Horizon 2020 Societal challenge 5
Climate action, environment, resource
Efficiency and raw materials

D4.6: Report on testing, security and scalability

LEAD AUTHOR: Marc Bonazountas & Pelagia Koutsantoni (EPSILON)

OTHER AUTHORS: Lluis Echeverria, Xavier Domingo, Marcel Ortiz, Mehdi Khoury

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
DELIVERABLE	D4.6: Report on testing, security and scalability
WP NAME/WP NUMBER	Serious Game development and testing / WP4
TASK	Tasks 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.7
VERSION	5.0
DISSEMINATION LEVEL	Public
DATE	30/06/2020
LEAD BENEFICIARY	EPSILON
RESPONSIBLE AUTHOR	Marc Bonazountas (EPSILON) & Pelagia Koutsantoni (EPSILON)
AUTHORS	Xavier Domingo (EURECAT), Lluís Echeverria Rovira (EURECAT), Marcel Ortiz (EURECAT), Mehdi Khoury (UNEXE), Ioannis Gitas (Auth), Stefanos Papaiodanidis (Auth)
REVIEWER	Lluís Echeverria Rovira (EURECAT)
COMMENTS	

DOCUMENT HISTORY

VERSION	INITIALS/NAME	DATE	COMMENTS-DESCRIPTION OF ACTIONS
1.0	LLUIS ECHEVERRIA	15/05/2020	TOC
1.1	MARCEL ORTIZ	16/05/2020	SECTIONS 3.1 AND 3.2
1.2	MEHDI KHOURY	16/05/2020	SECTIONS 3.4 AND 3.5
1.3	PELAGIA KOUTSANTONI	22/05/2020	HW RESOURCES AND USER FEEDBACK
1.4	LLUÍS ECHEVERRIA	23/05/2020	INTEGRATION CENTRE
1.5	MARCEL ORTIZ	25/05/2020	TESTING OF THE SYSTEM
2.0	PELAGIA KOUTSANTONI	26/05/2020	INTRODUCTION AND CONCLUSIONS
2.1	LLUIS ECHEVERRIA	27/06/2020	REVIEW
3.0	XAVIER DOMINGO	28/06/2020	REVIEW
3.1	PELAGIA KOUTSANTONI	30/06/2020	FINAL REVIEW
4.0	PELAGIA KOUTSANTONI	27/07/2020	SECTION 5
5.0	LLUÍS ECHEVERRIA	06/11/2020	COMMENTS REVIEW

5.1	PELAGIA KOUTSANTONI	26/11/2020	COMMENTS REVIEW

Addressing revision comments	
Comment	Response
The executive summary should be updated also including the results and conclusions from the section 5 on SIM4NEXUS SPACE, which has been included in last version of this deliverable. Also update sentence "The final section of this document summarizes the first users' feedback ….".	Executive summary is updated.
In the summary and section 5 an operational link to the SIM4NEXUS SPACE website should be included.	SIM4NEXUS SPACE link is now operational.
Editorial - corrections needed on Pg 67 "¡Error! No se encuentra el origen de la referencia.. » and « and to evaluate the out (?) produced for year 2019"	Corrected. Added link to Figure 23

Table of Contents

Figures

Tables

Executive summary

The development of the SIM4NEXUS Serious Game (SG) platform reached an important achievement, and is considered to be finished, thanks to the implementation and integration of its four main components through the S4N Integration Centre: i) the Graphical User Interface (GUI), ii) the Knowledge Elicitation Engine (KEE), iii) the SIM4NEXUS Database and iv) the System Dynamic Models Engine (SDM Engine). Each one of these elements has an essential role in the Game system and its crucial for the correct behaviour of the tool.

Five Case Studies have been integrated, Greece, Azerbaijan, Latvia, the Netherlands and the southwest of the UK, thus making possible the interaction with them through the GUI and the final test to validate the expected and correct behaviour of each one. In parallel, the Global Case Study has developed a demo tool which can be accessed through the S4N SG platform.

The present document describes, how the S4N Integration Centre is organized and deployed to coordinate the S4N platform and all the validation tasks implemented to ensure a high-quality system in terms of availability, capacity, interoperability, performance, reliability, robustness, safety, security, resilience and usability.

Addditionally, this document summarizes the first users' feedback collected during different training sessions in order to identify possible issues related to the S4N SG.

As proof, the latest version of the Serious Game GUI and the underlying connected KEE, S4N database and SDM Engine are available and free to play at this URL: https://seriousgame.sim4nexus.eu/.

The final section of this document presents the analysis procedures and the results of the SIM4NEXUS-Space task. SIM4NEXUS-SPACE aims to automatize the population of SDM through SDM data input collection using global satellite-based data inventories. The first results are also included and are very promising.

Changes with respect to the DoA

No changes.

Dissemination and uptake

This report is public, so it is accessible for everyone. However, the specific targeted audience of this report are the beneficiaries of the project.

Short Summary of results

The SIM4NEXUS Serious Game tool has been successfully implemented and is ready to be accessed by the players. The present report describes in a detailed way the development of the S4N Integration Centre, which coordinates the S4N platform, and all the validation tasks that have been implemented to ensure a high-quality system in terms of availability, capacity, interoperability, performance, reliability, robustness, safety, security, resilience and usability.

Evidence of accomplishment

The latest alpha version of the Serious Game is available and free to play at this URL: http://seriousgame.sim4nexus.eu

Glossary / Acronyms

Term	EXPLANATION / MEANING
API	Application Programming Interface
CS	Case Study
DSS	Decision Support System
GUI	Graphical User Interface
IC	Integration Centre
IE	Inference Engine
JWT	Json Web Token
KEE	Knowledge Elicitation Engine
ML	Machine Learning
OGC	Open Geospatial Consortium
RFC	Request For Comments
RL	Reinforcement Learning
S4N	Sim4nexus
SDM	System Dynamic Model
SOA	Service-Oriented Architecture
SR	Serious Game
SR	Semantic Repository
UI	User Interface
UK	United Kingdom
URL	Uniform Resource Locator
WP	Work Package
XML	Extensible Mark-Up Language
ET	Evapotranspiration
USGS	United States Geological Survey
CLC	CORINE Land Cover
MSI	Multi Spectral Instrument
OLI	Operational Land Imager

RF	Random Forests
SVM	Support Vector Machines
NDWI	Normalized Difference Water Index
BSI	Bare Soil Index
EVI	Enhanced Vegetation Index
MCARI	Modified Chlorophyll Absorption In Reflectance Index

1. Introduction

The SIM4NEXUS Serious Game platform has been developed and six Case Studies have been integrated, five as a Serious Games (Greece, Azerbaijan, Latvia, United Kingdoms and the Netherlands) and a last one, the Global Case Study, as a demonstration tool.

During the project, the different modules that compose the platform have been designed, developed, deployed and tested through the Integration Centre (IC). It acts as an agile monitoring and management tool to provide and ensure the correct quality and availability of the different S4N SG software modules.

The design and development of the S4N SG have been documented in Deliverable 4.5 and will not be covered in the present text but, probably, some of the developments and modules will be referenced.

The deployment has been conducted through Docker[1] technology, which simplifies and facilitates these kinds of procedures whilst allowing isolation and independence of the base Operative Systems and the underlying Hardware.

In parallel, throughout the progress of the development of the project, the different implementations have been specifically tested and validated with the aim of finally delivering an excellent outcomes thanks to the achievement of a high QoS in data pipelines and processing services between S4N clients and the cloud components, and seamless communication between all software components in integration activities.

The S4N SG has been presented and experienced by different final users during diverse training sessions and their feedback have been collected and analyzed with the intention of identifying possible improvements and/or problems (as an extension of the testing tasks) and to know which are their thoughts after being played the Game.

1.1 Structure of the document

The document is organized as follows:

- Section 1 is the introductory chapter, which provides the scope of the deliverable.
- Section 2 describes the architecture of the S4N Integration Centre.

[1] https://www.docker.com/

- Section 3 addresses all the validation tests performed to ensure the availability, capacity, interoperability, performance, reliability, robustness, safety, security, resilience and usability of the S4N SG platform.
- Section 4 contains the user feedback collected in different training sessions in terms of S4N SG acceptability.
- Section 5 outlines the SIM4NEXUS-Space approach developed for the use of satellite-derived input data for SIM4NEXUS.

2. The Integration Centre

The Integration Centre (IC) involves a set of technologies and procedures which have been specifically integrated to offer a simple but powerful tool to speed up the needed processes to deploy and maintain the S4N SG platform. The platform is composed of four main components (Figure 1) which are described in detail in D4.4 and D4.5:

- **GUI**: the visual part of the tool which aims to create a realistic virtual environment where the players can interact with the proposed Case Studies and learn about the complex connections between the nexus elements and the impact of applying different policies.
- **KEE**: the core of the SG. It acts as a central connector between the other SG components and implements all the Game logic based on the outputs from other WPs such as the SDMs (WP3) or the policy definitions (WP2).
- **S4N Databases**: All SIM4NEXUS data, either generated during the project, such as the Learning Goals (T4.1) or the policies (WP2), or by the players during the execution of the Game, are stored in the SIM4NEXUS database. It is divided into two components: i) the Semantic Repository (SR) (D4.4) and a relational database. Depending on the source, type and utility of the data, it will be stored in one of these two databases.
- **SDM Engine**: a specific key interface which has two main functionalities. First, it is in charge of integrating the SDMs (provided by WP3) to the KEE and, second, it manages their execution to simulate the different Game turns.

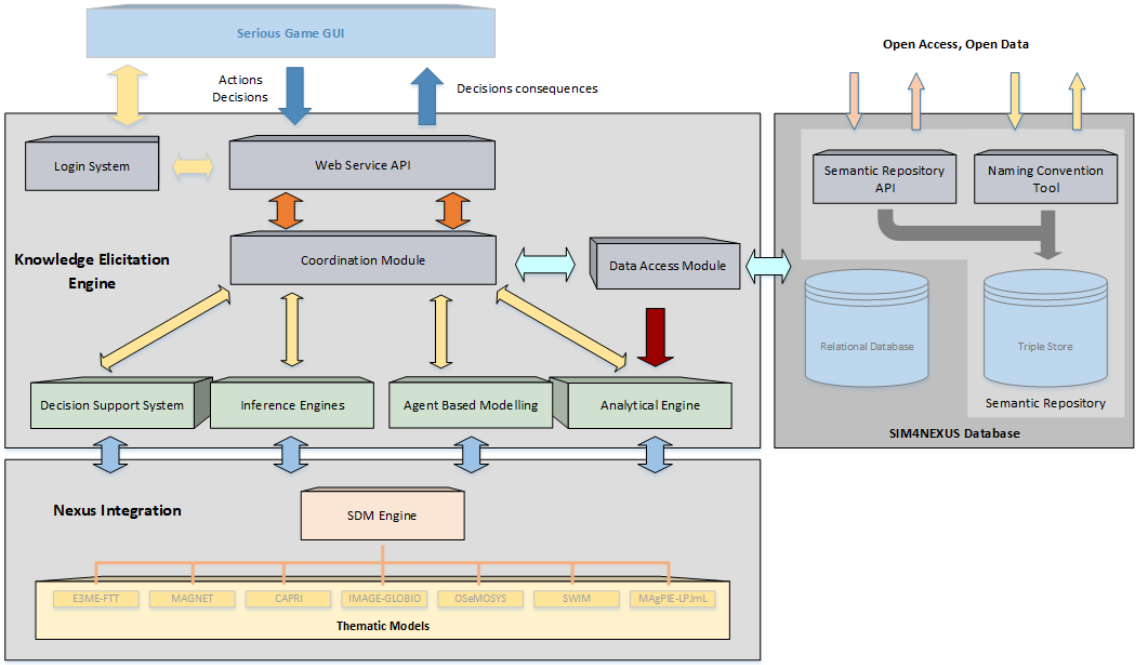


Figure 1. S4N SG Architecture

The development, test and deployment processes needed to implement a high software platform have been divided into (at least) three environments:

- **Production** (PRO): Final environment where all the modules and functionalities have been tested by the developers and other S4N partners. It corresponds to the official S4N SG web site (https://seriousgame.sim4nexus.eu/) and it is open and available for playing the Game. All the training sessions, where the users' feedback have been collected (section 4), have been held through this environment.
- **Pre-production** (PRE): Intermediate environment where the platform and all the modules and functionalities have been previously tested by the developers (in development environments). In this case, the S4N SG is ready to be tested by the S4N partners and experts in order to validate the expected/desired behaviour of the different SGs. Through this environment, the CS members have the opportunity to i) play the Game developed for their CS, ii) test the different components (Policy Cards, Policy Objectives, Policy Goals and Nexus Health) configured through the corresponding excel files and, iii) implement modifications and adapt the previous components (updating the excel files) to achieve the desired performance. From a development point of view, it is considered a second layer of manual testing.
- **Development** (DEV): The base environment where the S4N SG modules and functionalities have been developed and tested. From the SDM Engine to the KEE to the GUI, all the implementations have been thoroughly tested and validated (more details in section 3 Testing of the system) in order to ensure excellent quality and functionality. A series of (automatic) tests have been defined to cover different areas such Global components, Local components, load testing, browsers compatibility, screen resolutions, security, resilience, scalability and/or interoperability.
- **Auxiliar** development environments: When it was considered necessary, due to an important or big development which would imply deep modifications, a fork of the development environment was created and the new code was implemented and tested separately and finally, once it was considered ready, merged with the official DEV environment. This procedure was extremely useful to continue with the development and maintenance of the platform, prevent delays in the different implementations due to other developments and isolate the new (big) functionalities to ensure its correct development and test. Some examples are the implementation of the Login System or the development of Dynamic Policies.

The main software tool, used to manage the platform code and the different environments, is Git[2], a free and open-source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. During the project, EPSILON provided a Git repository (GitLab[3]) to develop the KEE and SR modules, and UNEXE did the equivalent (GitHub[4]) for the GUI developments.

Each of the previously listed environments corresponds to a specif Git code branch (Figure 2).

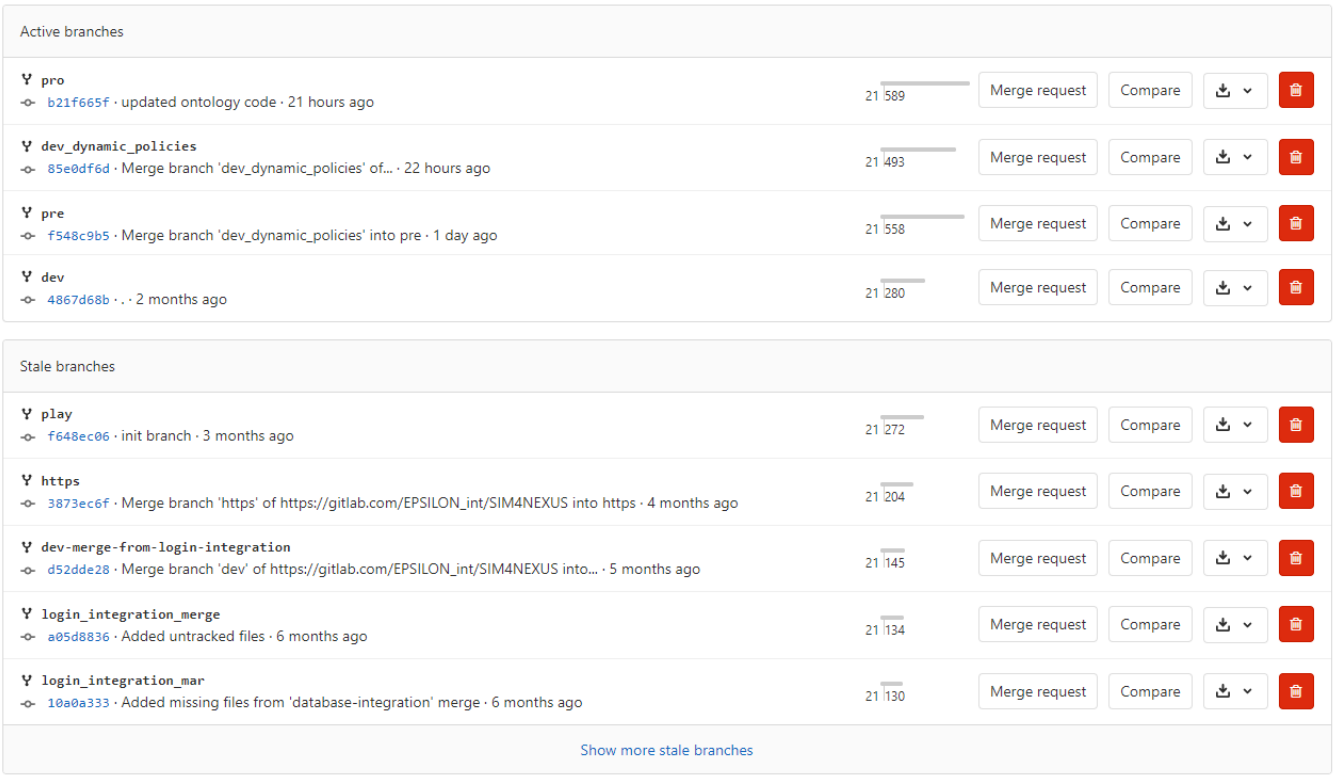


Figure 2. Gitlab repository interface showing different code branches.

Once a new functionality is developed, tested and validated in DEV environments, the PRE environment merges these modifications (from DEV) and it is updated to allow the corresponding tests. Finally, when it is agreed that the functionality is ready, the PRO environment executes the same process and make public the developed functionality. This

[2] https://git-scm.com/
[3] https://about.gitlab.com/
[4] https://github.com/

practice ensures the correct continuous and integration process to finally provide a high-quality outcome.

In terms of deployment and administration of the S4N environments, these tasks have been managed by Docker infrastructure. Each environment has its specific docker configuration to automatize the disposition of its corresponding modules.

With the objective of adding a new layer of security during the deployment of the new implementations, a Docker images version system has been designed. In detail, it allows immediate response in case the latest deployed version has any problems, through the (re)deployment of the previous Docker image version with the latest stable code.

2.1 Architecture

The S4N IC architecture (Figure 3. S4N Integration Centre architecture) has been planned to act as a base infrastructure where all the S4N environments will be deployed and made accessible to the public to be played, in case of PRO environment, or for testing purposes for the rest.

sim4nexus.eu is the domain under which different S4N resources can be found, such as the S4N official web page https://www.sim4nexus.eu/. From this domain, the sub-domain **seriousgame.sim4nexus.eu** has been defined to locate the S4N SG web site https://seriousgame.sim4nexus.eu/**.** During the development of the project, this subdomain has been redirected to the server infrastructure provided by EPSILON (detailed in section 2.2 of this document) where all the environments have been deployed. Through the subdomain and the configuration of different paths, all the environments (and the corresponding modules) have been published and made accessible from the internet.

The frontal face of the IC, and the manager of the URL paths and the corresponding internal routing, is an NGINX Reverse Proxy[5]. Proxying is typically used to distribute the load among several servers, seamlessly show content from different websites, or pass requests for processing to application servers over protocols other than HTTP. In the S4N SG platform, it is used with two objectives: first, to host and serve some of the S4N GUIs and, second, to redirect all the requests to the corresponding S4N resources/services deployed as Docker containers.

