Mathematically, graphs R_j play the same role of input structures for the merge algorithm as view graphs G_i. Thus, we come to a family of graphs $\mathbf{H} = \{H_1...H_{m+n}\}$,

$\mathbf{H} = \mathbf{G} \cup \mathbf{R}$, to be integrated modulo some correspondences (equivalences) between them. To set these correspondences, we may need to augment the view graphs with new elements derived by the operation of arrow composition. In this way we come to a family of augmented graphs $\overline{\mathbf{H}} = \{\overline{H}_1 ... \overline{H}_{m+n}\}$ together with a family of mappings (determined by correspondences) $\mathbf{h} = h_1 ... h_k$ between them.

3. Merge. The configuration (generalized span) $(\overline{\mathbf{H}}, \mathbf{h})$ is automatically merged according to the algorithm described in [2]. The procedure returns a cospan of graphs and mappings, $\mathbf{S} = (S, \overline{\iota'}_1 ... \overline{\iota'}_{m+n})$, $\overline{\iota'}_i : \overline{H}_i \to S$. Its head S may contain derived elements.

4. Normalization. In the merge graph S a subgraph S_0 should be chosen in such a way that any element in S can be derived from elements of S_0. Besides this technical requirement, the chosen subgraph should be compact and semantically meaningful, and should provide transparent meaning for derivations required to augment it up to S.

2.2 Implementation

Although we have shown that the theoretical framework worked well on a small number of sample scenarios [2], we would like to create an experimental tool to help us further evaluate the framework. An implementation of the framework that integrates UML sequence diagrams is currently under development. The tool itself will be implemented as an extension to an Eclipse-based UML modeling tool, for instance IBM Rational Software Architect (RSA). The reasons that we chose to create an extension rather than build a tool from scratch are 1)we will benefit from the existing sequence diagrams editor, and the diagrams exporting/importing features, 2)we can fully take advantage of the standard features of the Eclipse platform and a variety of plug-ins, such as EMF, UML2 and GMF.

During an initial investigation, we have discovered that a desired implementation needs at least the following six components as shown in Figure 1:

1. *SD2HG Transformer*: required by step 1 of our generic integration pattern, this automatic transformer would transform sequence diagrams, e.g., exported from RSA, to higher-order graphs, i.e., base, collaboration and occurrence graphs. Some existing transformation tools could be used to develop this component, for instance, some of the prominent choices including TXL [8] and XSLT [9]. A TXL-based transformer is currently under development.
2. *Mapping Editor*: this editor realizes step 2 of the pattern. It would take higher-order graphs generated by the *SD2HG Transformer* and produce new higher-order graphs that contain mappings between the input higher-order graphs. The users, i.e., requirements engineers, specify the correspondences between sequence diagrams, or more precisely, sequence diagrams represented as higher-order graphs by using this editor interactively. Moreover, any derived elements could also be explicitly specified by using this editor.

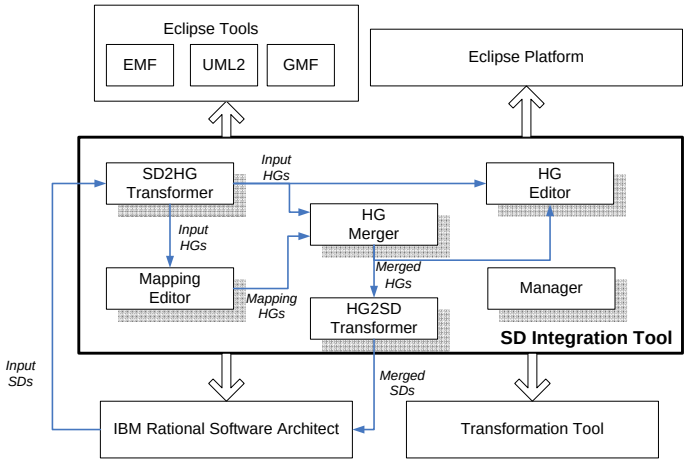

Fig. 1. The architecture of the proposed SD Integration Tool: the central and its contained boxes show the six components of the proposed tool; the thin arrows and labels illustrate the control/data flow; the bold arrows and the surrounding boxes denote the existing technologies the proposed tool would utilize.

