Horizon 2020 Societal challenge 5

Climate action, environment, resource

Efficiency and raw materials

D4.3: GAME LOGICS AND GENERAL SYSTEM REQUIREMENTS

LEAD AUTHOR: Francesc Guitart

OTHER AUTHORS: Angelos Anagnostopoulos, Gabriel Anzaldi, Chengzi Chew, Olympia Daskalou, Tim Verwaart, Eugene Westerhof

DATE: (31 – January –2017)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement NO 689150 SIM4NEXUS

PROJECT	Sustainable Integrated Management FOR the NEXUS of water-land-food-energy-climate for a resource-efficient Europe (SIM4NEXUS)
PROJECT NUMBER	689150
TYPE OF FUNDING	RIA
DELIVERABLE	D.4.3 Game Logics and general system requirements
WP NAME/WP NUMBER	Serious Game development and testing / WP 4
TASK	Task 4.2 Game logic definition – system requirements
VERSION	V3.4
DISSEMINATION LEVEL	Public
DATE	31/01/2017 (Date of this version) – 31/01/2017 (Due date)
LEAD BENEFICIARY	Eurecat
RESPONSIBLE AUTHOR	Gabriel Anzaldi (EURECAT)
AUTHOR(S)	Angelos Anagnostopoulos (EPSILON), Chengzi Chew (DHI), Olympia Daskalou (EPSILON), Francesc Guitart (EURECAT), Tim Verwaart (WUR), Eugene Westerhof (WUR)
INTERNAL REVIEWER	Xavier Domingo (EURECAT)

DOCUMENT HISTORY

VERSION	INITIALS/NAME	DATE	COMMENTS-DESCRIPTION OF ACTIONS
1	EURECAT	14/12/2016	1ST VERSION
2	ALL PARTNERS	23/01/2017	REVISION OF THE MAIN CONTRIBUTIONS TO THE 1ST VERSION
3	ALL PARTNERS	30/01/2017	FINAL VERSION

Table of Contents

Executive summary

Dissemination and uptake

To learn by doing is one of the main objectives of the Serious Game in order to let the game players learn the main interactions between NEXUS components. The immersive approach provided by Games is a unique method to provide a learning environment playing with economic, environmental, agricultural, touristic, etc. concepts. However, to be able to merge both learning and playing capabilities using gaming interfaces requires the development of procedures that permit to effectively present the desired concepts. These procedures are related to a step by step logic that maintain the user interested in a story development and motivate the user to reach the main learning goals. The efforts in this Deliverable have been invested in developing these procedure called Game Logics, depicted through flowcharts which try to define the options, actions and states that the users can face during game play.

For the development of the NEXUS Serious Game, the requirements for software and hardware components have been specified in this Deliverable, dividing this task into two main parts: Serious Game and Knowledge Elicitation Engine requirements specification. Through the definition of KEE requirements, it can be seen how it is an important piece in the Serious Game engine, as it will serve as a backbone for both information and knowledge during user interaction. The KEE will generate useful information for the NEXUS assessment, taking as input the interactions between the Serious Game and the user. To this end a first proposal for the production and development environments is presented in this Deliverable, as well as the main technologies used making stress in the openness and interoperability capabilities of the results obtained after its development.

Changes with respect to the DoA

Not applicable

Short Summary of results (<250 words)

The main results are the development of the Game Logics flowcharts focused in the user interaction during the game setup and the gameplay flowchart. Also the role of the user in the game has been defined by means of an entity-relationship diagram. In regard to the system requirements, they have been divided into two groups: Serious Game requirements and KEE requirements. Both software and hardware requirements have been specified providing a development environment based on VMs and specifying how this development environment will be deployed to production. Some technical solutions for the development have been proposed making a stress on the openness and interoperability of the final solution.

Evidence of accomplishment

Not applicable

Glossary / Acronyms

TERM	EXPLANATION / MEANING
ABM	AGENT BASED MODELLING
API	APPLICATION PROGRAMMING INTERFACE
ASF	APACHE SOFTWARE FOUNDATION
CRAN	COMPREHENSIVE R ARCHIVE NETWORK
DSS	DECISION SUPPORT SYSTEM
GIS	GEOGRAPHIC INFORMATION SYSTEM
GML	GEOGRAPHY MARKUP LANGUAGE
GPU	GRAPHICS PROCESSING UNIT
GUI	GRAPHICAL USER INTERFACE
HTML	HYPERTEXT MARKUP LANGUAGE
HTTP	HYPERTEXT TRANSFER PROTOCOL
HUD	HEAD UP DISPLAY
IE	INFERENCE ENGINE
JDK	JAVA DEVELOPMENT KIT
JRE	JAVA RUNTIME ENVIRONMENT
JVM	JAVA VIRTUAL MACHINE
KEE	KNOWLEDGE ELICITATION ENGINE
LXC	LINUX CONTAINER
OGC	OPEN GEOSPATIAL CONSORTIUM
OS	OPERATING SYSTEM
PC	PERSONAL COMPUTER
P&R	PENALTIES AND REWARD
REST	REPRESENTATIONAL STATE TRANSFER
SDI	SPATIAL DATA INFRASTRUCTURE

TERM	EXPLANATION / MEANING
SDM	SYSTEM DYNAMIC MODEL
SOAP	SIMPLE OBJECT ACCESS PROTOCOL
SQL	STRUCTURED QUERY LANGUAGE
SRTM	SHUTTLE RADAR TOPOGRAPHY MISSION
VM	VIRTUAL MACHINE
WCS	WEB COVERAGE SERVICE
WFS	WEB FEATURE SERVICE
WMS	WEB MAP SERVICE
WPS	WEB PROCESSING SERVICE
WS	WEB SERVICE
XML	EXTENSIBLE MARKUP LANGUAGE

1 Introduction

1.1 Scope

The current Deliverable aims to formalize Game Logics and Systems requirements. From the Game Logics point of view the game execution flow will be defined taking into account the learning goals definition from D4.1, but also the requirements coming from WP1, WP2, WP3 and WP5. The final objective is to define the logics behind the game to achieve the learning objectives following the “to learn by doing” procedure.

Flowcharts included in this document will define the steps to follow during the game execution and the responses provided to the game user. From the system requirements point of view, the information included in the system and the architecture needed to assist the game development will be specified. To this end, the outputs of Deliverable 4.1 will be used, as well as the many interactions with WP3 in order to structure the way that both the system and the Game will interact with the System Dynamic Models (SDM) and the Thematic Models. The main objective is to be able to sketch how the system will handle and manage the data to provide an immersive environment to improve and maximise the NEXUS learning results. While Game GUI and Logics is one of the most important pieces in the “to learn by doing”, the steps to guide the user, the options offered and the recommendations to take among all the options are also very important pieces to make the user feels in the game and understands the losses and gains. The software and hardware requirements to use in the Serious Gaming tool will also be defined.

1.2 Structure of the Document

This report is structured in 5 Chapters:

- **Chapter 2:** details the Game Logic definition and some important information to be included in the game, such as the user information. Through the construction of flowchart, the game logics will be described at high level. Also the game setup procedure is depicted through a flowchart. The user information needed during the game is sketched by means of an entity-relationship diagram followed by a form describing the desired information from the user to be captured by the KEE.
- **Chapter 3:** details the General Systems Requirements, splitting the Serious Game part and the KEE part. For the Serious Game, some GUI proposals are presented as well as the general architecture regarding communication and controllers. For the KEE, the different sub modules are presented, as well as an API that will permit the Serious Game to communicate with the KEE but also it will permit any other system to query and therefore get NEXUS assessment from the KEE.
- **Chapter 4:** concludes this deliverable, highlighting the main results and describing the future steps and work to do in coming months.
- **Chapter 5:** provides some references used throughout the document and provides some further reading in case the reader needs more specific information about the topics covered along this deliverable.

2 Game Logic Definition

The main objective of the Serious Game tool is to assist **policy makers and stakeholders to better understand and visualise policies** at various geographical and temporal resolutions, leading towards a better scientific understanding of the Nexus via unique immersive experience.

The game component for this project is both a visualisation tool to show results from the Knowledge Elicitation Engine (KEE) as well as a tool to explain to policy makers, students, and practitioners how different policies from Food, Energy and Water sector influence one another and therefore improves the way current policies are being made.

Figure 1 shows the policy making process in Europe currently (before integration and partial integration) and what the project aims to achieve.

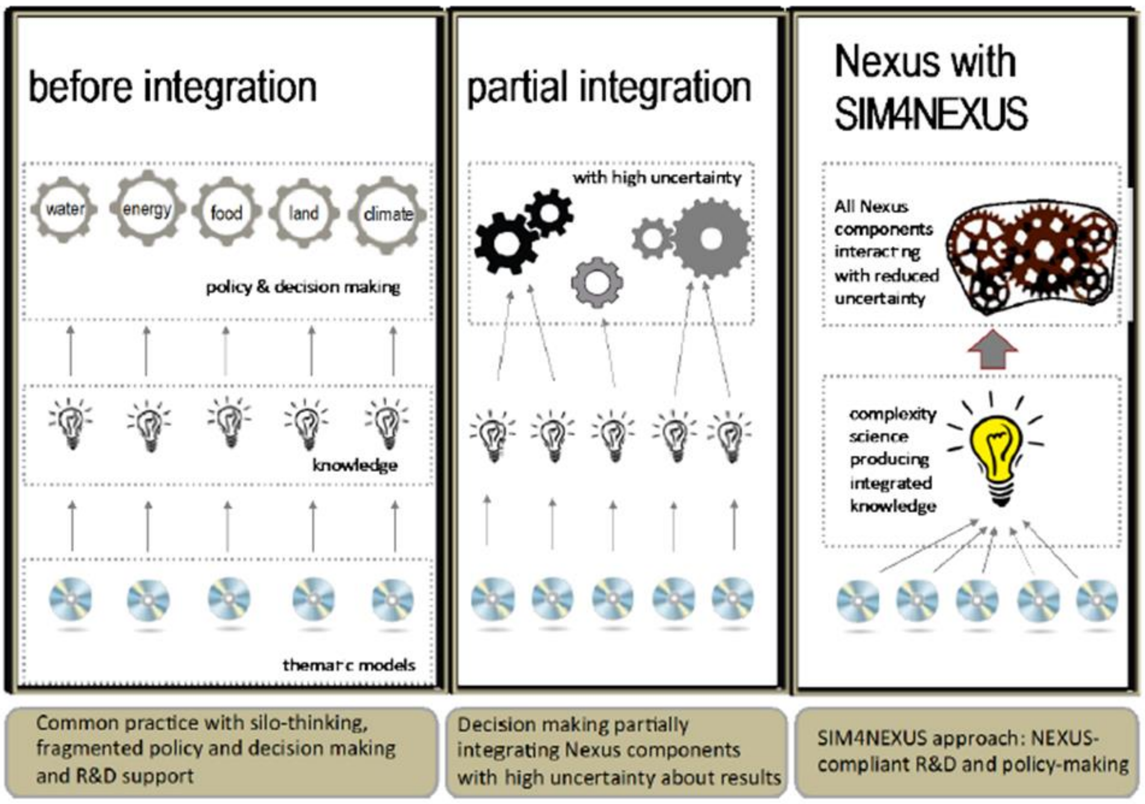
before integration
water
energy
food
land
climate
policy & decision making
knowledge
thematic models
partial integration
with high uncertainty
Nexus with SIM4NEXUS
All Nexus components interacting with reduced uncertainty
complexity science producing integrated knowledge
Common practice with silo-thinking, fragmented policy and decision making and R&D support
Decision making partially integrating Nexus components with high uncertainty about results
SIM4NEXUS approach: NEXUS-compliant R&D and policy-making