[5] https://docs.nginx.com/nginx/admin-guide/web-server/reverse-proxy/

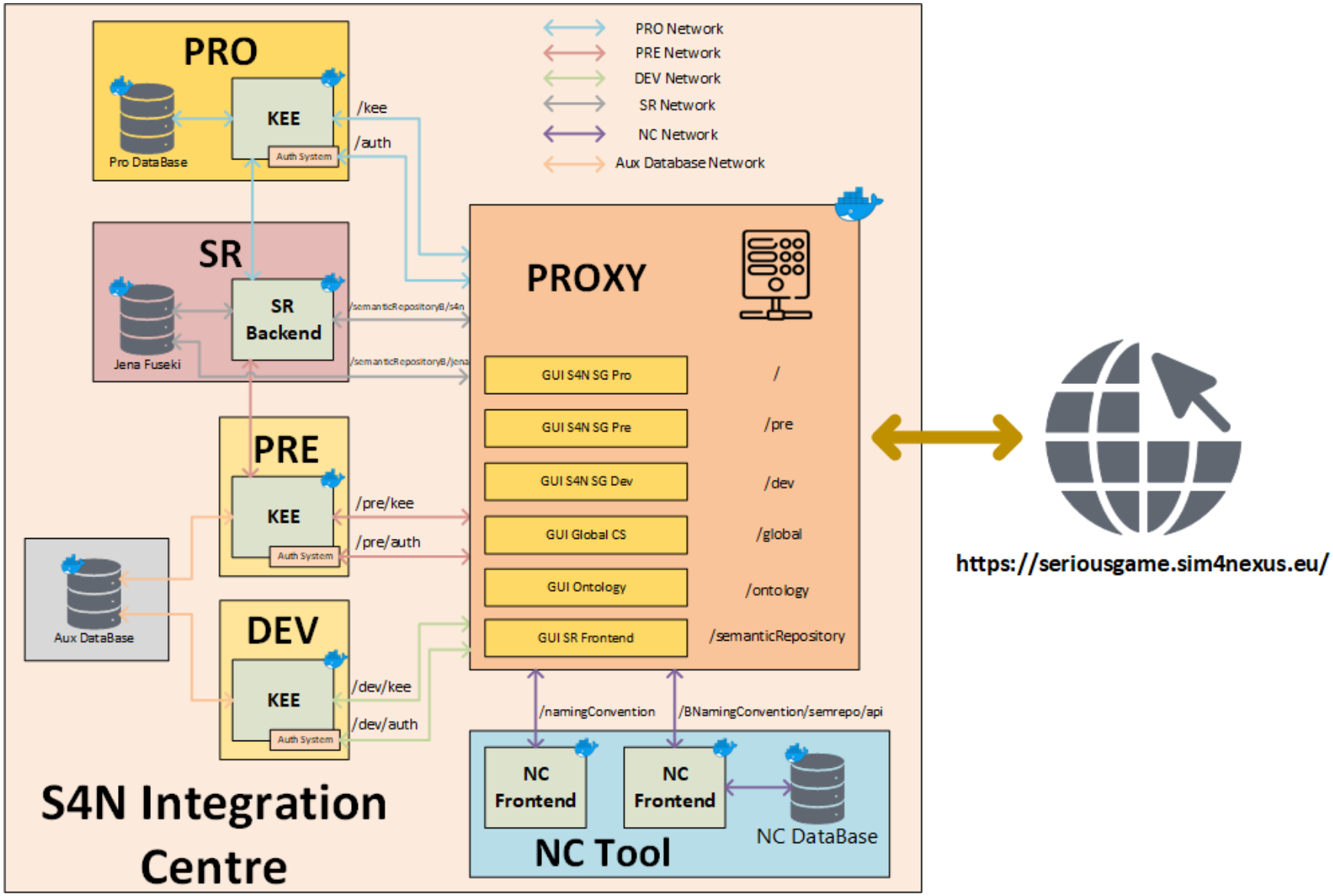


Figure 3. S4N Integration Centre architecture

Internally, the different S4N development environments and modules are encapsulated in Docker containers, as services, and are interconnected through a specific network topology which allows isolated communication between containers.

2.1.1 Proxy

Figure 4 shows a reverse proxy taking requests from the Internet and forwarding them to servers in an internal network. Those making requests connect to the proxy and may not be aware of the internal network.

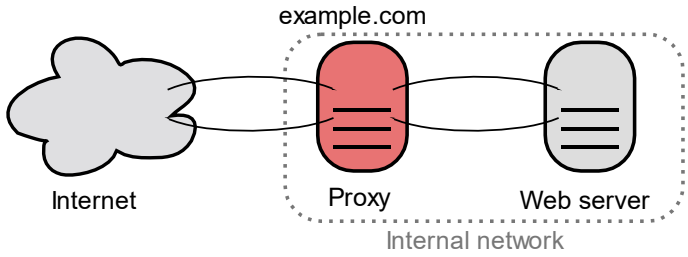


Figure 4. Reverse proxy concept[6]

In the following table (Table 1), the different URLs (and paths) are linked to the resource that identifies.

URL Resource (the URL root is https://seriousgame.sim4nexus.eu)	Service	S4N Env	Docker container
/	GUI S4N SG	PRO	s4n_nginx_ui_pro
/global	GUI Global CS	PRO	s4n_nginx_ui_pro
/ontology	GUI Ontology	PRO	s4n_nginx_ui_pro
/semanticRepository	GUI SR Frontend	PRO	s4n_nginx_ui_pro
/semanticRepositoryB/s4n	API SR Backend	PRO	sr_backend
/semanticRepositoryB/jena	API SR Jena	PRO	sr_jena
/namingConvention	GUI NC Frontend	PRO	naming_convention_frontend
/BNamingConvention/semrepo/api	API NC Backend	PRO	naming_convention_backend
/kee	KEE S4N SG	PRO	s4n_pro
/auth	KEE S4N SG	PRO	s4n_pro
/pre	GUI S4N SG	PRE	s4n_nginx_ui_pro
/pre/kee	KEE S4N SG	PRE	s4n_pre
/pre/auth	KEE S4N SG	PRE	s4n_pre
/dev	GUI S4N SG	DEV	s4n_nginx_ui_pro
/dev/kee	KEE S4N SG	DEV	s4n_dev
/dev/auth	KEE S4N SG	DEV	s4n_dev

[6] https://en.wikipedia.org/wiki/Proxy_server

Table 1. S4N Service URL locations

The services of the IC, listed in Table 2, have not been included in the previous table because they cannot be accessed from internet through a specific URL. They correspond to the internal databases of the S4N platform and can be directly accessed via IP for its management.

Service	S4N Env	Docker container
PRO database	PRO	postgres_pro
Aux database	PRE & DEV	postgres
NC database	PRO	naming_convention_mongo

Table 2. S4N database services

The SR and NC services have been developed and tested in DEV environments but have been not included in Figure 3 (neither Table 1 nor Table 2) since the services are completely operative and the corresponding DEV environments are no longer needed nor maintained.

2.1.2 Environments

To ensure a high-quality final system, the development of the S4N platform has been conducted through different development and validation environments allowing a continuous development procedure. From DEV environments, to PRE, to finally PRO, the new functionalities and all the implementations have been exhaustively tested and validated to provide a robust and consistent platform.

DEV: Development Environment

It corresponds to the initial environment where the S4N SG modules and functionalities have been developed and tested. From the SDM Engine to the KEE to the GUI, all the implementations have been thoroughly tested and validated in order to ensure excellent quality and functionality. A series of (automatic) tests have been defined (section 3 of this document) to cover different areas such Global components, Local components, load testing, browsers compatibility, screen resolutions, security, resilience, scalability and/or interoperability.

Since the developments have finished, it can be considered closed, but it will not be removed because, in case any code bug or problem is detected, DEV environment will be used to correct it.

Also, the SR and the NC tool have been initially developed in this environment to finally be moved to PRE and PRO after the corresponding validations.

DEV Aux: Other development environments

When it was considered necessary, due to an important or big development which would imply deep modifications, a fork of the development environment was created and the new code was implemented and tested separately and finally, once it was considered ready,

merged with the official DEV environment. This procedure was extremely useful to continue with the development and maintenance of the platform, prevent delays in the different implementations due to other developments and isolate the new (big) functionalities to ensure its correct development and test. Some examples are the implementation of the Login System or the development of Dynamic Policies.

All the Auxiliar DEV environments will be closed since their functionalities have been successfully merged to the DEV (and PRE and PRO) environments.

PRE: Pre-Production Environment

An intermediate environment where the platform and all the modules and functionalities have been previously tested by the developers (in development environments). In this case, the S4N SG is ready to be tested by the S4N partners and experts in order to validate the expected/desired behaviour of the different SGs. Through this environment, the CS members have the opportunity to i) play the Game developed for their CS, ii) test the different components (Policy Cards, Policy Objectives, Policy Goals and Nexus Health) configured through the corresponding excel files and, iii) implement modifications and adapt the previous components (updating the excel files) to achieve the desired performance. From a development point of view, it is considered a second layer of manual testing.

Similarly to the DEV environment, PRE will remain operative since the correction of possible bugs will also be checked here before it goes to PRO environment.

PRO: Production Environment

Final environment where all the modules and functionalities have been tested by the developers and other S4N partners. It corresponds to the official S4N SG web site (https://seriousgame.sim4nexus.eu/) and it is open and available for playing the Game. All the training sessions, where the users' feedback have been collected (section 4), have been held through this environment.

PRO environment will remain always up since it hosts the official S4N SG platform.

2.2 Hardware resources

Within the project, the SIM4NEXUS Serious Game (SG) has been hosted on a virtual private server (VPS) provided by EPSILON.

Server Details

VPS XL SSD	
IP	193.164.132.202
IPv6	2a02:c205:2018:9354::1
Location	Munich
VNC	213.136.66.222:63005
Host system	4580
OS	Ubuntu 18.04 (64 Bit)

Key technical specifications:

- state of the art hardware and virtualization based on KVM
- Ten CPU cores (Intel® Xeon® and AMD Epyc™ processors)
- 60 GB RAM
- 1600 GB of SSD disk space for a maximum data transfer rate
- 4 snapshots for a quick system restore
- unlimited traffic
- 1 Gbit/s port
- DDoS protection
- VNC access

All the tests presented in section 3 have been performed in this infrastructure.

Once the project finishes, the S4N SG platform will be deployed in a different server with similar technical specifications, to let the game be available during at least coming 2 years.

2.3 Deployment

As pointed out, Docker has been the key technology whereby the S4N platform and all its environments and modules have been deployed.

Each S4N environment has its own docker-compose[7] file that orchestrates the deployment of the required Docker containers (identified with the Docker symbol in Figure 1. S4N SG Architecture) corresponding to the different modules (such the GUI, KEE or S4N databases). Through this docker-compose file, the environments are configured and the network topology which connects the modules is specified.

Φive docker-compose files, and the underlying Docker containers, have been defined:

[7] https://docs.docker.com/compose/

- PRO env:
    - Nxing proxy: which contains the PRO GUI among other GUIs.
    - KEE
    - PRO database
- PRE env:
    - KEE
- DEV env:
    - KEE
- SR:
    - SR backend
    - Jena Fuseki
- NC:
    - NC Frontend
    - NC backend
    - NC database

The auxiliary database used in PRE and DEV environments has been deployed as a separate and isolated docker container into a specific network.

3. Testing of the system

Testing activity over S4N side involves a wide variety of operations which need to be classified. This classification brings up more clarity to those testing activities, their procedures and the final purpose of each one.

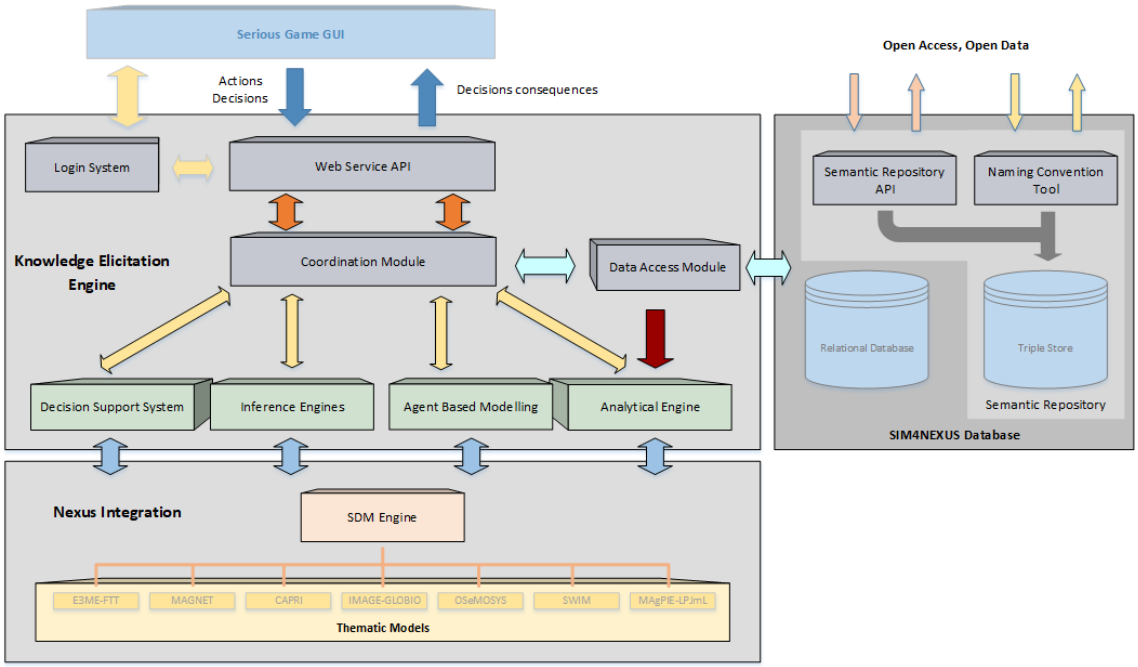


Figure 5: Serious Game tool Architecture.

Taking as a reference the already existent diagram which reflects the depth of the entire system (see Figure 5) it can be seen how, from a generalized point of view, the components constantly interact with each other, always waiting for the correct functionality as a key to continue working normally.

Several factors can alter this behaviour (pre-release versions, hot patches, versions conflict and many other human factors). These ones must be detected quickly and in as much detail as possible. To achieve this functionality of control over the existing system, a series of validation programs and scripts have been developed. These have integrated into their code the correct functioning that a subsystem of the Serious Game Architecture should present as well as its desired outcome.

The programs for controlling behaviour, development, outcome and evolution of the subsystems have all the aspects that a system can adopt during its operation. These aspects are classified as valid or invalid according to the purpose of the system in question. In the case that the system adopts an invalid position, it will quickly notify the scenario and the background that generated this outcome. With this data, the team is going to analyse where the error comes and how to solve it in the shortest time possible. In case the subsystem presents a non-defined or "new" behaviour for the test script, it will be quickly reported as such to improve the alert system, contemplate new scenarios and know how to solve them.

For the general purpose of the testing systems, the entire testing activity needs to be focused at different levels. This is intended to obtain both good parts of testing a Component and testing a Global System.

By testing Global Systems, the overall testing activities benefit from having a simplified and comprehensive view of the entire structure. This base allows to quickly detect an error among the procedures that exist between the subsystems. However, being able to quickly detect the errors in the global system have a negative side effect. This effect is the detail of the error. By making the test system more generic, it is easy to spot an error but more difficult to know the specific interaction that produced this error.

To solve this necessity of detail a second the second type of testing activity have been implemented, the Component testing. Those types of testing are linked by behaviour with the Global System test so, when an error is spotted in a generic location, the Components tests of this location take part into the action to provide more information on the error and detect its origin.

3.1 Global components

Going back to the already introduced Figure 5, the Serious Game Tool Architecture has four main components which can be divided into three principal data flows. Starting at the Serious Game GUI, it continually exchanges information with the Knowledge Elicitation Engine (KEE). Once the KEE is reached two possible paths can be followed. First, the S4N Database, to successfully store the information about the games gathered data. And, secondly, the Nexus Integration, which embeds the corresponding SDM of each Case Study to successfully simulate the scenarios of the Serious Game.

During this quick summary of the communication between components, there have been introduced three communication channels that need to be analysed to guarantee their correct behaviour (see Figure 6). This is intended to avoid unwanted scenarios, the consequence of the modification of the behaviour of any of these components.

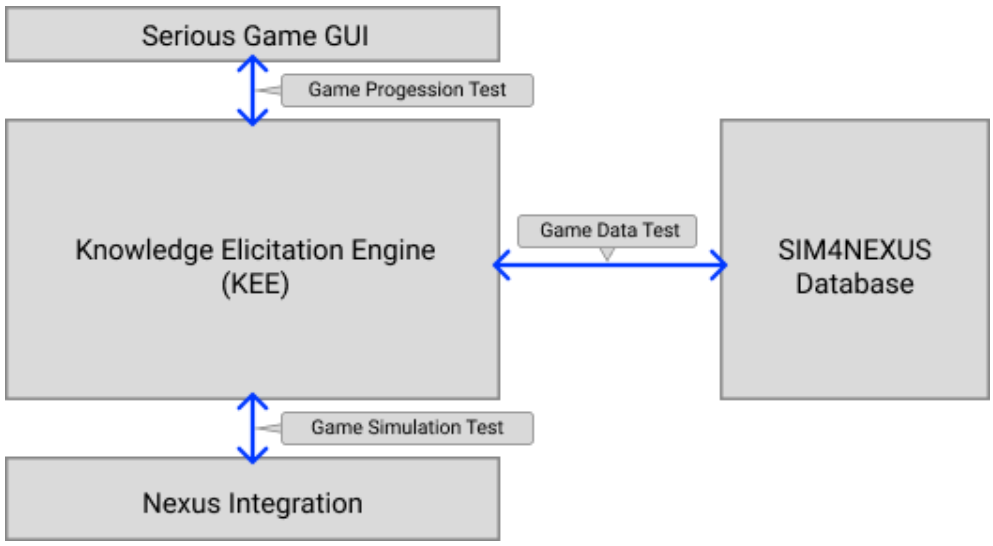


Figure 6: Simplified Serious Game Architecture connections

3.1.1 Game Progression Test

The goal of this Global System unit is to prove the correct behaviour among the Serious Game GUI and the Knowledge Elicitation Engine. It is important to notice that, even though they are global tests, they need to be isolated from other global tests. If these tests started to depend on each other, it would only trigger a great and unique general test.

For this reason, the Game Progression Test only tests the trigger of the communication between the Serious Game GUI and the KEE. By testing the game progression, it will only attach the Game Simulation Test to this Game Progression Test.

Requirements:

To trigger this Global System Testing unit, it is necessary to run the S4N_Global_TestingUnit.py script with the following options:

Table 3: Game Progression Test arguments

argv0	argv1	argv2
Configuration file name without the extension.	List of the Case Studies to be tested with the format:	Enable Game Progression Test (Y/N) with the format:
	[number,number, …] [2,3]	"Y" or "N" (without "") Y

The first two parameters (argv0 and argv1) will introduce the Testing unit into their scenario. By providing the Serious Game context and the Case Studies involved in it. Referring to the third parameter (argv3), it allows to enable the execution of two Global System testing units at once but completely isolated one from the other, the main purpose of it is to reduce execution time and complexity. If set to yes the SDM will run with all PolicyCards, PolicyGoals, Policy Objectives and Nexus Health formulas active to perform a full simulation. Otherwise, the system will run a dummy SDM which will only fail if the SDM has semantic exceptions on it.

Functionality:

This test will simulate the interaction of a user with the Serious Game GUI by providing HTTPS requests to the KEE to progress in the game evolution. The goal of the test is divided into different steps. Each one reported on console with the format:

>> [TEST UNIT]: Game Progression Test, Step (1/8), Trigger serious game initialization, ERROR

Meanwhile, the Test Unit goes one the Console logs will trigger similar outputs but increasing the steps of it. Once an unexpected behaviour is detected, the Test Unit will trigger a report of it. Reports logs follow the following format:

>> [TEST UNIT]: Game Progression Test, Step (1/8), Trigger serious game initialization, ERROR

Thanks to the combination of both types of console logs, the Global System Testing unit can provide feedback about which component is not following their desired behaviour based on its expected communication protocol.

