SEVENTH FRAMEWORK PROGRAMME

"Information Society Technologies"

D3.1
+Spaces platform overall Architecture

Project acronym: +Spaces

Project full title: Policy Simulation in Virtual Spaces

Contract no.: 248726

Workpackage:		3. System and Tools development
Editor:	Magdalini Kardara	ICCS/NTUA
Author(s):	Magdalini Kardara	ICCS/NTUA
	Fotis Aisopos	ICCS/NTUA
	Athanasios Papaoikonomou	ICCS/NTUA
	Omri Fuchs	IBM
	Michal Jacovi	IBM
	Zak Mandel	IBM
	Christoph Friedrich	SCAI Fraunhofer
	Ilias Spais	ATC
	Fanny Coudert	KULeuven
Authorized by		
Doc Ref:		
Reviewer	Michal Jacovi	IBM
Reviewer	Ido Guy	IBM
Reviewer	Ilias Spais	ATC
Dissemination Level		

The research leading to these results has received funding from the European Community's Seventh Framework Programme

Date	Author	Comments	Version	Status
01-06-2010	Magdalini Kardara	TOC	0.2	draft
13-07-2010	Magdalini Kardara	Assignment of sections	0.7	draft
26-07-2010	Magdalini Kardara	Ready for internal review	0.12	Pre-final
31-07-2010	Magdalini Kardara	Final	1.0	Final

The research leading to these results has received funding from the European Community's Seventh Framework Programme

Executive Summary

This document presents the overall architecture of the +Spaces platform. The platform design has been based on the requirements and specifications set out by WP2. The current document describes the process of mapping requirements to platform capabilities and outlines the platform architecture including a detailed description of the components layout, their internal architecture as well as of the interactions among them. It also includes the XML schemata that will be used for the representation of the generic entities in the framework of the +Spaces platform

The research leading to these results has received funding from the European Community's Seventh Framework Programme

Table of Contents

The research leading to these results has received funding from the European Community's Seventh Framework Programme

The research leading to these results has received funding from the European Community's Seventh Framework Programme

The research leading to these results has received funding from the European Community's Seventh Framework Programme

Table of Figures

The research leading to these results has received funding from the European Community's Seventh Framework Programme

1 Introduction

This document aims to outline the initial architecture design of the +Spaces platform.

Section 2 describes the design principles and processes followed during the design.

In Section 3 the main application scenario is presented and a brief overview of the use cases and functional requirements that are explained in detail in D2.2[1] is also given. What is more, the legal framework that is outlined in D2.3[2] is examined in order to translate to actual legal requirements that should be taken into account in the platform design. Through the study of the abovementioned requirements and specifications the platform capabilities are derived.

Section 4 presents in detail the architecture of the +Spaces platform. The overall architecture as well as the internal architecture of the components of the +Spaces platform is outlined here. Diagrams are used in order to better depict the associations and interactions between the various components. It should be noted that since the architecture design and implementation will follow an iterative process, the architecture presented in the current document reflects the developments that will take place during the first iteration. Though the general architecture of the platform is not likely to change greatly between iterations it is expected that several modifications in the components' design will take place. The final design for each iteration will be included in the components implementation reports that will be released in combination with the components.

The research leading to these results has received funding from the European Community's Seventh Framework Programme

2 + Spaces Design Principles and Process

1.1 *Design Principles*

1.1.1 Service Oriented Infrastructure

Service Oriented Architecture is an architectural paradigm based on reforming application functions and pieces of information into a "service" that can be accessed through a common interface regardless of the location of the function or of the piece of data.

The +Spaces SOAP based platform has to exhibit the following characteristics[3]:

- **Loosely Coupled Services**: Loose coupling is an approach to the design of distributed applications that emphasizes on agility (that is to adapt to changes). Loose coupling intentionally sacrifices interface optimization to achieve flexible interoperability among systems that are disparate in technology, location, performance, and availability. A loosely coupled application is isolated from internal changes in others by using abstraction, indirection, and delayed binding in the interfaces between the applications. As compared to traditional, tightly coupled applications, loosely coupled applications aim to be more reusable and adaptable to the unexpected.
- **Synchronous:** Supporting the synchronous invocation and execution of services, in the sense that when an end user or service requests information or invokes a function, a connection between the two end-systems must be maintained until a response is received.
- **Asynchronous**: Supporting also asynchronous interactions in which information is sent without the expectation of getting back an immediate response. This characteristic is very important in the cases where there is no requirement to maintain a connection between the two end-systems while waiting for a response.

1.2 *Design Process*

Task 3.2 "Middleware Design and Architecture" followed the Unified Process[4,5] principles and UML for the analysis and design of the +Spaces initial architecture and it is anticipated that this process will be also used by the development tasks during the implementation and validation of their components.

1.2.1 Unified Process

This Process is a framework which guides the tasks, people and products of the software design process. It is a framework because it provides the inputs and outputs of each activity, but does not restrict how each activity must be performed. The Unified Process is:

- **Iterative and Incremental:** The Unified Process has an iterative and incremental model. That is, the design process is based on iterations which either address different aspects of the design process or move the design process. In essence the end result is incrementally produced.

The research leading to these results has received funding from the European Community's Seventh Framework Programme

- **Use case driven:** In the Unified Process use cases are used to ensure that the evolving design is always relevant to what is required by the end user. In fact, the use cases act as the one consistent thread throughout the whole of the development process.

- **Architecture centric:** The Unified Process explicitly acknowledges the importance of the architecture for the successful completion of the project. It prescribes the successive refinement of the executable architecture thereby attempting to ensure that the architecture remains relevant.

1.2.2 Unified Modelling Language

Unified Modelling Language has been used in the analysis and design of the +Spaces platform in Task 3.2. The platform functionalities, models and processes presented in this document are illustrated using UML diagrams, at the level this is possible for an initial architecture, and these diagrams will be further detailed in the upcoming WP3 reports. Using UML, the Task 3.2 work is well organized and the results are efficiently capitalized from the development Tasks. More specifically, in the subsequent sections of this document we have used Sequence Diagrams and Component Diagrams.

Sequence diagrams model the flow of logic within a system in a visual manner, enabling both to document and validate the logic behind the system's development, and are commonly used for both analysis and design purposes. Sequence diagrams are the most popular UML artefact for dynamic modelling, which focuses on identifying the behaviour within a system. Other dynamic modelling techniques include activity diagramming, communication diagramming, timing diagramming, and interaction overview diagramming. Sequence diagrams, along with class diagrams and physical data models are the most important design-level models for modern business application development.

Also in this paragraph component diagrams are used. They are especially helpful in the initial architectural modelling efforts which focus on identifying the architectural landscape for the +Spaces system, as they enable modelling of the high-level software components and more importantly the interfaces to those components. Components may both provide and require interfaces. An interface is the definition of a collection of one or more methods, and zero or more attributes. The component diagram shows the interrelationships between different components that may be developed by different partners and in that way establishes and formal communication channel between them that is necessary during the development process.

The research leading to these results has received funding from the European Community's Seventh Framework Programme

3 +Spaces Platform Analysis

1.3 Use Case and Specifications Analysis

1.3.1 Application Scenario and Use Cases

The main application scenario is briefly described in the following section highlighting the technical challenges for the +Spaces platform and thereafter the results of the use case analysis are presented.

The research leading to these results has received funding from the European Community's Seventh Framework Programme

3.1.1.1 Application Scenario

This section will give a detailed overview of the generic application scenario in order to identify the challenges that need to be addressed and use them in order to outline the desired platform capabilities.

The application scenario is initiated by the policy maker who plans to introduce a new policy and wishes to measure public opinion and evaluate citizens' reactions. The policy maker (PM) will access the front end of the platform in order to create a new experiment in one or more of the virtual spaces (3D VWs and SNs) available. The PM will choose among the experiment types available (poll, debate, simulation) and configure the experiment accordingly. They will also select which virtual spaces to deploy it to, as well as the analysis services that they wish to use for further processing of the data. The analysis services available for selection will include the built-in +Spaces analysis services (data analysis, recommendation, and reputation) as well as external services with similar or complimentary functionality that may be later registered to the platform. The user's selection will be based on the operations that each service is able to offer as well as the price of each option. Upon selecting a service the user also agrees to pay the respective price for its operations.

After the PM has finished configuration, the experiment will be created on the selected virtual spaces, the analysis services will be notified and the experiment will be activated. The recommendation service, if selected, will suggest users that, based on the experiment description and the users' own interests, may be interested to participate; and send them invitations. The recommendations will be improved if the PM provides an initial list of relevant people, that the recommendation service will enhance. Recommendations will also be available on a per user basis.

Even with no specific experiment going on, virtual space users will be invited to participate in the +Spaces experience. They will be introduced with the +Spaces concept, presented with the expectations from them, and be asked to sign a consent form that will allow +Spaces to monitor their different activities in the virtual space.

After the experiment is activated, the virtual spaces users may participate in it with a variety of actions depending on the experiment type. All the actions (both those related to an experiment and other interactions) will be recorded in the platform, filtered and forwarded to the analysis services. In order to ensure data protection the virtual id of the users' identifies will not be sent to the services, aliases will be used instead. The analysis services will perform a real-time processing of the data they receive. In order to perform a more thorough analysis, the data analysis service may need the users' reputation rates.

The research leading to these results has received funding from the European Community's Seventh Framework Programme

While the experiment is active as well as after it has ended, the PM who created it is able to access the front end and view the results. The data analysis service will provide an analytical display of the results based on the filtering criteria defined by the user such as demographics (age, location etc), reputation rate etc.

The experiment will continue running until its ending clause has been satisfied. The ending clause might be one of the following:

- The expiration date has been reached
- The desired number of users has participated

After the end of the experiment, the users that have participated and have declared their interest in the outcome of the experiment will be notified of the final results.

3.1.1.2 Use Cases

Following the scenario and the initial technical analysis of it, use cases have been produced that describe the scenarios in a more consistent and formal way. The use cases have been presented extensively in Deliverable 2.2 [6] where more information concerning them can be found. These use cases are representative of the demands end-users will have on the +Spaces platform to provide the expected functionalities.

By analysing the use cases, various key capabilities have been identified. These are: Polling, Debating and Simulation in Virtual Spaces, Virtual Spaces Interoperability, Data Aggregation and Distribution, Data Analysis, Recommendation, Reputation, Data de-identification, Data Recovery, SLA Management, Trust and Security.

More details about these key capabilities for the platform can be found in section 3.1.1.2. In the following table the aforementioned capabilities are linked to the use cases they are associated with. The numbering of the use cases follows that of D2.2 report6.

Key Capability	**Use Case**
Polling, Debating and Simulation in Virtual Spaces	UC4, UC5, UC6, UC7, UC8, UC9, UC10, UC11, UC12, UC13, UC15, UC16, UC17, UC18, UC19, UC21
Virtual Spaces Interoperability	UC6, UC7, UC11, UC12, UC15, UC20
Data Aggregation and Distribution	UC8, UC13
Data Analysis	UC5, UC10, UC15, UC18, UC19
Recommendation and Reputation	UC5, UC10, UC14, UC15, UC18, UC19, UC21
Data De-identification	UC8, UC13
Data Recovery	

The research leading to these results has received funding from the European Community's Seventh Framework Programme

SLA Management	UC5, UC10, UC15, UC16, UC18
Trust and Security	

Table 1:+Spaces platform capabilities and use cases

It can be seen that two of the platform capabilities, namely Data Recovery and Trust and Security are not associated with any Use Cases. This is because they represent generic non-functional requirements (Fault tolerance and Security respectively) that are not derived directly from use cases but are of particular importance for our platform.

1.3.2 Functional and non-functional Requirements Analysis

The functional and non-functional requirements of the platform have been derived from the user requirements set out by the end users of the platform and are presented in detail in D2.2. Functional requirements are closely correlated with the Use Cases described in the above section. The association between functional and non-functional requirements and the platform capabilities are displayed in the following table.

Key Capability	Requirements
Polling, Debating and Simulation in Virtual Spaces	**Functional:** R1.1, R1.2, R1.3,
	Non-functional:
Virtual Spaces Interoperability	**Functional:** R2.1, R2.2
	Non-functional:
Data Aggregation and Distribution	**Functional:** R2.1, R2.2
	Non-functional:
Data Analysis	**Functional:** R3.7, R3.8, R4.9, R7.x
	Non-functional:
Recommendation and Reputation	**Functional:** R1.5, R4.4, R4.4, R4.7, R6.x
	Non-functional:
Data De-identification	**Functional:**
	Non-functional: Reliability
Data Recovery	**Functional:**

The research leading to these results has received funding from the European Community's Seventh Framework Programme

	Non-functional: Fault Tolerance – Robustness, Reliability
SLA Management	**Functional:**
	Non-functional: Reliability
Trust and Security	**Functional:** R5.2
	Non-functional: Security, Reliability

Table 2:+Spaces platform capabilities and functional requirements

It can be noticed that only a small set of Functional Requirements listed in D2.3 appears in the above table. This is because the table includes only functional and non functional requirements that are directly associated with a platform capability. The rest of the requirements platform have been taken into account in the platform specification and will be included in the architecture design as individual features and capabilities.

1.3.3 Legal Requirements

D2.3 contains an analytical report of the Legislative Framework that applies in the domains that the +Spaces project aims to tackle. Based on this analysis and in combination with the platform functionalities as required by the end users the following legal requirements have been extracted and will be considered during the design and implementation of the platform.

LR1. Processing personal data

User profile information is considered personal data and cannot be processed (even if pseudonymized, as long as the identity can be retraced by +Spaces service provider) without clearly informing the user on how the information will be used, and getting user consent. Collecting and processing personal information is a key factor for the success of our platform as all third party services need some level of personal information for their analysis. Data analysis services will need them for the filtering of the results according to used groups (based on sex, age, location etc) while recommendation and reputation services rely heavily on them for delivering more accurate estimations.

In order to comply with the legal restrictions we have decided to provide a consent form that clearly declares to all users that by participating to our experiments they are giving their consent for their profile information to be processed for the experiment purposes and that it will not be used for any other reason or revealed to third parties.

As an extra layer of protection we have decided to pseudonymize the data sent to third party services for analysis, so that only the abstraction layer on top of the virtual spaces is aware of the virtual spaces IDs of participating users. Full anonymization however will not be

The research leading to these results has received funding from the European Community's Seventh Framework Programme

supported as it is essential to be able to retract IDs from aliases for recommendation purposes. More on that can be found in the following section.

An additional issue that needs to be addressed here is the processing of sensitive data, which quoting from D2.3 includes data "*revealing* racial or ethnic origin, *political opinions*, *religious or philosophical beliefs*, trade-union membership, and ... data *concerning* health or sex life". As explained in the deliverable, in the case of sensitive data implicit consent of the user, i.e. making sure that he has received and read the information before the data are collected, is not enough, instead the user must give their explicit consent for their data to be processed. This means that the users need to perform some action indicating that they have read the consent form and agree to allow the processing of their data. In most cases this means ticking a box or pressing an "I agree" button. Several EU countries have introduced stricter legal provisions requiring the written consent of users, whereas others allow the processing of sensitive data made public by the user without requiring his or her express consent. The domestic data protection law applicable to the data processing is defined by the place where the data controller is established. A key point will therefore be to define which entity qualifies as data controller within the meaning of the applicable data protection framework. In the context of +Spaces, it is likely that we face a situation of joint controllership where both the policy maker who determines the purpose of the experiment and the the +Spaces provider who determines the essential elements of the means used to achieve this purpose will be deemed data controllers.   This requirement should therefore be dealt with on a case-by-case basis.

The issue of sensitive data needs to be taken into account in the +Spaces platform. Although end users will be instructed not to create experiments that encourage users to disclose such data , due to the political nature of the experiments as well as the option given to the users to speak their minds anonymously, one cannot eliminate the possibility that users themselves will reveal data falling under one of the aforementioned categories. In order to ensure compliance with European legislation, experiments that allow users to express their opinions freely (debated and simulation) will be accompanied by a consent form with a tick box. More specific requirements imposed by domestic laws will be timely taken into account.