3. *HG Merger*: an automatic higher-order graphs merger takes as input higher-order graphs from both the *SD2HG Transformer* and *Mapping Editor* and produces merged higher-order graphs. Basically, this component would provide the functionalities required by step 3 and 4 of the generic pattern.
4. HG2SD Transformer: since the ultimate goal of this tool is to output integrated sequence diagrams, this component, as the dual of the *SD2HG Transformer*, would transform higher-order graphs produced by the *HG Merger* back to sequence diagrams. Again, the transformation could be implemented using, e.g., TXL or XSLT.
5. HG Editor: an interactive graphical editor of higher-order graphs. Some obvious functionalities of this component include editing, modifying, and displaying higher-order graphs. Other functionalities, for instance, validating an existing or a newly created higher-order graph, should also be provided by this component. By visualizing the generated higher-order graphs, this editor would be very useful to help us validate the correctness of our *SD2HG Transformer*, *Mapping Editor* and *HG Merger*. An implementation based on the Eclipse Graphical Modeling Framework (GMF) [3] has already been completed.
6. SD Integration Tool Manager: an accessory component facilitates the overall sequence diagram integration process and manages the control/data flows between components. It would also act as the extension or plug-in to RSA.

2.3 Evaluation

To evaluate the implementation phase of this research, we will perform tests on different case studies, for instance, the wholesale-retail example [2] and the shut-

tle system project [7]. Then, the evaluation results on the implementation phase can be used as feedback for our theoretical framework. Finally, any modifications reflecting the feedbacks would help us fine tune our framework.

3 Related work

The idea of using algebraic colimit operations for model merge is known for a long time, e.g., in [1], but mainly in the context of data model merging. There are just a few works that employ categorical ideas and machineries for behavioral model merge, for example, [5]. The paper [5] is the only published application of colimits to scenario merge that we are aware of. In this paper, scenarios specified by MSCs are encoded as partially-ordered multisets – a well-known and deserved language. However, the string-based (rather than graph-based) formalization used by the authors results in a bulky definition of morphism and makes the entire integration procedure less transparent and less scalable beyond small examples.

4 Contributions

The contributions expected from this research are: 1) definition of a formal framework providing the necessary theoretical foundations for model integration in general and scenario integration in particular. 2) implementation of the framework in a modeling tool such as RSA. 3) a fully functional UML sequence diagram integration tool.

References

1. P. Buneman, S. Davidson, and A. Kosky. Theoretical aspects of schema merging. In *Advances in Database Technology - EDBT'92*, Springer LNCS # 580, 1992.
2. Z. Diskin, J. Dingel, and H. Liang. Scenario integration via higher-order graphs. Technical Report 2006-517, Queen's University, 2006. URL: http://www.cs.queensu.ca/TechReports/Reports/2006-517.pdf.
3. Eclipse. The Eclipse Graphical Modeling Framework. Online, http://www.eclipse.org/gmf/, 2007.
4. ITU-TS. Recommendation Z.120: Message Sequence Chart (MSC), 2000.
5. J. Klein, B.Caillaud, and L. Hlout. Merging scenarios. In *9th Int.Workshop on Formal Methods for Industrial Critical Systems*, ENTCS, pages 209–226, 2004.
6. Object Management Group, `http://www.uml.org`. *Unified Modeling Language: Superstructure. version 2.1.1. Formal/07-02-05*, 2007.
7. University of Paderborn Software Engineering Group. Shuttle system case study. Online, http://www.eclipse.org/gmf/, 2007.
8. TXL. About TXL. Online, http://www.txl.ca/nabouttxl.html, 2007.
9. W3. XSL Transformations. Online, http://www.w3.org/TR/xslt, 2007.