Electronic Communications of the EASST
Volume 21 (2009)

Proceedings of the
3rd International Workshop on
Multi-Paradigm Modeling
(MPM 2009)

A Pattern-Based Approach to Manage Model References

Juanjuan Jiang, and Tarja Systä

10 Pages

Guest Editors: T. Levendovszky, L. Lengyel, G. Karsai, C. Hardebolle
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

A Pattern-Based Approach to Manage Model References

Juanjuan Jiang and Tarja Systä

Tampere University of Technology
P.O.Box 553, FI-33101 Tampere, Finland

Abstract: Model references play an important role in model integration, especially when models belonging to different domains are to be integrated. They are also needed in various model transformation tasks. In some cases, they need to be instantiated systematically, following certain rules. This calls for an instantiation specification of model references.

In this paper we propose a pattern-based approach for modeling, specifying, and finally applying model references. We represent model references as so-called *collaboration patterns*, modeled as UML collaborations. We further describe the instantiation rules of collaboration patterns. A tool has been implemented for establishing the model references according to the specification, allowing the designer to assist in the process of semi-automated model reference instantiation. We demonstrate the usefulness of the approach and tools by applying them in designing Web service orchestrations.

1 Introduction

Reuse of externally provided business assets, e.g. components or data, is common practice in software development. This can be realized by invoking exposed interfaces or directly operating external data. Integration of software components can, however, be challenging. Therefore, support for software integration at both code and design level is desirable. In addition, tools to guide the software engineer in this task are needed.

Modeling the contents of a software component at different abstraction layers is relevant for several reasons. For instance, it can be used for guiding the development, especially in model-driven software development approaches, for documentation, and for serving as a validation criteria for the implementations.

For integrating different models, possibly belonging to different application domains, model references [Gre04] that link the models according to the integration requirements are needed. Such a model reference can be specified e.g. by an UML association. For ensuring that the model references are instantiated in a desired way, the instantiation sometimes needs to be done according to a certain order.

For example, let us consider a problem of designing a conference arrangement system. For cases where a conference is organized in a hotel, the conference system needs to communicate with the hotel management system. For instance, the organizer of a conference wants to reserve rooms for meetings in a hotel. The conference may consist of several workshops, which will take place in the previously reserved rooms. In addition, attendees need to register to the conference and access the meeting rooms to give a presentation. In this scenario, we can identify the following requirements for instantiation orders:

1. A workshop takes place in the reserved rooms and
2. Every attendee should register to workshop(s) before they give presentations and thus access a meeting room reserved for the workshop.

In this paper we propose a method and tool support for managing references among a set of models, possibly belonging to different domains. The support covers both specifying the model references as well as managing their instantiation orders. This is achieved by indicating model references using UML's CollaborationUses. We propose

1. A pattern, called *Collaboration pattern*, modeled as a UML Collaboration, to characterize model reference between different entities.
2. Use of UML's CollaborationUses to represent applications of collaboration patterns.
3. A prototype tool that guides the designer in model reference instantiation based on the usage of collaboration patterns. The instantiation is carried out semi-automatically.

The proposed approach and tool support can be used e.g. for model integration purposes. In this paper we demonstrate how the approach and tools are used to give semi-automated support for composing Web service orchestrations that integrate a set of Web service models.

2 Our approach

2.1. Information Reference and Model References

Before introducing the details of our approach, we give a definition for information references and model references. Let C be a set of UML classes and O a set of UML objects. Information reference IR, indicating a data reference to another class or object, is defined as a tuple:

$$IR = \begin{Bmatrix} < \mathrm{c1}, \mathrm{c2} >, \mathrm{c1}, \mathrm{c2} \in C, c1 \neq c2 \\ < o1, o2 >, \mathrm{o1}, \mathrm{o2} \in \mathrm{O}, \mathrm{o1} \neq \mathrm{o2} \end{Bmatrix}$$

In the above definition, <c1, c2> indicates a directed reference from class c1 to class c2, while <o1,o2> indicates a directed reference from object o1 to object o2. A model reference is an information reference that crosses model boundaries. It thus connects elements belonging to different models.