The testing steps for Game Progression Test are defined as:

Table 4: Game Progression Test Steps

Step	Action	Endpoint
Serious Game initialization	Request to	/kee/wps?identifier= sim4nexus_initialization_step
Download of Case Study scenario	Response from	/kee/wps?identifier= sim4nexus_initialization_step
Progression of turns: 2/6 to 6/6	Request to	/kee/wps?identifier= sim4nexus_simulation_step
End of Serious Game Session	Response from	/kee/wps?identifier= sim4nexus_simulation_step

Each of the previous steps can provide different information about which component is correctly performing their work and which component is showing up an unexpected behaviour.

Table 5: Game Progression Test outcomes

Step	Name	Outcome
1	Serious Game initialization	Response: [200], Expected behaviour, the KEE has received and understood the data provided to initialize a new Serious Game instance of the specified Case Study.
		Response: [400], Unexpected behaviour, the KEE is up and working ok but the request is not correctly formatted. If this is the first time the request is done it can be an HTTPS request format problem. If the request was working in previous versions of the Serious Game, this can be a modification in the KEE request required parameters.
		Response: [404], Unexpected behaviour, the KEE is up but not ready to process requests about creating a new Serious Game instance. The reasons can be because of a maintenance process or because the Case Study does not exist.
		No response: [Timeout], Unexpected behaviour, the KEE does not respond to the Serious Game GUI HTTPS request. The KEE might be down due to an unexpected error or because of maintenance tasks.
2		Response: Not empty and required fields are present,

	Download of Case Study scenario	Expected behaviour, the KEE has returned the entire Serious Game context of the specified Case Study.
		Response: Not empty and required fields are not present, Unexpected behaviour, the KEE is missing required fields in the response. This can be a sign of inconsistences inside KEE system.
		Response: Empty, Unexpected behaviour, the KEE has failed to launch the new Serious Game instance or the HTTPS response building system is showing some inconsistences.
3	Progression of turns	Response: [200], Expected behaviour, the KEE has received the modifications to be done over the existent Serious Game instance. It has applied those changes without any unexpected behaviour and returned a new scenario for the Serious Game instance.
		Response: [Not 200], As the simulation process is a progression of the initialization process any result different than 200 will be considered as an unexpected behaviour. The errors contained in this section will all be a consequence of an unexpected error or issue during the SDM simulation process of a Case Study given a scenario and a certain modifications to be done.
4	End of Serious Game Session	Last Response: [Turn 6 reached, 200], Expected behaviour. This step is highly linked to the previous one (Step 3). If the Game Progression unit of test manages to successfully receive the response of the simulation request made for the Turn 6, this will be considered as a successfully ended game session.
		Response: [Turn 6 not reached, not 200], Following the format of step 3. This step will consider an error any response for Turn 6 simulation which returns an HTTPS response different from 200.

3.1.2 Game Simulation Test

This test can be easily run by enabling a Boolean (yes/no) parameter among the S4N_Global_TestingUnit.py script. The Game Progression Test unit depends on being able to run the SDM at least once to generate an output scenario of the Serious Game instance.

As it was mentioned in the introduction of the 'Report of testing' section, this dependency among tests is dangerous as makes the entire execution foggy. The errors can be hidden by bigger ones which will point to an incorrect component failing on its duty.

The solution, when two components are highly dependent on each other, is to create a "dummy" version of one of them. To select which Component needs a dummy version of it is only required to follow the game flux and spot which is the component with less interaction over it. To this end, by making the Component (with the most interactive behaviour) a dummy, the overall testing activity is being seriously damaged leaving out outcomes that will not be able to be tested.

Taking the two involved components *Serious Game GUI* and *Nexus Integration* it can be easily spotted an overwhelming difference among the integration that each one can hold.

On one side, Serious Game GUI holds the Case Study variables which will modify the outcome of the next turn. These variables are set thanks to the free interaction of the User with the Policy Cards leaving into almost a n^n possibilities (where n take the value of the amount of PolicyCards). Which results is an important amount of interaction to be considered.

When comparing this interaction with the other side, the Nexus Integration, an important result comes up. The SDM Engine computes high amounts of information but, since it is a script, its parameters need to be fixed leading into a deterministic behaviour (adequate to be summed up into a dummy).

Requirements:

If this Game System unit needs to be tested, it can take advantage of the parameters introduced in subsection 3.1.1. The number of parameters will be the same but the third one must be activated.

Table 6: Game Simulation Test arguments

argv0	argv1	argv2
Configuration file name without the extension.	List of the Case Studies to be tested with the format:	Enable Game Progression Test (Y) with the format:
	[number,number, …] [2,3]	"Y" (without "") Y

Functionality:

This Global System test is based in a previous working test so it establishes an order of execution to be correctly tested. See Figure 7 to follow this explanation.

The previous test, which needs to be run and tested without errors, is the Game Progression Test unit (S4N_Global_TestingUnit.py script) with the given parameters:

Table 7: Example of simulation test arguments

argv0	argv1	argv2
Configuration file name without the extension.	List of the Case Studies to be tested	"N"

Note that the third parameter needs to be fixed to "False" or "No" by providing an "N" character. This will perform the Game Progression Test execution. This execution will only bring information about how Serious Game GUI and KEE are performing together. No information about the interaction between KEE and Nexus Integration will be provided as this last one it is a placeholder which only shows up the perfect behaviour for the SDM which needs to be tested later (see Figure 7, step 1).

Once the Global System test for Serious Game GUI and KEE finishes without errors, a deduction can be assumed in order to start performing the Game Simulation test. If Serious Game GUI and the KEE perform correctly taking a dummy version of the real Nexus integration, the original Nexus Integration component can be brought up to test their behaviour (see Figure 7, step 2).

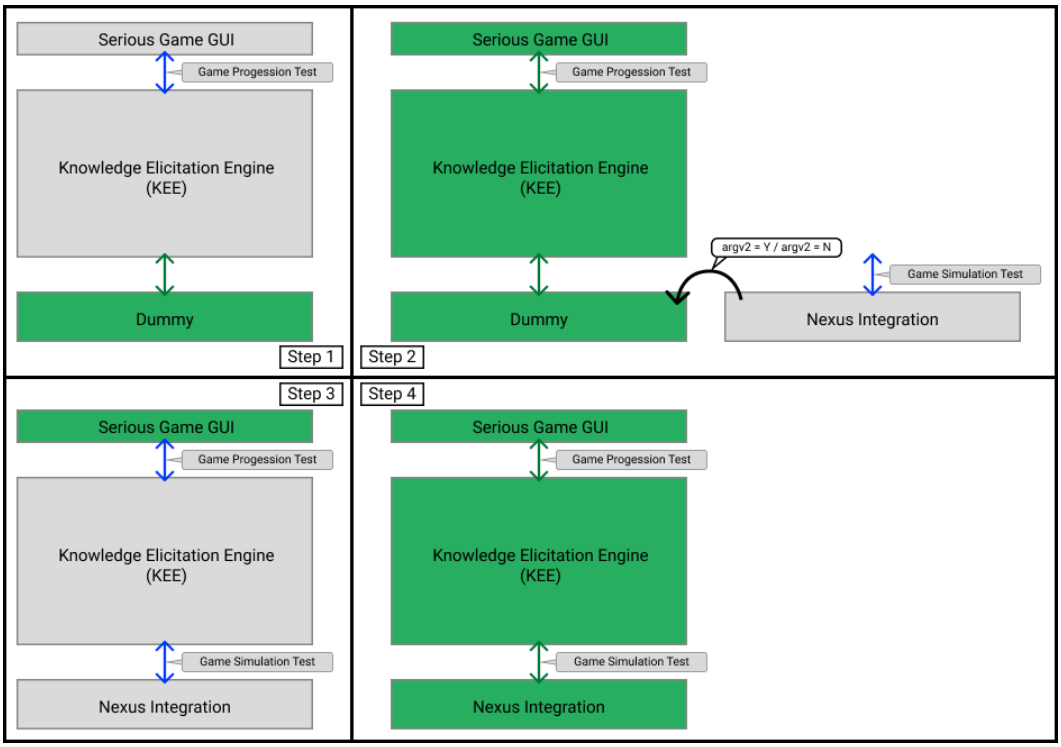


Figure 7: Game Simulation testing process

Performing the Test shown in Figure 7, step 3, there are no confirmations that the Serious Game GUI test will perfectly pass. But if this test does not pass, thanks to Steps 1 and 2 of the same Figure, it can be deduced that the problem is between the new interaction introduced among the KEE and the Nexus Integration. This entire procedure answers the original necessity of isolation among tests to know at any moment which system is failing.

As a consequence of this high dependence between the systems of progression in the game and simulation, the testing steps for Game Simulation Test are equivalent to the steps already introduced in the subsection 3.1.1.

Table 8: Game Simulation Test endpoints

Step	Action	Endpoint
Serious Game initialization	Request to	/kee/wps?identifier= sim4nexus_initialization_step
Download of Case Study scenario	Response from	/kee/wps?identifier= sim4nexus_initialization_step
Progression of turns: 2/6 to 6/6	Request to	/kee/wps?identifier= sim4nexus_simulation_step
End of Serious Game Session	Response from	/kee/wps?identifier= sim4nexus_simulation_step

The difference arrives when deductions need to be extracted from the responses of these steps. Inside the Game Simulation test, it is already known that the Game Progression unit works

well (as it is a requirement to perform this test). Thanks to this premise, the responses and errors get from the HTTPS requests will have different meanings, the vast majority of them will point out behaviours of the Nexus Integration System.

Table 9: Game Simulation Test outcomes

Step	Name	Outcome
1	Serious Game initialization	Response: [200], Expected behaviour, the KEE has successfully initialized a new instance of Serious Game. This instance has run until the temporal mark of 2020 proving that the Game Simulation System without interaction successfully works.
		Response: [400], Unexpected behaviour, the KEE is up and working ok as the response arrived. This error can be a consequence of two scenarios. The first of them is a wrongly formatted request. Meanwhile the second one refers to an exception product of running the Nexus Integration system to simulate the Serious Game until the timestamp mark of 2020. As the "dummy" version of the SDM was working ok, this error points out a problem with the Nexus Integration System.
		Response: [404], Unexpected behaviour, the KEE is up but not ready to process requests about creating a new Serious Game instance. The reasons can be because of a maintenance process or because the Case Study does not exist.
		No response: [Timeout], Unexpected behaviour, the KEE does not provide a response the Serious Game GUI HTTPS request. The KEE might be down due to an unexpected error or because of maintenance tasks.
2	Download of Case Study scenario	Response: Not empty and required fields are present, Expected behaviour, the Nexus Integration System correctly applies the stocks, interventions and history of actions for a Case Study
		Response: Not empty and required fields are not present, Unexpected behaviour, the KEE is missing required fields in the response. This can be a sign of inconsistences inside SDM code or missing data inside the Case Studies initialization data.
		Response: Empty,

		Unexpected behaviour, the KEE has failed to launch the new Serious Game instance or the HTTPS response building system is showing some inconsistences.
3	Progression of turns	Response: [200], Expected behaviour, the KEE has received the modifications to be done over the existent Serious Game instance. It has applied those changes without any unexpected behaviour and returned a new scenario for the Serious Game instance.
		Response: [Not 200], As the simulation process is a progression of the initialization process any result different than 200 will be considered as an unexpected behaviour. Any response different than 200 while running a simulation step after successfully running an initialization step (steps 1 and 2) will point out a problem with the policies system. This system encapsulates the PolicyCards, PolicyObjectives, PolicyGoals and NexusHealth systems.
4	End of Serious Game Session	Last Response: [Turn 6 reached, 200], As it was introduced in subsection 3.1.1. this step it is just a progression of step 3 but reaching the goal of the Serious Game (turn 6). Reaching the last turn without any problem (code 200) will mark the overall Game Simulation Test as a success.
		Response: [Turn 6 not reached, not 200], Following the same trend, getting an error during the simulation of the last turn is a strange behaviour if the previous 5 turns were correctly simulated. The main reason to reach this unexpected behaviour will be because of a misunderstanding between the KEE and the SDM about the desired length of the variables. Variables store values of each month. Any difference about the initial length of the value attribution system for these variables can produce this scenario.

3.1.3 Game Data Test

Regarding the tests about data storage, this process is highly isolable. Storing data consists into a link between a program and a Database Administrator to select, save, update or delete a specific amount of information.

By this definition, the Game Data Test follows a similar procedure as the one seen in subsection 3.1.2. There is an undeniable link with the Serious Game GUI, since the data which it is saved during the KEE procedures, becomes the context of the Serious Game instance which is running and being modified. The User will always have control over the information which is stored in the Database as their interaction with the applied PolicyCards are the modifiers of the base evolution of the Case Study which is being played.

As a result, again there will exist an execution order of the Global System tests. This order will be composed by an initial successful execution of the Game Progression Test followed by the execution of this Game Data Test. The procedure is exactly the same as the one followed in subsection 3.1.2. among Game Progression Test and Game Simulation Test.

This similitude does not mean that there will be a dependency in which refers to Game Simulation Test and Game Data Test, both tests are completely isolated and can be run individually or together as will be shown in Functionality section. The only element these tests have in common is the dependency with Game Progression Test, as it is necessary to first check the system that inputs data in order to simulate new data (Game Simulation Test) or store this data (Game Data Test).

Requirements:

Following the idea seen in the previous test, this one can also be tested taking advantage of the Game Progression Test unit (S4N_Global_TestingUnit.py script). The addition of this isolation among Game Simulation Test and Game Data Test brings up a new parameter to this script. By default, this parameter is set to "False" or "No" and will enable the execution of the Game Data Test.

Table 10: Game Data Test arguments

argv0	argv1	argv2	argv3
Configuration file name without the extension	List of the Case Studies to be tested with the format:	Enable Game Progression Test (N/Y) with the format:	Enable Game Data Test (N/Y) with the format:
	[number,number, …] [2,3]	"N", "Y" (without "") Y	"N", "Y" (without "") Y

The existence of both parameters (argv2 and argv3) reflects how the Game Simulation Test and Game Data Test can be run at the same time or individually. The execution of both tests at the same time will never lead to an error as these systems do not have any sort of communication among them.

Functionality:

This test functionality inherits from the structure seen in section 3.1.2. The base idea about having to previously run a more interactive test (Game Progression Test) will remain the same.

The main difference appears in terms of dependencies. This test, unlike the Game Simulation Test, is completely isolated from the progress of the game. Progression and simulation systems do not depend on the storage system to work. This storage system can easily be deactivated until its specific behaviour is checked.

As a consequence, the overall process (see Figure 8) will be quite similar to the process seen in Figure 7 but without the special need of having a dummy version of the system we want to test.

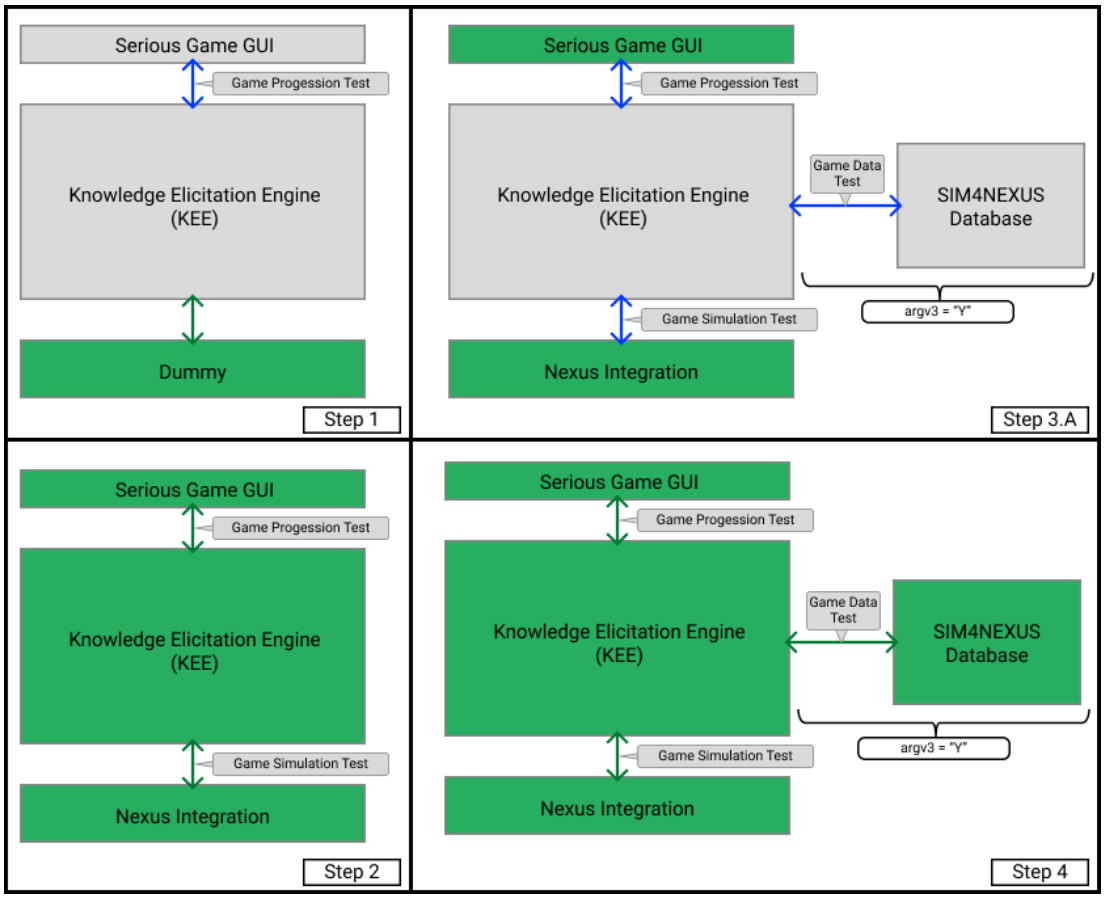


Figure 8: Game Data testing process

Following Figure 8, the preliminary testing activity starts like the one seen in section 3.1.2. taking a dummy version of the Nexus Integration system to test the Serious Game GUI (step 1) and end up with both systems successfully tested (step 2). From here, starts the procedure to check the behaviour of the SIM4NEXUS Database. Thanks to the third parameter passed to the script (argv3), the database functionality can be enabled (step 3). Once it is working, by creating a new Serious Game instance and making progress in their timeline, the changes applied to the base scenario will start to be persisted and checked inside the Database. The final result of this operation will be a complete system test (step 4).

It is important to notice that inside Figure 8, step 3 is referenced as "Step 3.A". This is an intended nomenclature. As it was previously introduced, the System Game Data test unit allows running without any dependency linked to the Game Simulation Test unit. Thanks to this, if the third parameter (argv3) it is set to "True" or "Yes" by providing an "Y" character, while argv2 is kept as deactivated, it will allow testing the entire Game Database System without having to previously test the Game Simulation System. Resulting in a new testing flow, starting at step 1 and directly jumping into "Step 3.B" (see Figure 9).

