Making Domain-Specific Models Collaborate

Audrey Occello[1], Olivier Casile[2], Anne-Marie Dery-Pinna[1], Michel Riveill[1]

[1] I3S Laboratory, France
{occello, pinna, riveill}@polytech.unice.fr
[2] IBM, France
casile@fr.ibm.com

Abstract. Many software vendors are pushing Domain-Specific Modeling (DSM) as a way to add business value to their middleware offerings. On the other hand, because the Service Oriented Architecture (SOA) aims at breaking enterprise silos, modern applications tend to be cross-industry and to cover simultaneously different domains. Domain-specific models need then to be composed in order to build domains with much larger scopes. In this paper, we propose a new approach of domain specific models composition, focusing on the specification of how models collaborate with each other.

Keywords. Domain-Specific Modeling, Model Driven Engineering, Model collaboration, Software components, Service Oriented Architecture.

1 Introduction

Model Driven Engineering (MDE) [1] consists of automatically building software from models. Domain-Specific Modeling (DSM) [2] is a particular area of MDE in which models are made of elements representing concepts that are part of a domain or expertise area. DSM languages and code generators allow for working directly with domain concepts and for code generation.

Many software vendors are pushing DSM as a way to add business value to their middleware offerings. IBM, for instance, is adding industry models and industry solutions to its traditional WebSphere offering, which already contains all the key components required for DSM: modeling tools, business process management, master data management, information integration, and content management. The industry models [3] proposed by IBM provide structured and deployable business content for a growing number of industries: Retail, Financial Markets, Banking, Insurance, Healthcare, Telecommunications, etc. As these models consist of integrated data models, processes, service models defined across business requirements, they can be seen as domain-specific models (DSMs) that are quite successful at accelerating the deployment of strategic business initiatives of major Companies.

Moreover, major business drivers such as mergers and acquisitions, competition and partnerships, have favored the increasing adoption of the Service Oriented Architecture (SOA) [4], which is changing the face of software applications. SOA fosters business and IT collaboration and enterprise-wide approaches. Because SOA aims at breaking enterprise silos, modern applications tend to be cross-industry and to cover simultaneously different domains. A direct consequence is that DSMs need to be composed in order to build domains with much larger scopes.

We propose to leverage software components principles [5], [6] in a MDE [1] approach of composition that focuses on the specification of how models collaborate with each other and preserves the identity of models. We believe that this approach is well-suited to DSMs because it preserves each model's domain and tooling. Only collaborations between models need to be handled additionally.

The remainder of this paper is organized as follows. First section 2 presents briefly the principles of the collaborative model approach. Then, section 3 shows how to apply the proposed approach to a practical example. Afterwards, section 4 describes related work. Finally, section 5 summarizes the benefits and limitations of our work.

2 Principles of the Collaborative Component-Based Model Approach

Leveraging principles of software components [5], [6], we propose the Collaborative Component-Based Model approach (CCBM). This approach achieves black-box reuse of models and preserves them: models are units of reuse and incremental composition, just as software components in Component Based Software Engineering (CBSE). This section presents how we assemble models like software components.

In the CCBM approach, composition does not create any new model. Composition is specified as a family of workflows that cross the unmodified input models. Every collaborative model specifies requirements that other models must fulfill in order to participate in a collaboration. Hence, composition is simply a set of models that, taken together, fulfill all the requirements stated by any of them.

2.1 Collaboration Specification

A collaborative model is not described solely by a set of concepts and relationships but also by a set of collaborative operations and collaboration templates (see Figure 1). Collaborative operations represent the points of collaboration of a collaborative model (i.e.: its operations that rely on others models to accomplish a task). Collaborative operations are represented with the "collaborative" stereotype[3]. The intended behavior of Collaborative operations is expressed by collaboration templates that specify an entire family of possible collaborations. The collaboration templates are represented as a variant of UML sequence diagrams. An example of such diagram and how to read it is given in section 3.

