FP7 ICT STREP Project

Deliverable D6.2

Learn PAd Simulation Environment: Refined Architecture and Prototype Implementation

http://www.learnpad.eu

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

Deliverable Number	:	D6.2
Title of Deliverable	:	Learn PAd Simulation Environment: Refined Architecture and Prototype Implementation
Nature of Deliverable	:	Report and Prototype
Dissemination level	:	Public
Licence	:	Creative Commons Attribution 3.0 License
Version	:	4.0
Contractual Delivery Date	:	October 31, 2015
Actual Delivery Date	:	October 30, 2015
Contributing WP	:	WP6
Editor(s)	:	Sarah Zribi
Author(s)	:	Sarah Zribi, Tom Jorquera, Jean-Pierre Lorré, Antonia Bertolino, Antonello Calabrò, Eda Marchetti, Francesca Lonetti, Nesat Efendioglu, Robert Woitsch and Alfonso Pierantonio
Reviewer(s)	:	Jean Simard

Abstract

D6.2 is the second deliverable of Learn PAd Work-Package 6, focusing on the Business Process Simulation environment. The document specifies the refined architecture of the Business Process Simulation Engine and the monitoring environment of the Learn PAd Platform on the one hand, and provides a first implementation of its initial prototype on the other.

In particular, the current document describes gamification techniques that complement the logical architecture of the simulator (presented in D6.1) in order to provide an entertaining and engaging session with a learner in a simulated work context. To this end, we identify the main gamification mechanisms of the simulation and monitoring environment, their functionalities and their interactions.

In the prototype part of this deliverable we briefly describe the delivered code. We start by detailing (1) how to work (get the sources, build and run the platform) with the Learn PAd Simulation and then (2) how to use it.

Keyword list

Simulation and Monitoring Environment Refined Architecture, Gamification Techniques, Scoring mechanisms, Certificates, Implementation, Prototype, Application Programming Interface.

Document History

Version	Changes	Author(s)
0.1	TOC	Sarah Zribi, Tom Jorquera, Jean-Pierre Lorré, Antonia Bertolino, Antonello Calabrò, Eda Marchetti, Francesca Lonetti, Nesat Efendioglu, Robert Woitsch and Alfonso Pierantonio.
1.0	First Draft	Sarah Zribi, Tom Jorquera, Jean-Pierre Lorré, Antonia Bertolino, Antonello Calabrò, Eda Marchetti, Francesca Lonetti, Nesat Efendioglu, Robert Woitsch and Alfonso Pierantonio.
2.0	Internal version	Sarah Zribi, Tom Jorquera, Jean-Pierre Lorré, Antonia Bertolino, Antonello Calabrò, Eda Marchetti, Francesca Lonetti, Nesat Efendioglu, Robert Woitsch and Alfonso Pierantonio.
3.0	Candidate release version	Sarah Zribi, Tom Jorquera, Jean-Pierre Lorré, Antonia Bertolino, Antonello Calabrò, Eda Marchetti, Francesca Lonetti, Nesat Efendioglu, Robert Woitsch and Alfonso Pierantonio.

Document Review

Release	Date	Ver.	Reviewers	Comments
ToC	01.09.2015	0.1		
Draft	10.10.2015			
Internal	12.10.2015		Jean Simard	
Candidate Final	30.10.2015			

Glossary, acronyms & abbreviations

Item	Description
Angular JS	Angular JavaScript
API	Application Programming Interface
BP	Business Process
BPM	Business Process Modeling
CEP	Complex Event Processor
GUI	Graphical User Interface
GWT	Google Web Toolkit
HTML	Hyper Text Markup Language
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
MVC	Model-View-Controller
SPA	Single Page Applications
URL	Uniform Resource Locator
WP	Work Package

Table of Contents

List of Figures

1. Introduction

The Learn PAd project aims at developing a social, collaborative and learning platform for civil servants. The platform supports both an informative learning approach based on enriched Business Process models and a procedural learning approach based on simulation and monitoring (learning by doing).

In this context, Work-Package 6 aims at providing a simulation environment where learners will engage in training activities interacting with the different participants (civil servants) of a given Business Process. The simulation environment integrates a Simulation Engine and a Monitoring Component. The main 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 Civil Servants. Besides, monitoring features provide feedbacks on the Business Process execution for the evaluation of learners.

In D6.1, Learn PAd Simulation Environment: Specification and Design, delivered on M12, we sketched the overall architecture of the Learn PAd Simulator environment and provided an initial definition of its main components, their functionalities, their behavior and their interactions. The latter were organized in *Robots Framework*, *Monitoring*, *Communication Middleware*, *User Facade*, *Simulation Engine*, *Persistence Layer*, *Simulator Graphical User Interface* and *Bridge*. Furthermore, in the same deliverable we provided initial technological choices about the composition of each subsystem.

D6.2 is the second deliverable of this WP and focuses on Task 6.3 entitled *Reference Implementation of the Infrastructure for Multi-Sessions Business Process Simulation and Monitoring*. In the next sections, the objectives of the deliverable are defined and the overall structure is highlighted.

1.1. Deliverable objectives

This current document provides a report and a prototype that cover:

- A **refinement** of the architecture of the Learn PAd simulation environment including the simulator. This refinement explores more game-like techniques for civil servants engagement.
- A **refinement** of the Monitoring component concerning: the elaboration of the feedbacks on the Business Process execution for the evaluation of learners and the interaction with the Core Platform.
- A first **implementation** and setting of an adequate simulation environment and technical documentations for both the Business Process Simulation Engine and the monitoring component.

1.2. Deliverable outline

This current document provides a report and a prototype and is structured as follows:

- Chapter 2 describes the methodology and the supporting tools we adopted to refine the initial Learn PAd Simulation Environment architecture presented in D6.1. In particular, we have enriched the simulator engine with the addition of two main gamification mechanisms including on-screen score and certificates (awards). Its functionalities, interactions and the set of Application Programming Interface (API) that come with it are also detailed.

- Chapter 3 describes the updated Monitoring components by means of a UML-like specification. The main improvements are related on the calculation of civil servant assessment score and the interaction with the Learn PAd Core Platform.
- In Chapter 4, we describe how we work with the Learn PAd Simulation platform. Precisely, we detail the download of the source code, the build of the platform and finally how to run it.
- Chapter 5 introduces a manual user giving assistance to use the Learn PAd Simulation environment and monitoring component.
- Finally, chapter 6 concludes the deliverable, with a summary of its contributions, and hints at future work directions.

2. Business Process Simulation Environment

Learn PAd project aims at developing a social, collaborative and learning platform for civil servants. By using the Learn PAd platform, workers in the Public Administrations are engaged in a holistic and collaborative learning experience, centered around an enriched model of the to be learned Business Process. In order to better represent the concepts being taught and get the learner engaged, game's aspects can be used.

This chapter introduces a refinement of the Business Process Simulation Environment that we have elaborated to address the above challenges. First, in section 2.1, we briefly remind the overall initial architecture and the functionality of its main components. Section 2.2 tackles the issue of gamification and introduces two main mechanisms for the Learn PAd platform (game scoring and virtual certificate reward). Finally, Section 2.3 focuses on the refinement of the architecture and details the different architectural changes.

2.1. Simulation environment architecture overview

The Learn PAd Simulation Environment provides the subsystem where users can simulate Business Processes interactively. This platform allows multi-sessions BP simulation and monitoring and can engage civil servants in both individual and collaborative simulation allowing them to train on the BP tasks by interacting with other civil servants. Moreover, the simulation environment includes an event-based monitoring framework aiming at providing feedbacks for the learner evaluation and check if the KPI are fulfilled by the learner when execution the simulation of the BP. Figure 2.1 presents the high level architecture includes Business Process simulation engine and the monitoring infrastructure as defined in D6.1 [1].

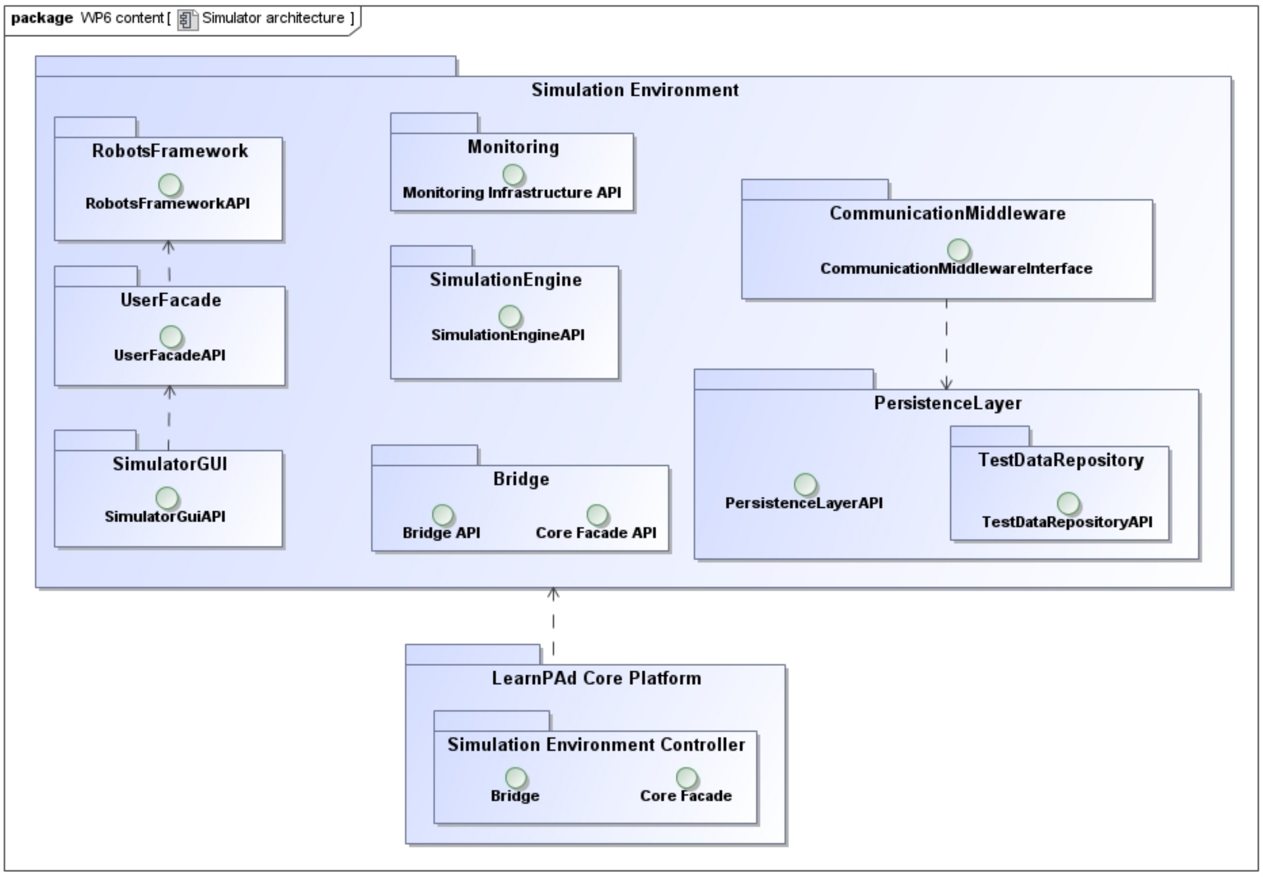


Figure 2.1 Simulation Environment Architecture

As depicted in the Figure above, the simulation framework includes 7 main components:

- **SimulationGUI** is in charge of the interactions between civil servants and simulator's components.
- **PersistenceLayer** stores the status of the simulation at each step in order to give the civil servant the ability to stop it and restart when needed.
- **RobotFramework** allows simulating the behavior of civil servants by mean of robots.
- **SimulationEngine** is the core component. It enacts BP and links activities with corresponding civil servants or robots.
- **Monitoring** collects the events occurred during the simulation and infers rules related to the BP execution. Additionally, the monitor component provides off-line civil servant assessment scores.
- **CommunicationMiddleware** provides event-based communication facilities between the simulation components according to the publish/subscribe paradigm.
- **UserFacade** is in charge of encapsulating civil servants or robots in order to make the learner interaction transparent to the other components of the architecture.

Each simulation component is exposed as a service and provides an API as unique point of access. The simulation platform interacts with the rest of Learn PAd components by means of the Learn PAd Core Platform and specifically through the Bridge component (including the Bridge and the Core Facade interfaces).

