An Approach to Define and Apply Collaboration Process Patterns for Software Development

Thuan Tan Vo, Bernard Coulette(✉), Hanh Nhi Tran, and Redouane Lbath

Institut de Recherche en Informatique de Toulouse, Toulouse, France
vtthuan89@gmail.com, {bernard.coulette,hanh-nhi.tran,
redouane.lbath}@irit.fr

Abstract. Complex system developments are more and more collaborative. Collaboration strategies largely depend on the development context at modelling, instantiation or enactment time. To put collaboration in action, we propose collaboration process patterns to define, reuse and enact collaborative software development processes. In this paper we describe the definition and application of collaboration patterns. Our patterns, inspired from workflow patterns of Van der Aalst, are described in CMSPEM, a Process Modelling Language developed in our team in 2014. In this paper, we briefly describe the CMSPEM metamodel and focus our presentation on two collaboration patterns: Duplicate in Sequence with Multiple Actors, Duplicate in Parallel with Multiple Actors and Merge. The approach is illustrated by a case study concerning the collaborative and representative process "Review a deliverable".

Keywords: Model driven engineering · Software development process · Collaboration · Collaboration pattern

1 Introduction

Nowadays, software systems are more and more complex, and development processes are usually collaborative. Indeed, these processes are enacted by several actors, possibly deployed on several sites, who work together on collaborative tasks with shared artifacts to achieve a common goal. To facilitate project management and improve the coherence during software process execution, collaboration must be identified, modeled and assisted. Once defined and approved, generic collaboration situations can be reused for further projects [12].

An efficient way to put reuse in action is to define and apply collaboration patterns. Some research works can be found in the literature about collaboration patterns (such as [7, 14, 23]), but very limited work has been done about their automatic application during software development.

In this paper, extension of [25], we propose an approach to define and apply collaboration patterns. More precisely, we describe a set of generic collaboration software patterns and propose a way to apply them automatically during a software development. This work is a continuation of our previous works on process patterns [21] and on collaborative software processes [15–17]. In the first work we proposed a language to

P. Desfray et al. (Eds.): MODELSWARD 2015, CCIS 580, pp. 248–262, 2015.
DOI: 10.1007/978-3-319-27869-8_14

represent process patterns and a mechanism to apply patterns at modeling time. In the second work, we defined the meta-model CMSPEM as an extension of the OMG standard SPEM for describing collaborative software processes. The work described here uses CMSPEM to represent collaboration patterns which are inspired from workflow patterns [22], and proposes mechanisms to apply collaborative patterns not only at modeling but also at instantiation or enactment time.

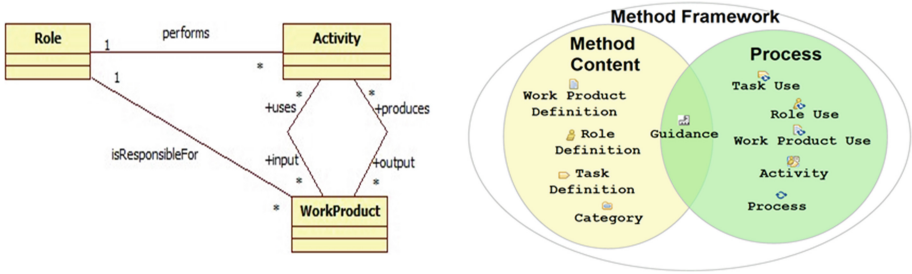
This paper is structured as follows. Section 2 presents the essential concepts of collaborative software process modeling. Section 3 presents a way to represent collaboration patterns. Section 4 shows how collaboration patterns can be applied at modeling, instantiation or enactment time. Section 5 presents a case study and a brief overview of the supporting tool prototype. Section 6 concludes this paper and proposes a short discussion and some perspectives.

2 Modelling Collaborative Processes

Several studies can be found in the literature about notions of process modeling and collaboration. In this section, we put the emphasis on Software process modeling languages, on the notion of collaboration in process enactment, and on the CMSPEM meta-model that was elaborated in our team, and finally on workflow patterns which are reference solutions mainly used in business process modeling.

2.1 Software Process Modelling

A software process is defined as a set of activities for developing, administrating and maintaining a software product [2, 8, 9, 11]. A software process model describes process elements and relationships among them. Process elements can be classified in two categories; primary elements are activities, roles, work products; secondary elements provide additional information on organizational and qualitative aspects of a process. Figure 1a shows the primary process elements and basic relations among them.



(a) Conceptual model of a process

(b) Two views of SPEM

Fig. 1. Key concepts of SPEM 2.0.