The collaboration templates are incomplete and generic specifications of collaborative operations in the sense that:

- they do not need to specify a complete operation body but only mention operations from other models that they need to call,
- they do not need to specify precisely the invoked operations but only the type or model that contains them,
- the names of required models, types or operations mentioned in a template can later be replaced by the names of more 'concrete' entities.

The operations invoked in a collaboration template define implicit constraints for models of collaboration partners. The sum of the constraints expressed in the collaboration templates of a collaborative model implies a set of generic model specifications for collaboration partners that we call model templates. Then, two models are never bound directly together. A collaboration is always defined between a collaborative model and a set of model templates. This property prevents us from having inconsistent "links" between models. As models never describe dependencies towards others existing real models but only through model templates, there is no need to track changes in other domains.

[3] This means that in a UML model operations have a specific stereotype to be used in class diagrams.

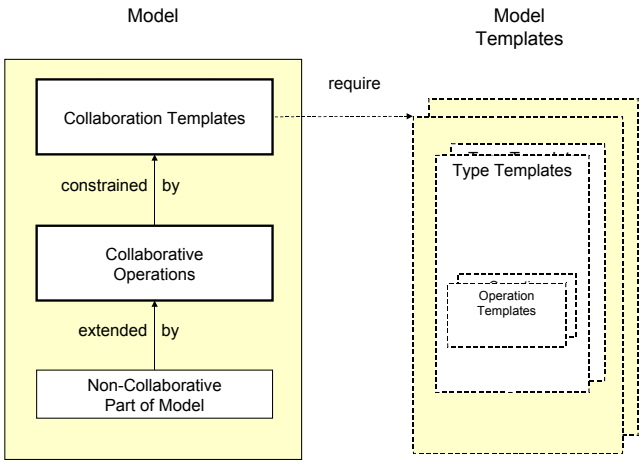

Fig. 1. Expressing collaborations

2.2 Collaboration Refinement and Concretization

To transfer model expertise into concrete applications, it is necessary to express model transformations. Using successive transformations, applications can be derived from models. This section explains how the CCBM approach takes into account the different steps of the MDE development life-cycle: transformations are expressed in CCBM by two high-level standard MDE operations on models: refinement and concretization.

Incompleteness and genericity of collaboration templates allow specifying an entire family of possible collaborations. This opens the door to incremental refinement of such specifications by stating more complete collaboration requirements. Collaborative models can be refined incrementally. One can: 1) add collaborative operations and collaboration templates to the refined model (model extension), 2) complete the collaboration templates by specifying precisely the called methods, not just their containing model or type (operation specialization).

Each refinement specifies model templates that are more specific in the sense that they either provide additional collaboration opportunities or restrict previously available collaborations, specifying them more precisely. Refinements can themselves be further refined specifying more complex or more concrete application families.

The refinement process ends when a collaborative model and its model templates are concretized (concretizing a model means mapping it to an executable form). This step is called concretization. It produces an application that consists of independent components linked by glue code that implements the specified collaborations.

To conserve the independence of each model and their identity at the code level, each model is concretized separately. In the DSM context, this means that each domain-specific model can be concretized using usual code generators. CBSE [5], [6] platforms are good code generator target because each model can be concretized directly as a component. Anyway, others concretizations are possible, including the option to implement all models simply as a set of classes (in Java, C++, etc). The drawback of this choice is that in the resulting application the individual models have no first class representation anymore.

To ensures well-formed collaborations[4] at the code level, each model template must correspond to a component (already coded or generated during the concretization) that fulfills the template by providing equivalent counterparts to the types and operations of the model template. This way the application architecture still reflects the structure of the models from which each component has been derived.

To preserve loose coupling of collaboration partners at the code level, the collaboration templates are concretized as glue code externally from the concretization of models. SOA [4], [7] orchestration languages or coordination languages [8], [9], [10] are well suited. However, collaboration templates can also be concretized using AOSD [11] techniques or as operation calls implementation in the collaborative operation body concretization. But in these cases, collaborations are not separately identifiable any more in the concretization of the models.