Although this initial Learn PAd Simulation Environment architecture covers a wide range of simulation features, it has been conceived, among other things, with a lack of challenges and motivations to engage and immerse civil servants. In section 2.3, we shall explain the departures from this preliminary architecture, namely:

- Integration with the core platform: how the interface with the core platform has been updated with latest developments.
- Addition of gamification mechanisms: what additions did we make to the architecture to handle gamification mechanisms.
- Change of form engine: what were the shortcomings of the initially chosen forms engine, and the replacement we select to address them.

2.2. Gamification challenges for Learn PAd platform

Since 2010, gamification (called also Gameful design) is a hot topic that everybody talks about. In [2], authors define it as the use of game design elements in non-game contexts. This definition is the most comprehensive and the best known. Gamification is a fast growing initiative, with the aim of increasing motivation and participating among users of non-game applications and is expected to revolutionize all aspects of life in the not too distant future [3] [4]. Its adoption in both industry and research has been widespread over a range of different domains (medical, rehabilitation, education, etc.). Gamification in learning and teaching according to Karl Kapp in [5] is the use of game-based mechanics and game thinking to engage, motivate action, promote learning and solve problems. In Deliverable 8.1-Addendum entitled User Perspective and Project Evaluation [6] we present a more detailed state of the art of gamification in a learning setting to motivate and to boost engagement the targeted Learn PAd Learners.

Thereafter, we present two important gamification mechanics that we can find for learning purposes and ensure more engagement with civil servants: **Game Scores** and **Virtual Certificate Reward**.

2.2.1. Game Scores

In learning context, it is important for civil servants to see the success displayed incrementally. To this end, Learn PAd Simulation Environment integrates a scoring mechanism in order to generate ranking of Civil Servants. Different scores are identified: Session Score, Business Process Score and Global Score. These are presented below.

2.2.1.1. Session Scores

During a simulation session, the simulation will provide:

- The session score (called *session_score*), i.e. the ongoing session score of each participating civil servant.
- An assessment value (called *absolute_session_score*) useful as boundary value for the session score.

Specifically, the session score is calculated using a weighted sum of scores attributed to the Civil Servant for each task of the Business Process realized during the simulation. Considering "n" the number of tasks executed by the civil servant during the learning session and "P" the weight of the task, the session score is computed as follows:

$$session\_score = \sum_{i=1}^{n} task\_score_i \, P_i$$

Each task of the Business Process is associated with a weight specified as a metadata. These metadata are attributed in the BP definition and defined by the modeler. The calculation of the score's task is based on several criteria, namely *number of attempts*, *Success/Fail* and finally *KPI* (e.g. response time). The formula below allows calculating this score:

$$task\_score = sucess * (\frac{1}{nb\_attempts} + \sum_{i=1}^{k} (\frac{expected\_KPI\_value}{observed\_KPI\_value}))$$

Where "k" is the number of *KPI* considered in the evaluation of the Civil Servants performances and success is a Boolean.
For what concerns the boundary values useful for the learning assessment, in collaboration with the monitoring component, the simulation environment can provide the *absolute_session_score,* which represents the maximum score that could be assigned to the civil servant during a simulation session. Supposing that the maximum obtained value of the *task_score* is equal to 2, the *absolute_session_score* is computed as:

$$absolute\_session\_score = \sum_{i=1}^{n} 2 \, P_i$$

This *absolute_session_score* computes an accuracy measure of the *session_score*. A *session_score* value closer to the *absolute_session_score* represents a better performance of the civil servant for the considered simulation session.

2.2.1.2. Business Process Scores

During a learning activity of a civil servant, different learning sessions could be executed on the same Business Process, each one referring to a different path. Therefore, the cumulative score obtained by the civil servant on the executed sessions could be considered a good indicator of the knowledge of the civil servant about the overall Business Process. Therefore simulation, through the collaboration with the monitoring component will provide the following values:

- The Business Process Score (called *bp_score*), i.e. the cumulative score obtained by the civil servant after the execution of different simulation sessions on the same business process.
- Two assessment values (called *relative_bp_score* and *absolute_bp_score*) used as boundary values for the *bp_score* to evaluate the acquired civil servant competency on the executed business process.
- A business process coverage percentage (called *bp_coverage*), i.e. the percentage of different learning sessions (paths) executed by the civil servant during the simulation of a business process.

In the following we provide more details about the above-mentioned scores.

The *bp_score* is computed as the sum of the maximum values of session_score(s) obtained by the Civil Servant during the simulation of a set of different k paths (over the overall number of paths) on a business process, according to the following formula:

$$bp\_score = \sum_{i=1}^{k} \max\,(session\_score_i)$$

Considering a *bp_score* and the set of *k* paths to which the *bp_score* is related to, the ***relative_bp_score*** is the boundary value representing the maximum score that the civil servant can obtain on the set of *k* paths. It is computed as the sum of the *absolute_session_score* according to the following formula:

$$relative\_bp\_score\ = \sum_{i=1}^{k} absolute\_session\_score$$

Considering all paths of a business process to which a *bp_score* is related to, the ***absolute_bp_score*** is an additional boundary value representing the maximum score that the civil servant can reach. It is computed as the sum of the *absolute_bp_score* for all the paths of the business process according to the following formula:

$$absolute\_bp\_score\ = \sum_{i=1}^{\#path} absolute\_session\_score$$

More the *bp_score* is close to the *relative_bp_score* more the civil servant reaches the maximum cumulative learning performance on the different simulated sessions.

Values of *bp_score* closer to the *absolute_bp_score* represent more complete civil servant knowledge about the overall business process.

The ***bp_coverage*** value is an additional measure for evaluating the completeness of the civil servant knowledge about the overall business process. It is computed as the percentage of different paths (*k*), executed by the civil servant during the simulation of a business process, over the paths cardinality as in the following:

$$bp\_coverage = \frac{k}{\#path}$$

When the civil servant executes all paths of the business process, the computed *bp_coverage* is 1. A *bp_coverage* value closer to 1 represents a better performance of the civil servant for the considered business process simulation.

2.2.1.3. Global Scores

During a learning activity of a civil servant, different learning sessions could be executed on the same Business Process, each one referring to a different path. Therefore, the cumulative score obtained by the civil servant on the executed sessions could be considered a good indicator of the knowledge of the civil servant about the overall Business Process. Therefore simulation, through the collaboration with the monitoring component will provide the following values:

- The overall score (called *global_score)* obtained by the Civil Servant on all the executed business processes.
- Two assessment values (called *relative_global_score and absolute_global_score*) useful as boundary values for the global score

In the following, we provide more details about the above-mentioned scores.

Considering all business processes executed by a Civil Servant and their *bp_score(s),* the **global_*score*** is computed as the sum of the *bp_score* as in the following formula:

$$global\_score = \sum_{i=1}^{n} bp\_score_i$$

Where "n" is the number of different business processes executed by the Civil Servant.

Considering the business processes executed by a civil servant and their *relative_bp_score*, the ***relative_global_score*** is a boundary value representing the maximum score that the civil servant can obtain on all the executed business processes. It is computed as the sum of the *relative_bp_score* obtained by the civil servant for the different "n" executed business process according to the following formula:

$$relative\_global\_score = \sum_{i=1}^{n} relative\_bp\_score_i$$

Considering the business processes executed by a Civil Servant and their *absolute_bp_score*, the ***absolute_global_score*** represents an additional boundary value representing the maximum score the civil servant can reach on all the executed business processes. It is computed as the sum of the *absolute_bp_score* obtained by the civil servant for the different "n" executed business process according to the following formula:

$$absolute\_global\_score = \sum_{i=1}^{n} absolute\_bp\_score_i$$

2.2.2. Virtual Certificate Reward

While some commercial games may use real gifts (e. g. money, gift cards, etc.), many games motivate players with virtual rewards as certificates. In Learn PAd context, Civil Servants who satisfy some conditions can be automatically awarded by the platform with a specific certificate (predefined by the Business Process modeler). Being awarded a certificate is important to the Civil Servant since it gives to him/her additional rights (represented by roles). For example, a Civil Servant with a good score in some Business Processes simulations and with considerable contributions to the simulation platform is awarded the "*expert*" certificate. This allows him/her to check answers from other Civil Servants for validation during a learning simulation.

The Learn PAd platform constantly monitors the activity of Civil Servants. When one of them satisfy the required obtaining conditions, the platform assigns him/her the corresponding certificate automatically and informs him/her about the new deserved rights.

The actual certificates available for learners are not defined a priori. Administrators to fit the specific needs of the organization can add them to the platform. For example, an administrator could add certificates, such as "beginner", "intermediate", "expert", "best contributor", etc., to the different topic categories previously defined, based on rules he defined himself regarding preexisting conventions present in the organization.

2.3. Simulation infrastructure refined API

This section presents the architectural changes we made in regard of the initial Learn PAd Simulation Environment architecture presented in deliverable 6.1 [1].

2.3.1. Integration with the Core Platform

One of the main changes from the initial architecture concerns the way a simulation is instantiated. This process has been streamlined according to the technical choices made for the integration between the components. The relevant changes are depicted on Figure 2.2.

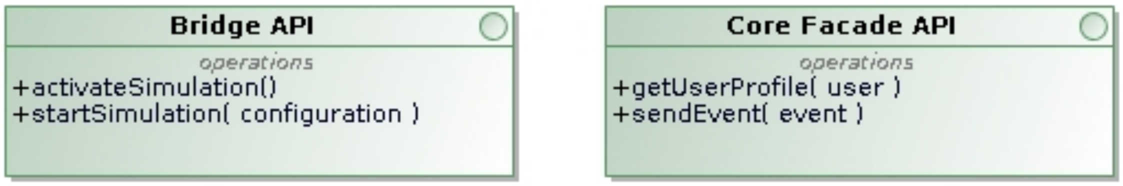


Figure 2.2 Modifications of the Bridge related to simulation instantiation

An illustration on how this integration is put in practice is shown on Figure 2.3. To keep it short, we will depict the case of a single-user simulation, where the related models and resources have already been loaded:

1. The Civil Servant selects a simulation to be instantiated.
2. The Core Platform requests the Simulator to activate the chosen simulation.
3. The Simulator returns a specific URL, containing a User Interface to configure the simulation.
4. The Core Platform displays the URL to the Civil Servant in its chosen fashion.
5. The Civil Servant configures and starts the simulation.
6. The Simulator requests to the Core Platform the profile info of the involved users
7. The Core Platform returns the requested info.
8. The Simulator updates the user UI to display the simulation interface.
9. The Simulator sends an event to the Core Platform to informs it that the simulation has started.
10. The Civil Servant interacts with the simulation.
11. The Simulator sends events to the Core Platform based on the Civil Servant actions.
12. When the Civil Servant has completed the Simulation, the Simulator updates the User Interface to inform him/her.
13. The Simulator then sends an event to the Core Platform notifying it the simulation has completed.

Note that this is a simplified description, as we only detail the interactions between the Simulator and the Core Platform. In the actual implementation, the interactions between the

Core Platform and the Civil Servant are not made directly. Instead, they take place through other intermediate components (such as the Collaborative Workspace).

