Co-Evolution and Consistency in Workflow-based Applications

Mario Sánchez*[1,2], Jorge Villalobos[1], and Dirk Deridder**[2]

[1] Software Construction Group, Universidad de los Andes, Bogotá, Colombia
{mar-san1,jvillalo}@uniandes.edu.co,
[2] System and Software Engineering Lab, Vrije Universiteit Brussels, Belgium
Dirk.Deridder@vub.ac.be

Abstract. Workflow-based applications are normally used in contexts where evolution is the norm. To address this, we are developing a model-based framework for building workflows, which takes inspiration on the ideas about separation of concerns. In this framework, we have identified co-evolution relationships and consistency problems related to evolution. In this short paper we briefly present our framework, and introduce a classification of co-evolution scenarios and consistency problems found within it. Finally, we discuss a strategy to take advantage of these relationships, and reduce the number of subsequent consistency problems.

1 Introduction

Workflow related technologies are currently used in a great variety of contexts. In particular, they are used for Business Process Management (BPM), that is to model, enact, control, monitor and improve business processes. Workflow-based solutions are frequently used in rapidly changing environments, and they should cope with changes to business rules and new requirements such as interaction with new technologies.

We are currently developing Cumbia, a model-based framework to build workflows with elements that can evolve with ease. Cumbia is based on the ideas of separation of concerns, and thus it has some similarities to projects such as AO4BPEL [1], PADUS [2] and AMFIBIA [3]. Moreover, in Cumbia it is also central the usage of synchronized executable models. These two features lower the coupling between elements in the workflows, and allows them to evolve independently and with less difficulties. Nevertheless, it is possible to identify co-evolution relationships where the change in one concern has a positive or negative influence in the evolution of the other concerns [4].

This paper explores situations where co-evolution appears in Cumbia-based applications, and it discusses problems related to them. In particular, it focuses

* Supported by the VLIR funded CARAMELOS project http://ssel.vub.ac.be/caramelos/ and by Colciencias.
** Funded by the Interuniversity Attraction Poles Programme - Belgian State Belgian Science Policy.

on the issue of keeping horizontal consistency, that is keeping consistency between elements defined at the same level of abstraction [5]. In Cumbia, this means keeping consistency between the metamodels used to represent each concern. A related problem, which we do not address in this paper, is keeping vertical consistency between metamodels and models. In [6, 7] there is a more comprehensive description of this problem, along with possible solutions that may also be applicable to our case. Section 2 presents the main details of Cumbia-based workflow applications and describes a sample scenario. Section 3 presents a classification of changes to Cumbia metamodels, discusses how they can led to consistency problems, and presents a strategy to reduce these problems. Section 4 presents the conclusions.

2 Cumbia

The main idea behind the design of Cumbia is that separating the elements of a workflow application among several concerns has beneficial effects that are greater than the costs of the extra complexity. In particular, these advantages include easier maintenance and evolution due to an increased modularity. To implement this idea, Cumbia defines a set of metamodels, one per concern, which are used to define the necessary models. Each one of these models is executable, and they are run in a synchronized fashion thanks to an event-based communication system and a weaving mechanism [8] (see figure 1). Synchronization between models maintains the global consistency of the workflow process.

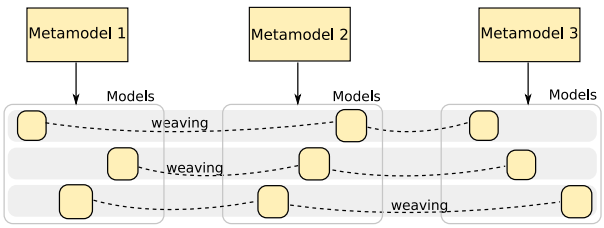

Fig. 1. Structure of models and metamodels in Cumbia

We have designed several metamodels and each one is useful for a specific concern. For instance, in the metamodel called XPM, the elements are used to model the control concern of a workflow process; in another metamodel, called XTM, the elements only define time restrictions. Figure 2 shows a sample process taken from the context of financial services, where the control and time concerns are present. This process defines a sequence of steps to study and approve a credit request, and a couple of time restrictions associated to these steps. The process is initiated when a customer applies for credit; then, it requires an in-depth study of the submitted request and of the financial history of the customer; finally, someone has to approve or reject the request based on the results

generated by both studies. The time restrictions specify that the study of the credit history should not take longer than 2 days, and that the entire process should be completed before the end of the month.