Among existing Software Process Modeling Languages (SPML), we decided to put the focus on the OMG standard SPEM 2.0 which is probably the richest modeling language for software process designers, in the sense where it favors reusability and is

open for execution expression [6]. Main primary concepts of SPEM 2.0 are *Role*, *Task* and *WorkProduct* which may have two views: definition and use (Fig. 1b). In the definition view, we find process elements (*Method Content*) which are intended to be reused into several processes; in the use view (*Process*), we find instances of those process elements in the context of real processes. For example, a *TaskDefinition* describes a reusable task whereas a *TaskUse* represents an instance of *TaskDefinition* in a given process.

2.2 Collaboration in Software Process Modeling

A process is said to be collaborative when it contains at least one collaborative activity, i.e. an activity which is performed by two or more human actors targeting the same goal. A collaborative activity is defined as a coordinated and synchronous task whose goal is to build and maintain a shared design of a problem [18, 20]. Collaboration has been largely studied in the literature as shows the review provided by [23]. In [19], the authors propose a classification of collaboration approaches based on prescriptive and descriptive formalisms.

CMSPEM meta-model is a prescriptive process modeling language defined by our team in the context of the GALAXY ANR project [17] whose objective was to propose a framework for supporting collaborative model driven development. CMSPEM is an extension of SPEM which allows defining collaborative software processes. CMSPEM supports both dynamic and static aspects of a process, allowing to enact process models with an appropriate tool. In the following of this section, we briefly present the structural and behavioral views of CMSPEM.

CMSPEM: Structural View. From a structural view, we added in CMSPEM a new package, called *CollaborationStructure*, that introduces the following concepts – *Actor*, *ActorSpecificWork* and *ActorSpecificArtifact* – and a set of related relationships. An *Actor* is a human participant who plays one or several roles in a process. An *ActorSpecificWork* represents the contribution of an *Actor* into a given *TaskUse*. An *ActorSpecificArtifact* represents a copy of a *WorkProductUse* for a given *Actor*.

Figure 2 below shows an extract of the CMSPEM metamodel concerning the *ActorSpecificWork* concept which represents the work performed by a given actor in a collaborative activity. As shown in the figure, a *TaskAssignment* relates an *ActorSpecificWork* to an *Actor*; an *ArtifactUse* relates an *ActorSpecificArtifact* to an *ActorSpecificWork*; an *ActorSpecificWorkRelationship* relates two *ActorSpecificWork*. This latter can be used to represent a precedence order between two *ActorSpecificWork*.

CMSPEM: Behavioral View. The behavior of a process must also be modeled to rigorously specify the process enactment (that may be also called "process execution").

In CMSPEM, we have chosen the state-machine formalism to express this behavior. A state-machine describes states of a given process element (activity or product), and transitions between them. We distinguish two types of transition: manual and automatic. A manual transition – called *OperatorEvent* – is triggered by an actor. An automatic transition is either a *ProcessStateChangeEvent* or a *ConditionalEvent*. Figure 3 shows the kernel of the behavioral part of CMSPEM. Each enactable process element is associated a state-machine.

Fig. 2. Concepts and relationships related to *ActorSpecificWork*.

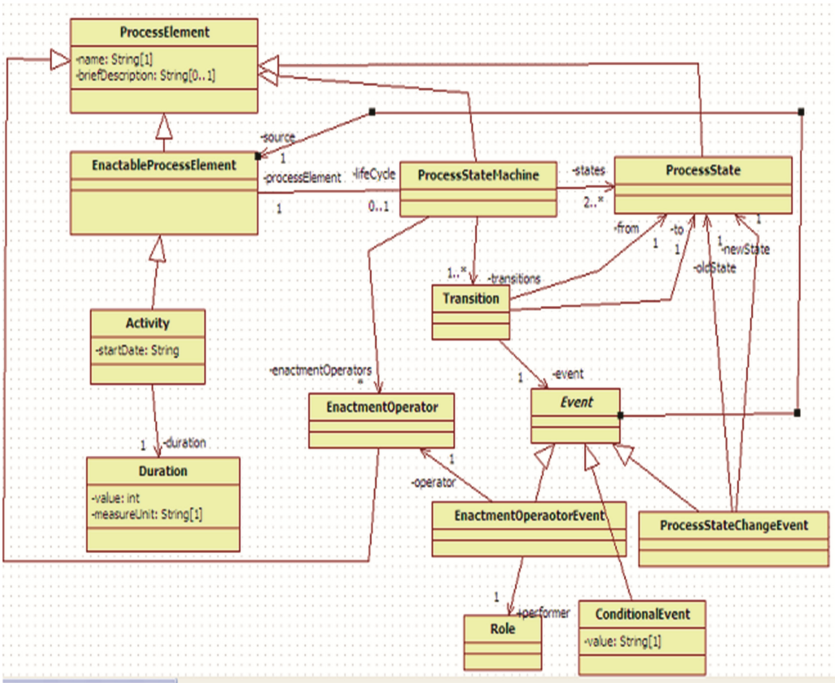


Fig. 3. Behavioral part of CMSPEM meta-model.