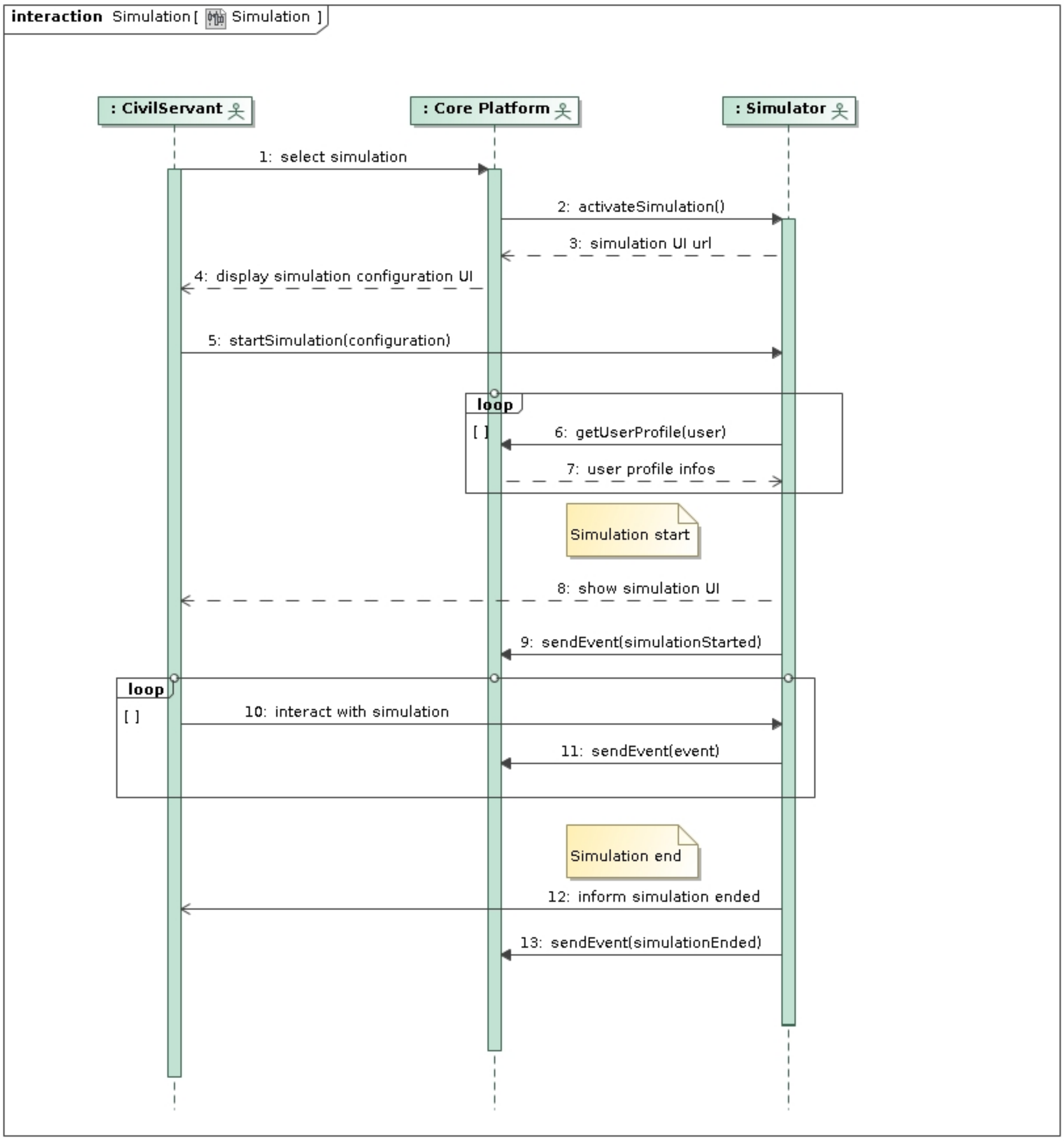


Figure 2.3 Simulation instantiation

2.3.2. Refined architecture for gamification

Figure 2.4 details the overall interaction between the Learn PAd components regarding the technical implementation of the gamification mechanisms. From this view, we can observe that the Simulator is mainly involved in the computation of the users session scores.

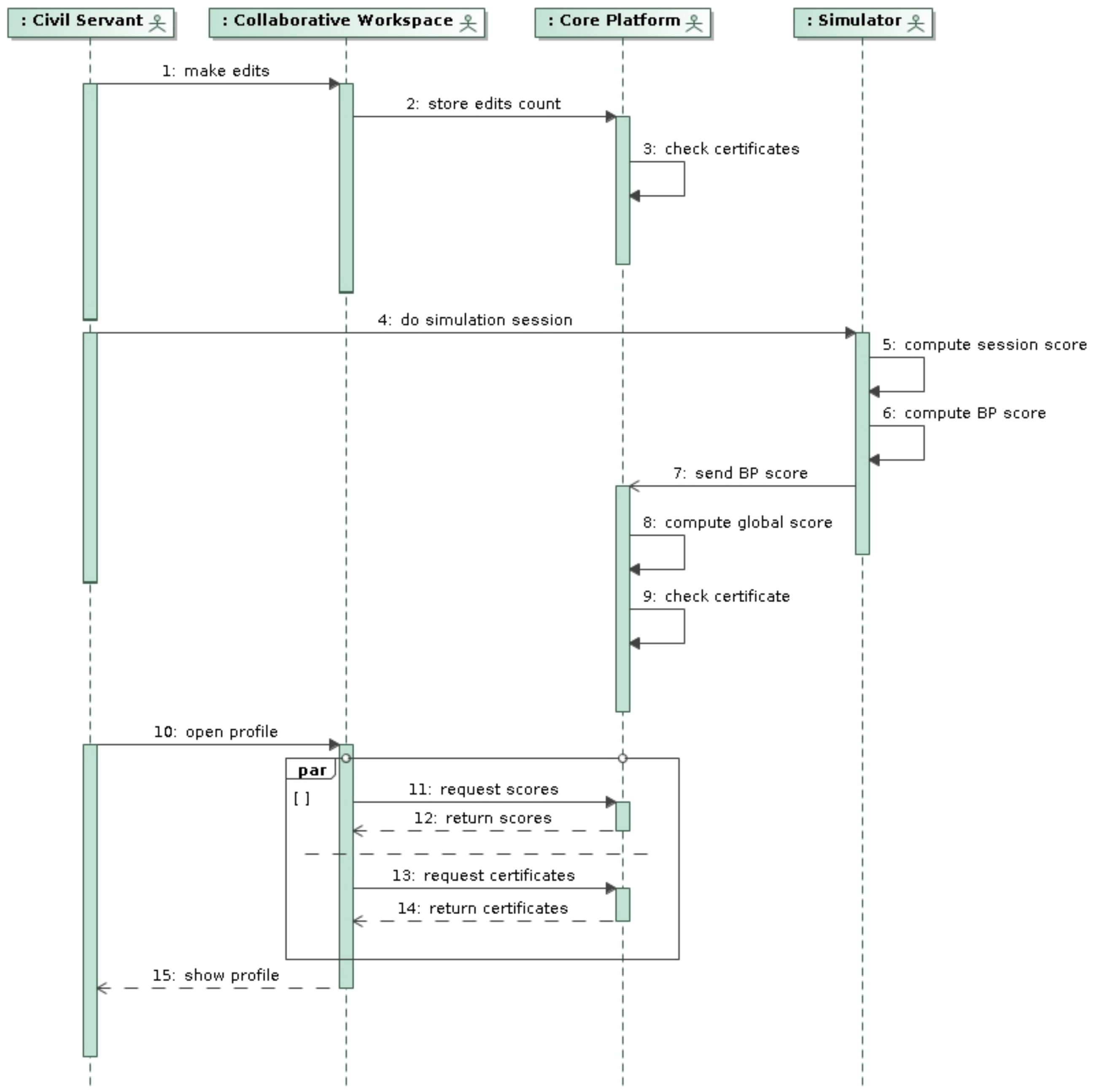


Figure 2.4 Overview of Learn PAd components integration for gamification

Based on these requirements, some components of the architecture were revised to include additional related functionalities. These additions are shown on Figure 2.5. They concern the gathering of indicators for calculating the session scores (number of attempts and time spent on a task), the computation of the ongoing session score, and its display on the Graphical User Interface.

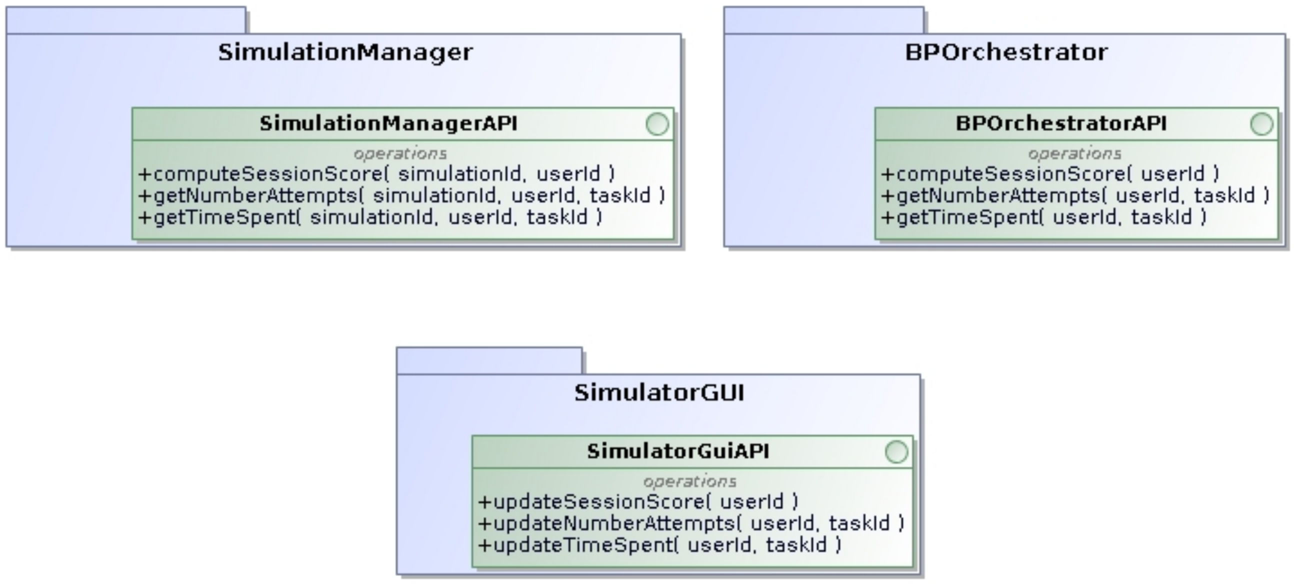

SimulationManager
SimulationManagerAPI
operations
+computeSessionScore(simulationId, userId)
+getNumberAttempts(simulationId, userId, taskId)
+getTimeSpent(simulationId, userId, taskId)
BPOrchestrator
BPOrchestratorAPI
operations
+computeSessionScore(userId)
+getNumberAttempts(userId, taskId)
+getTimeSpent(userId, taskId)
SimulatorGUI
SimulatorGuiAPI
operations
+updateSessionScore(userId)
+updateNumberAttempts(userId, taskId)
+updateTimeSpent(userId, taskId)

Figure 2.5 Architectural modifications inside the Learn PAd Simulator for gamification

2.3.3. Change of forms engine

The initial design of the Learn PAd Simulator involved the user of the FormaaS tool for the generation of the forms submitted to the civil servants. However the use of FormaaS have revealed to be too impractical for this task. The two main reasons for this were:

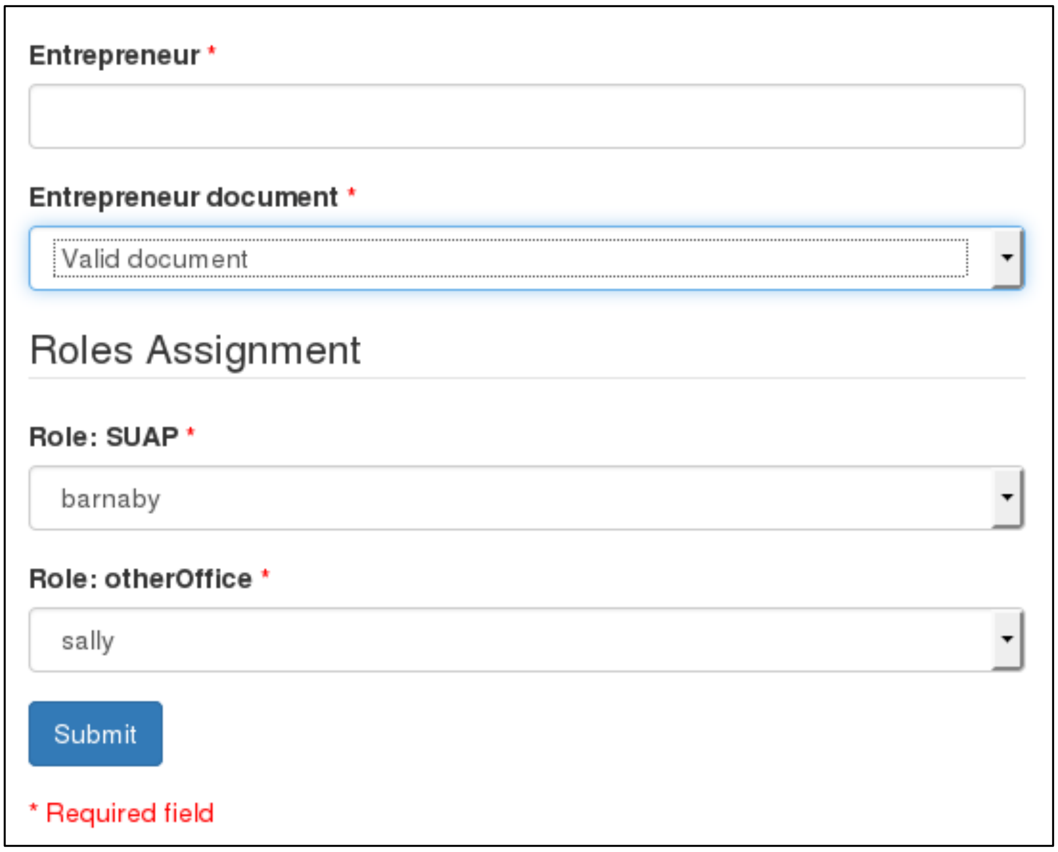
- FormaaS does not allow easily integrating forms into external web pages. This made difficult to create a consistent User Interface for the Simulator.
- FormaaS does not offer notification mechanisms for signaling that a form has been filled and submitted. Consequently, inefficient active pooling was needed to monitor of a Civil Servant had filled a form.