The CCBM approach facilitates the modeling of complex applications, including existing pieces of code or reusing existing models. The CCBM approach allows adding or replacing a model in an application built from collaborative models without producing again the entire application from a new composed model. During this process, the application code is conserved; only new collaboration templates and deduced model templates are concretized.

This approach also allows for the reuse of applications developed without MDE [1] since we do not need to have a model representation of an application in order to use it as a collaboration partner. This is important since most of the time industrial products are not built from scratch but implemented on top of older products built from heterogeneous technologies. In the context of DSMs, this property allows reacting quickly to changes in other domains that we rely on simply by replacing one DSM by another one and regenerating corresponding code.

3 Applying the CCBM Approach to a Real World Case Study

To illustrate the CCBM approach, consider the following practical example which was developed by IBM for a Telecommunication Company [12]. The purpose of the case study is to show how to extend the power of telecommunications to other industries. The solution, called "HealthTelco" hereafter, consists in providing individuals with a healthcare home-monitoring service relying on IP Multimedia services.

This solution is based on the reuse of two existing solutions: Telehealthcare and IP Multimedia Subsystem (IMS). Telehealthcare is based on the Healthcare Industry model (HPDM for Health Plan Data Model) that provides disease treatment by early intervention services to monitor patient health at home. IMS is based on the Telecommunication Industry model (TDW for Telecommunication Data Warehouse) that provides services such as Mailing, Voice conferencing, Instant messaging, SMS, etc. Hence, the HealthTelco solution clearly involves an ecosystem of domains with different business models, vocabularies, and technologies: patients, healthcare providers, and telecommunication service providers.

The HealthTelco solution is based on the collaboration between the two existing solutions so that the medical staff can be more efficient and react more quickly to patient alerts. The AlertMonitoring healthcare business process relies on the sendSMS, sendMail, and startConference services of the telco side. The different collaboration needs of the HealthTelco solution (see Figure 2) have naturally led to a SOA implementation: Integration has been done by modifying the orchestration of services of the Telehealthcare solution to add the calls to the IMS

[4] The behavior of the concrete application is an instance of the family of behaviors specified for the collaborative model.

services. Such an implementation could have been produced using the CCBM approach in conjunction with the two industry models and would have permitted to make explicit and thus more manageable the collaborations between the two industry models.

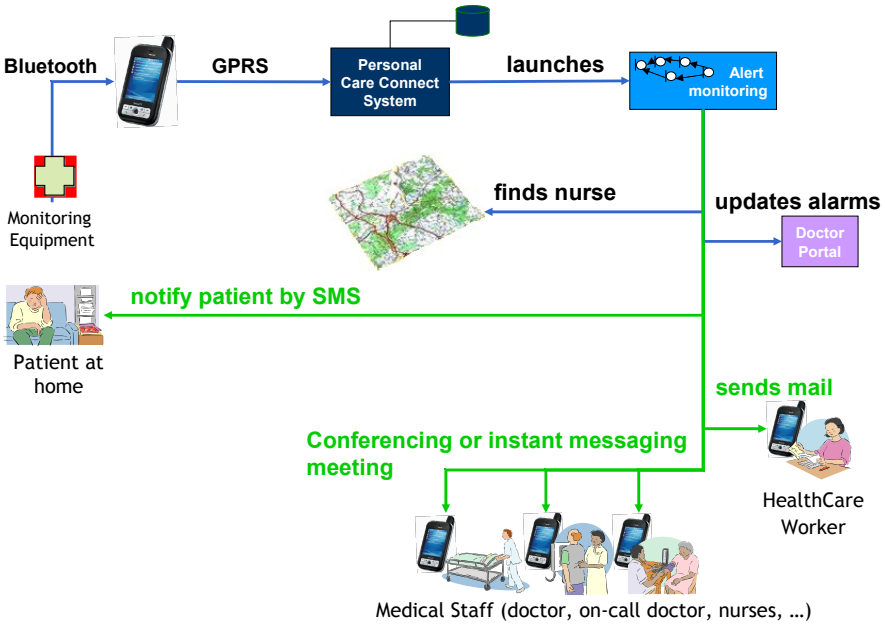

Fig. 2. HealthTelco solution's collaboration points

Let see how to apply the CCBM approach to this case study.

