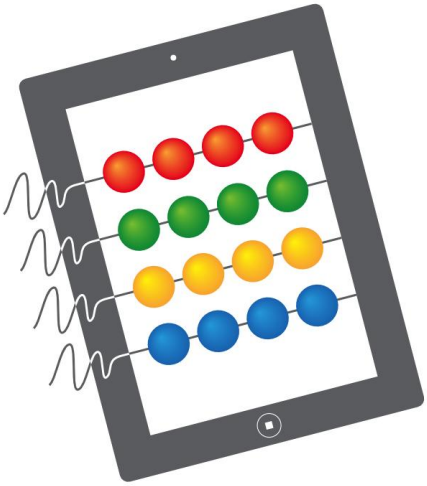
FP7 ICT STREP Project



LEARN PAD

Deliverable D6.1

Learn PAd Simulation Environment

Specification and Design

http://www.learnpad.eu

Project Number	:	FP7-619583
Project Title	:	Learn PAd Model Based Social Learning for Public Administrations

Deliverable Number	:	D6.1
Title of Deliverable	:	Learn PAd Simulation Environment: Specification and Design
Nature of Deliverable	:	Report
Dissemination level	:	Public
Licence	:	Creative Commons Attribution 3.0 License
Version	:	4
Contractual Delivery Date	:	31 January 2015
Actual Delivery Date	:	27 January 2015
Contributing WP	:	WP6
Editor(s)	:	Eda Marchetti
Author(s)	:	Antonia Bertolino, Antonello Calabrò, Tom Jorquera, Francesca Lonetti, Jean-Pierre Lorre, Eda Marchetti, Sarah Zribi.
Reviewer(s)	:	Frieder Witschel

Abstract

This document describes the Learn PAd Simulation Framework focusing in particular on the Simulation and Monitoring activities. For this, the main peculiarities of the proposed Simulation Framework are detailed as well as its main functionalities and architectural design. A model-based approach that will be integrated into the Monitoring Component is also proposed. This will be useful to assess the learning activity during Business Process simulation.

Keyword list

Simulation and Monitoring Environment Architecture, Simulation and Monitoring Environment Components, Simulation Environment and Monitoring Functional Specification.

Document History

Version	Changes	Author(s)
0.1	First Draft	Guglielmo De Angelis, Andrea Polini.
0.2	Layout	Guglielmo De Angelis, Andrea Polini
0.3	Improved styles	A. Bertolino, G. De Angelis.
1.0	ToC	Antonia Bertolino, Antonello Calabrò, Francesca Lonetti, Jean-Pierre Lorre, Eda Marchetti, Sarah Zribi.
1.5	Initial design	Antonia Bertolino, Antonello Calabrò, Francesca Lonetti, Jean-Pierre Lorre, Eda Marchetti, Sarah Zribi.
2	First draft completed	Antonia Bertolino, Antonello Calabrò, Francesca Lonetti, Jean-Pierre Lorre, Eda Marchetti, Sarah Zribi.
2.5	Final version of the design	Antonia Bertolino, Antonello Calabrò, Francesca Lonetti, Jean-Pierre Lorre, Eda Marchetti, Sarah Zribi.
3.0	Internal version	Antonia Bertolino, Antonello Calabrò, Francesca Lonetti, Jean-Pierre Lorre, Eda Marchetti, Sarah Zribi.
3.5	Review comments solved	Antonia Bertolino, Antonello Calabrò, Francesca Lonetti, Jean-Pierre Lorre, Eda Marchetti, Sarah Zribi.
4.0	Final editing	Antonia Bertolino, Antonello Calabrò, Francesca Lonetti, Jean-Pierre Lorre, Eda Marchetti, Sarah Zribi.

Document Review

Release	Date	Ver.	Reviewers	Comments
ToC	November 28, 2014	1.0		
Draft	December 26, 2014	2.0		
Internal	January 9, 2015	3.0	Frieder Witschel	
Candidate Final	January 31, 2015	4.0		

Glossary, acronyms & abbreviations

Item	Description
AngularJS	Angular JavaScript
API	Application Programming Interface
BP	Business Process
BPM	Business Process Modeling
CEP	Complex Event Processor
DOW	Description of Work
GUI	Graphical User Interface
GWT	Google Web Toolkit
MVC	Model-View-Controller
HTML	HyperText Markup Language
SPA	Single-Page Applications
WP	Work Package

Table of Contents

List Of Figures

1. Introduction

This deliverable focuses on the Learn PAd Simulation Framework that integrates Simulation and Monitoring activities. The objective is to provide facilities useful to learners for improving their knowledge about a given business process also by means of the interactions between different participants.

In the next sessions the goals of the deliverable and the relation between D6.1 and others Learn PAd deliverables are provided in more details. Finally the overall structure of the deliverable is highlighted.

1.1. Goals

In the context of Learn PAd project, WP6 aims at providing a simulation environment where learners will engage in training activities interacting with the different participants of a given business process. The simulation environment will also include monitoring features providing feedbacks on the business process execution for the evaluation of learners.

This document presents the first results of WP6 related to the following tasks (from the Learn PAd DoW).

- Task 6.1: Design of the Infrastructure for Multi-Sessions BP Simulation and Monitoring

  The main objective of this task is to design a simulation and monitoring architecture based on the requirements listed in WP1 and on the Learn PAd Platform sketched in WP2. This architecture will support the access and collaboration of learners involved into the Business Process and will embed a configurable monitoring framework aiming at revealing both the possible criticalities in the perception of the Business Process, and its associated learning contents from the learners. Monitored events will provide a feedback for the ranking activities and KPI definition presented in WP5.

- Task 6.2: Model-Based Testing Strategies Enabling BP Simulations

  In the simulation environment the involved parties will be in some cases simulated by synthesized software according with the interactions foreseen within the BP. This task investigates model-based testing approaches and strategies enabling the simulation of the functionalities that would have been provided by human or process actions.

The main purpose of this deliverable is to design the high level architecture of the simulation and monitoring environment. To this end, the internal architecture of the main components is presented and a detailed description of their functionalities, behaviour and interactions is provided. The proposed simulation environment can engage learners in both individual and collaborative simulation allowing them to train on the business process tasks by interacting with other learners. Moreover, it supports also the emulation of the learner behavior when no learners are available to be involved on the simulation. The simulation environment includes an event based monitoring framework aiming at providing feedbacks for the learner evaluation. The main goals of the monitoring infrastructure are: revealing the possible criticalities of the learning activity relative to the coverage of the business process; checking the business process execution time and finally provide feedbacks for evaluating the civil servant competency and the collaborative learning activities.

1.2. Related deliverables

The deliverable considers as inputs the results of the Learn PAd project presented in the following deliverables:

- D1.1. Requirements Report. This document reports on the collection and analysis of requirements for the Learn PAd system. The simulation environment design is compliant with the simulation requirements specification.
- D2.1. Deliverable D2.1 Platform Architectural Description. This document describes the high level architecture of the Learn PAd Platform and the main use case scenarios evidencing the components interactions. The Simulation Environment is one of the main components of this architecture and the functionalities of the Simulation Environment presented in this deliverable refer to Use Case 3.8.1 and 3.8.2.

The deliverable is also related to the following deliverables that will be delivered at M12:

- D5.1. Models for Setting the Wiki.
- D3.2. Design and Initial Implementation of Metamodels for Describing Business Processes in Public Administrations.

1.3. Structure of the deliverable

The deliverable is organized as follows. Chapter 2 provides details about the functional specification of the simulation and monitoring environment whereas Chapter 3 details the realization of the features defined in Chapter 2 in terms of Sequence Diagrams. Chapter 4 presents the architectural design of the Simulation Frameworks and a specific description of each of its components. In Chapter 5 a model based approach integrated into the monitoring environment that is useful to assess the thoroughness of the learning activity during Business Process simulation is presented. Finally, conclusions and future work are drawn in Chapter 6.

2. Functional specification of Learn PAd simulation framework

The Learn PAd Business Process simulation may 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. In order to improve the readability of this deliverable in the following a possible scenario of simulation execution is provided, where two possible learners, named Belak and Amelia collaborate through the Learn Pad simulation framework for executing a Business process.

In the considered scenario, before starting the simulation phase Belak, needs to choose between two options: the first one is about starting a new simulation of the business process and the second option offers the possibility to continue a previous simulation from a saved point.

Supposing that Belak chooses to start a new simulation, he becomes the coordinator and has to perform the activities as in the following.

As a first step of the simulation, Belak has to provide his role, to declare his level of competence, to select the business process to simulate and to choose the type of simulation that he wants to run. In the simulation framework three type of simulation are provided: individual, collaborative and mixed (more details about them are provided in Chapter 3).

In this example Belak chooses a collaborative one therefore he has to invite another learner (Amelia) to simulate the BP by sending her an email.

Once invited, Amelia joins the simulation platform and the simulation is activated.

Then Belak selects the difficulty level of the simulation between the three options available: elementary, intermediate or advanced.

At this point, the simulation is ready to be executed and the Simulation Framework notifies to Amelia when her turn to learn comes so that, she can starts interacting with the platform.

Finally, when all tasks of the Business Process will be completed the Learn PAd Simulation Framework provides to Amelia and Belak the possibility to receive an evaluation of the simulation.

2.1. Background and state-of the art of simulation

In learning context, business process simulation approaches are very popular since learners prefer simulation exercises to either lectures or discussions [1]

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 [3]. Nowadays more attention is given to business process oriented analysis and simulations [6].

Studies have shown that the global purpose of these existing business process simulation platforms is to evaluate business processes and redesign them when necessary. Obviously, this is not the aim of the Learn PAd simulator since its role is to simulate a previously validated business process for users (i.e. civil servants) in order to learn it.

Consequently, these platforms present several shortcomings regarding their applicability to a collaborative learning approach. Namely, no existing platform regroups all of the important functionalities of this approach: tools to easily define and adapt business processes, 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 visualisation and monitoring of a process execution flow (in order both to assist and evaluate the learners) [6], [1]. The main challenges of learning simulation are about collaborative simulation and the derived learning benefits. All these characteristics are required for a collaborative learning environment [3] and must be supported within the Learn PAd simulation platform.

For this reason a new architecture designed specifically to answer all of these concerns has been proposed, providing a flexible simulation environment with a strong support for collaboration and social interactions, as well as process visualization and monitoring. The main objective is consequently to provide an easy to use and user friendly environment for the civil servants 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.

In contrast and in some cases, during a business process simulation some of the required civil servants may not be present. Since the focus concern collaborative learning, in the Learn PAd simulation platform there is the possibility to replace missing learners by what is called: *Robots,* entities that mimic the behavior of a civil servant during the simulation session. Robots are virtual agents assuming some of the roles defined in the process. They are aimed to provide outputs that would have been produced by a human with the same input in the same context.

This make possible for a simulation to be either completed by:

i. a group composed fully of human civil servants;
ii. a group of civil servants assisted by robots, or even by
iii. a single civil servant with all the other roles fulfilled by robots.

This provides obvious advantages in the context of learning, avoiding the need for assembling a complete human team in order to start a learning session, or allowing an individual to focus on a specific part of a process. Consequently, the simulation environment of Learn PAd also includes appropriate mechanisms to support the behavioral emulation of the involved parties not available for the simulation.

Besides, as said above the Learn PAd simulation environment includes monitoring activities in order to provide feedbacks for the evaluation of learners, business processes, and associated learning contents.

2.2. Motivations and challenges for monitoring

Monitoring represents a key element for the management of communication networks and distributed systems. Large systems may potentially generate large number of events which need to be filtered and combined to detect unexpected behaviors or to estimate non functional properties of the system.
Monitoring has been defined as the process of dynamic collection, interpretation, and presentation of information concerning objects or software processes under scrutiny [7][6]. Event-based monitoring is a common approach for observing and analyzing the behavior of distributed systems [8].
Elaborating on [8], five core functions can be identified for a generic monitoring system:

1. Data collection: this function concentrates on the collection of raw data from the execution of the observed components. This can be done by either instrumenting the subject component (when this is possible) or by intercepting interactions among components through a proxy-based probe. A special case is represented by the built-in logging facilities that many systems provide natively.

2. Local interpretation: this function refers to the filtering that raw data go through before being interpreted at an aggregated level, to remove redundant or irrelevant information.
3. Data transmission: in distributed systems, this function takes care of gathering information from different originating nodes to a central (possibly not unique) node. Data transmission might exploit smart optimization algorithms (e.g., to delay data transmission or to give higher priority to certain information, when the network is subject to congestion).
4. Global interpretation: (also known as “correlation" or “aggregation"), this function makes sense of pieces of information that come from several nodes and puts them together in order to identify interesting conditions/events that can be observed only at an aggregated level. This function can be realized by means of a complex-event processing engine.
5. Reporting: this function deals with presenting the results of monitoring in a format that is meaningful to the “consumer" of the monitoring system. The consumer can be a human (e.g., a system administrator) or a program (e.g., a software component that implements the feedback loop in a self-controlled software system).

The monitoring infrastructure embedded in the Simulation Environment is designed in order to cover these five functions in a modular and flexible way and it is based on messages exchange. The main goal of the monitoring activity inside the simulation environment is to provide feedbacks for the evaluation of learners, business processes and associated learning contents. Monitoring mechanisms will allow to generate activity reports that will enable analysis and corrective actions. Such mechanisms will be useful for training and assessment purposes.

The monitoring activity will provide inputs for the learning process KPI evaluation. Specifically, monitoring feedback will be used for evaluating three important aspects of the learning process:

I. the time required for completing the simulation and executing the different tasks. To this aim the monitoring infrastructure will collect the start and stop time of each phase of the simulation considering also the pause/play actions of the Civil Servant and the execution time of single tasks/activities of the BP.
II. the civil servant competency. This competency will be evaluated from one side in terms of cardinality of “failures” of the business process execution. A failure of the business process execution will be detected by the simulation engine when data specified by the learner during a task/activity execution are not compliant with data stored in the Test Data Repository. The failures estimation will take into account also the criticality of the tasks/activities involved in the failures. From the other side the civil servant competency will be evaluated also considering the number of help requests to the expert or the requests of using the support of the documentation provided by the learning platform by the civil servant. The monitoring infrastructure will be able to trace both the number of failures and the cardinality of help requests to the experts and learning documentation.
III. the collaborative learning. To take trace of the collaborative learning activities the monitoring infrastructure will provide feedback for detecting the usage frequency of the platform and the learning activities performed both by the civil servants and the experts during the simulation. To this aim, the intensity of the communication in terms of number of chats and participant learners will be monitored.

2.3. Model based approaches

The adoption of business process modeling promotes and makes easier the use of model-based approaches for verifying the dynamic distributed systems.
Indeed monitoring has become an essential asset for controlling and managing distributed systems since early works such as [10], [7], [8]. A relatively recent survey [4] underlined a renewed interest in monitoring approaches and tools, due to the increasing complexity and pervasiveness of software systems. Particular interest has been dedicated to “smart” monitoring approaches, i.e., monitors enhanced beyond the passive observation of system executions, with the aim of preventing or anticipating potential risks. It is understood that such smart approaches cannot neglect any of the traditional challenges, and hence need to rely on top of solutions to address those ones as well.
As a trend, several researchers start to consider monitoring a useful instrument to observe the behavior of business processes not only to report about problems that have already occurred but also to predict likely problems in the near future. In [2], for example, the authors propose a choreography monitor for early detection of runtime faults, intended as deviations from admissible message exchanges between the cooperating services.
However, most of the predictive approaches remain limited to the elaboration of the passively captured executions. In contrast, in the model-based monitoring approach proposed in this deliverable the monitor is empowered also to raise a warning of not having observed for some time interesting behaviours or situations. This takes inspiration from the passive testing approaches, which refer to the observation of the input/output behavior of a system during normal operation for the purpose of detecting faults [8]. Every collected trace is matched to a model to verify if it conforms to the specified behaviour. The conformance check is done either off-line or on-line, therefore passive testing constitutes a sophisticated type of functional monitoring, in which the monitor is instructed to observe the traces permissible in the model.
Thus on the bases of passive testing inside the Learn PAd project a monitoring activity is introduced so to assess the thoroughness of the learning activity relative to the business processes in simulation.
The intuitive motivation becomes that if some part of the BPM has never been executed, the civil servant might have not simulated important steps of the business process and therefore his/her acquired knowledge not completed.
Thus, to assess the adequacy of a simulation execution it is important to identifying what are the relevant entities to be covered and monitoring if all of them, or otherwise what percentage, have been observed.
Through the analysis of monitoring data collected over the simulation sessions the Civil Servant from one side can become conscious of the level of knowledge he/she acquired and can receive and evaluation/score of the progress done. Form the other he/she can have a clearer picture of his/her exploration over a simulation, i.e to know exactly the entities (either the events, or message interactions, or business patterns) so far not executed, so either to timely decide how to continue the learning activity or get his/her evaluation score.
Exploiting the BPM specification, in Learn PAd project, the notion of learning session adequacy will be introduced and different adequacy criteria defined. The proposal is inspired by the notion of coverage assessment, which has been introduced in software testing decades ago [11]. The main difference between the traditional notion of testing adequacy and the learning session adequacy introduced here is represented by the concept of the observation window along which coverage is measured. Thus, the period in which learning session adequacy is assessed is established a priori by means of the observation window. This adapts to the monitoring activities performed in this deliverable to the classical concept of a test session or test suite on which traditional test adequacy is evaluated.

More formal definitions of learning session adequacy and adequacy criteria as well as examples of their evaluation will be provided in Chapter 5.

2.4. Use case scenarios

The simulation environment provides the subsystem where learners can simulate Business Processes interactively. Simulator is used by one or multiple civil servant(s) in order to learn processes. The simulation environment includes facilities for monitoring the execution of a business process and providing feedbacks for the learner evaluation.

The simulation framework distinguishes between the two following actors: *Civil Servant* and *Civil Servant Coordinator*.

The **Civil Servant** represents a generic participant to a business process simulation session. Learn PAd Simulation Framework provides to the Civil Servant a set of functionalities (see Figure 1):

- Join a current simulation session: Through this functionality, the civil servant can join an existing session to simulate a business process.
- Disconnect from a current simulation session: The civil servant can disconnect from a current session and reconnect after.
- Pause a current simulation session: The civil servant can pause the running of the current simulation.
- Play a current simulation session: The civil servant runs the simulation as an instance of a given business process.
- Answer forms: the civil servant answers forms when the Learn PAd Simulator notifies him/her that his/her turn comes.
- Chat: The Learn PAd platform offers to the civil servant three possibilities to chat: (1) with all other civil servants, (2) with selected civil servant(s) and (3) with task's experts available online.
- Ask for evaluation of the simulation session: When the simulation of the business process is finished, the civil servant may ask the Learn PAd platform for a global evaluation of the simulation session.
- Manage personal profile: the civil servant can manage his/her own profile by editing information and/or managing the avatar.
- Ask for static and log data: the civil servant can visualize statistics and logs of a simulation.
- Ask for help: The Learn PAd platform offers to the civil servant the possibility to ask for help by posting questions or also with the experts if they are online.
- Manage resources for the task's context: The civil servant can manage the resources related to the context of the task by adding a new one, liking it when he/she founds it interesting or simply read its content.