As can be seen in D2.1 and D2.2 the signing of consent forms has also been identified as a user requirement and a functional requirement (UR12.13 and R8.5 respectively)

LR2. User's access to personal data

After the users have given their consent for the processing of their data they must at any time be able to see what personal data of theirs have been collected and also ask for this data to be deleted. We will support this requirement by ensuring that within the experiment page or space links are provided through which the user can gain access to their personal data and in case they change their mind and no longer wish to participate, ask for this data to be deleted from the system. Authentication mechanism will be established at this point in order to ensure the true identity of the user.

The research leading to these results has received funding from the European Community's Seventh Framework Programme

LR3. Collecting no more data than necessary for the experiment purposes

According to EU law, only the amount of data that is necessary for the purposes of a specific application must be collected and processed by it. In the case of +Spaces this means participation to the experiment as well as other user actions that will be used in order to provide users with good recommendations and rate their reputation. We will determine those actions carefully in order to ensure that no unnecessary data is collected. Moreover, after the experiment has ended the data collected in the frame work of an experiment will be deleted from the platform. For their better performance, the analysis service may keep aggregated data or the conclusions their analysis for a specific user (e.g. reputation rates, likes and dislikes etc.) but it will not be possible to retrieve user's actions after the end of an experiment. The deletion of experiment related data after the end of the experiment has also been identified as a user requirement and a functional requirement (UR12.11 and R8.4 respectively).

1.4 Platform Capabilities

The platform capabilities have been derived from the specifications and requirements from the section above. The current section gives a brief overview of each capability and the technical challenges associated with each one in the context of +Spaces.

1.4.1 Polling, Debating and Simulation in Virtual Spaces

Each experiment application can be deployed in one or more virtual spaces. Some application types are planned to be supported in all of the targeted virtual spaces while others may be supported in only a subset. For each type of experiment and for each virtual space some setting up and configuration is performed, this is different between experiments and platforms. This includes creating accounts, users, and other platform specific operations such as allocating spaces, pages and other in-platform resources. The deployment of applications is the process of adding the specific application data to the platform, making it available for targeted virtual space users and setting up the process of allowing users to participate and use the application, as well as the process of collecting the data from the platform.

Polls typically include presenting a topic and one or more questions. The questions may encourage various types of responses, such as selecting a single possible answer, selecting several possible answers, rating or providing a level of agreement with a statement and unstructured responses.

Debating experiments are initiated by presenting a topic of interest, raising various arguments and questions, and calling for responses. Participants' contribution to debates, synchronous or a-synchronous, is typically not structured, and relates not only to the initial statement but also to responses by other users.

In simulation we collect data on users' behaviour, to understand the change in behaviour as a result of changing certain aspects of the virtual space (e.g., implementing a policy). Two ways are considered to learn the effect of the applied change: the first is to observe users'

The research leading to these results has received funding from the European Community's Seventh Framework Programme

operations before the change and after the change; and the second is to observe two setting and only apply the change in one setting, and compare the users' behaviour in the two settings.

1.4.2 Virtual Spaces Interoperability

A single experiment is deployed in several virtual spaces platforms possibly at the same time. For each type of experiment application we collect the full set of data required for deployment in each virtual platform. Analysing the required data we derive a level of abstraction for each type of experiment allowing us to define the experiment once and use that definition in all of the supported virtual platforms. This data is sometimes accompanied with specific platform information. In addition we define the full set of data for the results we collect from the various platforms.

We explore various mechanisms for sharing data directly between different virtual platforms. These mechanisms allow increasing the participation in applications in some platforms by targeting users from other virtual platforms. In addition, this supports enriching experiments and potentially provoking more responses from users by presenting data coming from other platforms. Data coming from users of virtual spaces other than the one a user is in, may be different due to the characteristics of the other platform such as typical profile of users, means of displaying the topic, limitations and methods of participation, etc.

1.4.3 Data Aggregation and Distribution

The aggregation and distribution of data is of great importance for our project since the success of the experiments is heavily dependent on the reliable transferring of data from the virtual spaces to the analysis services. The data refers to actions taking place in the virtual spaces that may be of interest to the analysis services. Since we aim to target highly populated VWs and SNs in order to attract the maximum number of participants, the platform needs to be able to aggregate and store large amounts of data. The abstract layer that will reside on top of the virtual spaces will transform the data retrieved to a common format and forward them to the +spaces middleware where they will be stored. The data will also be forwarded to the analysis services. As has been explained in the application scenario, each experiment will be assigned to specific analysis services based on the government user's selection. Actions relevant to an experiment will be forwarded only to its corresponding analysis services. Reputation and recommendation services will also receive a broader set of actions representing the user's general behaviour inside the Virtual Space and not directly linked to any experiment.

For privacy reasons, in order to avoide storing unnecessary information, when a certain time has elapsed since the end of an experiment all data gathered for the experiment purposes will be deleted. The analysis service might aggregate data for future use but will not keep any information connecting specific actions to a user.

The research leading to these results has received funding from the European Community's Seventh Framework Programme

1.4.4 Data Analysis

The objective of the Data Analysis service is to present the results from experiments in visual form and to help policy makers to interpret the results. It gathers pseudonymized data during experiments on polls, debates and simulations and allows the e-government frontend to query for intermediate and final results of experiments. The main service will be to return a link to a data analysis report, upon given an experiment ID. The link returned will be a URL which references an application server (e.g. Tomcat). The presentation layer of the data analysis service will provide interactive elements, which allow filtering of results. Data analysis of running experiments will happen in near real-time.

1.4.5 Recommendation and Reputation

This section describes two separate services, that are both based on social information and user created content gathered from interactions of +Spaces users in the virtual spaces. The two services will share a code base for aggregating and analyzing the information, and each service will add its own functionality.

The objective of the recommendation service is twofold: on the one hand it is intended to provide recommendations of interesting/relevant experiments for users, in order to attract more users to the experiments; on the other hand, it will provide recommendation of relevant people for experiments. Both types of recommendations (experiment recommendations and people recommendations) will be based on social information.

When an experiment first starts, its description and initial list of potential participants (if provided by the PM) will be passed to the recommendation service in order to produce a list of recommended people who should be invited to participate. The recommendations will be based on past interactions of people in the virtual spaces, based on which we can infer their interests and social network.

Any user visiting a virtual space, may occasionally (e.g., when idle) be presented with a recommendation to experiments (polls, debates, etc.) that are potentially of interest. These recommendations will be based on the user's topic of interest, as well as on the user's social network (friends expressed an interest in the recommended item).

The main objective of the reputation service is to protect the system against misuse, by identifying patterns of malicious behaviour and assigning low reputation rates to users who express these patterns. In addition, and based on the types of social interactions received from the virtual spaces, additional types of reputation may be defined, such as involvement, influence, and more. The data analysis service will gain from an influence reputation, as it may weigh different contributions based on the influence reputation of their contributors.

The results of the reputation service will also be used in order to support the moderator functionality. The moderation of debates in order to detect malicious users is an important user requirements and its effectiveness will rely heavily on the Reputation Service. The moderator will use reputation rates in order to better assess users and more easily identify and ban those with malicious behaviour.

The research leading to these results has received funding from the European Community's Seventh Framework Programme

1.4.6 Data De-identification (pseudonymization[7])

The +Spaces platform collects some participants' actions in the scope of the experiment in the virtual spaces. Data of such operations contains information about the operating user and at least the virtual space user-id.

We introduce mechanisms to support maximal user privacy with the limitations of requirements from the functional specification

The recommendation and reputation services collect and process information about users' actions. They rely on the history of the user's created content and interactions with other users and with the space, in order to produce reputation information; and they rely on the interactions of the user's social network for producing recommendations. For this purpose, the interactions of a single user need to be aggregated under a unique user entity. Anonymous data is thus not a viable solution.

We present a pseudonymization mechanism which creates a pseudonym for each user in each virtual space. Thus, a user identifier is not processed after the pseudonymization stage.

Transaction pseudonym system, in which a user is assigned a new pseudonym for each transaction is a stronger privacy and de-identification mechanism however due to the requirements highlighted above we cannot make use of such a system.

Since the pseudonyms are only exposed internally within the middleware, pseudonyms usability is not considered important in our context, whereas unlinkability is important. To further enhance the unlinkability by third parties the pseudonymization mechanism will use randomly generated pseudonyms, when creating a pseudonym for a new user, while using the same pseudonym for each user over time.

1.4.7 Data recovery

As has been mentioned above, the reliable transferring of data from the virtual spaces to the analysis services is a key priority for our platform. In order to ensure that no actions are lost due to service failure. a recovery mechanism needs to be setup. In our design we have considered two alternative options. The first option was to establish a reliable messaging mechanism between the middleware and the services that ensures that all messages are delivered and delete messages after they have been sent. The second option was to store all actions for active experiments and instead of focussing on reliable messaging to build a recovery mechanism that allows services to retrieve past messages in cases of failure.

Choosing the first option would require setting up a FIFO messaging buffer between the middleware and each available service. After ensuring that the message has been delivered to the service it would be deleted from the corresponding buffer. One negative implication of this option is that we would need to dynamically create a new buffer for each available service as well as have multiple instances of an action to each buffer. It would also mean that it would not be possible for the services to retrieve past actions from the middleware as no actions are stored. So in the case that during the running of an experiment a service

The research leading to these results has received funding from the European Community's Seventh Framework Programme

decides that it also requires other types of actions than the ones it subscribed for initially it will not be possible to retrieve them.

As a result we have decided to implement the second option, namely store all actions safely in the middleware from where services can retrieve them at any time. We will also build a data recovery mechanism that allows services to poll for past actions based on the timestamp of the latest action that they received. The recovery mechanism will then return a list of all actions that occurred between the timestamp and the current time. Elaborate filtering options will be enabled so that it is possible to retrieve only specific types of actions or actions from a specific virtual space.

1.4.8 SLA Management

The aim of the +Spaces project is to provide government agencies with a powerful platform that allows them to retrieve large amounts of data from the Virtual Spaces and process them through the means of specialised analysis tools. For the scope of the +Spaces platform, the built-in analysis services that will be implemented and showcased will be the Data Analysis, Recommendation, and Reputation services described in 4.1.1.15. In order, however, to make the +Spaces platform a commercially viable tool beyond the end of the project, we need to allow for external services to be registered and managed by the platform. This will give government users the option to choose between different services with similar functionalities as well as allow for the addition of different types of processing tools that have not been foreseen during the project's implementation. Government users will be able to select the services they wish to use for their experiment based on the level of Quality of Service offered by each service provider as well as the price requested. The QoS monitoring and billing will be made through the use of Service Level Agreements (SLA).

SLAs are a powerful tool for Service Providers (SP), allowing them to offer Quality of Service (QoS) guarantees to potential customers. SLA contracts have been widely accepted by the SOA community as a means to establish a level of trust between the two parties (Service Provider and Consumer) and are currently considered by many as a prerequisite for all commercial SOA applications

In our platform, SLAs will be established between the analysis service providers and the government users and will be monitored by the platform. Upon registration, the service providers will offer an agreement containing the QoS guarantees and pricing terms for the use of their services. On the creation of a new experiment, and more specifically during the service selection process, the government user will be presented with all the available offers. Upon the selection of a service, an agreement will be established between the two parties. While the experiment is running, the +spaces platform will monitor usage in order to evaluate conformance to the promised QoS terms and calculate charges. The SLA framework is presented in more detail in 4.1.1.12.

The research leading to these results has received funding from the European Community's Seventh Framework Programme

1.4.9 Trust and Security

Security is always an important parameter for any application that gathers and operates on sensitive data. Since the project is about policy simulation, it is vital to protect the exchanged messages from corruption and unauthorized access. We will try to satisfy three of the basic security requirements which are confidentiality, integrity and authenticity of data.

Confidentiality is about preventing the disclosure of information to unauthorized individuals or systems. To achieve this goal, we will encrypt the SOAP messages exchanged between the various services. Integrity is about taking measures to ensure that the messages have not been tampered with during their exchange . We will use the notion of digital signatures for this purpose. Finally, authentication is the process of determining whether something is, in fact, what it is declared to be. Such a requirement is satisfied through the use of authentication tokens like digital certificates, username tokens or SAML tokens.

The primary goal should be to protect the privacy of users by adequately protecting their personal data . Data security and de-identification , as described in 3.2.6, are the “tools” to achieve that.

The research leading to these results has received funding from the European Community's Seventh Framework Programme

4 +Spaces Platform Design

1.5 *General Overview*

Figure 1 presents the high level architecture of the +Spaces system. As can be seen in the figure, the +Spaces consists of two layers, the VS Management layer and the +Spaces Middleware layer.

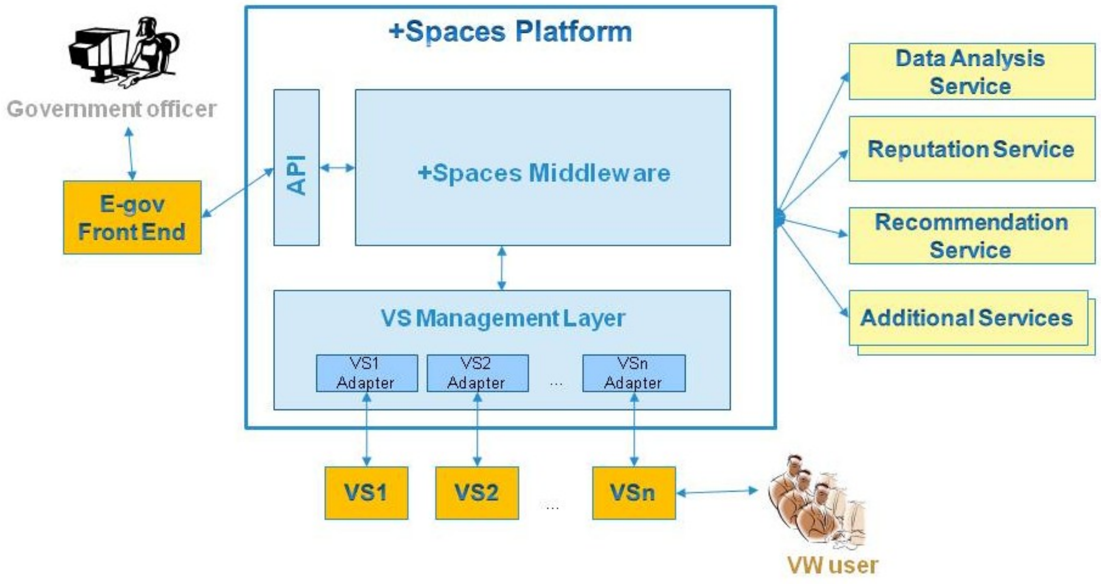


Figure 1:+Spaces High level architecture

The VS Management acts as an abstraction layer on top of the Virtual Spaces. It provides a common interface to the underlying Virtual Spaces and can be used as a standalone tool. For each participating Virtual Space an adaptor will be developed acting as a bridge between the Virtual Space and the +Spaces platform.

The +Spaces Middleware contains the main functionality of the platform. It handles the communication and data flow between the government end users, the VS Management Layer and the built-in and external analysis services. It offers data persistence, storing all information about experiments, actions and services and supports a recovery mechanism that allows analysis services to retrieve their data in case of failure.

The +spaces platform is able to manage services that enhance the functionality of the platform by providing powerful processing functionalities. Three built-in analysis services will be developed in the framework of the +Spaces project: the Data Analysis, Recommendation, and Reputation Services. The platform will however support the addition of more services, with similar or different functionalities, giving the government users a wider selection of tools.

The research leading to these results has received funding from the European Community's Seventh Framework Programme

Finally the e-gov Front-End is a graphical interface that interacts with the platform via the +Spaces API and provides government end-users with an easy to use access point to the platform functionality.

