Defining and using Collaboration Patterns for Software Process Development

Tan Thuan 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

Keywords: Model Driven Engineering, Software Development Process, Collaboration, Collaboration Pattern.

Abstract: Collaboration patterns are an efficient way to define, reuse and enact collaborative software development processes. We propose an approach to define and apply collaboration patterns at modelling, instantiation or execution time. Our patterns, inspired from workflow patterns, are described in CMSPEM, a Process Modelling Language developed in our team. 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 process "Review a deliverable".

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 on several sites, that 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 should be identified, modeled and assisted. Once defined and approved, generic collaboration situations can be reused for further projects.

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 (Verginadis et al., 2010; Herrmann T., et al. 2003; Erickson, 2000), but very limited work has been done about their automatic application during software development.

In this paper, we describe a set of generic collaboration software patterns and propose a way to apply them automatically. This work is a continuation of our previous works on process patterns (Tran et al., 2011) and on collaborative software processes (Kedji et al., 2011, 2013). In the first work we proposed a language to 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 in this paper uses CMSPEM to represent collaboration patterns which are inspired from workflow patterns (Van der Aalst, website), 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 some perspectives.

2 MODELLING COLLABORATIVE PROCESS

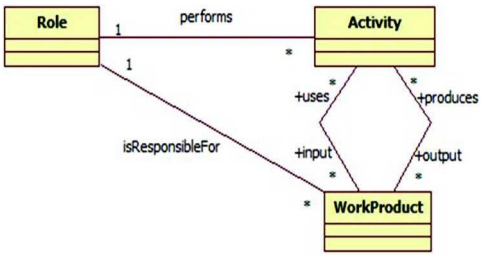
Several studies can be found in the literature about notions of process modeling and collaboration. In this section, we put the emphasis on Software process modeling languages, the notion of collaboration in process enactment, the CMSPEM meta-model that was elaborated in our team, and workflow patterns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

2.1 Software Process Modeling

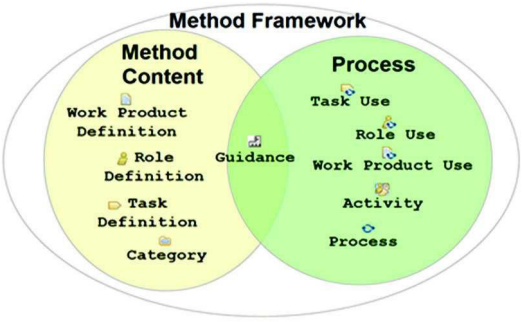
A software process is defined as a set of activities for

developing, administrating and maintaining a software product (Feiler et al, 1992). 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.



(1a) Conceptual model of a process



(1b) The two views of SPEM 2.0

Figure 1: Key concepts of SPEM 2.0.

Among existing software process modeling languages, 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. Main primary concepts of SPEM 2.0 are *Role*, *Task* and *WorkProduct* which may have two views: definition and use (Figure 1b). In the definition view, we will find process elements (*Method Content*) which are intended to be reused in several processes; in the use view (*Process*), we will find instances 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, each collaborative activity being 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 (Roschell et al., 1994). Collaboration has been largely studied in the literature as shows the review provided by (Verginadis et al., 2010). In (Potrock et al., 2009), the authors propose a classification of collaboration approaches based on prescriptive and descriptive formalisms.

CMSPEM meta-model is a prescriptive Process Modeling Language that was defined by our team in the context of the GALAXY ANR project (Kedji et al., 2014) and whose objective was to propose a framework for supporting collaborative model driven developments. 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. In the following of this section, we briefly present the structural and behavioral views of CMSPEM.

2.2.1 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*.

2.2.2 CMSPEM: Behavioral View

The behavior of a process must also be modeled to rigorously specify the process enactment (that may be also called *execution*).