Figure 1: Current policy making process in Europe (before integration and partial integration) and what the project aims to achieve

2.1 Scope of the Game

The SIM4NEXUS serious game consists of many different components including the Knowledge Elicitation Engine (KEE), System Dynamic Models (SDMs) and underlying Thematic Models. The interaction among all these models is summarized in Figure 2: the SDMs are the active components simulating the bio-physical and socio-economic processes explored in the case studies. The KEE is used to collect and analyse data from the games, such as the policy choices that players with particular background and expertise make, the consequences for the nexus domains following from the simultaneous choices players have made, and the players' learning, expressed as the evolution of their scores in the game.

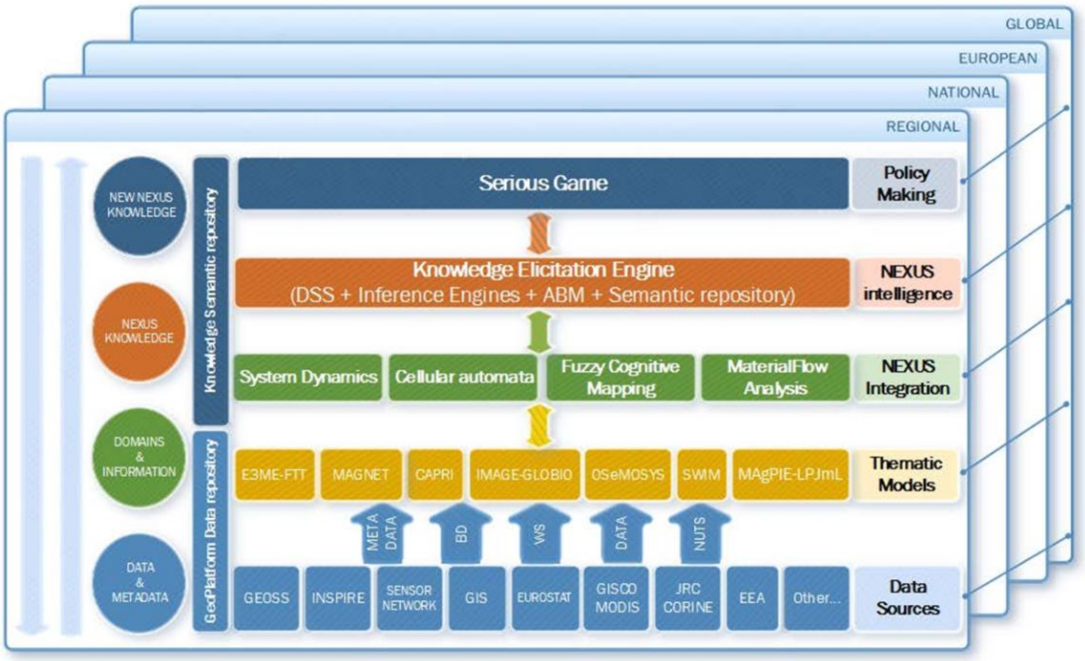

Figure 2: Interaction among all SIM4NEXUS models

The game will also be used in 12 case studies: the Global and European case studies and the national and regional studies depicted in (Figure 3). Each case study will be considered as a separate "level" in the game.

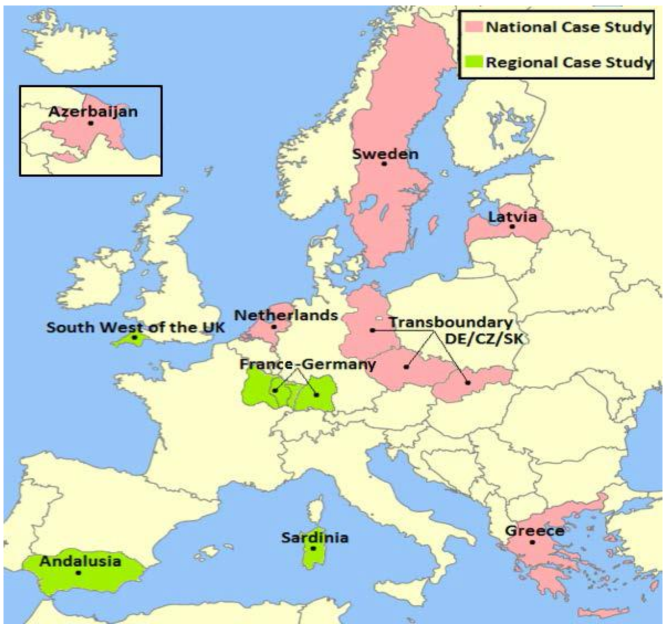

Figure 3: Map of all SIM4NEXUS case studies

The goal of the game is to learn about different policies on the nexus and how these policies impact a particular case study through a "learning by playing" approach. This approach is summarised in Figure 4.

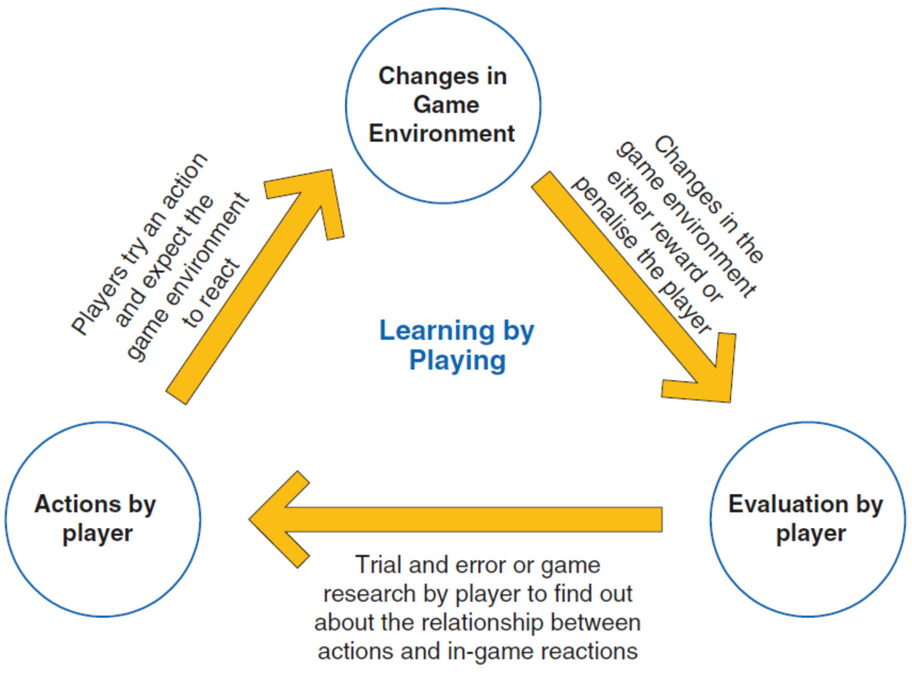
Changes in Game Environment
Changes in the game environment either reward or penalise the player
Evaluation by player
Trial and error or game research by player to find out about the relationship between actions and in-game reactions
Actions by player
Players try an action and expect the game environment to react
Learning by Playing

Figure 4: Learning by doing flowchart

Based on this concept the game play for the game is as follows (Table 1):

Table 1: Main SIM4NEXUS Serious Game concept

As a player, you represent policy makers in the various sectors in a particular area – food, energy, water, climate and land use. Your aim is to fulfil the targets (objectives) set out by the national or international bodies by changing or adapting new policies in your area. To succeed in the game, you must learn to fulfil these targets by mixing and matching various cross sector policies without compromising the existing status quo of the other sectors.

2.2 Content to be Included

This section summarizes the main content to be included in the Serious Game, both trough the interfaces and the Logics that the game contains, but also through the assistance of the Knowledge Elicitation Engine (KEE), that can provide information and further knowledge to stress the immersions of the game user in each case study. The identification and formalization of the content to be imparted is important for the Game Logics requirements definition, as the logics behind the game must guide the user through these contents in order to impart the knowledge to be imparted in each case study. Also, this identification is required for the construction of the KEE as well as the way both the Game and the KEE architecture will communicate the knowledge and information. This content is divided in 3 main parts:

1. **Core Experience** – What is the player experiencing as they play the game?

   The core experience in the game is to play the role of policy makers in food, energy, water, climate, land use. In the game, the player will typically start off with separate “silo-thinking” approaches towards decision making and policy implementation. Over the course of playing the game, they will be encouraged to change towards a more integrated NEXUS-compliant policy implementation approach and decision making.

2. **Base Mechanics** – What does the player actually do?

   The player will have a target at the start of each turn of the game and he/she will have to implement policies to try to achieve the target. The turn ends when the player has decided on the policies which are to be implemented to achieve the targets and clicked on "next turn" button. The game will compute the policies made and an analysis of the decisions will be displayed in the following turn, with a new target to achieve for the turn.

   The targets are envisioned to be displayed in a step-by-step manner to the player. This will help guide the player on what to do during the game play.

3. **Penalties and Reward (P&R) System** – What behavior within the game is encouraged or discouraged?

   Silo-thinking in decision making and policy implementation within the game is discouraged, whereas integrated NEXUS-compliant decision making is encouraged. For every target in each turn, the player is encouraged to look at policies in all sectors and consider them to achieve a target.