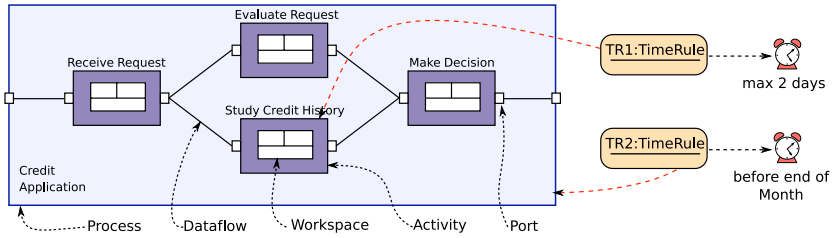

Fig. 2. An XPM process with XTM time restrictions

This particular process shows most elements of XPM and XTM. It is composed by four *activities* that are connected through *ports* and *dataflows*. Each activity has a distinct *workspace* and each workspace executes a specific atomic task; activities enclose workspaces and handle all the synchronization and data management issues. XTM specifies the structure and behavior of *TimeRules*, *Alarms*, and other elements of the metamodel that have been omitted for brevity.

Every metamodel in Cumbia is based on something that we have called *open objects*. The fundamental characteristic of an open object is that it is formed by an *entity*, a *state machine* associated to the entity, and a set of *actions*. An entity is just a traditional object with attributes and methods. It provides an attribute-based state to the open object and its methods implement part of its behavior. The state machine materializes an abstraction of the life-cycle of the entity. It allows other elements to know the state of the open object and react to its changes. Finally, actions are pieces of behavior that are associated to transitions of the state machine. When a transition is taken, its actions are executed in a synchronized way.

The interaction between open objects is based on two mechanisms: event passing and method calling. In the specification of a state machine, each transition has an associated source event: when the open object receives that event, that particular transition must be taken. This mechanism not only serves to synchronize open objects, but also serves to keep the state machine consistent with the internal-state of the entity. Each time the latter is modified, it generates an event that changes the state in the automaton. Finally, other events are generated when transitions are taken and it is thus possible to synchronize open objects using state changes. The other interaction mechanism, method calling, is synchronous and is used to invoke the methods offered by the entities of open objects. These entities can receive method calls from two sources. There can be calls coming from external sources, such as user interaction or other related applications. Additionally, the actions associated to transitions usually invoke

methods of other entities, and thus they play a central role in the coordination of the entire model.

The sample process shows elements from different concerns that need to be coordinated: the time rules depend on the execution of an activity and a process. Since all these elements are open objects, they are coordinated using the two coordination mechanisms described previously. However, in order to synchronize elements from different concerns, it is necessary to describe how they should be coordinated. This requires the usage of a weaving language that defines how specific elements from different models are related. Note that these relations are always specified between models and never between metamodels; this means that metamodels are independent, and that their elements are not aware of the existence of other metamodels. They offer weaving hooks (events, state machines and methods) but the connections are only established between instances that appear in specific models. More details about Cumbia are provided in [8].

3 Co-Evolution and Inconsistencies in Cumbia

One of the advantages of the identification and separation of the various concerns involved in a workflow process, is that they permit the independent evolution of each concern. However, between concerns there are dependencies of different kinds. In particular, the expressiveness of a metamodel depends also on the elements and synchronization hooks offered by the other metamodels. For instance, the time concern would be nearly useless if the control concern would not offer the hooks (methods and events) to time the execution of the activities. Thus, changes in a concern can have both positive and negative effects on related concerns. However, the current definitions of our metamodels do not have explicit descriptions of the dependencies. Thus, it is difficult to detect the inconsistencies introduced by a given evolution step. On the other hand, the explicit definition of such dependencies would create the extra difficulties of having to create, manage, maintain, and evolve these additional descriptions. In this section, we explore scenarios of co-evolution that can surface in Cumbia-based applications and we discuss their associated consistency problems.