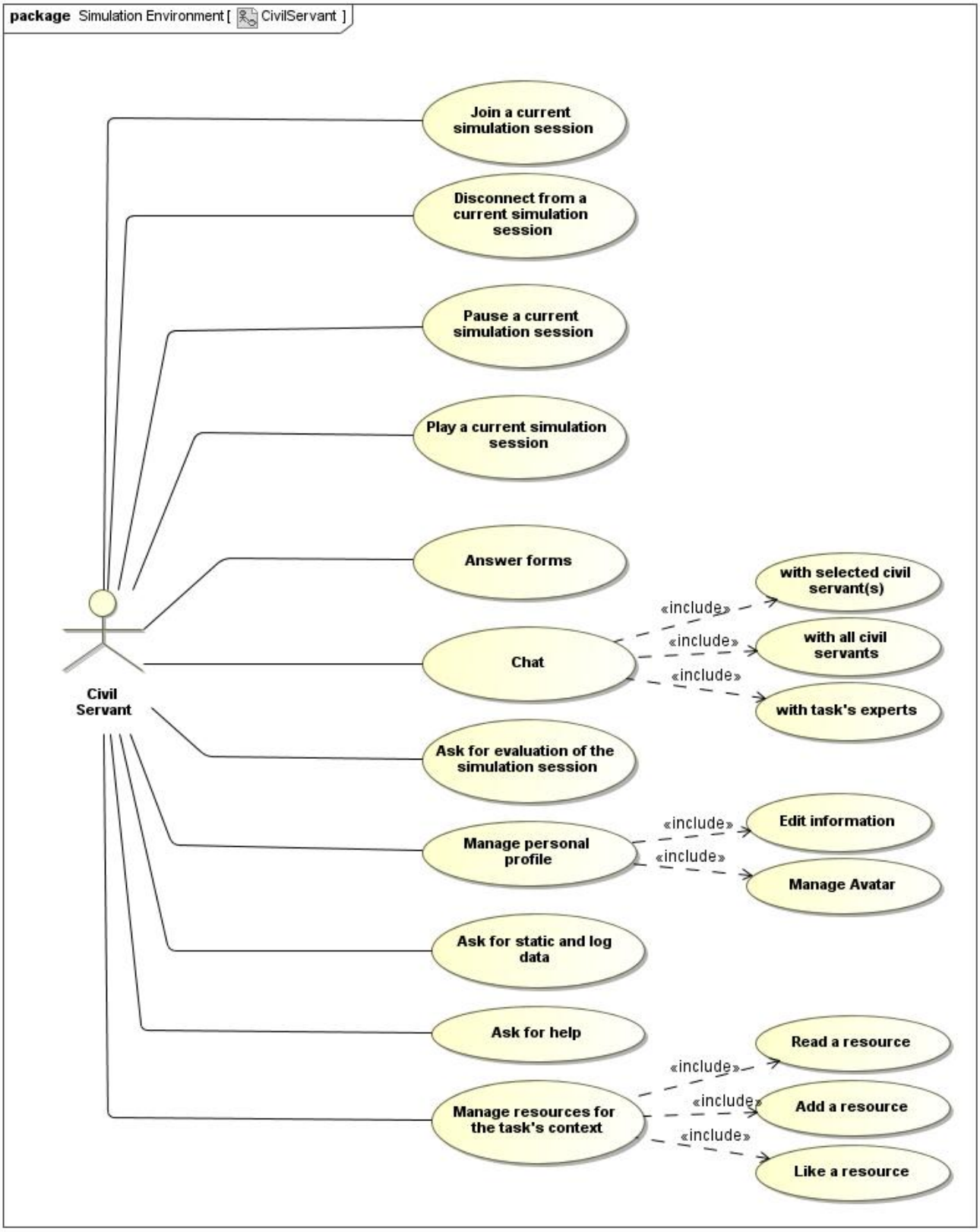


Figure 1 Civil Servant main functionalities

The **Civil Servant Coordinator** is, by default, the one who started the new business process simulation session. As depicted in Figure 2, in addition of the functionalities provided to the civil servant, the civil servant coordinator is responsible of managing the simulation session that he/she started. The civil servant coordinator can:

- Stop a current simulation: the civil servant coordinator can stop the running of a simulation.
- Restart a current simulation session: after stopping a simulation, the civil servant coordinator is in charge to restart its running.
- Invite civil servant(s) to join: when a collaborative simulation session is started, the civil servant coordinator is responsible to invite a civil servant for each role needed in the business process.
- Cancel civil servant invitation: the civil servant can cancel any invitation.
- Kick civil servant(s) from a current simulation session: the civil servant has the possibility to kick a civil servant who joins it.

Moreover, the civil servant coordinator may appoint another civil servant to be the new coordinator of the current simulation session.

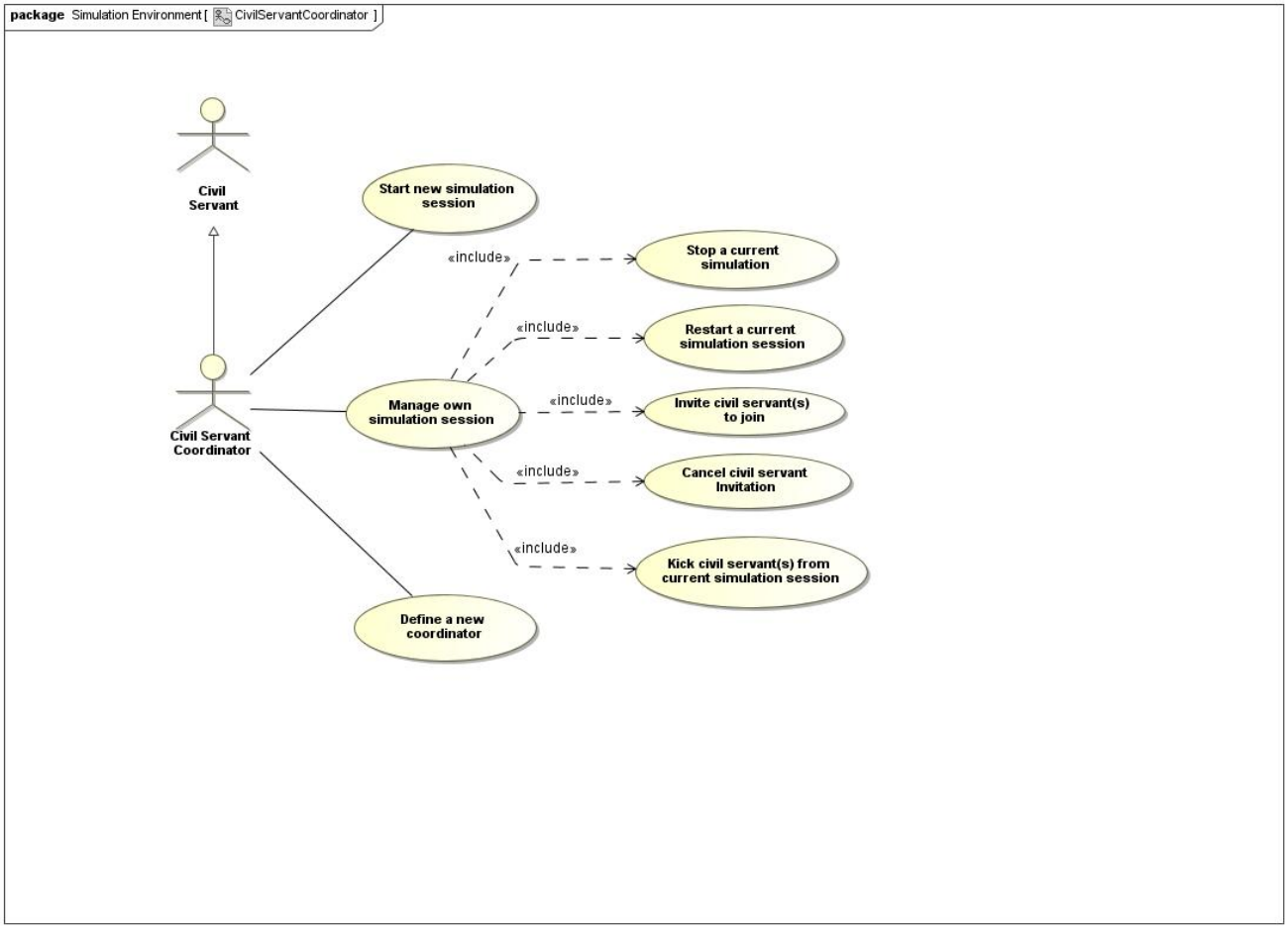


Figure 2 Civil Servant Coordinator main functionalities

During a simulation session the Civil Servant interacts with the Simulation Environment by means of Learn PAd Core Platform and Collaborative Workspace as detailed in UC3.8.1 of Deliverable 2.1.

In particular, with reference to Use Case 3.8.1 and 3.8.2 of deliverable D2.1 a more detailed description of the behavior and interactions of Learn PAd simulation and monitoring environment is provided.

The realization of the functional specification will be provided in terms of Sequence Diagrams and textual description in Chapter 3 while the architectural components belonging to the Simulation Framework will be detailed in Chapter 3.

3. Realization of simulation framework functional specification

In this chapter the functional specification of the simulation framework is presented in terms of Sequence diagrams and textual description. In particular the scenarios reported here refine the UC3.8.1 and UC3.8.2 of Deliverable 2.1.
The simulation framework functionalities have been split into three different phases:

- Initialization in which the simulation environment is set up.
- Activation in which the participants to the simulation are invited
- Execution in which the participants effectively collaborate each other during a learning session.

In the next sections further details about these three phases are provided.

3.1. Simulation initialization

In this section the main activities necessary to setup the simulation environment will be described. For improving readability of the diagrams in this document the overall simulation initialization sequence diagram has been split into two parts Figure 3, and Figure 4 respectively. Referring to Chapter 0 section 2.4, Figure 2, the simulation initialization is activated when a Civil Servant Coordinator executes the “Start new simulation session”. The scenario reported below describes the interactions of the simulation environment with the Learn PAd core platform.
In this scenario the architectural components involved will be:

- Core Facade and Bridge API: API of components that allow a tool to interact with the Learn PAd core platform. They provide the adaptation layer that must be implemented to integrate an existing system with Learn PAd. They are used by the Learn PAd core to communicate back to the tool specific events that might be interesting for it, or to invoke functionalities that these tools expose to the external environment (D2.1);
- SimulationEngineAPI: the subsystem where Civil Servants can simulate Business processes interactively;
- TestDataRepositoryAPI: The simulation environment database in which the information and data related to the simulation execution are stored.
- MonitoringInfrastructureAPI: API of the Complex Event Processor component, that is an event-based monitoring based on publish-subscribe paradigm. The Complex Event Processor is in charge to collect and correlate data useful both for the subsequent KPI evaluation and assessment of learning level/competency acquired by the Civil Servants participating in the simulation.

As shown in the diagram of Figure 3 the simulation initialization activity begins with the ActivateNewSimulation message sent in parallel through the BridgeAPI to the SimulationEngine and MonitoringInfrastructure. This causes initialization of the two components and the retrieval of required test data useful for the simulation (message n. 3) from the TestDataRepository. The historical data could be used also during the simulation execution for controlling and verify the responses of the Civil Servant(s). Then both SimulationEngine and MonitoringInfrastructure retrieve by means of the CoreFacadeAPI the necessary additional information useful for the simulation execution.

In particular data required include:

- the BP model to be simulated and the other associated models that the BP model refers to (message 4);
- the additional data and resources related to one or more models (message 5);
- the profile of a civil servant (message 6).

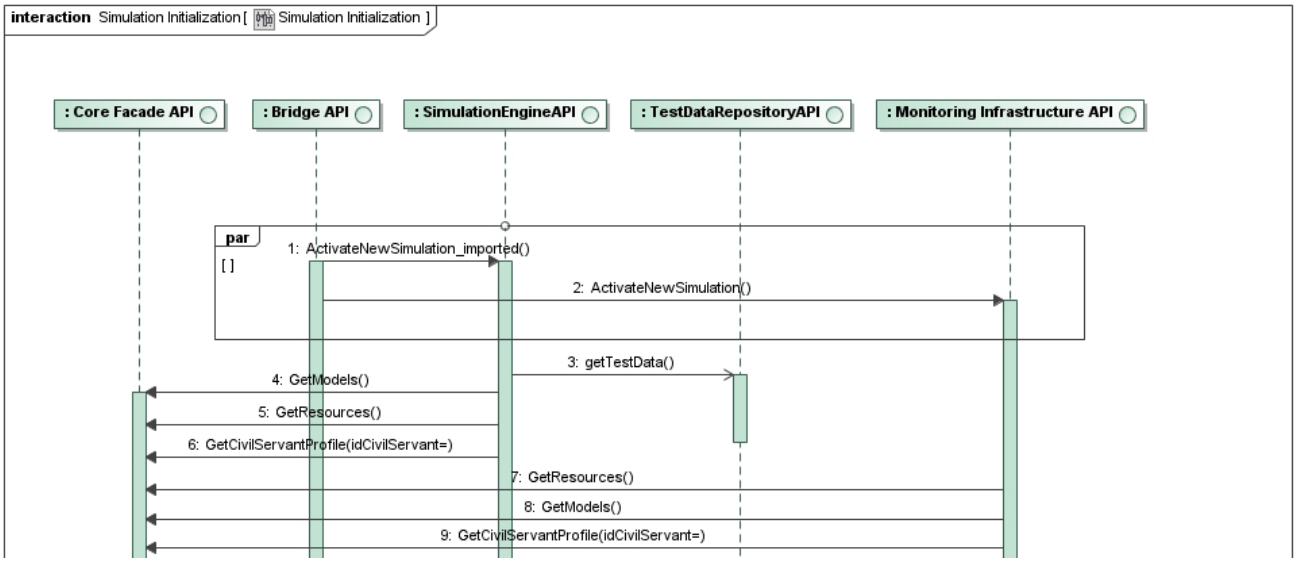


Figure 3 Simulation Framework initialization (1/2)

From the dedicated interface (BridgeAPI), the Learn PAd core platform will provide the model resources and others data useful to the simulation (messages 10-15).
The monitoring infrastructure initializes its components (message 16) and notifies to the SimulationEngine the end of the setup operations and the channel where to publish the events useful to the monitoring activities that will occur during the simulation (message 17). Then the SimulationEngine notifies to the Learn PAd core platform that it is ready to start the simulation (message 18).

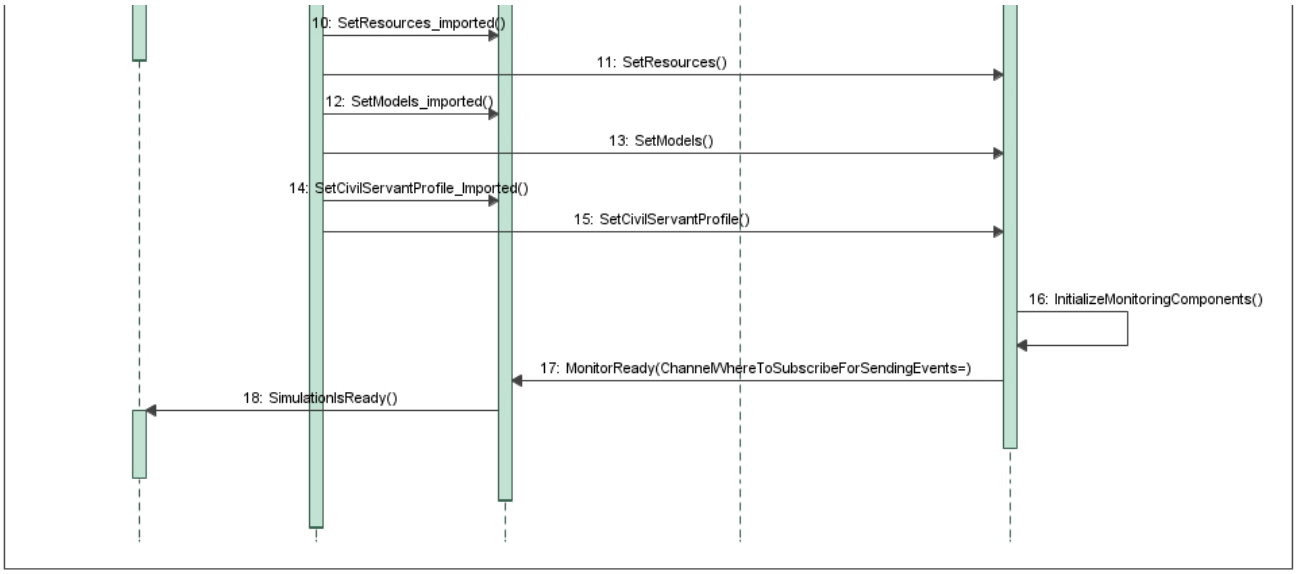


Figure 4 Simulation Framework initialization (2/2)

3.2. Simulation activation

In this section, the main steps needed to activate a simulation are described. For improving readability of the diagrams in this document the overall simulation activation sequence diagram has been split into different figures as will be detailed in the following.
The sequence diagram, presented in Figure 5 shows how a Civil Servant or a Civil Servant Coordinator can use the components of the Learn PAd environment in order to activate a simulation. Referring to Chapter 0 section 2.4 Figure 2 the Civil Servant Coordinator is

attributed to the Civil Servant who activated the new simulation. In particular he/she is in charge of executing the functionality called “Manage own simulation session”.
This scenario involves all the architectural components of the simulation initialization scenario plus the following components:

- SimulatorGuiAPI: which is the interface by means the Civil Servants can interact with the simulation framework;
- RobotFrameworkAPI: which is the interface for requesting the generation of specific services able to emulate human activity during the simulation.

The first interaction of the simulation activation scenario consists on a notification to the SimulationEngineAPI through the Bridge API, that a new simulation can start (message 1).
All the simulation participants are then notified of the beginning of a simulation section by means of the SimulationGuiAPI (message 2 in Figure 5). Through the same interface each participant to the simulation first defines his/her role (message 3). Then, he/she selects the type of simulation to execute (message 4).

At this stage three options are given to the Civil Servant Coordinator who can select the type of simulation he/she wants to execute: **individual**, **collaborative** or **mixed**. Hereafter the interactions between the components of simulation framework necessary for implementing these option are summarized.

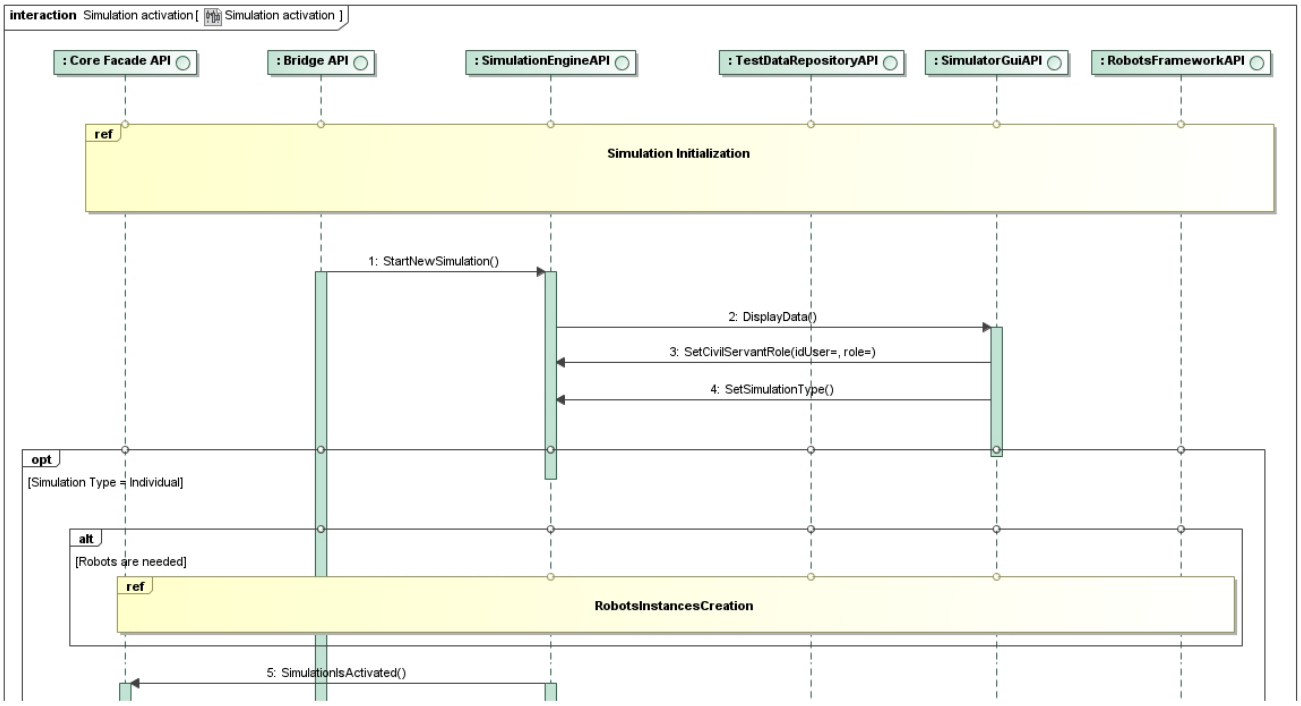


Figure 5 Simulation activation (setup and individual option) 1/3

- **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 specific services called Robots (see section 4.2 for more details). The creation of robots instances is performed before the simulation execution, therefore (as shown in the Figure 6) a request is sent to the RobotsFrameworkAPI to create the required instances.

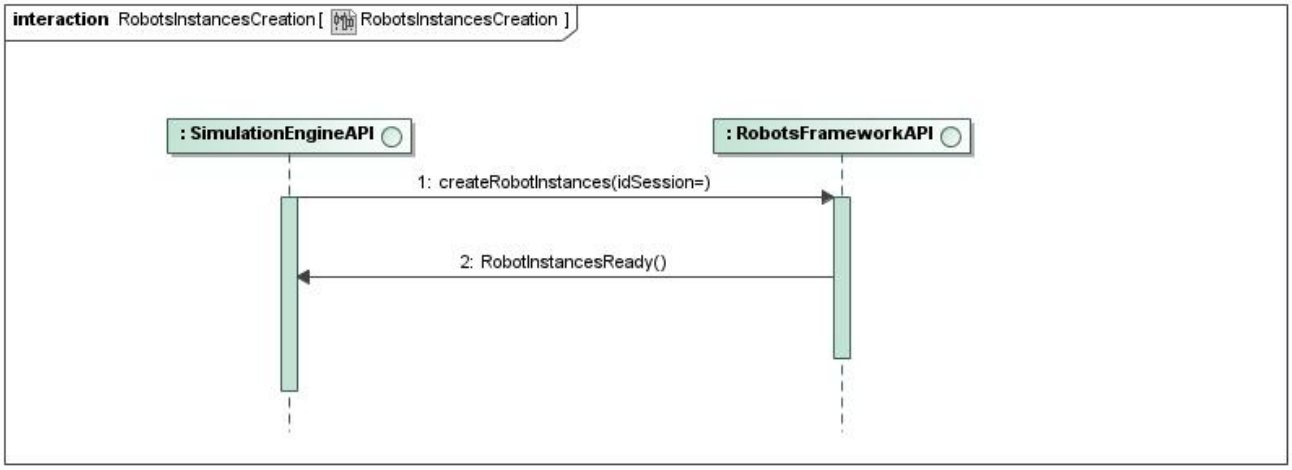


Figure 6 Robot Instances Creation

Finally, the SimulationEngineAPI notifies the Learn PAd core platform that the simulation can be executed (message 5).