- The first step to describe the collaborations between the two Industry model based solutions consists of identify which operations need to invoke operations of the other solution. In this case, we know that the collaboration is one way: the Telehealthcare solution model need to use the IMS solution model to obtain the HealthTelco solution. The launch operation of the Personal Care Connect System part of the Telehealthcare model is responsible to start the AlertMonitoring business process that contains the collaborations with Telco operations. Then launch has to be modeled as a collaborative operation using the corresponding stereotype.
- The second step consists of expressing for each collaborative operation which other operations it needs from other models for implementing its functionality. In Figure 3, the most left time line represents the launch collaborative operation associated to the currently defined collaboration template. The other time line represents the CommunicationManagement model template. This model template defines implicit constraints for collaboration partners to be used to perform the needed communication features. At concretization time, the CommunicationManagement model template will be replaced by the IP Multimedia Subsystem. Arrows represent operations required by the collaborative operation.

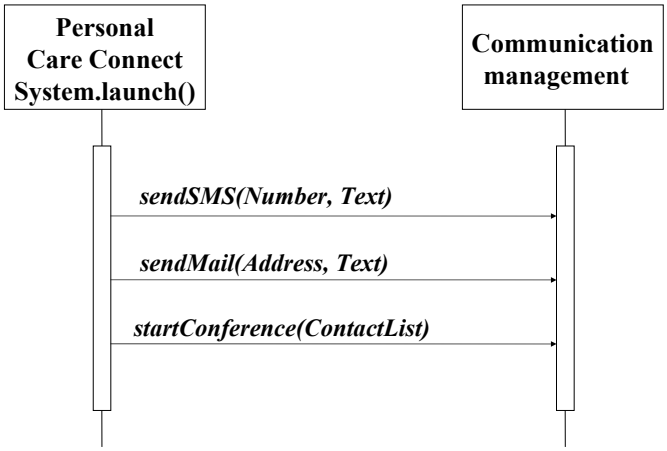

Fig. 3. HealthTelco's collaboration template

The concretization step is not shown because it strictly produces the same implementation of the HealthTelco solution as the existing one: each industry solution model is concretized as an independent system and the collaboration between the two systems (i.e.: the HealthTelco's collaboration template) corresponds to the orchestration of the AlertMonitoring business process. Note that we are able to integrate the Telehealthcare solution with another solution providing such desired communication facilities at any time using the CCBM approach. This other "communication solution" would have to conform the communication model template. If the conformance is fulfilled, the integration of the two solutions only consists in concretizing the collaboration template as new glue code.

4 Related work

Loose coupling at the model is not a novel idea in the MDE [1] area. Aspect oriented modeling or model composition approaches that are based on general-purpose models such as [13], [14], [15], [16], [17], [18], [19] advocate a separation of models representing different concerns and the merging of models from several smaller ones. Because DSM is driven by industry needs such composition by merging does not seem to be well suited for DSM as it does not respect the way Enterprises make business. For instance, the Lines of Business of an enterprise, which are focused on a specific business domain, have no reason to *merge*; they simply need to *collaborate* efficiently, but remaining focused on their core business as shown in section 3. Similarly, two business partners with their own business model will collaborate rather than merge.

Moreover the exploitation of DSM through code generation or interpretation of DSMs is possible because both the language and tooling fit the requirements of only one domain. The CCBM approach works well with these development practices: each model produces code using its own code generator (the CCBM approach does not make modification on such generator) and the collaborations are concretized as glue code between the code of model using specific techniques such as SOA [4] or AOSD [11] as explained in section 2.2. In contrast, merging DSMs means merging domains and also constructing multi-domain code generators then domain specific code generators cannot be reused "as-is".