3.1 Co-Evolution Scenarios in Cumbia Metamodels

The following classification identifies some possible changes applicable to Cumbia metamodels, and the impact of these changes in the related metamodels. Having such a classification is the first step towards the management and correction of evolution and inconsistency issues. Similar classifications have been proposed in [6, 7]. However, these classifications and analysis are not directly applicable to our context as they rely on specific features of MOF and ECORE based metamodels and models. Open objects and the separation of concerns create a wide range of new scenarios for co-evolution.

New Elements. Usually, adding new elements to a metamodel does not have a

direct effect in related metamodels since not every element needs to be related to elements in other metamodels. However, the addition of elements augments what can be said with the metamodel. Due to the addition of the new element, related metamodels can also evolve to support new relationships. A special situation appears when the new element specializes an old element or concept. In this case, it is possible that the new element may be compatible with the other metamodels. In general, the addition of elements to a metamodel does not create consistency problems nor does it break existing working workflow models. A similar result is reported in [6] with respect to the introduction of classes.

To illustrate this kind of change, we added to XPM an extra element called MultiActivity. A MultiActivity is similar to an Activity, but it can run in several parallel instances of the same workspace. In the sample process, a MultiActivity can be used to replace the activity called Study Credit History and distribute the operation between several employees of the bank. Depending on the implementation of the MultiActivity and of its state machine, it may be necessary to make changes also in XTM to support the application of time restrictions. In our implementation, MultiActivities are externally so similar to Activities that they are undistinguishable for XTM. The trade-off is that time restrictions are only applied to the execution of the whole MultiActivity and not to the execution of the single workspaces.

Remove elements. When an element is removed from a particular metamodel, in others it may be necessary to modify or remove some elements. Thus, all the involved metamodels lose some expressiveness. Only in a few occasions the removal of an element can be innocuous to the other metamodels.

For example, if MultiActivities were removed from XPM, then XTM would not be affected as time rules over activities do not have any specific details to handle MultiActivities. On the contrary, if all activities were removed from XPM, time rules over activities might need to disappear altogether from XTM. The current problem that we are facing is that we still do not have explicit dependencies between time rules and activities as they are part of the semantics of the time rule. Thus, if we remove an element from a metamodel, we should manually detect the possible problems created by this action.

Modification of elements. Modifications to elements of a metamodel can have very different consequences depending on their nature. Because of their diversity, it is necessary to study these changes in detail to discover incompatibilities. Here we analyze two types of modifications that are exclusive to our approach and have co-evolution effects.

Addition of states. If the number of states in a state machine of an open object augments, because of the introduction of new intermediate states, there may be a positive co-evolution effect on related metamodels. As the number of states and transitions grows, the granularity of the coordination hooks becomes finer, and thus the co-evolution effect brings an augmentation in the expressiveness of the weaving language.

To illustrate this, we will use suppose that the state machines of activities in XPM have only transitions from 'Inactive' to 'Active' and then back to 'Inactive'. Within this structure, it is impossible for time restrictions to distinguish the time required to execute the task from the time required to retrieve the data it needs. If the transition between 'Inactive' and 'Active' is refined to include the intermediate state 'Retrieving Data' where data is retrieved, then there is a positive effect in XTM because of the more detailed join points available.

Removal of states. The removal of states has the opposite effect to the addition of states: it reduces the granularity of coordination because there are fewer join points available, thus reducing the expressiveness of the weaving language. This negative effect on the related metamodels can lead to the removal of elements that are rendered useless.

For example, if XTM had time restrictions specifically designed to control the time required to get the data that has to be processed, then the removal of state 'Retrieving Data' from the state machine of the element Activity would lead to the removal of the XTM time restriction because it would be impossible to weave it correctly.

Other changes to the state machine. Other changes to the state machines that change its structure have co-evolution effects that are more difficult to analyze. Similarly, consistency problems are more difficult to detect. However, as long the semantics of the elements does not change much, it is possible that the eventual consistency problems might be fixed only through changes to the weaving.

