A Relationship-Driven Approach to View Merging

Mehrdad Sabetzadeh, Shiva Nejati, Steve Easterbrook, Marsha Chechik
Department of Computer Science, University of Toronto
Toronto, Ontario, Canada
{mehrdad,shiva,sme,chechik}@cs.toronto.edu

ABSTRACT

A key problem in view-based software development is merging a set of disparate views into a single seamless view. To merge a set of views, we need to know how they are related. In this extended abstract, we discuss the methodological aspects of describing the relationships between views. We argue that view relationships should be treated as first-class artifacts in the merge problem and propose a general approach to view merging based on this argument. We illustrate the usefulness of our approach by instantiating it to the state-machine modelling domain and developing a flexible tool for merging state-machines.

Categories and Subject Descriptors

D.2.1 [**Software Engineering**]: Requirements/Specifications - *methodologies*.

General Terms

Modelling, Design, Distributed Development.

Keywords

View-Based Modelling, View Merging.

1. INTRODUCTION

An effective way to manage complexity and promote flexibility in large-scale software development is to describe a proposed system using loosely-coupled partial models, known as *views* [8]. Each view considers the system from a particular angle, making it possible to scope the amount of information that analysts need to deal with at any given time [7]. Further, by providing means for separating the contributions of different sources, views allow stakeholders to express and maintain their individual perspectives on the system [6].

A key problem in view-based development is *view merging*. From time to time, we need to integrate information from several views into a single one in order to gain a unified perspective, to understand interactions between views, or to perform various types of end-to-end analysis. View merging spans several application areas. In information systems, view merging is an important step during conceptual database design for constructing an abstract schema that captures the data requirements of all the involved participants [4]. In ontology research, view merging may be employed to build a global ontology from a set of local ontologies originating from different communities [9]. Software engineering deals extensively with view merging – several papers study the subject in specific domains including early requirements [14], use-cases [12], class diagrams [2], software architectures [1], and state-machines [13, 16, 10].

View merging frameworks differ in several factors including the modelling notations they support, the contexts in which they operate, and the assumptions they make about the nature of views and their intended use. Despite these differences, there are a number of common concerns that all merge frameworks face. These concerns mostly have to do with the *methodological* requirements of view merging and involve important questions about how to specify the relationships between views, maintain traceability between views and their merges, and capture the evolution of views during merge.

In this extended abstract, we focus on the question of how to specify the relationships between views. Specification of view relationships is a crucial step in the merge process for characterizing how the contents of different views overlap or interact. In the past several years, we have been studying the view merging problem in different domains. With hindsight, we have identified two key factors concerning the specification of view relationships:

1. Views are often developed by distributed teams. As a result, one can never be entirely sure how the concepts in different views relate to one another. Therefore, it is important to be able to hypothesize and explore alternative ways of interrelating the concepts in a set of views and to configure view merges based on these alternatives.

2. View merging may be utilized to achieve different goals at different stages of development. For example, in early stages, we may merge a set of views as a way to unify the vocabularies of different stakeholders. For this purpose, it is convenient to treat views as syntactic objects without rigorous semantics. In later stages, however, we often want to account for the semantics of views and produce merges that preserve certain semantic properties. This means that we may need to apply different merge algorithms to a set of views as they move through different stages of development. These algorithms typically require different types of relationships to be built between views

FSE'06, November 5–11, 2006, Portland, Oregon.

(e.g. syntactic and semantic). Hence, we need the flexibility to define different types of view relationships and to associate each type with certain merge algorithms.

These factors emphasize the need to make explicit all information about the relationships between views, and indeed suggest that view relationships should be treated as *first-class artifacts* in the view merging problem.

In light of this observation, we propose a general approach for view merging whereby the relationships between views can be specified explicitly and used for driving the merge process. Figure 1 shows an outline of our approach: Given a set of views, users can define different types of relationships between views and for each type, explore alternative ways of mapping the views to one another. The views and a selected set of relationships are then used to compute a merge by applying an appropriate merge algorithm. The merge is then presented to users for further analysis which may lead to the discovery of new relationships or the invalidation of some of the existing ones. Users may then want to start a new iteration by revising the relationships and executing the subsequent activities.

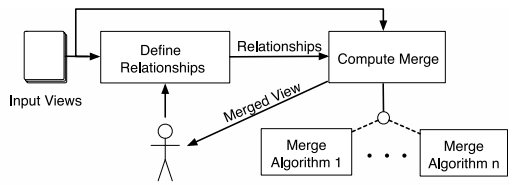

Figure 1: Outline of our approach

We have implemented a proof-of-concept tool, TReMer [15], based on our proposed approach. We describe this tool in the next section.

