Synchronization Contexts as a Means to Support Collaborative Modeling

Niels Pinkwart

University of Duisburg-Essen, Faculty of Engineering,
47048 Duisburg, Germany
pinkwart@collide.info

Abstract. This paper presents an approach to support collaborative modeling with graph based representations. In particular, the problem of partially shared models with associated semantics is addressed, and an architectural solution to enable flexible modes of partial application synchronization under the constraint of retaining a common semantics in the shared model parts is presented.

1 Introduction

In science, the term *model* refers to a schematic, simplified and idealized representation of an object or a domain in which the relations and functions of the elements of the objects are made explicit. There is an analogy between the model and the object it describes in the sense that these two are structurally identical. Modeling is understood as the activity of creating, manipulating and using models. As models are a simplified and manageable means of understanding complex real phenomena, the importance of modeling in a number of professional and educational usage scenarios is evident [6].

A general function that computers can have in the domain of modeling is that they can serve as tools that execute models or run simulations based on models. Both is possible for many rather formal modeling languages like, e.g., Petri Nets [10] or System Dynamics [3]. Current networked computer systems are technically able to go beyond this. Archival and retrieval functions can, e.g., foster the exchange and re-use of modeling material. Networking also principally enables a cooperative synchronous use of modeling tools.

Among the variety of representations that can serve as a means for modeling, this paper concentrates on graph based ones - i.e., models consisting of visually explicit objects and their relationships. Several studies [8,12] indicate that this representational type (as opposed to, eg., textual forms or formulas) has certain advantages - including aspects like guidance for the users, the explicit structure of the model, and the fact that graphs seem to be inviting to users to "try out" creative solutions, which is an important aspect in modeling.

In addition, graph based representations are widely used, and the variety of modeling languages that rely on graph based representations (or that can, as one alternative, be represented in such a notation) is impressing. More formal

H. Fuks, S. Lukosch, and A.C. Salgado (Eds.): CRIWG 2005, LNCS 3706, pp. 145–152, 2005.

languages with exactly defined semantics are, e.g., Petri Nets [10], System Dynamics [3], or Entity-Relationship diagrams [2]. Languages with an intermediary level of formality (i.e., some parts of the expressions allow for an automatic interpretation and simulation, while others do not) include, e.g., most diagram types of the unified modeling language UML [1]. Finally, there are also a lot of qualitative modeling techniques that make use of graph based representations. In those, the object and link *types* can usually be exactly distinguished and are possible subject of interpretation, the *content* of these objects and links however is usually not accessible to computer based interpretation techniques. Examples of this category include mapping techniques like concept mapping [9].

This paper describes an approach that, retaining openness concerning the range of supported graph based modeling languages, focuses on the critical question of how to gain flexibly while (partially) sharing models that have associated formal or semiformal semantics.

2 Synchronization Contexts in Graph Based Modeling

2.1 Basics

Attempting to formalize shared graph based models, it is reasonable to start with a description of visual graph models:

Definition 1. *Let $\mathcal{N}$ be a set of elements called* node types*, and let N be a set of nodes. Then a mapping dom : $N \mapsto \mathcal{N}$ is called a* node type mapping*. The image of dom, written dom(N), is called* node domain *of N. Edge type mappings and edge domains are defined in analogy. If $dom_N : N \mapsto \mathcal{N}$ and $dom_E : E \mapsto \mathcal{E}$ are node type and edge type mappings, then a graph G=(N,E) is called a* typed graph *over ($\mathcal{N}$,$\mathcal{E}$).*

The following definition adds visual information to the concept of typed graphs:

Definition 2. *Given two sets V_N and V_E, called* visual node attributes *and* visual edge attributes*, then a pair L=(λ_N, λ_E) of mappings with $\lambda_N : N \mapsto V_N$ and $\lambda_E : E \mapsto V_E$ is called a* layout *of a Graph G=(N,E).*

A typed graph with associated visual attributes is referred to as a *visual typed graph*. This definition of a layout for a graph is abstract in the sense that it does not prescribe concrete sets V_N and V_E. Typical parameters would, e.g., be cartesian coordinates. Yet, a variety of alternatives (like, e.g., explored in visual language theory), are possible here.