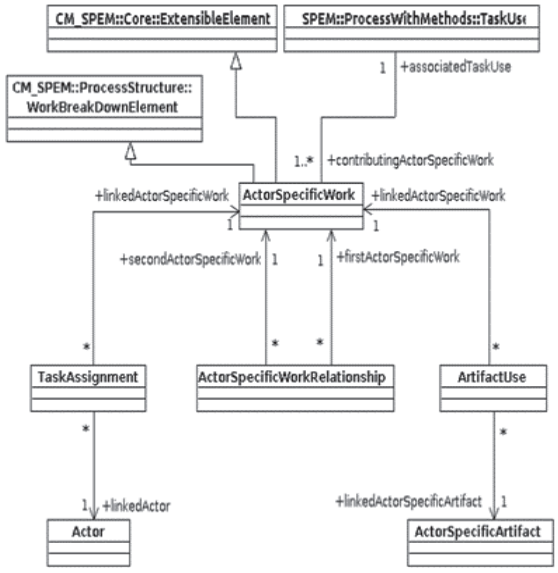


Figure 2: Concepts and relationships related to ActorSpecificWork: extract of CMSPEM metamodel.

In CMSPEM, we have chosen the state-machine formalism to express this behavior. A state-machine describes the states of a given process element (activity or product), and transitions between them. We distinguish two types of transition: manual, 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 lifecycle represented by a state-machine that is composed of states and transitions.

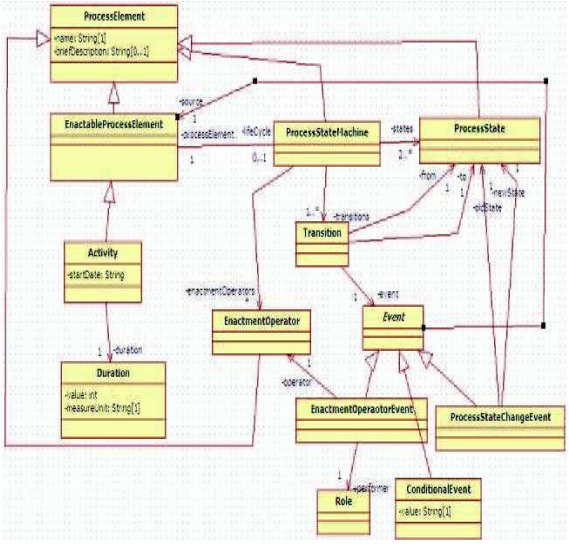


Figure 3: Behavioral part of CMSPEM meta-model.

Figure 4 illustrates, in a concrete syntax associated to 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".

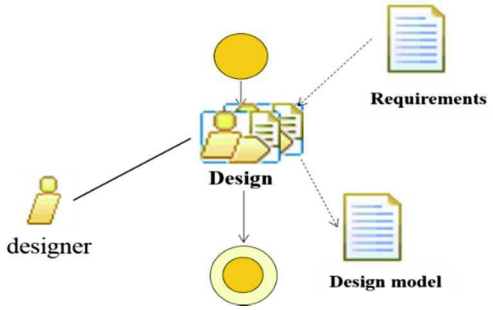


Figure 4: Collaborative "Design" activity expressed in a concrete syntax conform to CMSPEM.

Design activity's behavior is described as the state machine shown in Figure 5. The states through which the 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 *DesignModel* is validated) or the *Invalidated* state which means that the design is not validated and thus should be reworked.

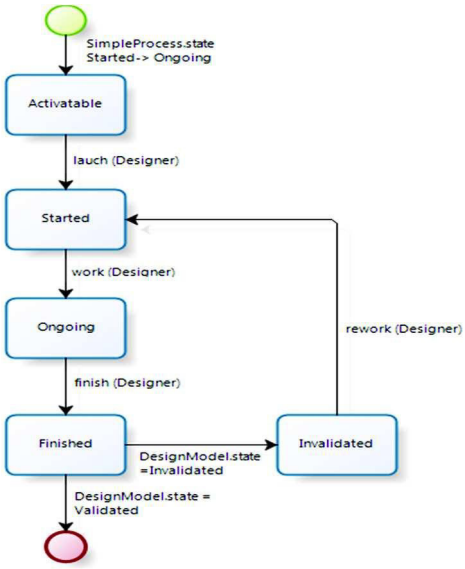


Figure 5: Behaviour of the "Design" activity.

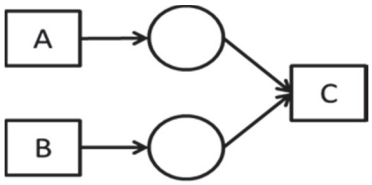


Figure 6: Synchronization workflow pattern.