The main advantage of FormaaS concerns its interface for creating forms. However, since the forms are fully automatically generated during the simulation, we did not make use of this functionality.

For these reasons, FormaaS has been replaced with the open-source JavaScript Object Notation (JSON) Form library [7]. JSON Form avoids FormaaS shortcoming in this regard as it is not a full-fledged form creation interface and is instead designed to be integrated in an existing UI. An example for form generated with JSON Form can be seen on Figure 2.6.

2.3.4. Test Data Repository schema

In order to validate the submissions made by the civil servants during a simulation, we defined a first version of the Test Data Repository. The repository stores information about the expected outputs of a task based on the observed inputs (representing contextual information about the case). When a civil servant submits information for validation, the Simulator checks if there is an entry for the task in the test data repository corresponding to the current context, by checking the inputs of the entries corresponding to the task. If one or more entries correspond, their expected outputs are matched with the outputs submitted by the user. If the outputs of at least one entry match so the task submission is validated, otherwise it is rejected.


Entrepreneur *

Entrepreneur document *

Valid document

Roles Assignment

Role: SUAP *

barnaby

Role: otherOffice *

sally

Submit

* Required field

Figure 2.6 Example of a form generated with JSON Form

To represent the content of the repository, a database schema have been defined where the overall repository is defined as a map of processes keys to tasks validation sets:

```
{"process1" : task_validation_set1,
"process2" : task_validation_set2,...}
```

A task validation set correspond to map of tasks keys to a list of validation entries:

```
{"task1": [ entry1, entry2, ...],
"task2": [ entry3, entry4, ...],...}
```

And a validation entry contains observed input values and corresponding expected output values:

```
{"task1": [ entry1, entry2, ...],
"task2": [ entry3, entry4, ...],...}
```

The overall repository schema can be thus represented as follows:

```
{"process1" : {
    "task1" : [
     {
       "input" : {
         input1 : value1,
         input2 : value2, ...},
       "output" : {
         output1 : value1,
         output2 : value2,...}     },
     { "input" : {
     input1 : value3,
     input2 : value4,...},
       "output" : {
         output1 : value5,
         output2 : value6,...}    },...    ],...},...}
```

3. Monitoring Infrastructure

As already presented in D6.1 the simulation environment is equipped with a monitoring facility that allows providing feedbacks on the business process execution and learning activities. In this section, we present the new or updated monitoring sub-components introduced into the Monitoring Infrastructure for score evaluation or improving overall monitoring behavior.

In particular to avoid code details (see sub-section 4.1) the new functionalities are presented here at design level by means of UML notation.

In particular Section 3.1 presents the overall monitoring behavior and its interaction with the rest of the Learn PAd environment. Sub-section 3.2 shows the new architectural view of the monitoring infrastructure detailing the additional sub-component.

3.1. Monitoring use case view

In this section the overall functionalities of the monitoring component are depicted. Figure 3.1 below details the interaction of the monitor component with the Learn PAd environment and specifically with the Simulation Engine and the Learn PAd Core Platform.

In particular the interaction with the latter has been introduced to provide a means for requesting civil servant assessment levels even independently for any ongoing simulation execution. Monitor component has been enhanced in order to collect and store historical data about learners' assessment and simulation executions.

As shown in Figure 3.1 the Simulation Engine interacts with the Monitor Component for sending the BPMN model that is going to be simulated and the events necessary for monitoring the ongoing simulation. These activities include the BPMN paths extraction and the rules generation useful for simulation and data collection. On the other side the Core Platform interacts with the Monitor Component for managing the Civil Servant assessment that includes the updating of the Civil Servant score during a simulation or the retrieving of historical data concerning the assessment level of specific Civil Servants.

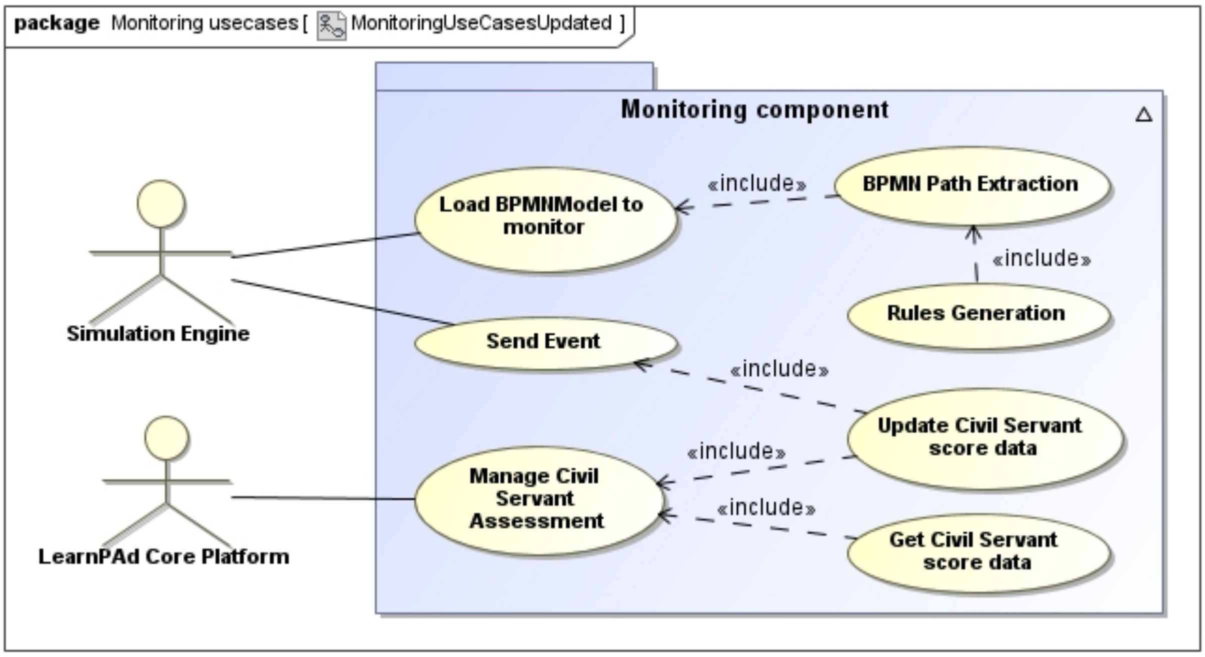

package Monitoring usecases [MonitoringUseCasesUpdated]
Monitoring component
Simulation Engine
LearnPAd Core Platform
Load BPMNModel to monitor
BPMN Path Extraction
«include»
Rules Generation
«include»
Send Event
«include»
Update Civil Servant score data
Manage Civil Servant Assessment
«include»
«include»
Get Civil Servant score data

Figure 3.1 Monitoring components use case view

In Figures 3.2 and 3.3 the implementation of the use cases “*LoadBPMNModel to monitor”* and “*Send Event”* are presented by using activity diagrams.

In particular, as evidenced in Figure 3.2 the Simulation Engine sends the BPMN Model to the Monitor component (specifically to the Monitor Manager sub-component as detailed in section 3.2) that checks some specific properties on the BPMN itself and prepares the proper response for the Simulation Engine.

Each time the Simulation Engine notifies a specific event to the Monitor Component (specifically to the Monitor Manager sub-component as detailed in Section 3.2), the activities of Figure 3.3 are executed. In this case the Monitor Component verifies which of its internal rule could be satisfied and updates the corresponding civil servant accuracy score. The new values are then sent through the Learn PAd Core Platform and made available for specific queries (for instance KPI evaluation).