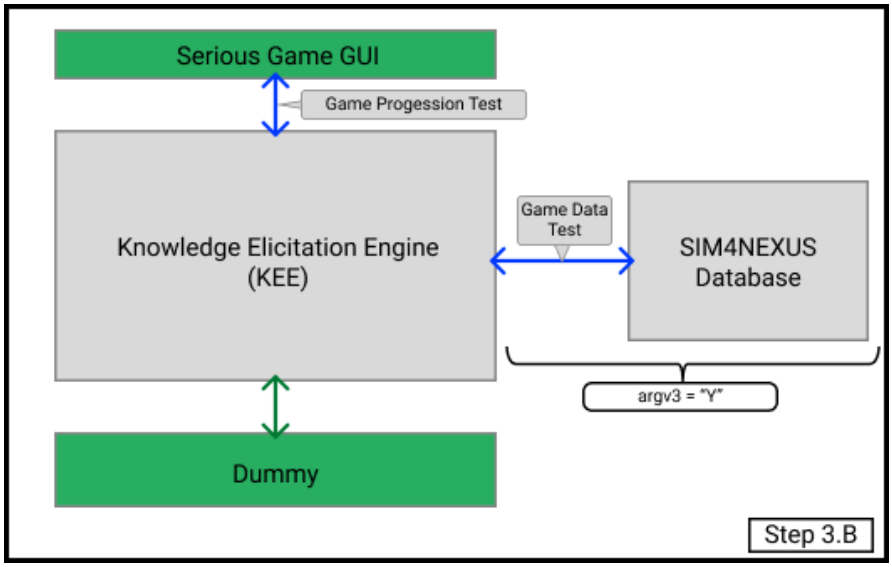


Figure 9: Game Data alternative testing process

Different from the behaviour seen in subsections 3.1.1. and 3.1.2., this Global System Test it is strongly isolated from the rest. As a consequence of it, the Game Data Test steps will not have any similitude with the steps seen in previous sections. This system is only responsible for storing the data related to those modifications that were passed to the Nexus Integration in order to compute an outcome scenario different from the expected base scenario.

Table 11: Game Data Test steps

Step	Action	Endpoint
Store Serious Game modifiers	Request to	/kee/wps?identifier= sim4nexus_initialization_step
Get and check Serious Game data	Request and response from	/games/findBySessionId

Some of the previous considerations need to be taken into account before performing this test. First, this test can only be performed if the Game Progression Test has ended with a successful result. The second consideration to be taken into account refers to the Nexus Integration System. It is not required to previously pass the Game Simulation Test since a "dummy" version of it can take place. As it was introduced before, this dummy version is always the same and it is already proved that it does not lead to errors.

Thanks to these considerations the responses provided by the system will focus their reasoning in the Game Data system.

Table 12: Game Data Test outcomes

Step	Name	Outcome
1	Serious Game initialization	Response: [200], Expected behaviour, the storage process has successfully ended. The Database Management System has successfully performed an INSERT operation inside the Game class. Content should be ready to be checked by performing a SELECT operation (step 2).
		Response: [400], Unexpected behaviour, by receiving a generalized error (code 400) from the KEE it can be spotted an error during the storage process. This can be a consequence of different causes: - An incorrect database initialization. - An incorrect definition of the classes used by the Database System. The database route is currently down performing an exception during the storage process.
		No response: [Timeout], Unexpected behaviour, the KEE does not provide a response the Serious Game GUI HTTPS request. The KEE might be down due to an unexpected error or because of maintenance tasks.

2	**Get and check Serious Game data**	**Response: [200, correctly formatted],** Expected behaviour, by performing a SELECT over the Database with the Game Session id, the entire Serious Game Instance can be collected and analyzed. When this analysis shows up that all the fields match the model and nothing is set to "Null" or "Empty", the overall process can be considered as a success.
		Response: [200, wrongly formatted], Unexpected behaviour, this happens when receiving an object without the expected fields in the model. This situation is a clear indicator of a wrong specification among the Game Database system models.
		No response: [Timeout], As this step also involves a communication with the KEE API and the Database Manager System, receiving an exception for timeout can be caused by: KEE is down or under maintenance Database instance is down or under maintenance

3.2 Local components

Once the Global Testing system has been introduced and explained, following the guidelines seen in the introduction, the following procedure can be presented. The Component Testing units have the mission of finding the exact system which is causing the error. This spotting procedure is based on a much more reduced and specific test scope.

Component testing will be a layer under Global System Testing providing much more information about which system is failing and why (see Figure 10). As it was seen in point 1.1. subsections, Global System Test provides useful information about which communication is failing and in which system the error is located at. However, when it refers to solving the error, its report is still pretty general and involves various subsystems. This will result in slow and cumbersome work for the programmer as it would be necessary to check the subsystems and their operation one by one. When the Components Testing layer is added and is coordinated with the already existent Global System Tests, their cooperative work leads into a precise report of the flux state and the error location.

Given an overall vision of which communication is failing (Global System Testing). Components involved in this communication will perform specific checks along with the program flow of execution. Their purpose is to detect at which exact point the flow stops. Once this information is gathered and contrasted with the information about the failing communication, more precise deductions can be extracted.

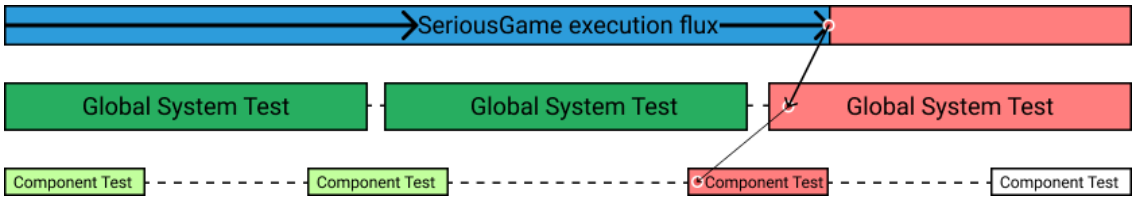


Figure 10: Serious Game flux with exceptions

If all the problems are solved or the Global System Tests and Component Tests did not report any error, the output will be a successful flux evolution (see Figure 11). When this status is reached the overall system is proved as stable.

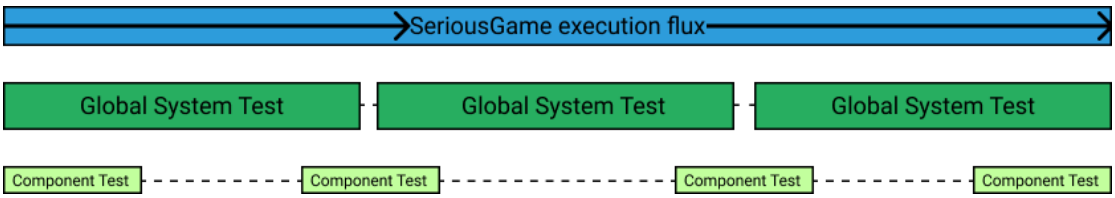


Figure 11: Successful Serious Game flux

Based on the simplified view of the Serious Game architecture presented in Figure 6. A common pattern for these communications can be spotted. Each communication between systems will previously involve a necessary unavoidable communication with the KEE. This structure adds dependency into the Knowledge Elicitation Engine but it also adds order and format. Thanks to having all the communications managed by KEE, the entire Component Test System can be built around it. This system is not allowed to be introduced inside the already existent code of the KEE. Test scripts should be easily spottable and isolated from the code. This is to avoid unwanted dependencies or influences from the already existent code (global variables, existent functions, etc.). By isolating the tests scripts it also makes easier to scale them into a bigger or more specific version thanks to having all the imported features under control.

The resulting system will be an entire new script (S4N_Component_TestingUnit.py) which will contain the necessary Component Tests on it (see Figure 8). Its input will focus on a specific Global system failing. Depending on this input it will trigger a specific set of Component Tests to spot at which point the exception is located at.

The Components Test, inside this environment, will not have a specific System to test but a specific component. Since a component can be involved in different systems, this testing unit needs to be able to concatenate and execute this Component Tests in different combinations depending on the System that is failing.

Components tests will be classified based on the KEE subsystems and their interaction with other components of the Serious Game. This will lead to two main tests (Figure 12).

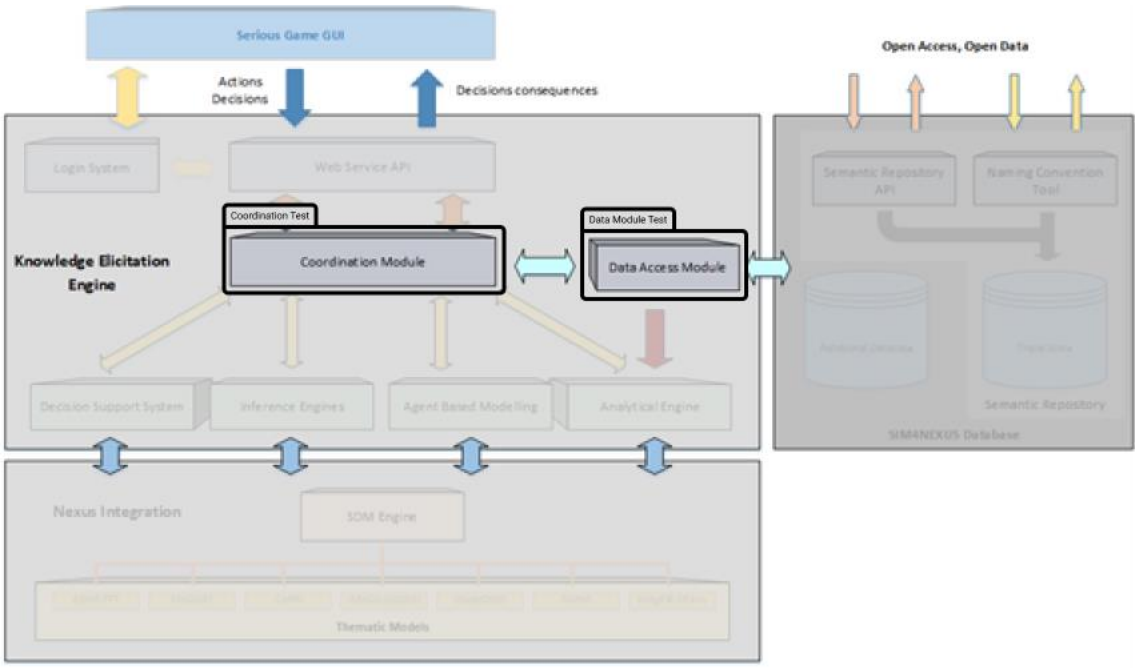


Figure 12: Serious Game Component-based Tests architecture

Following Figure 12, and the centralized structure of the KEE, only one main Component-based test would be needed. The entire execution flux goes through the Coordination Module and, by checking which subsystem is failing at this point, the location of the exceptions can be spotted.

Actually, two Component Tests were needed because of the integration with the Database. When a program needs to work with a database, a Database Manager System is needed. This is an intermediary between the KEE and the Database. Its functionality is to establish and keep a working connection with the Database as well as perform SQL queries over it. Because of the importance of the privacy of the data is highly recommended to have this component isolated from the main flux of the application. This is the reason why KEE Coordination Module does not have access to this Data Access Module to know stats about how the Database is performing. It is necessary to retrieve this information from the Database Manager System, check the privacy of it and send it to this Coordination Module. As a result, this specific component is tested individually.

When this Component Tests ends, the exception can be classified into two scenarios. First, the exception is allocated inside the KEE. In this case, the exception will be returned as well as the subsystem which is causing it. In which refers to the second scenario, when the exception is allocated outside the KEE, the level of detail is still acceptable. In those cases, the exception is classified following:

Table 13: Serious Game systems exceptions

System	Exception
Serious Game GUI	Invalid format of the data provided in the request
Nexus Integration	Exception inside the SDM script
SIM4NEXUS Database	Exception caused by the Database specification

Component Tests can be run with different purposes.

- **Running Global System Tests along with Component Tests:** Perform a more exhaustive analysis over the Serious Game system.
- **Running only Global System Tests:** Perform a fast check over the overall application. This type of testing procedure is faster than the one presented before but is less accurate. It can be used as a guideline while adding new subsystems to the application.
- **Running only Component Tests:** When a specific Component Test is being executed without the rest, it is because the input of it is going to be a Global System which is failing. This type of execution will be a subtype of point 2 when outputs a failing Global System Test.

3.2.1 Coordination Module Test

Coordination Module Test checks different steps of the Serious Game instance creation and simulation processes, taking as an important input the Global System which is being analysed. Depending on the System, this test will reach different conclusions when an exception is thrown.

The main procedure involves the test of each component which conforms the Knowledge Elicitation Engine. To this end, the simplest way is to initialize and run a Game Session. By doing this every system is put under control.

The main difference with the existent procedure seen in subsection 1.1.1. is related to the provided request. By adding the parameter "Component-check" to "True", the KEE is going to perform a component-based run. This run procedure takes the one executed by the initialization and simulation steps, and divides them into components depending on the action performed at every time. The common procedure of these steps will be:

1. Parse Serious Game GUI HTTPS request content
2. Validate Serious Game GUI HTTPS request content
3. Validate registered user's credentials
4. (Initialization) Simulate the initial step / (Simulation) Simulate next step
5. Retrieve PolicyCards, PolicyObjectives, PolicyGoals and NexusHealth components based on the Case Study
6. Perform Game data persistence
7. Build the response
8. Return the response

Almost every step involves a component to be tested. By enabling the "component-check" mode, these steps will run inside test units which will take the result of the operation. Based on the validity of this result, the test will keep running or an exception will be thrown pointing out the exact component which produced it.

The classification of the previous steps into components will result as:

Table 14: Coordination Module Test steps

Coordination Module (Web Service API)	Login System	SDM Manager Module	Coordination Module (Web Service API)
1,2	3	4	7,8

In which refers to steps 5 and 6, these will be part of the Data Access Module. Both steps are included inside the Data Access Module, managed by the Database Manager System. The KEE can know the overall result of the operations but not the cause of them (because of privacy). Otherwise, we would be able to indirectly debug the database. Because of this, steps 5 and 6 will form part of a specific Component Test only accessible by the Database Manager System.

Requirements:

This test can be run by executing the Component Testing unit (S4N_Component_TestingUnit.py script) with the following arguments:

Table 15: Coordination Module Test arguments

argv0	argv1	argv2
Configuration file name without the extension	List of the Case Studies to be tested with the format:	Enable Coordination Module Test (N/Y) with the format:
	[number,number, …] [2,3]	"N", "Y" (without "") Y

By providing an affirmative character "Y" to the test, the already introduced HTTPS request parameter "component-check" will be set to True. For this Component Testing Unit, providing an "N" as the third character will immediately turn off the test. As these tests are intended to be run after running Global Tests. It makes no sense running them without the testing functionality off. The parameter is required to know if the programmer, who is running the program, does not run it by error as it takes much more time and resources than a normal Serious Game computational process due to the components checks.

Functionality:

As it was previously introduced, to run this specific test it is required to provide a new parameter to the HTTPS request (component-check). Thanks to it, the given outputs will provide information about how the test is performing and at which step has ended. By reaching a response containing "step 8" as the final step, the test will be considered as successful.

It is important to notice that this debug process will omit the previously introduced steps 5 and 6. To do so, it is going to test the process using a dummy version of the Case Study SDM inside the Nexus Integration system. This version does not differ to the one seen in subsections 1.1.2. and 1.1.3. The main difference is the extra layer of Component Testing running under the normal flux seen in those subsections. Thanks to this layer, the system will correlate an error with a System and a component without any special effort.

The parameters of the HTTPS request would have the following format:

Table 16: Initialization Step request parameters

Service	Request	Version	Identifier	Datainputs	Component-check
WPS	Execute	1.0.0	sim4nexus_initialization_step	<payload>	<True/False>

In which refers to the "datainputs" parameter, its content will be the one that has already been introduced in previous deliverables (see Deliverable 3.8, subsection 4.2.1, Tables 12 and 13). This presented request will require the same parameters for both possible scenarios. The first scenario, the initialization of a new Serious Game instance (Initialization Step). The second possible scenario, the progression and simulation of one more turn inside an existing instance of a Serious Game (Simulation Step).

An additional important feature of this system is the obligatory usage of the "player_id" parameter located inside the payload. This parameter is required as the system will never run in debug mode if the user, who is asking for this feature, is not an administrator. Otherwise, this will suppose an information leak as players would be able to test which components of the game are more sensitive to changes.

When the parameter "player_id" is provided, the flux of execution will take this id as well as the cookies "access_token_cookie" and "refresh_token_cookie" from the request. On its next step, it will retrieve the user's email from those cookies. If the cookies are missing or the email does not match the provided "user_id", the execution will terminate there. Another possible scenario would be a "user_id" which matches the cookies information but this user is not an administrator. The result will end up being the same as in the previous scenario, the execution will end right there.

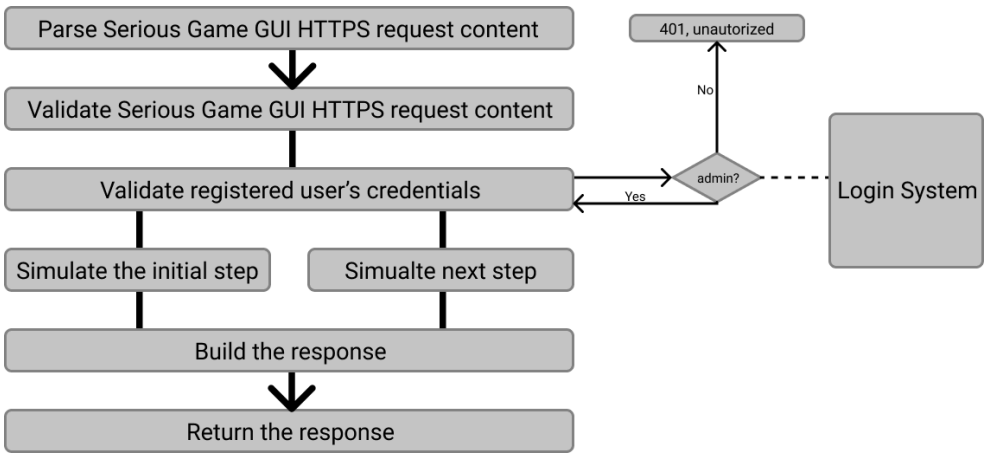


Figure 13: Credentials System implication in general execution flux

As it can be appreciated in Figure 13, the credentials validation takes place after the steps 1 and 2. For those cases, notifying any user about why the system failed would not be a data leak problem. The reason is because of this fail reports will only have information about the request they provided and which fields format does not match the desired format KEE uses.

Therefore, not only would it be safe, but it would also provide information to the user as to why its format is wrong.

Step	Name	Outcome
1	Parse Serious Game GUI HTTPS request content	Response: [200], Expected behaviour, the provided payload of the HTTPS request matches the JSON format. Response: [not 200, step 1] Unexpected behaviour, the provided payload of the HTTPS is not a JSON. User must provide a valid JSON payload.
2	Validate Serious Game GUI HTTPS request content	Response: [200], Expected behaviour, the provided JSON matches the required fields keys and values types. Response: [not 200, step 2] Expected behaviour, the provided JSON is missing some required keys or the provided values does not meet the desired data types. User must provide a valid JSON payload.
3	Validate registered user's credentials	Response: [200], Expected behaviour, the User who has made the request is a valid administrator of the Serious Game. Response: [not 200, step 3] Unexpected behaviour, the User who has made the request is not allowed to perform it.
4	(Initialization) Simulate the initial step / (Simulation) Simualte next step	Response: [200], Expected behaviour, the Nexus Integration System performs as is expected. Response: [not 200, step 4] Unexpected behaviour, there is a problem with the Nexus Integration System. Most surely linked to the SDM python version and its behaviour.
5	Build the response	Response: [200],

		Expected behaviour, after getting the data from the Nexus Integration process, all the required fields are present and match the desired format. Response: [not 200, step 5] Unexpected behaviour, there is a problem with the data provided by the SDM. The script might run ok but the output format does not match the desired one.
6	Return the response	Response: [200], Expected behaviour, the HTTPS response has been successfully sent. Response: [not 200, step 6] Unexpected behaviour, there is an error with the communications system of the KEE or the User has disconnected previous to get the response.