2.3 Workflow Patterns

Workflow patterns are reusable generic process fragments which are of high interest for describing collaborative processes. Thus, we studied the workflow patterns proposed in (Van der Aalst, website) 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 the *Synchonization* pattern which is a basic control flow pattern.

3 AN APPROACH TO COLLABORATION PATTERNS

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 a set of collaboration patterns from workflow patterns that have proven to be efficient in process modeling. 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 (Vo Tan T., 2013). 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.

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*. 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.

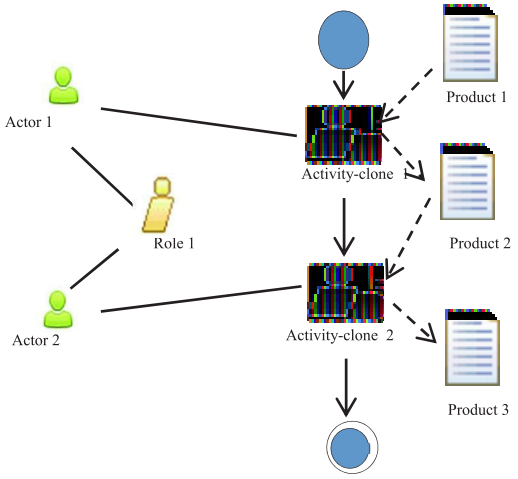


Figure 7: Pattern Duplicate in Sequence with Multiple Actors (DSMA): activity diagram in CMSPEM.

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

Problem and Context: This pattern represents a collaborative situation where 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 in charge of merging the output products into a unique one.

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. Figure 8 shows this pattern in CMSPEM for two actors.

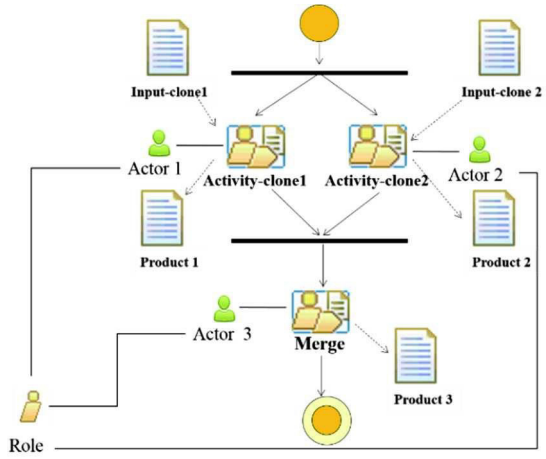


Figure 8: Pattern Duplicate in Parallel with Multiple Actors and Merge (DPMAM): activity diagram in CMSPEM.

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.

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.

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 section 2.1.2), 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.

In the next section, we describe the case study that we performed and the tool prototype developed as a proof of concept.

5 REALIZATION AND CASE STUDY

5.1 Case Study

We have applied our approach to the process "*Review a Deliverable"* performed during the ANR Galaxy project (see Figure 9). 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*, is collaborative, and thus done by several reviewers. 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 a 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 section 3: DSMA, DPMAM. We address the modeling phase, and only consider here the structural solution proposed by collaboration patterns.

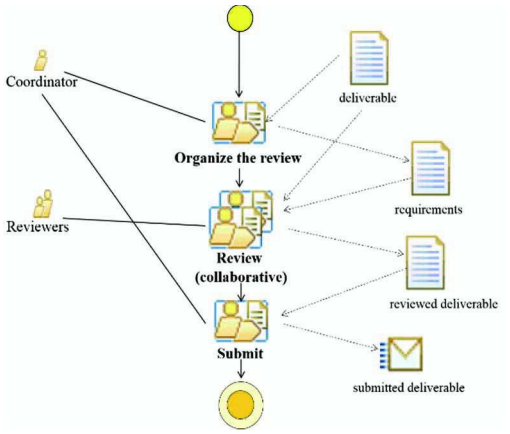


Figure 9: Process model "Review a deliverable" of the case study.

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 (Figure 10). It was the case during the Galaxy project.