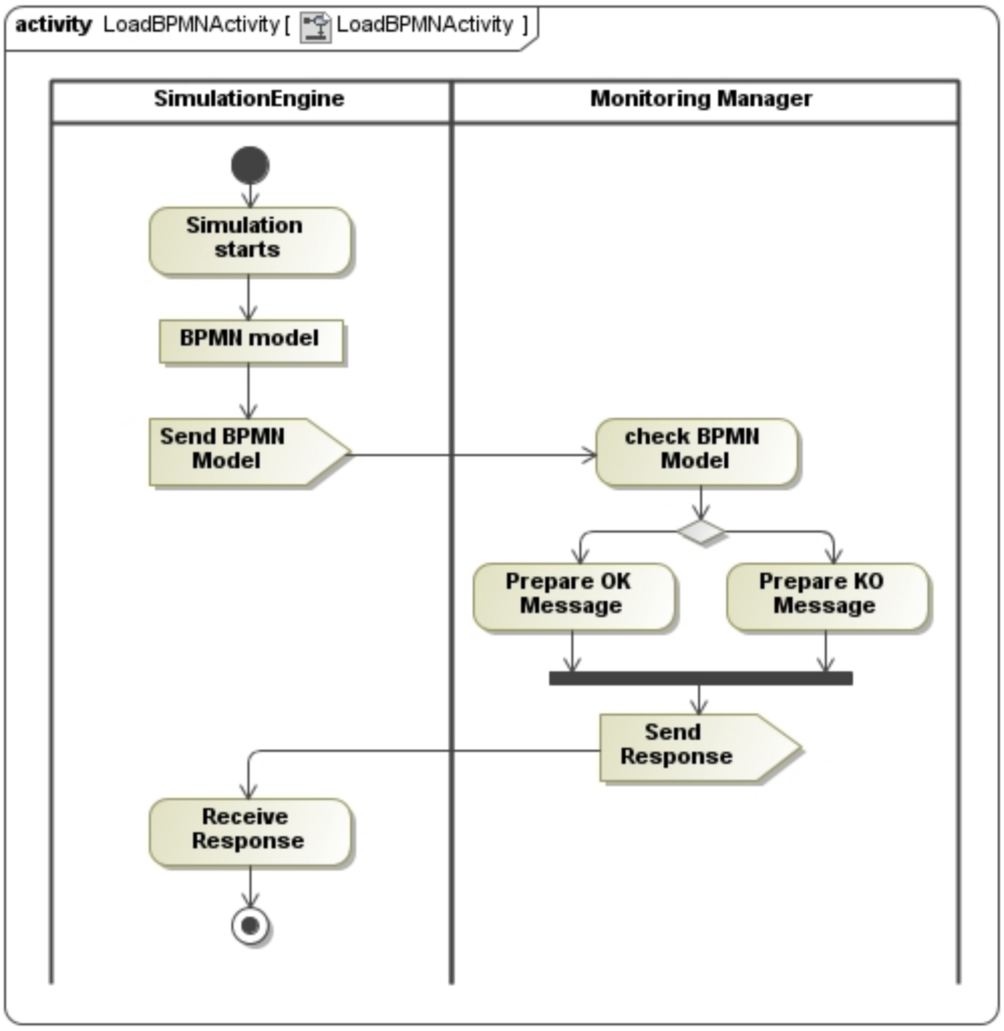


Figure 3.2 LoadBPMNModel to monitor activity diagram

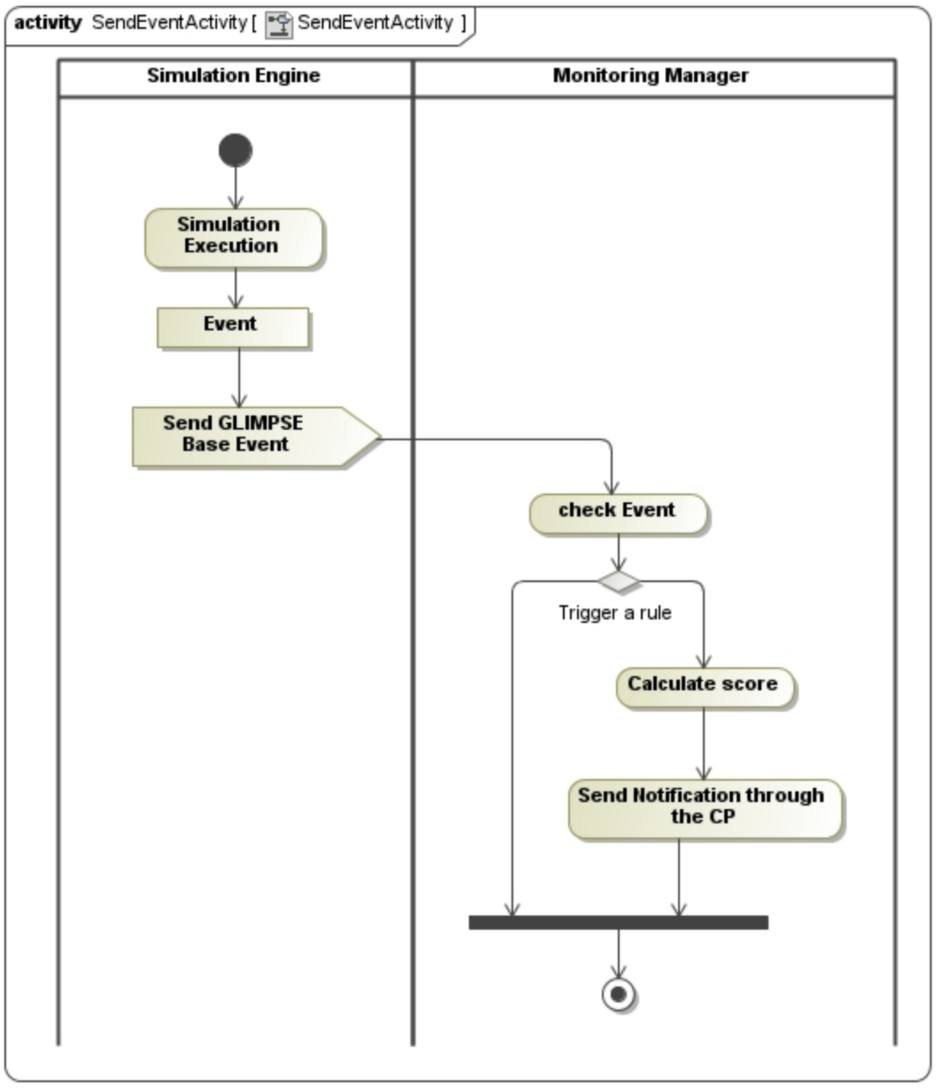


Figure 3.3 Send Event activity diagram

3.2. Monitor component architecture

In D6.1, we provided a first design of the monitoring infrastructure and detailed its components and functionalities. Here a refined and complete design of the monitoring infrastructure is presented. It includes three new components that are:

- DBController. It handles all the interactions within the Data Base useful for monitoring activities.
- Learner Assessment Manager. It evaluates the learner activities and is in charge to calculate the different scores.
- Monitoring Manager component. It is the orchestrator of the overall Monitoring Infrastructure.

These are shown in pink in Figure 3.4 as detailed in Sections 3.2.1 and 3.2.2.

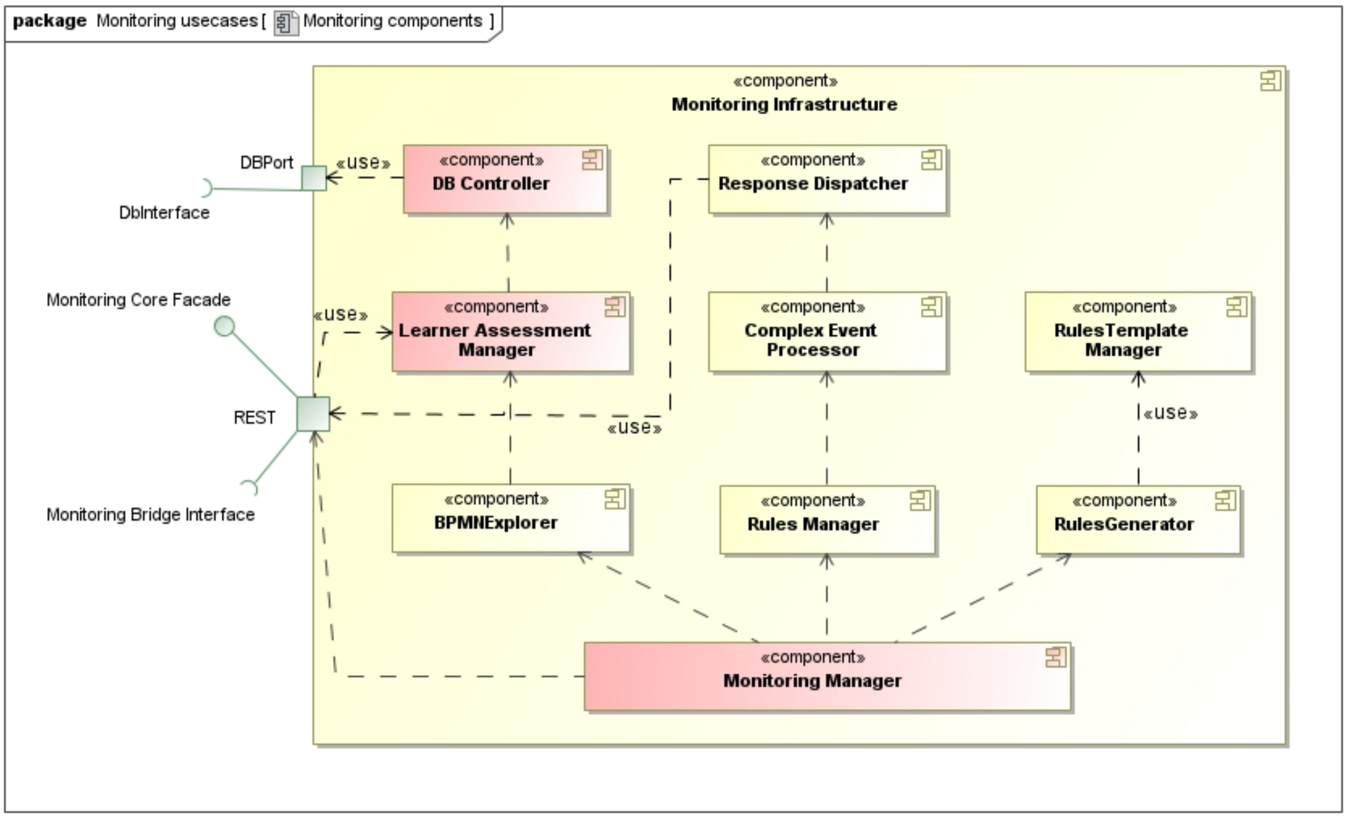

package Monitoring usecases [Monitoring components]
«component» Monitoring Infrastructure
DBPort
DbInterface
«use»
«component» DB Controller
«component» Response Dispatcher
Monitoring Core Facade
«use»
«component» Learner Assessment Manager
«component» Complex Event Processor
«component» RulesTemplate Manager
REST
«use»
«use»
Monitoring Bridge Interface
«component» BPMNExplorer
«component» Rules Manager
«component» RulesGenerator
«component» Monitoring Manager

Figure 3.4 Monitoring components view Event activity diagram

Again for aim of readability, we provide here a brief description and some basic design diagrams. We refer to D6.1 for the complete design description of the remaining monitoring sub-components that are:

- *Complex Event Processor (CEP)*. It is the rule engine, which analyzes the events, generated by the business process execution.
- *BPMN explorer*. It is in charge to explore and save all the possible entities (Activity Entity, Sequence Flow Entity, Path Entity) reachable on a BPMN.
- *Rules Generator*. It is the component in charge to generate the rules needed for the monitoring of the business process execution.
- *Rules Template Manager*. It is an archive of predetermined rules templates that will be instantiated by the Rules Generator.
- Rules Manager. It is in charge to load and unload set of rules into the complex event processor and fire it when needed.
- Response Dispatcher. It is a registry that keeps track of the requests for monitoring sent to the monitoring infrastructure.

More details are available in the source code of their implementation.

3.2.1. Data Base Controller

This component has been introduced to satisfy the Learn PAd requirements of having storage of simulation executions data. Specifically the DB Controller manages the updating of the civil servant score during a simulation or the retrieval of historical data concerning the assessment level of the civil servants.

The DB controller interacts with the Learner Assessment Manager to get the different scores defined in sub-section 2.2.1 and exposes the DBInterface including methods for interacting with the Database. The DB controller guarantees that the implementation of the Learner Assessment Manager is independent from any implementation of the Database that could be internal of the monitoring component or remotely accessed.

The DB design required for the correct Monitoring component execution is reported in Figure 3.5.

3.2.2. Learner Assessment Manager

This new component evaluates the learner activities and is in charge to calculate the different scores. It has been introduced for satisfying the Learn PAd requirements for Civil Servant assessment. It will be further refined according to the requirements, KPIs definition and requirements that will be expressed in the WP5.

Learner Assessment Manager is also in charge of retrieving, whenever necessary and independently from any ongoing simulation, the data necessary for the score evaluation and updating them on the database. Data collected during monitoring of business process execution can be used for providing feedbacks for the continuous tracking of the process behavior and measurement of learning-specific goals.

Through the interaction with the Simulation Engine managed by the *Monitoring Manager* sub-component the Learner Assessment Manager receives the *session_score*s and computes the assessment of the civil servant learning activity.

All scores computed by the Learner Assessment Manager are then stored in the DB by interacting with the DB Controller described in Section 3.2.1.

In the Figure 3.6, we present an extract of the interaction of the different sub-components of the Monitoring Component when the Civil Servant assessment scores have to be managed.

In particular each time the Simulation Engine notifies an event to the Monitor Component (specifically to the Complex Event Processor) and a rule is triggered (i.e. a path on the business process has been completed) the Learner Assessment Manager computes the different assessment scores as above mentioned. It then updates them on the database by interacting with the DB Controller and notifies the Learn PAd Core Platform through the interaction with the Response Dispatcher.