To enhance visual typed graphs to "real" models, two more ingredients are necessary: syntax and semantics. While syntactic issues can be addressed using constraints over the visual typed graphs, the following definition for semantics is based on the formal notion of computational model semantics [5]:

Definition 3. *Let $\mathcal{G}$ be a set of visual typed graphs. Then a couple (D,Ip), consisting of a* semantic domain *D and a* semantic mapping *$Ip : \mathcal{G} \mapsto D$ is called a* graph semantics *for $\mathcal{G}$.*

This notion of graph semantics is independent of node and edge domains. Though this may appear unusual, as typically the semantics of graph based models are tightly bound to a modeling language and its primitives, the approach of partially decoupling semantics from concrete node and edge domains can contribute to model interoperability: it allows for defining semantic mappings that bridge gaps between modeling languages.

Although the semantic mapping is defined on the graph level, a lot of modeling languages allow for a decomposition of the semantic mapping in the sense of concrete node and edge semantics: here, notations like $Ip(n_G)$ (n being a node in the graph G) and, correspondingly, $Ip(e_G)$ make sense. Two examples:

- The semantics of a node in a calculation tree can be defined as the result of computing the value of the subtree.
- In the field of Petri Nets, the semantics of a place node can be identified as the number of tokens that the place contains.

2.2 Synchronization Contexts

One of the distinguishing factors between the present work and comparable approaches in the domains of metamodeling and visual languages is the explicit support for collaborative usage scenarios with flexibly shared representations. Though typically the concrete support for these mechanisms will be done on the concrete implementation level, there is a foundational issue that can be dealt with well already on the conceptual level: sharing graph structures, there is a possible discrepancy between flexibility of synchronization and coherence (or closure) requirements of models. The logical consequence of aiming at maximum degree of flexibility in sharing graph structures is to allow for a synchronization of arbitrary substructures (i.e. subgraphs) to the extreme case of having only single nodes synchronized. These partially shared structures have interesting application areas and allow for flexible work modes. An example for this is the following: with partially synchronized graphs, it is possible for users to privately work on the construction of a model and to "publish" only selected parts of it, e.g., a subgraph that contains some explicitly marked result elements. Insight into the way that these results were elaborated does not necessarily have to be granted to the group. There are also many ways of orchestrating these private/shared collaboration scenarios with partially shared models, as exemplified in the domain of mathematics [7].

While this degree of flexibility sounds attractive, there are also situations in which partially shared models may be problematic. Apart from the general question how edges could be coupled without also sharing the nodes that an edge connects, a critical point is that partially sharing models may lead to discrepancies between the semantics of the shared model parts. This is due to the

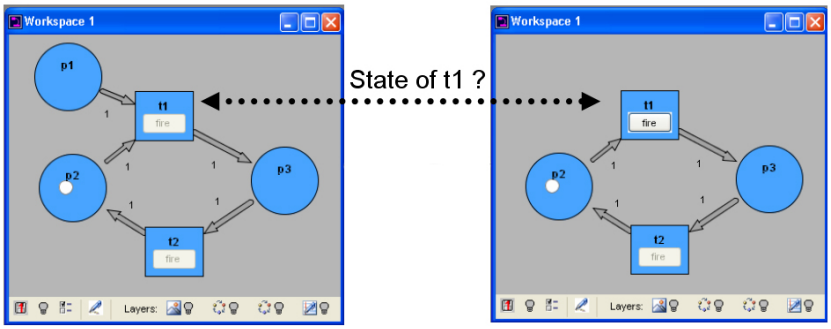

Fig. 1. The problems of partial synchronization

fact that the semantic mapping function is, in general, not context-free. It is not only the general graph semantics that varies, but also that of single nodes which are contained in both of the partially shared models. Figure 1 illustrates this problem with the example domain of Petri Nets. The two workspaces are partially synchronized and differ only in the presence of one single place (p1) and its connection to the transition t1. This causes the semantics of t1 to change, and in particular also affects the semantics of the whole graph: the left net is dead, whereas the right one is non-terminating.