1.6 *VS Management Layer*

VS Management Layer is responsible for providing an interface between the +Spaces Middleware Layer and external systems, such as 3D virtual worlds and social networking sites. The layer's main goal is the transparent interoperability between all kinds of virtual worlds.

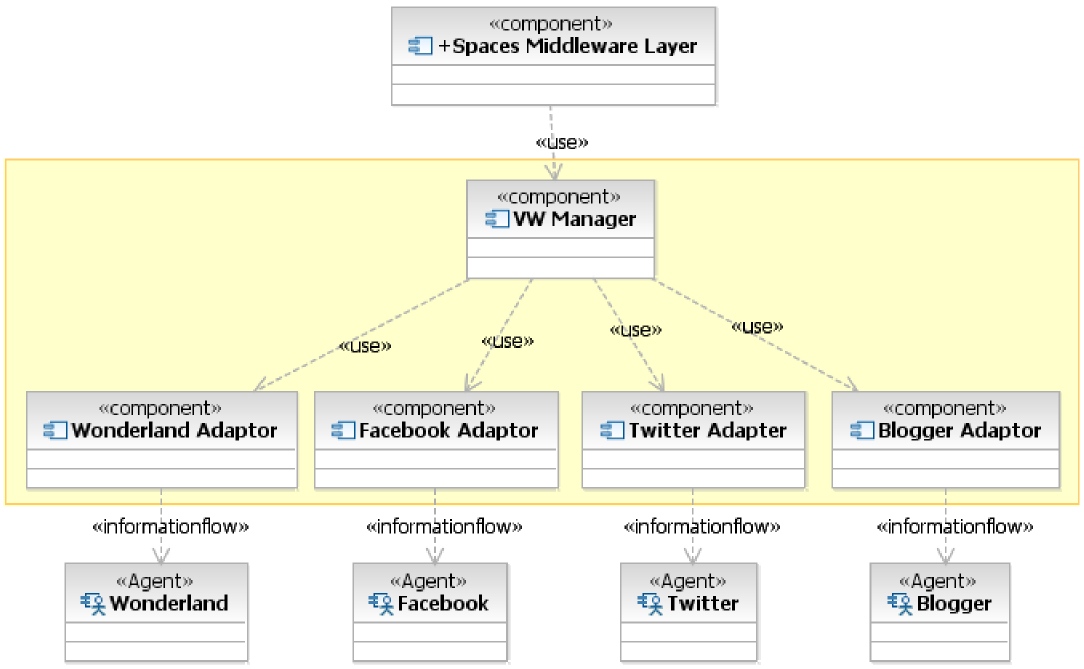


Figure 2: VS Management Layer Architecture

The layer includes a central component named VS Manager, which will centralize and manage all the work with plurality of components named VS Adaptors, which will provide a technical communication to virtual worlds.

The layer will provide an interface for VS Adaptors management, scheduling of VS resources and deployment of experiments into virtual worlds.

VS Adaptors will monitor utilised resources in virtual worlds and report actions back to VS Manager component, which will match these actions to a relevant experiment, replace real users information with their pseudonyms ad forward actions to an upper layer (in case of +Spaces, to Experiment Manager component).

The research leading to these results has received funding from the European Community's Seventh Framework Programme

The layer will be designed to be generic and can be reused in any other project.

1.6.1 VS Manager

VS Manager is a central component of the layer, and is responsible for centralizing the work of all VS Adaptors and to implement a convenient interface to a Middleware Layer. The manager itself will include several internal components, each responsible for handling part of the work, which will be defined on a low level design. For example, a resource scheduling component, a users' pseudonymization component, etc.

Management of VS Adaptors:

The VS manager is responsible for managing the VS adaptor components, their registration, configuration, and the retrieval of their status.

- **Adaptor registration**
  A new adaptor will be installed by the system administrator as a separate application. After adaptor installation, the administrator will declare the adaptor to the VS Manager, using the application interface (to avoid unauthorized access). Upon a new adaptor declaration, the manager component will register it in its database. The adaptor component itself is responsible for reporting its status to the manager component.

- **Configure adaptors**
  A main configuration will be performed upon adaptor registration. The manager component will load the adaptor configuration from a database and will configure the adaptor.
  The manager will provide a configuration application interface for configuration changes. On each configuration change, it will store relevant configuration in its database and forward updates to a relevant adapter(s). Configuration parameters will be evaluated during low level design and grouped in 2 categories: common parameters for all adapters and adaptor specific parameters. Probably, adaptor specific parameters will be injected by administrator directly into adaptors, using one of the known mechanism, such as properties file.
  The project aims to create a web administration console for layer, that will concentrate all needed configuration parameters.

- **Get adaptors list**
  The adaptors list items will include ID, type, name of VS and some other basic characteristics that will be defined later.

- **Get adaptor capabilities**
  Adaptor capabilities include poll/debate/simulation, ability to create a new resource programmatically, etc.

- **Get adaptor statuses**
  This includes adaptor status (starting, ready, error) and status of connection to VS.

The research leading to these results has received funding from the European Community's Seventh Framework Programme

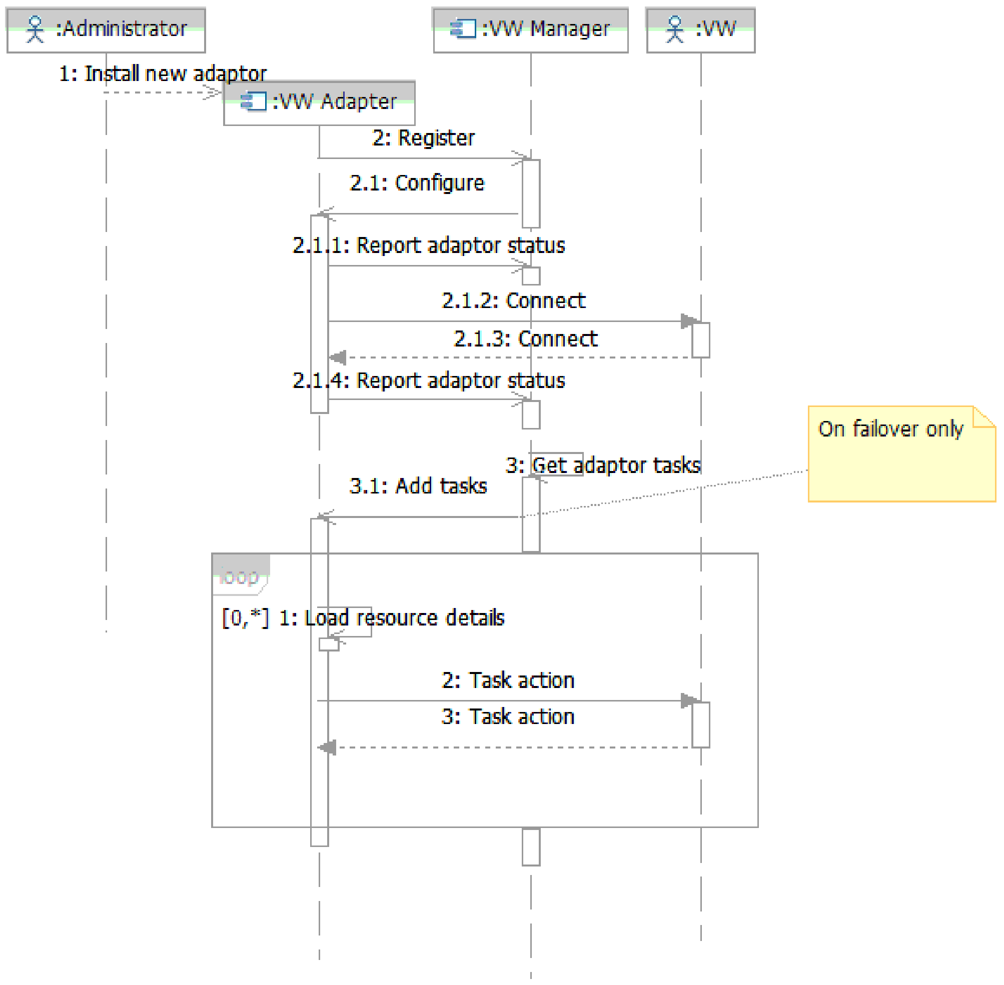


Figure 3: VS Adaptors registration

VS resources handling:

The manager component will be responsible for handling all allocated resources in virtual spaces, to handle their reservation, scheduling, status, etc. It will be responsible for storing the information in its database and will be able to restore it on failover. The information that will be stored is general information only, while resource specific parameters will be saved by the adaptors.

- **Add a new resource**
  The resource is allocated externally by a government employee and declared to a system using the application interface. The basic resource details will be stored in tables of the VS Manager, for reservation and scheduling. The manager will notify the relevant adaptor about a newly allocated resource, and the adaptor is required to preserve the resource specific information.

- **Manage resources reservation and scheduling**
  The manager will be responsible to handle resources availability, to allow resource reservation based on time range.

The research leading to these results has received funding from the European Community's Seventh Framework Programme

- **Get available resources**
  There will be several types of query: currently available resources, resources at specific time range, etc.

- **Create a new resource (if applicable)**
  Not all adaptors will support this functionality. If an adaptor supports such functionality, it will return created resource parameters to the manager component, which will be able to preserve it and to add to the scheduling mechanism.

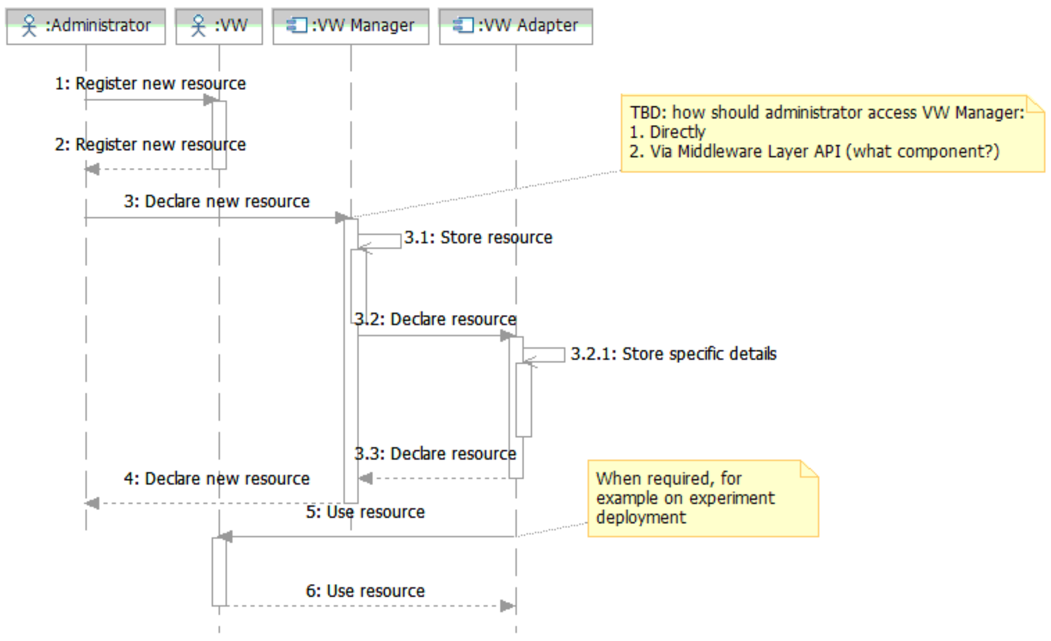


Figure 4: Allocate VS Resource Sequence Diagram

Experiment handling:

The manager will provide an application interface for deploying experiment details on selected resources. For example, to deploy poll information on a selected poll booth(s) in 3D world.

- **Deploy experiment on selected resource**
  The manager will check selected adaptor status, will check the resource scheduling and will pass deployment information to selected adaptor. It will store experiment ID for utilized resource, in order to be able notify upper layer on detected activity related to the resource. The manager will also store a details related to monitoring task of adaptor, in order to be able to restart this task after failover or cancel if the task was expired.

- **Stop experiment deployment on selected resource**
  The manager will clean its database information and pass the request to the selected adaptor.

The research leading to these results has received funding from the European Community's Seventh Framework Programme

- **VS actions handling**
  The VS Manager will accept all reported actions from VS Adaptors, replace user names by their pseudonyms (see below), load relevant experiment ID and pass the action to an upper layer.

- **Notify VS user**
  The manager will support passing of user notification requests from upper layer to selected adaptor.

Users' pseudonymization:

The manager will be responsible to hide real user names from the upper layer. For this purpose it will create a pseudonym for each involved VS user and will create bi-directional mapping in its database (optional: to use LDAP).

On each reported action from a VS world, the manager will try to find the user pseudonym in its database and to replace the real name with it. On failure to find defined pseudonym, it will create a new pseudonym.

On each request to notify VS user, the manager will find a real user VS name by its pseudonym and replace it in all requests to VS Adaptor. On failure to find the name, the manager will not pass the request and return error to the upper layer.

Failover recovery:

The manager component will be able to handle both recovery of the whole system and recovery of specific adaptor. In both cases the manager will load configuration information and configure adaptor(s). Then it will retrieve all the tasks related to running experiments and for each task will check resource allocation expiration. Expired allocations will be cancelled. Unexpired tasks will be passed to recovered adaptors and adaptors will be responsible to try to recover missed actions and to report them to the system.

Sharing of actions between virtual worlds:

Optionally, the VS Manager will be able to share activities between virtual worlds, by reinjection of users' votes, opinions and other activities from one adaptor to another adaptor(s). The adaptors will be responsible to display these events on a selected resource(s).

In this way users will be exposed to opinions of users from other worlds, and will have the possibility to distribute experiment links to additional accounts in many virtual worlds.

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2011] under grant agreement no. 248726

1.6.2 VS Adaptors

VS Adaptor components implement a technology bridge to specific virtual worlds. For simplicity, it is assumed that each component will handle connection to one virtual world only. Each adaptor will implement a required protocol, specific for each virtual world.

Adaptors will be responsible to store virtual world connection parameters, to store allocated resources details, and to provide the following functionality:

1. To request configuration information from VS Manager on start-up
2. To open connection to virtual world and to report connection status to VS Manager
3. Deploy experiment details on selected resource(s) in virtual world
4. Monitor experiment progress and report relevant actions to VS Manager
5. Pass messages to selected users
6. Recover missed actions after failure recovery (whenever applicable)
7. Create new resources in virtual world (wherever applicable)
8. Display activities from other worlds (votes, opinions), injected by VS Manager

4.1.1.1 Open Wonderland Adaptor

The Open Wonderland (OWL) VS adaptor shall be responsible for managing an experiment in an OWL installation, collecting and reporting experiment results to the VS Manager. The adaptor shall be part of an OWL installation, and shall be deployed using the OWL modules mechanism.

Functions

The adaptor shall expose an API that allows deploying, managing and stopping experiments. The VS adaptor shall provide the following functions itemised in section 4: 1, 2, 3, 4, 5, 8.

Communications

The adaptor shall communicate with an OWL installation using a RESTful API. The API shall require authentication.

The VS adaptor shall report relevant actions to the VS Manager via a mechanism yet to be decided.

4.1.1.2 Facebook Adaptor

Overview

The research leading to these results has received funding from the European Community's Seventh Framework Programme

The Facebook adaptor is responsible for publishing information in the Facebook platform, managing the published information, and monitor users' responses.

We consider two approaches for experiments on Facebook, and for different experiments we may use one or the other.

- **Facebook application**

  Facebook applications are third party applications (+Spaces is the third party in this case) that are hosted on third party servers. Facebook users may 'add' an application and allow the application to access various information and functions of their Facebook account. This may include list of friends, age, gender and other profile information as well as permission to perform functions like publish a status on the user's behalf. We are interested mainly in information and not functionality.

  The application itself appears as an internal page within Facebook and may contain various content and functions.

- **Use Facebook inherent functions**

  Some types of experiments may take advantage of internal Facebook building blocks such as joining groups, commenting on walls, 'liking' topics or pages, becoming friends, etc. We will explore the options to programmatically create manipulate and monitor such elements using the Facebook API.