[20] proposes to combine several DSMs. However, the aim is to split a monolithic model corresponding to a DSL into several partial ones to simplify their management. The approach is limited to interdependent partial models that are contained in the same domain area (even if several DSLs are used) in contrast with the CCBM approach. Moreover, in their approach, references are included directly in DSM specifications and are always by name whereas the concept of collaboration templates in the CCBM approach provides indirect and external bindings and delays the choice of collaboration partners at the concretization time. Then collaboration templates can be reused in several contexts.

Integration can also happen at other levels than in the languages: at code level using middlewares such as component platforms [21], [22], [23], [24] or using aspect oriented techniques [11] for example. Although these approaches allow modelers focus on their specific problem domain, they are not reusable in the sense that the integration is often specific for each component code. By defining collaboration templates, the CCBM approach makes it possible to reuse the integration part.

5 Conclusion

In this paper we have addressed the issue of a synergetic combination of model-driven and component based software engineering. We believe that the CCBM approach is particularly of interest for DSM [2]. This makes it possible to bridge completely different domains and build applications on the union of these domains without having to merge them. The tooling specific to each domain can be reused "as-is" to produce code from models, only the code specific to the integration part need to produced apart as a concretization of collaboration templates. As we follow a loose coupling approach, the concretization of such collaboration as glue code can be easily performed through techniques such as SOA [4] or AOSD [11].

Future work will consist in describing constraints on collaborations so that the concretization of a template is not based only on syntactic constraints. The use of OCL [25] to represent and compare semantic constraints on the models, which is common in large-scale modeling environments, will be a logical evolution of the CCBM approach. Future work will also consist in studying the analysis/verification aspect of the proposed approach. As we used component principles, the CCBM approach can rely on other techniques used in a CBSE context such as architecture description languages (ADLs). We will study how we can reuse and customize analysis work in ADLs [26], [27], [28], [29], [30], [31], [32].

References

1. Schmidt, D.C.: Model-Driven Engineering. IEEE Computer **39** (2006) 25–32
2. Cook, S.: Domain-Specific Modeling. The Architecture Journal (2006)
3. IBM: Ibm industry models. http://www-306.ibm.com/software/data/ips/products/industrymodels/ (2007)
4. Natis, Y.V.: Service-oriented architecture scenario. Gartner, Inc (2003)
5. Szyperski, C.: Component Software: Beyond Object-Oriented Programming. Addison-Westley (1999) *ISBN: 0-201-17888-5*.
6. Heineman, G., Councill, W., eds.: Component-Based Software Engineering, Putting the Pieces Together. Addison-Westley (2001) *ISBN: 0-201-70485-4*.
7. Oasis: Web services business process execution language. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel (2005)
8. Berger, L.: Mise en oeuvre des interactions en environnements distribués, compilés et fortement typés: le modèle MICADO. Thèse de doctorat, Université de Nice-Sophia Antipolis (2001)
9. Lumpe, M., Achermann, F., Nierstrasz, O.: A Formal Language for Composition. In Leavens, G., Sitaraman, M., eds.: Foundations of Component Based Systems. Cambridge University Press (2000) 69–90