   The P&R system will be in 3 parts:

   i. Key indicators across all NEXUS components. These key indicators are yet to be defined and will require inputs from WP2, WP3, and WP5. It is noted that while it is not possible to have all key indicators showing positive values all the time, the player will be rewarded when there are more indicators showing positive results than vice versa. At this moment of writing, there is also no consideration to weigh the indicators yet and the assumption is that all indicators will have the same weight. This may change as the project processes.

   ii. Events within the game. Events are news happening "on the ground" which adds a societal and cultural aspect to the game. These events will be narrated in the same tone as the shared socioeconomic pathways and will be triggered based on the decisions the player made in the game. There will also be uncertainties in event triggers to add more realism in the game, e.g. the occurrence of extreme events such as economic crisis or disaster events. There will be 3 category of events informational events which are neutral, negative events which will penalize the player by deducting points and positive events which will reward the player with bonus points.

   iii. Score. There will be a score for the player. This score will indicate how successful the player is applying NEXUS-compliant decision making in achieving the targets in the game. Every progression in the time step of the game will add to the score to encourage the player to continue, every policy implemented will add to this score and the events will add to the score.

2.3 Feedback across spatial scales

The feedback between spatial scales of policy making can facilitated by the KEE. The KEE collects data about players' policy choices and the resulting consequences for the diverse NEXUS components. This mechanism can be used to feed new targets, resulting from games at global or continental scale, into games at national or regional scale. For instance, some policy may result from playing a game on the European level. This policy may entail new targets on national and regional levels. By playing games with these new targets with national and regional policy makers, data can be collected about their reactions and the consequences for the NEXUS components. The KEE can then inform the policy makers on the European level. Figure 5 illustrates this feedback loop.

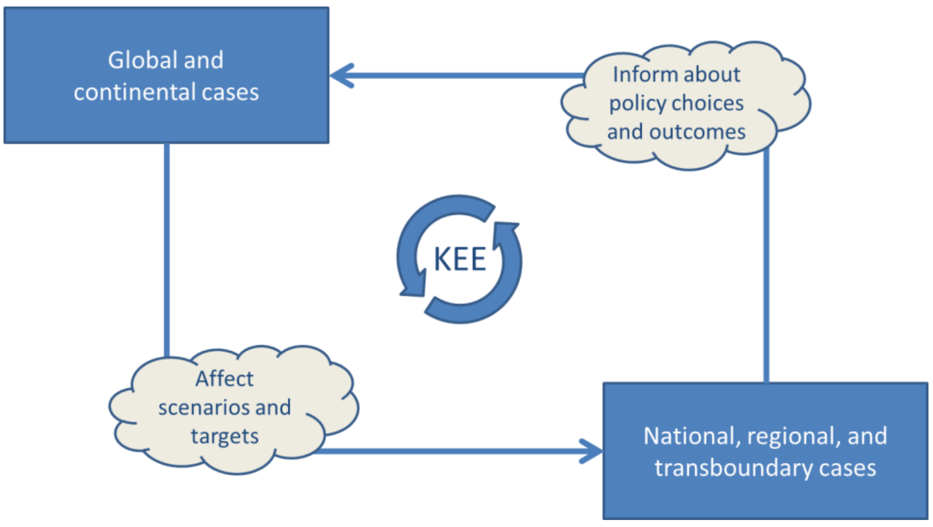
Global and continental cases
Inform about policy choices and outcomes
KEE
Affect scenarios and targets
National, regional, and transboundary cases

Figure 5: Feedback across spatial scales facilitated by the KEE

2.4 Game design flowchart

This section provides one of the main objectives of this Deliverable is to provide the definition of the Serious Game logic. This logic will be defined by means of flow charts, these flow charts have the unique capacity to show the steps to follow for the user inside the game and their order by connecting them with arrows. This diagrammatic representation illustrates a solution model to the logics behind the Serious Game. This section also provides how the user information will be structured in terms of the logical structure to be used in the Serious Game and the required information to assist the groups of user and classify the different actions in the KEE. The logical structure is defined by means of an entity-relationship diagram that has the capacity of describing inter-related things of interest in a specific domain of knowledge, in this case the game user. The entity-relationship model is composed of entity types (which classify the user information) and specifies relationships that can exist between instances of those entity types.

The description of the game design in terms of user (inter)actions has been divided in two related flowcharts: user registration and game play. The user registration flowchart (Figure 6) begins with a login/registration stage. In order to let the KEE provide correct answer, it is important that the characteristics of the player are known. This can be accomplished by requiring the player to login to the game, or - if the player is new -by requiring to register with the game in order to obtain the background (student, policymaker, general public, etc.) and area of expertise (Nexus component) of the prospective player.

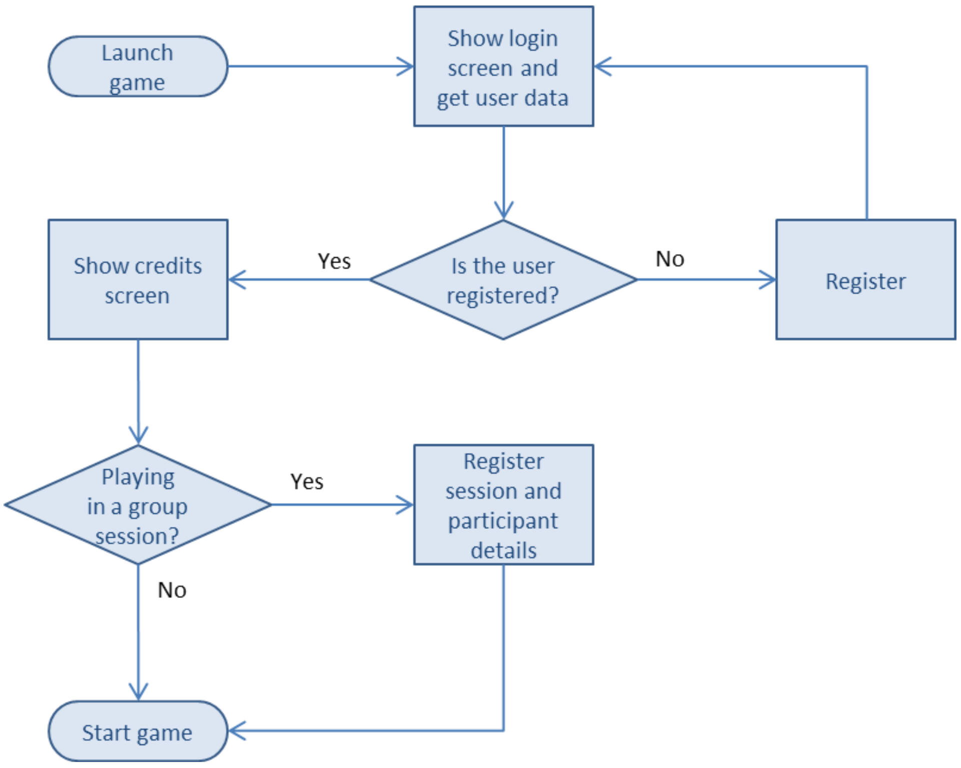
Launch game
Show login screen and get user data
Is the user registered?
Yes
No
Register
Show credits screen
Playing in a group session?
Yes
Register session and participant details
No
Start game

Figure 6: User registration flowchart

A first proposal of the information required in the Serious Game for the user is depicted in Figure 7

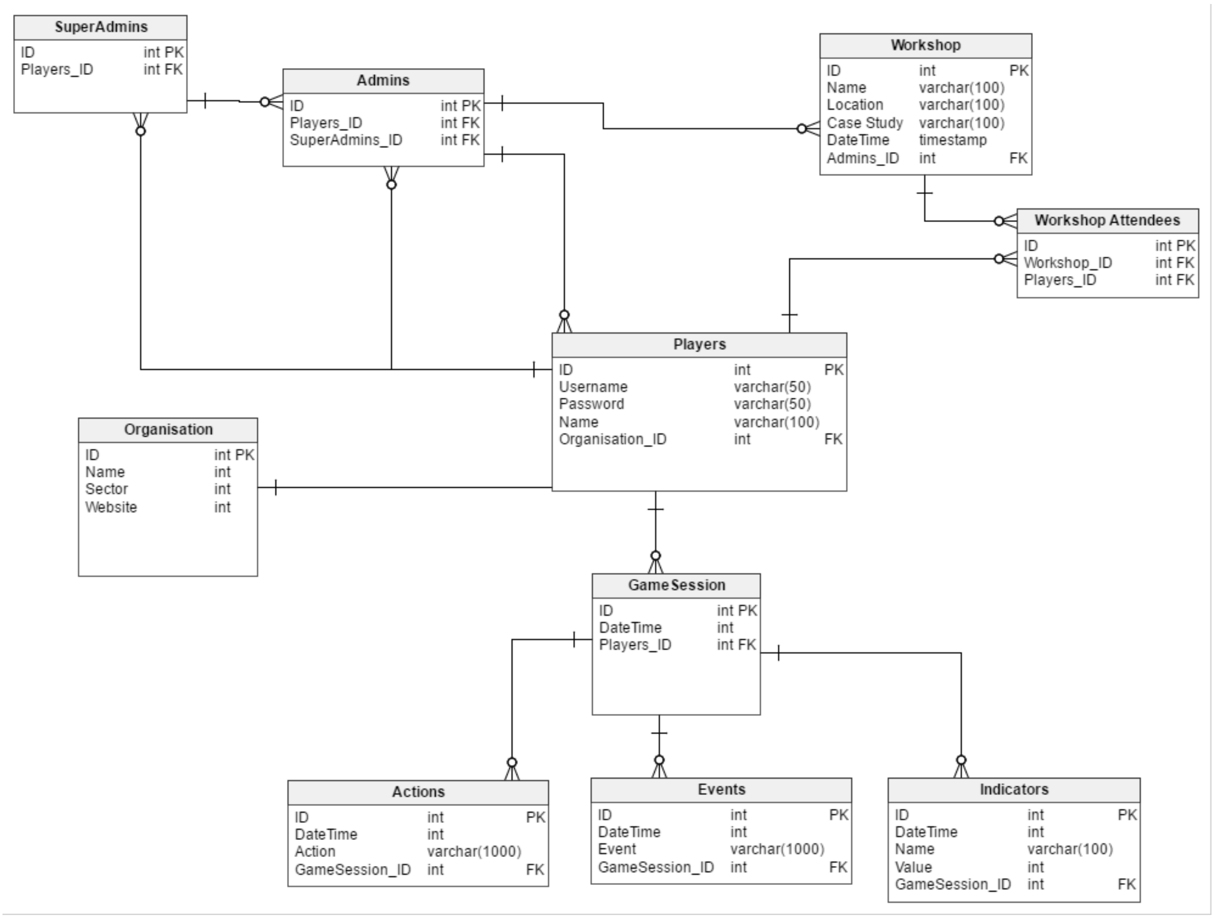

Figure 7: Entity relationship diagram of the user information used in the Serious Game

When a player has logged on the system opens the main menu. The game can be played by an individual player. Alternatively, it can be used in a group session moderated by a facilitator. During this stage, the setting of the game must also be defined since it is important for the KEE to know the context of the game play (e.g. does it concern for a workshop with policymakers, is it for education purposes with a group of students etc.). Therefore, the facilitator can register the participants in a session recording information related to:

- Group or singular playing
- Type of the participant
- Expertise level regarding the Nexus
- Level of cooperation across policy areas

Additionally there is more information about the game user that it is required to assist in the decision support system, as well as the other modules of the KEE to better assist and generate knowledge from the game development. This information, is related to the user, but not used during the game logics development. Therefore, the game should present a questionnaire where the information presented in Table 2 is included. It will be investigated how this information can be presented to be included in a non-intrusive manner.