- **Collaborative Simulation:** this option of simulation involves the collaboration of several human participants (no robots instances are involved) so to cooperate each others for the completion of the execution of process and improve own learning level. 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 cooperating 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 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. Even though the simulation framework supports also some asynchronous tasks execution between simulation participants this is not the main purpose this architecture.
  According to Chapter 0, Section 2.4, Figure 1 during the simulation, 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 he/she with another one among those available, or replace he/she with a Robot.

This scenario is represented in the Sequence Diagram of Figure 7 that starts with the Civil Servant connection procedure.

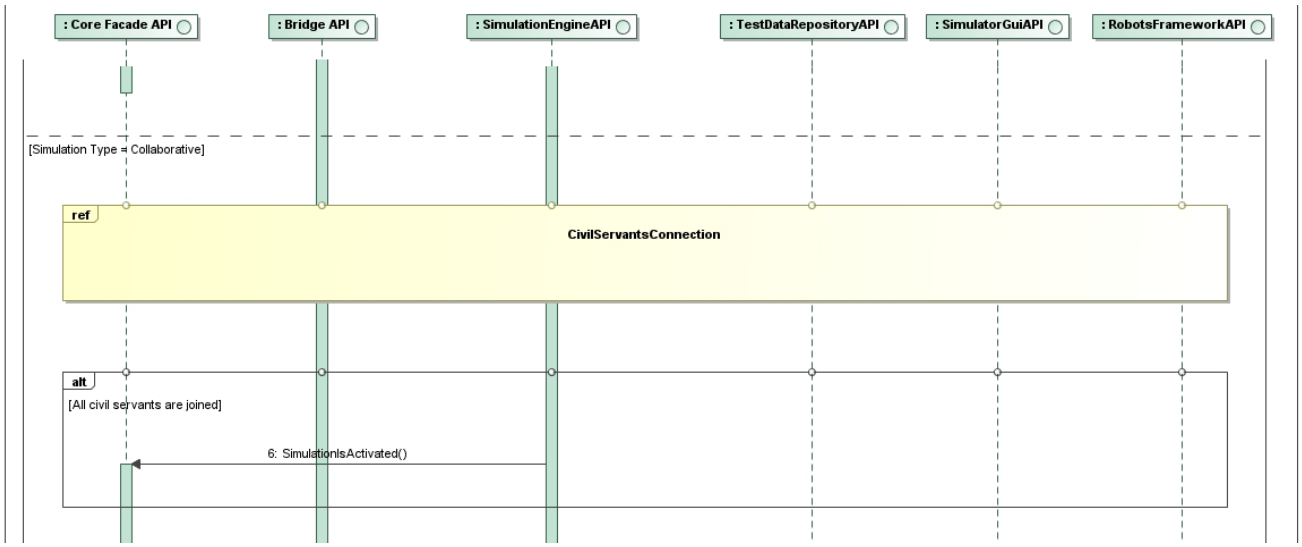


Figure 7 Simulation activation (collaborative) 2/3

Considering in particular the Civil Servant Connection procedure, as show in Figure 8, the SimulationEngine gives to the Civil Servant Coordinator the possibility to invite other participants by means of the interaction with the SimulatorGUIAPI (message 1). At this point the Civil Servant Coordinator, sends through the SimulatorGUIAPI the invitation requests to all the Civil Servant that he/she wants to join the simulation (message 2). Then the SimulantionEngine receives the reply to invitations (message 3) and consequently the SimulationEngineAPI and the MonitoringInfrastructure retrieve the profiles of the Civil Servants who will join the simulation (messages 4 and 5).

Here a profile is the set of data that defines the characteristics of the invited Civil Servant such as the skills and competency level.

As in Figure 8, if an invited Civil Servant is become currently not available or he/she leaves the learning session (disconnection) the Civil Servant Coordinator can either send an invitation to others available Civil Servant or decide to replace it with a Robot (in this case, the simulation turns on a mixed simulation as will be detailed here below).

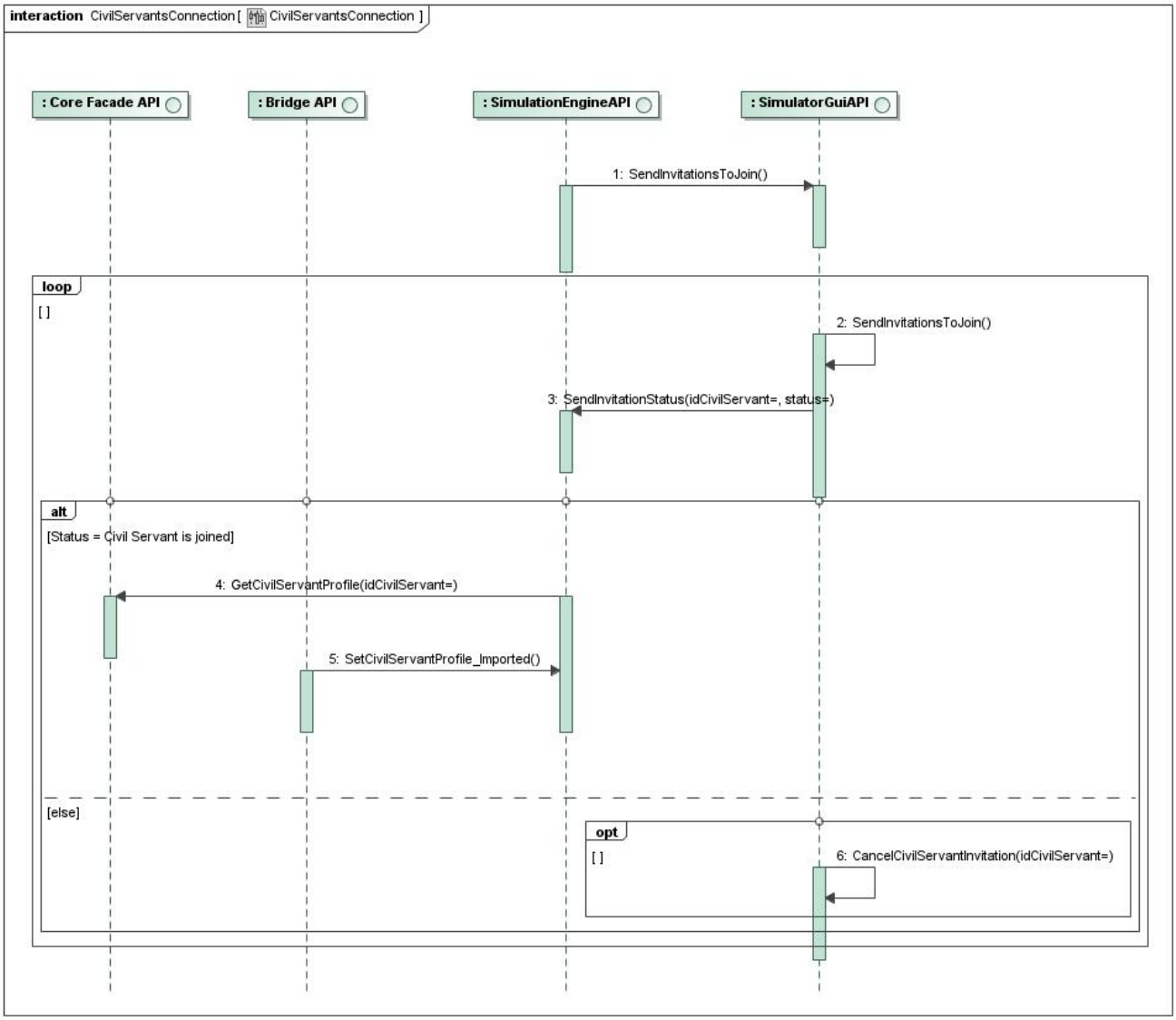


Figure 8 Civil Servant Connection

- **Mixed Simulation (**Figure 9**):** 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).
  During the mixed simulation the simulation framework loops until the following conditions are not satisfied (message 7):
  - The required instances of robots are ready (see Figure 6 for Robots creation procedure);
  - All the invited Civil Servants have completed the connection procedures.

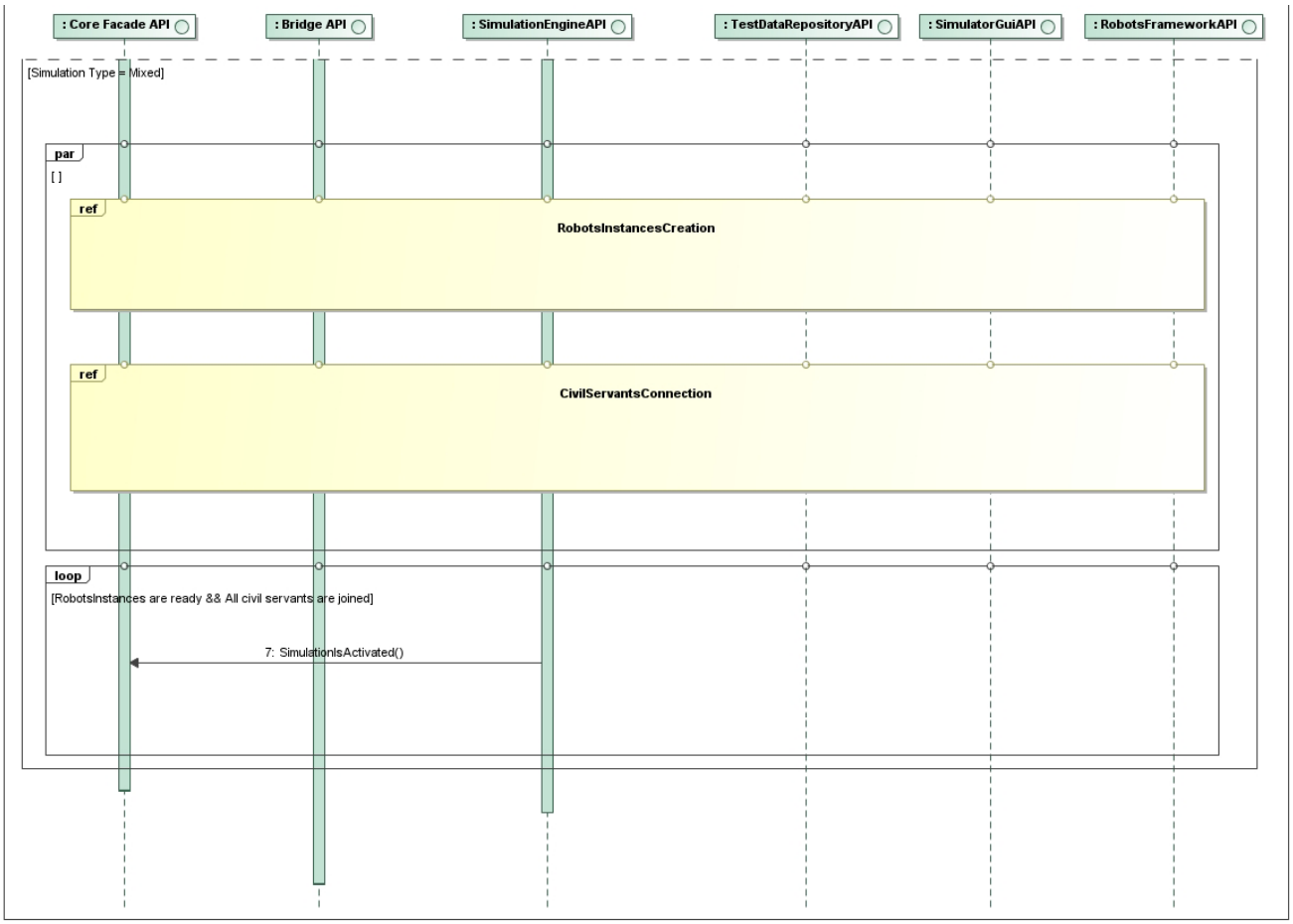


Figure 9 Simulation activation (mixed) 3/3

3.3. Simulation execution

This section describes the main activities of the simulation execution shown in Figure 10. Referring to Chapter 0, Section 2.4, Figure 1 the simulation is activated when a Civil Servant executes the “Play a current simulation session” Use Case.

This scenario involves all the architectural components of the simulation initialization and activation plus the following components:

- CepAPI: which is the interface for interacting with the kernel of the monitor component so to let analysis and evaluation of different events occurring during the simulation.
- ResponseDispatcherAPI: which is the interface by means the monitor component interacts with the PersistencyLayer.
- PersistencyLayerAPI: is the interface for interaction with the simulation framework repository containing all the information relative to the simulations execution.

The Sequence Diagram of Figure 10 refers to the scenario detailed in Section 3.2 as a precondition, i.e starts when the simulation activation phase is completed. At this point the SimulationEngine generates the events related to the execution of tasks of the Business Process (message 1) and sends them to the CepAPI (Complex Event Processor API) component of the Monitoring infrastructure for their evaluation (message 2).

The simulation execution continues managing the BP tasks and sending notification to the Civil Servants by means of SimulationGuiAPI (first *opt* interaction operator reportend in

Figure 10) till a special kind of event is received. In the simulation executions four types of special events are managed:

- END: risen when the BP execution reaches the natural end of the model.
- STOP: risen when the Civil Servant Coordinator decides to early terminate a simulation execution.
- PAUSE: risen when a Civil Servant decides to pause the simulation execution.
- PLAY: risen 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. This is in charge of evaluating and inferring complex events and stores the results of the monitored session analysis to the PersistencyLayerAPI through the ResponseDispatcherAPI.

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.

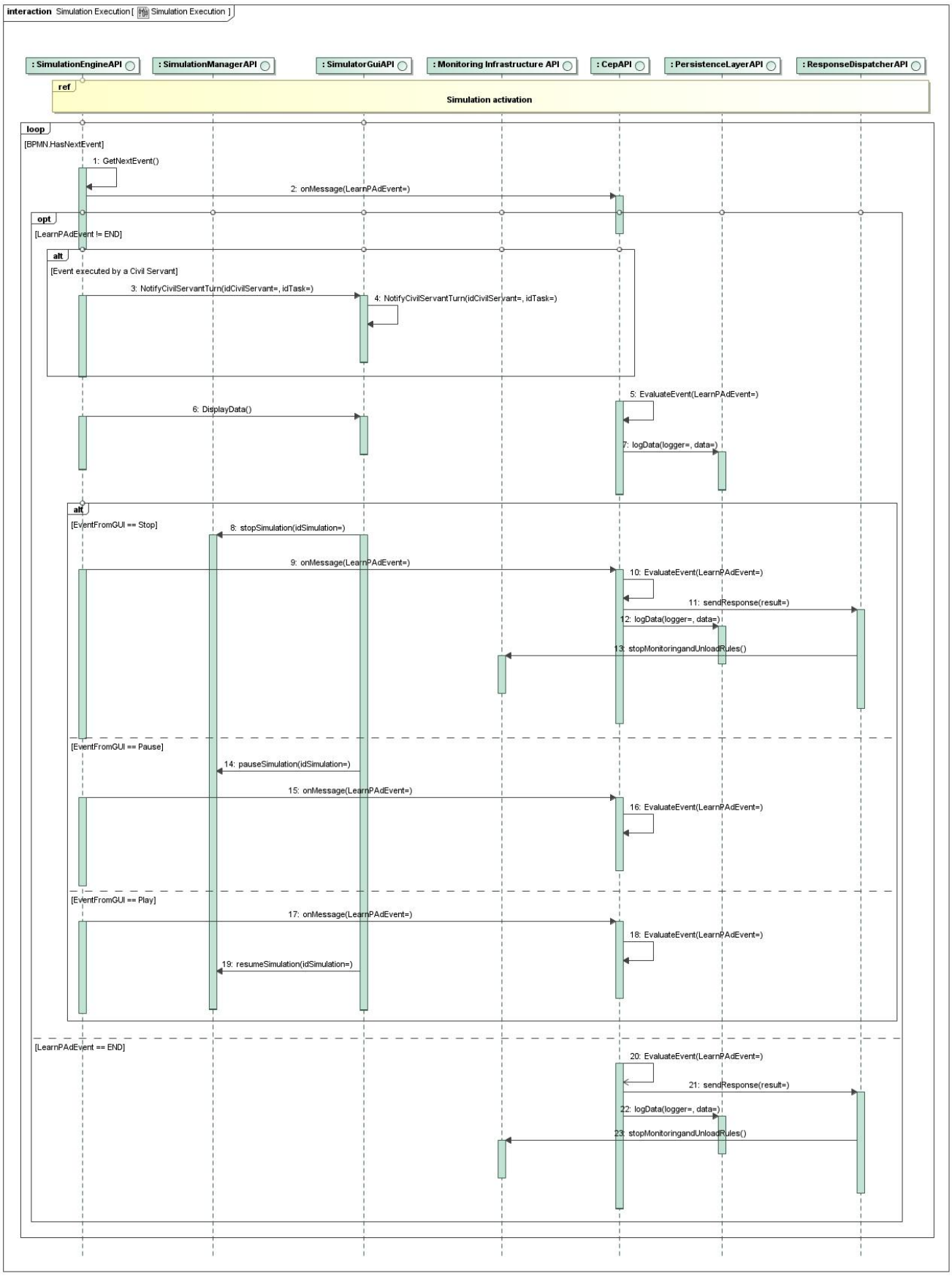


Figure 10 Simulation execution

4. Simulation Environment architecture

This section describes the high level architecture of the Learn PAd Simulation Environment, its main components, their purpose, the interfaces they expose, and how they interact with each other. In particular, Figure 11 shows the high level architecture in terms of a component diagram where the main simulation environment components are represented together with their interfaces.

In the rest of this section, a more detailed description of these components in terms of sequence and component diagrams is provided as well as the specification of the interfaces and their interaction by means of a communication middleware. Moreover, for each component, the description of its behavior and its main structure will be provided. Foreseen related technologies are also described in the corresponding paragraphs.

Each component is exposed as a service and provides an API as a unique point of access.

In particular, the Simulation Environment interacts with the rest of Learn PAd components by means of the Learn PAd Core Platform and specifically through the Bridge and the Core Facade interfaces. More details about this interaction are provided in Section 3 (System Architecture) of Deliverable 2.1.

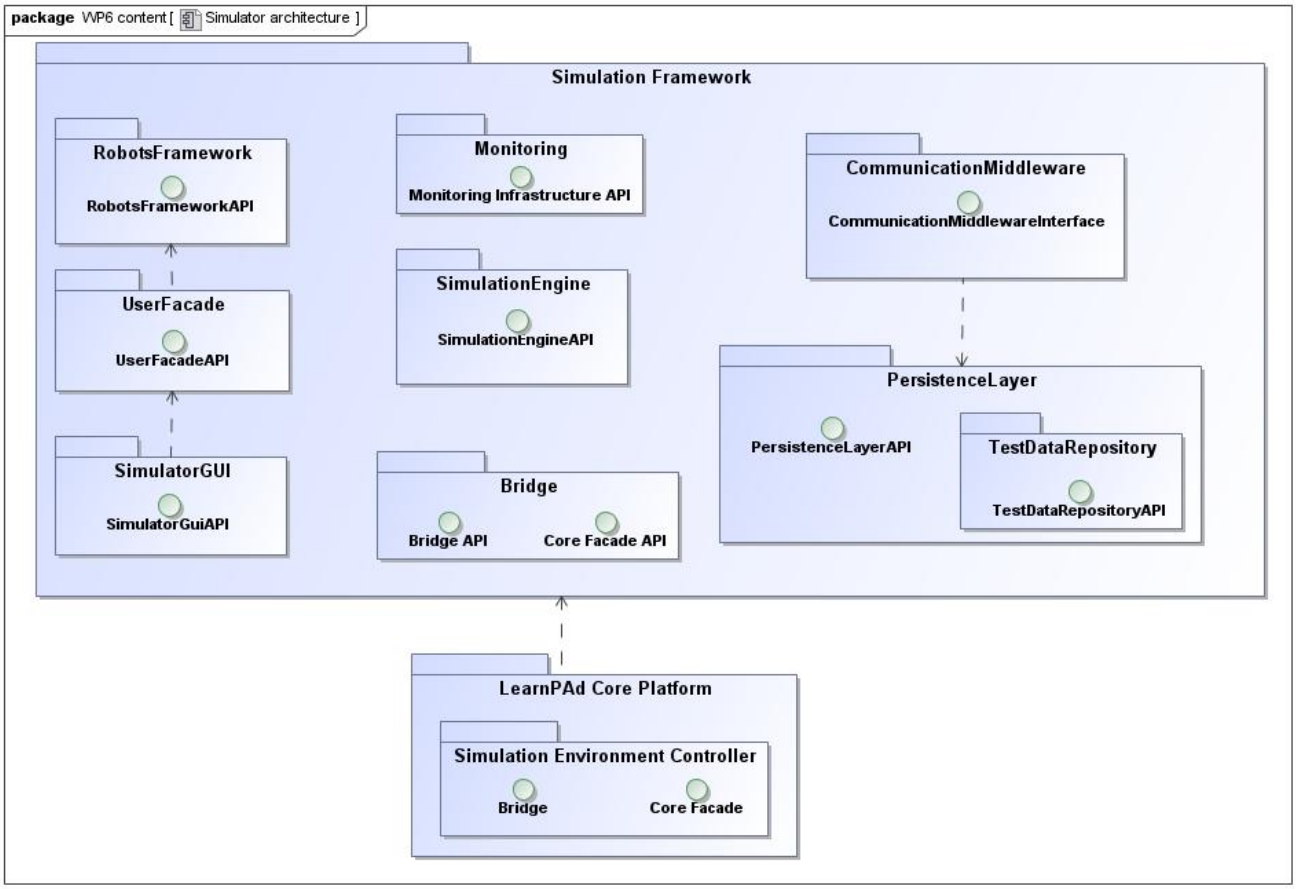


Figure 11 Simulation Framework architecture

As highlighted in Figure 7, the Simulation Environment components are:

- SimulatorAPI: API of the simulator. This API is used by external components in order to access simulator's features.
- PersistenceLayer: this component is in charge of the persistency required by the simulator components as the data storage, the indexing and the retrieval. It stores

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

- Communication middleware: this component provides event based communication facilities between the simulation components according to the publish / subscribe paradigm.
- SimulationGUI: this component is in charge of the interactions between learners and simulator's components.
- SimulationEngine: this is the core component of the Learn PAd simulator. It enacts business processes and links activities with corresponding civil servants or robots.
- RobotFramework: this framework allows, when necessary, to simulate the behavior of a civil servant.
- Monitoring: this component collects the events occurred during the simulation and infers rules and properties related to the business process execution.
- UserFacade: this component is in charge of encapsulating real or simulated civil servants (i.e. robots) in order to make the user interaction transparent to the other components of the architecture.