Communications

In the case of inherent functions, we will use Facebook RESTful API. In the case of a Facebook application the application will either be collocated with the Facebook adaptor, or expose a RESTful API.

Facebook application must be hosted on servers accessible from the internet, not behind firewall or NAT.

Retrieved results, comments and other user interactions are sent from the adaptor to the VSManager.

4.1.1.3 Twitter Adaptor

Overview

The twitter adaptor is a separate stand-alone application. Its function is to publish information in the twitter micro-blogging platform, to enable the +Spaces applications, and collect information back from twitter.com.

The research leading to these results has received funding from the European Community's Seventh Framework Programme

In Twitter, hashtags are a common mechanism for tagging tweets, categorizing them and making finding them easier. The twitter adaptor will make use of the twitter search API utilizing user-ids as well as hashtags to collect information and provide it back to the middleware.

Functions

The adaptor exposes an API that allows deploying, managing and stopping experiments, and uses a continuous channel to report in-space user actions back to the middleware.

Deploying an experiment in twitter includes publishing one or more tweets by one or more users. These tweets may include external links. The tweets typically include +Spaces and specific experiment hashtags.

The Adaptor will collect information about the experiment, i.e tweets with our hashtags, directed to our user accounts, retweets etc, and in addition collect tweets made by participating users, to provide the required data for the recommendation and reputation services.

Communications

The adaptor communicates with the twitter platform using its RESTful APIs. Some methods in the twitter RESTful API requires authentication using the OAuth protocol, specifically 'Tweeting' or publishing information on behalf of a user. For that +Spaces will have one or more tweeter accounts, and the required OAuth authorization tokens to publish information.

Some of the methods for retrieving information back from twitter do not require authentication.

Retrieved results, comments and other user interactions are sent from the adaptor to the VSManager.

External links

Due to some of twitters characteristics such as unstructured text and 140 characters limitations, some of the experiments may include links to external web pages. We will use on of the tinyurl mechanisms to shorten the urls. The external web pages utilize regular web pages flexibility for presenting and collecting data. We use a mechanism exposed by twitter allowing us to redirect users browser to twitter.com with a status parameter that is them inserted to the tweet text box, if the user is logged in. This method enables us to mix flexible web presentation techniques with twitter's 140 characters text limitations.

External pages must be hosted on servers accessible from the internet, not behind firewall or NAT.

The research leading to these results has received funding from the European Community's Seventh Framework Programme

4.1.1.4 Blogger Adaptor

Overview

The Blogger adaptor deploys and monitors experiments in the Blogger.com blogging platform.

Functions

The Blogger adaptor allows publishing new posts in a blog and posting comments, on behalf of the experimenters. It monitors comments made by other users, and can retrieve the available information about the commenting users. The available information varies since users may be identified using several different identity mechanisms each containing different data elements. For Blogger.com users, this information may contain also blogs the user follows and blogs the user owns.

Blogger.com also supports widgets and embedding web pages in a blog, the adaptor also handles embedding such web pages and collecting usage information from them.

Communication

The Adaptor communicates with the Blogger using RESTful API. Methods for posting new blogs, new posts and new comments requires user authentication. We use https user/password authentication with Google accounts. Retrieving comments and public profile information is done over http with no authentication; results are presented in an ATOM format.

Retrieved results, comments and other user interactions are sent from the adaptor to the VSManager.

1.7 Middleware Layer

This section presents in detail the components of the +spaces platform middleware. For each component a detailed design layout is provided, including functionality overview, internal architecture and interfaces.

1.7.1 Experiment Manager

This component is the main component that interacts with the system's front end and UI. All of the user functions are concentrated and managed in the experiment manager. The experiment manager operates as an orchestrator of the various platform components.

It communicates directly with the Services Manager, VS Manager, Front end, and Configuration and SLA managers.

As the endpoint of the platform before the frontend component, this component is responsible for exposing platform capabilities and current condition, such as available services, available virtual spaces and virtual space resources. It enables creation of an experiment that will be deployed in various virtual spaces. This process includes assigning

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2011] under grant agreement no. 248726

the proper services to the experiment, retrieval of the recommended users, and ordering the VS Layer to activate the experiment. As actions made by participants begin to flow back from the VS Layer, the Experiment Manager stores them, evaluates the specific policy and distributes them to the assigned services via Services Manager component.

The frontend component may ask the experiment manager for results, in which case the experiment manager will refer the request to the data analysis service and return a reference to the front end where results can be viewed.

Experiment control, modifications, management and termination are also controlled by the Experiment Manager.

The experiment manager exposes data about ongoing experiments within the platform.

Experiment Manager	
High Level Description	The component is responsible for managing all experiments and actions related to them
getServices	
@WebMethod Collection <Service> getServices()	
Description	Returns a collection of the available analysis services.
getVirtualSpaces	
@WebMethod Collection <Space> getVirtualSpaces()	
Description	Returns a collection of the available virtual spaces.
createExperiment	
@WebMethod String createExperiment (Experiment experiment)	
Description	Create an experiment based on the data in the experiment object and returns an experiment ID.
newAction	
Void newAction (Action action)	
Description	Used for reporting a new participants action in one of the virtual spaces.
storeAction	
Void storeAction(Action action)	
Description	Persists the actions details.
evaluatePolicy	
evaluatePolicy (Policy policy)	
Description	Evaluates the policy associated with an experiment in order to decide how to process an action.
getExperimentData	
@WebMethod Collection <String> getExperimentData(String experimented, Collection<String> propertiesNames)	
Description	Returns current values for experiment parameters such as number of participants experiment duration etc.
getExperimentResults	
@WebMethod URL getExperimentResults (String ExperimentID)	

The research leading to these results has received funding from the European Community's Seventh Framework Programme

Experiment Manager	
Description	Returns a URL to a web page where a report can be downloaded or a web report can be viewed.

1.7.2 ServicesManager

4.1.1.5 Overview

The ServicesManager is the component responsible for managing the analysis services as well as managing and coordinating all communication between them and other middleware services. The ServicesManager accepts registration requests from analysis services, registers them in the +Spaces platform and maintains their details. When a new experiment is created, based on the government user's selection, the ServicesManager assigns the experiment to the analysis services. Throughout the experiment, when an action occurs, it will initiate notifications to the services assigned to this experiment.

4.1.1.6 Component Design

The ServicesManager will consist of three classes:

- The ***RegistrationManager*** will be responsible for accepting and handling new registrations to the +Spaces platform.
- The ***ServiceRegistry*** will be responsible for managing all registered services. It will accept queries from the ExperimentManager and the Front-End in order to return all the available services that are registered to the platform. It will also communicate with the SLA Manager in order to retrieve AgreementOffers associated with each service.
- The ***ExperimentAssignmentManager*** will be responsible for managing the associations between the Services and the Experiments Assigned to them. It will interact with the NotificationManager in order to send notifications for new experiments as well as actions associated with an experiment. It will also communicate with the SLAManager in order to create an Agreement when a new assignment has been created.

The research leading to these results has received funding from the European Community's Seventh Framework Programme

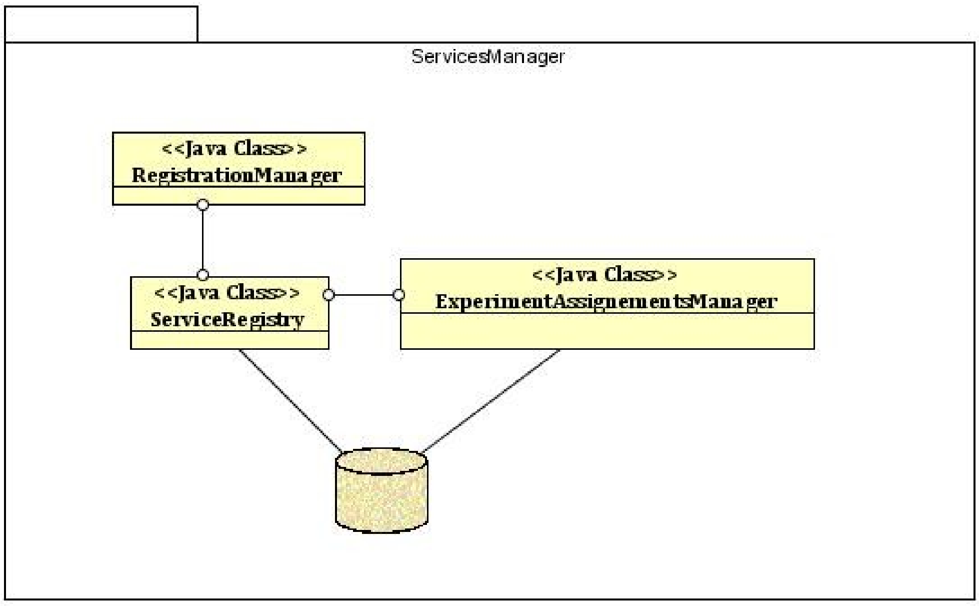


Figure 5: ServicesManager class diagram

4.1.1.7 Interface

Services Manager	
High Level Description	Manages built-in and external analysis services and handles communication between middleware and them
register	
@WebMethod RegistrationOutputType register(RegistrationInputType serviceRegistration)	
Description	Registers a new analysis service to the platform
getAvailableServices	
List<RegisteredService> getAvailableServices()	
Description	Returns a list of all registered services that are currently available.
List<RegisteredService> getAvailableServices(AnalysisServiceType serviceType)	
Description	Returns a list of all registered services of the given type that are currently available.
assignServicesToExperiment	
void assignServicesToExperiment(String experimentID, List<String> services)	
Description	Assigns the analysis services to the experiment and notifies the services.
setPolicy	
void setPolicy(PolicyType type)	
Description	A notification from the ExperimentManager that is responsible for evaluating the forwarding policy informing the service of the new policy
getServiceURLForExperiment	
List<URL> getServiceURLForExperiment(String experimentID, AnalysisServiceType type)	
Description	Returns analysis services of the given type that are assigned to the experiment
List<URL> getServiceURLForExperiment(String experimentID)	

The research leading to these results has received funding from the European Community's Seventh Framework Programme

Services Manager	
Description	Returns all analysis services that are assigned to the experiment
getExperimentRecommendationsForUser	
List <String> getExperimentRecommendationsForUser(String UserID)	
Description	Polls recommendation service and returns a list of all recommended experiments for this user
getRecommendedUsersforExperiment	
List <String> getRecommendedUsersforExperiment (String experimentID)	
Description	Polls recommendation service and returns a list of all recommended users for this experiment
getReputationForUser	
@WebMethod ReputationRate getReputationForUser (String userID)	
Description	Polls reputation service and returns reputation rate for users
getUsersWithReputation	
@WebMethod List<String> getUsersWithReputation(ReputationRate rate)	
Description	Polls reputation service and returns users with reputation rate higher than the given rate

1.7.3 NotificationManager

4.1.1.8 Overview

The NotificationManager will be responsible for notifying the analysis services of new experiments as well as actions taking place in the Virtual Spaces. There are various types of actions that can take place in the framework of a Virtual Space. While in the case of polls and debates one can expect a small and easy to predefine set of action types, in the case of simulation, depending on the specific scenario, the types of actions that can be associated with an experiment may vary greatly. The analysis services may be interested in all of the actions that may take place in a VS or in a small fraction of them. For example, reputation services are generally interested in getting as much information for the user as possible in order to produce a more trustworthy reputation rate while a data analysis service might only need voting events.

The decision on what action needs to be sent to each service is taken based on subscriptions. In order to receive notifications for a specific type of action the service needs to subscribe to notifications associated with this particular type. It should also be possible to subscribe to all action types available.

The research leading to these results has received funding from the European Community's Seventh Framework Programme

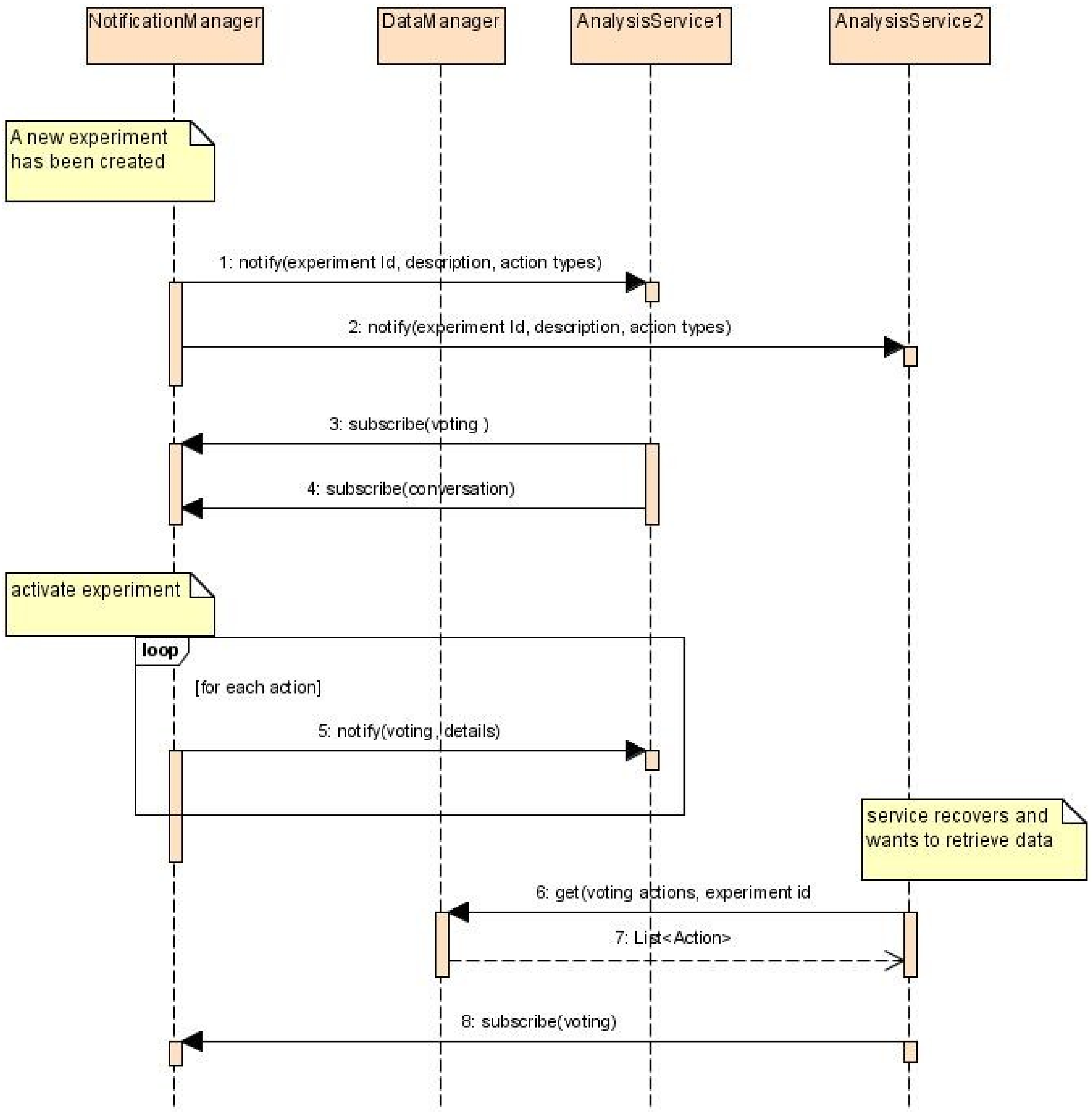

NotificationManager
DataManager
AnalysisService1
AnalysisService2
A new experiment has been created
1: notify(experiment Id, description, action types)
2: notify(experiment Id, description, action types)
3: subscribe(voting)
4: subscribe(conversation)
activate experiment
loop
[for each action]
5: notify(voting, details)
service recovers and wants to retrieve data
6: get(voting actions, experiment id
7: List<Action>
8: subscribe(voting)

Figure 6: Services Subscribe to new experiment

Upon creation of a new experiment the analysis services receive a notification. The notification contains the ID of the new experiment, experiment description and tags, as well as the action types available. After receiving the notifications, the services must send subscription requests in order to specify the action types they are interested in receiving. In order to avoid loss of data, the +Spaces platform will allow some time for the services to subscribe and then activate the experiment. If, however, a service fails to subscribe to action types in time (due to failure or other delay) it can always subscribe after the activation of the experiment and retrieve the lost actions via the data recovery mechanism.