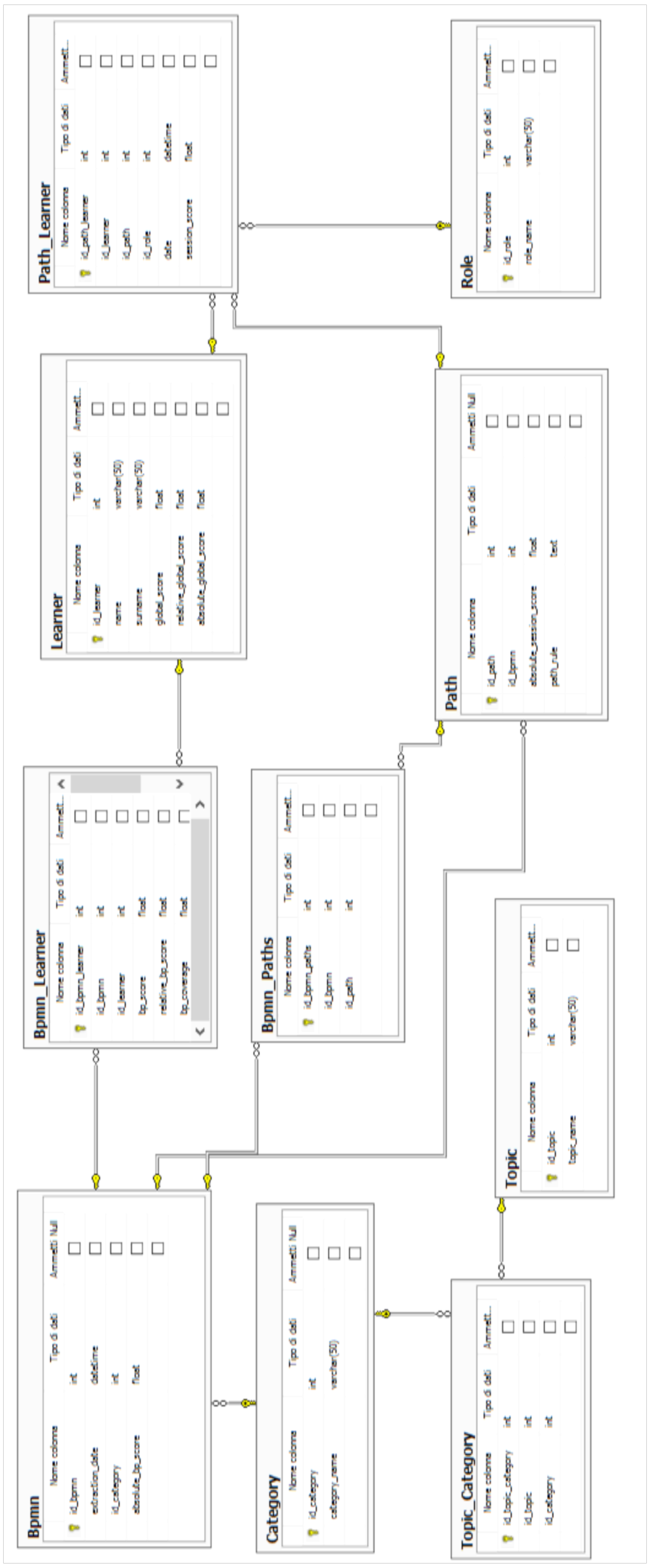


Figure 3.5 Data Base structure

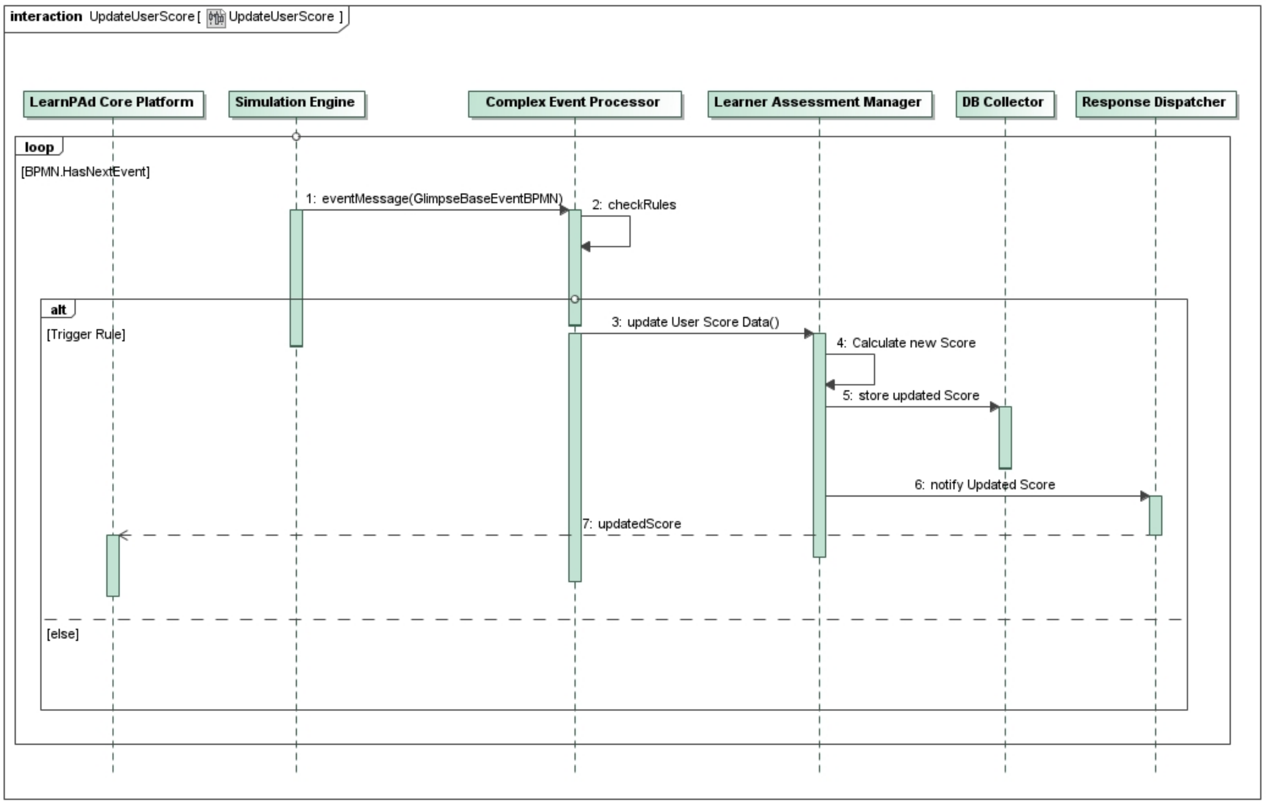


Figure 3.6 Update user scores sequence diagram

As highlighted in the Monitoring Component architecture (Figure 3.4), the Learner Assessment Manager exposes a REST interface for allowing the Learn PAd Core Platform to get Civil Servant assessment scores whenever necessary.

3.2.3. Complex Event Processor

As described in D6.1 business process events collected by means of a set of probes are analyzed and correlated by the Complex Events Processor (CEP). It is the rule engine that analyzes the simple events generated by the simulation engine and correlates them in order to infer more complex events. It is realized using Drools Fusion that is able to correlate events and detect patterns of the business process simulation. In particular, the BPMN event structure used in the CEP has been enhanced with respect to that presented in D6.1 as shown in Figure 3.7. The new information included in the simple event structure is:

- The time expected for the completion of the event.
- The ID of the simulation session in which the event has been detected.
- The ID of the BP task.
- The ID of the SubProcess to which the event refers to and finally the role of the learner who executes the event.

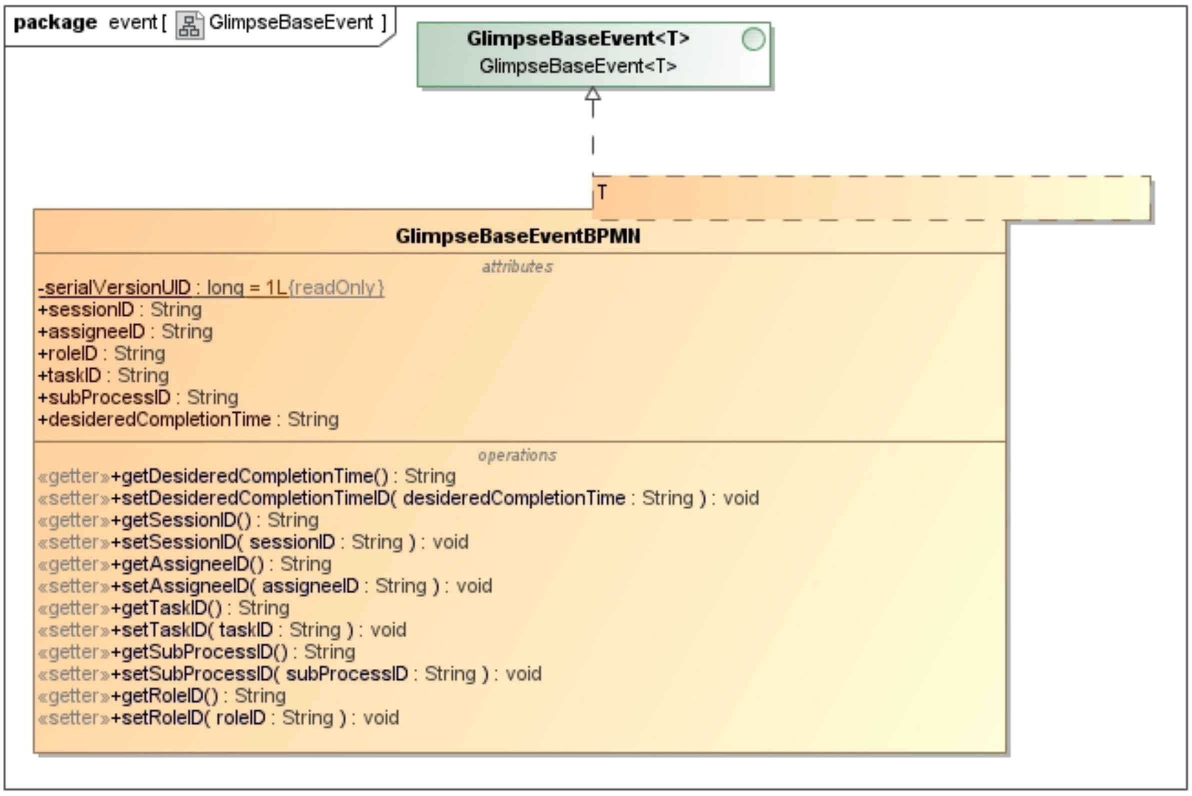

package event [GlimpseBaseEvent]
GlimpseBaseEvent<T>
GlimpseBaseEvent<T>
T
GlimpseBaseEventBPMN
attributes
-serialVersionUID : long = 1L{readOnly}
+sessionID : String
+assigneeID : String
+roleID : String
+taskID : String
+subProcessID : String
+desideredCompletionTime : String
operations
«getter»+getDesideredCompletionTime() : String
«setter»+setDesideredCompletionTimeID(desideredCompletionTime : String) : void
«getter»+getSessionID() : String
«setter»+setSessionID(sessionID : String) : void
«getter»+getAssigneeID() : String
«setter»+setAssigneeID(assigneeID : String) : void
«getter»+getTaskID() : String
«setter»+setTaskID(taskID : String) : void
«getter»+getSubProcessID() : String
«setter»+setSubProcessID(subProcessID : String) : void
«getter»+getRoleID() : String
«setter»+setRoleID(roleID : String) : void

Figure 3.7 Event structure

3.2.4. Monitoring Manager

It is the orchestrator of the overall Monitoring Infrastructure. It interacts with the Learn PAd Core Platform through the REST interfaces (core facade and bridge interface) and is in charge to query the Rules Manager. It interacts with the BPMN Explorer and the Rules Generator. This component initializes the overall monitoring infrastructure allocating resources, instantiating the Complex Event Processor and instrumenting channel on which events coming from the simulation engine will flow.

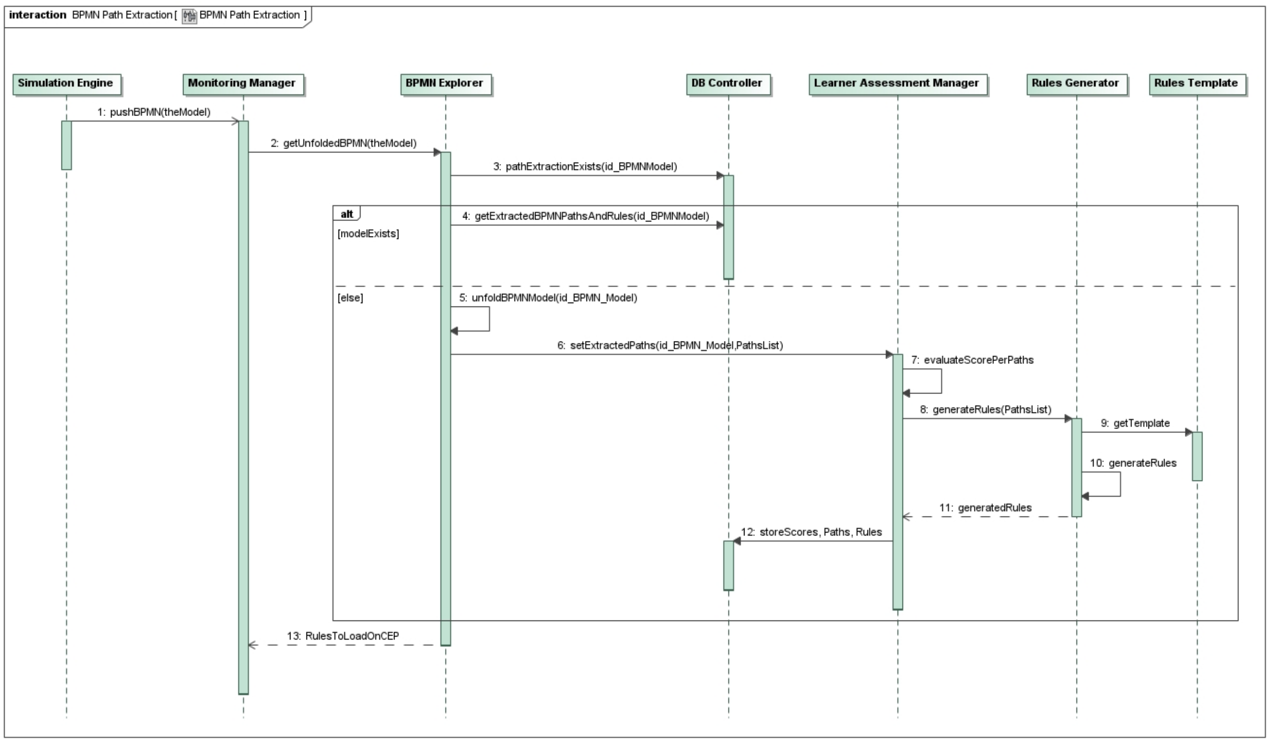

interaction BPMN Path Extraction [BPMN Path Extraction]
Simulation Engine
Monitoring Manager
BPMN Explorer
DB Controller
Learner Assessment Manager
Rules Generator
Rules Template
1: pushBPMN(theModel)
2: getUnfoldedBPMN(theModel)
3: pathExtractionExists(id_BPMNModel)
alt
[modelExists]
4: getExtractedBPMNPathsAndRules(id_BPMNModel)
[else]
5: unfoldBPMNModel(id_BPMN_Model)
6: setExtractedPaths(id_BPMN_Model,PathsList)
7: evaluateScorePerPaths
8: generateRules(PathsList)
9: getTemplate
10: generateRules
11: generatedRules
12: storeScores, Paths, Rules
13: RulesToLoadOnCEP