Figure 4 illustrates, in a concrete syntax of CMSPEM, a simple example of *"Design"* activity with *"Requirements"* as input, and *"Design Model"* as output. This activity is a collaborative one (represented by a double rectangle) in the usual case where several designers work together to produce the *"Design model"*. Design activity's behavior is described as the state machine shown in Fig. 5.

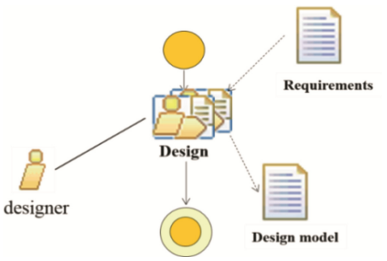


Fig. 4. Collaborative "*Design*" activity expressed in CMSPEM.

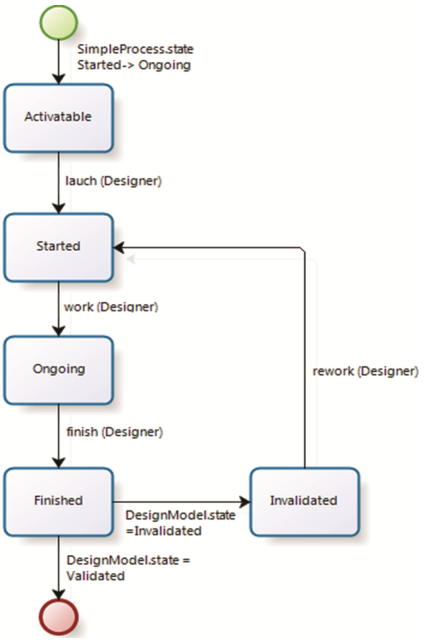


Fig. 5. Behaviour of the Design activity.

The states through which the *Design* activity passes are *Activatable*, *Started*, *Ongoing* and *Finished*. These states are reached by means of *«OperateurEvent»* transitions—launch, work or finish—triggered by a designer. From Finished state, depending on the current state of *DesignModel*, a *«ConditionnalEvent»* transition determines whether the next state will be the terminal state (corresponding to the

fact that *DesignModel* is validated) or the Invalidated state which means that the design is not validated and thus should be reworked.

2.3 Workflow Patterns

Patterns have been highly used for software process modelling [1, 3, 5, 10, 13]. Workflow patterns are reusable generic process fragments which are of high interest for describing collaborative processes. Thus, we studied the workflow patterns proposed in [22] which are reference patterns. It is a set of 42 generic patterns grouped into 8 parts: *Basic Control Flow Patterns*, *Multiple Instance Patterns*, *State-based Patterns*, *Cancellation and Force Completion Patterns*, *Iteration Patterns*, *Termination Patterns*, *Trigger Patterns*. Figure 6 illustrates, as an example, the *Synchonization* pattern which is a basic control flow pattern.

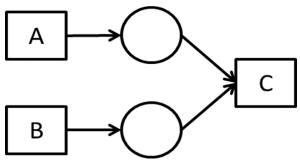


Fig. 6. Synchronization workflow pattern.

3 An Approach to Collaboration Patterns

3.1 Introduction

Collaboration process patterns are development strategies that can be applied either at modeling time or later at instantiation or enactment time. As any pattern, a collaboration pattern can be defined by a recurrent problem, a solution and an application context. We decided to derive in a first time a set of collaboration patterns from elementary workflow patterns that have proven to be efficient in process modeling [21]. Indeed, most of collaboration strategies can be described by means of control flows such as sequence, parallelism, merging, concatenation, etc.

We have defined a set of collaboration patterns that can be found in [24]. In the following of this section, we illustrate two of them that we consider as representative of collaboration strategies: *DuplicateInSequenceWithMultipleActors*, *DuplicateInSequenceWithMultipleActorsAndMerge*. They are described in a graphical syntax associated to CMSPEM. For each pattern, we briefly present below the recurring problem, the application context, and a solution described as an activity diagram.

3.2 Pattern "Duplicate in Sequence with Multiple Actors" (DSMA)

Problem and Context. This pattern represents a collaboration in sequence among actors playing the same role in a given activity. The recurring problem is the one where

human resource is limited in a given enterprise, but constraint time is not too strong. So in this context, it is possible to apply a sequence-based pattern.

Solution. The same activity (cloned) is done by different actors playing a given role. They work in sequence on a product elaborated by another actor. The resulting product becomes the input for the next actor. Figure 7 shows this pattern as an activity diagram in CMSPEM for two abstract actors called *Actor1* and *Actor2*.