In the next sub-sections, more details about each of these components are provided. A discussion about possible technologies is also provided.

4.1. GUI

Simulator Graphical User Interface is in charge of the interactions between the civil servants and the simulator's components. The aim of this part is, in one hand, to detail the content of the learner interface layouts and, in a second hand, to provide information about technologies.

As explained before, a Learn PAd Business Process simulation may 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.

GUI mockups are provided in the following Section 4.1.1

However, the main features of the user interface are highlighted in Figure 12 below.

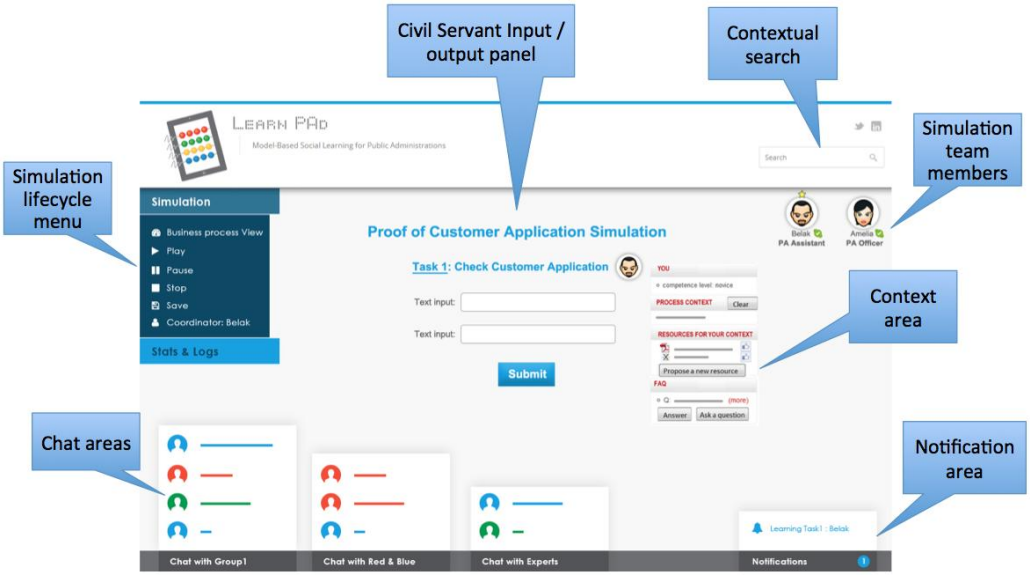


Figure 12 Main user interface

The **Simulation Lifecycle menu** allows the learner to choose among:

- Business process view: to obtain a graphical representation of the current business process.
- Play: to run a simulation as an instance of a given business process.
- Save: to save the current simulation. The Civil Servant can exit the simulation and restart the simulation from the point in which it has been saved.
- Pause: to pause the running of the current simulation.
- Stop: to stop the current simulation and exit. Learner is asked whether he would like to save before exiting.
- Coordinator: provides the name of the coordinator of the current session and allows him/her to modify it. Only the Civil Servant Coordinator has the possibility to designate another Civil Servant to be the new coordinator of the simulation session.
- Stats&Logs: using an analytic dashboard, allow the learner to display statistics and logs about all activities that carried out by the simulator engine as well as all the learner's provided information.

Concerning the other facilities of the GUI, here below a brief description is provided:

- **Chat areas**: spaces for learners to chat either one by one, one with the group of Civil Servants or with experts connected to the current simulation. In particular the chat areas will have a crucial role in case of interaction within an expert.
- **Notification area**: provides notifications to the learner.
- **User Input / output panel**: this area contains forms for learners interactions i.e. to send or request him/her data or information.
- **Context area**: provides the document related to the simulation, additional information or links to material that may be useful during the simulation activity.
- **Simulation teams members**: learners involved in the simulation have a special placeholder so to distinguish them from the coordinator.
- **Contextual search**: allows to search among different kinds of information depending of the current displayed layout (users, processes, simulations, etc.).

4.1.1. GUI Mockups

Referring to the example, provided in D5.1, several GUI mockups that have been designed to illustrate the simulation are presented.

Before starting the simulation phase, the Civil Servant, named Belak, needs to choose between two options of simulation (see Figure 13). The first one is about starting a new simulation of a business process and the second option offers the possibility to continue a previous simulation from a saved point.

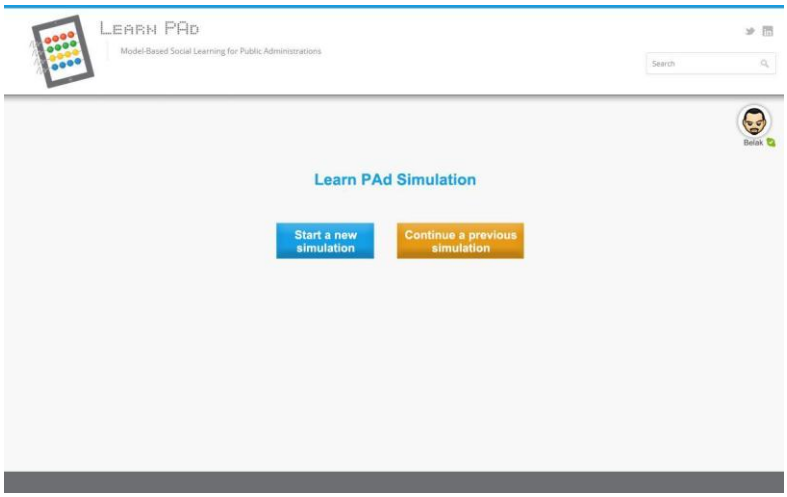


Figure 13 Learn PAd Simulation entry point

Then, the Civil Servant chooses to start a new simulation. Consequently, Belak is the coordinator and, as shown in Figure 14, he has a star above his avatar to distinguish him from the other Civil Servants.

As a first step of the simulation, Belak has to provide his role, to mention his level of competence, to select the business process to simulate and to choose the type of simulation that he wants to run. In our example Belak chooses to run a collaborative simulation.

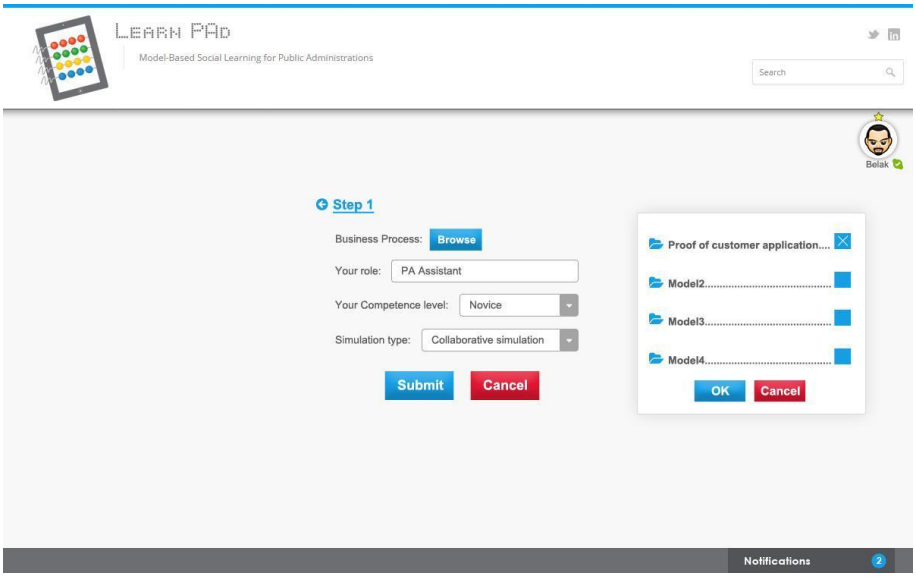


Figure 14 Learn PAd Simulation – Step 1

As a second step of the collaborative simulation's execution, Belak has to invite the other Civil Servants needed to simulate the BP. To do that and as detailed in Figure 15, he has only to enter for each Civil Servant his email and the role he/she will play in the simulation.

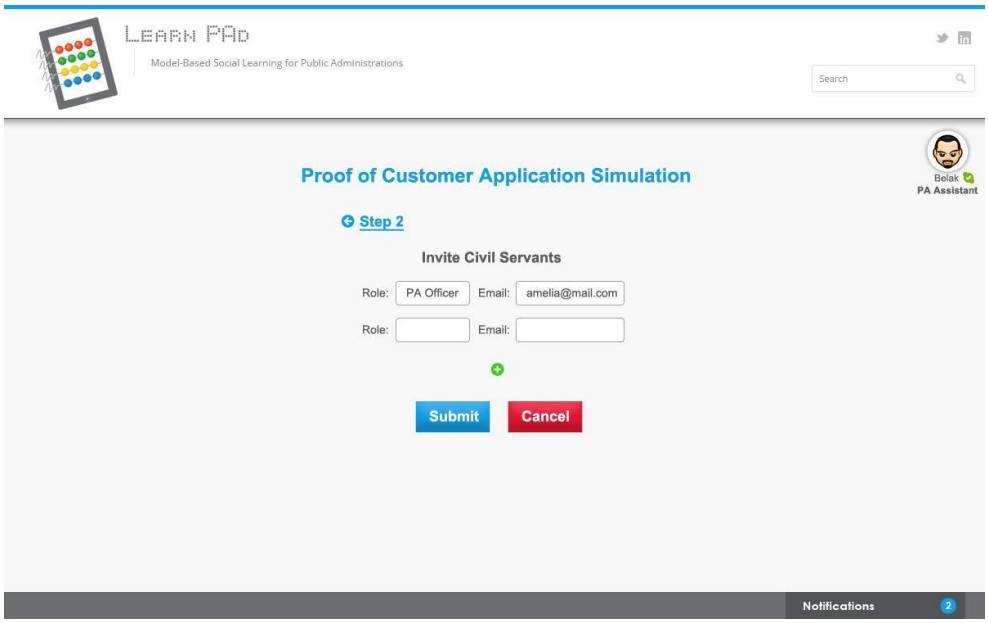


Figure 15 Learn PAd Simulation – Step 2

Once the invited Civil Servant (Amelia) joined the simulation platform, the simulation is activated. The Civil Servant Coordinator selects the difficulty level of the simulation: elementary, intermediate or advanced (see Figure 16) and then the simulation of the business process is ready to be executed.

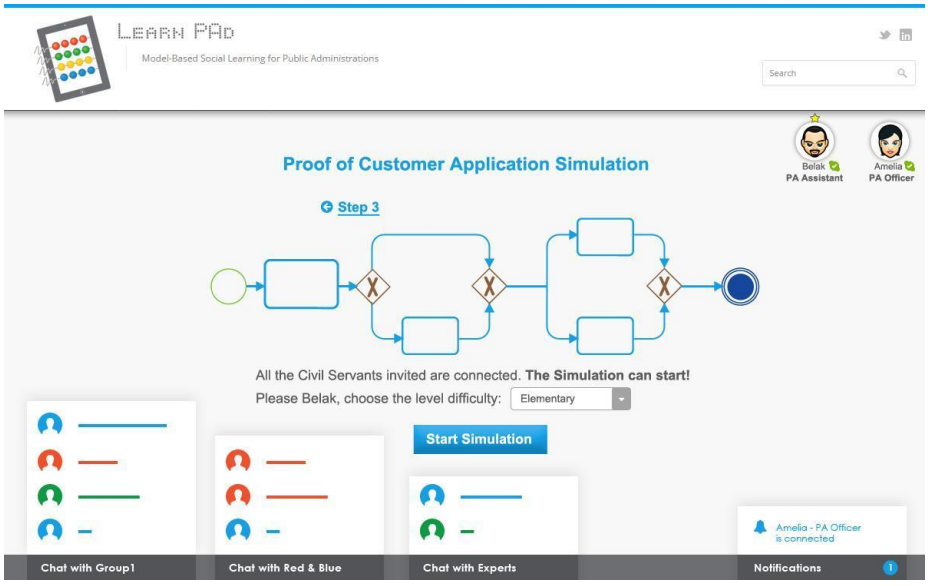


Figure 16 Learn PAd Simulation – Step 3

Then, as detailed in Figure 17, for each task of the business process and until the end, the Simulation Platform notifies the Civil Servant when his/her turn to learn comes and presents the corresponding form related to the business process task. The Civil Servant fills this form and submits his/her inputs to the Learn PAd platform. Moreover, as shown in the Figure beside the form, the Learn PAd platform 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. In addition, the simulation platform presents in the Process Context area a short summary of the test data with which the current simulation is played.

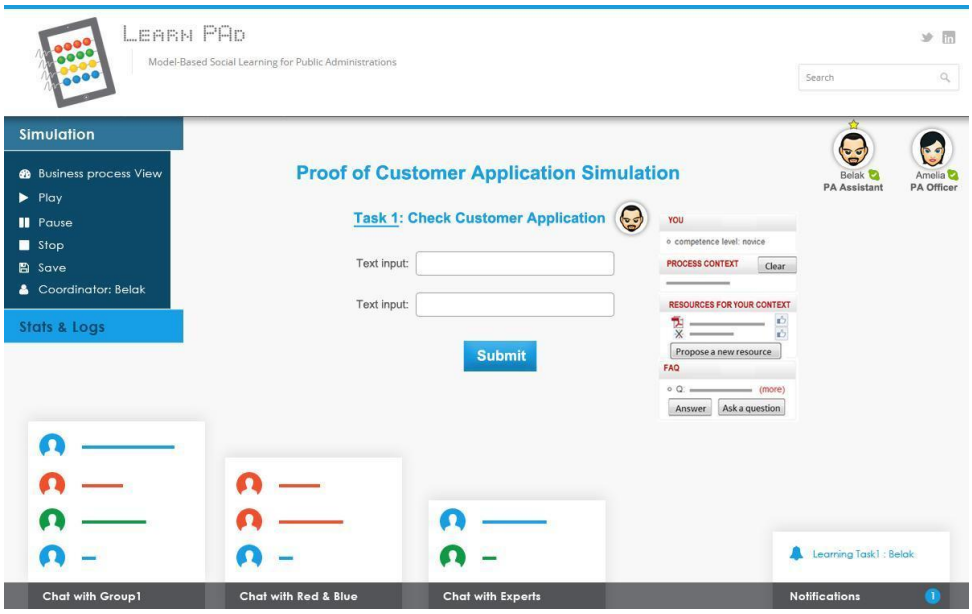


Figure 17 Learn PAd Simulation – Performing forms

Finally, when all tasks of the business process are simulated and the simulation is finished, Learn PAd simulator platform offers the possibility to the Civil Servant Coordinator and to each Civil Servant participated into the simulation to receive an evaluation of the simulation (see Figure 18).

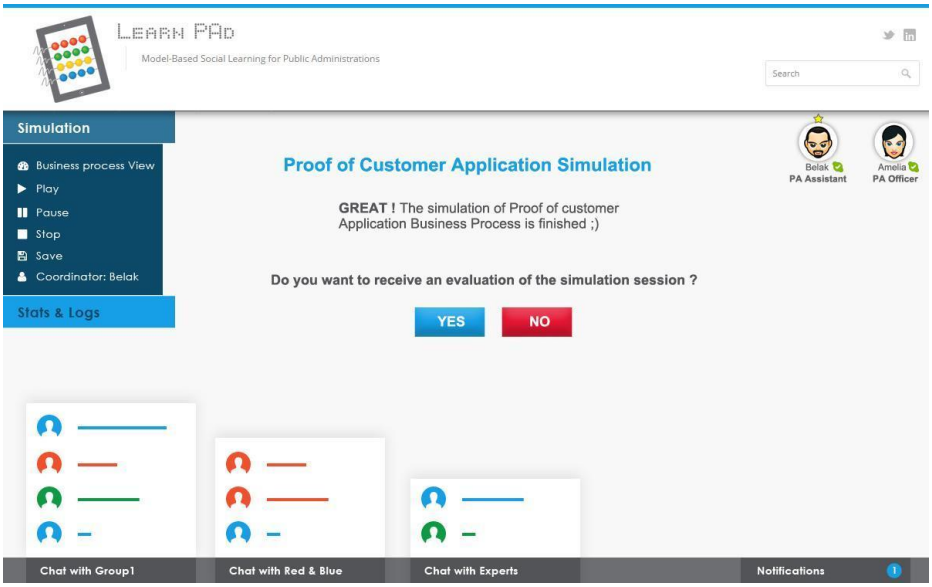


Figure 18 Learn PAd Simulation – End of the simulation

4.1.2. GUI technological choices

One of the main requirements dealing with GUI technology is compatibility with the XWiki platform that is a web application. Consequently, simulation GUI should also be a web application. The main available technologies for developing Rich Internet Applications are Google Web Toolkit (GWT) and AngularJS briefly described below:

- **GWT** is an open-source set of tools that allows developers to create and maintain complex JavaScript applications in Java. Google GWT was first released on May 16th, 2006. However, Generating JavaScript is not enough anymore especially in the context of multi-devices applications.

- **AngularJS**, commonly referred to Angular, is an open-source web application framework to address many of the challenges encountered in developing Single-Page Applications (SPA). Its goal is to simplify both development and testing of such applications by providing a framework for client-side Model–View–Controller (MVC) architecture, along with components commonly used in rich internet applications.

In Learn PAd project the Angular has been chosen for two reasons: from one side the very large community of users and developers working with this application can be useful both for solving possible issues and integration challenges; from the other the deep experience and knowledge in the use of this application from some Lean PAD partners (like XWiki and Linagora) can strongly speed up the development process of this critical part.

4.2. Robots Framework

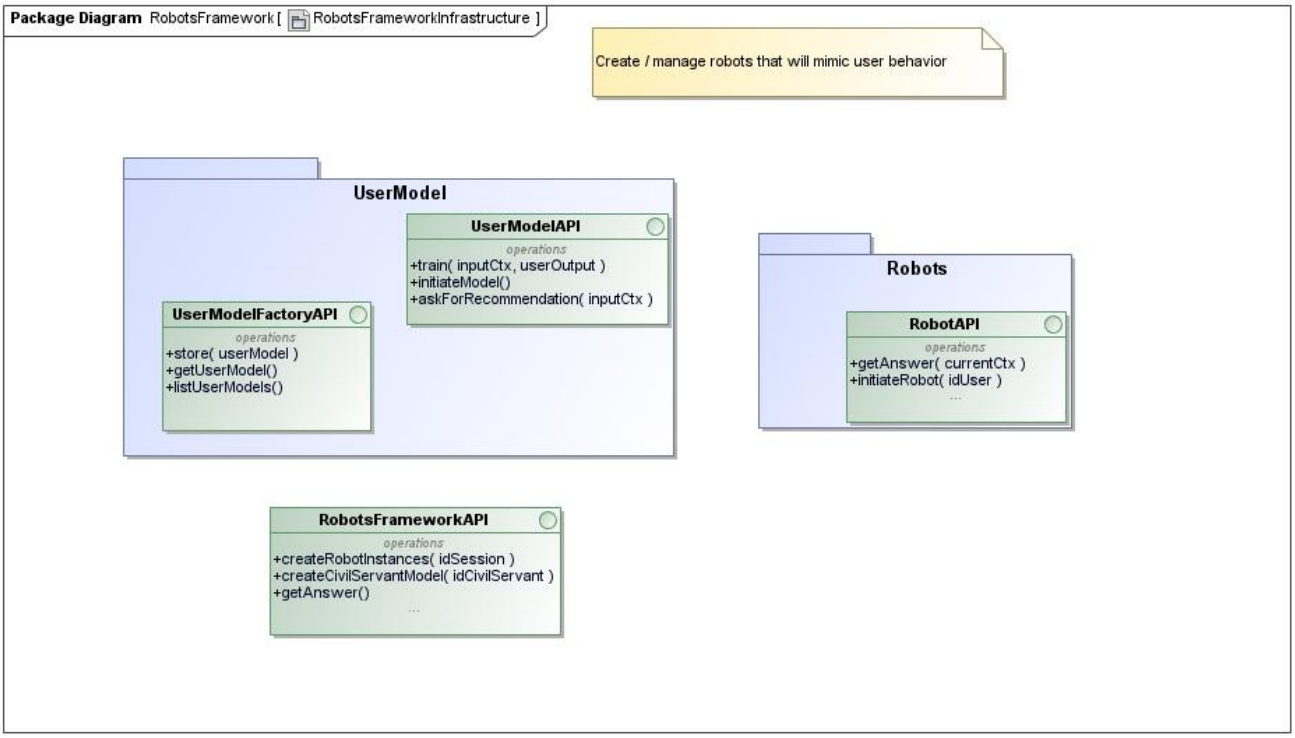


Figure 19 Robots Framework

Business Process simulation for Learn PAd platform involves civil servants taking different roles according to the content they have to learn. During such simulation some of the required civil servants may not be present. Robots are software entities that mimic missing learner's behavior during the simulation. They are aimed to provide outputs that would have been produced by a human with the same input in the same context.