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 (cloned activity). Each reviewer – that is Peter, Paul or Tracy – produces his proper review by updating the deliverable. Peter, who plays the *reviewer* role, as the two *others*, is also in charge of the merging activity, whose goal is to merge the results of the 3 reviews included his own.

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 3 documents produced by the reviewers and to update the deliverable accordingly.

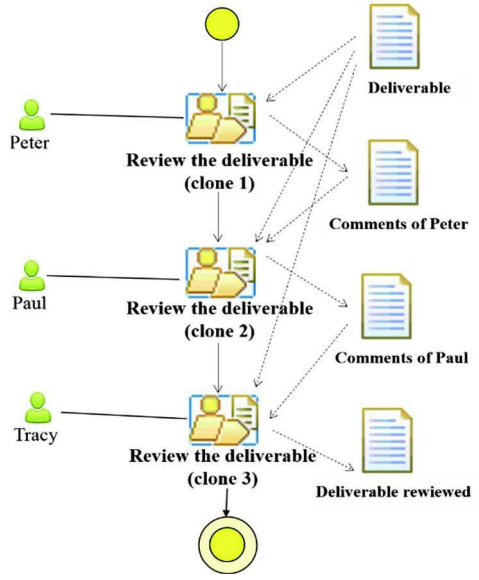


Figure 10: Activity diagram resulting of DSMA application.

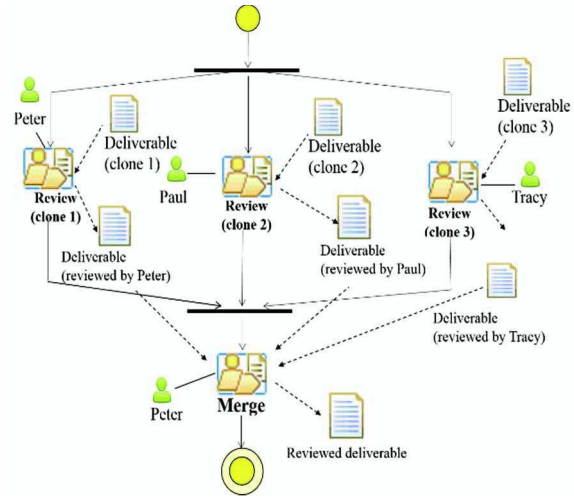


Figure 11: Activity diagram resulting from DPMAM application.

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*. The process model is shown (tree representation) on Figure 12.

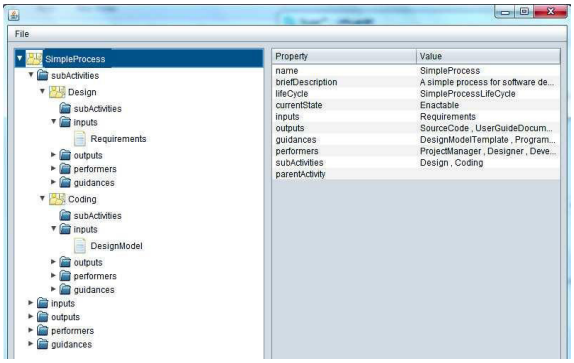


Figure 12: Example of collaborative simple process.

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.

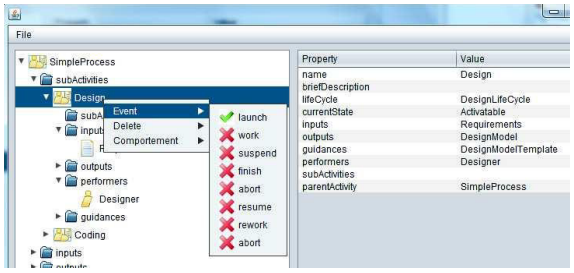


Figure 13: Interface of the tool: manual action triggering.

To perform the collaborative Design activity, one can choose one collaboration pattern in an existing repository, for instance the DSMA pattern. As shown in Figure 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.