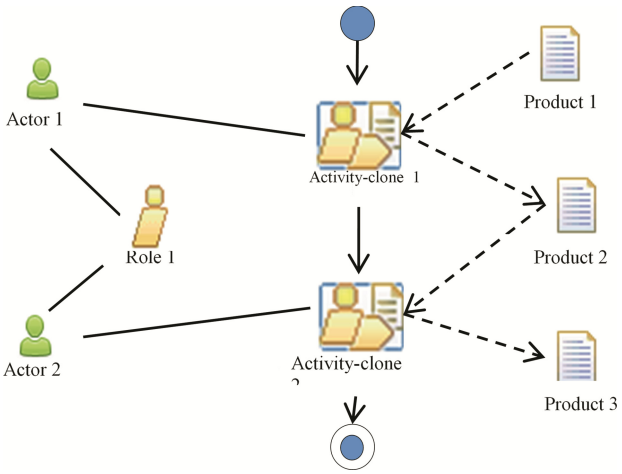


Fig. 7. Pattern *Duplicate In Sequence With Multiple Actors* (DSMA).

It contains abstract cloned activities having one input product and one output product. Each activity is enacted in sequence by different actors playing the same role. For example, this pattern could be used for enacting activities such as *Design a software*, *Review a document*, *Test a program*, etc.

3.3 Pattern "Duplicate in Parallel with Multiple Actors and Merge" (DPMAM)

Problem and Context. This pattern represents a collaborative situation where several actors work on the same cloned activity with the same role. The problem occurs whenever outputs are specific of each actor. In other words, each actor has his own point of view on the activity. This pattern is suitable when several actors are available at the same time, meaning that activities can be enacted in parallel. One of the actors is then in charge of merging the output products into a unique one. This merging activity may be hard to perform in case of conflicting products.

Solution. Cloned activities are enacted in parallel with the same product (cloned) as input. Their termination is synchronized and then followed by a merging activity performed by one of the actors. Figure 8 shows this pattern with two actors working on the same cloned activity, with abstract names.

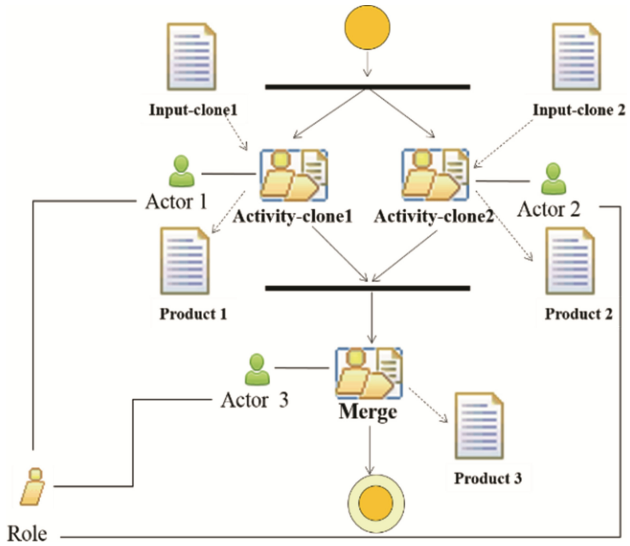


Fig. 8. Pattern *Duplicate in Parallel With Multiple Actors and Merge* (DPMAM).

This pattern could be used for enacting activities such as: *Test a software component*, *Review a deliverable*, *Evaluate a submission*, etc.

This pattern has a specific variant where the *Merge* activity is replaced by a *Concatenate* one. Indeed, the concatenation can be seen as a particular case of merging. This variant may be used whenever *Product1* and *Product2* are disjoint.

3.4 Conclusion

We have put the emphasis in this section on two representative collaboration patterns derived from elementary workflow patterns. However, as discussed in Sect. 6 there are many collaboration strategies so that the set of collaboration process patterns is potentially very big.

4 Application of Collaboration Patterns

Whenever a collaborative activity is identified, one can search for patterns to apply. These patterns are supposed to be stored into a repository. One can note that pattern application can be done at modeling time and/or at instantiation time and/or at enactment time. At modeling time, the application of a collaboration pattern consists in identifying a collaborative activity, choosing a collaboration pattern without instantiating it and refining the activity diagram by unfolding the activity. Unfolding is based on the structural solution (activity diagram) of the pattern which serves as a template. The result of the application of patterns is a refined process model. The choice of the best collaboration pattern to apply is an important issue but it is out of the scope of this paper. To apply such patterns at modeling time, one must know in advance that an activity will be enacted

as a collaborative one. It is not always the case since this information may be known later.

At instantiation time, the goal is to take into consideration the real resources that will be used in a given project, that is to say products in input and output, actors playing a given role on a given activity, etc. For each collaborative activity, one must choose a collaboration pattern to apply, thus identify and instantiate the actors (real persons) that will collaborate, define the products to clone, and unfold the activity as explained above.

