Design of a Simulation Framework for Model-based Learning

Sarah Zribi[1], Antonello Calabrò[2], Francesca Lonetti[2], Eda Marchetti[2], Tom Jorquera[1] and Jean-Pierre Lorré[1]

[1]*Linagora, 75 route de revel 31400, Toulouse, France*

[2]*Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo", CNR, Pisa, Italy*

Keywords: Model-based Learning, Simulation, Business Process.

Abstract: In model-driven learning, simulation of Business Process is a key step for improving the learners's skills and enhancing teaching performance. In this paper, we provide the architectural design and the main functionalities of a model-based learning simulation framework. The main objectives of the proposed framework are: i) enable user-friendly learning simulation with a strong support for collaboration and social interactions; ii) provide a process-driven learning environment that allows emulation of the learner behavior when no learners are available to be involved on the simulation; iii) include event-based monitoring aiming at providing feedbacks for the learner assessment.

1 INTRODUCTION

In recent years, inside many industrial contexts and application domains, Business Process (BP) models have increased their importance due mainly to the possibility to easily allow formal specification, provide accepted and concise definitions and taxonomies, and develop an executable framework for overall managing of the process itself.

Specifically, in learning environment the availability of this kind of models makes easier the adoption of Business Process simulation for teaching purposes. Usually, a simulation attempts to mimic real-life or hypothetical behavior to see how processes or systems can be improved and to predict their performance under different circumstances. In learning environment, commonly BP simulation enhances student learning and problem-solving so to improve students's knowledge. Thus, models and simulations become a foundation for improving the learners's skill, enhancing teaching performance and providing a comprehensive framework. Different conceptual and mathematical models have been proposed for model-based learning and several types of simulations, including discrete event and continuous process simulations, have been considered (Blumschein et al., 2009). However, the main challenges of existing learning simulation proposals are about collaborative simulation and the derived learning benefits.

In this paper, we present a new simulation framework providing a flexible simulation environment with a strong support for collaboration and social interactions, as well as process visualization and monitoring. Model-based learning simulation can be compared to a collaborative game where a team of players composed of one coach and any number of learners work together in order to achieve a common goal. The main objective is consequently to provide an easy to use and user-friendly environment for the learners in order to let them take part of the process when their turn comes, assuming different roles according to the content they have to learn. The principal contribution of this paper is the design of an infrastructure for simulation of model-based learning. The simulation environment includes an event-based monitoring framework aiming at providing feedbacks for the learner evaluation and allows multi-sessions as well as collaborative BP simulation. In the rest of the paper we first briefly introduce some background concepts (Section 2), then is Section 3 we present the main components of the simulation framework architecture whereas in Section 4 we describe its main functionalities. Finally, conclusion concludes the paper.

2 BACKGROUND

The proposal of a simulation framework for model-based learning originated in the context of the Model-Based Social Learning for Public Administrations (Learn PAd) project (LearnPAd, 2016). The developed Learn PAd platform will support an informative learning approach based on enriched BP models, as well as a procedural learning approach based

Zribi, S., Calabrò, A., Lonetti, F., Marchetti, E., Jorquera, T. and Lorré, J-P.
Design of a Simulation Framework for Model-based Learning.
DOI: 10.5220/0005688606310639
In *Proceedings of the 4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2016)*, pages 631-639
ISBN: 978-989-758-168-7

on simulation and monitoring that will allow users to learn by doing. In learning context, BP simulation approaches are very popular since learners prefer simulation exercises to either lectures or discussions (Anderson and Lawton, 2008). Simulations have been used to teach procedural skills and for training of software applications and industrial control operations as well as for learning domain specific concepts and knowledge, such as business management strategies (Clark and Mayer, 2011). Nowadays, more attention is given to business process oriented analysis and simulation (Jansen-Vullers and Netjes, 2006). Studies have shown that the global purpose of these existing business process simulation platforms is to evaluate BPs and redesign them, whereas in the last years simulation/gaming is establishing as a discipline (Crookall, 2010). However, these platforms present several shortcomings regarding their applicability to a collaborative learning approach. Namely, no existing platform regroups all of the main functionalities of a learning simulation solution such as facilities for providing a controlled and flexible simulated environment (for example allowing to switch between possible outcomes of a task, in order to explore the different paths of a process), good visualization and monitoring of a process execution flow (in order both to assist and evaluate the learners) (Crookall, 2010). The main challenges of a learning simulation are about collaborative simulation and the derived learning benefits. To answer all of these concerns a new learning simulation framework is designed in this paper, providing a flexible simulation environment with a strong support for collaboration and social interactions, as well as process visualization and monitoring. The important objective is consequently to provide an easy to use and user-friendly environment for the learners in order to let them take part of the learning process assuming different roles.

