Collaborative Processes Behavioral Modeling

Mamadou Lakhassane Cisse

Student Research Abstract: Collaborative Processes Behavioral Modeling

Mamadou Lakhassane Cisse
IRIT Laboratory
Toulouse
France
mamadou.cisse@irit.fr

PROBLEM AND MOTIVATION

Software engineering is a deeply collaborative activity. Therefore, having tasks executed by more than one actor is common. Such collaborative processes allow development teams to be more efficient. They need to be adequately represented in order to serve as a development model (for example in agile developments). For that, we need a dedicated language allowing to model different collaborative situations.

So far, some attempts to describe a collaborative process via models have been made [1][2]. However, they do not address the topic of tasks and roles organization within the development team neither concepts about the states of each task and action to perform to go from one state to another. In the other hand, SPEM [3], which is the standard for software processes modeling, does not address the execution of any kind of task and still less tasks executed by several persons. Some approaches [4] deal with tasks sequencing but they remain insufficient to precisely describe the reality of software development. They depict a "weak collaboration" in contrast to "strong collaboration" that is how the same task is shared between actors playing the same role. Besides modeling, the question of collaborative processes execution is also hardly addressed, especially to support strong collaboration. For this purpose, we need an executable language to clearly define collaborative situation semantics.

To address these issues, we have proposed to manage collaborative processes by providing (1) a metamodel equipped with constructs to describe multi-instance tasks; (2) a set of collaborative patterns; (3) an operational semantics enabling execute a multi-instance task based on the selected pattern; (4) a process management system supporting flexibility by late binding so that users can choose, at enactment time, appropriate collaborative patterns corresponding to their organizational model. Considering system-of-systems, the underlying interest lays in the fact that our approach can ease collaboration between the tasks enacted in the different interconnect systems. Those systems can also share the same outputs/inputs. In that case, it is necessary to have a mechanism allowing to know which system must produce which specific part of the output and synchronize it.

This abstract is structured as followed. The next section deals with the related work followed by the approach and its uniqueness. Implementation section briefly presents our prototype. Finally, we make a conclusion and introduce perspectives on future works.

BACKGROUND AND RELATED WORK

Many approaches for representing collaborative approaches have been proposed. We summarize here some of them.

In previous works, we addressed this topic by producing CMSPEM [8], a SPEM extension adding concepts to support collaboration in software processes. In CMSPEM, Kedji et al. introduced concepts to represent the stakeholders of a project, the artifacts manipulated and tasks to execute. Collaboration dynamic aspects are formalized through a system of events. Vo et al. [5] have defined some collaborations patterns that are a way of defining, reusing and enacting collaborative software development processes. In [5], they define two collaborations pattern which serve as collaboration strategies: Duplicate in Sequence with Multiple Actors and Duplicate in Parallel with Multiple Actors and Merge. Yakindu [6] provides an editor for editing and simulating statecharts but is not process-centered.

APPROACH AND UNIQUENESS

To bring solution to the issues addressed in the first section we propose an approach which allows to represent collaborative software processes and also their execution. Our approach includes the Collaborative Processes Behavioral Metamodel (CPBM). CPBM defines collaborative tasks behavior via UML state-machines. CPBM is composed of three packages: CPBM_Core, CPBM_Behavior and CPBM_Execution. The latter encompasses necessary concepts to execute a software process. It introduces concepts about tasks being performed by multiple actors. The behaviors of CPBM runtime elements are represented by states as *Instantiated, In Progress, Finished, ...* *ResourceElement* represents any kind of executor. It can be a human or a tool. A single task instance (STI) is the main executable concept. It represents a work unit assignable to a single actor or a tool. The execution of an STI depends on the definition of some collaboration strategies [5].