At enacting time, the goal is to enact (execute) the process which can be seen as its root activity. Execution of the process must respect the behavioral description of the process (as explained in Sect. 2.2), and in particular the lifecycles that are assigned to process elements. Actors participate in the execution of some manual tasks. It is possible and even necessary to differ the application of collaboration patterns until this enacting time. Indeed, some decisions depend on dynamic information (availability of actors, time constraints, etc.). The principle of the pattern application is the same as for the two previous cases.

5 Realization and Case Study

We describe in this section the case study that we performed and the tool prototype developed as a proof of concept.

5.1 Case Study

We have applied our approach to the process "Review a Deliverable" intensively applied during the ANR Galaxy project (see Fig. 9).

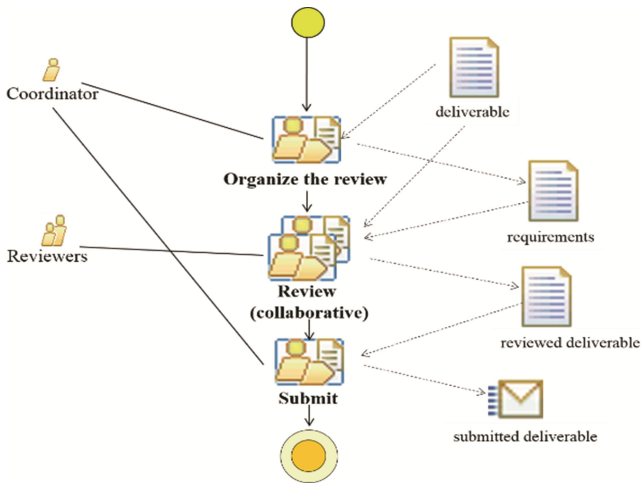


Fig. 9. Process model "Review a deliverable" of the Case Study.

Though it is a quite simple process, it is a real one and it is representative of collaborative processes. This process is made of 3 activities: *Organize the review*, *Review the deliverable*, *Submit the reviewed deliverable*. The second one – *Review the deliverable*, done by several reviewers, is a collaborative one. The reviewing process is organized by a coordinator who specifies requirements to be satisfied by the reviewers. Let us suppose that the collaborative activity *Review the deliverable* is done by 3 reviewers: *Peter*, *Paul* and *Tracy*.

In the following, we present 2 strategies of collaboration corresponding to the application of the 2 collaboration patterns presented in Sect. 3: DSMA, DPMAM. We address the modeling phase, and only consider here the structural solution proposed by collaboration patterns.

Application of Pattern DSMA (Duplicate in Sequence with Multiple Actors). This pattern is applicable in the case where the 3 reviewers can work in sequence one after the other, and whenever there is enough time to achieve the reviewing activity (Fig. 10). It was the case during the Galaxy project.

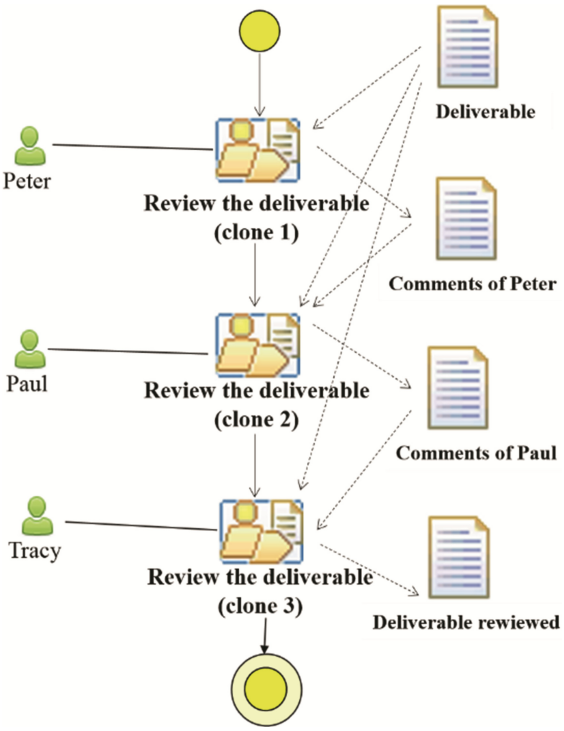


Fig. 10. Activity diagram resulting of DSMA application.

The order in which the reviewers must work is important because the last one finishes the reviewing work. We suppose here that the same input deliverable is in entry of the 3 cloned activities, which means that a reviewer does not update the deliverable. *Peter*

produces comments on the deliverable. *Paul* adds his own comments to those of Peter. *Tracy* produces the reviewed deliverable by analyzing *Paul's* comments.