2.2. UML Collaboration and CollaborationUse

UML Collaboration describes a structure of collaborating elements (roles), performing a specialized function, which collectively accomplishes the desired functionality [OMG09]. Each role is linked to others by connectors. Collaboration is often used as a means for specifying a pattern. CollaborationUse, in turn, represents application of the pattern described by a Collaboration.

2.3. Collaboration pattern - A pattern for model references

We propose a specific pattern, called *collaboration pattern* to identify a model reference between two model entities. Collaboration patterns can also be used to identify information references inside one model. A collaboration pattern owns two roles, a Consumer role and a

Provider role, denoted by Rconsumer and Rprovider, respectively. A collaboration pattern CollP is defined by the following tuple:
CollP = <$R_{consumer}$, $R_{provider}$, <$R_{consumer}$, $R_{provider}$>>,
where <$R_{consumer,}$ $R_{provider}$> indicates a model reference between a Consumer role and a Provider role.

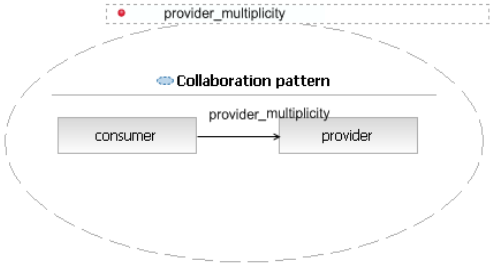

Figure 1. A collaboration pattern

According to the definition, a collaboration pattern is presented as a UML Collaboration and rendered as a dashed ellipse icon, as shown in Figure 1, where Consumer and Provider roles are depicted as rectangles. A directed association connecting the two roles indicates an model reference, namely, a Consumer role refers to (a) Provider roles(s) for retrieving information. A template parameter called provider_multiplicity identifies the number of providers the consumer refers to. For example, provider_multiplicity assigned to one means that a consumer only refers to information from one provider in each instantiation.

2.4. Applying collaboration patterns

Applying a collaboration pattern is a procedure of binding Consumer and Provider roles to classes, or to the roles in another collaboration pattern, which has been already applied. Consequently, an entity bound with a Consumer role can access the entity bounded with a Provider role through the link defined in the collaboration pattern.

The operation of applying a collaboration pattern is defined as follows. Let C be the finite set of classes the models contain, RO the set of roles that have been bound, and Rcp be the set of roles contained by a collaboration pattern to be applied. Then the binding operation B of the collaboration pattern is defined as $B(R_{cp}) \in C \cup RO \sim R_{cp}$, which prevents bindings to the roles of the pattern itself.

The result of applying a collaboration pattern is represented by UML CollaborationUses with bindings. According to the definition of th binding operation, a CollaborationUse may be linked to a class or to another CollaborationUse by binding its roles to the roles of the other CollaborationUse.

Let us continue with the example conference arrangement system. The hotel system and the conference system can now be integrated by collaborationUses as shown in Figure 2. Each collaborationUse indicates an application of the collaboration pattern, in which the white box presents the Consumer role, the grey one presents the Provider role, and the line between roles indicate an information reference. The bindings of roles are shown by dashed directed lines. Figure 2 shows four information references, namely, *reserve*, *occupy*, *register*, and *access*.

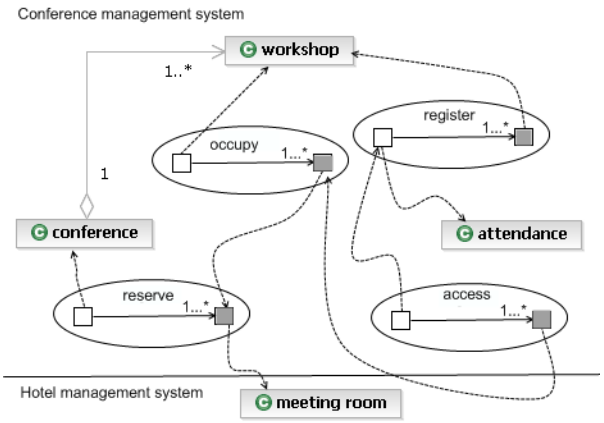

Figure 2. An integrated conference arrangement system, including added CollaborationUses

A binding to a class indicates that all instances of this class would be qualified candidates for the model reference. A binding to a role, say A, however, allows only the instances bound with the role A to be the valid candidates for the model reference. This feature reflects the first instantiation condition. Furthermore, the role that casts the binding depends on the role accepting the binding. The bindings, namely dependencies, among multiple roles of CollaborationUses imply the instantiation order of model references, that is, the second instantiation condition.