The sequence diagram describes the abovementioned process. Both services are notified of the new experiment but AnalysisService2 fails to subscribe in time for the new experiment due to service failure. AnalysisService1 on the other hand subscribes normally to the action

The research leading to these results has received funding from the European Community's Seventh Framework Programme

types it is interested in (voting and conversation) and as soon as the experiment is activated it starts receiving notifications.

When AnalysisService2 recovers from failure, it uses the data recovery mechanism described in 4.1.1.10 in order to retrieve older messages and then subscribes to the new experiment. It should also be noted that in case of failure the service will first use the data recovery mechanism in order to retrieve notifications for new experiments so even if the service fails before receiving the notification it will still be able to recover all actions in the experiment.

4.1.1.9 Component Design

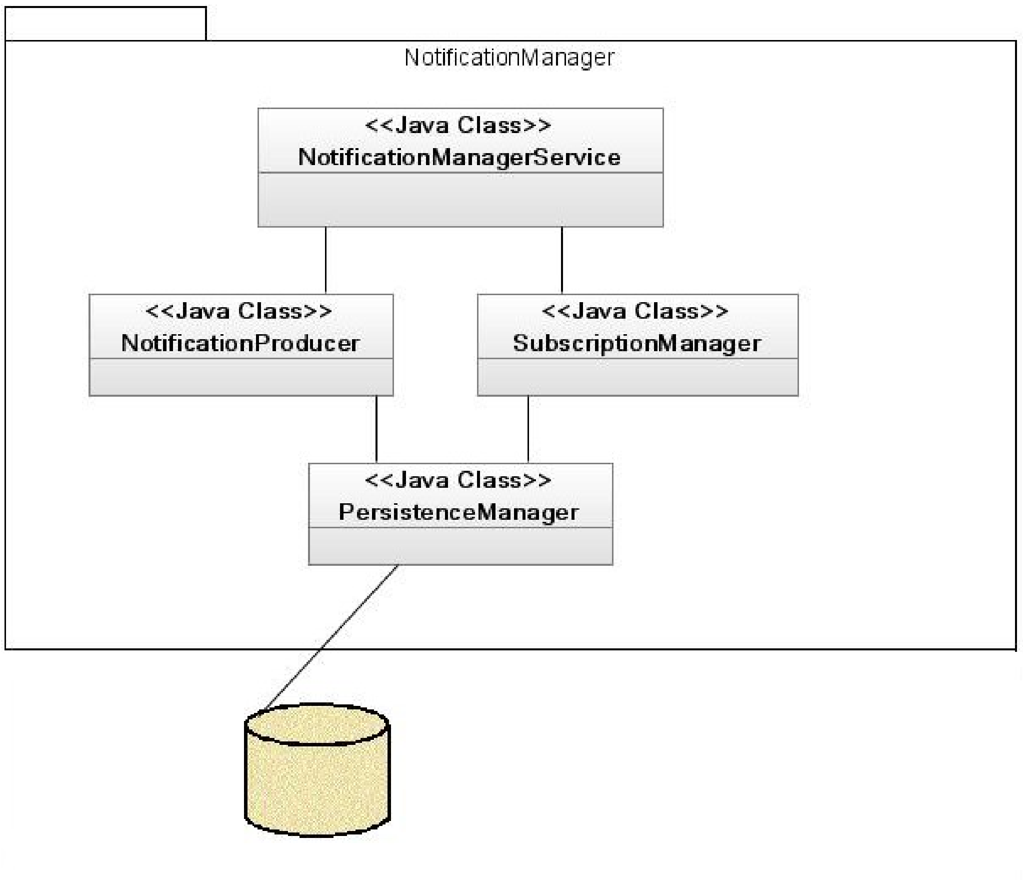
The architecture of this component has been designed following the OASIS Publish-Subscribe Notification for Web services design pattern specification. WS-Notification is a family of related white papers and specifications that define a standard Web services approach to notification using a topic-based publish/subscribe pattern. The Publish-Subscribe Notification for Web Services sets the general requirements for the WS-Notification family of specifications, describes each of the specifications that make up this family and defines a set of terms and concepts used in the specifications. The WS-Base Notification specification defines the Web Services interfaces for notification producers and notification consumers, as well as for the subscription manager, handling the subscriptions. It includes standard message exchanges to be implemented by service providers that wish to act in these roles.

The main notions and entities involved in the subscription and notification processes as described in the aforementioned specifications are the following[8]:

- A **Situation** is an event that is known to the NotificationProducer and may be of interest to external parties
- A **Notification** is a description of a Situation (represented by a **Notify** message) that the NotificationProducer wishes to communicate to other entities.
- A **NotificationProducer** is responsible for producing Notifications for those NotificationConsumers for which Subscriptions have been registered. The NotificationProducer will send Notifications for different Situations to each NotificationConsumer based on the parameters specified in the respective Subscription request representing the topics of their interest.
- A **NotificationConsumer** is an endpoint capable of receiving Notifications produced by a NotificationProducer as a result of Subscription.
- *A* **Subscription** represents the relationship between a NotificationConsumer and a NotificationProducer and contains the topic of interest for which the former wishes to *receive Notifications from the latter.*
- A **SubscriptionManager** is an endpoint that implements message exchanges associated with querying and manipulating Subscription resources.
- A **Subscriber** is the entity that sends the Subscription requests (represented by a **SubscribeRequest** message) to a NotificationProducer.

The research leading to these results has received funding from the European Community's Seventh Framework Programme

The architecture of the component is shown below


NotificationManager
<<Java Class>> NotificationManagerService
<<Java Class>> NotificationProducer
<<Java Class>> SubscriptionManager
<<Java Class>> PersistenceManager

4.1.1.10 Interface

The methods providing the basic functionality required can be seen below:

Notification Manager	
High Level Description	Notifies analysis services for new experiments and actions held in the VSs. Also, receives and manipulates subscriptions for already registered services.
notifyOfNewExperiment	
void notifyOfNewExperiment (int experimentID, String [] services)	
Description	Sends notification for a new experiment creation to the subscribed services and handles subscriptions.
notifyOfAction	
void notifyOfAction (EndpointReferenceType serviceURL, String notificationMessage, Action action)	

The research leading to these results has received funding from the European Community's Seventh Framework Programme

Notification Manager	
Description	Send notification for a new VS action to the subscribed services.
subscribe	
@WebMethod SubscribeResponse subscribe (Subscribe subscribeRequest)	
Description	Create a subscription for a registered service on a new topic.
unsubscribe	
@WebMethod UnsubscribeResponse unsubscribe (Unsubscribe unsubscribeRequest)	
Description	Unsubscribe from a topic.
pauseSubscription	
@WebMethod PauseSubscriptionResponse pauseSubscription (PauseSubscription pauseSubscriptionRequest)	
Description	Pause an active subscription.
resumeSubscription	
@WebMethod ResumeSubscriptionResponse resumeSubscription (ResumeSubscription resumeSubscriptionRequest)	
Description	Resume a paused subscription.

1.7.4 DataManager

4.1.1.11 Overview

The DataManager component will provide analysis services with an access point to the +Spaces database, allowing them to recover older data that they have lost due to failure. As has been explained above, besides distributing it to the services, the +Spaces platform will store all data regarding the experiments and the actions taking place in the virtual spaces and associated with them. In case of failure, services will poll the DataManager for the actions and experiments that they have missed based on the id of the latest action that they received.

Depending on the number of events (actions or new experiments) that have occurred since the last message received (so, essentially depending on how long the service was down), the Datamanager will either return a list of all events or, in case the number of events exceeds a threshold, a message stating that the list of actions will follow. In the latter case the recovery mechanism will be activated in order to create an xml file containing all the

The research leading to these results has received funding from the European Community's Seventh Framework Programme

events and send them to the service. The threshold will have a default value but will also be able to be defined by the platform administrator via the ConfigurationManager.

Apart from data recovery in case of failure, in which case all events after the time of failure need to be retrieved, the DataManager could also be used for retrieving data for other reasons. One example is if during the running of an experiment a service becomes interested in actions of a type to which it did not subscribe at the beginning of the experiment. In that case, the service will need to subscribe to the NotificationManager for this action type in order to receive future actions but also to poll the DataManager for such actions that occurred before the time of failure. In order to allow for such cases, elaborate polling options will be enabled so that it is possible to retrieve a set of actions based on user defined criteria.

4.1.1.12 Interface

DataManager	
High Level Description	Implements the data recovery mechanism allowing the service to query the database for older experiments or actions
getExperiments	
@WebMethod RecoveredExperimentsReturnType getExperiments(String experimentID)	
Description	Returns all experiments that have been assigned to the service and are newer than the experiment with the specified ID
getActions	
@WebMethod RecoveredActionsReturnType getActions(String actionID)	
Description	Returns all actions that are associated with an experiment assigned to the service, their action type matches the service's subscription and are newer than the actions with the specified ID.
@WebMethod RecoveredActionsReturnType getActions(String actionID, String ExperimentID)	
Description	Returns all actions that are associated with the given experiment, their action type matches the service's subscription and are newer than the actions with the specified ID.
@WebMethod RecoveredActionsReturnType getActions(String actionID, String ExperimentID, ActionType actionType)	
Description	Returns all actions that are associated with the given experiment, are of the given action type and are newer than the actions with the specified ID.

1.7.5 ConfigurationManager

The configuration manager is an internal system management component and is not relevant to specific use case scenarios.

The research leading to these results has received funding from the European Community's Seventh Framework Programme

Overview and motivation: All of the +Spaces middleware components are configurable, and operate differently with different configuration parameters.

The configuration manager is the focal point for system configuration. It contains and exposes all of the configuration parameters for the middleware components.

The configuration manager prevents re-developing configuration mechanisms for each component. Also, it is a single point of administration control, where the entire middleware's behaviour is determined.

Properties types: the configuration properties are divided to two groups:

- **System wide properties**

  This group contains properties that apply to the entire system and are cross-components. These properties will be read and acted upon by several middleware components and do not include information that is component specific. Typical system wide properties include urls, ports, jndi names etc.

- **Components specific properties**

  This group contains properties that are relevant only to specific components. Typical component specific properties include various thresholds, timeout values etc.

Defaults: Configuration parameters have default values.

Persistency: Each parameter is made persistent by the configuration manager.

A snapshot of the configuration values can be taken, saved and then restored, to support quick system-wide configuration changes.

User interface: the configuration manager exposes a GUI allowing an administrator to change various configuration parameters values.

Communication: there are two modes of communicating the configurations values from the configuration manager to the components:

- o **request-response mode**: in this mode a component issues a request to the configuration manager asking for one or more relevant properties values. The configuration manager sends the values in a response. This mode is typically used when a component starts or re-starts it's operation.
  - ▪ This mechanism will use RMI calls.
- o **Notify mode**: In this mode the configuration manager initiates a notification to one or more of the components to announce the change in one or more of the configuration properties.

The research leading to these results has received funding from the European Community's Seventh Framework Programme

- This mechanism will use a JMS BUS.

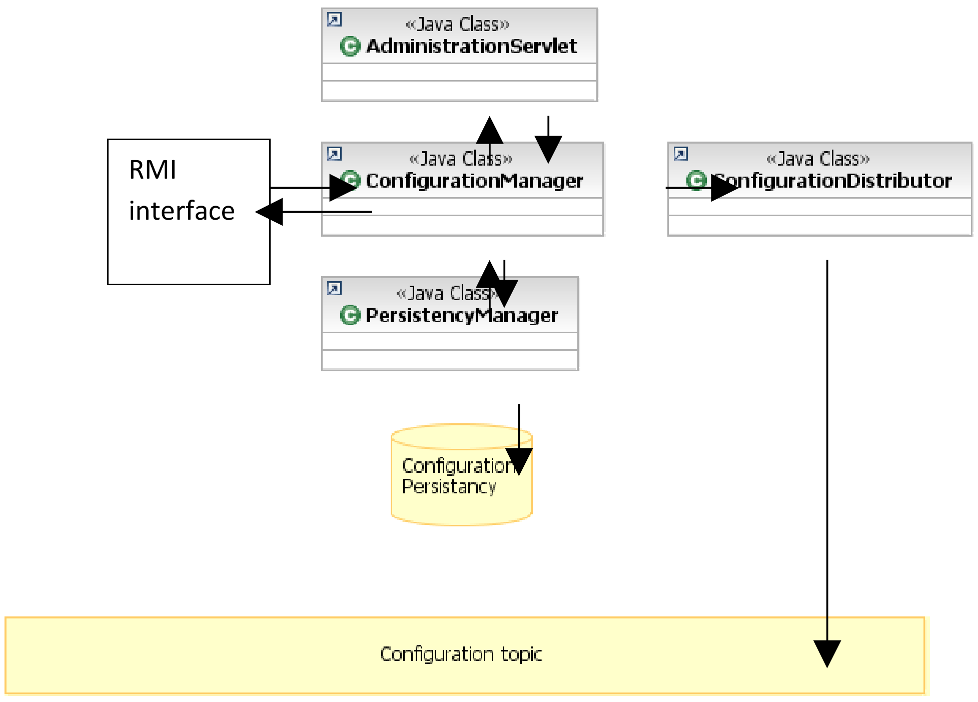


Figure 7: Configuration Manager Design

1.7.6 SLAManager

4.1.1.13 Overview

The SLAManager will create, monitor and evaluate Service Level Agreements between analysis services and government organisations. The SLAs will allow service providers to give QoS guarantees as well as set pricing terms for the usage of their services. The SLAs between the two parties will be based on the agreement offer that the service provider gives when registering and is agreed upon by the government user when selecting the service. The monitoring results describing the service usage that will be used in order to evaluate the SLA and decide whether there has been a violation of the terms or estimate the charges are retrieved from the ExperimentManager.

4.1.1.14 Component design

This component has been designed following the WS-Agreement Specification from the Open Grid Forum (OGF)[9]. Web Services Agreement Specification (WS-Agreement) is a Web Services protocol for establishing agreement between two parties, such as between a service provider and consumer. It defines schemas for specifying agreements, as well as the

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2011] under grant agreement no. 248726

Web Service interface for managing agreement life-cycle, including creation, expiration, and monitoring of agreement states.

In our case, when registering a new Service (e.g. Data Analysis Service), the service provider provides an agreement template, defining guarantee and pricing terms for his service. The Service Manager must invoke the createAgreementTemplate method of the SLAManager, so that the latter can create an agreement offer object and store it to the Database. Upon the specific Service's choice for a new experiment by a Government user, the Experiment Manager retrieves the provider's offer from the SLA Manager through the getAgreementOffer method and the end-user is asked to confirm the agreement offer terms, as defined in this template. Then the Service Manager invokes the createAgreement method of the SLAManager that stores the created SLA into an XML file, based on the WS-Agreement Standards described above. Moreover, a polling service running within this component must periodically request for a list of the active experiments running, as well as for the current value of specific metrics (e.g. Poll votes) from the Experiment Manager, so as to confirm the SLA validity and update the end-user billing.

4.1.1.15 Interface

SLA Manager	
High Level Description	Creates SLAs based on the WS-Agreement specification and periodically checks the validation of the agreements and updates billings.
createAgreement	
CreateAgreementOutputType createAgreement (CreateAgreementInputType agreementOffer)	
Description	Creates a Service Level Agreement and stores it in an XML file.
createAgreementTemplate	
void createAgreementTemplate (EndpointReferenceType serviceURL, AgreementTemplateType agreementOffer)	
Description	Create a new agreement template for the registered service
getAgreementOffer	
AgreementType getAgreementOffer (EndpointReferenceType serviceURL)	
Description	Returns the provider's service agreement offer, so that the end user consents.