Application of Pattern DPMAM (Duplicate in Parallel with Multiple Actors and Merge). This pattern is applicable in the case where the reviewers are available at the same time and thus can work in parallel. Figure 11 shows the activity diagram of the pattern's solution. The same deliverable (clone) is in input of each *Review* activity (cloned activity). Each reviewer – that is *Peter*, *Paul* or *Tracy* – produces his proper review by updating the deliverable. *Peter*, is also in charge of the merging activity, whose goal is to merge the results of the 3 reviews included his own. This merging task cannot be automated since conflicts may appear among the reviewed deliverables. It is up to Peter to solve such conflicts in collaboration with the other reviewers.

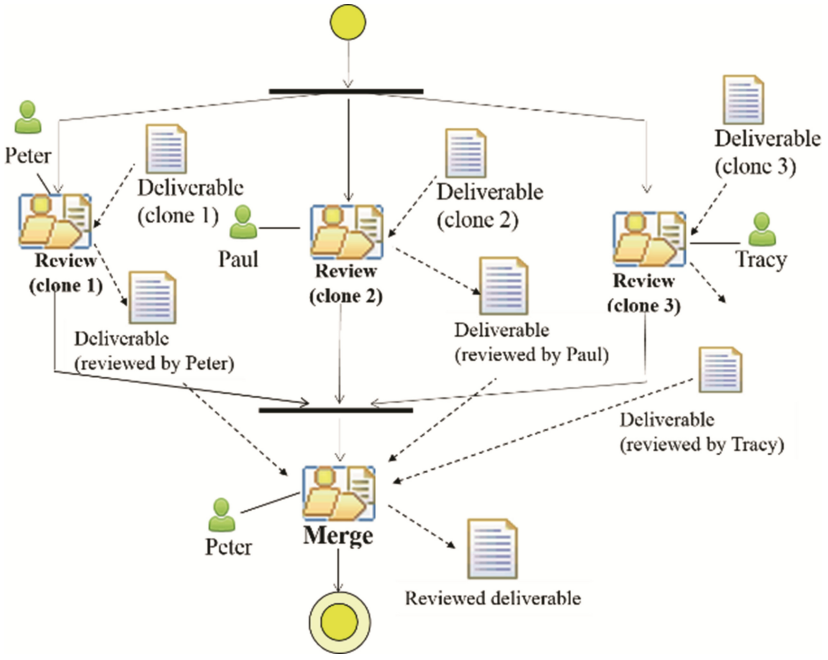


Fig. 11. Activity diagram resulting from DPMAM application.

A variant of this pattern is the following one: each reviewer produces a document containing his comments without modifying the deliverable. In this case, the merger (*Peter*) would have to analyze the three documents produced by the reviewers and to update the deliverable accordingly.

5.2 Supporting Tool Prototype

We have developed a tool prototype for supporting collaborative processes enactment. It is written in Java JEE. To represent a process, we first developed a textual Process

Modeling Language (PML). A process model – described with this PML – is then represented as a tree.

So far, we have implemented the *Duplicate in Sequence with Multiple Actors* pattern (DSMA) described above. Other patterns are being implemented. To illustrate the tool, we have chosen a very simple software process composed of 2 classical activities: *Design* and *Coding*. Figure 12 shows the process model in tree representation.

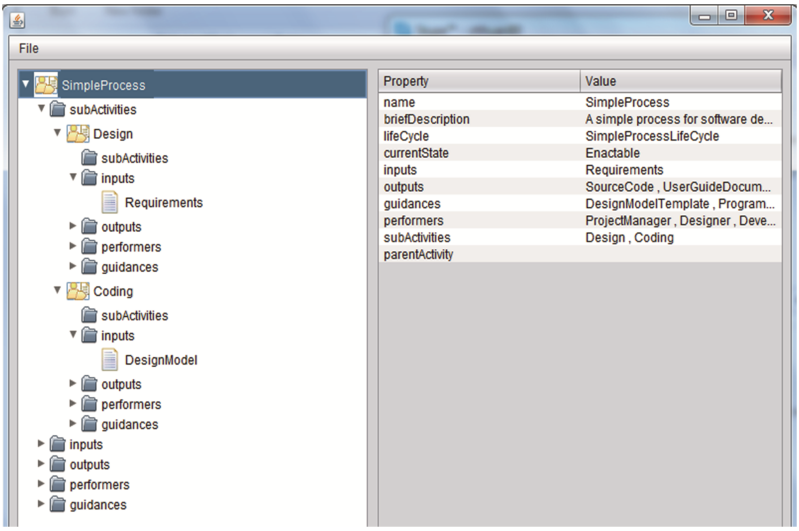


Fig. 12. Example of collaborative simple process.