3.2.2 Data Access Module Test

Referring to the isolated part of the Component Testing, this one will perform a similar test to the one seen in subsection 1.2.3, but with a completely different scope. Inside the Data Access Module test, the focus is to know if the data was successfully and correctly stored inside the database.

Since this test is located inside the same script (S4N_Component_TestingUnit.py) it allows running both Component Tests, one after the other, or isolated one from the other. By only running the Data Access Module test means that the Coordination Module test has already been tested and ends with a successful result. If this previous test is ignored, it will result in an ambiguous outcome.

To check its goal, Data Access Module test takes as a premise that the Coordination Module is already working. Thanks to this supposition, the testing script will perform a game initialization for the specified Case Study. If during initialization the program exits with an unexpected event, it can be ensured that the problem is inside the database.

After getting a positive or negative response from the initialization process, the most precise part of the test will take place. Taking the entire game session as a response, thanks to this GameSession object, the Component Testing can look for irregularities among the data. If everything is considered correct, the test will end up with a successful state.

Requirements:

Sice this Component Test unit shares the same script as the previous one, the execution arguments provided to the script are going to inherit from the first one.

Table 17: Data Access Module test arguments

argv0	argv1	argv2	argv3
Configuration file name without the extension.	List of the Case Studies to be tested with the format:	Enable Coordination Module Test (N/Y) with the format:	Enable Data Access Module Test (N/Y) with the format:
	[number,number, …] [2,3]	"N", "Y" (without "") Y	"N", "Y" (without "") Y

Even inheriting from the execution seen in subsection 1.2.1. some changes can be appreciated. By attaching a third argument "argv3", the S4N_Component_TestingUnit script will replace the default False value. This value was disabling the Data Access Module Script

until now. To enable it, it is required to attach a positive character "Y" on this third argument when running the script.

Functionality:

Following Figure 10, the flux of this Component Test can be explained in much more detail. Once the process of creating and storing an instance of a Serious game is done, the Data Access Module test can take place.

This test is going to retrieve a GameSession object by providing the id used to create the Serious Game instance. The necessary HTTPS request will match the format:

Table 18: findBySessionId endpoint specification

Get Game by SessionId		
KEE url	/games/findBySessionId	
Parameters	Name	Value
	id	(String) 2a687a30-e565-de93-49b9

Response	<pre>{
	"case_study_id": "4",
	"id": 166,
	"score": 712685706.8152567,
	"session_id": "2a687a30-e565-de93-49b9-454061704b94",
	"start_date_time": "Wed, 03 Jun 2020 13:13:57 GMT",
	"state_evolutions": [
	{
	"applied_policies": [
	{
	"policy": "1",
	"region": 1,
	"session_id": "2a687a30-e565-de93-49b9",
	"turn": 1
	}
	],
	"game_id": 166,
	"id": 122,
	"turn": 1
	},
	{ … }
	],
	"status": "finished",
	"user_id": 0
	}</pre>	

Once the GameSession object is successfully retrieved, the validation process can take place (see comparison process, "Yes" output on Figure 14). This process is going to take all the parameters locally stored inside the script and compare each of them with the GameSession Object. If the store process has been done correctly, every single comparison should be equal, not only in value, but also in data types.

On the other hand, (see comparison process, "No" output on Figure 14), if the request to retrieve a GameSession Object returns 404 (Resource not found) or 400 (Internal error), this fill point out a storage problem. The causes can be due to data type mismatch or Database tables inconsistencies.

As a result, the Component Test is going to prompt out an Error and the specific location of it (Database models) or even the specific field of the table which is wrong (Validate Game Instance error outputs). If no error is obtained, the entire test procedure will be considered successful.

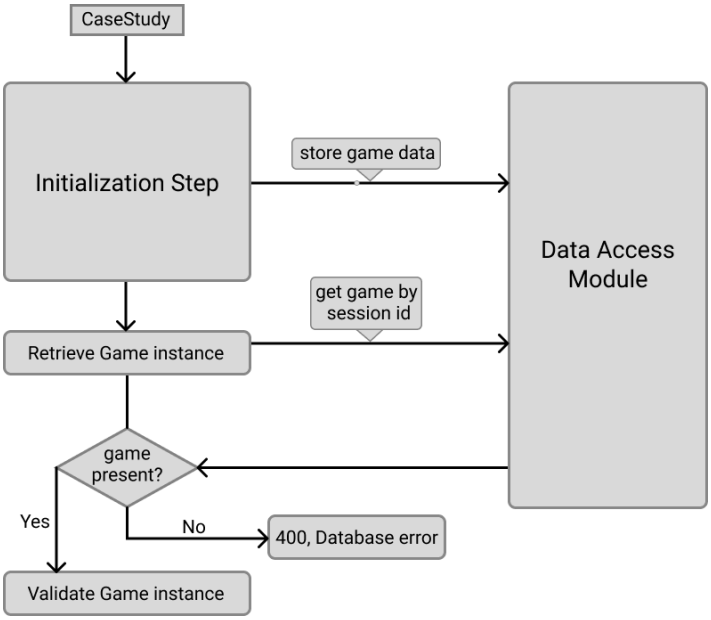


Figure 14: Data Access Module Test execution flux

Table 19: Data Access Module Test outcomes

Step	Name	Outcome
1	Initialization Step	Response: [200], Expected behaviour, the Initialization Step shouldn't fail at this point. Response: [not 200, step 1] Unexpected behaviour, if this system fails, the reason is because previous component Test wasn't run with a successful ending state.
2	Retrieve Game Instance	Response: [200], Expected behaviour, the Game Session object with the provided game_session_id exists and can be returned. Response: [404, step 2] Unexpected behaviour, the object was never stored inside the database. This can be because of a format issue or a bad definition of the Database tables. Response: [400 or not 200, step 2] Unexpected behaviour, while trying to retrieve the object the Database throws an internal exception. The most possible scenario is a value stored with a conflictive data type format. The recommended procedure is to delete the game instance to avoid more exceptions on the Database side.

3.3 Load testing

Referring to the Serious Game, testing the workload is directly linked to the number of concurrent players the system can hold. In terms of activity, users can be comprehended between different actions. Each one of these actions needs to be taken into considerations and compared with the rest to proportionate information about the stress which can hold the server-side.

Table 20: Serious Game user actions workload analysis

User action	Engine Workload	Max. repetitions/user
Initialize a new Serious Game instance	High	1
Simulate the next turn of a scenario	Medium	6
Interact with policies	None	Multiple
Idle	None	Multiple

During a game session, the user activity will be inside one of the previous categories. Not all of these categories are important when performing the load testing but it's good to know that there will be low activity points as well as high ones. The overall user interaction is unpredictable. Because of that, the best way to test is falling into the scenario known as "Worst Case Scenario". Inside of it, the variables become forced to its extreme to prove the system at high specifications.

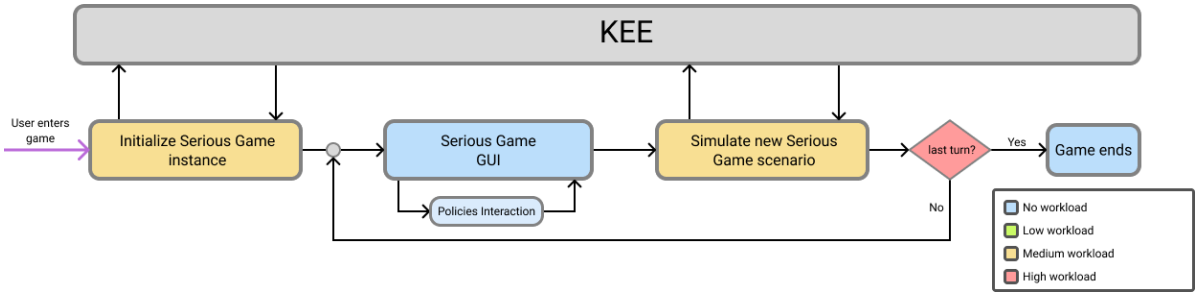


Figure 15: Serious Game GUI flux of interaction with workload heatmap

As it can be appreciated in the previous Table, as well as inside Figure 15, the only sections which accept parameters are the Initialization of the Serious Game and the Simulation of the next step inside the scenario. Taking into account the Worst Case Scenario and the maximum value those actions can take per user, it ends up into considering all the user interactions as an entire game being played from turn 1 to turn 6.

On top of the parametrization process of the Serious Game there is the CaseStudy. It is hard to specify a perfect CaseStudy to run the Load Tests. As a solution, the script is going to run these tests against three different scenarios (Greece, Azerbaijan and Latvia). Each scenario represents a different amount of workload for the server-side (in order, high, low and medium).

The main concern is to reproduce the interaction per user. This interaction is going to be capable to generate different amounts of workload on the KEE side.

The WorkLoad_test.py script is going to take those groups of actions as a template and simulate the access of multiple concurrent users (see **Figure** 16). As a more specific description, the script is going to take the already introduced basic User interaction, seen in **Figure** 15, and introduce it inside a function. Once the function is defined, the program will create as many threads as the test required. Every single one of these threads is going to run concurrently forcing the KEE under a Load Test.

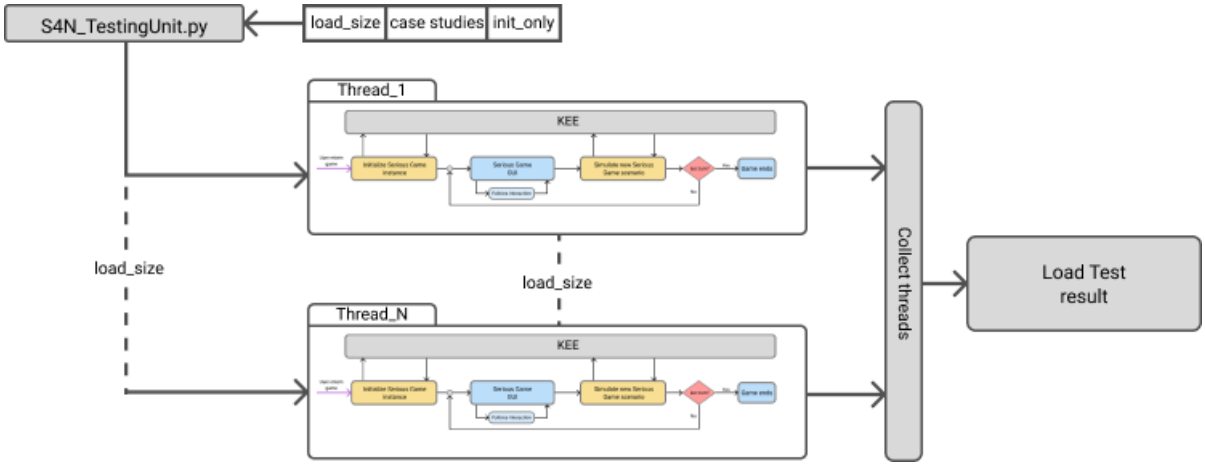


Figure 16: Load Test flux of execution

Requirements:

Depending on the arguments set, the test can be focused on different aspects of the interaction.

Table 21: Load Test arguments

argv0	argv1	argv2	argv3
Script name without the extension.	Amount of users (default: 40, max: 0)	List of the Case Studies to be tested (default: the most complex)	Run only initialization steps (default: No, "N")
	number 40	[number, number, …] [2,3]	"N", "Y" (without "") Y

Functionality:

When it comes to the procedures which can be performed taking this already presented script. There are two main procedures possible to perform.

Table 22: Initialization Load Test specification

Initialization Load Test			
Parameters	**argv1**	**argv2**	**argv3**
	50	3	N
Aproximate time/session	0.9159 seconds		
Possible outcomes	**Step**	**Outcome**	
	Build initialization HTTPS payload	**[No exception]:** Expected behaviour, the arguments provided match the valid parameters of the Serious Game. **[Invalid data Exception]:** Unexpected behaviour, some of the arguments provided are not available in the current Serious Game version.	
	Request initialization step	**[No exception]:** Expected behaviour, the URL built matches the format established by the initialization step of the KEE. **[Error 400]:** Unexpected behaviour, the URL provided does not match the current KEE format. Load testing script might be outdated or invalid data was provided.	
	Receive initialization step	**[No exception]:** Expected behaviour, HTTPS response provides information about the new Serious Game instance. **[Error 500 or no data on response]:** Unexpected behaviour, the Initialization step has failed due to database workload (error 500) or due to the maintenance process.	
	End and collect threads	**[No exception]:** Expected behaviour, the arguments provided match the valid parameters of the Serious Game. **[Thread Exception]:** Unexpected behaviour, some threads still expect a response from the KEE and the timeout has been triggered. Slow internet connection or database saturated.	

Output example	

```
User: 84, Thread PID: 13860, HTTPS CODE: 200 OK
User: 85, Thread PID: 4164, HTTPS CODE: 200 OK
User: 86, Thread PID: 10912, HTTPS CODE: 200 OK
User: 87, Thread PID: 15936, HTTPS CODE: 200 OK
User: 88, Thread PID: 7916, HTTPS CODE: 200 OK
User: 89, Thread PID: 13696, HTTPS CODE: 200 OK
User: 90, Thread PID: 11364, HTTPS CODE: 200 OK
User: 91, Thread PID: 12892, HTTPS CODE: 200 OK
User: 92, Thread PID: 13584, HTTPS CODE: 200 OK
User: 93, Thread PID: 1224, HTTPS CODE: 200 OK
User: 94, Thread PID: 8180, HTTPS CODE: 200 OK
User: 95, Thread PID: 9808, HTTPS CODE: 200 OK
User: 96, Thread PID: 844, HTTPS CODE: 200 OK
User: 97, Thread PID: 6012, HTTPS CODE: 200 OK
User: 98, Thread PID: 8472, HTTPS CODE: 200 OK
User: 99, Thread PID: 1188, HTTPS CODE: 200 OK
User: 100, Thread PID: 3372, HTTPS CODE: 200 OK
```

Table 23: Full Load Test specification

<table>
<tr><td colspan="4">Full Load Test</td></tr>
<tr><td rowspan="2">Parameters</td><td>argv1</td><td>argv2</td><td>argv3</td></tr>
<tr><td>50</td><td>3</td><td>Y</td></tr>
<tr><td>Aproximate time</td><td colspan="3">0.9159 seconds</td></tr>
<tr><td rowspan="5">Possible outcomes</td><td>Step</td><td colspan="2">Outcome</td></tr>
<tr><td colspan="3">Steps 1 to 3 from Table 22 Simulation Test</td></tr>
<tr><td>Build simulation HTTPS payload</td><td colspan="2">[No exception]:
Expected behaviour, the arguments provided match the valid parameters of the Serious Game.

[Invalid data Exception]:
Unexpeected bahaviour, some of the arguments provided are not available in the current Serious Game version. Invalid data might be provided.</td></tr>
<tr><td>Request simulation step</td><td colspan="2">[No exception]:
Expected behaviour, the URL built matches the format stablished by the simulation step of KEE.

[Error 400]:
Unexpeected bahaviour, the URL provided does not match the current KEE format. Load testing script might be outdated or invalid data was provided.</td></tr>
<tr><td>Receive simulation step</td><td colspan="2">[No exception]:
Expected behaviour, HTTPS response provides the information about the following turn of Serious Game instance.

[Error 500 or no data on response]:
Unexpeected bahaviour, the Initialization step has failed due to database workload (error 500) or due to mantinance process.</td></tr>
<tr><td>End and collect threads</td><td colspan="2">[No exception]:
Expected behaviour, the arguments provided match the valid parameters of the Serious Game.

[Thread Exception]:
Unexpeected bahaviour, some threads still expect a response from the KEE and the timeout has been triggered. Slow internet connection or database saturated.</td></tr>
</table>

Output example	User: 1, Thread PID: 8820, HTTPS CODE: 200 OK User: 2, Thread PID: 5312, HTTPS CODE: 200 OK User: 3, Thread PID: 2704, HTTPS CODE: 200 OK User: 1, Turn: 1/6, Thread PID: 8820, HTTPS CODE: 200 OK User: 2, Turn: 1/6, Thread PID: 5312, HTTPS CODE: 200 OK User: 4, Thread PID: 6544, HTTPS CODE: 200 OK User: 1, Turn: 2/6, Thread PID: 8820, HTTPS CODE: 200 OK User: 5, Thread PID: 1868, HTTPS CODE: 200 OK User: 3, Turn: 1/6, Thread PID: 2704, HTTPS CODE: 200 OK User: 4, Turn: 1/6, Thread PID: 6544, HTTPS CODE: 200 OK User: 5, Turn: 1/6, Thread PID: 1868, HTTPS CODE: 200 OK User: 3, Turn: 2/6, Thread PID: 2704, HTTPS CODE: 200 OK User: 8, Thread PID: 18408, HTTPS CODE: 200 OK User: 7, Thread PID: 18456, HTTPS CODE: 200 OK User: 9, Thread PID: 5356, HTTPS CODE: 200 OK User: 1, Turn: 3/6, Thread PID: 8820, HTTPS CODE: 200 OK User: 5, Turn: 2/6, Thread PID: 1868, HTTPS CODE: 200 OK User: 3, Turn: 3/6, Thread PID: 2704, HTTPS CODE: 200 OK User: 9, Turn: 1/6, Thread PID: 5356, HTTPS CODE: 200 OK

In terms of workload, the Case Study '3', shown as an example, has the lowest workload the Serious Game is able to generate. As it was presented on the introduction of this section, to test different amounts of workloads two more Case Study were added to the test. Each test output can be seen down below.

The following load tests have been conducted against the PRE environment, which is an identical copy of PRO. Based on the S4N development server, which has been detailed in section 2.2, the Apache server is configured accordingly to the HW resources. The main configurations, which are directly related to these tests, are the following:

- Processes: (9) The number of worker processes (instances of the WSGI application) to be started up and which will handle requests concurrently.
- Threads (2) The number of threads in the request thread pool of each process for handling requests.
- max-clients: (18 = 9*2) The maximum number of simultaneous client connections that will be accepted. This will default to being 1.5 times the total number of threads in the request thread pools across all process handling requests.

Full Load Test			
Parameters	**argv1**	**argv2**	**argv3**
	50	2 (Greece)	N
Average time/initialization	50s		

The average time for an isolated initialization step for the Greek CS is 5.5s.

Some of the requests have been denied due to the time spend to process them and the server configuration.

Full Load Test			
Parameters	**argv1**	**argv2**	**argv3**
	50	3 (Azerbaijan)	N
Average time/initialization	0.99s		

The average time for an isolated initialization step for the Azerbaijan CS is 0.1s (note the difference in complexity against the Greek CS).

Full Load Test			
Parameters	**argv1**	**argv2**	**argv3**
	50	4 (Latvia)	N
Average time/initialization	9s		

The average time for an isolated initialization step for the Latvia CS is 1.5s.

The Workload Script also allows to run multiple CaseStudies performing a random choice of each one to set the Thread Case Study. This option can be enabled thanks to the 4th argument given to the script. This procedure is going to extract the closest version of a real-life scenario where players play different games from different case studies in different stages.

Full Load Test				
Parameters	**argv1**	**argv2**	**argv3**	**argv4**
	50	2,3,4	N	N
Average time/initialization	8s			

The Apache server has been able to support and manage all the requests, queuing and processing them to finally return the corresponding response. If more performance is required, the server capacity must be increased to support it, but no further developments will be needed since the Apache server is able to adapt its behaviour by only modifying its configuration.