The main components of Robots framework are shown in Figure 19. They will interact so to implement robots behavior. In particular the interaction will depend on the availability of what is called *Historical data.* In particular historical data represents the data saved in the TestDataRepository during a previous simulation sessions. They record information provided by an expert that took the role of the civil servant. In particular the situations considered are:

- when historical data are available in the TestDataRepository they are used to deduce the correct behavior of the simulated learner.

- in case of unavailability or incompleteness of these data, an expert may be asked to provide historical data manually.
- in case of the unavailability of both historical data and expert, a knowledge based agents approach able to deduce output from the current situation could be investigated. For such schema, a machine learning framework will be designed to implement Learn PAd robots. It will be based on learning algorithms applied to business processes logs: the business process simulator provides logging facilities that store information about “real learners” behavior.

The approach that will be investigated in the Learn PAd project will consist into two main steps:

- during “standard” simulation, namely when the real learner is present, the simulator logs contextual information as well as data provided by the learner. This information is used to learn the civil servant behavioral model that is a part of the robot.
- during simulation involving robots that replace the real learners, the previous model is used in order to recommend an answer.

It is important to notice that in case a machine learning approach is used, the suggested answer may not be the most suitable one since the learning approach is based on previously collected answers provided by learners. The current learner will be aware of this risk.

4.2.1. Robots technological choices

The Robot Framework will be implemented as a set of Java classes. The Facade and Proxy design patterns will be used as depicted in Figure 20 to encapsulate learner/robot behavior. The UserFacade will delegate either the civil servant or the robot proxy depending on the presence of the “real civil servant”.

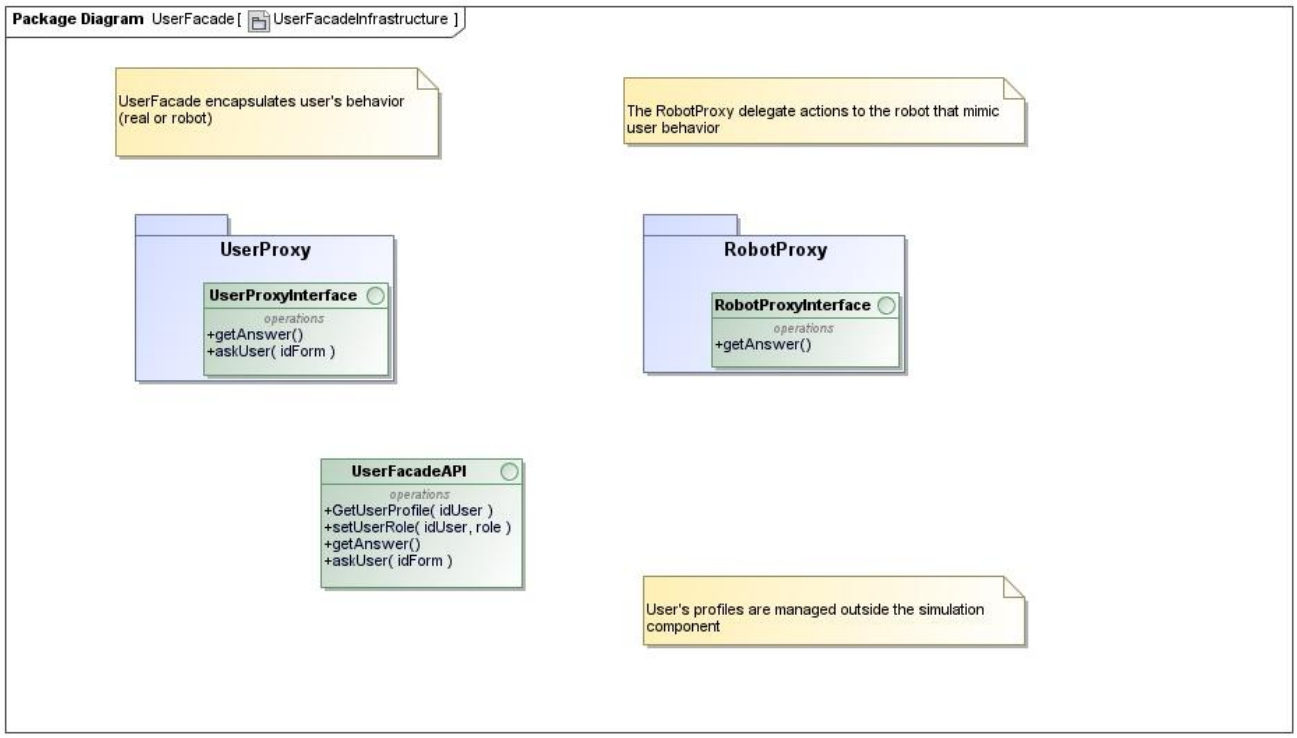


Figure 20 Robot Framework Implementation

4.3. Simulation engine

The 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 connected previously, described robots are used in order to mimic their behavior.

Main components of the engine are described in Figure 21.

A simulation manager is provided in order to manage Business Processes' life cycle according to the current context (create, stop, resume, kill, etc.).

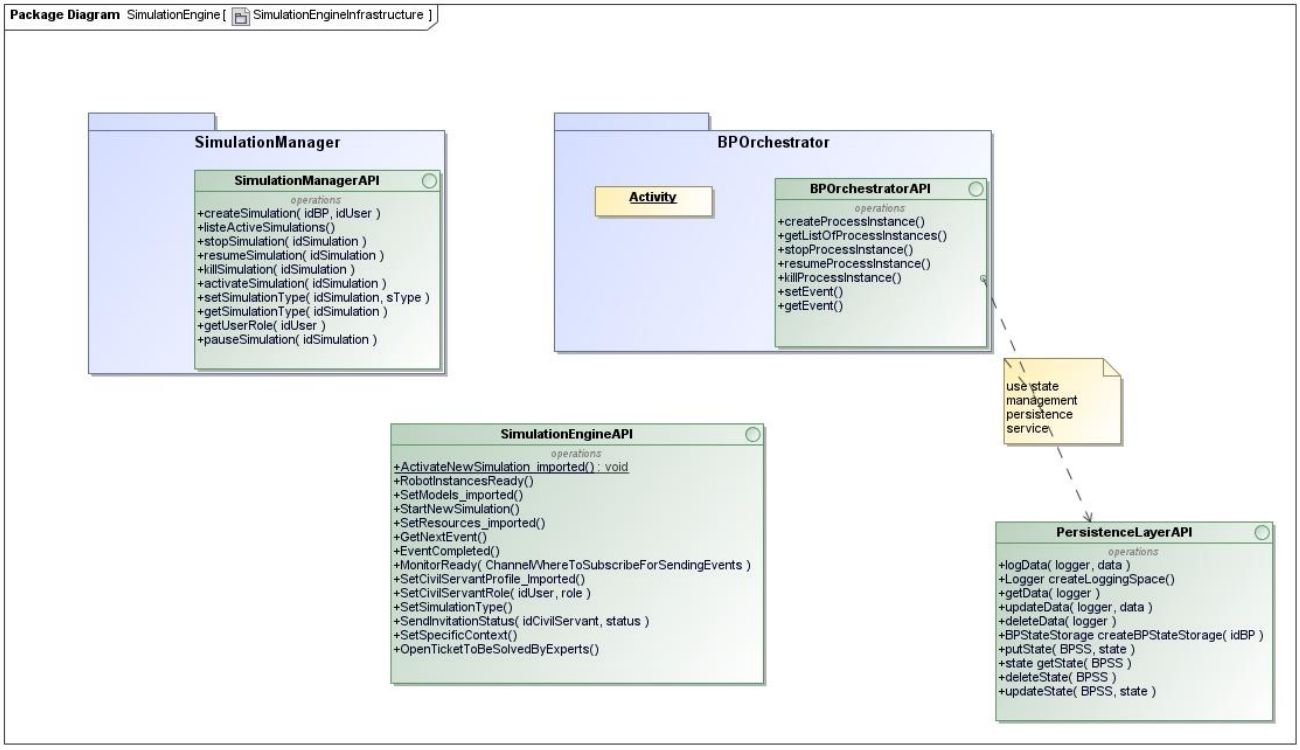


Figure 21 Simulation Engine Infrastructure

Business processes are made of two kinds of activities:

- Human activities involve learners (civil servants) that 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.
- Mocked activities involve software application to compute the treatment associated to the activity. Software mocks of those applications are provided in order to complete simulation.

When the engine invokes a human activity the corresponding learner (civil servant) is asked to provide input through a form. Those forms are managed by a form engine that delegates tasks 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.

4.3.1. Business Process orchestrator

Business Process orchestrator takes in charge the step by step execution of a given Business Process 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 orchestrator is connected with the Forms Engine in order to take in charge users and robots input/output.

An open-source BPMN 2.0 compliant BP Engine should be selected for this purpose. Available solutions are either Activiti[1] or Camunda[2] or jBPM[3]; all of them are available under Apache 2.0 Licence. In Learn PAd project, Activiti has been selected because it has a broad coverage of the BPMN 2.0 standard.

In order to collect inputs from learners during a simulation session, a form engine has been defined so to design and run the proper corresponding forms. Details of the form engine are provided below.

4.3.2. Forms Engine

Forms Engine allows dynamic forms creation and complex forms processing for web applications. The processing of a form involves the verification of the input data, calculation of the input based on the information from other input fields as well as dynamic activation or hiding of the data fields depending on the user input. Inside the Learn PAd project the javascript Form editor, called FormaaS, will be adopted and revised. FormaaS has been developed by Linagora, it allows to design and run javascript forms and to quickly define forms and executable code.

The following pictures (Figure 22 and Figure 23) give screenshots of the FormaaS editor and corresponding form.

[1] http://activiti.org

[2] http://camunda.org

[3] http://www.jbpm.org

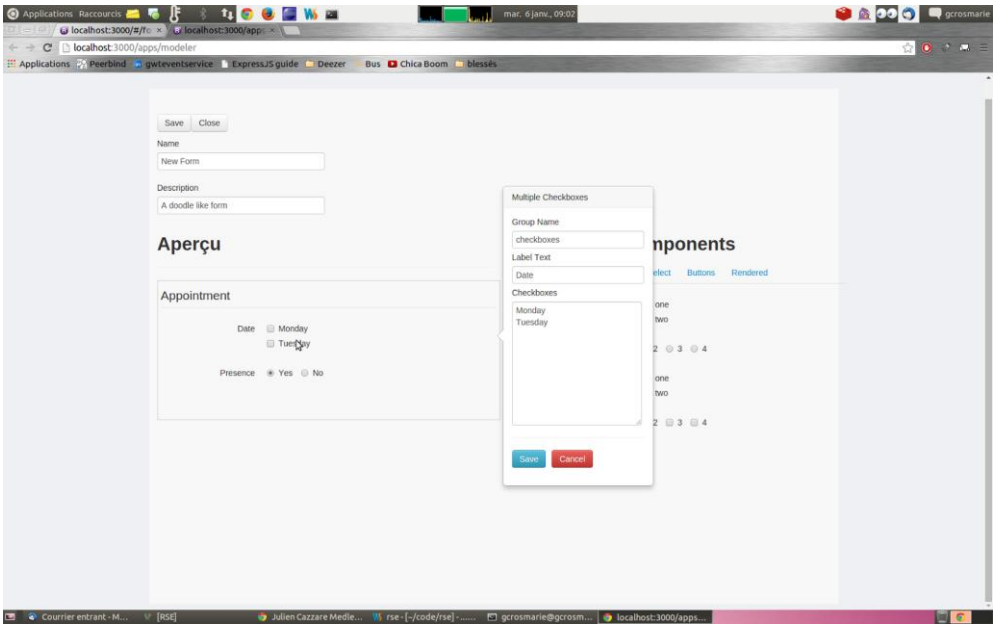


Figure 22 FormaaS Editor screen

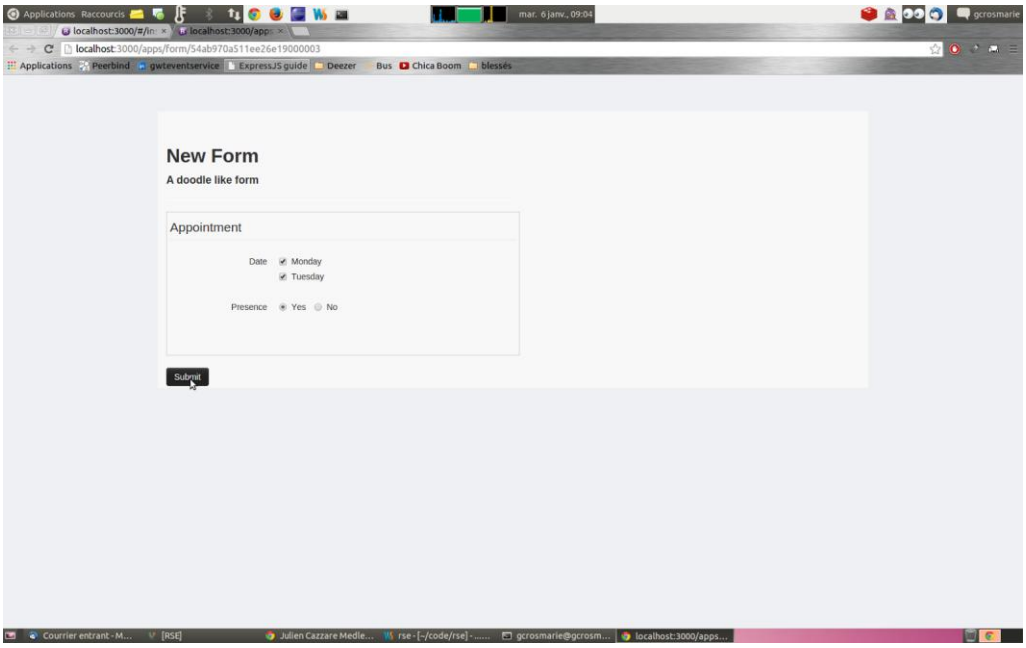


Figure 23 Form instance within FormaaS

4.3.3. Simulation activity validation

During the simulation process, the input generated by the learner (Civil Servant) can be validated on line by means of data stored on TestData Repository component. The latter (detailed ahead in section 4.4.3) is in charge to store and maintain the data related to the execution of a learning session previously validated by a learning expert. Figure 24 shows the sequence diagram related to this process. For the sake of simplicity the details of internal and management calls between components have been omitted so to focus only to the most important components interactions.

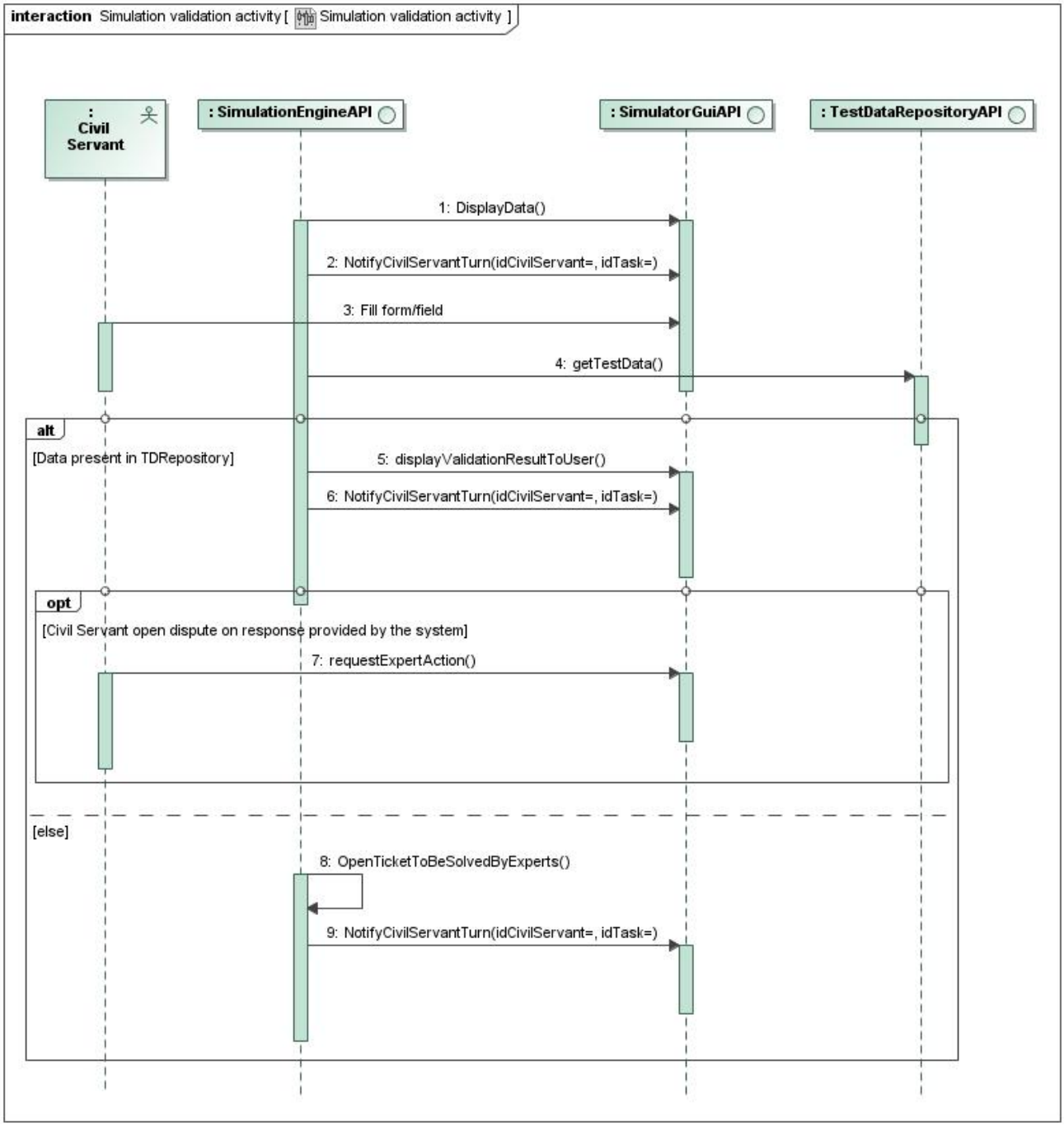


Figure 24 Simulation Validation diagram

As reported in Figure 24 the validation process is divided into three main sub-cases:

1. generic case: it occurs when the simulation engine has 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.
2. 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 with 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.
3. 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.

In both cases 2 and 3, 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.

4.3.4. Analytics

All activities carried out by the simulation engine are stored into the persistence layer. Analytics features will be provide facilities to display logs.