3 SIMULATION FRAMEWORK ARCHITECTURE

In this section, we describe the high level architecture of the proposed simulation framework, its main components, their purpose, the interfaces they expose, and how they interact with each others. In particular, as depicted in Figure 1 each component is exposed as a service and provides an API as a unique point of access. Inside the Learn PAd infrastructure, the proposed simulation framework interacts with the Learn PAd components by means of the *Learn PAd Core Platform* and specifically through the *Bridge* and the *Core Facade* interfaces. Moreover, in the Learn PAd vision two levels of learners have been considered: the civil servant who is the standard learner, and the civil servant coordinator who is a generalization of the civil servant who is in charge to activate and manage a simulation session.

The simulation framework components are:

SimulationGUI: it is in charge of the interactions between learners and simulator's components.

PersistenceLayer: it stores the status of the simulation at each step in order to give to the civil servant the ability to stop it and restart when needed.

RobotFramework: it allows to simulate the behavior of civil servants by means of robots.

SimulationEngine: this is the core component of the simulation framework. It enacts business processes and links activities with corresponding civil servants or robots.

Monitoring: it collects the events occurred during the simulation and infers rules related to the business process execution.

Communication middleware: it provides event-based communication facilities between the simulation components according to the publish/subscribe paradigm.

UserFacade: it is in charge of encapsulating real or simulated civil servants (i.e. robots) in order to make the learner interaction transparent to the other components of the architecture.

In the following more details are provided.

3.1 GUI

As depicted in Figure 2, the main features of the simulation GUI are: the possibility of chatting and notification, the mutual interaction by input/output panel, the visualization of simulation members, the possibility of reading and searching for documents and additional information useful during the simulation activities (Context Area) and the usage of standard facilities as for instance `Play`, `Save`, `Pause`, `Stop` (Simulation lifecycle menu).

During a simulation, the civil servant coordinator selects the difficulty level of the simulation that can be elementary, intermediate or advanced. For each task of the BP and until the end, the simulation framework notifies the civil servant when his/her turn to learn comes, so that the civil servant fills the required forms and submits them. Moreover, the simulator offers the possibility to the civil servant, when he/she needs some help, to chat with online experts, or to consult recommended resources and FAQ that fit for the current context. Finally, when all tasks of the BP are simulated, the simulator provides an evaluation of the simulation.

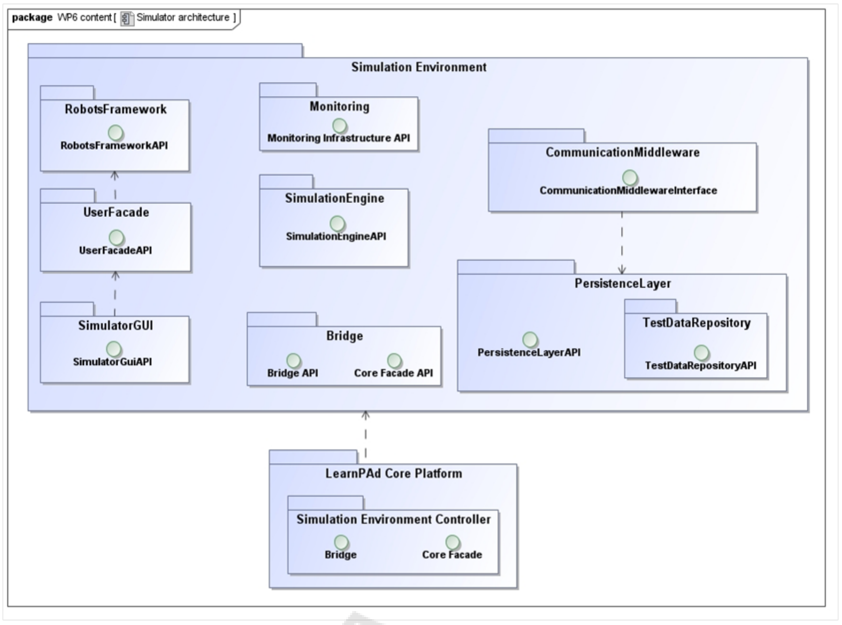

Figure 1: Simulation Framework Architecture.