3.4 Browsers compatibility

Different web browsers react differently to the WebGL code that manages the visualisation of up to tens of thousands of elements on the screen. Due to limited resources, work has been done to ensure compatibility with two recent browsers that are available on all existing platforms: Chrome (version 83.0.4103.116 -Official Build) and Firefox (version 77.0.1). Other browsers such as Safari, or Edge would require additional work to ensure compatibility.

3.5 Screen resolutions

Overall, the User Interface was developed to be used from either a laptop or a desktop computer. As such the Serious Game does not accommodate tiny touch screens for portable devices such as smartphones or tablets. The user interface screen resolution is optimised for

the standard size 1920x1080 but can also cope with smaller or greater window sizes due to the pan/zoom capability integrated into the design.

See the picture below where browsers windows of different sizes can be open to zoom and pan on the same view.

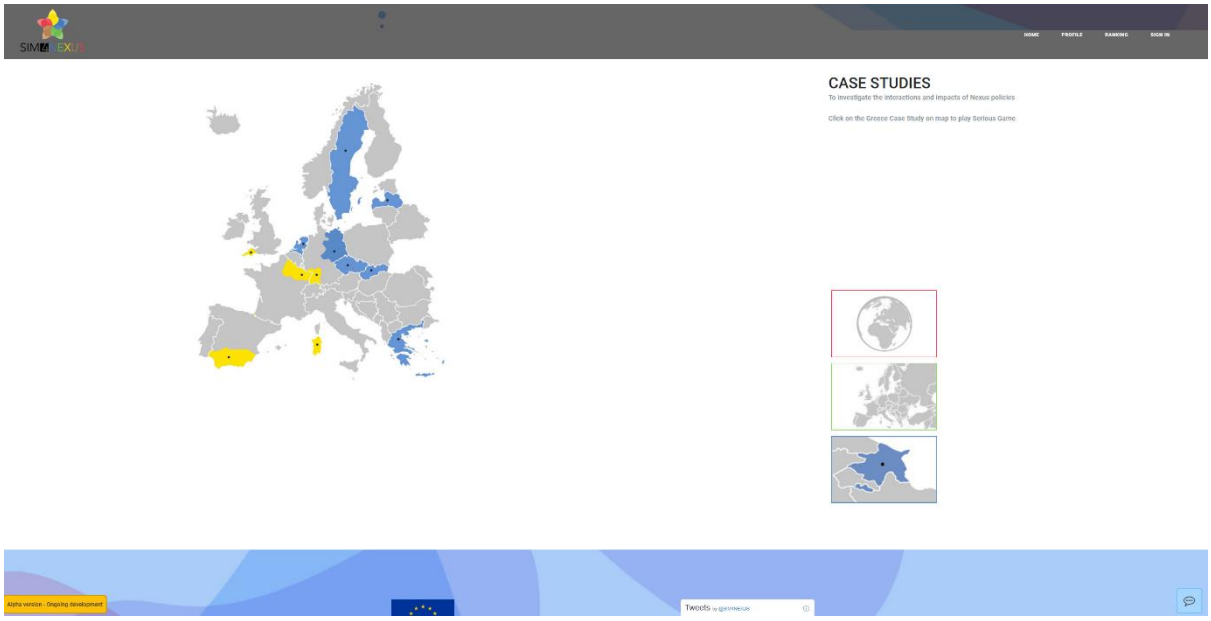


Figure 17. Map view

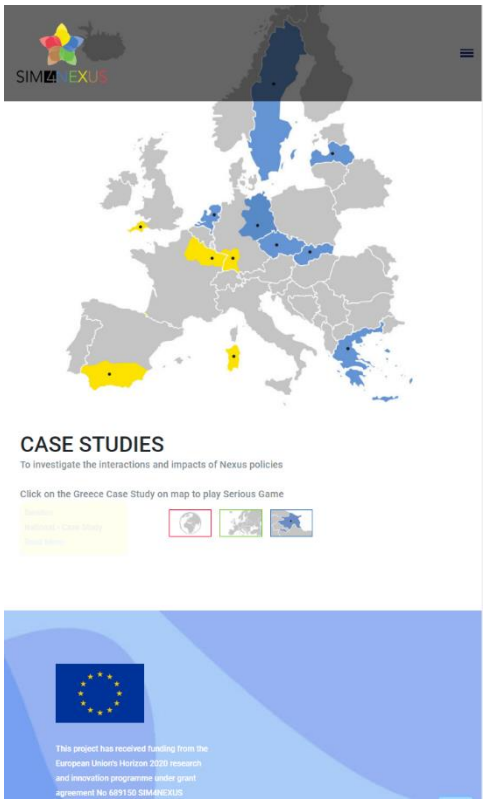


Figure 18. Map view

3.6 Security

3.6.1 Login System

From a user point of view, the login system provides authentication and authorization through a Json Web Tokens [8] (JWT) technology, which are an open, industry-standard RFC 7519[9] method for representing claims securely between two parties.

The JWT are established during the login process and shared during all the communications through the cookies.

Several tests have been developed to validate this system and different approaches have been followed to force it trying to break its security layer.

The login system relies on the Python Flask library and its extension for JWT. Thanks to it, all the tests have been successfully conducted.

3.6.2 HTTPS

At a different level, another important issue related to the security topic is the communication layer between the S4N clients and the S4N platform.

In this case, the communication is based on The Hypertext Transfer Protocol[10] (HTTP), which is an application protocol for distributed, collaborative, hypermedia information systems. In terms of security, the S4N platform relies on Hypertext Transfer Protocol Secure[11] (HTTPS), which is an extension of the HTTP protocol. It is used for secure communication over a computer network, and is widely used on the Internet. In HTTPS, the communication protocol is encrypted using Transport Layer Security (TLS) or, formerly, Secure Sockets Layer (SSL). The protocol is therefore also referred to as HTTP over TLS, or HTTP over SSL.

3.7 Resilience

As mentioned in D4.5, the KEE relies on the HTTP Apache Server[12] to provide a secure, efficient and extensible server that provides HTTP services in sync with the current HTTP standards. Thanks

---

[8] https://jwt.io/
[9] https://tools.ietf.org/html/rfc7519
[10] https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol
[11] https://en.wikipedia.org/wiki/HTTPS
[12] https://httpd.apache.org/

to the characteristics of this tool, together with the way the KEE has been designed and implemented, the KEE has the required capacities to isolate any problem and continue working while ensuring other basic characteristics such as availability, capacity, interoperability, performance, reliability, robustness, safety, security, and usability.

The KEE has been designed as a Web Service which is finally hosted by the Apache server. Once deployed, the Apache server distributes the incoming requests to different threads where KEE replicas are listening. This level of parallelization allows the first idea of resilience and, during the process of the request processing, the KEE has also different mechanisms (such as an exceptions system) to catch any problem and continue working normally.

A simple, but clear, proof of its resilience is all the tests performed to validate its functionalities. These tests finally succeed due to the KEE capacity to continue working in a normal way after a problem appears (in this case forced fails).

Other services such the S4N databases delegate its functionalities to well known and extensively validated tools like PostgreSQL[13] or Apache Jena Fuseki[14] which provides resilience by default.

3.8 Scalability

Thanks to the way the S4N SG platform has been designed and the tools used to host it, the different modules (UI, KEE and S4N databases) can be easily distributed along with different servers. This modularity is a key characteristic that allows independence between S4N services and finally results in a decoupled system.

Again, the way the KEE manages the data flows and the authentications procedures tolerates distributed processes to scale the solution.

In the same line as the previous section, the Apache server acts as the first layer of scalability to distribute the requests to KEE replicas. On the other hand, Docker infrastructure also provides the needed mechanisms to scale other services to different Docker nodes.

[13] https://www.postgresql.org/
[14] https://jena.apache.org/documentation/fuseki2/

As a proof, the load tests (section 3.3) show how the KEE can be escalated (through Apache server configuration) to provide the needed capacity to support an increased number of requests.

3.9 Interoperability

The interoperability capacity between the internal S4N SG platform modules has been validated during all the tests. Based on different communication standards, the isolated services are able to communicate between them in order to implement the desired data flows.

Externally, the Semantic Repository represents an advance through the publication of an open catalogue of Nexus information and their linkage with policy. We mean, the semantic repository offers the community the possibility to analyse openly the implications of a policy in certain topics (policy objectives) considering the Nexus. At the technology level, the main innovation with the semantic repository relies on data navigation using facets. That means this tool provides navigation through the information using the properties of the defined entities (navigation using the semantics of the information).

Combining these both aspects, the SR contributes to semantic interoperability of the water sector by offering a catalogue of variables under a common and standardised data model as it is SAREF4WATR. Due to the use of these standardised data models, the semantic repository offers common data exploration, accessibility and sharing.

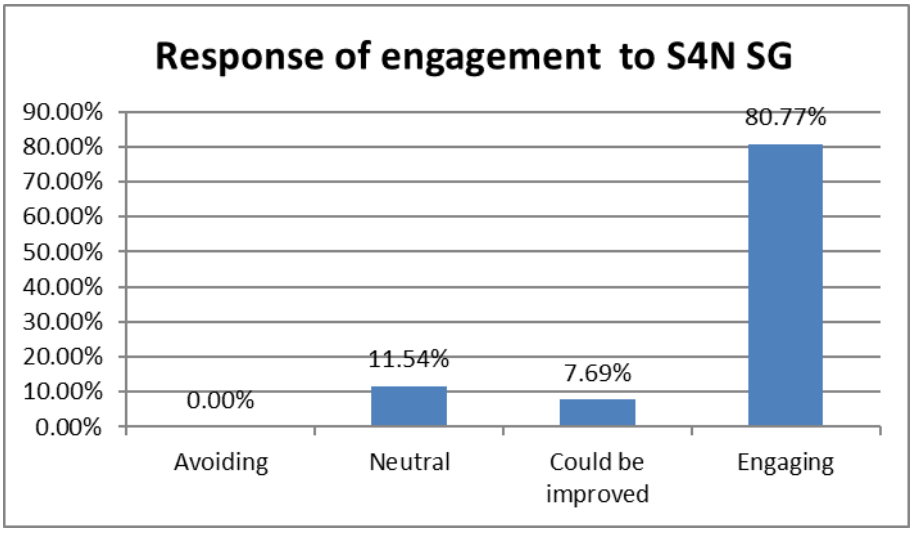
4. User feedback

Feedback was provided from two training sessions in Azerbaijan (6) and in Greece (5) having as participants members from the partners. Also, there were additional participants (26) of the Dresden Nexus Conference 2020 (https://www.acteon-environment.eu/en/actualites/key-message-dresde-nexus-2020/) in Germany.

Overall, the feedback from the end-users of SDM provided a positive attitude in the total spectrum of questions around the Serious Game and the operating framework.

Exceptional was the outcome of engagement with SDM where more than 80% of people found it engaging (see Fig.1), without having any negative response of avoiding it.

Figure 19: Response of engagement to the Serious Game

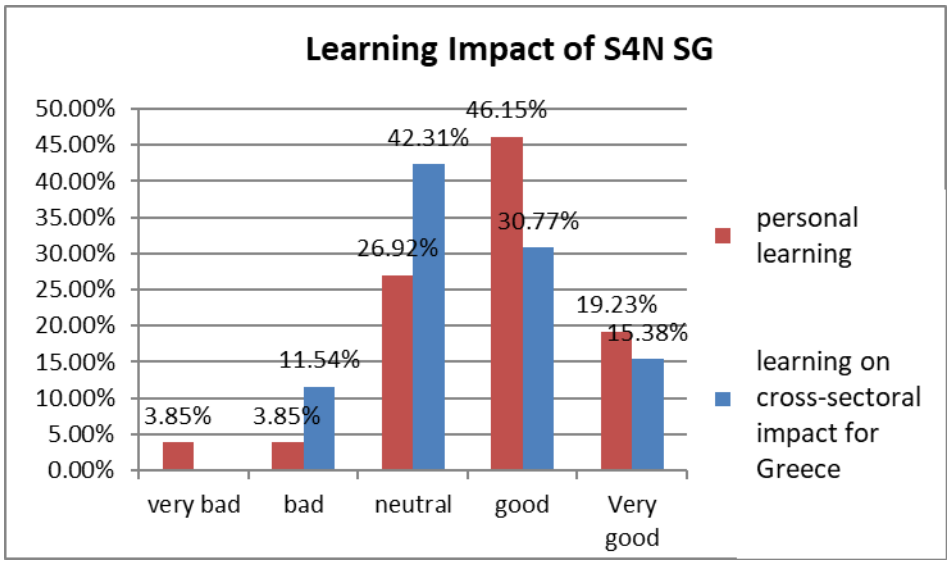


As regards the characteristics of the S4N SG to learning environments and its potential contribution to the specific needs and prerequisites of learning outcomes, there was a significant separation of results:

- Firstly, there was a mixed understanding from participants as regards the objectives of learning specifications. However,
- most of them where above average in understanding it and personal learning from the specific SG (cyan bars at Fig. 2); especially about the cross-sectoral impacts of water, energy, food, land and climate policies applied for the case study in Greece (blue bars on Fig. 2).
- a considerate number of participants had issues in understanding and learning from it standing in between, while there were some participants providing even negative response to the learning outcomes of the SG. Here, some proposals have been provided as potential measures of further learning capacity of SG. The response of the

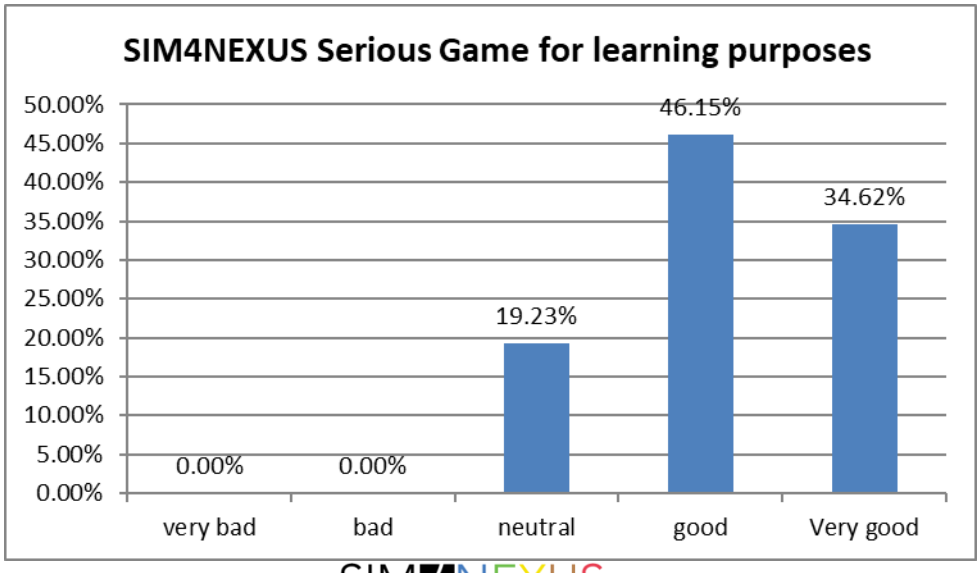
training sessions in Azerbaijan and Greece was divided: from Azerbaijan they have been satisfied with the overall performance and goals of the platform, asking a "pregame" as potential first base of "getting-to-know" the overall environment. The responders from Greece have shown some more moderated enthusiasm, understanding its capacity but finding it more complex in order to be played properly.

Figure 20: Response of learning impact of the S4N Serious Game



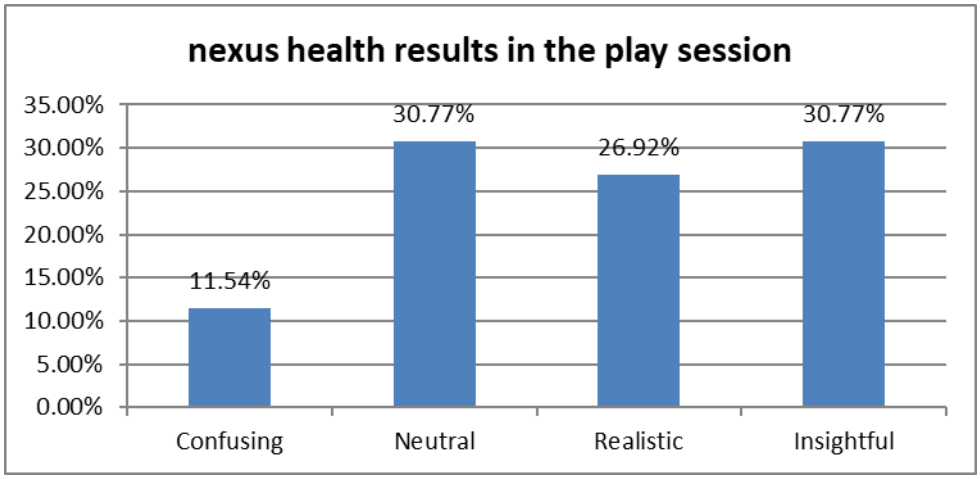
- Secondly, there was a complete, positive response to the use of S4N SG for learning purposes to students, stakeholders and/or policymakers (Fig. 3), showing the overall capacity of the SDM for learning purposes. Yet, there was a small percentage with neutral thinking on that, providing some proposals that have to be taken under consideration.

Figure 21: Response of potential of the use of S4N SG for learning purposes



- As regards the outcome of the nexus health results in the play session as a result of the policies implemented, there was a mixed sharing of answers (Fig. 4) and, at a certain point, there is a need to see more built-in explanation of game functionalities, cards, graphs, nexus health etc. The successful deployment of S4N SG and further dissemination of its use may stand upon the prerequisites that will erase issues that caused confusion to participants.

Figure 22: Response of potential of the use of S4N SG for learning purposes



No software issues were detected during the sessions, nor new functionalities were requested by the users.

Despite it was not identified by the final users, during the SG tests realized by S4N partners in the PRE environment, it was identified the possibility to define a new view in the SG website to show the results and decisions made by different players during their game sessions. Finally, this idea was implemented through the Ranking page (see Figure 23), where the top players that achieved the best nexus health scores for each CS are shown, followed by the leaderboard, a list of all the previous users' sessions with their respective score and policy decisions.