The research leading to these results has received funding from the European Community's Seventh Framework Programme

1.8 *Built-in Analysis Services*

In this section the built-in analysis services that will be developed within the project framework are described here. Presented in the following sections are the functionality overview and the interfaces of the services, the detailed internal architecture for them will be outlined in separate deliverables (D.3.5.x).

1.8.1 Data Analysis Service

Overview

The objective of the Data Analysis service is to gather pseudonymized data during experiments and provide intermediate and final results of experiments to the government users. The presentation layer of the data analysis service will provide interactive elements, which allow filtering of results. Data analysis of running experiments will happen in near real-time.

Interface

DataAnalysisService	
High Level Description	Gathers experiment related data in order to create a graphical analysis of experiment results
getDataAnalysisURL	
@WebMethod URL getDataAnalysisURL(String experimentID)	
Description	This method returns the URL that contains the results of the experiment in visual form. The presentation layer of the data analysis service will provide interactive elements, which allow filtering of results. Data analysis of running experiments will happen in near real-time.

1.8.2 Recommendation Service

Overview

The recommendation service subscribes to the middleware (as can be seen in the sequence diagrams) for receiving social network information (user operations with the space and with other users). The information is pseudonymized, as the recommendation service gathers the history of operations of unique users, in order to study their interests and construct their social network.

Interface

RecommendationService	
High Level Description	Provides recommendations for users by analysing the history of users' actions.

The research leading to these results has received funding from the European Community's Seventh Framework Programme

1.8.3 Reputation Service

Overview

Like the recommendation service, the reputation service receives social network information from the system and estimates a reputation rate for users based on their history of actions.

Interface

ReputationService	
High Level Description	Provides reputation rate estimation for users by analysing the history of user's actions.
getReputationForUser	
@WebMethod ReputationRate getReputationForUser – (String userID, ReputationType reputationType)	
Description	Given a user, the service returns a reputation rate for this user, expressing the user's reliability with respect to malicious use. Based on the types of social interactions received from the virtual spaces, additional types of reputation may be defined, such as involvement, influence, and more. In such a case, the call will be given an additional parameter, with the requested type of reputation.
getRecommendationsForUser	
@WebMethod List getUsersWithReputation (ReputationRate reputationRate, ReputationType, reputationType)	
Description	Given a reputation rate (and type, when relevant), the service returns a list of users that match the desired reputation (or have a reputation that is equal or higher than the given rate).

1.9 +Spaces FrontEnd

+Spaces Front End is the presentation layer that lies between the platform and the end users. As most of the common UI, +Spaces front end is a friendly, eye-pleasing and easy to use GUI for the user, and its design will help him/her to exploit the functionalities that will be offered by the +Spaces platform. It has also a project-specific appearance which can be individualized with the use of several skins and skin objects. It is accessible through internet for all potential users, authenticated or not, depending on the services desired. A more detailed analysis of the interface will be included in D3.6.

+Spaces front end is also called e-Gov Front End as it refers to Governmental offices / policy makers' domain of +Spaces end users. To this end, it interacts with +Spaces API in an appropriate way in order to handle, retrieve and display data to the policy makers. In addition, it is also connected with data analysis service to provide the desired statistical reports to the end user. Consequently, e-Gov Front End will present to the policy maker's

The research leading to these results has received funding from the European Community's Seventh Framework Programme

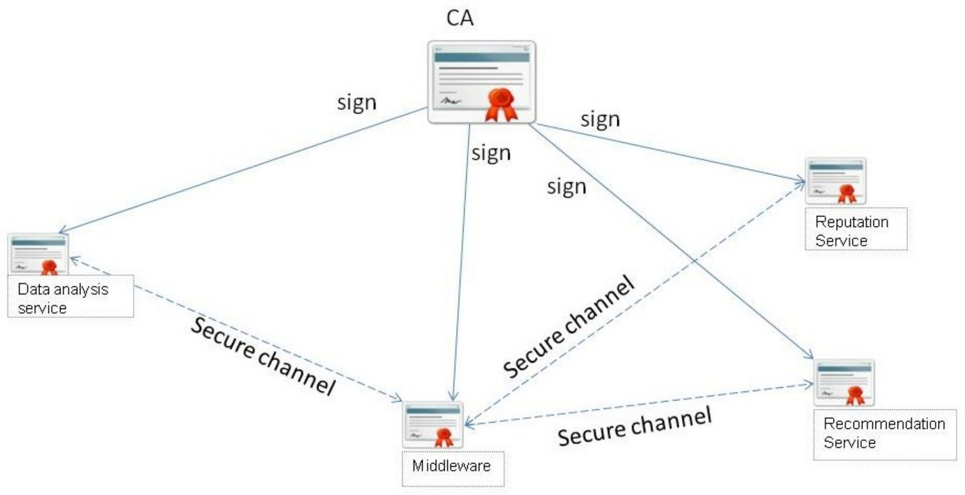
end users domain the functionalities/modules that are defined for all the category's roles (content submitter, debate moderator, administrators). Finally, they will be described in details in D3.6[10] and some of them, reported as examples, are the following: user roles management, login/out functionality, monitoring, system administration, content management, moderation, deployment of an experiment, presentation of statistical reports and data analysis, participation in an experiment etc.

1.10 Platform Security

According to the architecture, the platform will protect the messages exchanged between the middleware and the analysis services (e.g. recommendation, reputation, and data analysis services). To accomplish that , we will use the WS-SecureConversation standard, that works in conjunction with WS-Security , WS-Trust and WS-Policy .

WS-SecureConversation starts with a handshake between the parties involved, in order to establish a security context. A security context provides session based security, rather than establishing new keys for every message. It is based on the concept of session key (like in TLS/SSL) and it is more lightweight than the WS-Security standard, in the case of frequent message exchanges.

The security context can be created by using username tokens, mutual certificates or SAML tokens. In our case, we will use the certificate option. Through a security tool (e.g. Openssl) we will create a certificate authority (CA) which will be trusted by all entities in the system. Each entity will be assigned a new certificate, signed by our CA. Finally, we will use the concept of keystore and trustore. The keystore is the place where each entity keeps the keys that belong to it while trustore is the repository for the trusted CAs. The proper configuration of keystores/trustores can ensure that confidentiality, integrity and authentication are satisfied in our platform.



The research leading to these results has received funding from the European Community's Seventh Framework Programme

Figure 8:+Spaces security through certificates

1.11 Overall Architecture

1.11.1 Component Diagram

Figure 9 depicts the component diagram of the +Spaces platform.

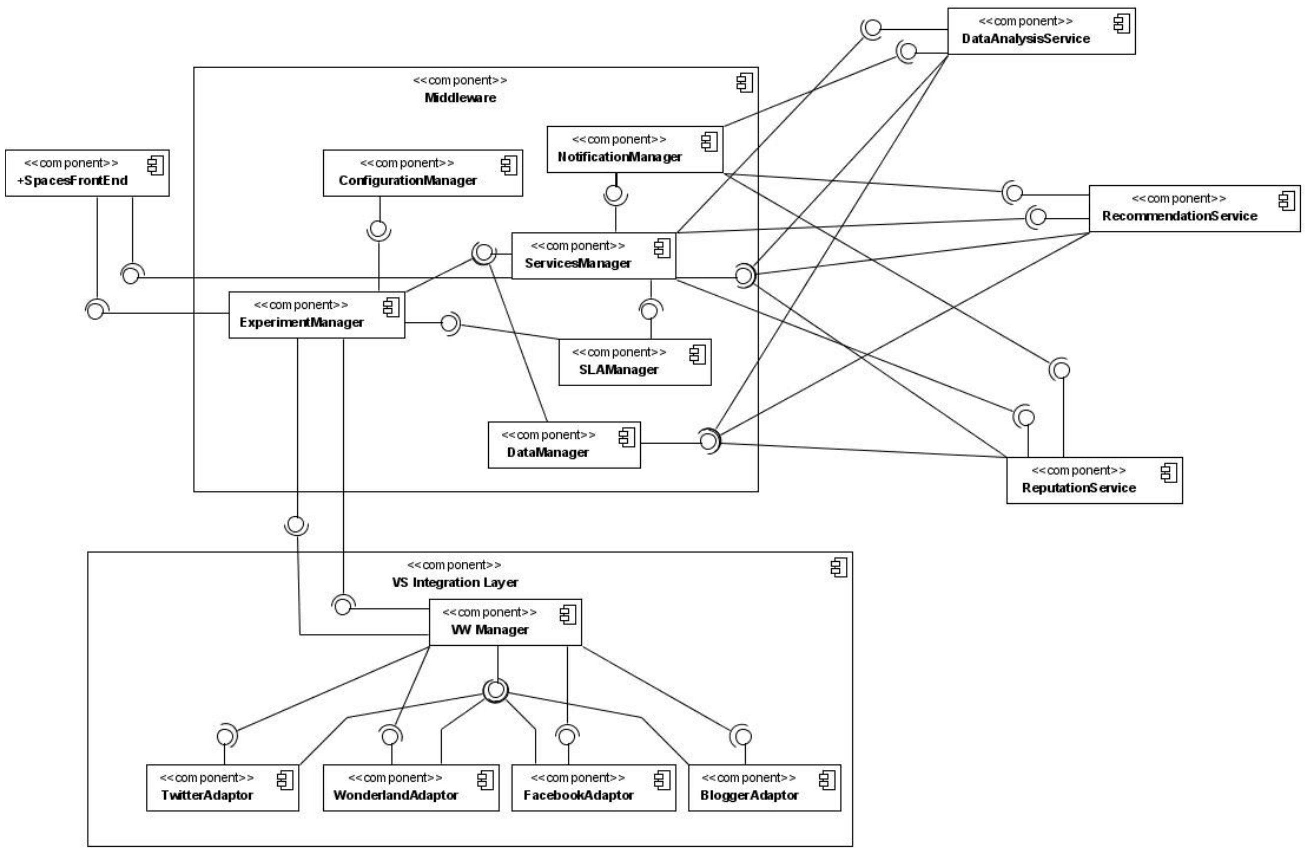


Figure 9:+Spaces component diagram

1.11.2 Interactions between components

In the following sections the sequence of interactions between the components of the +spaces platform are displayed through UML Sequence diagrams.

1.11.2.1 Sequence Diagram 01: Service Registration and Recovery

Figure 10 depicts the sequence of interaction between +Spaces middleware services and analysis services during registration as well as in the case of recovery after service failure. The *Data Analysis Service* is depicted in the diagram but the process is the same for all analysis services, both built-in and external.

The research leading to these results has received funding from the European Community's Seventh Framework Programme

The registration process is initiated by the *Data Analysis Service* that performs a registration request to the platform. As has been explained in 4.1.1.12, apart from the service related data such as the service endpoint and operation description, the registration request may optionally contain an agreement offer describing the promised QoS level as well as the pricing terms for service usage. The agreement offer is forwarded to the *SLAManager* and stored there .

The lower part of the diagram depicts the data recovery process

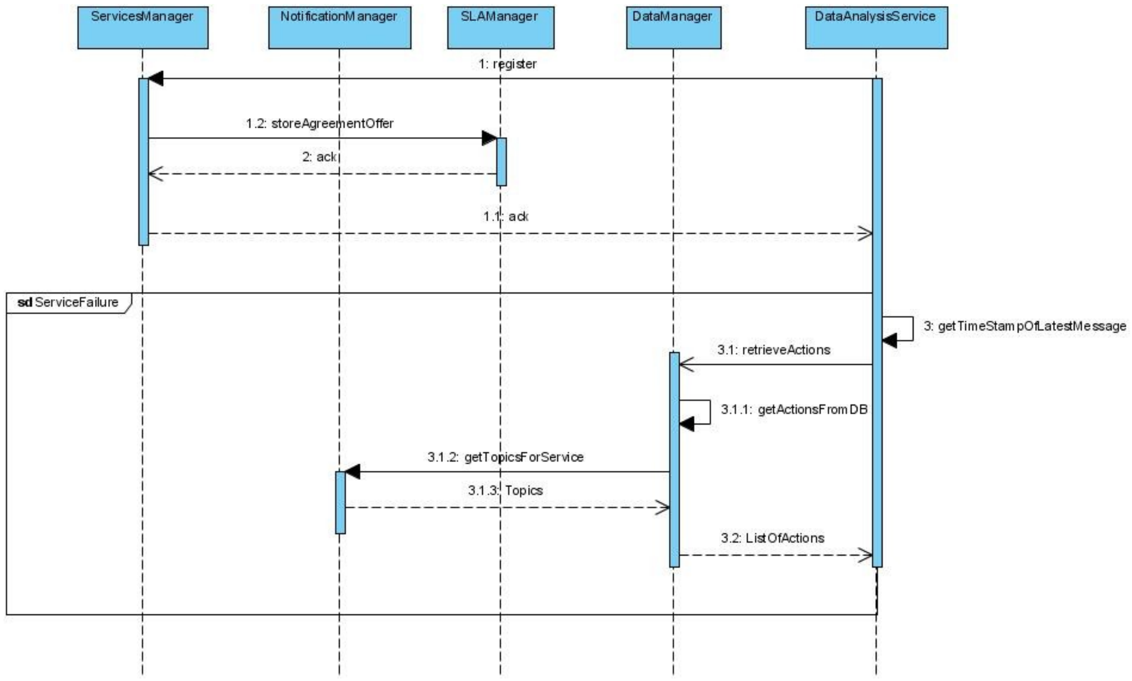


Figure 10: Sequence Diagram 01: Service Registration and Recovery

1.11.2.2 Sequence Diagram 02: Experiment Creation

Figure 11 depicts the sequence of interaction between +Spaces components during experiment creation. The scenario is initiated by the government user who accesses the platform front-end in order to create a new experiment.

When the end-user requests the creation of a new experiment s/he will be asked to select several configuration options as described in D2.2. One of the configuration parameters is the analysis services that will be used for the analysis of the experiment data. In order to get get all the available analysis services, the front-end will invoke the *ServiceManager*. For each registered service, the *ServiceManager* will invoke the *SLAManager* in order to get the agreement offer associated with the use of this service and return a list with all services. The services with the respective description and agreement offer will be returned to the front-end and presented to the user in order to make his/her selection.

When the user selection is made, the front-end will invoke the *CreateExperiment* method of the *ExperimentManager* with the configuration parameters as input parameters. The *ExperimentManager* will then store experiment details and forward the request to the VS

The research leading to these results has received funding from the European Community's Seventh Framework Programme

Manager. For each VS adaptor, the *VSManager* will invoke the respective method for creating a new experiment in the Virtual Space.

After the experiment has been deployed in the Virtual Space and before it has been activated the *ExperimentManager* will invoke the *ServicesManager* in order to assign the selected services to the experiment. This means creating an agreement between the end-user and the analysis service provide based on the agreement offer proposed by the service provider (and accepted by the end-user by selecting the service) and then notifying the service of the experiment creation.

Before activating the experiment, the *ExperimentManager* will also ask the *ServicesManager* to poll the Recommendation Service for a list of potentially interested users for this specific experiment. Based on the experiments description tags and the users' history the Recommendation will come-up with a list of users that are more likely to be willing to participate to the experiment and to whom the new experiment will be advertised. The list of users will then be passed to the *VSManagement* layer. The experiment will subsequently be activated on all Virtual Spaces and the users will be able to participate.