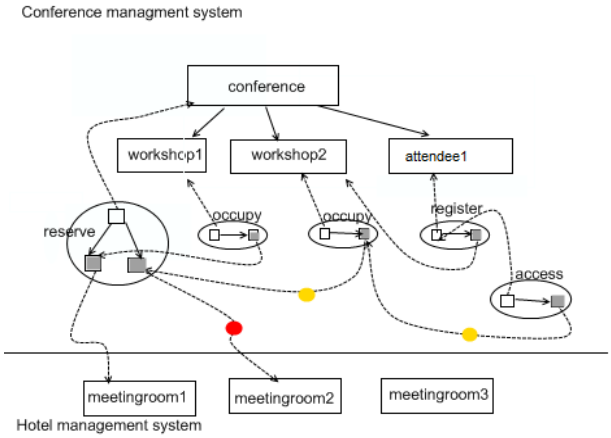

Figure 3. The instance model of the integrated conference arrangement system

Figure 3 illustrates the two instantiation conditions at the instance level. The ellipses show the instances of collaborationUses. They are called *collaborations* hereafter in this paper. Since the provider role of *reserve* collaboration is bound with *meeting room* class in Figure 2, all the meeting rooms are valid candidates for creating the *reserve* collaboration. Let us now

assume that the user has chosen meetingroom1 and meetingroom2, which are reserved for the conference. Again, because the provider role of *occupy* collaboration is bound to the provider role of *reserve* collaboration at the model level as Figure 2 shows, only the objects bound to the provider roles from *reserve* collaboration can be chosen at the instance level. As a result, *workshop* is indirectly linked to either *meetingroom1* or *meetingroom2* via the provider role of *reserve* collaboration, as Figure 3 shows.

By observing the binding relationships of collaborations in Figure 3, we can conclude the following: (1) *meetingroom1* and *meetingroom2* have been reserved for the conference, and (2) the workshops occupy the meeting rooms reserved for the conference. *meetingroom3* is thus not a valid place to accommodate the workshops, since it has not been reserved by the conference. According to the binding relationships marked by yellow dots, the provider role of *access* collaboration is indirectly bound to *meetingroom2*. In other words, *attendee1* who has been registered to *workshop2* can only access the *meetingroom2*. From this example we can notice that the bindings of roles represent the instantiation conditions.

From maintenance point of view, using collaboration patterns instead of associations to manage model references has several benefits. For example, suppose that the binding marked by a red dot (dark grey in black and white printouts) in Figure 3 is broken, and thus the provider role of *reserve* is disconnected from *meetingroom2*. Then the bindings marked by yellow dots (the light grey in black and white printouts), which are directly/indirectly linked to the provider role of *reserve*, lead to the provider roles of *occupy* and *access* and are indirectly disconnected from *meetingroom2* also. Similarly, if the binding marked by a red dot is changed to link with *meetingroom3*, the provider role of *occupy* and *access* are correspondingly indirectly linked to *meetingroom3* as well. In other words, if the meeting rooms reserved for the conference are changed or cancelled, the meeting rooms occupied by the workshops and accessed by the attendees are also changed or cancelled. Assume that information references are described by associations or links. If an information reference is now changed e.g. to cancel one of the reserved meeting rooms, the modelling tool should generate an event to inform all the objects that are directly or indirectly affected, e.g. *workshop2* and *attendee1*. Otherwise, the instance model would become invalid. By using collaboration patterns and with proper tool support, keeping model valid becomes thus much easier.

A collaboration pattern is suitable for specifying and managing the instantiation conditions of model references between two neighboring abstraction layers, e.g. between MOF[OMG09] metamodel layer (M2 layer) and model layer (M1 layer), or between model layer (M1 layer) and instance layer (M0 layer).

3. Tool support

We have implemented a prototype tool called ModelCollabration to support the use of collaboration patterns. ModelCollaboration tool collaborates with and another prototype tool, INARI [Ham04], to utilize its interactive pattern instantiation mechanism. INARI is a generic tool platform for developing task-based model processing applications.