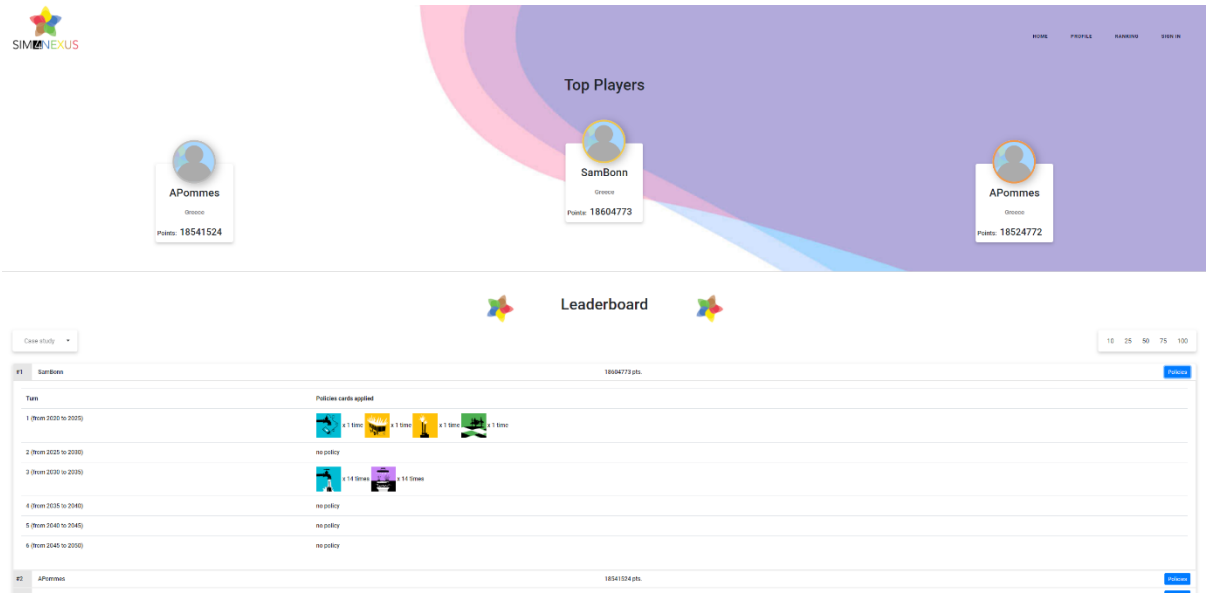


Figure 23. S4N SG Ranking Page

5. SIM4NEXUS - SPACE

5.1 Introduction

Aim has been to develop process, and test satellite derived datasets for selected SDM variables, and to evaluate the out produced for year 2019 following the schematic approach of (**Error! Reference source not found.**). The short effort considered:

- Satellite data products specification
- Data preparation
- Application of the SDM
- Result evaluation

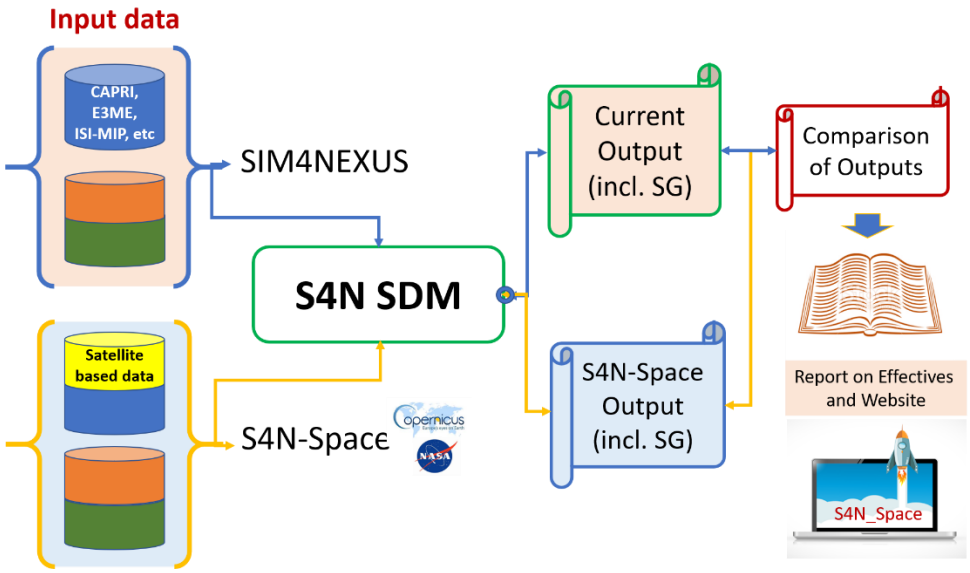


Figure 24: The S4N-Space Concept

5.2 Methodology

The methodology workflow is depicted in Figure 25:

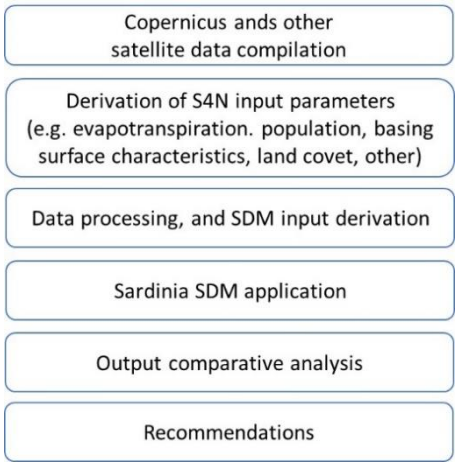


Figure 25: Methodology Workflow

5.3 Satellite data and products specification

Study of available satellite datasets revealed that several of the variables used by the SDM can be immediately replaced with satellite derived values (Table 24). Additional variables for the SDMs or other applications can also be replaced via correlations.

Table 24: SDM variables - satellite products correspondence

VARIABLE	SATELLITE PRODUCT
ET_OPEN_BODY (EVAPOTRANSPIRATION)	TOTAL MONTHLY ACTUAL EVAPOTRANSPIRATION (ET) USGS
POPULATION	GLOBAL HIGH-RESOLUTION POPULATION DENOMINATORS PROJECT
BASIN_SURFACE	WWF HYDROSHEDS
RICE_AREA	CORINE
PASTURE_AREA	>>
FRUIT_AREA	>>
OLIVE_AREA	>>
GRAPE_AREA	>>

5.3.1 Evapotranspiration

Evapotranspiration (ET) is the sum of water evaporation and transpiration from vegetation. The satellite data used to replace the standard evapotranspiration variable were derived from version 4 SSEBop Evapotranspiration product produced by the United States Geological Survey (USGS). The product consists of monthly images that represent evapotranspiration in millimeters worldwide, with pixel size 1 km. From the USGS website

https://earlywarning.usgs.gov/fews/product/460#:~:text=Monthly%20Period&text=Evapotranspiration%3A%20Evapotranspiration%20(ET)%20is,the%20period%202000%20to%20present. we obtain the info: "*Actual ET (ETa) is produced using the operational Simplified Surface Energy Balance (SSEBop) model (Senay, et al. 2013) for 2003 to present. The SSEBop setup is based on the Simplified Surface Energy Balance (SSEB) approach (Senay, et al. 2011) with unique parameterization for operational applications. It combines ET fractions generated from remotely sensed MODIS thermal imagery, acquired every 10 days, with reference ET using a thermal index approach. The unique feature of the SSEBop parameterization is that it uses pre-defined, seasonally dynamic, boundary conditions that are unique to each pixel for the "hot/dry" and "cold/wet" reference points. The original formulation of SSEB is based on the hot and cold pixel principles of SEBAL (Bastiaanssen, et al. 1998) and METRIC (Allen, et al. 2007) models*."

5.3.2 Population

This dataset is produced with info from WorldPop (www.worldpop.org, School of Geography and Environmental Science, University of Southampton; Department of Geography and Geosciences, University of Louisville; Departement de Geographie, Universite de Namur) and Center for International Earth Science Information Network (CIESIN), Columbia University (https://www.worldpop.org/geodata/summary?id=26939).

It is the estimated number of people per grid cell, with resolution of 3 arc (approximately 100m at the equator). The units are number of people per pixel with country totals adjusted to match the corresponding official United Nations population estimates that have been prepared by the Population Division of the Department of Economic and Social Affairs of the United Nations Secretariat (2019 Revision of World Population Prospects). The mapping approach is Random Forest-based dasymetric redistribution, which uses various satellite derived data products like land cover, roads, building maps, satellite night lights, vegetation, topography, etc. (https://www.worldpop.org/methods).

5.3.3 Basin surface

HydroSHEDS is a mapping product that provides hydrographic information for regional and global-scale applications in a consistent format (https://www.hydrosheds.org/). It offers a suite of geo-referenced data sets (vector & raster) at various scales, including river networks, watershed boundaries, drainage directions, and flow accumulations. HydroSHEDS is based on high-resolution elevation data obtained during a Space Shuttle flight for NASA's Shuttle Radar Topography Mission (SRTM). In the present case, the watershed boundaries were used to calculate basin area for the SDM.

5.3.4 Land cover

CORINE Land Cover (CLC) inventory is part of the Copernicus Land Monitoring Service. It is produced by the majority of countries by visual interpretation of high-resolution satellite imagery. In a few countries semi-automatic solutions are applied, using national in-situ data, satellite image processing, GIS integration and generalization (https://land.copernicus.eu/pan-european/corine-land-cover). The CLC product is provided in both vector and raster form. The raster version has a pixel size of 100m.

5.3.5 Additional satellite data

Sentinel-2 2A and Landsat-8 SR products were also used. Sentinel-2 carries the Multi Spectral Instrument (MSI) sensor, which has the ability to capture spectral information in the visible, the red-edge, the infrared, and the short-wave infrared part of the electromagnetic spectrum. The spatial resolution ranges from 10m to 60m depending on the spectral band. Landsat-8 uses the Operational Land Imager (OLI) which captures spectral information in the visible, the infrared, and the short-wave infrared part of the electromagnetic spectrum (but not red-edge). The spatial resolution is 30m, except on panchromatic and thermal bands, which are 15m and 100m, respectively. The detailed Large-Scale International Boundary Polygons dataset was used to extract the Sardinian coastline shapefile. This dataset is provided by the United States Office of the Geographer and can be accessed using this url: https://catalog.data.gov/dataset/global-lsib-polygons-detailed-2017dec29.

5.3.6 Data preparation

To ensure the satellite data products would be compatible input variables to the SDM, pre-processing steps were necessary.

CORINE

The CORINE dataset was downloaded and the area including Sardinia was clipped. The corresponding CORINE classes used to represent the SDM variables are shown in Table 25

Table 25: SDM variables – CORINE classes correspondence

SDM VARIABLES	CORINE CLASSES
RICE_AREA	RICE FIELDS
PASTURE_AREA	PASTURES
FRUIT_AREA	FRUIT TREES AND BERRY PLANTATIONS
OLIVE_AREA	OLIVE GROVES
GRAPE_AREA	VINEYARDS

The above CORINE classes were then extracted in separate shapefiles (Figure 26). These shapefiles represent the area of the selected classes for the year 2018. In order to calculate a similar land cover map for the year 2019, a classification algorithm was employed.

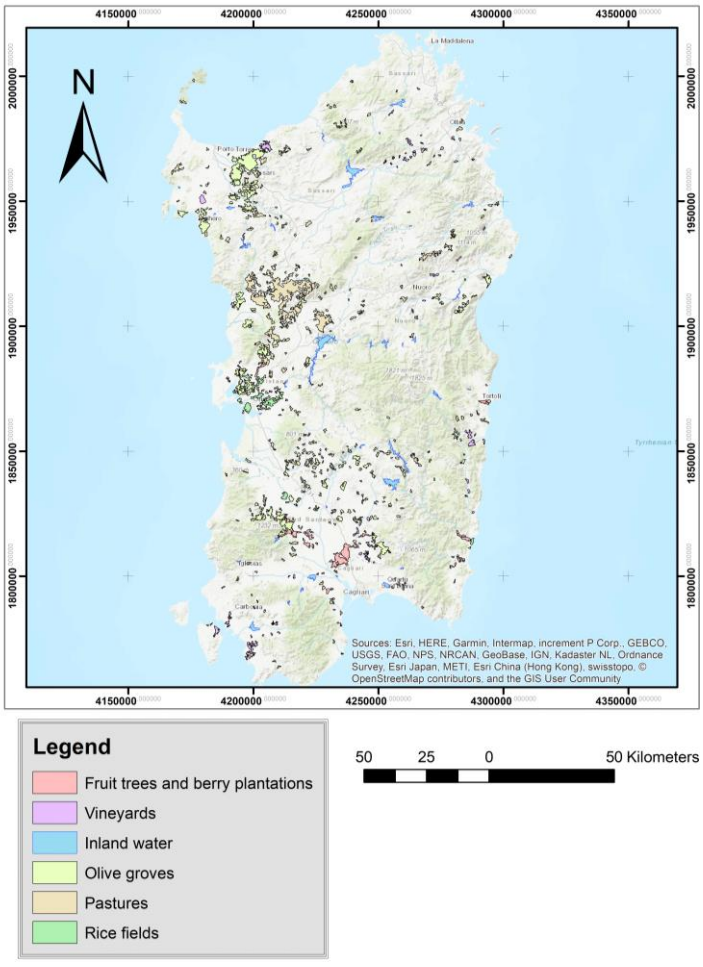


Figure 26: Map of Sardinia with the selected CORINE classes

Land cover mapping:

Popular classification algorithms for land cover mapping include Random Forests (RF), Support Vector Machines (SVM), and Neural Networks (NN). Two of these methods were tested for their classification accuracy (SVM and RF) and Random Forests were found to perform higher. Random forests is a popular classifier that has been extensively studied in land cover mapping in recent years (Gislason, et al. 2006; Waske and Braun 2009; Pelletier, et al. 2016). Random Forests is a machine learning algorithm, and as such, requires training data in order to be able to classify.

The training data were extracted from both Sentinel-2 and Landsat-8 using CORINE classes as labels and area indicators. Several spectral indices were calculated in order to improve the classification result, including NDVI (Normalized Difference Vegetation Index) (Rouse Jr, et al. 1974), NDWI (Normalized Difference Water Index) (Gao 1996), BSI (Bare Soil Index) (Jamalabad 2004), EVI (Enhanced Vegetation Index) (Huete, et al. 1999), and MCARI (Modified Chlorophyll Absorption in Reflectance Index – calculated only for Sentinel because of its red-edge band requirement) (Daughtry, et al. 2000). The classification algorithm was written in the Google Earth Engine's code editor (https://code.earthengine.google.com/) and the classification was performed using Google Earth Engine for six Sentinel-2 tiles and four Landsat-8 tiles needed to cover Sardinia. Figure 3 shows a snapshot of the process of classification and validation (Figure 27):

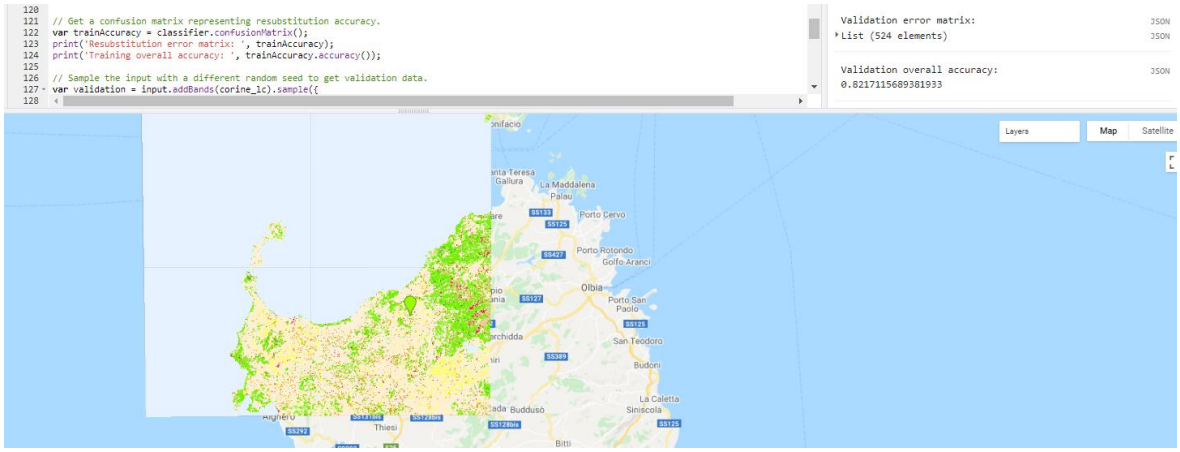


Figure 27: Snapshot of classification & validation carried out on Google Earth Engine

The validation was carried out by classifying a random selection of pixels from 2018 images and validating with the CORINE labels from 2018 (Table 26).

Table 26: Classification overall accuracy

DATA SOURCE	OVERALL ACCURACY
LANDSAT	69.82%

SENTINEL	75.83%

The results of both classifications were carefully studied, and the area for each predicted class was calculated (Table 27):

Table 27: Total area for each class calculated from the classified images

LANDSAT		SENTINEL	CORINE
SARDINIA 2019 PREDICTIONS		SARDINIA 2019 PREDICTIONS	SARDINIA 2018 GROUND TRUTH
LANDCOVER	TOTAL AREA (KM2)		
RICE FIELDS	0.945	0	50.2077
PASTURES	0	1.016	405.6913
FRUIT TREES	4.9257	257.366	100.0042
OLIVE GROVES	137.9385	3.0816	414.7706
VINEYARDS	4.9257	13.4708	90.6166

Although, the overall accuracy of the classification was satisfactory, the predicted areas of the selected classes had significant changes when compared to the ones derived from CORINE for the year 2018. Thus, it was deemed that the results were unreliable, and the high overall accuracy was attributed to high classification accuracy over water, which covers large parts of the images. Instead of the classified images, the original CORINE areas were used, assuming the actual changes in land cover from 2018 were less significant than those predicted by the classification algorithm.

Total Monthly Actual Evapotranspiration dataset

Twelve images, representing the monthly actual evapotranspiration of 2019 were downloaded. Each pixel value represented the evapotranspiration in millimeters (Figure 28).

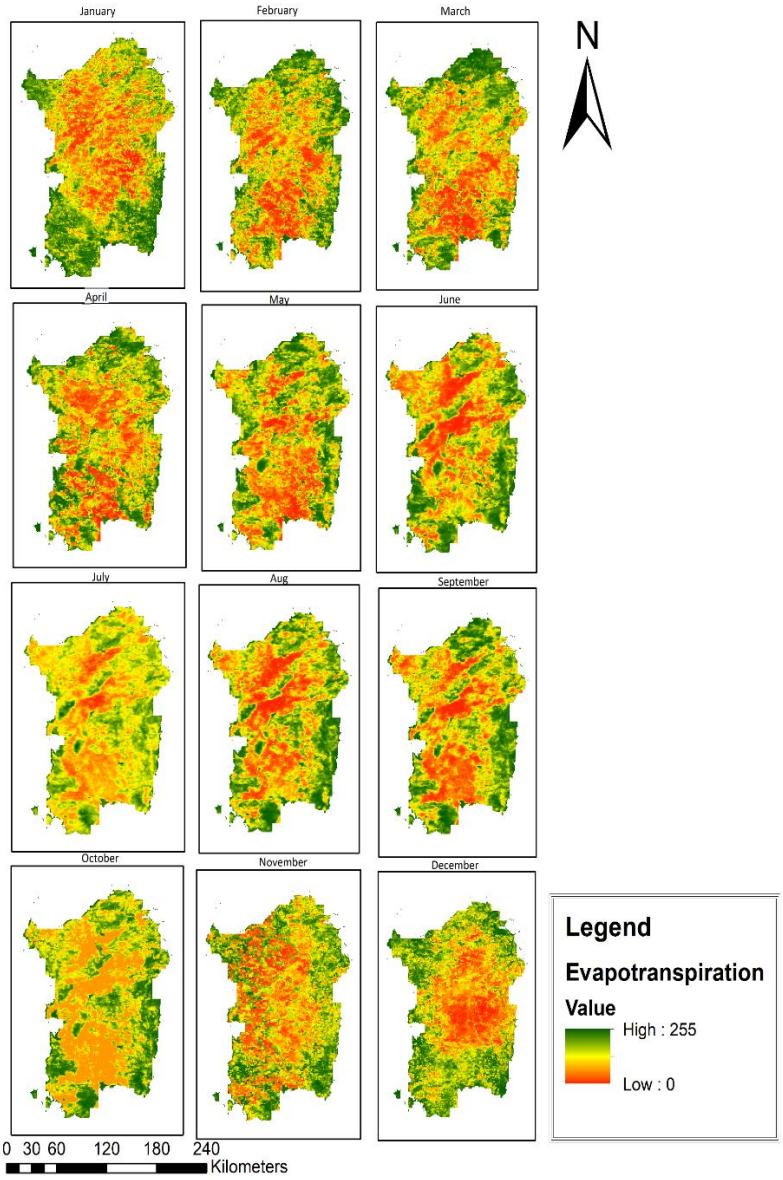


Figure 28: Monthly actual evapotranspiration maps for the year 2019

Since the SDM takes as input the evapotranspiration in open bodies of water, the intersection of monthly actual evapotranspiration and the inland water shapefile from CORINE was calculated and the pixel values of the resulting rasters were summed to the total evapotranspiration for each month of 2019.