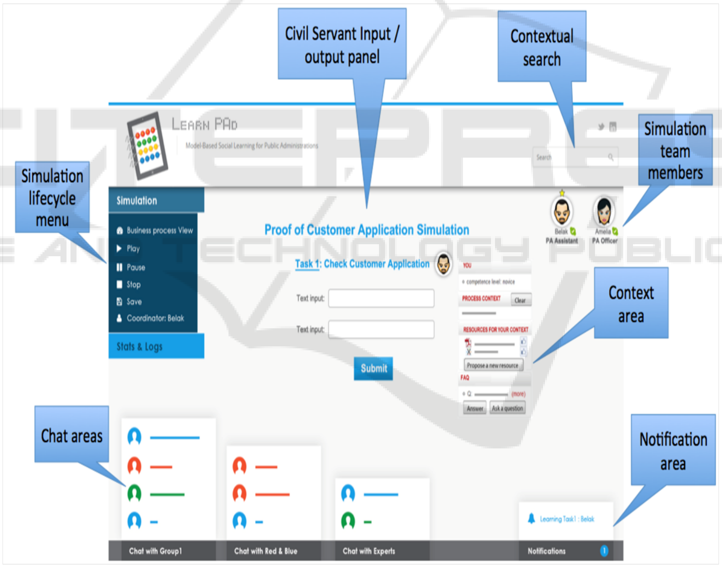

Figure 2: Main User Interface.

3.2 Persistency Layer

The Persistency Layer component is in charge to store simulation logs and business process states. Its main sub-components are:

Logger: It is in charge of storing time-stamped event data coming from the simulation engine. These event data are produced each time a treatment is triggered by the engine: BP activity invocation, user input/output, etc.

BPStateStorage: This component allows to store/retrieve/delete/update the state of a given simulation associated to a BP. This is a key feature that allows the user to freeze a simulation, log out and come back to the simulation later on. The component manages all the necessary data and stores them into a database.

TestDataRepository: This component collects the historical data that relate to the simulations executions. Historical data specify input and output information/values corresponding to paths (or subpaths) and activities of the BP and are inserted and validated

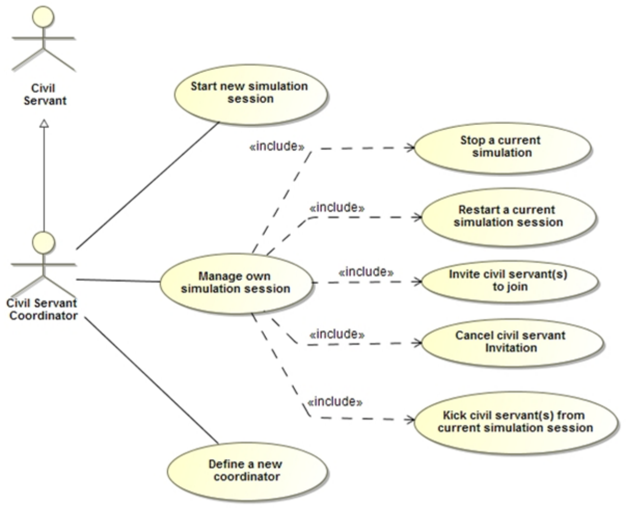
Civil Servant
Civil Servant Coordinator
Start new simulation session
Manage own simulation session
Define a new coordinator
«include»
Stop a current simulation
Restart a current simulation session
Invite civil servant(s) to join
Cancel civil servant Invitation
Kick civil servant(s) from current simulation session

Figure 3: Simulation Environment Functionalities (Civil Servant Coordinator).