Table 2: Questionnaire with user information

Question	Possible Answers
Age	14-99/Prefer not to say
Sex	Male/Female/Prefer not to say
Country	List of world countries/Prefer not to say
Education	List of education/Prefer not to say
Are you playing an individual user or in a group session?	Yes/No
When in a group session:	
What type of participants?	Students/policy makers/experts
Expertise level	Water: low ………… high Energy: low ………… high Land use: low ………… high Climate: low ………… high Agri/Food low ………… high
Level of cooperation across policy areas allowed in the group session:	low ………… high

After registration, the game is initialized. All the data collection methods used in the game will follow all the procedures that have been implemented within SIM4NEXUS project for data collection, storage, protection, retention and destruction and confirmation ensuring that data collection complies with national and EU legislation. The ethics issues involved in the SIM4NEXUS study concern general ethical issues of informed consent, anonymity and confidentiality associated with the voluntary involvement of human participants in the European Union. For more information on ethics and data management aspects, please refer to Deliverable 9.1 and Deliverable 4.2.

The ethics issue involved in the SIM4NEXUS project is in the area of the collection and use of personal data and the general ethical issues of informed consent, anonymity and confidentiality associated with the voluntary involvement of participants in SIM4NEXUS research activities in Europe.

The gameplay flowchart (Figure 8) depicts how a user can interact with the game and what actions are taken as response to user inputs. First, a user must select a case study scenario, i.e. a particular case study combined with a predefined set of input data such as a climate scenario. The number of time steps (turns) and the indicator target values for individual players are fixed, but when playing in a group setting, the facilitator may adjust the indicator target values.

After selecting a case study scenario, a user is presented with a narrative explaining the game and the case study, can change policy options, and indicate when (s)he is ready for the next turn. When the user is ready for the next turn, the simulation engine will run the system dynamics model (SDM) for the next time step, and report the new indicator values to the to the users. Changes in policy options and simulated outcomes are recorded by the KEE. Subsequently the graphical displays in the gameplay screen are updated and the user(s) can take a new turn, as long as the game is not completed. When the game is completed, i.e. when all predefined time steps have been passed, the final evaluation is performed and the score is reported to the user and reported to the KEE.

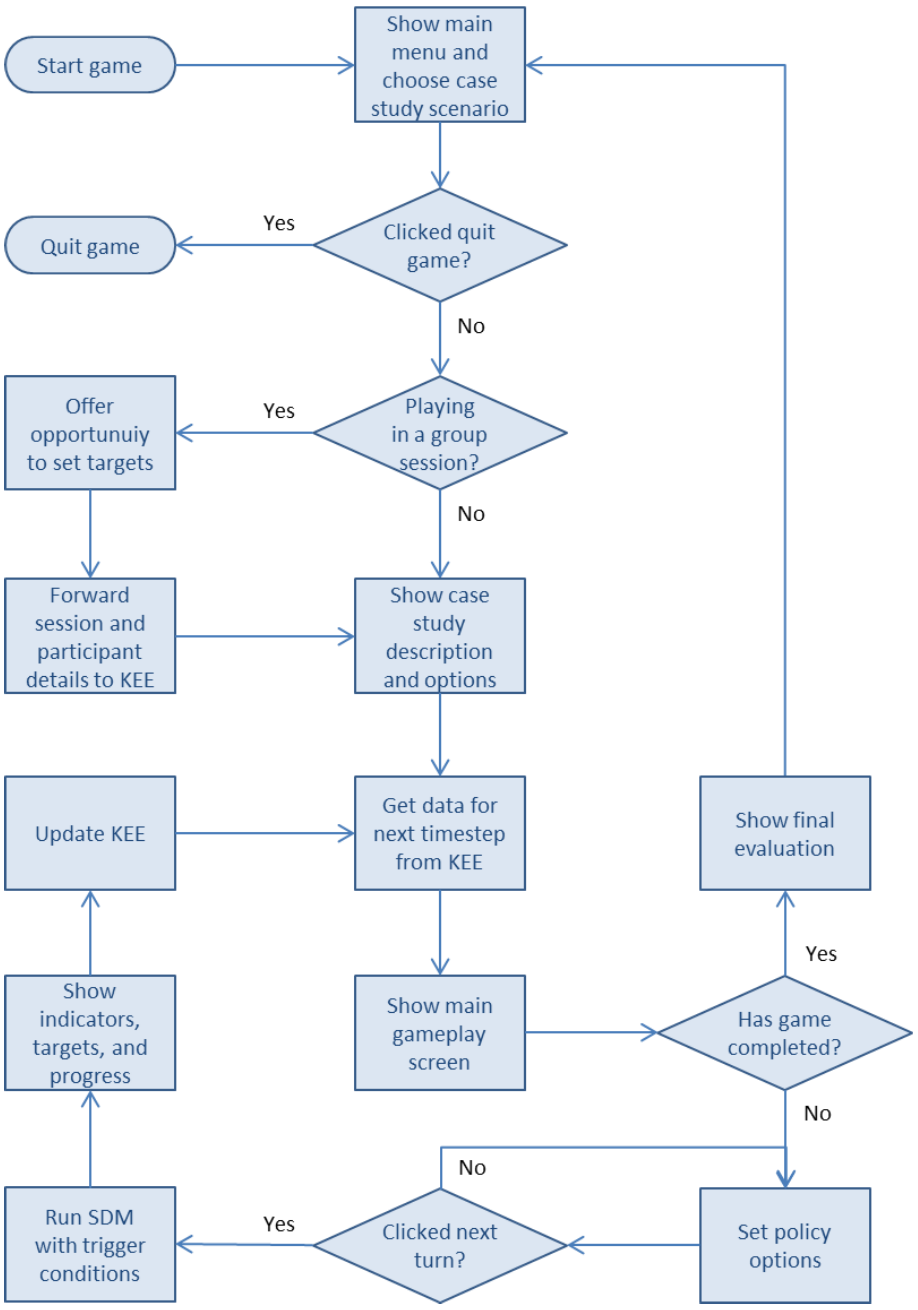
Start game
Show main menu and choose case study scenario
Quit game
Yes
Clicked quit game?
No
Offer opportunuiy to set targets
Yes
Playing in a group session?
No
Forward session and participant details to KEE
Show case study description and options
Update KEE
Get data for next timestep from KEE
Show final evaluation
Yes
Show indicators, targets, and progress
Show main gameplay screen
Has game completed?
No
No
Run SDM with trigger conditions
Yes
Clicked next turn?
Set policy options

Figure 8: Gameplay flowchart

2.5 Player's performance assessment

At the end of each game, a score is computed that indicates how close the policy targets are met by the player. The score is based on the indicator values resulting from the player's policy choices. SIM4NEXUS deliverable D4.1 contains an inventory of relevant indicators for each case study (Table 17 in the 30 November 2016 version of D4.1). A first approach to include final score of a game session based on quantifiable results, is the weighted sum of squared differences between target values and actual indicator values at the end of the game. During the detailed specification of the game instantiations for each case study, the weight factors for each instantiation can be determined (some weight factors may actually equal zero, when some indicators are irrelevant to evaluate a player's performance).

The score can be computed as follows:

$$S = 1 - \sum_i w_i \frac{(\max\{0, m_i(g_i - x_i)\})^2}{(g_i - x_{0i})^2}$$

Where:

- the score $\boldsymbol{S}$ is based on a weighted sum of squared, normalized, differences between indicators $\boldsymbol{x_i}$ and targets $\boldsymbol{g_i}$;
- the $\boldsymbol{x_{0i}}$ stand for the starting value of the indicator at the beginning of the game;
- for indicators that must be maximized $\boldsymbol{m_i = 1}$ and for indicators that must be minimized $\boldsymbol{m_i = -1}$;
- the sum of the weights $\sum_i \boldsymbol{w_i}$=1;
- $\boldsymbol{S = 1}$ is the maximal score to be attained (full compliance with all targets).

Such a score can be computed over all indicators and per policy area, making clear on which areas to focus in order to improve the general score. Thus it can serve as a basis to advice users and explain opportunities to improve their performance in nexus management.

At this point, this is a first approach, and during the development of the game it can be improved and modified to include events and non-quantifiable results with the aim to have a better player's performance assessment.

2.6 Possible use cases

This section describes how the game can be used in different organizational settings. First it describes the use of the game by a single player, controlling all policy options. Then it describes the options for playing the game in sessions led by a trainer or group facilitator, where participants play roles of policy makers in particular nexus domains. The session is concluded by a description of the options to play the game with artificial agents. In the latter case, users take the roles of particular policy makers while other roles are fulfilled by artificial agents, based on data collected by the knowledge elicitation engine.

- Single player controlling all policy options

  In the game setup, the user has opened a new game session and initialized a predefined scenario. The user has not selected a role as policy maker in a particular nexus domain. In the main gameplay screen the user is presented with policy options enabled for all nexus domains. Thus, while playing subsequent turns, the user can interfere with policies in all domains and attempt to achieve a balanced set of targets across the entire nexus. The evaluation is expressed in a score applying weight factors for the different targets. This setting for playing the game is particularly suitable for education and training to offer insight into relations across the entire nexus.

- **Multiple players, each taking policy makers' roles on particular nexus domains**

Playing games with groups where participants take different roles, requires a group facilitator to set up the game. In the simplest case, the facilitator can ask the players for their policy decisions and enter these into a game set up as in the single player case described above.

As a future extension of the game, we envision a more advanced multi-player setting, where the facilitator starts a game (in a game facilitator role) and assigns the players roles as policy makers in a particular nexus domain. The game logic is then running on a central server, but players are presented with an individual main gameplay screen in which all policy options and indicators are visible, but only the policy options related to the selected role can be changed. Turns are synchronised on the central server and the next time step is not taken until all participants have made their policy choices.