WolrdPop dataset

The Italian population data for the year 2019 were downloaded and clipped using the Sardinian coast shapefile (Figure 29). The pixel values represented the population count and were summed to calculate the total Sardinian population.

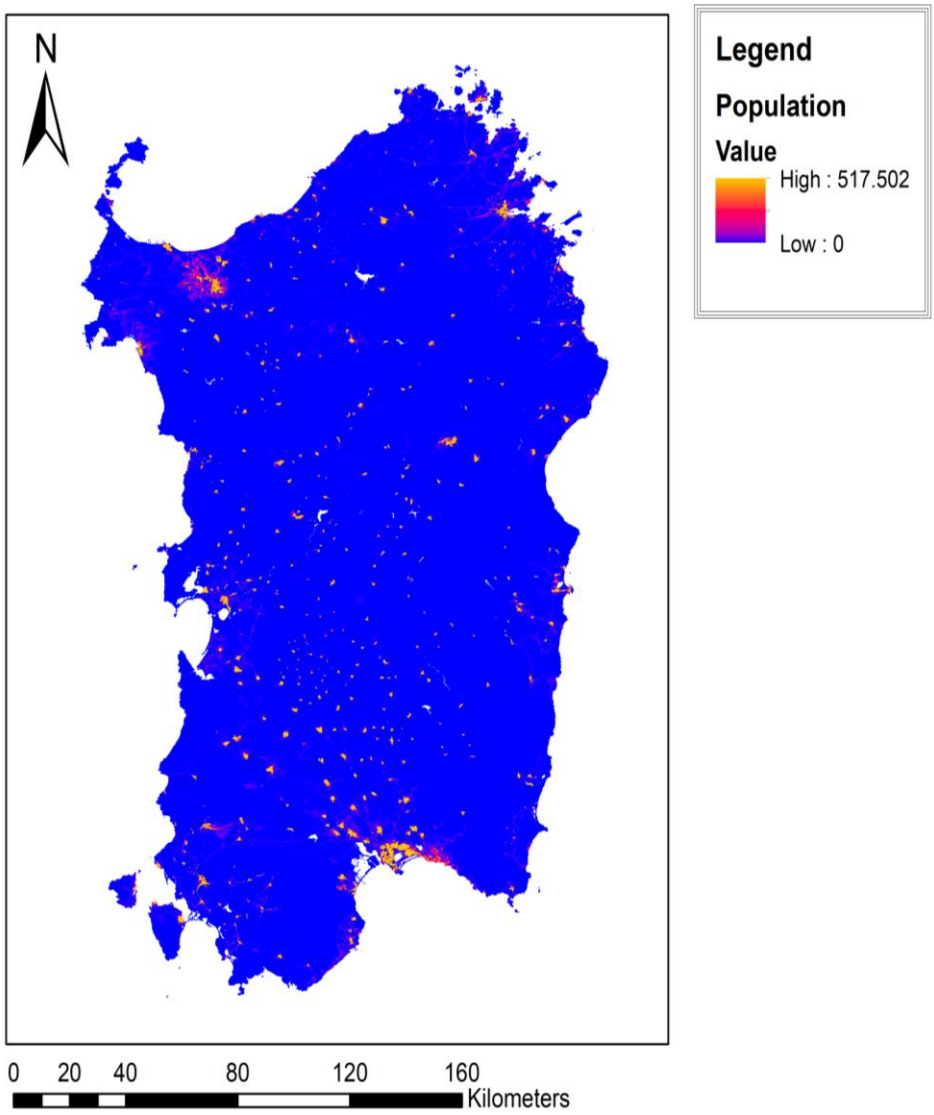


Figure 29: Population map of Sardinia for the year of 2019

HydroSHEDS

The SDM required the area of only the basins that contributed to the largest water reservoirs. By filtering the inland water shapefile from the CORINE dataset, a shapefile containing only bodies of inland water with above average area was created. The basins from the HydroSHEDS dataset that contained the aforementioned large bodies of water, were then selected and their total area was calculated (Figure 30).

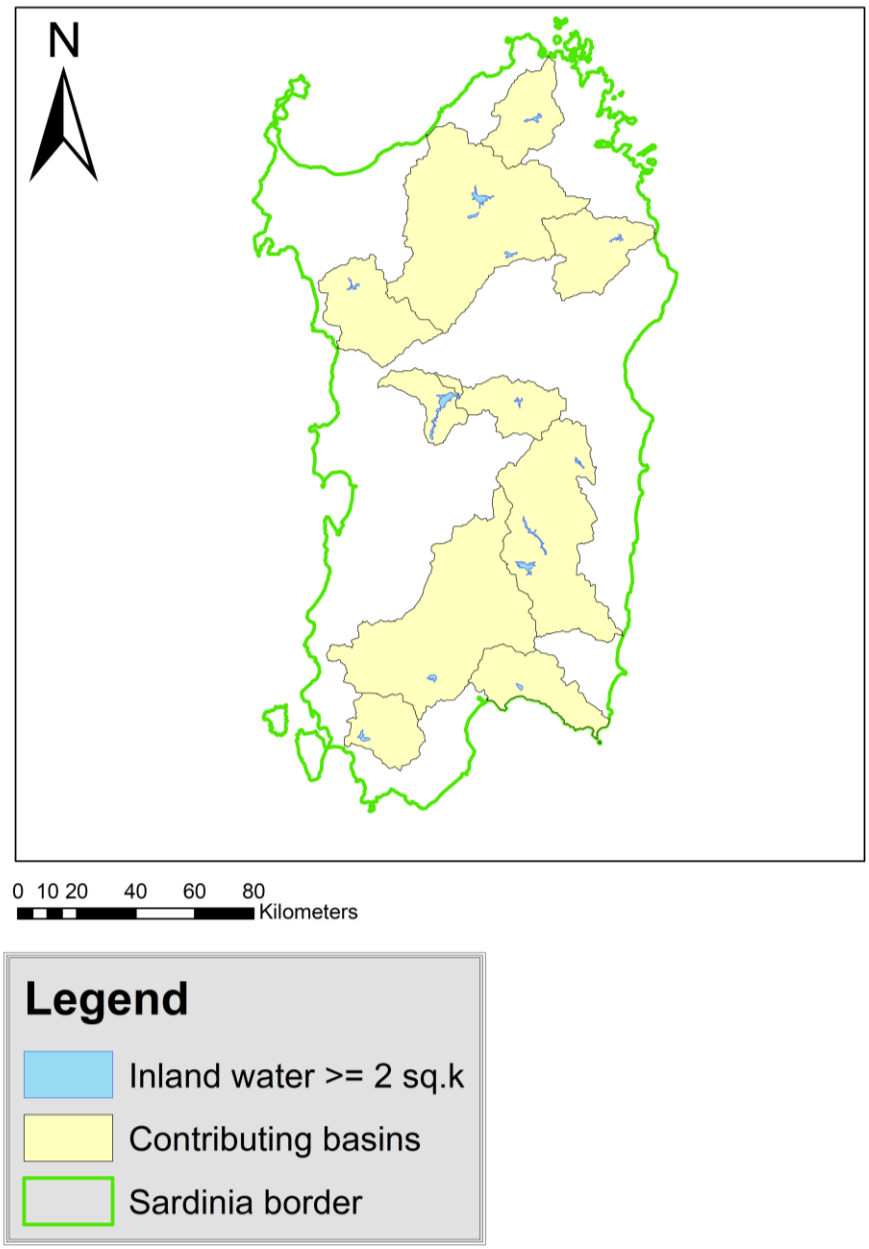


Figure 30: Basins contributing to the largest water bodies

5.4 Application of the SDM

The data derived from the satellite sources replaced the variables:

- ET_open_body
- Basin_surface
- Population
- Rice_area
- Pasture_area
- Olive_area
- Grape_area
- Fruit_area

The ET_open_body variable was inserted to the SDM with a CSV file, while the other variables were initialized inside the script. The Python script containing the SDM was executed once with the standard inputs and once with the new satellite derived inputs.

5.5 Results & Comments

The results produced are shown in Table 28, Figure 31 and Figure 32. It is apparent that:

- Results calculated with the "standard" inputs are quite similar to the results calculated with "satellite" derived inputs.
- The difference between the two results increases up to a point and then stays relatively stable (**Error! Reference source not found.**).

This was a short effort aimed to **demonstrate** the use Copernicus and other satellite data in the implementation of S4N. We believe however:

- The effort was worth because it proves that S4N can in the future take advantage of satellite data and in this context to produce applications faster and easier.
- Assuming an appreciable effort is devoted, it is more or less certain, that a large percentage of the S4N Input can be derived from satellite data for multiple SDMs.
- The above leads to conclusions that S4N Nexus can be re-designed, to accommodate and relax a user from cumbersome input, via a "preprocessor" to be based on satellite data and readily available databases. This can lead do an upgraded product that can be world-wide applied if redesigned to accommodate to a large extent satellite and readily available databases.
- The above leads the road to a "next effort" aimed to broaden the current S4N to a S4N-space.

Table 28: Reservoir stock calculated from standard and satellite derived inputs

	STANDARD INPUTS	SATELLITE DERIVED INPUTS
T	RESERVOIR	
0	2000	2000
1	2093.918	2064.891
2	2103.674	2030.855
3	2097.636	1951.012
4	2075.567	1823.877
5	1895.131	1532.872
6	1613.884	1155.646
7	1331.442	804.4948
8	1183.179	620.1981
9	1129.173	552.6637
10	1134.255	551.2428

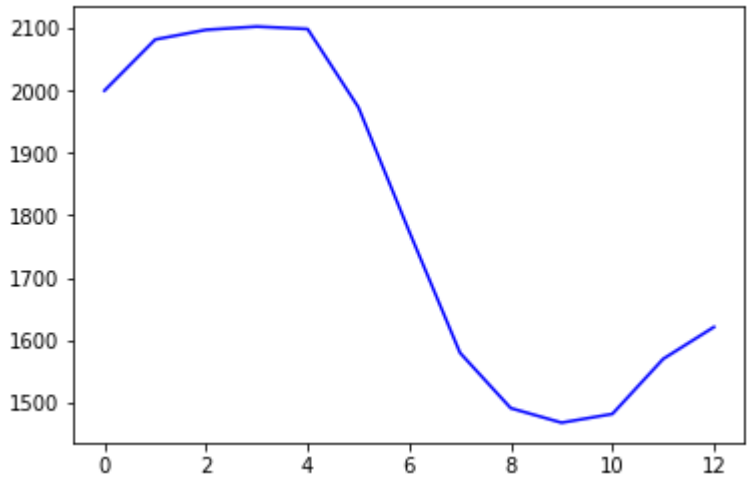


Figure 31: Reservoir stock calculated from standard inputs

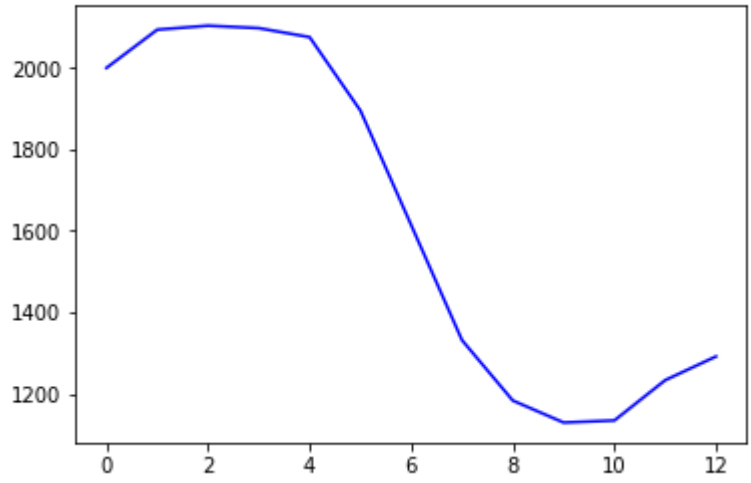


Figure 32: Reservoir stock calculated from satellite derived inputs

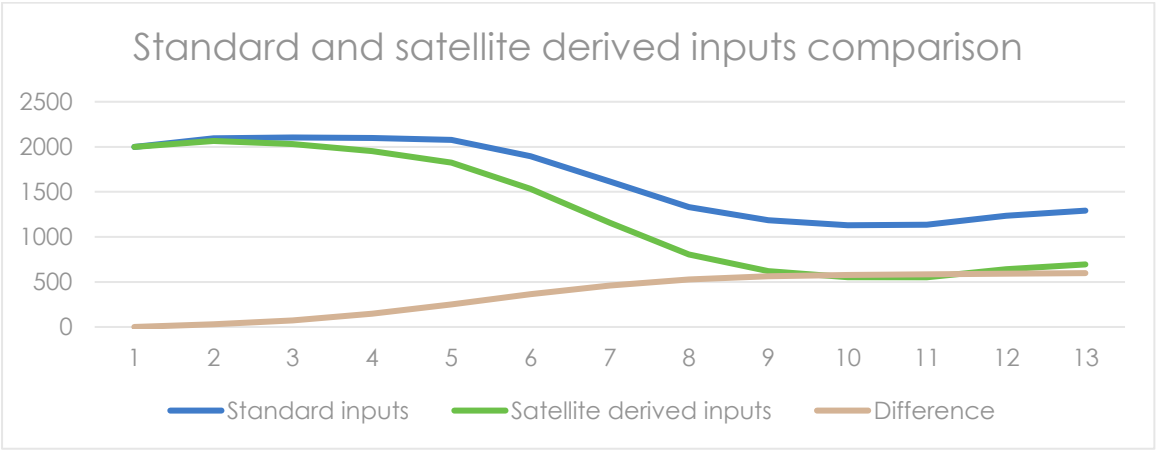


Figure 33: Standard and satellite derived inputs comparison

5.6 References

Allen, R. G., M. Tasumi and R. Trezza (2007). **Satellite-based energy balance for mapping evapotranspiration with internalized calibration (METRIC)—Model** Journal of irrigation and drainage engineering **133**(4): 380-394.

AURORA (2016-2019). **AURORA: Advanced Ultraviolet Radiation and Ozone Retrieval for Applications.** Contract No. H2020-EO-2015/687428 funded by the European Commission H2020 Programme. Coordinated by CONSIGLIO NAZIONALE DELLE RICERCHE, Italy. CORDIS database and http://www.aurora-copernicus.eu/

Cortesi, U., M. Bonazountas, A. Argyridis, K. Verberne, C. Tirelli, V. Palla (2018). A NOVEL APPROACH FOR ULTRAVIOLET RADIATION AND OZONE RETRIEVAL VIA COPERNICUS FOR APPLICATIONS: THE "AURORA" PARADIGM. **Virtualis: Social, Spatial and Technological Spaces in Real and Virtual Domains,** EDITOR: PROF SAVIOUR FORMOSA, UNIVERSITY OF MALTA, GATE: https://www.researchgate.net/profile/Saviour_Formosa; **Targeted Publication in 2021**

Cortesi, U. et al (2019). **Working with Copernicus for our Future**, abstract and oral presentation at National Conference on Copernicus Systems and Application for the Philippines, 11 March, 2019

Bastiaanssen, W. G., M. Menenti, R. Feddes and A. Holtslag (1998). "A remote sensing surface energy balance algorithm for land (SEBAL). 1. Formulation." Journal of hydrology **212**: 198-212.

Daughtry, C., C. Walthall, M. Kim, E. B. De Colstoun and J. McMurtrey Iii (2000). "Estimating corn leaf chlorophyll concentration from leaf and canopy reflectance." Remote sensing of Environment **74**(2): 229-239.

Gao, B.-C. (1996). "NDWI—A normalized difference water index for remote sensing of vegetation liquid water from space." Remote sensing of Environment **58**(3): 257-266.

Gislason, P. O., J. A. Benediktsson and J. R. Sveinsson (2006). "Random forests for land cover classification." Pattern Recognition Letters **27**(4): 294-300.

Huete, A., C. Justice and W. Van Leeuwen (1999). "MODIS vegetation index (MOD13)." Algorithm theoretical basis document **3**(213).

Jamalabad, M. (2004). Forest canopy density monitoring using satellite images. Geo-Imagery Bridging Continents XXth ISPRS Congress, Istanbul, Turkey, 2004.

Mils, A., M. Bonazountas (2019). **COPERNICUS: National Conference on Copernicus Technology and Applications, European Union Brings together the Copernicus program to the Philippines**. Final report, Contract No. 2018/402508, EC Brussels, COWI Coordinator. https://eeas.europa.eu/delegations/philippines/59354/european-union-brings-copernicus-programme-philippines_ar

Pelletier, C., S. Valero, J. Inglada, N. Champion and G. Dedieu (2016). "Assessing the robustness of Random Forests to map land cover with high resolution satellite image time series over large areas." Remote sensing of Environment **187**: 156-168.

Rouse Jr, J., R. Haas, J. Schell and D. Deering (1974). Paper A 20. Third Earth Resources Technology Satellite-1 Symposium: The Proceedings of a Symposium Held by Goddard Space Flight Center at Washington, DC on December 10-14, 1973: Prepared at Goddard Space Flight Center, Scientific and Technical Information Office, National Aeronautics and Space ….

Senay, G. B., S. Bohms, R. K. Singh, P. H. Gowda, N. M. Velpuri, H. Alemu and J. P. Verdin (2013). "Operational evapotranspiration mapping using remote sensing and weather datasets: A new parameterization for the SSEB approach." JAWRA Journal of the American Water Resources Association **49**(3): 577-591.

Senay, G. B., M. E. Budde and J. P. Verdin (2011). "Enhancing the Simplified Surface Energy Balance (SSEB) approach for estimating landscape ET: Validation with the METRIC model." Agricultural Water Management **98**(4): 606-618.

Sušnik, J., Chew, C., Domingo, X., Mereu, S., Trabucco, A., Evans, B., ... & Brouwer, F. (2018). **Multi-stakeholder development of a serious game to explore the water-energy-food-land-climate nexus: The SIM4NEXUS approach**. Water, 10(2), 139.

Waske, B. and M. Braun (2009). "Classifier ensembles for land cover mapping using multitemporal SAR imagery." ISPRS Journal of Photogrammetry and Remote Sensing **64**(5): 450-457.

Conclusions

Following the planning depicted in the Grant Agreement, the current version of the SIM4NEXUS Serious Game tool implements all the defined modules (GUI, KEE, S4N Database and SDM Engine) and provides all the requirements needed to play the SIM4NEXUS Game and achieve one of the main SIM4NEXUS project goals.

In order to integrate and manage all the S4N modules, the S4N Integration Centre has been accurately designed and developed to finally coordinate the SIM4NEXUS platform, which includes several development envirnoments.

In addition, five Case Studies have been already fully integrated, Greece, Azerbaijan, Latvia, the Netherlands and the southwest of the UK, thus making possible the interaction with them through the GUI and the final test to validate the expected and correct behaviour of each one. In parallel, the Global Case Study has developed a demo tool which can be accessed through the S4N SG platform.

Finally, to provide a high-quality system in terms of availability, capacity, interoperability, performance, reliability, robustness, safety, security, resilience and usability, several tests have been developed and integrated to the S4N Integration Centre to validate the S4N SG performance covering the previously mentioned topics.

Through the present document, the S4N IC and all these validation and testing tasks have been documented and exhaustively described.

As proof, the latest version of the Serious Game GUI and the underlying connected KEE, S4N database and SDM Engine are available and free to play at this URL: https://seriousgame.sim4nexus.eu/.