Usage of INARI and ModelCollaration tools proceeds as follows. First, INARI guides the user to instantiate model elements, except model references defined in CollaborationUses. After that, the user chooses a UML element, say A, i.e. an object or a class, created by INARI, and asks ModelCollaboration tool to create an information reference related to it. After

receiving this request, ModelCollaboration tool searches for the information references in the instance model, in which the selected element can play the Consumer role. If such an information reference is found, ModelCollaboration tool continues to search for available UML elements, e.g. objects in the instance model or roles in collaborations, that can play the Provider role in this information reference. If any such UML elements are found, ModelCollaboration tool provides them as suggested candidates for the information reference to the user. After the user has chosen a pair of UML elements, ModelCollaboration tool generates a collaboration between the two elements.

To clearly show an information reference, ModelCollaboration tool usually hides the generated collaboration. Instead it generates an additional link directly between the UML elements. However, these generated links are used for visualization purpose only. Essentially, such links point to the links in the collaboration.

4. Example

Web services aim at making machine-to-machine interaction easy by providing a way to integrate systems loosely and independently from the platforms and programming languages used. A web services system is a composition of a number of standards that are related and linked. Consequently, there are a number of collaborations among them. This example targets at the collaborations of two standards, namely, WSDL [W3C08] and BPEL[OAS08]. Specifically, to generate a BPEL document, information is extracted from the WSDL documents of cooperating Web services in a step by step fashion.

More specifically, an *action* in BPEL needs to use a service interface, so that the *action* can invoke the operations contained in the linked interface. This *action* refers to variables in BPEL, which are required to be typed as the *messages* contained in the linked *operation*. These model references are related. Therefore, we have to instantiate them under certain conditions based on their relationships. Thus, we add several CollaborationUses to present model references between a BPEL model and a WSDL model as shown in Figure 4.

INARI is firstly applied to create the BPEL elements that are partly shown in Figure 5. We have marked BPEL elements by grey colour to distinguish them from the WSDL elements. As we have discussed, a BPEL document needs some information from a WSDL document. For example, the user chooses a class, e.g. *receive*, and asks ModelCollaboration tool to create the instances of information references related to it. ModelCollaboration tool suggests two operations (see Figure 5) as the candidate providers of the model reference *action2operation*. An information reference has been created between *receive* and *partnerLink1* before instantiating *action2operation*. As a result, *receive* has been indirectly linked to the hotel service. Therefore, ModelCollaboration tool only returns the instances of the operation defined in hotel service, namely, *get_availaberoom* and *room_reserver*, to the user.

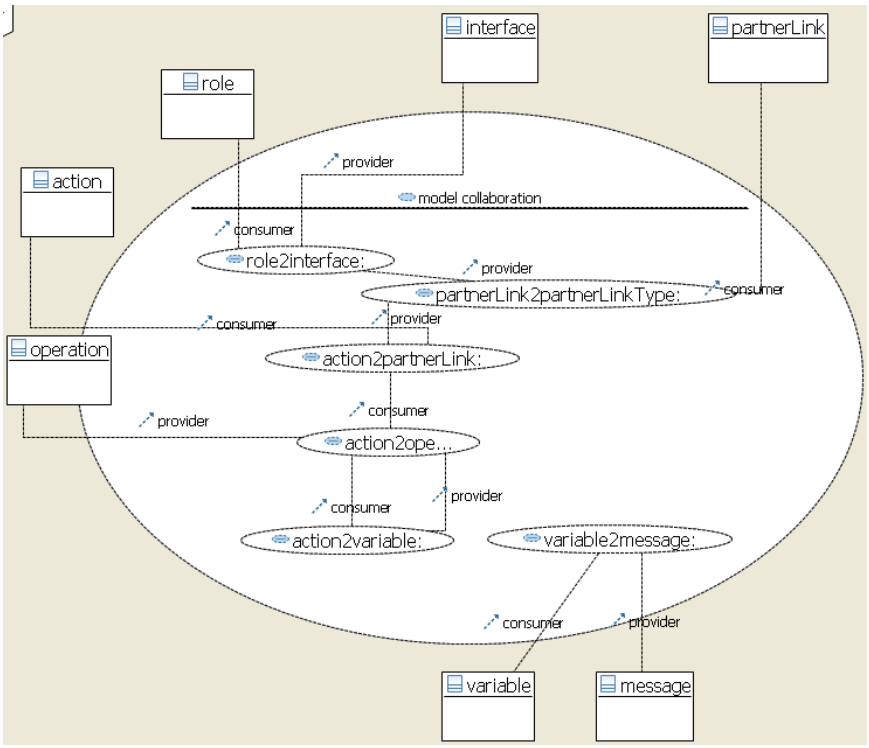

Figure 4. Application of collaboration patterns to integrate BPEL and WSDL models