- **Playing the game with artificial agents**

As another future extension, we envision artificial agents participating in the game. During the games as described above, the KEE collects data on the policy choices the players make and the resulting outcomes in terms of indicators. Since users have supplied information on their background and expertise when registering for the game, the KEE can learn the behaviours of policy makers in particular domains. Combined with other knowledge, the collected data can be used to design artificial agents, playing the roles of policy makers. The results attained by human players can be used to configure agents on a scale ranging from focus on one particular nexus domain (neglecting the other domains) to full awareness of all domains. In this setting, users can select a role as policy maker in one of the nexus domains, and then play the game as in the multiplayer setting.

3 General System Requirements

This section contains the whole system requirements, taking into account both the Serious Game *per se* and the backbone architecture that will host the services needed to assess to the NEXUS concept.

The serious game consists of many different components. All these components will form the visualisation part of the serious game. The following breakdown is only meant for informative purposes.

- **Communications controller**: this component will connect to the KEE to retrieve data from KEE and to pass information from the user to the KEE
- **Terrain controller**: this component will display the elevation terrain from open source datasets, e.g. open street map, SRTM, google map etc.
- **Grid controller**: this component will display gridded data to the user
- **Events controller**: this component will manage the data obtained from the KEE and select what to display to the user depending on the input of the user
- **Graphical interface (GUI)**: this component will allow the user to interact with the game and displays the relevant information to the user

The communications controller will be in charge of handling the communication requirements of the game and transforming them to queries to the KEE. In the KEE side, all queries will be processed through a standardized API that will provide all the information needed in the Serious Game side. The KEE is formed by the following modules:

- **System Dynamic Models (SDM):** it will be based in an R simulation module that will perform simulations for each Study Case, taking as an input the current status of the game and the policies to execute. The results will drive the game to a new state in the future
- **Semantic Repository:** will provide of data the game, regarding the Study Case, but also regarding the player and other different parameterizations within the game
- **Decision Support System (DSS):** it will provide analysis and recommendations, to both the players in the game and domain experts. It is the main part in the KEE that will help to better understand the linkages in the Nexus
- **Agent Based Module (ABM):** it will perform tasks of adversary movements in the game

Figure 9 depicts the Serious Game interaction flowchart among modules. The rest of the section will specify the requirements of both the Serious Game and the rest of the system including the KEE and its API.

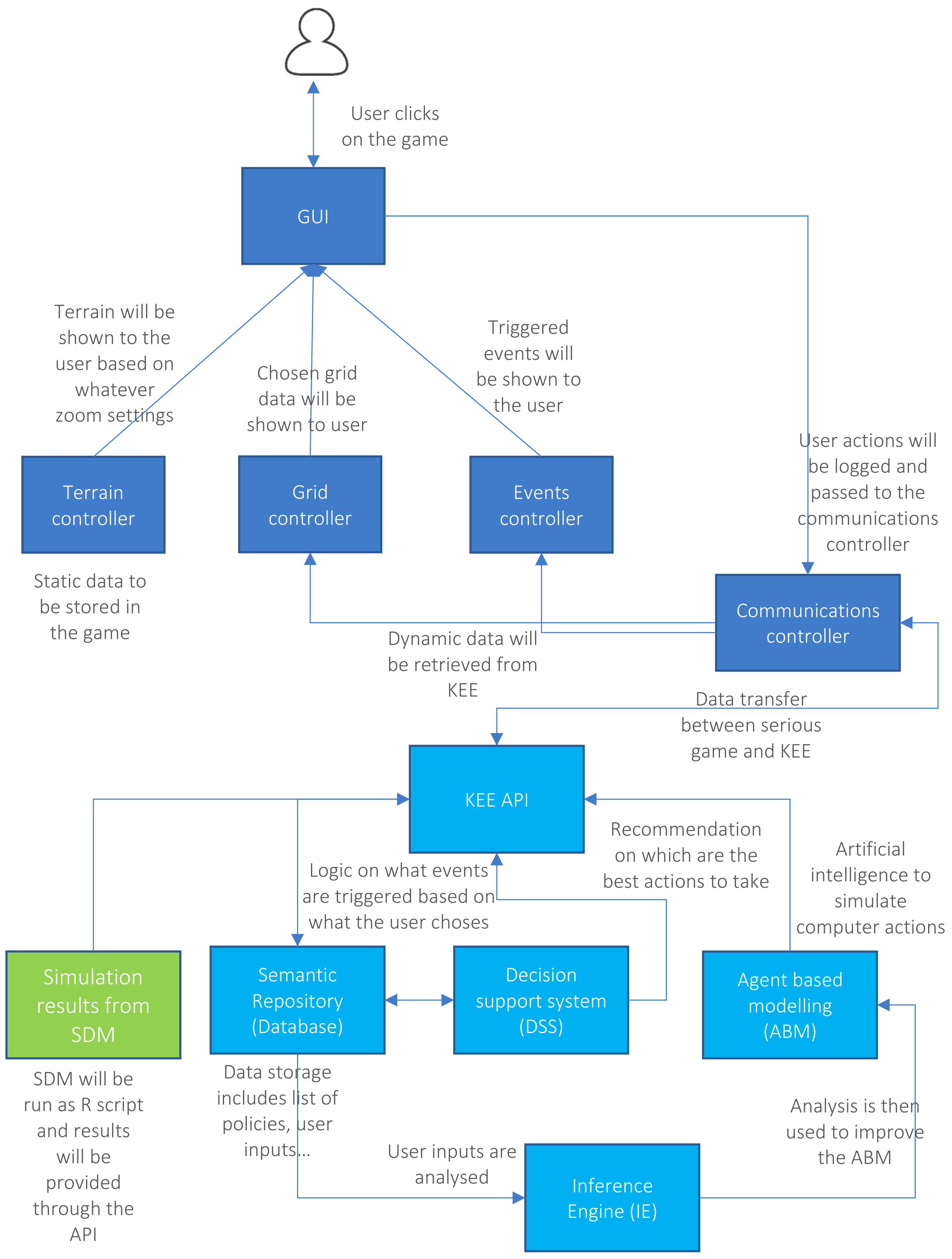
User clicks on the game
GUI
Terrain will be shown to the user based on whatever zoom settings
Chosen grid data will be shown to user
Triggered events will be shown to the user
User actions will be logged and passed to the communications controller
Terrain controller
Grid controller
Events controller
Static data to be stored in the game
Communications controller
Dynamic data will be retrieved from KEE
Data transfer between serious game and KEE
KEE API
Recommendation on which are the best actions to take
Artificial intelligence to simulate computer actions
Logic on what events are triggered based on what the user choses
Simulation results from SDM
Semantic Repository (Database)
Decision support system (DSS)
Agent based modelling (ABM)
SDM will be run as R script and results will be provided through the API
Data storage includes list of policies, user inputs…
User inputs are analysed
Inference Engine (IE)
Analysis is then used to improve the ABM

Figure 9: General system flowchart

3.1 Serious Game Requirements

3.1.1 Visual Style

Terrain Style

The proposed visual style is a 3D space with realistic terrain. Visualisation of gridded data and events are not yet finalised. The 3D realistic terrain mock up is as shown below (Figure 10).

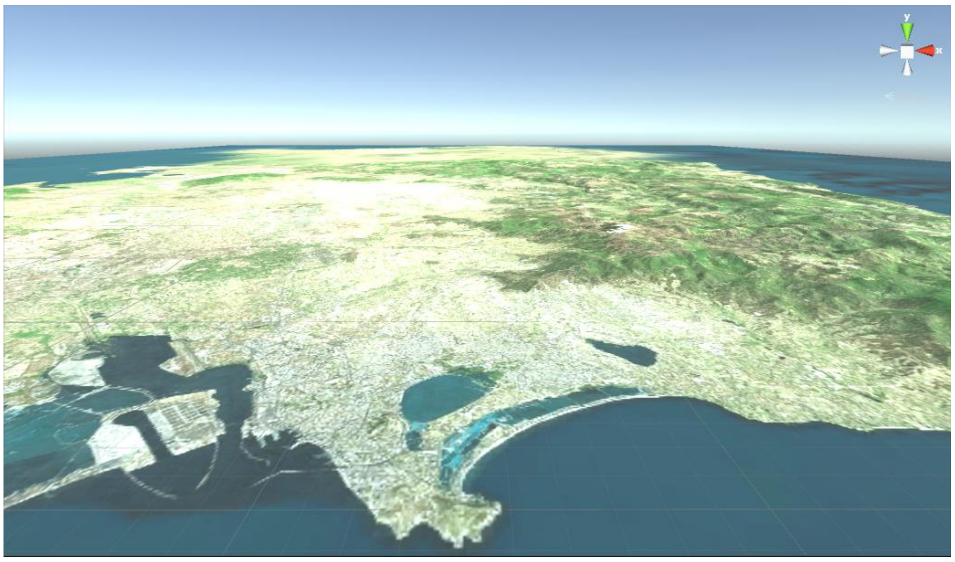

Figure 10: 3D realistic terrain mock up

Graphical User Interface (GUI) Mock-ups

The graphical user interface will be made to fit the target audience, it should be easy to navigate, provides clear call to action and interactive buttons and be able to display all the important information from the KEE to the end user. It should also has a more serious feel to reflect the science behind the game.

This section does not show the actual GUI style and design but only shows the mock-ups which forms the key elements of the game.