Any general attempt to retain a common semantics between only partially shared (and therefore non-identical) models has to face one problem: either one single global semantics is preserved in the system (and the result is a mismatch between local representation and global semantics), or the semantics is only related to the respective local models. In the latter case, the problem is (as shown in figure 1) the non-existence of a common result.

One possible strategy to deal with this problem is to restrict the degree of flexibility concerning sharing entities. If the semantic mapping of a node does not depend on other entities, then it is reasonable to allow this node to be coupled independently of any other elements in the model graph. Otherwise, the (recursively determined) set of needed model elements has to be included in the set of shared elements:

Definition 4. *Let G=(N,E) be a visual typed graph with a semantics Ip(G), and let $n \in N$ be a node of G. If* n *has an associated semantic value (i.e., the semantic mapping $Ip(n_G)$ of* n *in* G *is defined), then a* synchronization context of n in G, *denoted by $Sync(n_G)$, is a subgraph of G containing n so that $Ip(n_G) = Ip(n_{Sync(n_G)})$. A function $S : N \mapsto \mathcal{P}(G)$ so that each node is mapped to a corresponding synchronization context is called* synchronization context mapping. *A function $\mathcal{S} : N \times G \mapsto \mathcal{P}(G)$ which accepts a node and a graph (containing that node) as input and returns a subgraph which is a synchronization context of the node in the graph is called a* generic synchronization context mapping.

Definition 5. *A synchronization context $Sync(n_G)$ is called* minimal *if no real subgraph of $Sync(n_G)$ fulfills the synchronization context condition for n in G.*

Proposition 1. *Let G=(N,E) be a visual typed graph with a semantics $Ip(G)$, and let $n \in N$ be a node of G with defined semantic mapping $Ip(n_G)$. Then a minimal synchronization context of n in G exists but is, in general, not unique.*

Proof. A trivial synchronization context of n in G is obviously G itself, so that the existence is shown. The fact that $Sync(n_G)$ is in general not unique can be shown with a counterexample: a calculation tree consisting of the root node n_1 of type "×", and three child nodes n_2, n_3, n_4 of n_1 that are all of type "number" with $Ip(n_2) = Ip(n_3) = 0$ and $Ip(n_4) = 1$. Here, two different minimal synchronization contexts of n_1 in G are spanned by the node sets $N_1 = \{n_1, n_2\}$ and $N_2 = \{n_1, n_3\}$.

The proof of proposition 1 shows that the minimal synchronization context of a node in a graph can depend on the values of semantic attributes. This means that upon a change in semantics (e.g., caused by a model simulation step), the minimal synchronization context may change. Using synchronization contexts as foundations for partially coupled models, this has to be taken into account: in collaborative work contexts, a non-minimal but stable synchronization context may be superior to a minimal but frequently changing one.

For a number of modeling languages, minimal synchronization contexts can be defined easily, as the following example illustrates for the case of Petri Nets:

Example 1. Petri Nets are visual typed graphs that have the node type set $\mathcal{N} = \{place, transition\}$. For a visual typed graph G=(N,E), a minimal synchronization context mapping S is as follows (for reasons of simplicity, only the nodes that span the synchronization context graph are given):

$$S(n) := \begin{cases} \{n\} & \text{if type(n) = place} \\ \{n\} \cup \{m \in G : (m,n) \in E \vee (n,m) \in E\} & \text{else} \end{cases}$$

This expresses that places can by synchronized node-wise, whereas the activation state and therefore the semantics of transitions depends on their input and output places, and thus on their complete neighborhood.