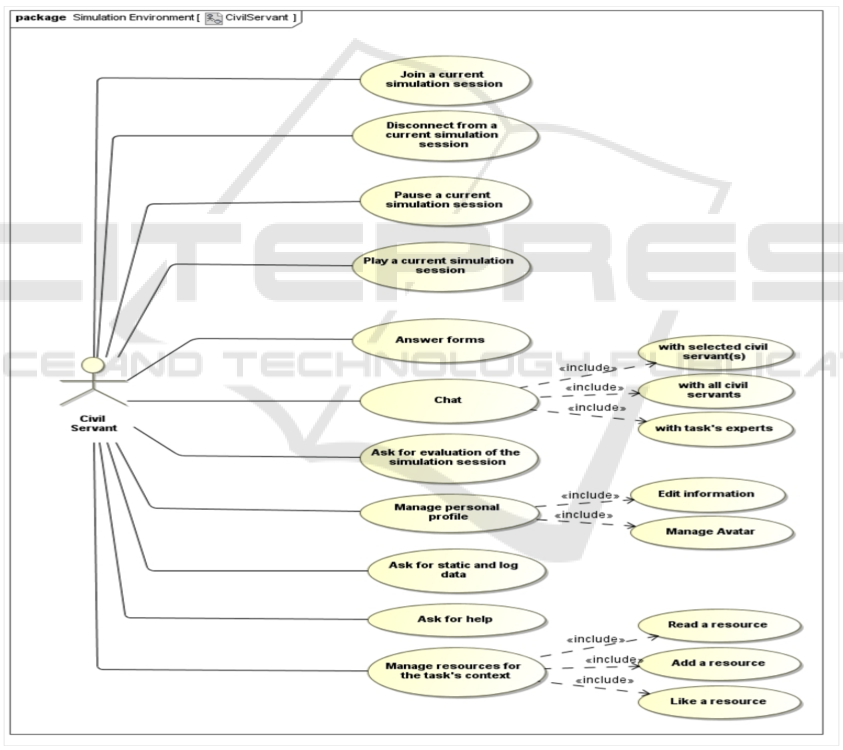
package Simulation Environment [CivilServant]
Civil Servant
Join a current simulation session
Disconnect from a current simulation session
Pause a current simulation session
Play a current simulation session
Answer forms
Chat
with selected civil servant(s)
with all civil servants
with task's experts
Ask for evaluation of the simulation session
Manage personal profile
Edit information
Manage Avatar
Ask for static and log data
Ask for help
Manage resources for the task's context
Read a resource
Add a resource
Like a resource
«include»

Figure 4: Simulation Environment Functionalities (Civil Servant).

by experts. These data can be used for verifying the correctness of the information provided by a civil servant during a simulation session as well as for inferring the robot behavior.

3.3 Robots Framework

During a simulation some of the required civil servants may not be present, therefore robots mimic missing learners behavior. They are aimed to provide outputs that would have been produced by a human with the same input in the same context. The Robots are implemented on the basis of the availability of historical data, i.e. the data saved in the *TestDataRepository* during a previous simulation session and provided by an expert who takes the role of the civil servant. In case of unavailability or incompleteness of these data, either an expert may be asked to provide historical data manually or a knowledge-based approach able to deduce output from previously collected answers provided by learners, is applied.

3.4 Simulation Engine

Simulation engine takes in charge the simulation of a given business process instance. It takes the form of an orchestration engine that invokes treatments associated to each activity of the current process. Such workflow may involve multiple civil servants taking different roles that may be present or not. For those that are not available, robots are used in order to mimic their behavior. A simulation manager is provided in order to manage BP lifecycle according to the current context (create, stop, resume, kill, etc.). Business processes are made of two kinds of activities: i) Human activities involve civil servants who should provide information in order to complete the task. The concept of human activity is used to specify work which has to be accomplished by people; ii) Mocked activities involve robots to compute the treatment associated to the activity. When the simulation engine invokes a human activity the corresponding civil servant is asked to provide input through a form. Those forms are managed by a form engine that delegates task to a robot if necessary. All the state information necessary to restart a specific simulation are stored “on the fly”. The civil servant may decide to freeze a running simulation, to store it, to backtrack to a previous stored state and to logout. He/she will be able to resume it later.

Business Process orchestrator takes in charge the step by step execution of a given BP instance. Such BP instance is made of a BPMN description enriched with necessary run-time information such as end-points of software applications mocks, user id, etc. The BP engine is connected with the Forms Engine in order to take in charge users and robots input/output.