10. J-C., C.: Corods: A coordination programming system for open distributed systems. In: Langages et modèles à objets LMO'2001. Volume 7 of L'Objet., Le Croisic, France, Hermès (2001) 11–26
11. Filman, R.E., Elrad, T., Clarke, S., Aksit, M.: Aspect-Oriented Software Development. Addison-Wesley Professional (2004) *ISBN-10: 0321219767*.
12. IBM: Ibm demo presentations showcased at 3gsm (ibm/telus healthcare composite service). http://www-03.ibm.com/industries/telecom/doc/content/landingdtw/2043135102.html?g_type=pspot (2007)
13. AOM group: Aspect Oriented Modeling web site. www.aspect-modeling.org (2007)
14. Straw, G., Georg, G., Song, E., Ghosh, S., France, R., Bieman, J.M.: Model composition directives. In Baar, T., Strohmeier, A., Moreira, A., S. J. Mellor, e., eds.: UML 2004 - The Unified Modeling Language. Model Languages and Applications. 7th International Conference. Volume 3273 of LNCS., Lisbon, Portugal, Springer (2004) 84–97
15. Reddy, R., France, R., Ghosh, S., Fleurey, F., Baudry, B.: Model composition - a signature-based approach. In Aldawud, O., Elrad, T., Gray, J., Kienzle, M.K.J., Stein, D., eds.: 7th International Workshop on Aspect-Oriented Modeling. (2005)
16. Clarke, S.: Extending standard uml with model composition semantics. Sci. Comput. Program. **44** (2002) 71–100
17. Brunet, G., Chechik, M., Easterbrook, S., Nejati, S., Niu, N., Sabetzadeh, M.: A manifesto for model merging. In: GaMMa '06: Proceedings of the 2006 international workshop on Global integrated model management, New York, NY, USA, ACM Press (2006) 5–12
18. Pottinger, R., Bernstein, P.: Merging models based on given correspondences. In: Proceedings of 29th International Conference on Very Large Data Bases (VLDB'03), New York, NY, USA (2003) 862–873
19. Bèzivin, J., Bouzitouna, S., Fabro, M.D.D., Gervais, M.P., Jouault, F., Kolovos, D.S., Kurtev, I., Paige, R.F.: A canonical scheme for model composition. In Rensink, A., Warmer, J., eds.: ECMDA-FA 2006. Volume 4066 of LNCS., Springer (2006) 346–360
20. Warmer, J., Kleppe, A.: Building a flexible software factory using partial domain specific models. Technical Report ISBN 951-39-2631-1, University of Jyväskylä (2006)
21. Group, T.O.M.: CORBA 3.0 New Components Chapters. OMG Document ptc/2001-11-03 (2001)
22. Bruneton, E., Coupaye, T., Stefani, J.B.: The fractal component model. http://fractal.objectweb.org/specification/index.html (2004)
23. Plasil, F., Balek, D., Janecek, R.: SOFA/DCUP: Architecture for component trading and dynamic updating. In: Proceedings of ICCDS'98, Annapolis, Maryland, USA (1998)
24. Roman, E., Ambler, S.W., Jewell, T.: Mastering Enterprise Java Beans II and the Java 2 Platform. Enterprise edn. John-Wiley & Sons Inc. (2002)
25. Warmer, J., Kleppe, A.: OCL: The constraint language of the UML. Journal of Object-Oriented Programming (1999)
26. Zelesnik, G.: The unicon language reference manual. Technical report, School of Computer Science, Carnegie Mellon Universtity, Pittsburgh, USA (1996)
27. Garlan, D., Monroe, R., Wile, D.: ACME : An architecture description interchange language. In: Proceedings of CASCON'97, Toronto, Ontario, USA (1997) 169–183
28. Allen, R.: A Formal Approach to Software Architecture. PhD thesis, Carnegie Mellon University (May 1997) CMU Technical Report CMU-CS-97-144.
29. Luckham, D.C., Vera, J.: An event-based architecture definition language. IEEE Transactions on Software Engineering (1995) 717–734
30. Magee, J., Kramer, J.: Dynamic structure in software architectures. In: Proceedings of ACM SIGSOFT'96: Fourth Symposium on the Foundations of Software Engineering (FSE4), San Francisco, CA, USA (1996) 3–14
31. Medvidovic, N., Oreizy, P., Robbins, J.E., Taylor, R.N.: Using object-oriented typing to support architectural design in the C2 style. In: Proceedings of ACM SIGSOFT'96: Fourth Symposium on the Foundations of Software Engineering (FSE4), San Francisco, CA, USA (1996) 24–32
32. Gorlick, M.M., Razouk, R.R.: Using weaves for software construction and analysis. In: Proceedings of the 13th International Conference on Software Engineering (ICSE13), Austin, TX, USA (1991) 23–34