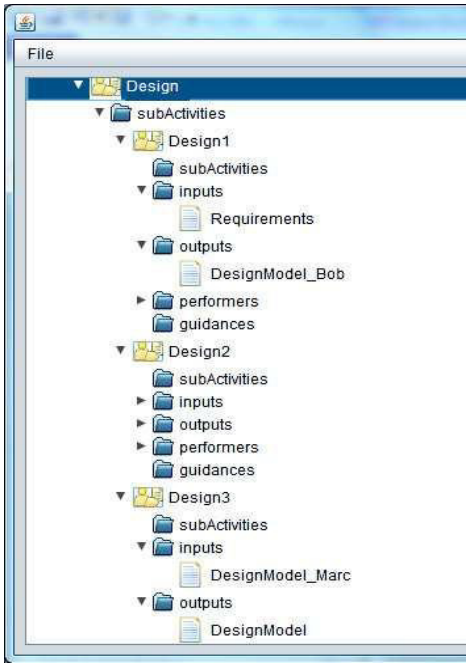


Figure 14: Process model resulting from DSMA pattern application.

6 CONCLUSION

Our work mainly addresses collaborative software process modeling and enactment. For that sake, we decided to define and apply collaboration patterns inspired from workflow patterns whose efficiency has been largely proven.

In this paper, we have proposed an approach to firstly (1) model collaboration patterns in CMSPEM, and secondly (2) apply them during software development. 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. The tool is operational, but other collaboration patterns should be implemented in the prototype.

As main perspectives of this research work, we are considering several topics at short and longer terms. First we intend to enrich the base of collaboration patterns, and to manage them thanks to

a repository. It will be also necessary to improve the tool prototype, and to apply our approach to larger collaborative software development processes.

REFERENCES

Beck, K., Cunnimgham, W.,. "Using pattern languages for object-oriented programs". *s.l.: Proceedings of OOPSLA87, 1987.*

Benali, K., Derniame J. C. Proceedings of the European Workshop on Software Process Technology. 1992, Norway.

Buschmann F., Meunier R., Rohnert. 1996. Pattern-Oriented Software Architecture - A System of Patterns. John Wiley.

Coad P., North D. et Mayfield M. 1995. "Object Models – Strategies, Patterns and Application". *Yourdon Press Computing Series.*

Diaw S., Lbath R., Coulette B. 2011. Specification and Implementation of SPEM4MDE, a metamodel for MDE software processes. In SEKE, Miami - USA Knowledge Systems Institute , p. 646-653.

Erikson. T. Lingua Francas for Design: Sacred Places and Pattern Languages. NewYork : ACM Press, 2000.

Feiler P., Humphrey W. Software Process Development and Enactment: Concepts and Definitions, 1992.

Finkelstein, A., Kramer, J., Nuseibeh, B. Software Process Modelling and Technology, 1994.

Fowler, M. 1997. Analysis Patterns, Reusable Object Models. Addison-Wesley, 1997.

Fuggetta A., Woft A. Software Process.1996. John Wiley & Sons.

Gamma E., Helm R., Johnson R., et al. 1994. Design Patterns: Elements of Reusable Object-Oriented Software. Addison Wesley.

Herrmann T., et al. Concepts for Usable Patterns of Groupware Applications. 2003.

Kedji K.A., Coulette B., Nassar M., Lbath R., Tran H. N., Ton That M. T. 2011 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 2011, Birmingham.*

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.

Kedji, K. A., Lbath R., Coulette B., NASSAR, M., Barrese L., Racaru F. 2014. Supporting collaborative development using process models: a Tooled Integration-focused Approach. *Journal of Software* : Evolution and Process (JSEP). February 2014, Wiley online library. DOI: 10.1002/smr.1640.

Mehra A., Grundy J., and Hosking J. 2005. A generic approach to supporting diagram differencing and merging for collaborative design. ACM.

Poltrock, S., Handel, M. 2009. Modeling collaborative behavior: Foundations for collaboration technologies. I*n 42nd Hawaii International Conference in System Sciences.*

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.

Van der Aalst W. Workflow Patterns. http:// workflowpatterns.com/

Verginadis Y., Papageorgio N., Apostolou D., Mentzas G. 2010. *A review of patterns in collaborative work. GROUP 2010*: 283-292.

Vo Tan T. 2013. Application de patrons de collaboration lors de la mise en œuvre de procédés collaboratifs. Master thesis, Toulouse, June, 2013.