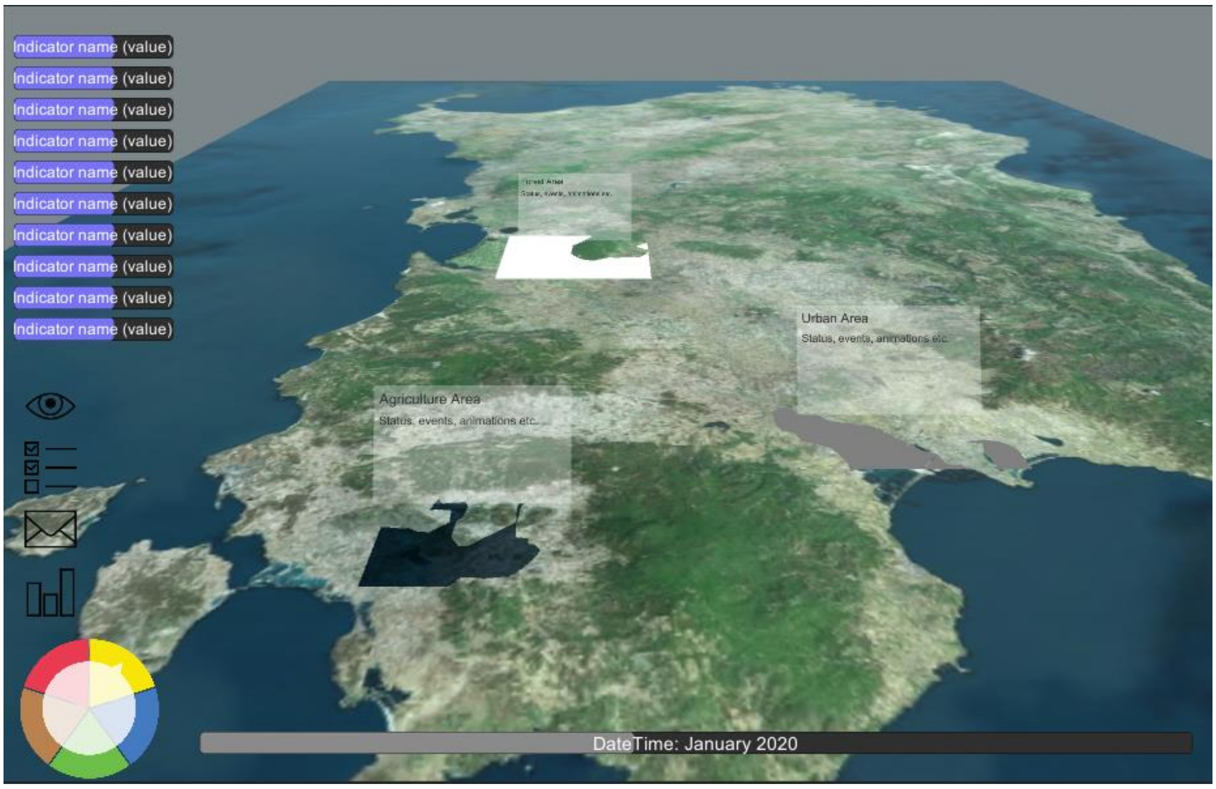

Figure 11 Mock-up of the game HUD and terrain

It is envisioned that the game consists of a 3D terrain of the case study area with key areas of interest, shown as shapes on the 3D map in the above figure. These areas of interest reflect the possible land use of sectors relevant to the case study, for example, it can consist of urban area, agriculture area, forest area and so on. The areas of interest will also contain events, animations and status to visually show the impacts of policies made in the game which may not may not be quantifiable.

The head up display (HUD) of the game consists of several elements – indicator area, additional options area, policy selection area as well as time. The indicator area is shown in the top left hand corner of the above image and will show the key indicators for a specific sector (e.g. climate). The user can change the indicators by turning the dial at the bottom left hand corner of the HUD. The additional options area is shown in the mockup as 4 separate icons. The first icon (eye) will show the overview of all the policies which are implemented and how nexus compliant they are. The second icon (checklist) will display the objectives of the case study and whether or not the user is achieving them.

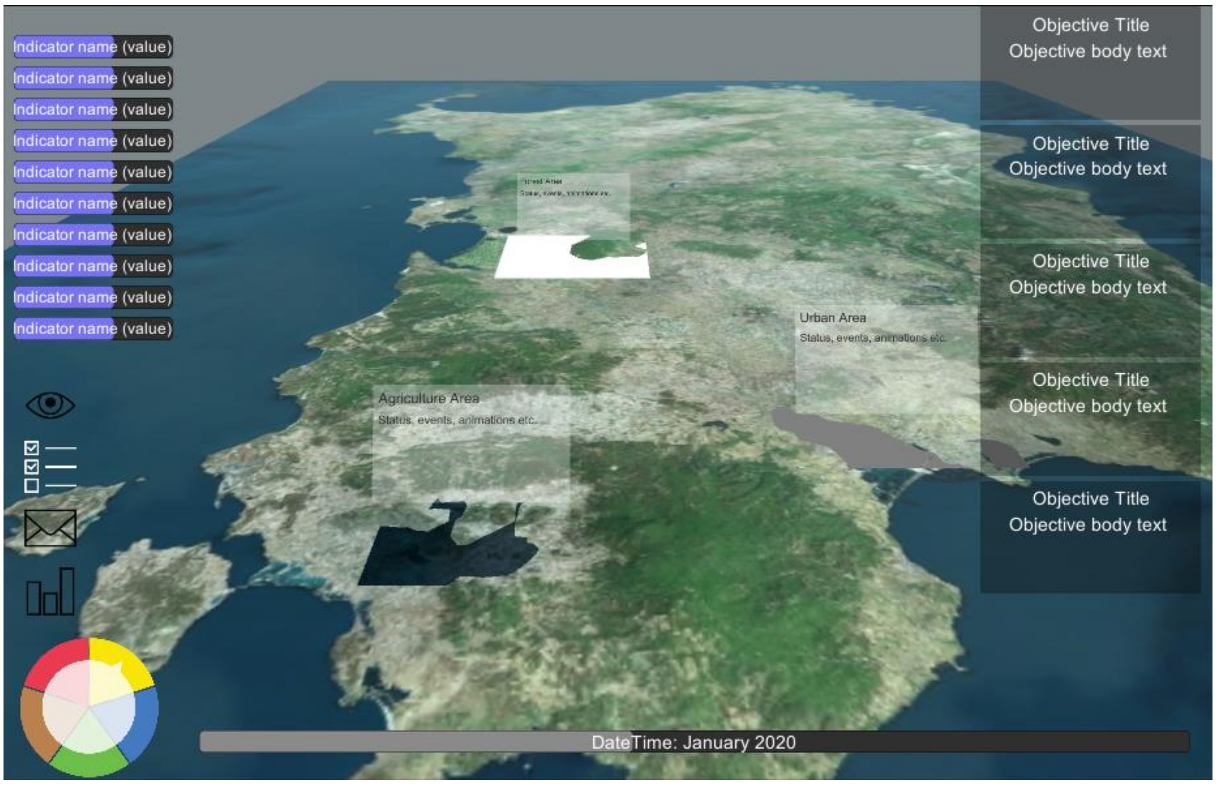

Figure 12 Mock-up of the HUD with the objectives of the game being shown

The third icon (envelope) will show all the events in the game which are triggered by the policies implemented. The fourth icon (bar chart) will display the historical key indicator values over the time which the user has played the game.

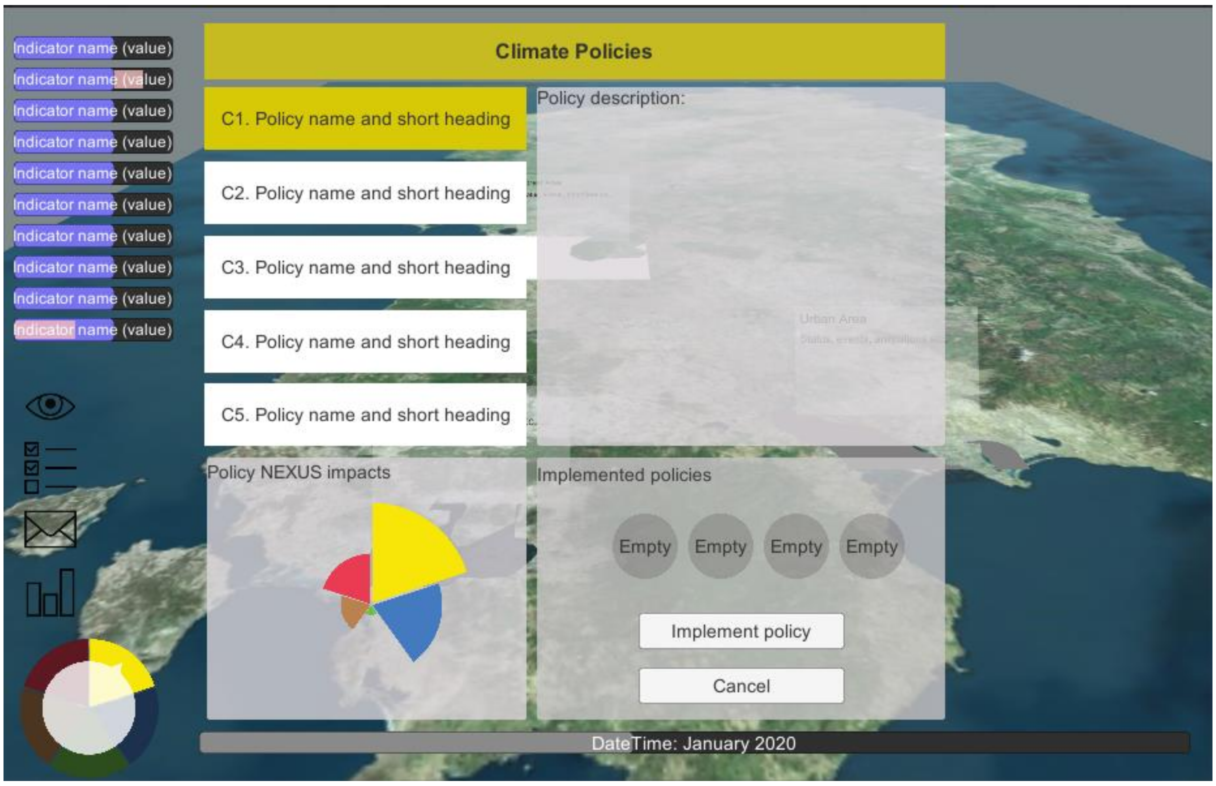

Figure 13 Mock-up of the game with the climate policy options being activated, showing the selected policy, the affected key indicators of that policy and the potential impacts of the other NEXUS area.

The policy selection area which consists of a colored dial as shown in the mockup will allow the user to select and choose which policies to implement. The dial has two layers, the first is the dial, by rotating the dial, and the user would be able to change the key indicators shown. By clicking on the colored circular sector, the user would be able to choose the policy to be implemented. Each colored circular sector represents a specific sector, for example, yellow represents climate policies. While the policy options menu is active, the user would be able to select different policies to implement. The indicator area will also highlight the affected indicators based on the selected policy display. It could for example show an increase in one particular indicator but a decrease in another. In the Policy NEXUS impacts area, the concept here is to show what the impacts on the other NEXUS area, a bigger impact in one particular NEXUS sector will show a part of the circle. The user will also be able to view other impacted indicators by changing the dial.

The time area, will show the current timestamp of the game and will also indicate when the game will end.

3.1.2 Game development platform requirements

The Serious Game will be developed using Unity 3D[1] to provide a web-based tool playable form many devices. Among Unity's main features there are its unique cross-platform capabilities which make Unity one of the most used platforms PC, consoles, mobile devices and websites game development.