Figure 3.8 Monitor Manager structure

4. Working with the Learn PAd Simulation Environment

In the following we provide a brief tutorial on how to access the code currently under development. Only for official review purposes, a frozen version that contains the prototype formally released at M21 along with this deliverable can be instead retrieved from the link: http://www.learnpad.eu/docs/lp-simulation-environment-D6_2.tgz

4.1. Get the sources

The Learn PAd Simulation Environment, containing both the simulation engine and the monitoring component, is implemented in Java language and its source code is stored in the main Learn PAd platform repository. This platform repository is accessible using this link: https://github.com/LearnPAd/learnpad.git.

Precisely, the source code of the simulator is stored from the "*lp-simulation-environment*" subfolder (https://github.com/LearnPAd/learnpad/tree/master/lp-simulation-environment). It can be downloaded with the following command:

git clone https://github.com/LearnPAd/learnpad.git

Inside the *lp-simulation-environment* subfolder live the simulation engine and the monitoring infrastructure, respectively in the *simulator* and *monitoring* folders.

4.2. Building and running the Learn PAd Simulation Environment

Once, the source code is downloaded, we need to build the simulation and monitoring platforms. To do so, go into the specific platform folder, and run relative the build bash script (./build).

It should create a new folder named "*out*" inside the "*lp-simulation-environment*", containing the executable files for the simulator, as well as *start* and *stop* scripts.
The simulation environment is normally built as a part of the whole platform (as described in the *README*[1] of the overall Learn PAd Platform). However it can also be built independently.

After building the simulation environment, the following command allows you to run the simulator:

./out/start

Then, to access the Learn PAd Simulator GUI, you should:

- Open a web browser
- Enter this URL http://localhost:8081

Finally, when you want to stop the simulation platform, from your terminal, you have to use this command:

./out/stop

[1] The readme is available on: https://raw.githubusercontent.com/LearnPAd/learnpad/master/README.md

5. Using the Learn PAd Simulation Environment

5.1. Process initialization

Currently, it is only possible to load a new Business Process definition to the simulator via the Core Platform or the Simulator REST API. In order to load a BPMN file named "processfile.bpmn20.xml" inside the Simulator via the REST API, you have to enter the following command in the terminal:

```
curl --verbose --request POST --header "Content-Type: application/json" --data-ascii "file:///path/to/processfile.bpmn20.xml" "http://localhost:8081/learnpad/sim/processes"
```

5.2. Process activation

Before running a simulation, you need to configure it. To do so, you have to go the simulator Web interface using this URL: http://localhost:8081.

At the landing page, click on "Launch a Process" (see Figure 5.1).

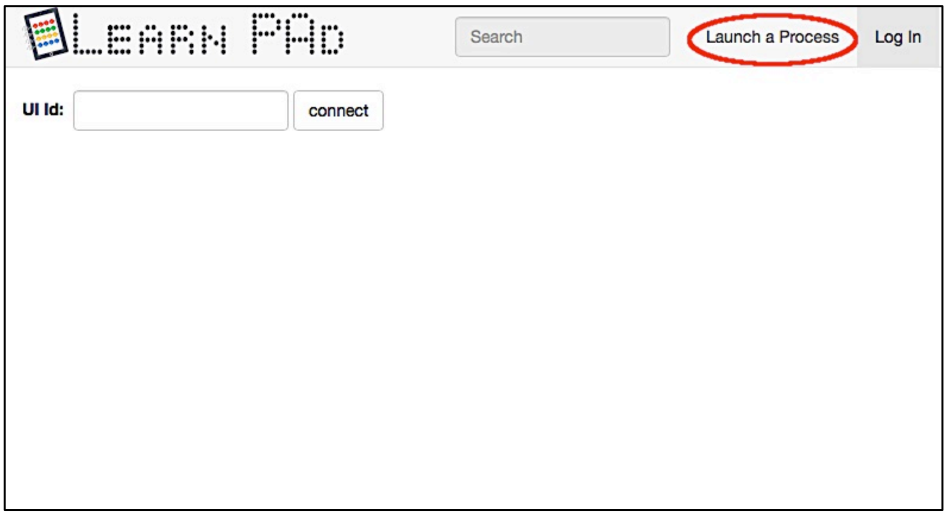
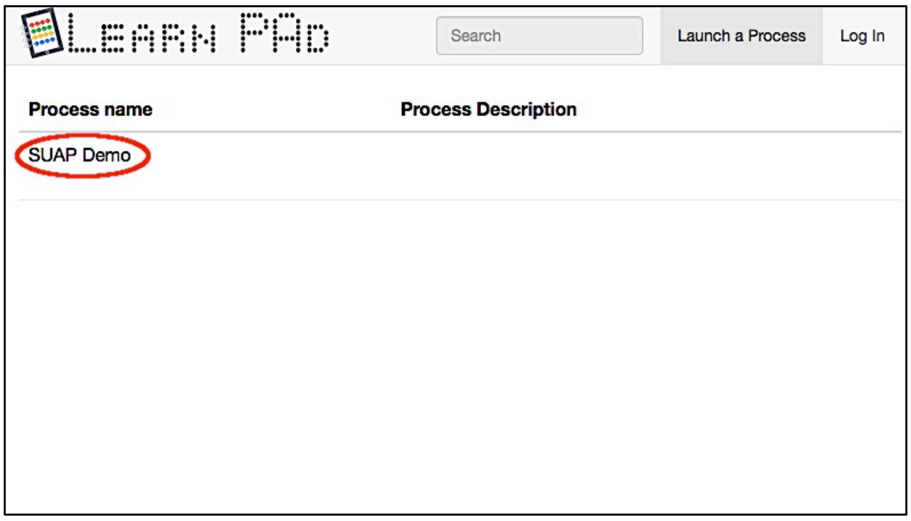


Figure 5.1 Process activation – Step 1

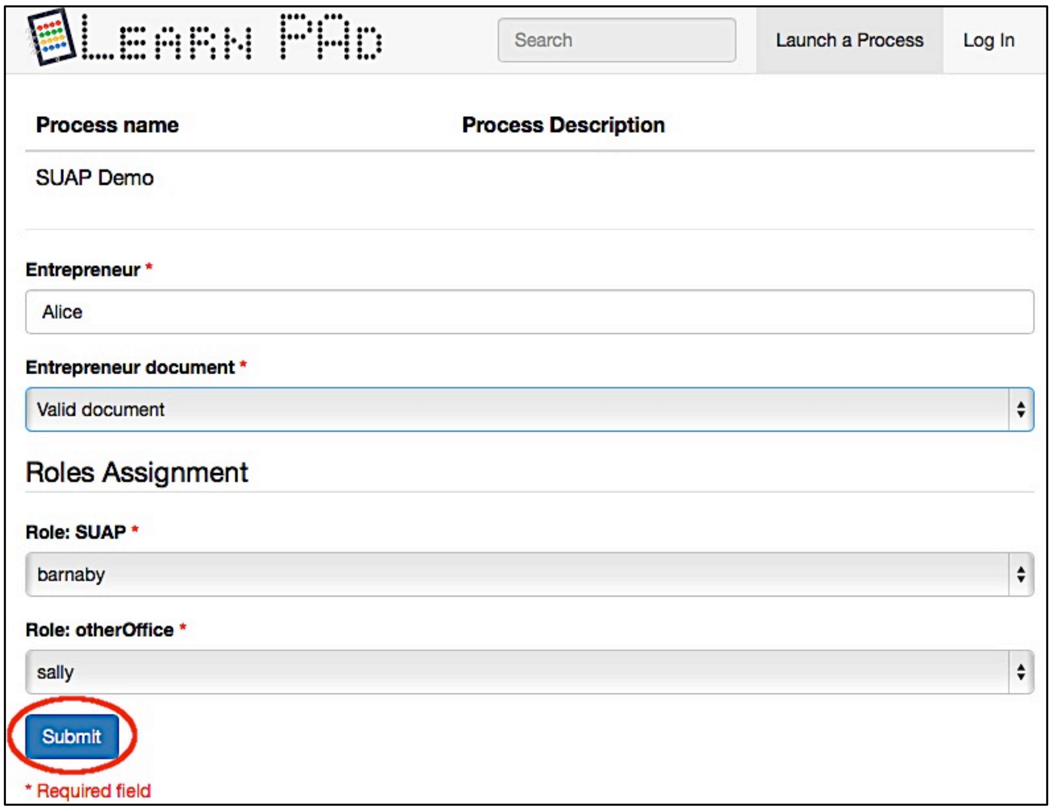
This should open another page in a new browser tab, displaying the list of available processes (see Figure 5.2). Then, click on the name of the process you want to activate.



Process name	Process Description
SUAP Demo	

Figure 5.2 Process activation – Step 2

This should display a form containing the configuration options for the process, along with the process roles to be assigned to users. After setting all the configuration options to the desired values, click on the "*Submit*" button to continue (Figure 5.3).



Process name	Process Description
SUAP Demo	

Entrepreneur *
Alice

Entrepreneur document *
Valid document

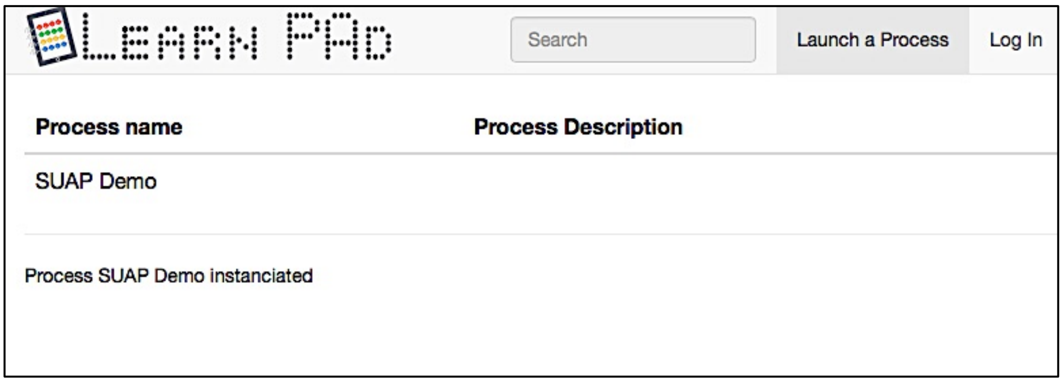
Roles Assignment

Role: SUAP *
barnaby

Role: otherOffice *
sally

Figure 5.3 Process activation – Step 3

A message should confirm that the process has been instantiated correctly (Figure 5.4).