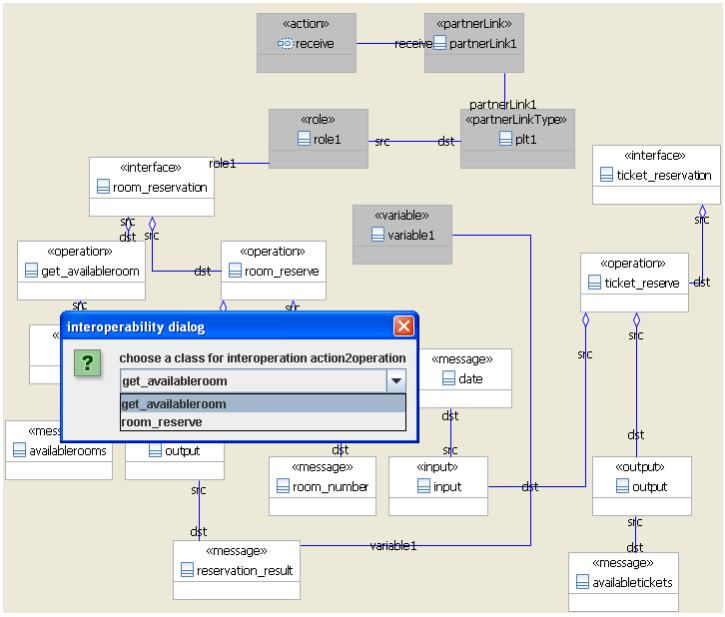

Figure 5. A screenshot of an instantiation of a model reference

In the BPEL design example, Collaboration patterns are applied to link two different metamodels (WSDL and BPEL). ModelCollaboration tool was used during the metamodel instantiation phase to ensure that model references were instantiated correctly based on the instantiation conditions. Further, with the help of ModelCollaboration tool, the user does not need to search the whole models and look for the corresponding entities to instantiate model references. Instead, the entities associated with model references are automatically provided to the user when requested.

5. Related works

The instantiation orders of model references can also be defined by a given a set of OCL constraints. These OCL constraints can be evaluated by an OCL engine, which may provide a true/false feedback or probably an exception report if the evaluation fails. According to the evaluation results, what the user can do is to try different instantiation ways and re-evaluate the instance model whether it is valid based on the OCL constraints. We now argue that even such evaluation results help in instantiation, user still expects more helpful suggestions while doing the instantiation rather than an evaluation report given afterwards. Furthermore, OCL constraints are expressed by text and the instantiation order is indicated implicitly by the evaluation results. From the comprehension point of view, it demands for a visualization to explicitly specify these instantiation conditions of model references.

In our solution, model references are specified by UML collaborations, which are in turn analyzed and serialized into a sequence. In a case where a full specification of interaction orders and other behavioral aspects are needed, using different diagram types, like UML interaction and state diagrams, could be used for. That is, however, not our goal. Instead, we are only interested in assuring the instantiation orders of model references between different models. Also, we do not assume any global identifies, which would be required for matching model elements when different diagram types are to be processed in collaboration. Another important advantage over the use of interaction and state diagrams or OCL constraints is that our approach can preserve model validity in cases where models are changed, e.g. when model references have been removed or changed.

Lahtinen [Lah06] has argued that a tool support is needed to assist the creation of models especially in the case of complex metamodels. Therefore, they propose an approach for task-driven creation of models, assuming that the metamodel has been given as a UML class diagram. In their approach, they describe the rules for instantiating metamodel by a so-called instantiation plan, given in a form of a pattern. Lahtinen's work can similarly be applied for instantiations from the model layer to the instance layer. Comparing with Lahtinen's work, our approach concentrates on instantiating model references that link several models. Lahtinen's work, on the other hand, aims at an overall instantiation of a model. Moreover, our approach gives the instantiation rules of model references at the model level, still using the form of UML, whereas Lahtinen's work describes the instantiation rules in a form of a INARI pattern.