The research leading to these results has received funding from the European Community's Seventh Framework Programme

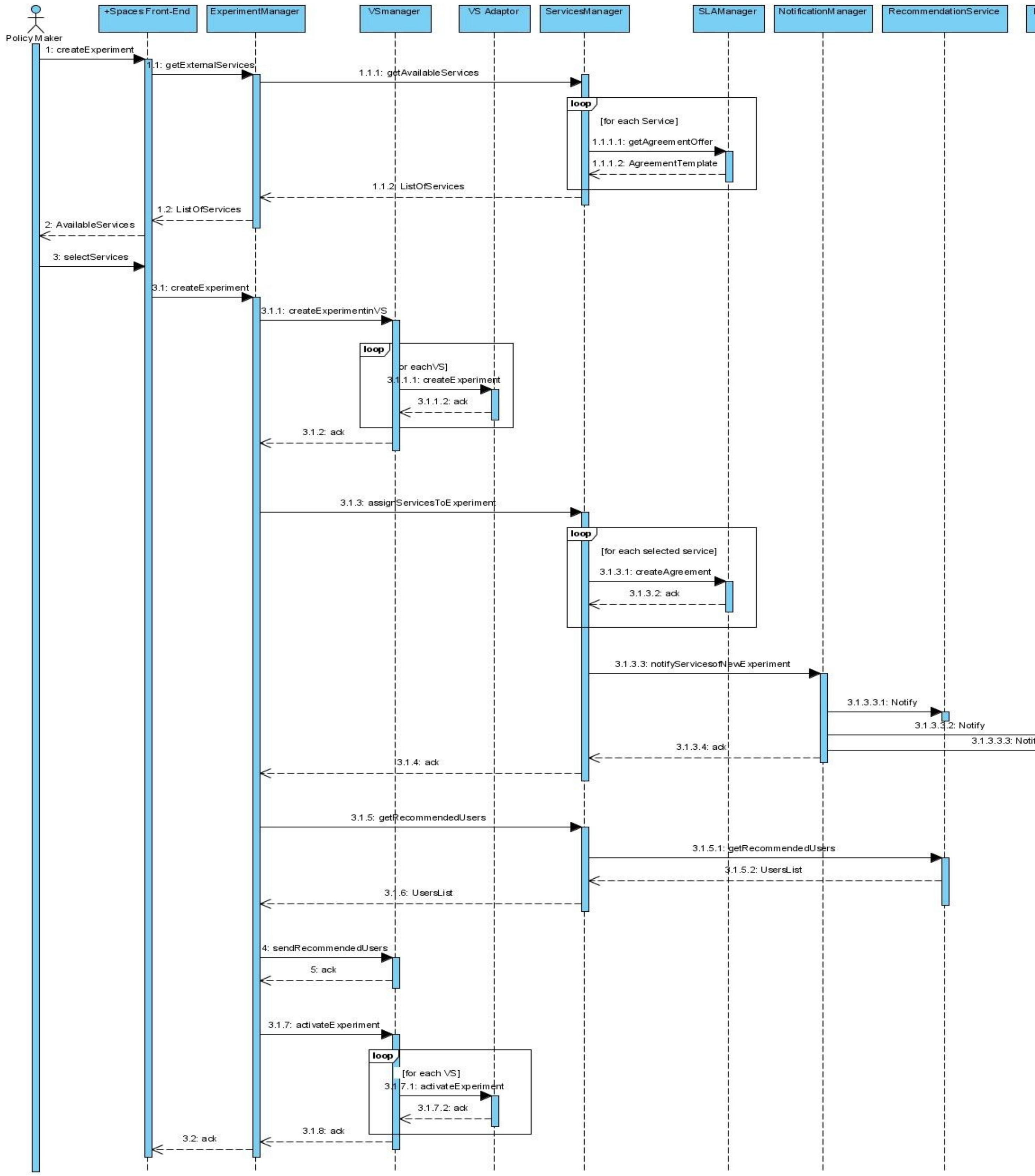


Figure 11: Sequence Diagram 02: Experiment Creation

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2011] under grant

1.11.2.3 Sequence Diagram 03: Actions Distribution

Figure 12 describes the sequence of interactions between the components that are involved in the distribution of actions taking place in a Virtual Space to the analysis services.

The sequence is initiated when an action is performed in a Virtual Space. Depending on the virtual space implementation the adaptor will either forward each action to the adaptor or the adaptor will poll the virtual space API for actions. The adaptor will then forward the action to the *VSManager*. After de-identification of the action which is done by replacing the user id with a pseudonym, the *VSManager* will forward the action to the +Spaces platform and more specifically the *ExperimentManager*.

The *ExperimentManager* will store the actions details and poll the *ConfigurationManager* in order to get the action distribution policy associated with this particular experiment. The policy can be either to send each action as a separate message or to send the actions periodically. The policy may also contain a clause, for example depending on the rate of incoming actions the *ExperimentManager* may need to switch between the two policies. After getting the policy from the *ConfigurationManager*, the *ExperimentManager* will evaluate the policy and associated clause and decide on how to send the actions. If the policy has changed since the last action sent, the *ExperimentManager* will notify the *ServicesManager* on the new policy in order for the two services to be in synch.

If the policy is to send each action separately the policy will send the action directly to the *ServicesManager*. Otherwise it will gather actions and periodically forward them to the *ServicesManager*. Upon receiving the action(s) the *ServicesManager* will retrieve the analysis services associated with the experiment and invoke the *NotificationManager* which will then filter the actions based on the subscriptions of the analysis services (see 4.1.1.7) and send notifications for each action to the services.

The research leading to these results has received funding from the European Community's Seventh Framework Programme

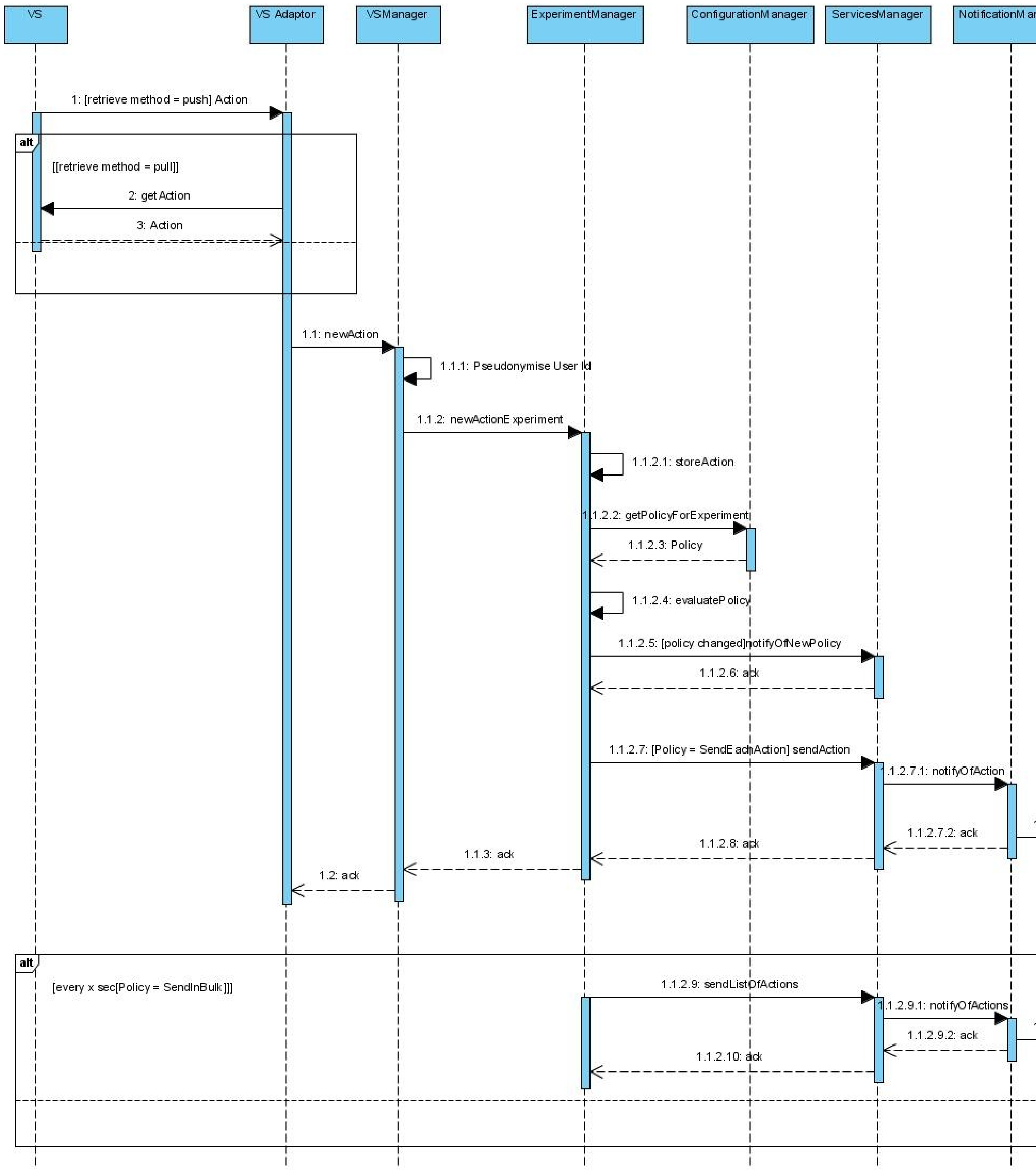

VS
VS Adaptor
VSManager
ExperimentManager
ConfigurationManager
ServicesManager
NotificationMan
1: [retrieve method = push] Action
alt
[[retrieve method = pull]]
2: getAction
3: Action
1.1: newAction
1.1.1: Pseudonymise User Id
1.1.2: newActionExperiment
1.1.2.1: storeAction
1.1.2.2: getPolicyForExperiment
1.1.2.3: Policy
1.1.2.4: evaluatePolicy
1.1.2.5: [policy changed]notifyOfNewPolicy
1.1.2.6: ack
1.1.2.7: [Policy = SendEachAction] sendAction
1.1.2.7.1: notifyOfAction
1.1.2.7.2: ack
1.1.2.8: ack
1.1.3: ack
1.2: ack
alt
[every x sec[Policy = SendInBulk]]]
1.1.2.9: sendListOfActions
1.1.2.9.1: notifyOfActions
1.1.2.9.2: ack
1.1.2.10: ack

Figure 12: Sequence Diagram 03: Actions Distribution

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2011] under grant

1.11.2.4 Sequence Diagram 04: Experiment Results

The sequence of interactions taking place when the end user accesses the front-end in order to retrieve the results is depicted in Figure 13.

The *+Spaces Front-End* polls the *ExperimentManager* in order to get back the results. The *ExperimentManager* will retrieve the experiment details from the database and poll the *ServicesManager* for the *DataAnalysisService* that has been assigned this experiment. The *ExperimentManager* will return this information to the Front-End which in turn will poll the *DataAnalysisService* with the experiment ID in order to get the web page with the graphic results and present them to the user.

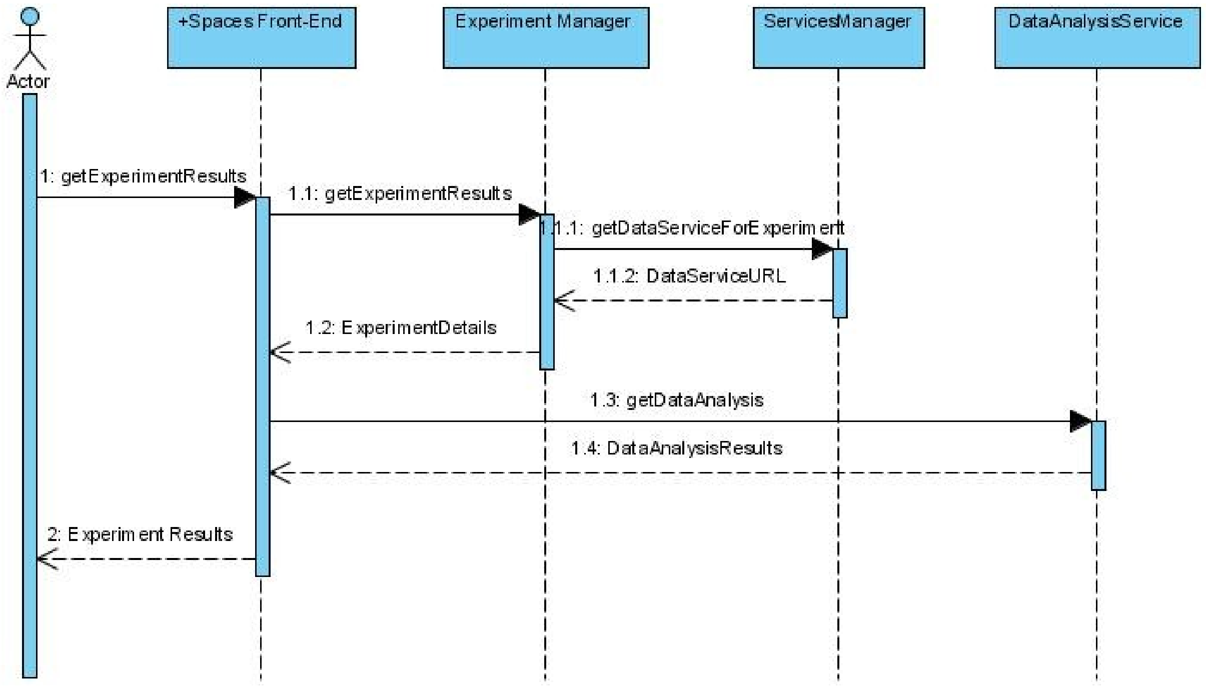


Figure 13: Sequence Diagram 04: Experiment Results

1.11.2.5 Sequence Diagram 05: Recommendations

The Recommendations sequence diagram depicted in Figure 14 is initiated when the VS adaptor wants to find recommendations for a specific user. The adaptor will poll the VS Manager which in turn will de-identify the user ID by replacing with a pseudonym and ask the *ServicesManager* for recommendations. The *ServicesManager* will poll the *RecommendationService* and return a list of experiments that might potentially be of interest to the user.

The research leading to these results has received funding from the European Community's Seventh Framework Programme

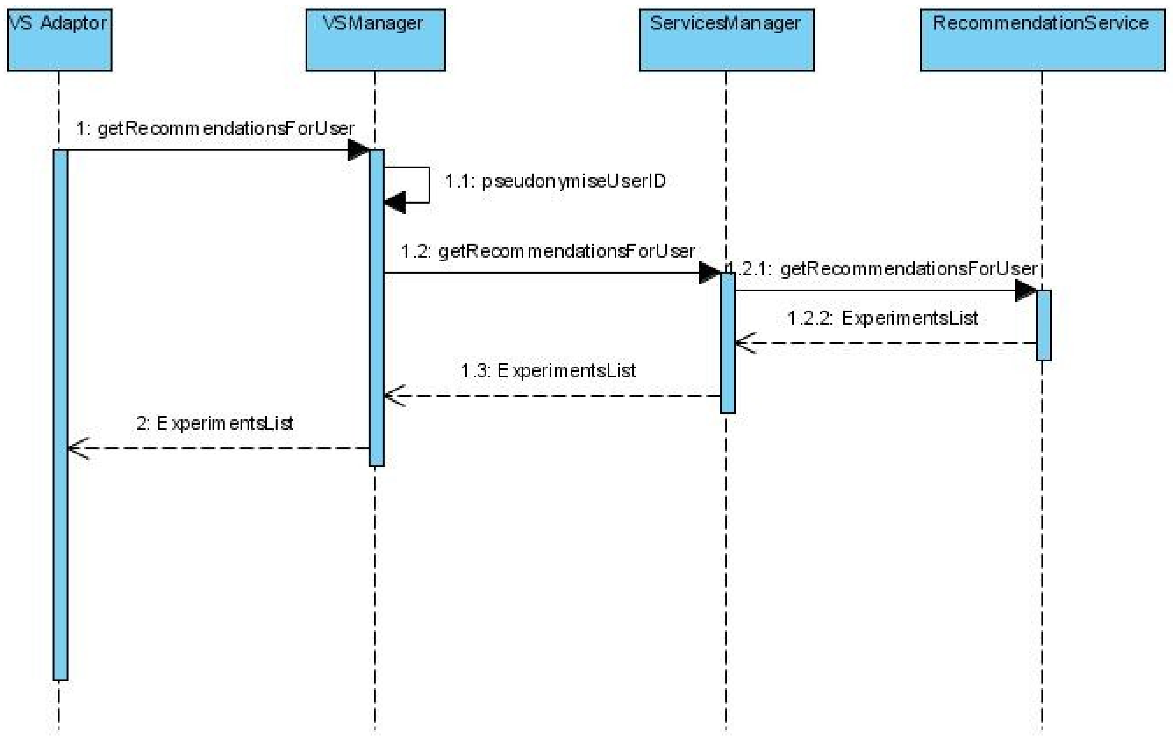


Figure 14: :Sequence Diagram 05: Recommendations

1.11.2.6 Sequence Diagram 06: Reputation

When the DataAnalysisService wants to retrieve the reputation rating for a specific user that participated in an experiment or to get all users that have a reputation above a specific threshold, it polls the ServicesManager. The ServicesManager retrieves the ReputationService that is responsible for this experiment and polls it for the reputation of the specific user or all users with a high reputation. The ServiceManager then sends the reputation results to the DataAnalysisService.