As reported in Figure 5, during the simulation process, the input generated by the civil servant (messages 1-4) can be validated on line by means of data stored on *TestDataRepository* component.

In particular the validation process is divided into three main sub-cases:

- generic case: it occurs when the simulation engine have to validate the learner input. In this case the simulation engine will check through the *TestDataRepository* if the input can be considered correct or not and provides a feedback to the Learner (messages 5,6 of Figure 5).
- extra-evaluation required: this sub-case is activated when the learner (Civil Servant) input and the validation obtained by the simulation framework do not match each other. In case the learner is sure of the correctness of his/her reply, he/she has the possibility to open a dispute requesting an extra evaluation from one or more experts (message 7 of Figure 5).
- validation missed: in this last case the simulation framework has insufficient information to evaluate the learners input. This could happen because Test Data Repository does not contain a valid and certified by expert data (messages 8,9 of Figure 5).

In both second and third case, the simulation engine will raise an alert to an expert and let the learner continue the simulation (if experts are not available online), and the result of that validation will be provided to the learner as soon as possible.

3.5 Monitoring

The simulation framework is equipped with a monitoring facility that allows to provide feedbacks on the business process execution and learning activities. This monitoring allows to collect data of interest during the run-time business process execution and is based on an event based monitoring infrastructure (Bertolino et al., 2011) which has the peculiarity of decoupling the events specification from their collection and processing. This monitoring facility can be independent from any specific business process modeling notation and execution engine. Data collected during monitoring of business process execution are used for deriving the key performance indicators (KPI) that allow continuous tracking of the process behavior and measurement of learning-specific goals. We refer to (Calabrò et al., 2015b) for a detailed description of the monitoring components.

4 FUNCTIONAL SPECIFICATION OF THE LEARNING SIMULATION FRAMEWORK

The simulation framework provides the subsystem where learners can simulate BP interactively and is used by one or multiple civil servant(s) in order to learn processes. Figures 3 and 4 show the main functionalities that the Simulation Framework provides at the stakeholders. As mentioned in Section 3, the simulation framework distinguishes between the two following actors: the civil servant coordinator who is in charge of starting a simulation session and the civil

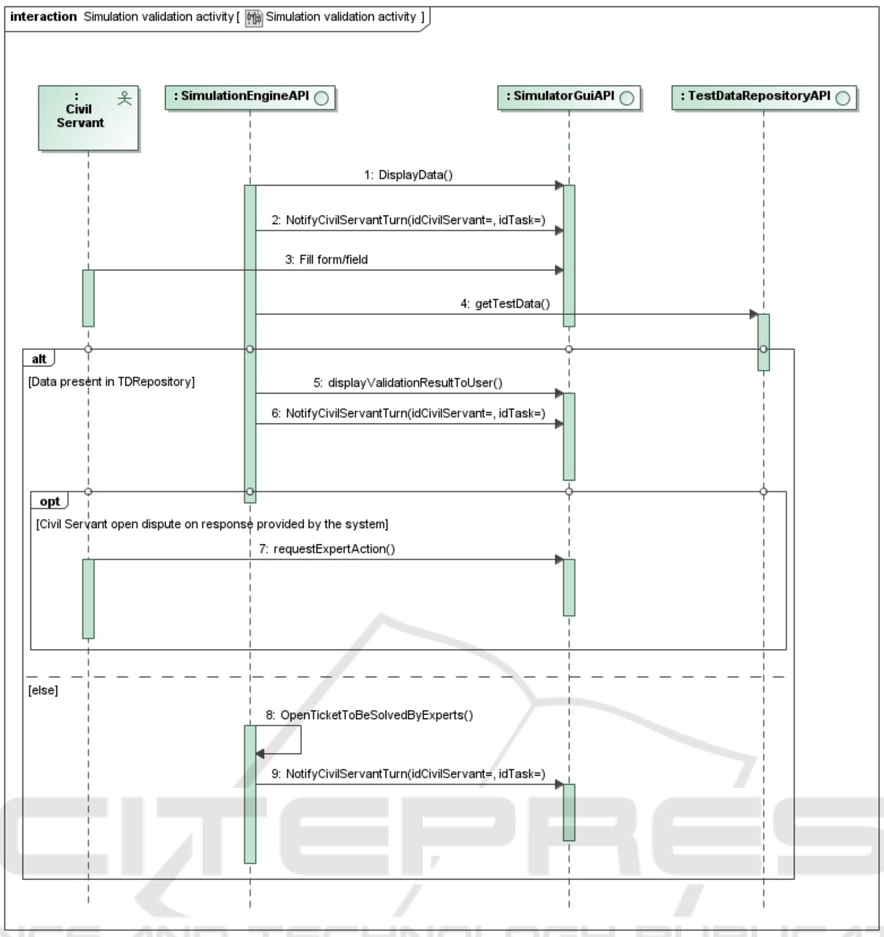
interaction Simulation validation activity [Simulation validation activity]
: Civil Servant
: SimulationEngineAPI
: SimulatorGuiAPI
: TestDataRepositoryAPI
1: DisplayData()
2: NotifyCivilServantTurn(idCivilServant=, idTask=)
3: Fill form/field
4: getTestData()
alt
[Data present in TDRepository]
5: displayValidationResultToUser()
6: NotifyCivilServantTurn(idCivilServant=, idTask=)
opt
[Civil Servant open dispute on response provided by the system]
7: requestExpertAction()
[else]
8: OpenTicketToBeSolvedByExperts()
9: NotifyCivilServantTurn(idCivilServant=, idTask=)