2. TOOL SUPPORT

Our tool, TReMer, is an instantiation of the approach outlined in Figure 1 to the domain of state-machine modelling. TReMer currently supports two view merging algorithms, *structural* and *behavioural*, as described below.

Structural merging is used during early phases of development where the most important goals are: exploring the ontological relationships between the terms used in different views, aligning the conceptual structures of different stakeholders, toleration of inconsistencies and deferral of their resolution to later phases, and producing an abstract perspective of the entire system being developed. Our structural merging algorithm treats state-machines as graphs [13]. Relationships between state-machines are captured by subgraphs, also referred to as *connectors*. Merges are computed based on a category-theoretic concept called *colimit* [3]. Colimit computation is based on syntactic relationships between graphs, without regard to their underlying semantics. Therefore, merges may not preserve the semantic properties of state-machines.

Our second algorithm, behavioural merging, is motivated by the need to preserve the behavioural properties of a set of state-machines in their merges. The desirable behaviours of each state-machine are described using temporal logic formulas. Behavioural merging is more suitable for late stages of development where the goal is to obtain a sound and operational model of the system under development. In behavioural merging, it is assumed that any disagreements between stakeholders have already been resolved, and as such, all remaining discrepancies between the behaviours of the input views are treated as variabilities in the views' intended functionality. These variabilities are represented as conditional behaviours in the merge [11]. Relationships between state-machines are specified by binary relations over their state spaces. The merge is defined as a *common refinement* of a set of state-machines with respect to their relationships [16, 10]. This guarantees the preservation of temporal properties. Merges computed by our behavioural merging algorithm can potentially be used for synthesizing a correct implementation of the final system.

Concrete examples for illustrating our tool are provided in a separate appendix[1]. For details of our merge algorithms, refer to [13] and [10].

3. FUTURE WORK

For future work, we plan to extend our view merging approach to consider other important methodological concerns such as traceability and evolution. We are also studying how the merge operation interacts with other model manipulation operations such as matching and differencing. Some initial work in this direction has been reported in [5].

Acknowledgments. We thank the referees of the FSE'06 poster session for their insightful comments. Financial support was provided by OGS, NSERC, BUL, and an IBM Eclipse innovation grant.

4. REFERENCES

[1] M. Abi-Antoun, J. Aldrich, N. Nahas, B. Schmerl, and D. Garlan. Differencing and merging of architectural views. In *ASE*, 2006. (To appear).

[2] M. Alanen and I. Porres. Difference and union of models. In *UML*, pages 2–17, 2003.

[3] M. Barr and C. Wells. *Category Theory for Computing Science*. CRM, Montréal, Canada, 1999.

[4] C. Batini, M. Lenzerini, and S. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):323–364, 1986.

[5] G. Brunet, M. Chechik, S. Easterbrook, S. Nejati, N. Niu, and M. Sabetzadeh. A manifesto for model merging. In *Wkshp. on Global Integrated Model Management*, 2006.

[6] S. Easterbrook and B. Nuseibeh. Using viewpoints for inconsistency management. *SE J.*, 11(1):31–43, 1996.

[7] A. Egyed. *Heterogeneous View Integration and its Automation*. PhD thesis, USC, USA, 2000.

[8] A. Finkelsetin, J. Kramer, B. Nuseibeh, L. Finkelstein, and M. Goedicke. Viewpoints: A framework for integrating multiple perspectives in system development. *SEKE*, 2(1):31–58, 1992.

[9] Y. Kalfoglou and M. Schorlemmer. Ontology mapping: The state of the art. In *Semantic Interoperability and Integration*, number 04391 in Dagstuhl Seminars, 2005.

[10] S. Nejati and M. Chechik. Let's agree to disagree. Tech. Rep. CSRG-530, U. of Toronto, 2005.

[11] S. Nejati, M. Sabetzadeh, M. Chechik, S. Easterbrook, and P. Zave. Matching and merging of statecharts specifications. Submitted for publication, 2006.

[12] D. Richards. Merging individual conceptual models of requirements. *RE J.*, 8(4):195–205, 2003.

[13] M. Sabetzadeh and S. Easterbrook. Analysis of inconsistency in graph-based viewpoints: A category-theoretic approach. In *ASE*, pages 12–21, 2003.

[14] M. Sabetzadeh and S. Easterbrook. View merging in the presence of incompleteness and inconsistency. *RE J.*, 11(3):174–193, 2006.

[15] M. Sabetzadeh and S. Nejati. TReMer: A tool for relationship-driven model merging. In *FM*, 2006. Demo.

[16] S. Uchitel and M. Chechik. Merging partial behavioural models. In *FSE*, pages 43–52, 2004.

[1] Available from: `http://www.cs.toronto.edu/~mehrdad/pub/`