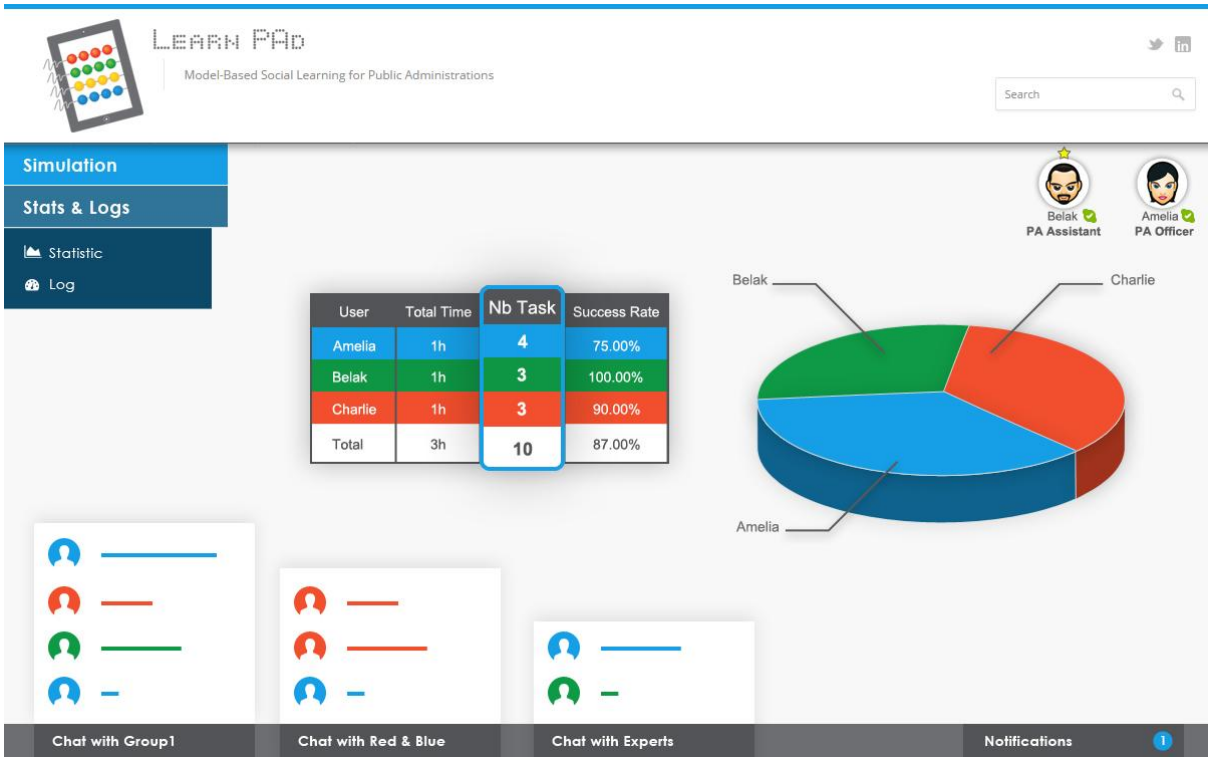


Figure 25 Learn PAd Simulation - Statistics and Logs

In Figure 25 a possible visualization of the statistics the civil servant is able to get after a simulation is provided. Statistics may concern various indicators, both regarding the overall performances of learners for this simulation, or about how the civil servants performed in a specific simulation instance. For example, indicators can be: the number of times the simulation has been run, average duration of activities, processes, and so on. Some information can be selected to obtain a more detailed view with additional statistics or indicator charts.

Inside the Learn PAd project the use of Kibana[4], an open-source component will be considered. Kibana is a highly scalable user interface for ElasticSearch that allows to efficiently search, graph, analyze and otherwise make sense of data.

[4] http://www.elasticsearch.org/overview/kibana/

4.4. Persistence Layer

The Persistency Layer component is in charge to store simulation logs and business process states. Its main sub-components are depicted in the Figure 26 and are described in the following sections.

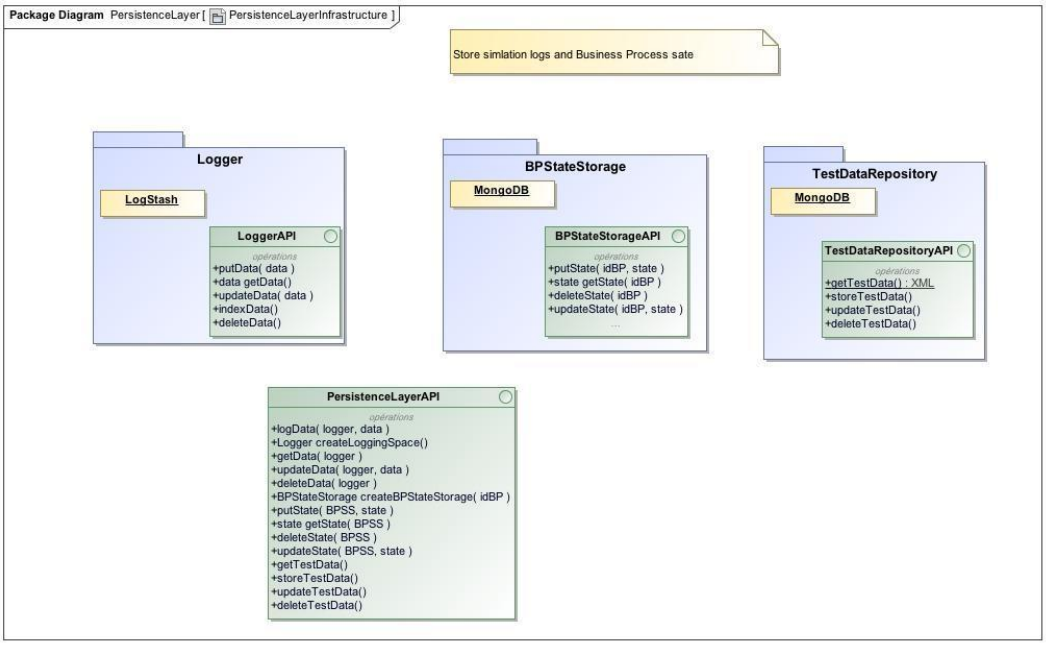


Figure 26 Persistency Layer Infrastructure

4.4.1. Logger

The Logger 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. Events are stored into a database that ensures persistency. In the Learn PAd project the use of LogStash [5]open-source component will be considered so to collect logs, parse them, and store them for later use whereas ElasticSearch will be used as a backend datastore. The latter provides a distributed, multitenant-capable full-text search engine with a RESTful web interface and schema-free JSON documents.

4.4.2. BP StateStorage

The Business Process State Storage 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. In the Learn PAd project the use of open-source database MongoDB[6] will be considered.

5 http://logstash.net/

6 http://www.mongodb.org/

4.4.3. TestDataRepository

This component collects the historical data that relate to the simulations executions. Historical data are inserted and validated by experts either during the “training” cases or when the learner (Civil servant) asks help to the expert.

Historical data are relative to validated paths (or subpaths) of the business processes and specify input and output information/values corresponding to each activity. The historical data can be used for:

1. verifying the correctness of the information provided by a Civil servant during a simulation session;
2. inferring the robot behavior in case of individual or mixed simulation.

Historical data are updated by using the data stored in the persistence layer once validated by an expert as described above.

The technology used for the TestDataRepository will the same adopted for the BPStateStorage component i.e. MongoDB an open-source No-SQL document database.

4.5. Communication middleware

Communication middleware provides event based communication service among simulation components based on publish / subscribe pattern. In Figure 27 the main components of the Communication middleware are shown. The Persistence Layer has been encapsulated into the Communication middleware in order to make it uniformly available from any simulation’s component.

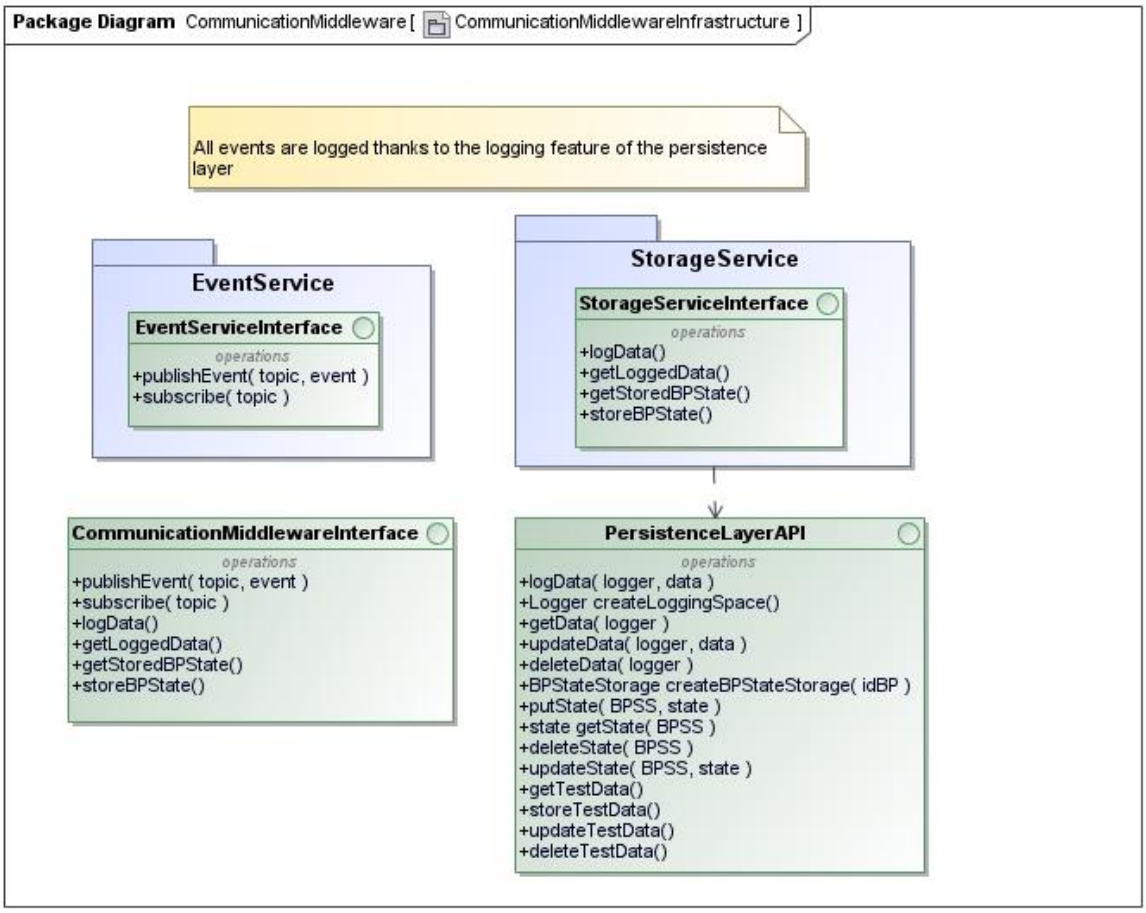


Figure 27 Communication middleware

4.6. Monitoring

The simulation environment is equipped with monitoring mechanisms that allow to provide feedbacks on the business process execution useful for the learning activities assessment and KPI evaluation. Figure 28 presents the components view of the Monitoring Infrastructure. For each component a package representation with the main operations signature is provided in Figure 29. Before detailing the monitoring components, the monitoring setup and the overall monitoring process are described. More detailed descriptions of the monitoring components are provided in the following sections.

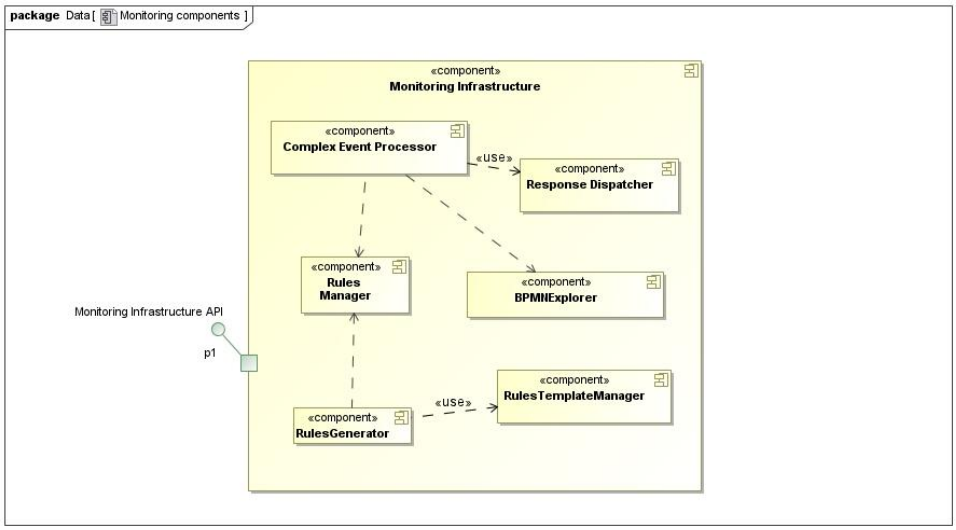


Figure 28 Main components of the Monitoring Infrastructure

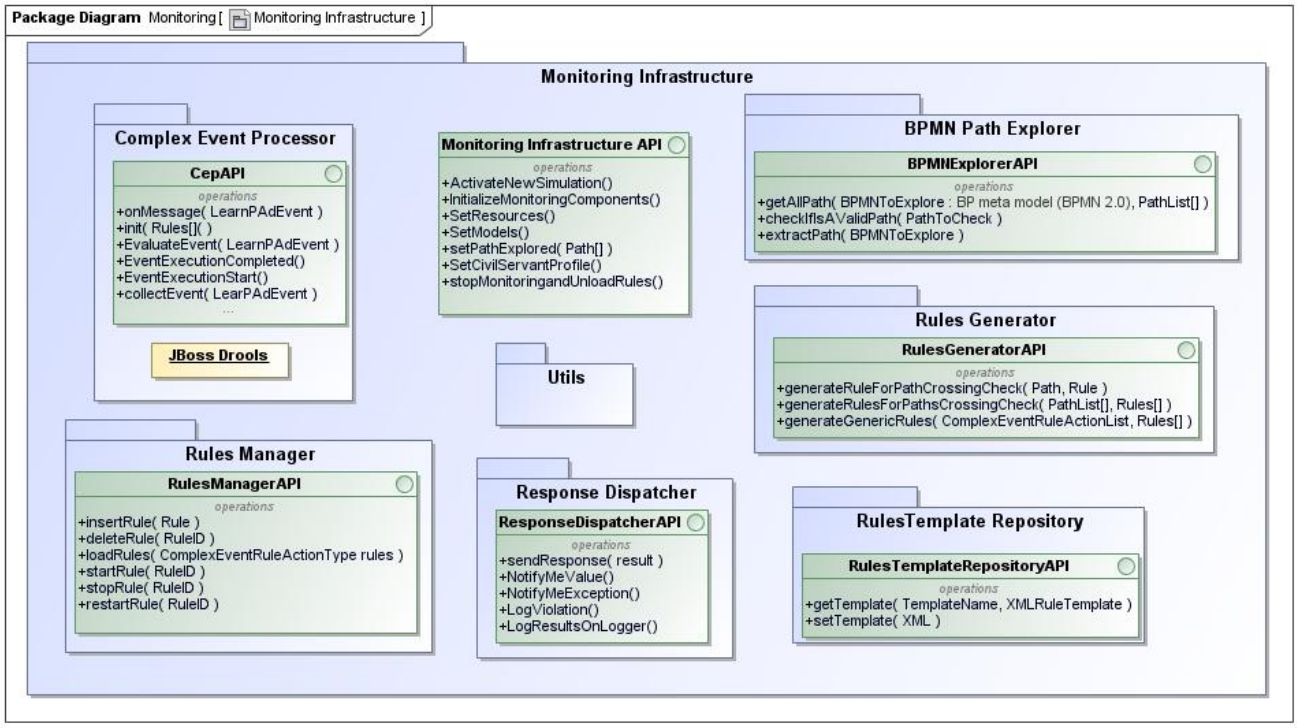


Figure 29 Package description of Monitoring Infrastructure

4.6.1. Overall functional description

The Simulation environment uses facilities provided by monitoring to check if execution patterns will be respected during the execution of a business process. In order to do that, the simulation engine sends a monitoring request through a message to the Monitoring Infrastructure specifying the set of events, failures detections and time constraints useful for evaluating the learner competency and simulation time completion.

Figure 31 shows the initialization and setup of the monitoring infrastructure. Specifically, once received an ActivateSimulation message from the SimulationEngine, the Monitor infrastructure, through its API, starts to gather all the information (resources, models, KPI parameters) needed for the execution by querying the Learn PAd Platform through the CoreFacade API and receiving the response through the Bridge API. After loading the resources the Monitor Infrastructure calls the BPMNPathExplorer component in order to start the generation of all the feasible paths (detailed in Section 4.6.5). At the same time, a request (GenerateGenericRules) for generating rules is sent to the RulesGenerator, providing it with a set of data related to parameters to be monitored during the execution.

Specific requests (GenerateRulesForPathsCrossingCheck) for generating the rules able to check if and when the CivilServant will execute specific paths of the Business Process are also sent to the RulesGenerator. The monitoring rules are generated according to a rule template that is stored in the RulesTemplateRepository. After initializing the CEP (Complex Event Processor) for the monitoring evaluation and loading the rules on the RuleManager, a message for notifying that the monitoring is ready to listen the events is sent to the SimulationEngine.

The events generated during the simulation and produced by the Simulation Engine, are sent to the Monitoring Interface and gathered by the CEP (Complex Event Processor). Those events, called Learn PAd Events, are structured objects containing all the information useful for monitoring the simulation. In Figure 30 the structure of a Learn PAd Generic Event is presented.

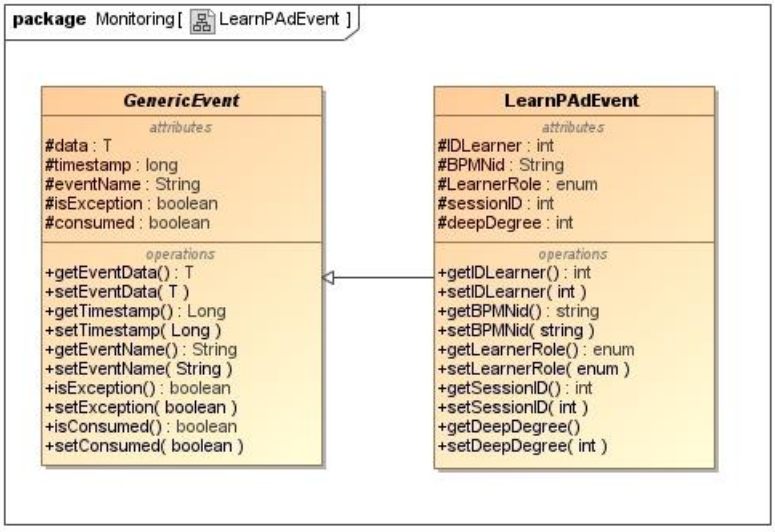


Figure 30 Learn PAd Generic Event

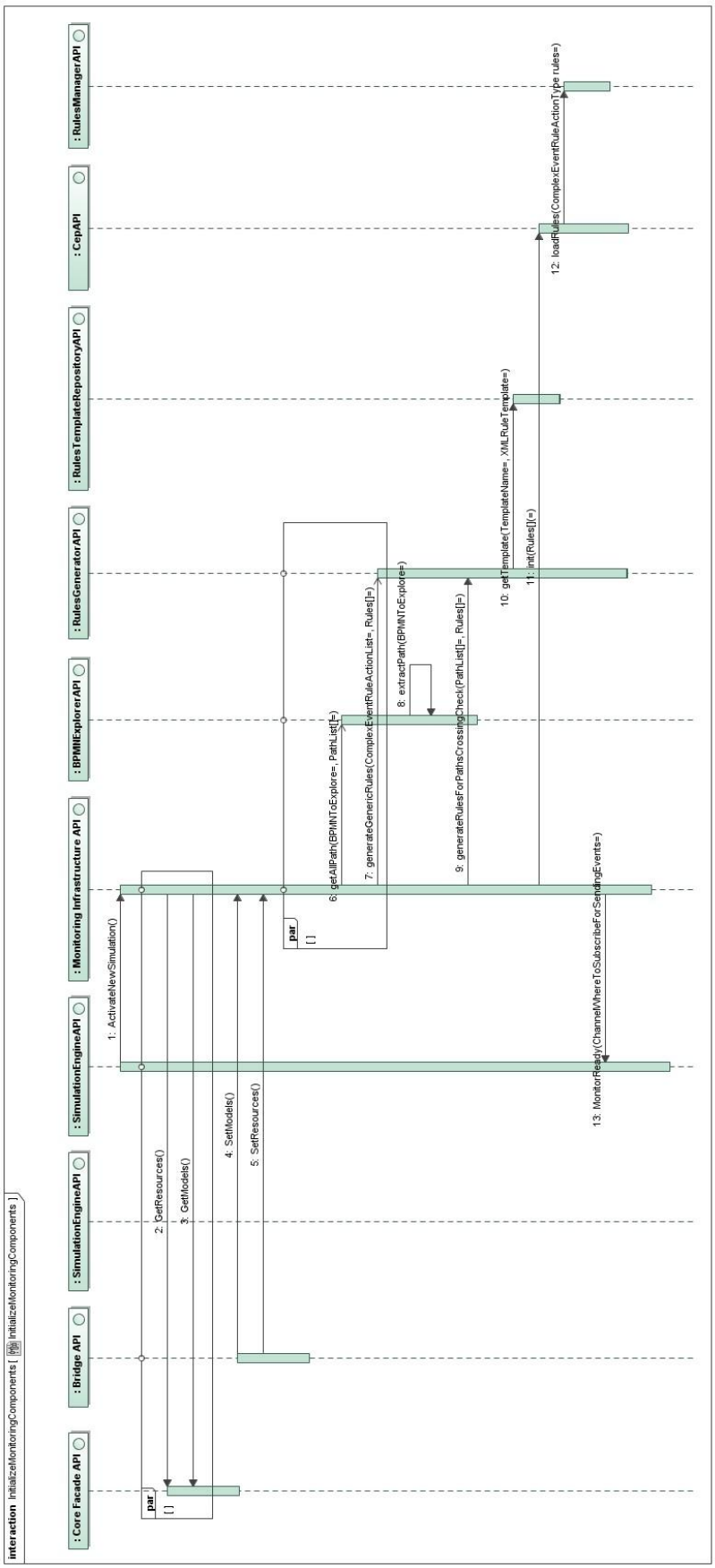


Figure 31 Monitoring Initialization

4.6.2. Monitoring Infrastructure API

The MI API is the public interface of the Monitoring Infrastructure that is in charge to orchestrate and manage all the communications between Simulation and Monitoring and contains also the methods to interact with the Bridge of the Simulation Environment (Core Facade API and Bridge API) in order to gather and store all the information needed and provided by the Learn PAd core platform. More in detail, this API manages a set of components in charge to fetch requests coming from the simulation engine, initializes all the monitoring components involved in a simulation getting and setting the resources and the models (BPMN, KPI parameters specification etc.) needed for the simulation.