A strict consideration of synchronization contexts solves the dilemma between coupling flexibility versus coherence of models. If minimal synchronization contexts are used, the solution is even optimal in the sense that only the "absolutely required" information is shared. Yet, even apart from the dynamics of the minimal synchronization contexts (which may be a serious problem for collaborative work scenarios), one problem remains: there is no generic calculation algorithm for a minimal synchronization contexts. Especially in the case of modeling languages with non-formal semantics (like, e.g., concept maps), it is even unclear what such an algorithm should calculate. The next section of this paper shows an approach to solve at least some of these challenges.

3 Reference Frame Synchronization

Typically, all the concepts presented in the previous section have to be combined in order to express the characteristics of a certain modeling language. E.g., a specific constraint mapping usually belongs to a particular set of node and edge types, and a graph semantics may in turn rely on certain syntactic integrity constraints. For this reason, a central concept that bundles together all the ingredients makes sense. This can be conceived as the formalized abstraction of a visual modeling language:

Definition 6. *Let $\mathcal{N}$ denote a set of node types and $\mathcal{E}$ a set of edge types, and let V_N and V_E be visual node and edge attributes. For a set C of constraint mappings, a semantic domain D, a semantic mapping Ip, and a generic synchronization context mapping S, the tuple $\mathcal{R} = \langle \mathcal{N}, \mathcal{E}, V_N, V_E, C, D, Ip, \mathcal{S} \rangle$ is called a* Reference Frame.

Based on this conceptual notion, a system architecture that allows for dynamically plugging in Reference Frames has been implemented in Java [11]. Details about type definitions, constraint mappings, and model semantics are beyond the scope of this paper - focusing on the group work support, we concentrate on the description of the synchronization contexts in the following:

The synchronization context mapping has an explicit representation in the `ReferenceFrame` interface: a method `synchronizeContext(Node, JGraph)` accepts a node and a visual typed graph as parameters. In conformance with definition 4, the policy for implementations of this method is that it calculates a synchronization context of the parameter node in the parameter graph, and couples the whole context instead of only the node. This way, an attempt of synchronizing a node in a graph with other graph instances can be processed *locally*, and lead to a coherently synchronized subgraph.

An alternative to this, which disburdens the developer from implementing the (non-trivial) method, would be the *automatic* calculation of (in the best case minimal) synchronization contexts. However, such a calculation is practically not reasonable unless restricting the semantics calculation severely. In particular, the following three steps would be required: (1) a way to make explicit all the variables in nodes, edges, graph, and the Reference Frame itself which belong to the semantics, (2) a method to compare these variables with partially synchronized applications, and (3) a technique to build the minimal synchronization context based on the results of the comparison. In particular the third point is the problematic one:

- Straightforward algorithms that simply test which subgraph is a minimal synchronization context are no real option due to their complexity, especially taking into account that (in step 2) this is a distributed algorithm.
- The idea of relying on the comparison results (step 2 in the algorithm) does not work either: even if the nodes and edges with varying semantics are known, the step of determining which elements of the graph must minimally be synchronized in order to "repair" the inconsistency cannot be derived easily due to the (required!) openness of the semantic mapping function.

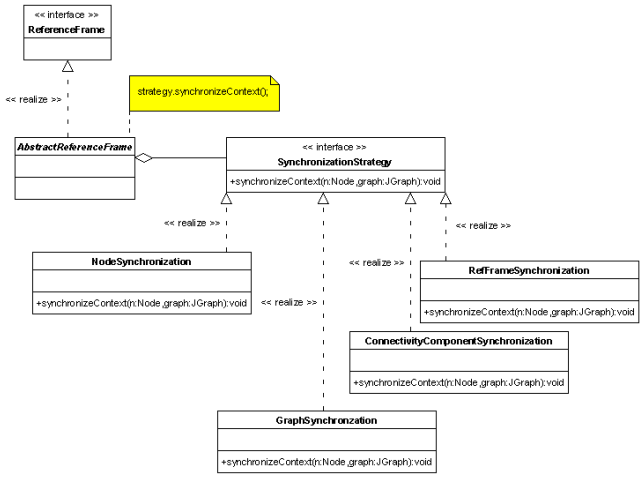

Fig. 2. Synchronization contexts as strategies for Reference Frames

– Finally, algorithms that go into detail about the interdependencies and are able to calculate a suitable synchronization context based on the comparison results are very similar (and not less complex!) than the ones required for implementations of `synchronizeContext(Node, JGraph)`.