3.2 Strategy to handle Co-Evolution and Consistency

Co-evolution relationships in Cumbia can have either positive or negative effects, but are rather difficult to identify. Furthermore, the changes to metamodels can have consequences that are difficult to trace. A possible strategy to handle this complexity involves two parts: first, decoupling as much as possible the metamodels, and then reducing the fragility of the join points.

The coordination mechanism offered by the open objects offers a certain degree of decoupling because the event mechanism is only based on names. However, metamodels can still be designed in ways that make them tightly coupled to other metamodels, such as when the elements require too many details of other metamodels to be synchronized. Ideally, a metamodel should be as generic as possible, and its design should depend on concepts of other concerns, rather than on the structure of elements of other metamodels. For instance, the time rules of XTM were designed to be applied to activities, in a large sense of the word; thus, XTM time rules can be applied to XPM activities, but also to BPMN or BPEL activities.

The second step involves the definition of coordination points that are coarser, from the point of view of the weaving language, but that are specialized within each metamodel version. With this improvement it would be possible to establish weavings with coarse events such as the 'Activation' of an activity, and then let the metamodel map the coarse event to a specific state change in a state machine.

4 Conclusions

This paper presented some possible evolution scenarios in Cumbia, and discussed the associated co-evolution effects and consistency problems. The core of this paper was an initial classification of changes related to metamodel evolution that can affect Cumbia based workflow applications. This is an initial classification that will serve as the base for studying how to describe dependencies between concerns, and how to use these dependencies to better handle the co-evolution effects and the likely inconsistencies. Some possible approaches to manage these problems might include the usage of traces [6, 7] to keep track of changes to metamodels, the application of repair actions to solve inconsistencies [9], and the deduction of co-evolution effects [6].

This paper only addressed co-evolution effects between metamodels. Moreover, in Cumbia there are several interesting problems related to vertical consistency that need to be addressed in the future. The main problem here is keeping consistency between the models and their metamodels, taking into account issues such as what to do with running instances. Finally, another kind of consistency related problems have to do with the evolution of models and of the weaving code.

References

1. Charfi, A., Mezini, M.: Aspect-oriented workflow languages. In: On the Move to Meaningful Internet Systems 2006. LNCS, vol. 4275, pp. 183–200. Springer Berlin / Heidelberg (2006)
2. Braem, M. et al.: Isolating process-level concerns using Padus. In: BPM 2006. LNCS, vol. 4102, pp. 113–128. (2006)
3. Axenath, B., Kindler, E., Rubin, V.: AMFIBIA: a meta-model for integrating business process modelling aspects. In: International Journal of Business Process Integration and Management 2007 - vol. 2, no. 2, 120–131 (2007)
4. Deridder, D.: A Concept-Centric Environment for Software Evolution in an Agile Context, Phd Thesis, Vrije Universiteit Brussel (2006)
5. Engels, G., Kster, J., Heckel, R. and Groenewegen, L.: Towards Consistency Preserving Model Evolution In: Proceedings of IWPSE 2002, ACM. (2002).
6. Wachsmuth, G.: Metamodel Adaptation and Model Co-adaptation. In: Proceedings of the 21st European Conference on Object-Oriented Programming (ECOOP'07), Springer-Verlag, 2007, 4069.
7. Gruschko, B., Kolovos, D. and Paige, R.: Towards Synchronizing Models with Evolving Metamodels. In: Proceedings of the Workshop on Model-Driven Software Evolution (MODSE 2007), 2007.
8. Sánchez, M., Villalobos, J.: A flexible architecture to build workflows using aspect-oriented concepts. In: Workshop Aspect Oriented Modeling (Twelfth Edition), Belgium (2008).
9. Nentwich, C., Emmerich, W. and Finkelstein, A.: Consistency Management with Repair Actions. In: Proceedings of the 25th Int. Conf. on Software Engineering (ICSE 2003), May 3-10, Portland, Oregon, USA, IEEE Computer Society, 2003, 455-464.