This interface is the entry point for enacting the Model Based approach proposed in Chapter 5, initializing the BPMN Path expansion and Rule generation.

4.6.3. Complex Events Processor

The Complex Event Processor (CEP) is the rule engine which detects the events, generated by a probe instantiated into the Simulation Environment and correlates them to infer more complex events.

Several complex events rule engines can be used as Drools Fusion, VisiRule, RuleML .

Our instance is realized using Drools Fusion, that is able to detect patterns and take countermeasures in real-time, monitor goals expressed through the KPI parameters and monitor the BPMN path coverage.

Considering a generic rule, a sequence diagram has been generated (see Figure 32) to clarify the basic interaction among components.

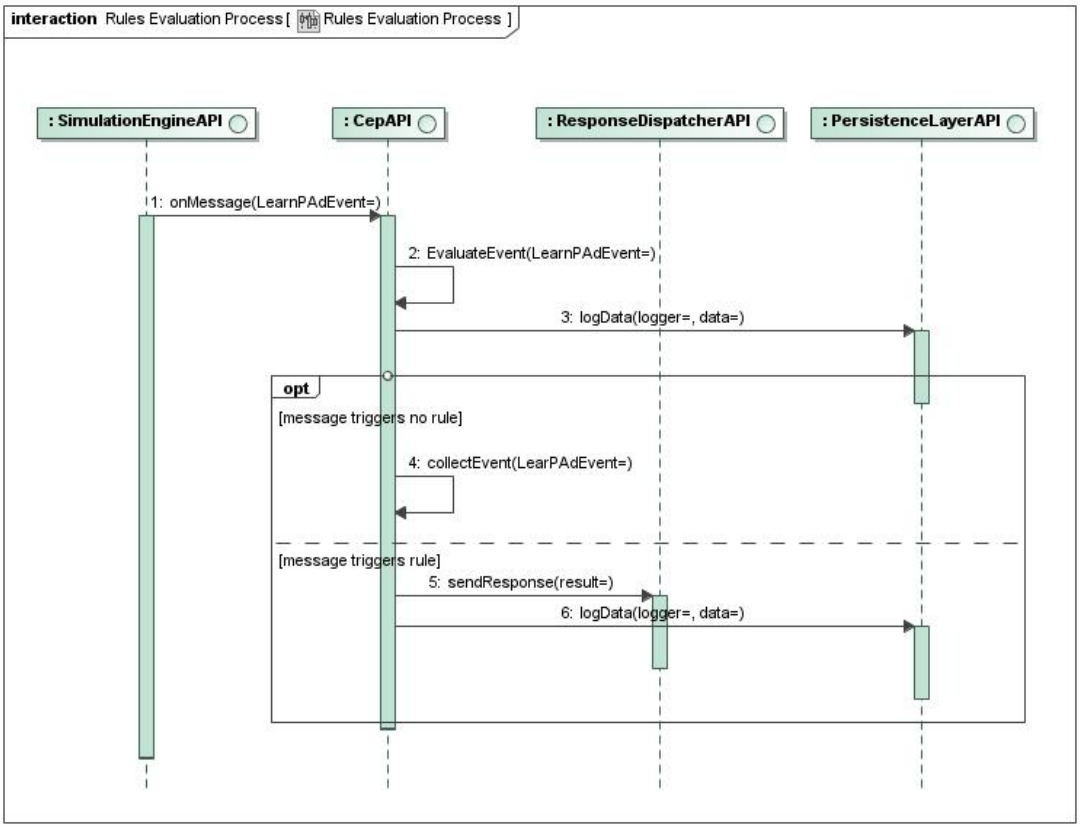


Figure 32 Rule evaluation process

The CEP is listening for incoming messages from the Simulation Engine. Once received a message through the dedicated session channel, the CEP evaluates the Learn PAd event and logs the analysis into the Persistency Layer.

If the event triggers no rule, the event is just collected into the Event Stream of the CEP, moreover, if the event triggers a rule, the CEP sends a notification request to the Response Dispatcher in order to notify the completion/occurrence of a *ruleset* to whom requested the notification.

4.6.4. Response Dispatcher

The Response Dispatcher is a registry that keeps track of the RFM, Request For Monitoring, sent to the infrastructure. Once it receives the advice of a rule firing, pattern completion, or property violation from the CEP, it looks on the registry for the channel on which the consumer/requester of the evaluation is listening for response and sends to it the response.

4.6.5. BPMN Path Explorer

This component is in charge to explore and save all the possible paths reachable by a Civil Servant on a BPMN.

The paths extraction is realized by an optimized unfolding algorithm that exploits advantages provided by the use of BPMN 2.0. The goal is to derive an acyclic graph, defining a partial order on its nodes. In particular, the exploration reduces the required space and time thanks to a more efficient management of the interleaving among different activities, taking into account the characteristics of a BPMN 2.0 model and of pools, parallel and exclusive gateways, and tasks sending and receiving messages within the model. More details about the BPMN exploration approach are in [5].

Once extracted the paths will be provided to the Rules Manager that through the Rules Generator will create, using the templates of rules stored into the Template Manager, a set of rules that aims to check the coverage of all the feasible paths of the loaded BPMN.

The paths extraction is useful to support the learning session adequacy presented in Chapter 5.

4.6.6. Rules Generator

The RulesGenerator is the component in charge to generate the rules starting from the information provided by the caller using the templates stored into the RulesTemplateManager.

These rules are generated according to the specific properties to be monitored. A generic rule consists of two main parts: in the first part the events to be matched and the constraints to be verified are specified; the second part includes the events/actions to be notified after the rule evaluation.

The Listing below shows a fragment of a generic Drools rule that could be used to monitor the completion time of a learning task. Assuming that during the business process simulation the civil servant executes the “checkApplication” task, the events of start and stop of this task are notified by the simulation engine to the monitoring infrastructure. The monitoring rule first detects the events “checkApplicationStart” and “checkApplicationStop” that represent the start and stop events associated to the “checkApplication” task, and then

checks that the event “checkApplicationStop” happens after 60s with respect to the event “checkApplicationStart”. If the latency between “checkApplicationStart” and “checkApplicationStop” is higher than 60s a time constraint violation is raised and notified to the PersistencyLayer component.

```
rule "TASK_COMPLETION_$TASKNAME$"
no-loop
salience 10
dialect "java"
when
	$checkApplicationStartEvent : Learn PAd Event(this.isConsumed == true,
		this.getTimeStamp == $_TIMESTAMP_$,
		this.getEventName == "$TASKSTART_PLACEHOLDER$");
	$checkApplicationStopEvent : Learn PAd Event(this.isConsumed == false,
		this.getEventName == "$TASKEND_PLACEHOLDER$",
		this after[0,60s] $aEvent);
then
	$checkApplicationStopEvent.setConsumed(true);
	update($checkApplicationStopEvent);
	ResponseDispatcher.NotifyMeRule("Time constraint violation occurred on $TASKNAME$");
	retract($checkApplicationStartEvent);
	retract($checkApplicationStopEvent);
end
```

4.6.7. Rules Template Manager

This component is an archive of predetermined rules templates that will be instantiated by the RulesGenerator when needed. A rule template is a rule skeleton, the specification of which has to be completed by instantiating a set of template-dependent placeholders. The Rule Generator will instantiate the template with appropriate values inferred from the specific properties to be monitored. Second, once the synthesis of the new set of rules is completed, the new rules are loaded by the Rule Generator into the Rules Template Manager. For the sake of completeness, the Rules Template Manager can also include sets of static rules that do not depend on the generative process discussed above.

4.6.8. Rules Manager

The complex event detection process depends directly from the operation done by the Rules manager component. This is in charge of managing all the knowledge bases loaded into the complex event processor, to load and unload set of rules and to fire them when needed.

5. Learning session adequacy

In this section exploiting the BPM models specification, more details are provided about the novel notion of learning session adequacy introduced in this deliverable.

The intuitive motivation becomes that if some part of the BPM has never been executed by one or more civil servants, they could have missed important information about the business process and therefore their acquired knowledge not completed.

This concepts is directly connected with the monitoring data collected during a learning session: If during the monitoring activity of the business process the monitor does not report some event in this context means: either that during a learning session civil servant didn't make the event happening or that robots instance should be improved in order to allow the complete execution of a BP.

Thus the proposal of this section goes in the direction of making the monitor wiser in the sense of Socrates's saying, i.e., to enable a monitor "to know that it does not know". To do this it is necessary first to explicitly define in the target of the observation and then set coverage adequacy criterion on them .

In the rest of this section the concept of learning session adequacy is introduced and a set of coverage adequacy criterion are presented.

5.1. Basic definitions

In this section the generic concept of a learning adequacy criterion, without binding its definition to a specific coverage measure, is introduced. In fact, the notion of learning adequacy is neutral with respect to both the entities to be covered (i.e. activity, tasks, paths and so on) and the application domain.

Usually in coverage criteria set of entities that must be covered is defined for example all statements or all branches in the program control flow. If during the system execution all entities are covered, the coverage criterion is said to be satisfied; otherwise, the percentage of covered entities is used as a quality measure. Correspondingly, to assess the adequacy of a simulation execution it is important to identifying what are the relevant entities to be covered and monitoring if all of them, or otherwise what percentage, have been observed.

However even if intuitively very close to the classical definition of adequacy the learning adequacy implies some significant differences. One first difference concerns the period along which the coverage is measured. Usually the adequacy is measured over an execution of a testing session and the obtained measure refers to the overall execution. In monitoring, however there is not a similar notion of a test session, since monitoring observes how the system spontaneously behaves according with the civil servants decisions. In principle, the monitor observations are continuously collected over a simulation, whereas for analysis purposes it is necessary to set some limits upon the observed execution traces.

Therefore, a notion similar to that of a test session is defined, by associating the monitoring coverage measure with the length of a considered observation period. Intuitively, a sliding observation window over a time measurement unit can be established, which could be either continuous (e.g. the execution traces collected in the last 120 sec) or discrete (e.g., the most recent 15 traces).

Another difference concerns the fact that monitoring coverage is no longer a monotonically increasing function as it happens in classical definition: as more traces are monitored, one entity that was covered before could not be covered anymore. Thus monitoring coverage in

simulation session, within the different observation windows, could increase or decrease, allowing for detecting possible dynamic modifications/behaviour of the learners interactions. Thus the monitoring coverage can be defined as:

Definition 1: Denote $r_i \in R$ the *i*-th entity to be covered, and by $\delta_i \in \Delta$ the length of its associated observation window. The learning adequacy criterion C dynamically measures the coverage on R for a given entity *i* at each time unit t as follows:

$$C[R, \Delta](t) = \frac{\sum_{i=1}^{|R|} \lambda_i(t)}{|R|}$$

where for $r_i \in R$ and $\delta_i \in \Delta$

$$\lambda_i(t) = \begin{cases} 1 & if\ 0\ r_i\ is\ covered\ at\ least\ once\ in\ [t-\delta_i, t] \\ 0 & otherwise \end{cases}$$

According to this definition the length of the δ_i could be different for each r_i, or could be the same for all entities. In summary the definition of learning adequacy introduces the following concepts:

- an “adequate monitor" is a monitor that is instructed to keep track of a set of entities r_i to be covered in a window δ_i (this is similar to the instrumentation phase of coverage testing);
- a monitor coverage analyser is a tool that, at every instant *t*, can provide a coverage measure as in Def.1 and, if this is less than 1, can provide a list of those entities that have not been covered;
- an entity that is not covered is an indication that the learner has not simulated some entities of the business process and for a period and therefore his/her generic knowledge could be updated. In such a case warnings could be raised by the learning coverage analysis.

5.1.1. Entity definition

Inside a simulation session the definition of what is an entity to be covered can be provided at different levels and with different targets. Within Learn PAd the following definitions can be considered[7]:

Activity:

- activity entity: given a BP, an activity entity is one of the activities specified in the BP that can be executed at least once.
- activity coverage domain: considering a BP the activity coverage domain is the set of all the activity entities of the BP;
- percentage of activity coverage: with reference to Def.1 the percentage of activity coverage at time t is given by 100*C, where R is the activity coverage domain.

Consequently, at a given instant a simulation is adequate with respect to the activity coverage criterion if the percentage of activity covered is 100% (or greater than an established threshold level).

[7] Note that in the current definition the distinction between optional entity and mandatory one is not considered.

Sequence flow:

- sequence flow entity: given a BP a sequence flow entity is one of the sequence flow[8] specified in the BP that can be executed at least once.
- sequence flow domain: considering a BP the sequence flow coverage domain is the set of all the sequence flows entities of the BP;
- sequence flow coverage: with reference to Def.1 the percentage of sequence flow coverage at time *t* is given by 100*C, where R is the sequence flow coverage domain.

Consequently, at a given instant a simulation is adequate with respect to the sequence flow coverage criterion if the percentage of sequence flow entities covered is 100% (or greater than an established threshold level).

Path:

- Path entity: given a BP a path entity is one of the paths specified in the BP that can be executed at least once.
- Path domain: considering a BP the path coverage domain is the set of all the path entities of the BP;
- Path coverage: with reference to Def. 1 the percentage of path coverage at time t is given by 100*C, where R is the path coverage domain.

Consequently, at a given instant a simulation is adequate with respect to the path coverage criterion if the percentage of path entities covered is 100% (or greater than an established threshold level).

Learn PAd example

In this section in order to clarify the concept of learning session adequacy introduced in this section some examples, taken from the deliverable D5.1 are presented. For aim of completeness few details about the BP considered are provided, remanding to D5.1 for the complete description. The BP considered is the Mock-up of real process “Student Admission” of D5.1 section 3.1.1 that refers to the process for regulating the admission of applicants to the study program MSc in Business Information Systems (BIS) within the school of business at FHNW. Figure 33 shows the overview of the process considered.

[8] Sequence Flow is used to show the order in which activities of a process will be performed. A Sequence Flow connection is represented with a solid line and a solid arrowhead in a Business Process Model.

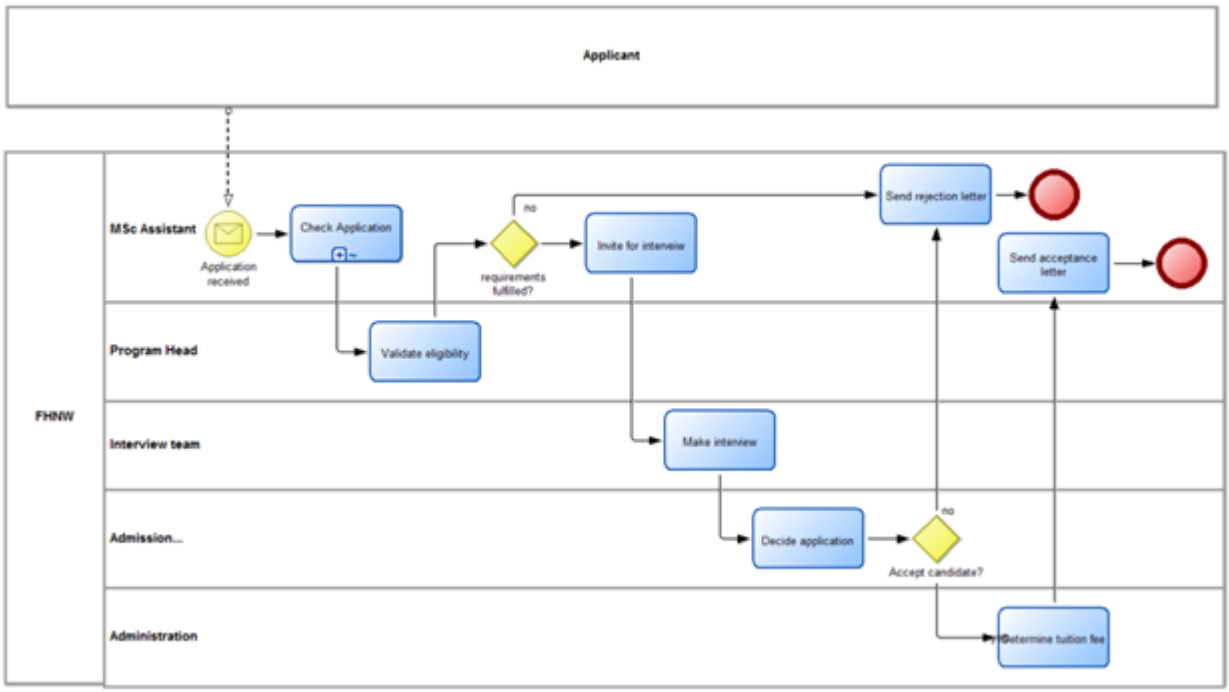


Figure 33 Overview of the student admission process (from D5.1)

As example if inside a simulation session of this BP the entity has been defined at the granularity of an activity, the definitions provided in the previous section can be instantiated as in the following:

- Activity entity: it is one of the activities of the BP of Figure 33 like: Invite for interview, Make Interview, Send acceptance letter and so on.
- activity coverage domain: this is the set of all the activities entities of the BP of Figure 33. In this case the activity coverage contains 13 activity entities.

Supposing that the observation window for the assessment of the learning adequacy has been fixed to a simulation execution and that the set of monitored (executed) activity entities that have been observed in the observation windows are the ones marked with a (blue) arrow in the following figure:

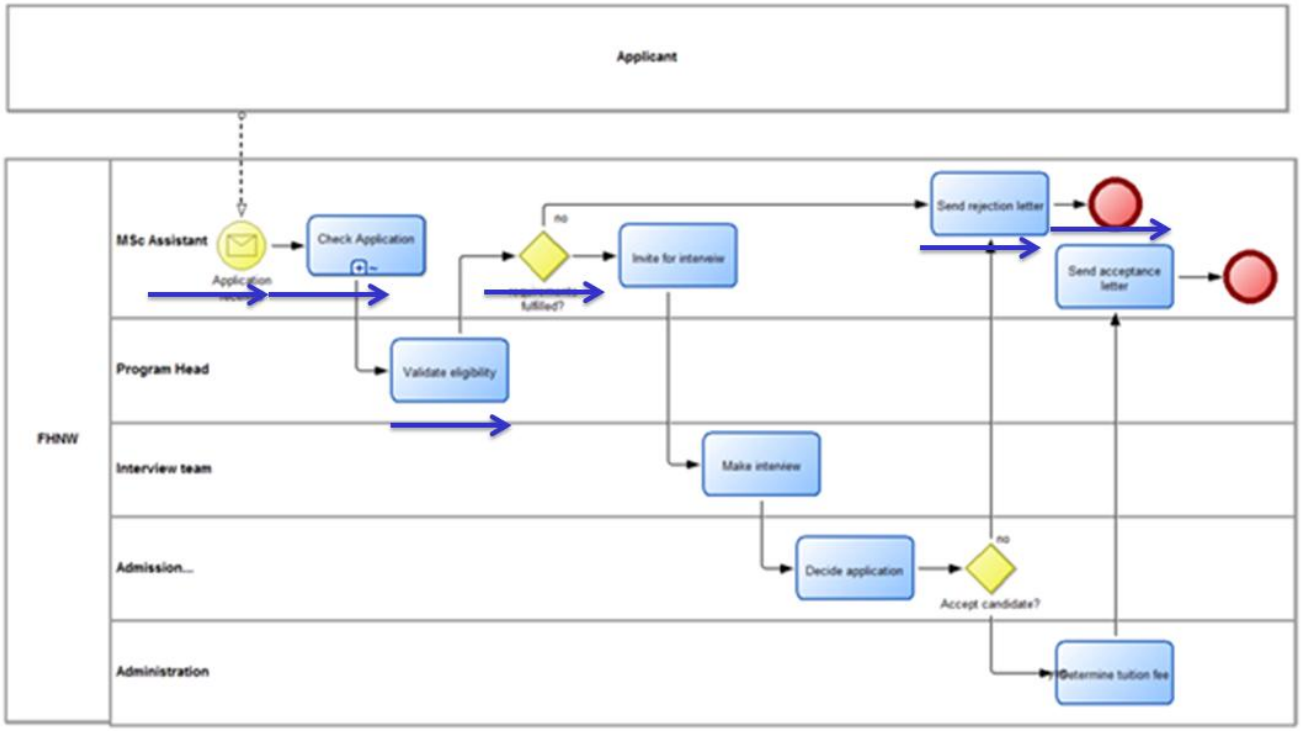


Figure 34 Entities monitored over a observation windows

In this case the percentage of activity coverage within the established observation windows is 6/13 *100 = 46,15 % where:

- 6 is the number of activity entities monitored in the observation window (the activity entities marked with an arrow)
- 13 is the cardinality of activity coverage domain.

If the learning adequacy level established were 100% of the activity entities the current simulation would not be adequate with respect to the activity coverage criterion.

Considering instead the case in which inside a simulation session of BP of Figure 33 the entity has been defined at the granularity of a path, the definitions provided in the previous section can be instantiated as in the following:

- path entity: it is one of the paths of the BP of Figure 33 and evidenced with arrows in different colour in Figure 34.