Enactment of this process is based on the state machines associated to its process elements, including the *Design* activity. At any time of the process enactment, a set of actions is proposed to the current actor depending on the current state.

In the following, we consider the *Design* activity which may be seen as a collaborative one. Let us suppose that this activity is performed in an iterative way by a set of 3 designers. Figures 13 shows the actions proposed to each designer at the beginning of the process; one can notice that only the *launch* action is executable.

To perform the collaborative *Design* activity, one can choose one collaboration pattern in an existing repository, for instance the DSMA pattern. As shown in Fig. 14, three Design activities are performed in sequence in conformity with DSMA's solution. The first one, *Design 1*, done by *Bob*, takes *Requirements* as input and produces *DesignModelBob* as output. This latter product becomes the entry of the second activity, done by Marc, and so one.

It is obvious that this simple process is not a significant case study that would demonstrate the scalability of our approach. However we do not really have any scalability issue with our approach because the number of collaborative activities is always limited in a given process. So the size of the process model is not a significant criterion for the proof of concept.

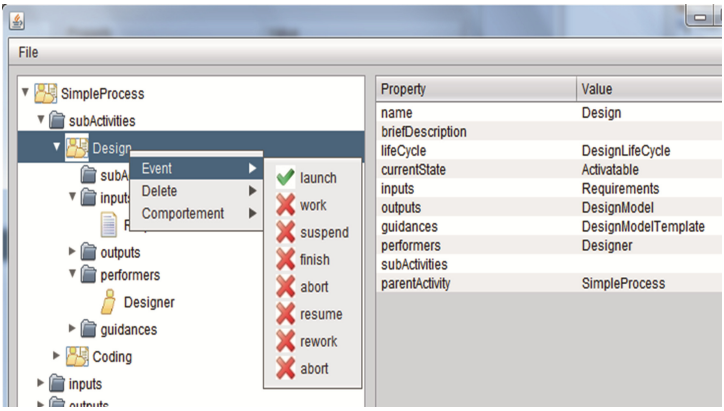


Fig. 13. Interface of the tool: manual action triggering.

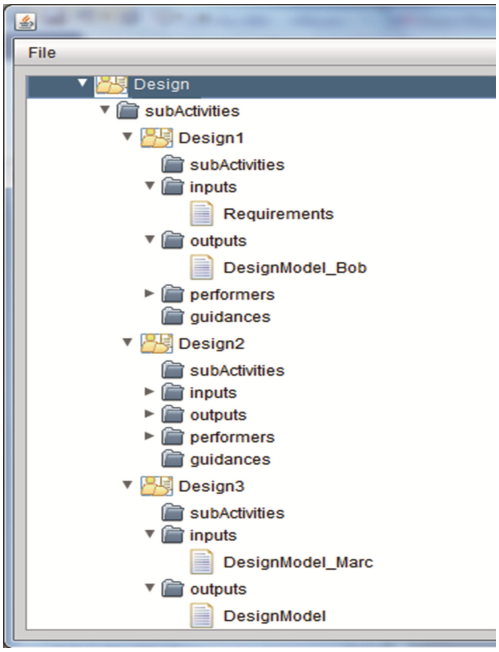


Fig. 14. Process model resulting from DSMA pattern application.

6 Conclusion

We have been working for years on software process modeling and enactment. In the GALAXY ANR project, we defined a new metamodel, extended from SPEM, to model and enact model-based collaborative processes [17]. But putting collaboration in action is very hard and human-dependent, so we decided to provide process designers and

developers with means to perform collaborative tasks. For that sake, we defined a methodology to represent and apply collaboration patterns.

Work Done. In this paper, we have proposed an approach to firstly (1) model collaboration patterns in CMSPEM, and secondly (2) apply them at modeling time (by a process designer) or during software development (by chief projects). So far our process patterns are inspired from workflow ones [22] but our proposition may be applied to any kind of patterns. Our proposition has been validated (as a proof of concept) on a simple but realistic case study. A prototype supporting the approach has been also developed.

Discussion and Perspectives. At short time, our future work will target the supporting tool. First we intend to enrich the set of collaboration patterns available and to organize them through a repository. It is also necessary to improve the tool prototype, to automate the application of collaboration patterns, and to apply our approach to larger collaborative software development processes.

Collaboration is a complex and non-deterministic task which depends on several parameters: availability of human resources, time and budget constraints, etc. Without any guidance for choosing collaboration patterns, the project chief may have big difficulties to do optimal choices. So we are thinking of introducing the notion of collaboration strategy and heuristics for assisting process designer (at modelling time) and project chief (at instantiation and enactment times). The idea is to consider a strategy as a meta-pattern that is a coherent set of collaboration patterns which may be applied in a given context. To represent collaboration strategies, we are working on the definition of a metamodel which is an extension of the one proposed in [4].