The sequence is depicted in Figure 15.

The research leading to these results has received funding from the European Community's Seventh Framework Programme

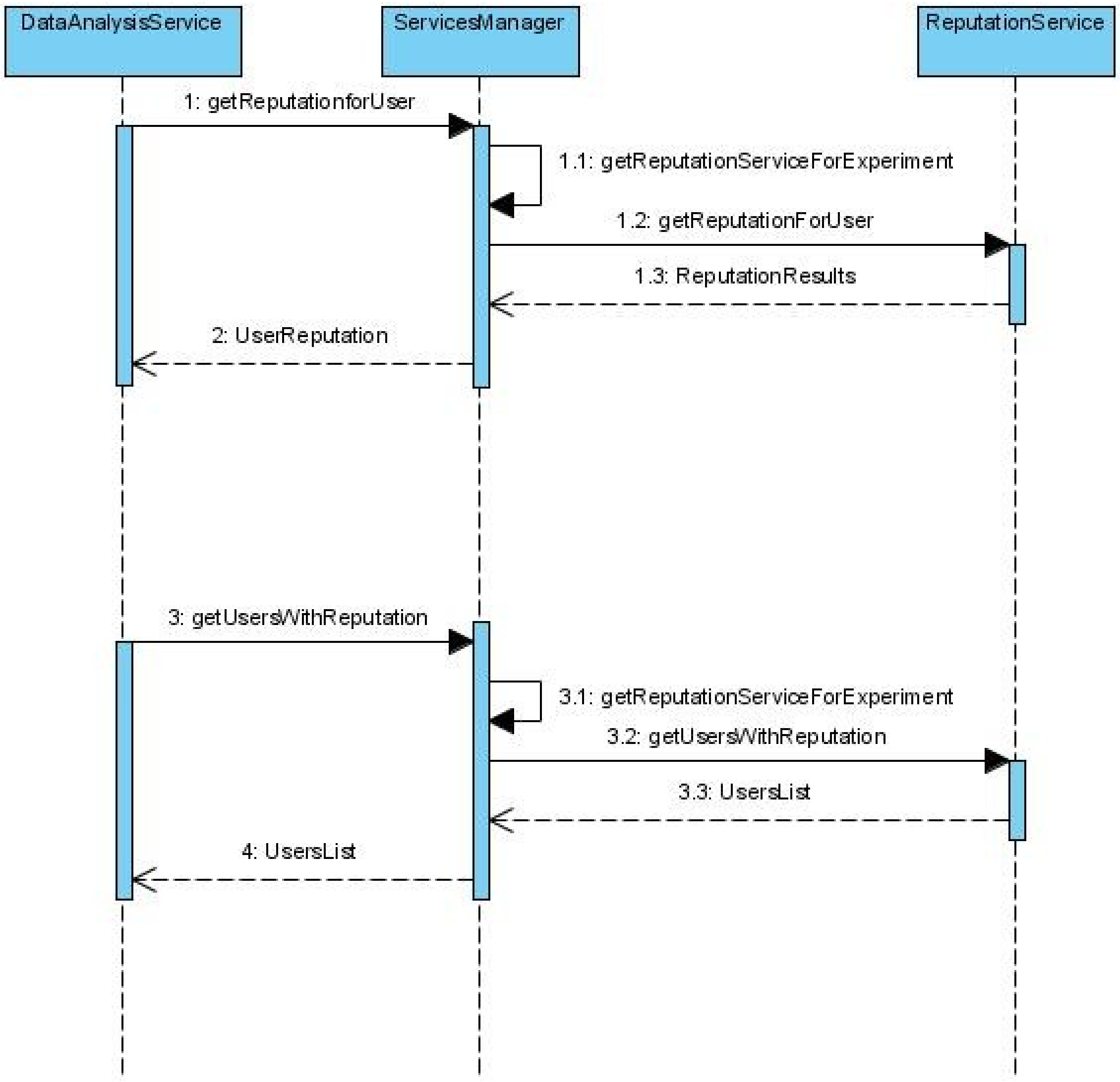

DataAnalysisService
ServicesManager
ReputationService
1: getReputationforUser
1.1: getReputationServiceForExperiment
1.2: getReputationForUser
1.3: ReputationResults
2: UserReputation
3: getUsersWithReputation
3.1: getReputationServiceForExperiment
3.2: getUsersWithReputation
3.3: UsersList
4: UsersList

Figure 15 :Sequence Diagram 06: Reputation

1.12 Data Schemata

The data XML schemata for the representation of entities used in the +Spaces project will be outlined in this section. As this is still an early stage only data types that can be defined at this point are presented here, some are left to be presented at later deliverables (i.e. the component implementation reports). Even in the data types defined here modifications may be necessary at a later point and these will be included as well in the aforementioned reports.

1.12.1 EndpointReferenceType

name	type	description	N	Reference
address	AttributedURITy	URI of the	[1..1]	4.1.1.15

The research leading to these results has received funding from the European Community's Seventh Framework Programme

	pe	EndpointReference		
metadata	MetadataType	Metadata of the EndpointReference	[1..1]	4.1.1.15
referenceParameters	ReferenceParametersType	Other Reference Parameters of the EndpointReference	[1..1]	4.1.1.15
any	List<Object>	List of any objects	[1..1]	

1.12.2 AttributedURIType

name	type	description	N	Reference
value	String	String value of the URI	[1..1]	

1.12.3 MetadataType

name	type	description	N	Reference
any	List<Object>	List of any objects	[1..1]	

1.12.4 ReferenceParametersType

name	type	description	N	Reference
any	List<Object>	List of any objects	[1..1]	

1.12.5 Subscribe

name	type	description	N	Reference
consumerReference	EndpointReferenceType	EndpointReference of the notification consumer	[1..1]	4.1.1.15
filter	FilterType	Filters the subscription topic	[1..1]	4.1.1.15
initialTerminationTime	GregorianCalendar	Subscription termination date	[1..1]	
subscriptionPolicy	SubscriptionPolicyType	Subscriber requirements for policy	[1..1]	4.1.1.15

1.12.6 FilterType

name	type	description	N	Reference

The research leading to these results has received funding from the European Community's Seventh Framework Programme

topic	String	Subscription topic	[1..1]	

1.12.7 SubscriptionPolicyType

name	type	description	N	Reference
messageRate	int	Defines notification message rate	[1..1]	

1.12.8 SubscribeResponse

name	type	description	N	Reference
subscriptionReference	EndpointReferenceType	EndpointReference of the subscription	[1..1]	4.1.1.15
currentTime	XMLGregorianCalendar	Subscription time	[1..1]	
terminationTime	XMLGregorianCalendar	Subscription termination date	[1..1]	

1.12.9 Unsubscribe

name	type	description	N	Reference
consumerReference	EndpointReferenceType	EndpointReference of the notification consumer	[1..1]	4.1.1.15
subscriptionReference	EndpointReferenceType	EndpointReference of the subscription	[1..1]	4.1.1.15

1.12.10 UnsubscribeResponse

name	type	description	N	Reference
subscriptionReference	EndpointReferenceType	EndpointReference of the deleted subscription	[1..1]	4.1.1.15
currentTime	XMLGregorianCalendar	Unsubscription time	[1..1]	

1.12.11 PauseSubscription

name	type	description	N	Reference
consumerReferenc	EndpointReferenceType	EndpointReference	[1..1]	4.1.1.15

The research leading to these results has received funding from the European Community's Seventh Framework Programme

e		of the notification consumer		
subscriptionReference	EndpointReferenceType	EndpointReference of the subscription	[1..1]	4.1.1.15

1.12.12 PauseSubscriptionResponse

name	type	description	N	Reference
subscriptionReference	EndpointReferenceType	EndpointReference of the paused subscription	[1..1]	4.1.1.15
currentTime	XMLGregorianCalendar	subscription pause time	[1..1]	

1.12.13 ResumeSubscription

name	type	description	N	Reference
consumerReference	EndpointReferenceType	EndpointReference of the notification consumer	[1..1]	4.1.1.15
subscriptionReference	EndpointReferenceType	EndpointReference of the paused subscription	[1..1]	4.1.1.15

1.12.14 ResumeSubscriptionResponse

name	type	description	N	Reference
subscriptionReference	EndpointReferenceType	EndpointReference of the resumed subscription	[1..1]	4.1.1.15
currentTime	XMLGregorianCalendar	subscription resume time	[1..1]	

1.12.15 Notify

name	type	description	N	Reference
message	NotificationMessage	message	[1..1]	4.1.1.15

1.12.16 NotificationMessage

name	type	description	N	Reference

The research leading to these results has received funding from the European Community's Seventh Framework Programme

subscriptionReference	EndpointReferenceType	EndpointReference of the subscription	[0..1]	4.1.1.15
topic	String	Subscription topic	[0..1]	
producerReference	EndpointReferenceType	EndpointReference of the notification producer	[0..1]	4.1.1.15
message	String	Textual description of notification	[1..1]	

1.12.17 CreateAgreementInputType

name	type	description	N	Reference
agreementInitiator	EndpointReferenceType	EndpointReference of the agreement initiator	[1..1]	4.1.1.15
agreement	AgreementType	Agreement object on which the SLA will be created	[1..1]	External Reference[11]

1.12.18 CreateAgreementOutputType

name	type	description	N	Reference
agreementEPR	EndpointReferenceType	EndpointReference of the agreement	[1..1]	4.1.1.15

1.12.19 RegisteredService

name	type	description	N	Reference
ID	String	Service ID	[1..1]	
name	String	Service name	[1..1]	
serviceEPR	EndpointReferenceType	EndpointReference of the service	[1..1]	4.1.1.15
description	String	Service description	[0..1]	
agreement	AgreementType	Agreement object that sets the terms of usage of the service	[1..1]	External Reference[12]

1.12.20 RegistrationInputType

name	type	description	N	Reference

The research leading to these results has received funding from the European Community's Seventh Framework Programme

name	String	Service name	[1..1]	
serviceEPR	EndpointReferenceType	EndpointReference of the service	[1..1]	4.1.1.15
description	String	Service description	[0..1]	
agreement	AgreementType	Agreement object that sets the terms of usage of the service	[1..1]	External Reference[13]
agreement	AgreementType	Agreement object that sets the terms of usage of the service	[1..1]	External Reference[14]

1.12.21 RegistrationOutputType

name	type	description	N	Reference
success	boolean	Was the Registration Successful?	[1..1]	
description	String	Justification in case of failure	[0..1]	

1.12.22 RecoveredExperimentsReturnType

name	type	description	N	Reference
mode	Enum <String> {"RESULTS_INCLUDED", "RESULTS_TO_FOLLOW"}	Are the results included in this message as a list or will they be sent separately	[1..1]	
experiments	List <Experiment>	List of experiments, null if mode = "RESULTS_TO_FOLLOW"	[0..1]	
description	String	Justification	[0..1]	

1.12.23 RecoveredActionsReturnType

name	type	description	N	Reference
mode	Enum <String> {"RESULTS_INCLUDED", "RESULTS_TO_FOLLOW"}	Are the results included in this message as a list or will they be sent	[1..1]	

The research leading to these results has received funding from the European Community's Seventh Framework Programme

		separately		
actions	List <Action>	List of experiments, null if mode = "RESULTS_TO_FOLLOW"	[0..1]	
description	String	Justification	[0..1]	

1.12.24 PolicyType

name	type	description	N	Reference
PolicyType	Enum <String> {"SEND_EACH_ACTION", "SEND_IN_BULK"}	Type of actions forwarding policy	[1..1]	

1.12.25 AnalysisServiceType

name	type	description	N	Reference
ServiceType	Enum <string> {"DATA_ANALYSIS", "REPUTATION', "RECOMMENDATION"}	Type of analysis service	[1..1]	

1.12.26 Evidence

name	type	description	N	Reference
relatedPeople	<List> Person	List of related people	[1..1]	"Person data" type to be defined in 3.5.1
relatedTags	<List> Tag	List of related tags	[1..1]	"Tag" data type to be defined in 3.5.1

1.12.27 Recommendation

name	type	description	N	Reference
title	String	Title of recommendation	[1..1]	

The research leading to these results has received funding from the European Community's Seventh Framework Programme

like	URL	Recommended item	[1..1]	
source	String	Name of the virtual space from which the recommendation came	[0..1]	
score	float	Score of recommendation (between 0 and 1)	[1..1]	
evidence	Evidence	why the recommendation is given to the user	[1..1]	4.1.1.15

1.12.28 Experiment

name	type	description	N	Reference
name	String	Name of Experiment	[1..1]	
user	String	ID of user who created the experiment	[1..1]	
spaces	List <String>	IDs of the virtual Spaces where the experiments are deployed	[1..1]	
description	String	Free text description of the experiment	[1..1]	
configuration	Collection <ConfigurationParameter>	Set of configuration parameters	[1..1]	To be defined in 3.4.1
tags	List <String>	List of topics relevant to an experiment	[1..1]	

1.12.29 Action

name	type	description	N	Reference
user	String	ID of user who performed the action (pseudonymized)	[1..1]	

The research leading to these results has received funding from the European Community's Seventh Framework Programme

space	Strin>	IDs of the virtual space where the action took place	[1..1]	
type	Enum <String>	Action type	[1..1]	
description	String	Structured textual description of the action	[1..1]	

5 Conclusion

The current document has provided a detailed description of the initial architecture of the +Spaces platform design. Any changes and deviations from these architecture specifications will be included in the individual component implementation reports that will be released together with the component implementations. The first version of these deliverables is D3.3.1 Middleware Components Report, which is due in M12.

6 References

The research leading to these results has received funding from the European Community's Seventh Framework Programme

[1] +Spaces Project, " D2.2 Functional Specifications", ATC and other partners, June 2010

[2] +Spaces Project, " D2.3 Ethical Issues", KULueven, June 2010

[3] OASIS Service Oriented Architecture Reference Model http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm

[4] I. Jacobson, G. Booch, J. Rumbaugh, "The Unified Software Development Process", Addison-Wesley, 1999.

[5] J. Hunt, "Guide to the Unified Process featuring UML, Java and Design Patterns, Springer Professional Computing", Springer Verlag, Sept. 2003, page 33.

[6] +Spaces Project, " D2.2 Functional Specifications", ATC and other partners, June 2010

[7] http://en.wikipedia.org/wiki/Pseudonymization

[8] Web Services Base Notification Specification 1.3 (WS-BaseNotification), OASIS Standard, 1 October 2006, http://docs.oasis-open.org/wsn/wsn-ws_base_notification-1.3-spec-os.pdf

[9] Web Services Agreement Specification (WS-Agreement), OGF Standard, 14 March 2007, http://www.ogf.org/documents/GFD.107.pdf

[10] +Spaces Project, D3.6, "+Spaces API".

[11] WS-Agreement Schema: http://*schemas*.ggf.org/graap/2007/03/*ws-agreement*

[12] WS-Agreement Schema: http://*schemas*.ggf.org/graap/2007/03/*ws-agreement*

[13] WS-Agreement Schema: http://*schemas*.ggf.org/graap/2007/03/*ws-agreement*

[14] WS-Agreement Schema: http://*schemas*.ggf.org/graap/2007/03/*ws-agreement*