https://doi.org/10.1145/3167132.3167457

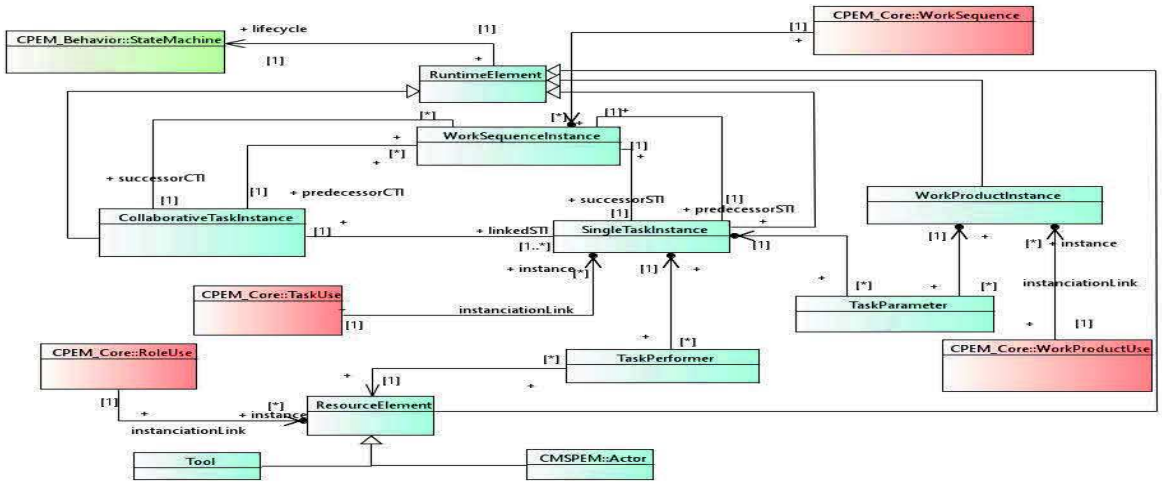

Figure 1: CPBM_Execution package structure

For this purpose, we can use workflow patterns-based strategies [7] to set the execution sequencing depending on the availability of resources and/or inputs and tasks synchronization. Strategies are dynamically chosen depending on the problem faced at enactment time. As an example, we can choose parallel or sequential execution depending on whether all the actors are available at the same time or not. Another reason motivating strategy choice, can be the link between instances. They can be linked through the sharing of artifact. This situation is observable when a given instance must use the output of another instance.

Finally, we have developed a prototype of a process engine for executing collaborative process.

IMPLEMENTATION

To handle the execution of collaborative processes, we have implemented a process engine Collaborative Process Engine.

Mainly our prototype allows project teams to upload a process model formatted in xml and generate all the single task instances to be enacted. Given project development is collaborative, the project manager can, choose how many instances of every task he wishes and also the collaboration strategy for each. The strategies depend on the chosen workflow patterns among those we addressed in our approach section. To this date, our prototype is working with a set of patterns based on parallel or sequential execution. The parallel pattern means that the STIs are executed simultaneously followed by a merge of the different outputs. The sequential one means that every STI must wait for the precondition to be fulfilled. Those STIs are assigned to actors with inputs and expected outputs. The resource assignation will allow every stakeholder to know his own tasks and the sequencing between them. Finally, every actor will be able to graphically visualize the states of his tasks.

CONCLUSION AND FUTURE WORK

This paper addresses a real-life modeling issue. Collaboration processes behavioral modeling is indeed an interesting research domain and has to be more deeply exploited.

During collaboration modeling, many elements have to be taken into account such as how to represent collaboration processes, which collaboration strategy to use, how to follow the execution of every instance of tasks, among all. To answer those issues, our contribution has been to propose an approach to manage collaborative processes with a flexible PMS supporting unexpected situations at enactment time and collaborative tasks performed by several actors.

Moreover, we have also implemented a Collaboration Process Engine (CPE) to support collaboration processes execution and tasks states visualization.

This paper has potentially opened fruitful research directions. The use of collaboration strategies has to be extended in order to allow our CPE to support more complex strategies. The behavior of the runtime elements is expressed through UML State Machines. In addition, we plan on supporting dynamic addition, removal of instances of tasks during execution time. Also, events and actions to switch from a state to another (e.g. In Progress to Finished) have to be refined to take into account resources availability and work product instances states.

REFERENCES

[1] Hawryszkiewycz, I.T., 2005. A metamodel for modeling collaborative systems. Journal of Computer Information Systems 45, 63–72.

[2] Cánovas Izquierdo, J.L., Cabot, J., 2016. Collaboro: a collaborative (meta) modeling tool. PeerJ Computer Science 2, e84. doi:10.7717/peerj-cs.84

[3] OMG-SPEM 2.0, http://www.omg.org/spec/SPEM/2.0 (2008)

[4] Briggs, R., Kolfschoten, G., Gert-Jan, V., & Douglas, D. (2006). Defining key concepts for collaboration engineering. Proc. AMCIS 2006, 17. ISO 690

[5] Vo, T.T, Coulette, B., Tran, H.N., Lbath, R. An Approach to Define and Apply Collaboration Process Patterns for Software Development. Model-Driven Engineering and Software Development, Springer, *CCIS series*, Vol 580, pp 248-262. 2015. ISBN 978-3-319-27868-1.

[6] Yakindu: Homepage. https://www.itemis.com/en/yakindu/state-machine/

[7] Van Der Aalst, W. M., Ter Hofstede, A. H., Kiepuszewski, B., & Barros, A. P. (2003). Workflow patterns. Distributed and parallel databases, 14(1), 5-51.

[8] Kedji, K.A., Lbath R., Coulette, B., NASSAR, M., Barese, 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.