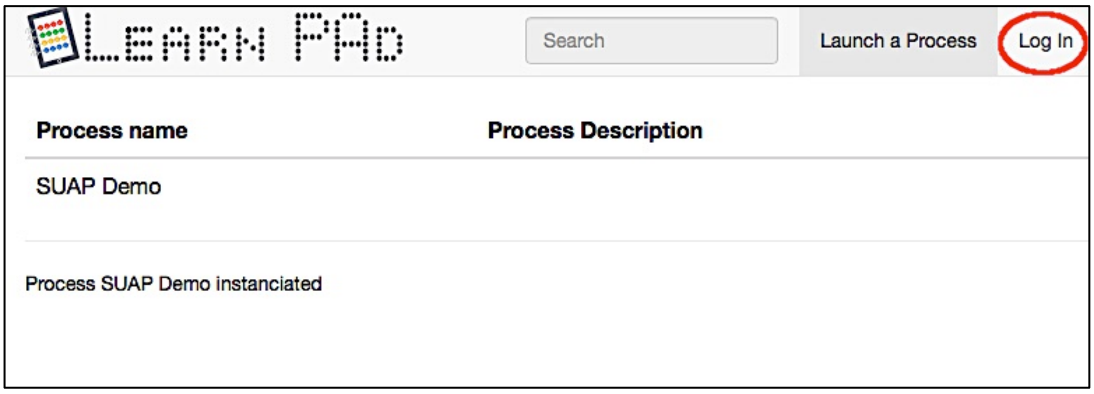
Process name	Process Description
SUAP Demo	

Process SUAP Demo instanciated

Figure 5.4 Process activation – Step 4

5.3. Process execution

After a process has been instantiated, the simulation is launched and the users can access it. From the Web Interface, you need to click on the "*Log In*" button (see Figure 5.5).

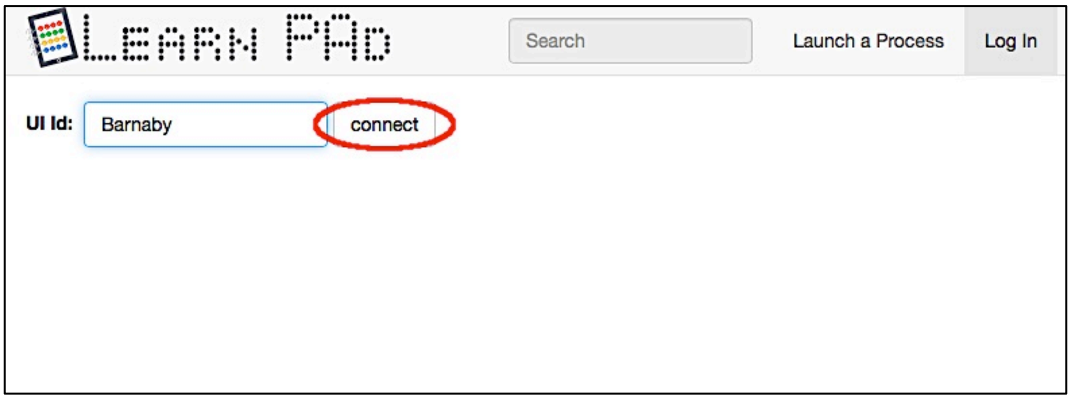


Process name	Process Description
SUAP Demo	

Process SUAP Demo instanciated

Figure 5.5 Process execution – Step 1

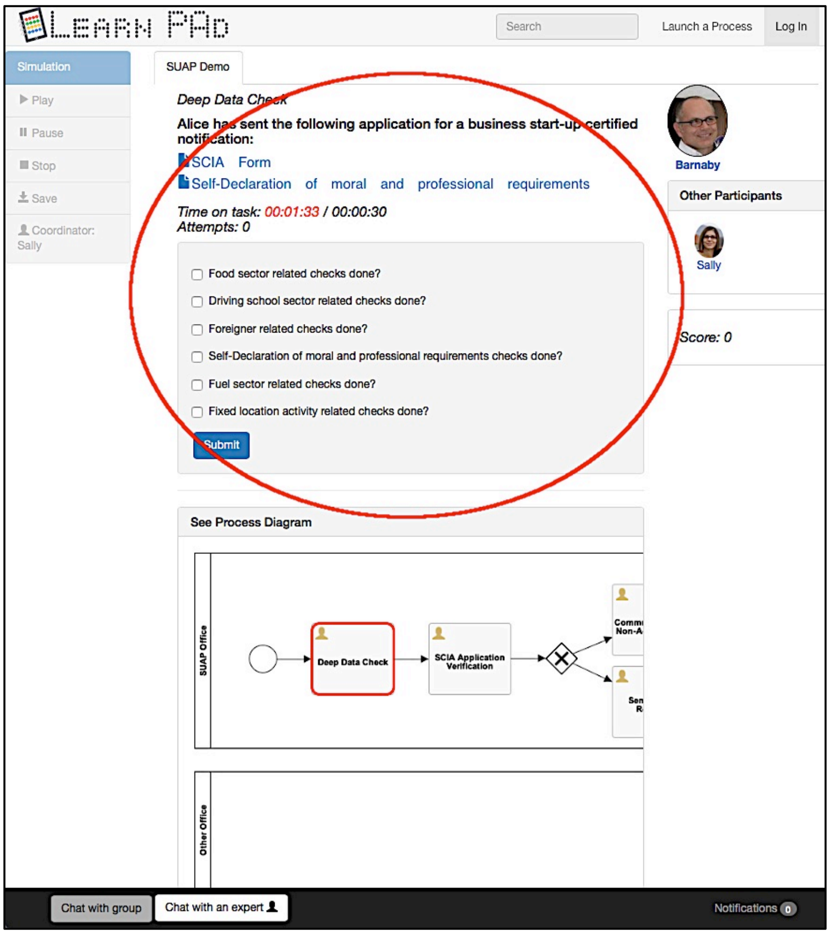
From there, you can connect with the ID of a user involved in the simulation, by entering the corresponding ID and clicking on "connect" (Figure 5.6).



UI Id: Barnaby

Figure 5.6 Process execution – Step 2

At connection, the interface should show the user the tasks to be completed (if any at this point of the process). For each task, the interface will display its name, a description, some information about time on task and number of attempts, as well as a form to be filled to complete the task (see Figure 5.7).



Deep Data Check

Alice has sent the following application for a business start-up certified notification:

SCIA Form

Self-Declaration of moral and professional requirements

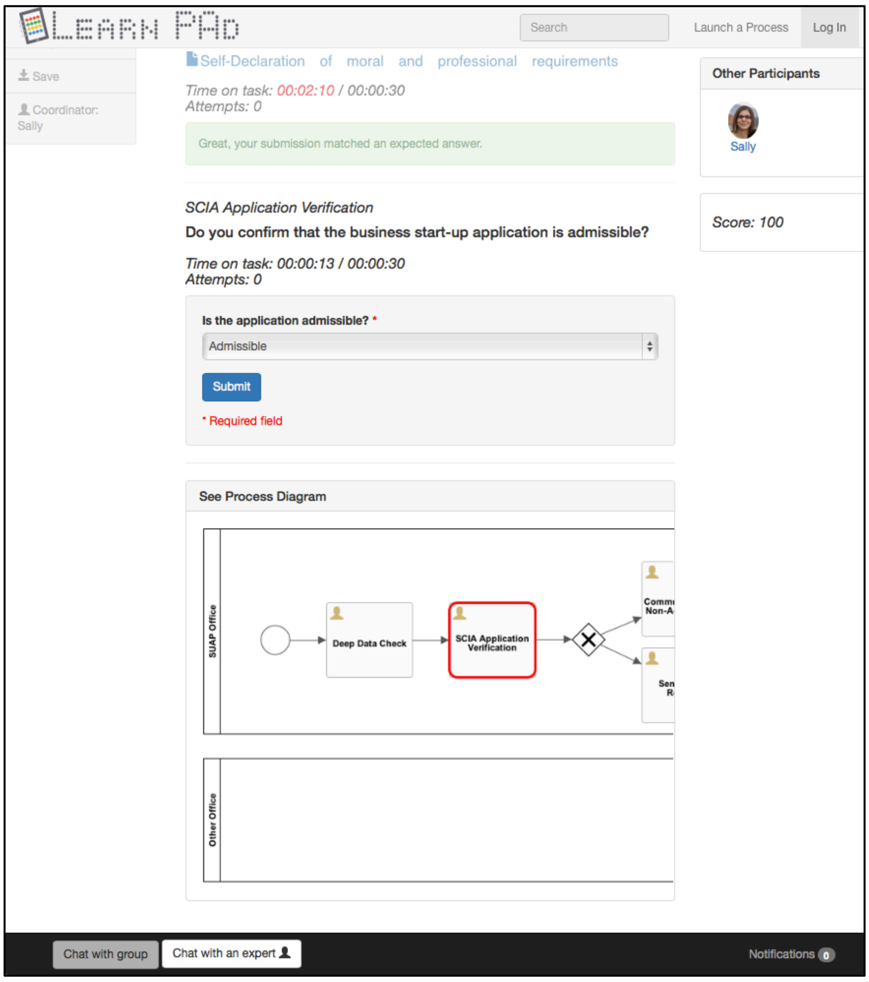
Time on task: 00:01:33 / 00:00:30
Attempts: 0

- [ ] Food sector related checks done?
- [ ] Driving school sector related checks done?
- [ ] Foreigner related checks done?
- [ ] Self-Declaration of moral and professional requirements checks done?
- [ ] Fuel sector related checks done?
- [ ] Fixed location activity related checks done?

Submit

Figure 5.7 Process execution – Step 3

After submitting the task, if all the entries are correct, the simulator will indicate that the task has been validated, and will display new tasks corresponding to the continuation of the process (Figure 5.8). Otherwise the simulator will indicate that the submission is incorrect and the user will have to try again.


Self-Declaration of moral and professional requirements

Time on task: 00:02:10 / 00:00:30
Attempts: 0

Great, your submission matched an expected answer.

SCIA Application Verification
Do you confirm that the business start-up application is admissible?

Time on task: 00:00:13 / 00:00:30
Attempts: 0

Is the application admissible? *
Admissible

* Required field

Other Participants
Sally

Score: 100

Figure 5.8 Process execution – Step 4

6. Conclusions and Plans of Future Work

This deliverable has reported on the definition of the refined Learn PAd Business Process Simulation Engine and Monitoring Component architecture, a first release of its initial prototype and technical documentations.

The preliminary Learn PAd Simulation Environment architecture has been conceived with a lack of challenges and motivations to engage and immerse civil servants. In D6.2, we have revisited and improved it by introducing new gamification mechanisms and we have detailed their functionalities and their interactions. To tackle these changes, the REST APIs have been also enhanced. Moreover, in order to provide a better user experience, we updated the Simulator Forms Engine (from FormaaS to JSONForm).

We also proposed a first version of the Test Data Repository data schema, to provide a first implementation of validation mechanisms for users submissions.

In this deliverable, we have also provided the description of the improved version of the monitoring component included inside the simulation environment. Updates concern mainly the Civil Servant learning assessment by means of calculation of scores relative to the overall BP execution. This document includes also a description of the delivered source code. Indeed, we provide a user-guide that helps to download, build, run and use the Learn PAd simulator.

The implementation of some components of the Learn PAd Simulation platform such as the Persistence Layer, the Robots Framework, Learner Assessment Manager, and DB Controller have to be finished in order to provide a common and integrated infrastructure.

References

[1] Learn PAd Project Team. D6.1: Learn PAd Simulation Environment: Specification and Design. Technical Report, January 2015

[2] Deterding, S., Khaled, R., Nacke, L., & Dixon, D. Gamification: Toward a definition. CHI 2011. Presented at the Computer Human Interaction, Vancouver, British Columbia, Canada: ACM

[3] Chatfield, T. (2010). Why games are the 21st century's most serious business. London: The Royal Society for the Encouragement of Arts.

[4] Schell, J. (2011). The pleasure revolution: Why games will lead the way. Google Tech Talk.

[5] KAPP, Karl M. The gamification of learning and instruction: game-based methods and strategies for training and education. John Wiley & Sons, 2012.

[6] Learn PAd Project Team, D8.1-Addendum: User Perspective and Project Evaluation Strategies, July 2015

[7] JSON Forim Library, available at: https://github.com/joshfire/jsonform.