To provide a web-based tool Unity makes use of WebGL[2], which is a cross-platform, royalty-free web standard for a low-level 3D graphics API based on OpenGL[3] ES 2.0, exposed through the HTML5[4] Canvas element as Document Object Model interfaces.

WebGL allows Unity to publish content as JavaScript[5] programs which use HTML5 technologies and the WebGL rendering API to run Unity content in a web browser. Unity WebGL supports all major desktop browsers to some degree. However, the level of support and the expected performance varies between different browsers. Table 3 provides an overview of browser features of interest to Unity WebGL content, and which browsers support them.

Table 3: Unity 3D desktop browser compatibility table

	Mozilla Firefox 42	Google Chrome 46	Apple Safari 9.0	MS Internet Explorer 11	MS Edge 13
WebGL support	Yes. GPU blacklists apply. WebGL may be unsupported for specific older graphics cards.	Yes. GPU blacklists apply. WebGL may be unsupported for specific older graphics cards.	Yes. Safari 8 and higher	Yes. IE 11 and higher	Yes
Web audio[6]	Yes	Yes	Yes	No	Yes

[1] https://unity3d.com/

[2] https://www.khronos.org/webgl/

[3] https://www.opengl.org/

[4] https://www.w3.org/TR/html5/

[5] https://www.javascript.com/

[6] The Web Audio API is required to play sound in Unity WebGL content.

	Mozilla Firefox 42	Google Chrome 46	Apple Safari 9.0	MS Internet Explorer 11	MS Edge 13
Full-screen support	Yes	Yes	No. Safari supports the HTML5 full-screen API, but no keyboard input when in full-screen mode, so unity will disable full-screen functionality when running in safari.	Yes	Yes
Cursor locking support	Yes	Yes	Yes. Firefox up to version 42 and safari will not support indexedDB for content running in an iframe. Firefox 43 and higher will fix this.	Yes	Yes
Websockets	Yes	Yes	Yes	Yes	Yes
Webrtc	Yes	Yes	No	No	Yes
WebGL 2.0	No. Firefox supports webGL 2.0, but it is disabled by default and needs to be enabled in about:config.	No	No	No	No. Chrome supports webGL 2.0, but it is disabled by default and needs to be enabled in chrome://flags.
asm.js aot compilation[7]	Yes	No	No	No	Yes

[7]asm.js is a subset of JavaScript for which a browser can specifically optimize. Browsers which implement asm.js support may be able to run Unity WebGL content faster, because Unity uses asm.js

3.2 Knowledge Elicitation Engine (KEE)

Following Figure 9 chart, the KEE modular structure can be sketched, as it will contain the following elements:

- **System Dynamic Models (SDM):** it will be based in an R simulation module that will perform simulations for each Study Case, taking as an input the current status of the game and the policies to execute. The results will drive the game to a new state in the future
- **Semantic Repository:** will provide of standardized data the game, regarding the Study Case, but also regarding the player and other different parameterizations within the game
- **Decision Support System (DSS):** it will provide analysis and recommendations, to both the players in the game and domain experts. It is the main part in the KEE that will help to better understand the linkages in the Nexus
- **Agent Based Module (ABM):** it will perform tasks of adversary movements in the game

From the point of view of the hardware requirements, the modules will be implemented in a cloud based infrastructure that contains all the environment required for the development of the different modules.

Then, another important issue to address are the communication capabilities that the KEE needs. It will be based in an Internet available communication stack implemented with the stat of the art standards and protocols to ensure Interoperability with the visual and logical of the game, but also with the rest of SIM4NEXUS modules. Figure 14 shows this modular structure and Sections 3.2.1 and 3.2.2 detail the rest of KEE requirements regarding its architecture and its API.

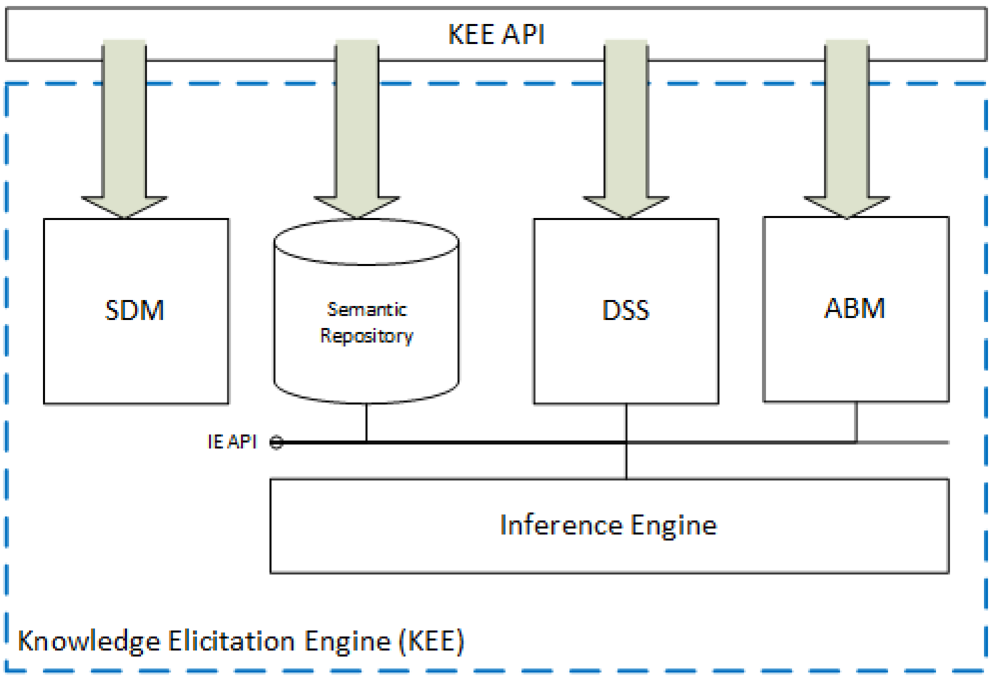

Figure 14: KEE modular structure

3.2.1 Architecture Development

The Knowledge Elicitation Engine (KEE) requires hardware and software infrastructure to operate properly. It has been determined that provisioning will take place in two stages:

- The development stage
- The production stage

Each stage has different needs, especially in terms of hardware, with the development stage being the precursor to the production one. Since functional requirements at the time of writing are still not fully determined, neither can the final production stage hardware requirements be precisely set.

However, partners currently need to perform local development and testing, before deploying their deliverable software modules on the production server, and they need to do so on an environment that will match, as closely as possible, that of the production stage. For that reason, the following work-flow has been suggested:

Virtual Machine (VM) will be provided to all the technical partners, based on Oracle's VirtualBox[8], a free and open-source hypervisor for x86/x64 computers, which can be installed on a number of host operating systems. The VM will have a software stack installed that will try to match, as closely as possible, that of the production server. Of course, as the project evolves and new software requirements may arise, that stack will evolve as well, and new versions of the VM will be rolled out.

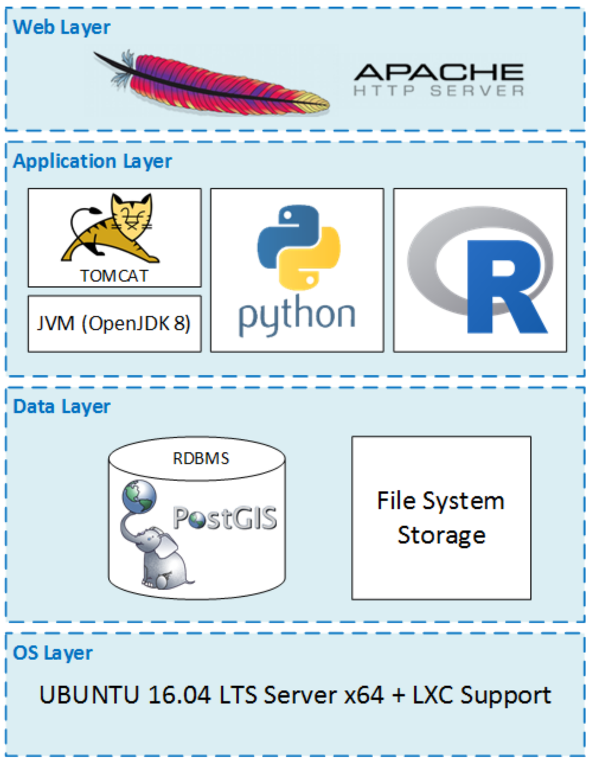

Figure 15: Software stack proposed for the production stage

[8] https://www.virtualbox.org/

In addition, as the project's software code-base expands, integration and version control systems (such a Jenkins[9] and Git[10]) will be installed and deployed in the VM, facilitating new software deployment.

In Figure 15, an illustration of the proposed software stack for the VM (and, eventually, for the production server) is available.

There are four distinct software layers discernible:

OS Layer

The operating system of choice will be the latest Ubuntu Server 16.04.1 LTS Edition. Ubuntu[11] Server is a part of the larger set of Ubuntu products and operating system developed by Canonical Ltd. Ubuntu server is a specific addition that differs a little bit from Ubuntu desktop, in order to facilitate installation on servers.

The deployment of the final production server will leverage the LinuX Container (LXC) technology at a base server of our choosing. The deployment will eventually be achieved using three different virtual containers (namely: web, app and DB). At the moment of writing, and for the VirtualBox VM, no LXCs will be deployed, and all software will reside directly on the base OS.

The production server will be constantly updated and backed up using incremental backups, based on the Btrfs, an open-source Copy-on-Write enabled file system, offered as an integral part of Ubuntu.

Data Layer

The persistence/database layer is powered by the well-known open source relational database manager PostgreSQL[12] version 9.5. The DB layer contains also PostGIS[13], an extension to PostgreSQL that adds support for geographic objects and location queries.

Additionally, the file system itself may be used for arbitrary bit-stream (*e.g.* user uploaded files) storage. References to all stored files, along with any required meta-data, should also be kept in the relational database.

Application Layer

The application layer will feature various run-time environments, each one accommodating distinct software modules, as developed by the various partners. Namely, those environments will include:

[9] https://jenkins.io/

[10] https://git-scm.com/

[11] https://www.ubuntu.com/

[12] https://www.postgresql.org/

[13] http://www.postgis.net/

• Java Virtual Machine (JVM) & Apache Tomcat

A Java[14] virtual machine (JVM) is an abstract computing machine that enables a computer to run a Java program. The Java Runtime Environment (JRE) is a software package that contains what is required to run a Java program. It includes a Java Virtual Machine implementation together with an implementation of the Java Class Library.