Figure 5: Simulation activity diagram.

servant who represents a generic participant to a simulation session. In particular, as in Figure 3, the civil servant coordinator can request to start a new simulation execution of a Public Administration business process or he/she can manage an ongoing one by for instance inviting/cancelling other civil servants. The civil servant coordinator can also restart/stop a current simulation session and redefine a new coordinator. On its turn, each civil servant has different possibilities (as listed in Figure 4) like for instance joining, disconnecting or pausing a simulation session, chatting, asking for evaluation/help, or managing his/her own profile.

The simulation framework functionalities have been split into three different phases: i) *Initialization* in which the simulation environment is set up; ii) *Activation* in which the participants to the simulation are invited; iii) *Execution* in which the participants effectively collaborate each other during a learning session. During the Activation phase, the civil servant can select the type of simulation he/she wants to execute. Specifically, three different types of simulation are provided:

Individual Simulation. The civil servant decides to execute the simulation without interacting with other human participants. In this case the other participants are emulated by means of *Robots* (see section 3.3 for more details). The creation of robots instances is performed before the simulation execution.

Collaborative Simulation. This option of simulation involves the collaboration of several human participants (no robots instances are involved). During the collaborative simulation, users can interact between them using chat instruments. This will improve performances of the overall learning session due to the possibility to rapidly share experience between human participants. This kind of simulation can be considered the most interesting from the learning point of view, because cooperation can make learning procedures more intensive and productive. Diversities will raise up and the opportunity to reflect upon encountered issues will help learners to improve