Model weaving is a generic operation that establishes fine-grained links from different models [Fab05]. It can be applied for several application domains, such as model integration. The type of links may vary depending on the relationship of elements in the models explored. A weaving model is not an executable entity. The model operations, e.g. model integration or translation between data sources, are executed by model transformations that use the weaving

model as a specification [Bez03, Arg03]. A transformation rule might include the rules of model references. The guards of transformation rules, which should be satisfied firstly to be able to execute the actual transformation, implicitly indicate the execution orders of rules. According to our experiences, those transformation approaches are powerful but rather complicated in specifying the transformation rules. Learning the transformation mechanism and composing a transformation rule can be a challenging task for a person who is not familiar with model transformations. Supposed that a model reference is the only problem to be solved e.g. in model integration, our approach can be a simple but useful option.

6. Conclusions

Software integration is an essential part in today's software development projects. Quite seldom software systems are built from scratch. Instead, the new systems reuse existing components. On the other hand, model-driven software development has rapidly gained popularity, not only in academic research but also in practice. These trends call for techniques and tools that support model integration. This, however, is not a trivial task. It requires techniques for managing model references between the integrated models.

In this paper we have proposed such an approach. More specifically, the approach and built tool support ensures that the model references are instantiated according to a certain instantiation order. In the proposed approach, UML CollaborationUses, by which the model references are specified, are added to link the different models desired to be integrated. The CollaborationUses might have relations among themselves, which implicitly indicate the instantiation conditions of the model references. We also use breadth-first searching algorithm to discover the relationships of CollaborationUses. We have implemented our approach as an Eclipse plugin, and applied it to BPEL design. The experimental results have shown that the approach and the tool indeed help the user in instantiating and understanding the model references.

Our future plans include applying the approach also for other purposes besides model integration. For instance, a model reference can indicate a correspondence between source and target models in a model transformation or a relationship between different models to be synchronized.

Acknowledgements

This research has been financially supported by TEKES, Nokia Research Center, Nokia Siemens Networks, and Solita. The authors thank Kai Koskimies for his valuable comments.

References

[Gre04] J. Greenfield, K. Short, S. Cook, and S. Kent, *Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools*, John Wiley & Sons, 2004.
[OMG09] Object Management Group (OMG), Unified Modeling Language (UML), http://www.omg.org/technology/documents/formal/uml.htm, 2009.
[Ham04] I. Hammouda, J. Koskinen, M. Pussinen, M. Katara, and T. Mikkonen, Adaptable Concern-based Framework Specialization in UML, In *Proc. of the 19th IEEE*

International Conference on Automated Software Engineering (ASE'04), 2004, pp. 78-87.

[W3C08] World Wide Web Consortium (W3C), Web Service Definition Language, http://www.w3.org/TR/wsdl, Dec.2008.

[OAS08] Organization for the Advancement of Structured Information Standards (OASIS), Business Process Execution Language, http://docs.oasis-open.org/wsbpel/2.0/wsbpel-specification-draft.html, Dec. 2008.

[Lah06] S. Lahtinen, J. Peltonen, I. Hammouda, and K. Koskimies, Guided Model Creation: A Task-Driven Approach, In *Proc. of the IEEE Symposium of Visual Languages and Human-Centric Computing*, 2006, pp. 89-94

[Fab05] M. Didonet Del Fabro, J. Bezivin, F. Jouault, and P. Valduriez, Applying Generic Model Management to Data Mapping, In *Proc. of Base de Donnees Avancees (BDA)*, 2005, pp. 17-20.

[Bez03] J. Bézivin, G. Dupé, F. Jouault, and J. E. Rougui, First experiments with the ATL model transformation language: Transforming XSLT into XQuery, In *Proc. of the OOPSLA'03 Workshop on Generative Techniques in the Context of MDA*, 2003.

[Arg03] A. Agrawal, G. Karsai, and F. Shi, Graph Transformations on Domain-Specific Models, Technical report, ISIS-03-403, Vanderbilt University, 2003.