- The main reasons for choosing the Java platform are:
- Cross-platform support
- High scalability
- Wide adoption
- Maturity
- Huge library code-base available
- Open source license
- Ease of installation on infrastructure
- Common code-base with Android

Both the development VM and the production server will run the OpenJDK runtime v8, an open-source implementation of the JVM specifications, considered to be fully compatible with Oracle's reference implementation. It is also part of the official Ubuntu repositories and, therefore, easily installed and maintained (i.e. updates are part of the OS update work-flow).

The Apache Tomcat[15] 8.x server, is an open source Java Servlet Container developed by the Apache Software Foundation (ASF). Tomcat implements several Java EE specifications including Java Servlet, Java Server Pages (JSP), Java EL, and WebSocket, and provides a "pure Java" HTTP web server environment in which Java code can run. It is also offered "natively" for Ubuntu, as part of the official software repositories, thus making its installation and maintenance an integral part of the operating system update work-flow.

• Python

Python[16] is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together.

The development VM and the production Server will feature Python v.3.5, the default Ubuntu repositories version.

[14] https://www.java.com/en/

[15] http://tomcat.apache.org/

[16] https://www.python.org/

- R

R[17] is an open-source language and environment for statistical computing and graphics. R provides a wide variety of statistical (e.g. linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering) and graphical techniques, and is highly extensible.

The development VM and the production server will both run R v3.3.2, as provided and maintained by the official Comprehensive R Archive Network (CRAN).

- Web Layer

All the software modules built in the context of the project, that will be accessible over the web, via HTTP/HTTPS, will be so using the Apache Web Server[18], an open source Web server creation, deployment and management software. It is designed to create Web servers that have the ability to host one or more HTTP-based websites. Notable features include the ability to support multiple programming languages, server side scripting, an authentication mechanism and database support. Apache Web Server can be enhanced by manipulating the code base or adding multiple extensions/add-ons.

The development VM and the production server will feature Apache v2.4, as maintained by the official Ubuntu repositories.

3.2.2 KEE API

The Knowledge Elicitation Engine will have to provide data coming from different sources and with different formats, Table 4 summarizes the main requirements.

Table 4: Communication requirements for KEE modules

	Interface	Data	Technical issues
Simulations From SDM	It has to be able to provide a remote processing interface	It will provide data about scenario evaluations	Thread safe environment The models will be written in R Results over features over a portion of land
Semantic Repository	It has to be able to provide an interface that recognises advanced queries about data	The information will be data related to a domain with semantic structure	High semantically structured responses

[17] https://www.r-project.org/

[18] https://httpd.apache.org/

	Interface	Data	Technical issues
Decision Support System	It has to be able to provide processing capabilities	It will reply with recommendations	Intense processing capabilities
Agent Based Modelling	It has to provide the actions simulating an intelligent adversary for the game	Actions to be applied in the user side of the game	High performance requirements High frequency calls
Inference Engine	heterogeneous interface providing advanced calculations for the rest of the modules	Heterogeneous structure	Intensive processing capabilities

The requirements for the communication are different for each module but all share the same necessities of exchanging knowledge in terms of sharing and reusing data, algorithms and procedures. It makes suitable to think that a feasible mechanism for the communication among modules are Web Services.

Web Services are wide used over the Web and provide some functionalities that the KEE can take advantage of. As explained in [1], the application of Web Services for web-based generalization processes would benefit from the established web-based data dissemination approaches, which are mostly implemented by Spatial Data Infrastructures (SDI). Hence Web Services enable on-demand and on the-fly generalization processes based on the most current data.

By the application of Web Services as mean of knowledge exchange, interoperability is achieved in terms of syntactic agreement partially thanks to the use of common standards. Most used Web Services standards are SOAP and REST, being SOAP the most used in industry. SOAP also has been used in many research projects such as [2] [3] [4]. These projects gave already good insights into the capabilities of Web Services, but did not reflect the geospatial issues due to the missing geospatial concepts within SOAP (e.g. feature encoding).

At the current moment there exist GIS Web Services that provide access to GIS data or functionalities over the internet in a standardized way. It has to be noted that a GIS Web Service is not an Internet mapping application, alternatively a GIS service can be consumed by, or integrated into, a web application. A GIS Web Service can be thought of as an Interface, by which an application accesses GIS data or functionality. GIS Web Services can provide geographic data, but they can also provide geoprocessing tasks, such as address matching, routing, or geocoding and always provided through standard internet protocols.

Among the advantages of using GIS web services one can find that: data does not need to be housed locally - can come from many sources, and maintained by the hosting entity; functionality is already provided, doesn't need to be built by the app developer; developers can use multiple services in their applications; GIS Web Services use standard formats regarding how they are accessed and what capabilities they have; and they are interoperability providers - can work across different platforms and applications and over networks.

One of the most used GIS Web Service suite is the one proposed by the OGC, which comprises some Web Services for different purposes:

- **WMS:** Web Map Service
- **WFS:** Web Feature Service
- **WCS:** Web Coverage Service
- **WPS:** Web Processing Service
- **WS Common:** Web Services Common

Taking into account the architecture described in Figure 14 and the requirements specified in Table 4, the KEE can benefit of incorporating two of these OGC services: WPS and WFS.

- **Web Processing Service (WPS): Is intended to be a standardized means of performing geoprocessing tasks over the Internet. It standardizes how inputs/outputs are described, how to request execution, how to handle output**

  The notion of the specification is to provide spatial processes through a standardized service interface over the Web based on a common transfer protocol, namely the Hypertext Transport Protocol (HTTP). The variety of spatial processes that can be described by the WPS is unlimited. A process description for each process is available through the WPS interface. Besides a title and an abstract the description Workshop of the ICA Commission on Map Generalisation and Multiple Representation – June 25th 2006 includes valid process parameters and their encoding. The client-service communication is based on the Extensible Markup Language (XML).

- **Web Feature Service (WFS): The Web Feature Service provides access and manipulation operations on geographic features using HTTP as the underlying protocol.** The WFS provides access to vector data and is therefore fundamentally different from a WMS which produces mere raster image representations of geospatial data as maps. A WFS can be cascaded; it can serve data that is located at some remote WFS. When transporting geospatial data, the interchange format is the Geography Markup Language (GML) and conforms to some GML application schema. The operations provided by the WFS are GetCapabilities, DescribeFeatureType, GetFeature, GetFeatureWithLock, GetGMLObject, LockFeature and Transaction.

- **Geography Markup Language (GML): GML is an XML encoding for the transport and storage of geographic information.** GML provides encodings for many concepts including features, geometry, coordinate reference systems, topology, time and metrics. GML is defined using XML Schema and, since GML is a complex standard, it is generally the case that a particular application only uses a subset of the GML Schema. In fact, it might be difficult to achieve interoperability in a community if the allowed GML elements and attributes are not restricted.

One of the most interesting features of OGC protocols is that there exist some implementations of the standards and services that are ready to use, benefiting the implementation of OGC standards in GIS applications. Table 5 contains some of the most notorious implementations available in the Internet.

Table 5: Implementations of WPS and WFS standards

Name	Licence	Service/Standard	Description
pywps	MIT	WPS	PyWPS is an implementation of the Web Processing Service standard from the Open Geospatial Consortium. PyWPS is written in Python. PyWPS was started by Jachym Cepicky as part of his project 'Connecting of GRASS GIS with UMN MapServer', supported by the German Foundation for Environment. He began to work on this project with a scholarship by GDF-Hannover that went from April to September of 2006.
52°North Web Processing Service	GPL-2.0	WPS	The 52°North Web Processing Service enables the deployment of geo-processes on the web in a standardized way. It features a pluggable architecture for processes and data encodings. The implementation is based on the current OpenGIS specification: 05-007r7. Its focus was the creation of an extensible framework to provide algorithms for generalization on the web.
WPS4R	N/A[19]	WPS	WPS4R is a solution for creating WPS processes based on annotated R-scripts. The code was developed in the FP7 projects UncertWeb and GeoViQua. The product website can be found at http://52north.org/wps4r.
OWSLib	BSD-4-Clause-UC	WPS/WFS	OWSLib is a Python package for client programming with Open Geospatial Consortium (OGC) web service (hence OWS) interface standards, and their related content models. OWSLib was buried down inside PCL, but has been brought out as a separate project in r481.

[19] Not assigned

Name	Licence	Service/Standard	Description
GeoServer	GPL	WFS	GeoServer is an open source server for sharing geospatial data. Designed for interoperability, it publishes data from any major spatial data source using open standards.

4 Conclusions

In this Deliverable the Game Logics has been defined through the development of two main flowcharts: Figure 6: User registration flowchart and Figure 8: Gameplay flowchart. The first one describes how the user will set up each game play. In this setting up process, an important point, regarding the collection of user data has been identified. User data is needed for the game logics to be developed, as, taking into account the objective of providing an immersive experience, the user is a central point in the game logics development. However, from the KEE point of view and in order to provide useful information, assess in regards to the NEXUS concept, and in general to do automatic learning from user interactions; the more information collected from the user, the better. A questionnaire has been developed in order to collect more information from Serious Game users, which will be included in the semantic repository and used by the KEE to generate further knowledge.

Additionally, in Chapter 3, the General Systems architecture has been proposed through a modular diagram. This diagram (shown in Figure 9) depicts the information flows and shows the interrelations between modules. As a summary, this diagram can be divided into two main parts: Serious Game and KEE. Regarding the Serious Game the main controllers have been identified (*i.e.* communication, events, grid and terrain) as well as the GUI, where some proposals that are under developments are shown. The communication controller will be the communicating point between the Serious Game and the KEE. Regarding the KEE part, a modular diagram of its components has been proposed in Figure 14: KEE modular structure. The Serious Game communication controller will make use of the KEE API, which by means of interoperable standards will provide sub modules communications capabilities adapted to the ones proposed. To this extent some OGC standards have been explored to investigate its suitability for the proposed use cases. With respect to the hardware and software requirements, Section 3.2.1 provides a stack development for the KEE architecture development. Moreover, this deliverable explains how the partners will work using VMs in a development environment and how the production environment will be constructed by means of highly available infrastructures. Software requirements are also provided in the attempt to safeguard a wide coverage for coming requirements during Serious Game development.

5 References

[1] T. Foerster and J. Stoter, "Establishing an OGC Web Processing Service for generalization processes," in *Workshop of the ICA Commission on Map Generalisation and Multiple Representation*, 2006.

[2] Neun and Burghardt, "Web Services for an Open Generalisation Research Platform," 2005.

[3] Neun., Burghardt and Weibel, "Spatial Structures as Generalization Support Services. In Proceedings of Workshop on Multiple Representation and Interoperability of Spatial Data," 2006.

[4] N. Regnauld, "Improving efficiency for developing automatic generalisation solutions," 2005.