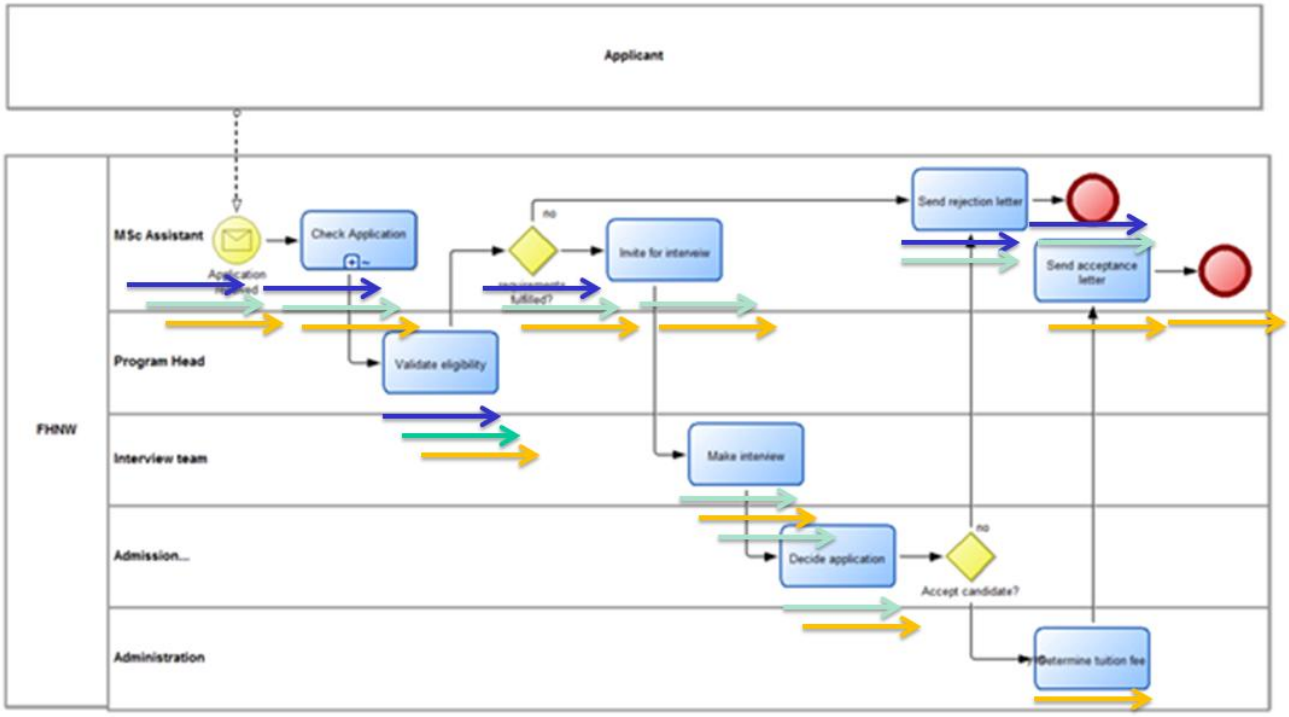


Figure 35 Path entities

Consequently the path coverage domain, which contains the set of all the paths entities evidenced in Figure 35, has in this case cardinality 3.

Supposing that the observation window for the assessment of the learning adequacy has been fixed to a 10 minutes of simulation execution and that the set of monitored (executed) path entities that have been observed in the observation windows is represented just by the one marked with blue arrows in Figure 34, the percentage of path coverage within the established observation windows is 1/3 *100 = 33 % where:

- 1 is the number of path entities monitored in the observation window, i.e 10 minutes of simulation execution
- 3 is the cardinality of path coverage domain.

If the learning adequacy level established in the observation windows were 33% of the path entities the current simulation is adequate with respect to the activity coverage criterion.

5.1.2. Relative coverage

Inside the Learn PAd a simulation session can involve different civil servants executing a different role in the BP. In this case it could be the necessity to establish the learning adequacy of each single participant and not only the overall learning adequacy. Thus whatever is the definition of entity, the monitoring collected data have to be analyzed for each single civil servant. Thus for each role a relative coverage representing the adequacy-by-role can be computed so to evaluate the specific degree of learning associated to each single learner.

At same time, independently by the civil servants and the entity considered, a BP could be composed by different sub-BPs each one associated to a different activity. In this case it could be necessary to evaluate the learning activity at a level of sub-BP of a certain level of explosion of the BP. Also in this case a relative adequacy-by-level coverage can be

considered. In the following more specific definitions for the relative coverage measures are provided.

Adequacy evaluated per role:

Given a BP where different role $M_1..M_k$ are involved. For each role L_i where $i \in [1..k]$ the following definitions can be considered:

Activity-by-role:

- activity-by-role entity(M_i): given a BP and a specific role Mi involved in the BP, an activity-by-role entity(M_i) is one of the activities specified in the BP that can be executed by the role Mi at least once.
- activity-by-role coverage domain (M_i): considering a BP and a specific role M_i involved in the BP the activity-by-role coverage domain is the set of all the activities-by-role entities that can be executed by the role M_i in the BP;
- percentage of activity-by-role coverage (M_i): with reference to Def.1 the percentage of activity-by-role coverage at time *t* is given by 100*C, where *R* is the activity-by-role coverage domain.

Consequently, at a given instant a simulation is adequate with respect to the activity-by-role coverage(M_i) criterion if the percentage of activity-by-role for the Role M_i covered is 100% (or greater than an established threshold level).

Sequence flow-by-role:

- Sequence flow-by-role entity(M_i): given a BP and a specific role M_i involved in the BP, a sequence flow-by-role entity(M_i) is one of the sequence flow specified in the BP that can be executed by the role M_i at least once.
- sequence flow-by-role coverage domain (M_i): considering a BP and a specific role M_i involved in the BP the sequence flow-by-role coverage domain is the set of all the sequence flow-by-role entities that can be executed by the role M_i in the BP;
- percentage of activity-by-role coverage (M_i): with reference to Def. 1 the percentage of sequence flow-by-role coverage at time *t* is given by 100*C, where *R* is the entity coverage domain.

Consequently, at a given instant a simulation is adequate with respect to the sequence flow-by-role coverage(M_i) criterion if the percentage of sequence flow-by-role for the Role M_i covered is 100% (or greater than an established threshold level).

Path-by-role:

- Path-by-role entity (M_i): given a BP and a specific role M_i involved in the BP, path-by-role entity(M_i) is one of the path specified in the BP that can be executed by the role M_i at least once.
- path-by-role coverage domain (M_i): considering a BP and a specific role M_i involved in the BP the path-by-role coverage domain is the set of all the path-by-role entities that can be executed by the role M_i in the BP;
- percentage of path-by-role coverage (M_i): with reference to Def.1 the percentage of path-by-role coverage at time *t* is given by 100*C, where *R* is the entity coverage domain.

Consequently, at a given instant a simulation is adequate with respect to the path-by-role coverage(M_i) criterion if the percentage of path-by-role for the Role M_i covered is 100% (or greater than an established threshold level).

Adequacy evaluated per BP level:

Given a BP composed by several sub-BPs where m indicates the level of depth of a sub-BP in a BP. For each level L_i where $i \in [1..m]$ the following definitions can be considered:

Activity-by-level:

- Activity-by-level entity(L_i): given a BP and a specific role L_i involved in the BP, an activity-by-level entity(L_i) is one of the activities specified in the BP that can be executed by the role L_i at least once.
- activity-by-level coverage domain (L_i): considering a BP and a specific role L_i involved in the BP the activity-by-level coverage domain is the set of all the activities-by-level entities that can be executed by the role L_i in the BP;
- percentage of activity-by-level coverage (L_i): with reference to Def.1 the percentage of activity-by-level coverage at time *t* is given by 100*C, where *R* is the activity-by-level coverage domain.

Consequently, at a given instant a simulation is adequate with respect to the activity-by-level coverage(L_i) criterion if the percentage of activity-by-level for the Role L_i covered is 100% (or greater than an established threshold level).

Sequence flow-by-level:

- Sequence flow-by-level (L_i): given a BP and a specific role L_i involved in the BP, a sequence flow-by-level entity(L_i) is one of the sequence flows specified in the BP that can be executed by the role L_i at least once.
- sequence flow-by-level coverage domain (L_i): considering a BP and a specific role L_i involved in the BP the sequence flow-by-level coverage domain is the set of all the sequence flow-by-level entities that can be executed by the role L_i in the BP;
- percentage of activity-by-level coverage (L_i): with reference to Def.1 the percentage of sequence flow-by-level coverage at time *t* is given by 100*C, where *R* is the entity coverage domain.

Consequently, at a given instant a simulation is adequate with respect to the sequence flow-by-level coverage(L_i) criterion if the percentage of sequence flow-by-level for the Role L_i covered is 100% (or greater than an established threshold level).

Path-by-level:

- Path-by-level (L_i): given a BP and a specific role L_i involved in the BP, path-by-level entity(L_i) is one of the paths specified in the BP that can be executed by the role Li at least once.
- path-by-level coverage domain (L_i): considering a BP and a specific role L_i involved in the BP the path-by-level coverage domain is the set of all the path-by-level entities that can be executed by the role L_i in the BP;
- percentage of path-by-level coverage (L_i): with reference to Def.1 the percentage of path-by-level coverage at time *t* is given by 100*C, where *R* is the entity coverage domain.

Consequently, at a given instant a simulation is adequate with respect to the path-by-level coverage(L_i) criterion if the percentage of path-by-level for the Role L_i covered is 100% (or greater than an established threshold level).

Learn PAd example

In this section in order to clarify the concepts of relative learning adequacy introduced in this section some running examples on the bases of the BP of Figure 33 are presented.

As evidenced in Figure 33 the BP involves 5 different roles (MSc assistant, Program Head, Interview team, Admission and Administrative). In this case it could be the necessity to establish the learning adequacy of each single participant and not only the overall learning adequacy. Thus supposing to select the Interview Team role and the activity as the granularity for entity, the definitions provided above for the activity-by-role can be instantiated as:

- Activity-by-role entity (Interview Team): it is the unique activity associated to the Interview Team role in the BP of Figure 33, i. e. Make interview.
- activity-by-role coverage domain (Interview Team): the set includes only the activities-by-role entities that can be executed by the role Interview Team in the BP of Figure 33. In this case the cardinality is 1.

Supposing that the observation window for the assessment of the learning adequacy has been fixed to a simulation execution and that the set of monitored (executed) activity entities that have been observed in the observation windows are the ones marked with a (blue) arrow in the Figure 34.

In this case the percentage of activity-by-role coverage for the role Interview Team within the established observation windows is 0/1 *100 = 0 % where:

- 0 is the number of activity-by-role entities monitored in the observation window for the role Interview Team
- 1 is the cardinality of activity-by-role coverage domain for the role Interview Team.

Thus the learning adequacy level is not adequate with respect to the activity-by-role coverage criterion.

Considering instead to select the MSc Assistant and the path as the granularity for entity, the definitions provided above for the path-by-role can be instantiated as:

- Path-by-role entity (MSc Assistant): is one of the (two) path specified in the BP of Figure 33 that can be executed by the role MSc Assistant. For aim of completeness the two paths are highlighted with arrows in different colour in the following figure.

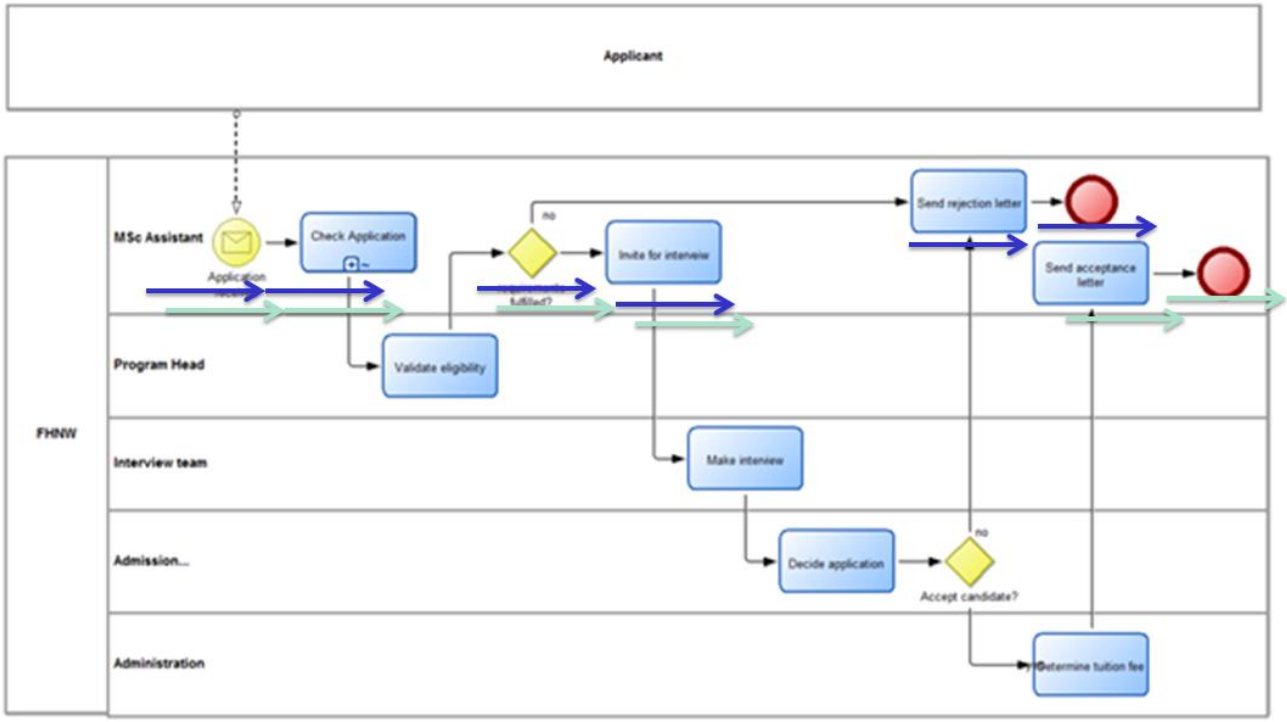


Figure 36 Path-by-role entity for MSc Assistant role

Consequently the path-by-role coverage domain for the MSc Assistant role has cardinality 2.

Supposing that the observation window for the assessment of the learning adequacy has been fixed to a 10 minutes of simulation execution and that the set of monitored (executed) path-by-role entities that have been observed in the observation windows is represented just by the one marked with blue arrows in the Figure 34, the percentage of path-by-role coverage within the established observation windows for the role MSc Assistant is 1/2 *100 = 50 % where:

- 1 is the number of path-by-role entities monitored in the observation window, i.e in 10 minutes of simulation execution
- 2 is the cardinality of path-by-role coverage domain.

If the learning adequacy level established in the observation windows were 50% of the path-by-role entities the current simulation is adequate with respect to the activity coverage criterion.

Independent of the civil servants and the entity considered, a BP could be composed by different sub-BPs each one associated to a different activity. For instance, referring to the BP of Figure 33 as reported in the deliverable D5.1, the activity Check application is further refined into the sub-process shown in Figure 37.

In this case it could be the necessary to evaluate the learning activity either at a first level (i.e. at level of Figure 33) or at second level (i.e. at level of Figure 37). Consequently a different relative adequacy-by-level coverage can be considered. In the following examples of instantiation of the definition for the examples of Figure 33 and Figure 37 are provided.

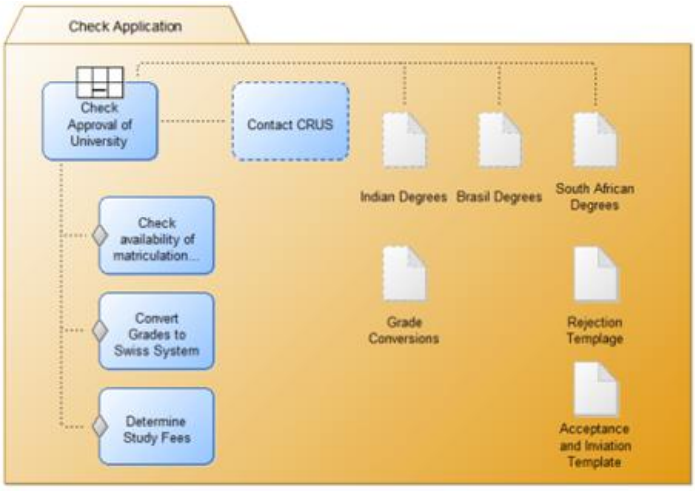


Figure 37 CMMN model of the knowledge-intensive sub-process "Check application" from D5.1

Thus supposing to select the MSc assistant as role, the activity as the granularity for entity depending on the level of detail the BP is considered the following definition can be instantiated:

At level 1:

- Activity-by-level entity(1): it is one of the activities the MSc Assistant can execute on the BP of Figure 33 like for instance Check Application, or Send acceptance letter.
- activity-by-level coverage domain (1): it is the set of activity-by-level entities that the role MSc Assistant can execute on the BP of Figure 33. In this case the cardinality of activity-by-level coverage domain is 8.

At level 2

- Activity-by-level entity(2): it is one of the activities the MSc Assistant can execute on the BPs of Figure 33 once the specific activity Check Application has been expanded into the sub-process of Figure 37. Therefore an activity-by-level entity can be either Send acceptance letter or Check approval for University or Determine Study Fees. Note that at this level the Check Application activity is not considered anymore because expanded into its sub-process.
- activity-by-level coverage domain (2): it is the set of activity-by-level entities that the role MSc Assistant can execute on the BP of Figure 33 expanded with the information of Figure 37. In this case the cardinality of activity-by-level coverage domain is 12 (7 activities considered in the level 1 and 5 for the level 2).

Supposing that the observation window for the assessment of the learning adequacy has been fixed to a simulation execution and that the set of monitored (executed) activity-by-level entities that have been observed in the observation windows are the ones marked with a (blue) arrow in the Figure 34, the following measure can be derived:

At level 1

- percentage of activity-by-level coverage (1) relative to MSc assistant role within the established observation windows is 6/8 *100 = 75 % where:
- 6 is the number of activity-by-level entities at level 1 monitored in the observation window for the MSc assistant role
- 8 is the cardinality of activity-by-level coverage domain at level 1 for MSc assistant role

Thus if the learning adequacy level established in the observation windows were 80% of the activity-by-level entities (1) the current simulation is not adequate with respect to the activity-by-level coverage criterion.

At level 2

- percentage of activity-by-level coverage (2) relative to MSc assistant role is within the established observation windows is 10/12 *100 = 83,33 % where:
- 10 is the number of activity-by-level entities at level 2 monitored in the observation window for the MSc assistant role. In this case all the activities of Figure 37 have been monitored during the simulation execution.
- 12 is the cardinality of activity-by-level coverage domain at level 2 for MSc assistant role

Thus if the learning adequacy level established in the observation windows were 80% of the activity-by-level entities (2) the current simulation is adequate with respect to the activity-by-level coverage criterion.

6. Conclusions and plans of future work

In this deliverable the architectural design of the simulation and monitoring environment is presented with a particular focus on the definition of the functionalities and interactions among its components.

The design of some parts of the architecture such as the Test Data Repository will be improved through new information that will be provided by scheduled deliverables.

This architectural design will be the input for the forthcoming prototypes implementation activities that will help us to refine the simulation and monitoring environment design and discover missing elements.

The inputs from WP5 about KPI notation will be taken into account for refining the monitoring activities.

References

[1] Philip H. Anderson and Leigh Lawton. Business simulations & cognitive learning: Developments, desires, and future directions. Simulation & Gaming, 40:193–216, 2009.

[2] L. Ardissono, R. Furnari, A. Goy, G. Petrone, and M. Segnan. Monitoring choreographed services. In Innovations and Advanced Techniques in Computer and Information Sciences and Engineering, pages 283-288. Springer Netherlands, 2007.

[3] Ruth C. Clark and Richard E. Mayer. e-Learning and the Science of Instruction: Proven Guidelines for Consumers and Designers of Multimedia Learning. Pfeiffer, 2011.

[4] N. Delgado, A. Gates, and S. Roach. A taxonomy and catalog of runtime software-fault monitoring tools. IEEE Transactions on Software Engineering,30(12):859-872, dec. 2004

[5] Damiano Falcioni, Andrea Polini, Alberto Polzonetti and Barbara Re. Direct verification of BPMN processes through an optimized unfolding technique, QSIC 2012 – 12th International Conference on Quality Software, in Xi'an, China, 27th – 29th Aug 2012, pp. 179 – 188

[6] M. H. Jansen-vullers and M. Netjes. Business process simulation a tool survey. In In Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN, 2006.

[7] Jeffrey Joyce, Greg Lomow, Konrad Slind, and Brian Unger. Monitoring distributed systems. ACM Trans. Comput. Syst., 5(2):121–150, 1987.

[8] D. Lee, A. Netravali, K. Sabnani, B. Sugla, and A. John. Passive testing and applications to network management. In Proceedings of the International Conference on Network Protocols, pages 113-122,1997

[9] Masoud Mansouri-Samani and Morris Sloman. Monitoring distributed systems. Network and distributed systems management, pages 303–347, 1994.

[10] B. Plattner and J. Nievergelt. Special feature: Monitoring program execution: A survey. Computer, 14(11):76-93, nov. 1981.

[11] S. Rapps and E. Weyuker. Selecting software test data using data flow information. IEEE Transactions on Software Engineering, SE-11(4):367-375, 1985.