In order to assist the developer in the task of defining suitable synchronization contexts, a number of typical algorithms that are applicable for a variety of modeling languages can be pre-defined and implemented in form of a Strategy pattern [4]:

Single Node. This simple implementation does not add any nodes to be additionally synchronized. This algorithm makes sense if the semantics of a node does not depend on the surrounding context in the graph.

Whole Graph. The second trivial case always synchronizes the whole graph upon the attempt to synchronize one node. This strategy guarantees a synchronization context, but obviously in most cases synchronizes too much.

Connectivity Component. In modeling languages where the graph structure plays an important role, the semantics may often be retained if the connectivity component that a node belongs to is synchronized along with the node.

Subgraph induced by Reference Frame. Typically, for "closed" (non-interoperable) modeling languages the subgraph of the graph which consists only of types known by the Reference Frame is a good candidate for a synchronization context.

4 Conclusions and Outlook

This paper introduced a means for adding a degree of flexibility to shared workspace systems that make use of graphs as primary representations: us-

ing synchronization contexts, also semantically rich structures can be partially shared - the method ensures that the joint parts in all the involved applications have the same "local interpretation". Current versions of the Cool Modes software [11] support the synchronization contexts as presented in this paper.

Ongoing research deals with the question if the minimal synchronization contexts, though formally providing shared semantically rich artifacts for collaborative work, are sufficient in the sense of producing a shared *understanding* in the involved user group - or if, on the other hand, there are certain situations in which a full model sharing is more suitable than partial sharing with even well-designed synchronization contexts and appropriate awareness mechanisms.

References

1. G. Booch, I. Jacobson, and J. Rumbaugh. *The Unified Modeling Language User Guide*. Addison Wesley Professional, Boston, MA (USA), 1998.
2. P. P.-S. Chen. The entity-relationship model - toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, 1976.
3. J. W. Forrester. *Principles of Systems*. Pegasus Communications, Waltham, MA (USA), 1968.
4. E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns. Elements of reusable Object-Oriented Software.* Addison-Wesley Professional, Boston, MA (USA), 1995.
5. D. Harel and B. Rumpe. Meaningful modeling: What's the semantics of "semantics"? *Computer*, 37(10):64–72, 2004.
6. D. H. Jonassen. *Computers as Mindtools for Schools*. Prentice Hall, Upper Saddle River, NJ (USA), 2000.
7. M. Kuhn, H. U. Hoppe, A. Lingnau, and M. Fendrich. Evaluation of exploratory approaches in learning probability based on computational modelling and simulation. In *Proceedings of the IADIS conference of Cognition and Exploratory Learning in Digital Age (CELDA)*, pages 83–90, Lisbon, Portugal, 2004. IADIS Press.
8. S. Löhner, W. R. van Joolingen, and E. R. Savelsbergh. The effect of external representation on constructing computer models of complex phenomena. *Instructional Science*, 31:395–418, 2003.
9. J. D. Novak and D. B. Gowin. *Learning How to Learn*. Cambridge University Press, Cambridge, England, 1984.
10. C. A. Petri. *Kommunikation mit Automaten (communication with automata)*. Schriften des Rheinisch-Westfälischen Instituts für Instrumentelle Mathematik, Bonn, Germany, 1962.
11. N. Pinkwart. A plug-in architecture for graph based collaborative modeling systems. In *Proceedings of the 11th International Conference on Artificial Intelligence in Education (AI-ED)*, pages 535–536, Amsterdam, The Netherlands, 2003. IOS Press.
12. D. D. Suthers and C. D. Hundhausen. An experimental study of the effects of representational guidance on collaborative learning processes. *Journal of the Learning Sciences*, 12(2):183–219, 2003.