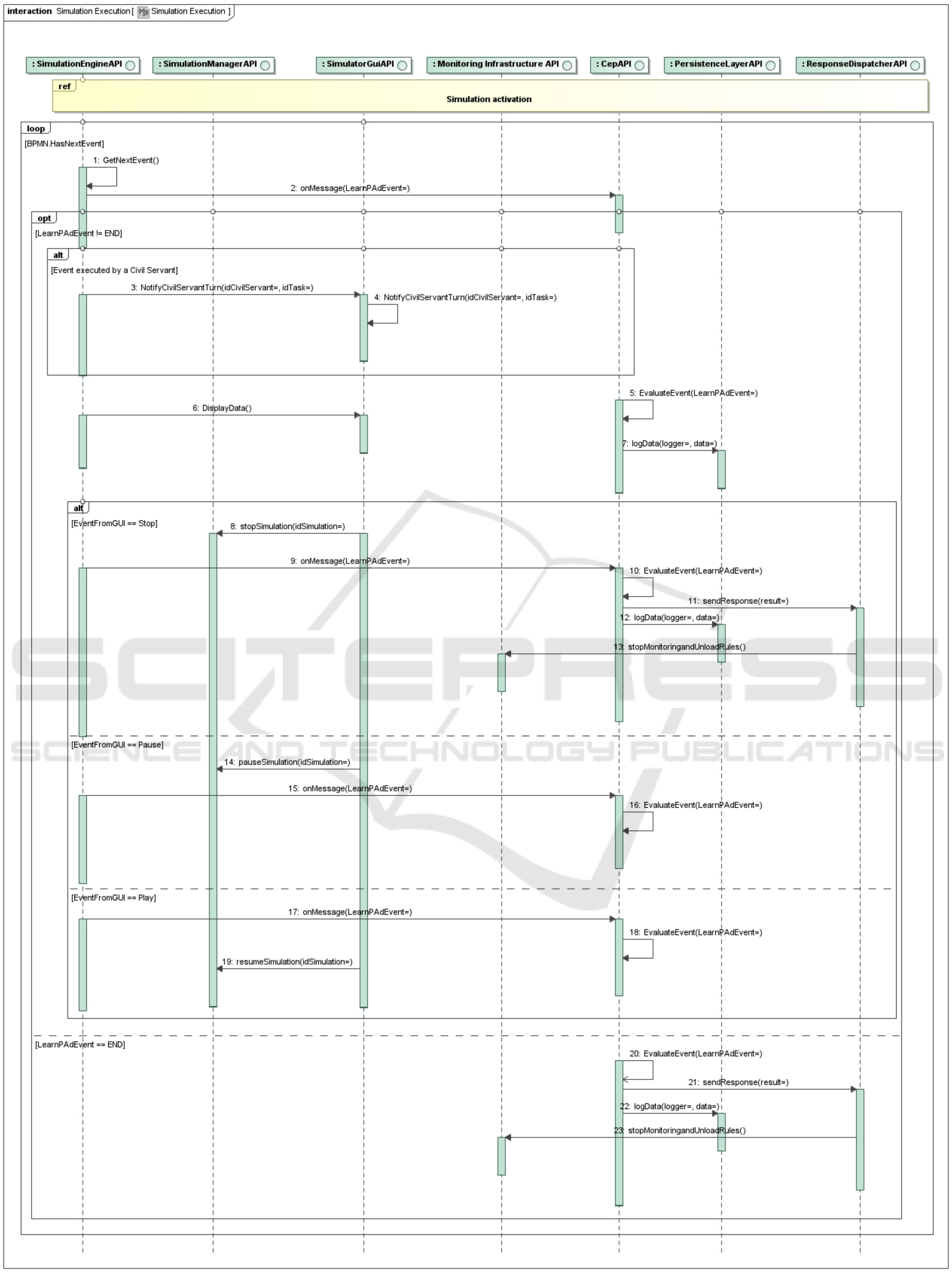
interaction Simulation Execution [Simulation Execution]
: SimulationEngineAPI
: SimulationManagerAPI
: SimulatorGuiAPI
: Monitoring Infrastructure API
: CepAPI
: PersistenceLayerAPI
: ResponseDispatcherAPI
ref
Simulation activation
loop
[BPMN.HasNextEvent]
1: GetNextEvent()
2: onMessage(LearnPAdEvent=)
opt
[LearnPAdEvent != END]
alt
[Event executed by a Civil Servant]
3: NotifyCivilServantTurn(idCivilServant=, idTask=)
4: NotifyCivilServantTurn(idCivilServant=, idTask=)
5: EvaluateEvent(LearnPAdEvent=)
6: DisplayData()
7: logData(logger=, data=)
alt
[EventFromGUI == Stop]
8: stopSimulation(idSimulation=)
9: onMessage(LearnPAdEvent=)
10: EvaluateEvent(LearnPAdEvent=)
11: sendResponse(result=)
12: logData(logger=, data=)
13: stopMonitoringandUnloadRules()
[EventFromGUI == Pause]
14: pauseSimulation(idSimulation=)
15: onMessage(LearnPAdEvent=)
16: EvaluateEvent(LearnPAdEvent=)
[EventFromGUI == Play]
17: onMessage(LearnPAdEvent=)
18: EvaluateEvent(LearnPAdEvent=)
19: resumeSimulation(idSimulation=)
[LearnPAdEvent == END]
20: EvaluateEvent(LearnPAdEvent=)
21: sendResponse(result=)
22: logData(logger=, data=)
23: stopMonitoringandUnloadRules()

Figure 6: Simulation execution.

their knowledge and better understand the problem. For activating a simulation, the system requires that all the civil servants involved have joined the session in order to provide an online collaborative environment. Moreover, the simulation framework also supports also some asynchronous tasks execution among simulation participants. If a civil servant does not satisfy the simulation requirements or time constraints, the civil servant coordinator may decide either to kick the civil servant, or to swap him with another one among those available, or replace him with a *Robot*.