References

1. Beck, K., Cunnimgham, W.: Using pattern languages for object-oriented programs. In: Proceedings of OOPSLA 1987 (1987)
2. Benali, K., Derniame, J.C.: Proceedings of the European Workshop on Software Process Technology, Norway (1992)
3. Buschmann, F., Meunier, R., Rohnert, H.: Pattern-Oriented Software Architecture - A System of Patterns. Wiley, New York (1996)
4. Cánovas Izquierdo, J.L., Cabot, J.: Enabling the collaborative definition of DSMLs. In: Salinesi, C., Norrie, M.C., Pastor, Ó. (eds.) CAiSE 2013. LNCS, vol. 7908, pp. 272–287. Springer, Heidelberg (2013)
5. Coad, P., North, D., Mayfield, M.: Object Models – Strategies, Patterns and Application. Yourdon Press Computing Series, Prentice-Hall (1995)
6. Diaw, S., Lbath, R., Coulette, B.: Specification and implementation of SPEM4MDE, a metamodel for MDE software processes. In: SEKE, pp. 646–653. USA Knowledge Systems Institute, Miami (2011)
7. Erikson, T.: Lingua Francas for Design: Sacred Places and Pattern Languages. ACM Press, New York (2000)
8. Feiler, P., Humphrey, W.: Software process development and enactment: concepts and definitions. In: Second International Conference on the Software Process, Continuous Software Process Improvement, pp. 28–40. IEEE (1993)

9. Finkelstein, A., Kramer, J., Nuseibeh, B.: Software Process Modelling and Technology. Wiley, New York (1994)
10. Fowler, M.: Analysis Patterns, Reusable Object Models. Addison-Wesley, Boston (1997). 1997
11. Fuggetta, A., Woft, A.: Software Process. Wiley, New York (1996)
12. Gallardo, J., Bravo, C. Redondo, M.A.: A model-driven development method for collaborative modeling tools. J. Netw. Comput. Appl. Arch. **35**(3), 1086–1105 (2012). ACM.
13. Gamma, E., Helm, R., Johnson, R., et al.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison Wesley, Boston (1994)
14. Herrmann, T., et al.: Concepts for usable patterns of groupware applications. In: Proceedings of the 2003 International ACM SIGGROUP Conference on Supporting Group Work. ACM, Florida, 9–12 November 2003
15. Kedji, K.A., Coulette, B., Nassar, M., Lbath, R., Tran, H.N., Ton That, M.T.: Collaborative processes in the real world: embracing their essential nature (regular paper). In: International Symposium on Model Driven Engineering: Software & Data Integration, Process Based Approaches and Tools - colocated with ECMFA, Birmingham (2011)
16. Kedji, K.A., Ton That, M.T., Coulette, B., Lbath, R., Tran, H.N., Nassar, M.: A tool-supported approach for process modeling: application to collaborative processes. In: 18th Asia Pacific Software Engineering Conference (APSEC), Hochiming City (2011)
17. Kedji, K.A., Lbath, R., Coulette, B., Nassar, M., Barrese, L., Racaru, F.: Supporting collaborative development using process models: a tooled integration-focused approach. J. Softw. Evol. Process (JSEP), February 2014. doi:10.1002/smr.1640, Wiley Online Library
18. Mehra, A., Grundy, J., Hosking, J.: A generic approach to supporting diagram differencing and merging for collaborative design. In: ACM (2005)
19. Poltrock, S., Handel, M.: Modeling collaborative behavior: foundations for collaboration technologies. In: 42nd Hawaii International Conference in System Sciences (2009)
20. Roschelle, J.: The construction of shared knowledge in collaborative problem solving. In: O'Malley, C. (ed.) Computer Supported Collaborative Learning. NATO ASI Series F computer and Systems Sciences, vol. 128. Springer, Heidelberg (1994)
21. Tran, H.N., Coulette, B., Tran, D.T., Vu, M.H.: Automatic reuse of process patterns in process modeling. In: ACM Symposium on Applied Computing (SAC 2011), Taiwan (2011)
22. Van der Aalst, W.: Workflow patterns. http://workflowpatterns.com/
23. Verginadis, Y., Papageorgio, N., Apostolou, D., Mentzas, G.: A review of patterns in collaborative work. In: GROUP 2010, pp. 283–292 (2010)
24. Vo, T.T.: Application de patrons de collaboration lors de la mise en œuvre de procédés collaboratifs. Master thesis, Toulouse, June, 2013
25. Vo, T.T., Coulette B., Tran H.N., Lbath R.: Defining and using collaboration patterns for software development. In: International Workshop on Cooperative Model Driven Development, Colocated with MODELSWARD 2015. SciTe Press, February 2015