Mixed Simulation. This type of simulation requires the participation of both humans and robots. This usually happens when there are not enough civil servants to cover all the necessary roles to execute a BP or if one or more civil servants leave the ongoing simulation (disconnection or kick). The activation of a mixed simulation can be done only if the following two constraints are met: i) the required instances of robots are ready; ii) all the invited civil servants have completed the connection procedures.

The sequence diagram presented in Figure 6 is one of the realizations of the use case “Manage own simulation session” of Figure 3 and shows the main activities of the simulation execution. This sequence diagram refers to simulation activation sequence diagram as a precondition, not presented here for space limitations. Once simulation activation phase is completed the Simulation Engine generates the events related to the execution of tasks of the BP (message 1) and sends them to the Monitor for their evaluation (message 2). The simulation execution continues managing the BP tasks and sending notification to the civil servants by means of *SimulationGuiAPI* till a special kind of event is received. Four types of special events are managed: *END* rises when the BP execution reaches the natural end of the model; *STOP* rises when the civil servant coordinator decides to early terminate a simulation execution; *PAUSE* rises when a civil servant decides to pause the simulation execution; *PLAY* rises when a civil servant decides to recover a paused simulation. During the execution the simulation engine, continuously generates events and forwards them to the *CepAPI*. According to the event executed the civil servants participating to the simulation receive through the *SimulationGUI*, the documents and suggestions related to each task. If the BP execution has reached the end, an *end* event is sent to the *CepAPI* in order to stop the monitoring session and unload resources.

5 CONCLUSIONS

In this paper, the architectural design of the simulation and monitoring framework is presented with a particular focus on the definition of the functionalities and interactions among its components. The framework is currently under evaluation inside the Learn Pad project. However, the preliminary results provided in (Calabrò et al., 2015b; Calabrò et al., 2015a; Calabrò et al., 2016) evidenced positive feedbacks, especially concerning the possibility of executing collaborative simulation and providing learners assessment. Moreover, the design of some parts of the architecture, such as the Test Data Repository, will be refined and improved through comments and hints that will be collected over the project duration.

ACKNOWLEDGEMENTS

This work has been partially funded by the Model-Based Social Learning for Public Administrations project (EU FP7-ICT-2013-11/619583).

REFERENCES

Anderson, P. H. and Lawton, L. (2008). Business simulations and cognitive learning: Developments, desires, and future directions. *Simulation & Gaming*.

Bertolino, A., Calabrò, A., Lonetti, F., and Sabetta, A. (2011). Glimpse: A generic and flexible monitoring infrastructure. In *Proc. of the 13th EWDC*, pages 73–78.

Blumschein, P., Hung, W., and Jonassen, D. H. (2009). *Model-based approaches to learning: Using systems models and simulations to improve understanding and problem solving in complex domains*. Sense Publishers.

Calabrò, A., Lonetti, F., and Marchetti, E. (2015a). Kpi evaluation of the business process execution through event monitoring activity. In *Proc. of Third International Conference on Enterprise Systems*.

Calabrò, A., Lonetti, F., and Marchetti, E. (2015b). Monitoring of business process execution based on performance indicators. In *Proc. of Euromicro-SEAA*.

Calabrò, A., Lonetti, F., Marchetti, E., Zribi, S., and Jorquera, T. (2016). Model-based learning assessment management. In *Proc. of MODELSWARD*.

Clark, R. C. and Mayer, R. E. (2011). *E-learning and the science of instruction: Proven guidelines for consumers and designers of multimedia learning*. John Wiley & Sons.

Crookall, D. (2010). Serious games, debriefing, and simulation/gaming as a discipline. *Simulation & gaming*, 41(6):898–920.

Jansen-Vullers, M. and Netjes, M. (2006). Business process simulation–a tool survey. In *Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools, Aarhus, Denmark*, volume 38.

LearnPAd (2014-2016). Model-Based Social Learning for Public Administrations European Project (EU FP7-ICT-2013-11/619583). http://www.learnpad.eu/.