© All Rights Reserved

SEVENTH FRAMEWORK PROGRAMME

THEME ICT-2009.1.2

"Internet of Services, Software and Virtualization"

D4.2.1
Integrated SocIoS Platform

Project acronym: SocIoS

Project full title: *Exploiting Social Networks for Building the Future Internet of Services*

Contract no.: 257774

Workpackage:	WP4	Integration and Testing	
Editor:	I. Spais		ATC
Author(s):	P. Brigden		ATC
	T. Papaoikonomou		ICCS/NTUA
	D. Geifman		UH
	S. Laurière		Cognium
	S. Kremer-Davidson		IBM
Authorized by	D. Varvarigou		ICCS/NTUA
Doc Ref:			
Reviewer	G. Papadakis		ICCS/NTUA
Dissemination Level	PU		

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2011] under grant agreement no. 257774

© All Rights Reserved

SocIos Consortium

Beneficiary Number	Beneficiary name	Beneficiary short name	Country	Date enter project	Date exit project
1(coordinator)	Institute of Communication and Computer Systems/National Technical University of Athens	ICCS/NTUA	Greece	Month 1	Month 30
2	IBM Haifa Research Lab	IBM	Israel	Month 1	Month 30
3	Athens Technology Center	ATC	Greece	Month 1	Month 30
4	Google Ireland Limited	Google	Ireland	Month 1	Month 30
5	Cognium Systems	Cognium	France	Month 1	Month 30
6	Center for the Study of the Information Society, University of Haifa	HU	Israel	Month 1	Month 30
7	Deutsche Welle	DW	Germany	Month 1	Month 30
8	Stefi Productions S.A.	Stefi	Greece	Month 1	Month 30
9	Katholieke Universiteit Leuven (K.U.Leuven) – Interdisciplinary Centre for Law and ICT	KULeuven	Belgium	Month 1	Month 30

Document History

Version	Date	Changes	Author/Affiliation
v.0.1	08-11-2011	Draft version distributed	ATC
v.0.2	26-02-2012	Revised version distributed	ATC
v.0.3	12-03-2012	Prefinal version	ATC
v.0.4	13/03/2012	Incorporate comments received from partners	ATC
v.0.5	14/03/2012	Final Version	ATC, ICCS

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2011] under grant agreement no. 257774

© All Rights Reserved

Executive Summary

This is a supplementary document that describes the software environment for performing the functionalities exposed by the 1st integrated SocIoS prototype. The first version of the prototype is described. The components included in the platform will continue to evolve, while the first pilot is running, and the final version will be described in a later deliverable.

As this is a prototype deliverable, the source code of the following software modules included in the prototype are also delivered:

- Core services framework,
- Event Detection service,
- Social Filtering Service,
- Flexi Price,
- Translation service and
- Front End

IBM delivered the code (for Facebook and Twitter adaptors and for Recommendation and Reputation Services) to ATC under a special license agreement. No other partner or legal entity can have access or can use IBM code.

The platform exposing the 1st SocIoS prototype can be accessible via the following link:

http://frontend.sociosproject.eu/Scenario1.aspx

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2011] under grant agreement no. 257774

© All Rights Reserved

Table of Contents

Executive Summary 3

1 Introduction 1

1.1 About this report 1

1.2 Structure of the document 1

2 Core Services Framework 2

2.1 Overview 2

2.2 Specification on components' integration - interfaces 2

2.3 Technical requirements of the tool 4

2.4 Installation and Usage Guidelines 4

2.4.1 Installing the core services 4

3 Recommendation Service 5

3.1 Overview 5

3.2 Specification on components' integration - interfaces 6

a) getRecommendedPersons 6

b) getRecommendedPersonsPerGroup 7

3.3 Technical requirements of the tool 8

3.4 Installation and Usage Guidelines 9

3.4.1 Installation Instructions 9

3.4.2 Installing the service 9

3.4.3 Usage Guidelines 10

4 Reputation Service 11

4.1 Overview 11

4.2 Specification on components' integration - interfaces 11

a) getPersonReputationScore 11

4.3 Installation and Usage Guidelines 12

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2011] under grant agreement no. 257774

4.3.1 Installing the service 12

4.3.2 Usage Guidelines 12

5 Event Detection 13

5.1 Overview 13

5.2 Specification on components' integration – interfaces 14

5.3 Technical requirements of the tool 14

5.4 Installation and Usage Guidelines 15

5.4.1 Installing the service 15

5.4.2 Usage Guidelines 15

6 Social Filtering Service 16

6.1 Overview 16

6.2 Specification on components' integration - interfaces 16

a) GroupService 16

addMembers /removeMembers 16

createGroup 17

deleteGroup 17

getGroupMembers 18

getGroups 18

b) SocialFilteringService 18

getRecentActivities 18

6.3 Technical requirements of the tool 19

6.4 Installation and Usage Guidelines 20

6.4.1 Usage Guidelines 20

7 FlexiPrice 21

7.1 Overview 21

7.2 Specification on components' integration - interfaces 22

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2011] under grant agreement no. 257774

a) assign_price 22

b) get_transactions 23

7.3 Technical requirements of the tool 24

7.4 Installation and Usage Guidelines 25

8 Translation Service 26

8.1 Overview 26

8.2 Specification on components' integration - interfaces 26

8.3 Technical requirements of the tool 26

8.4 Installation and Usage Guidelines 27

8.4.1 Installing the service 27

8.4.2 Usage Guidelines 27

9 Front End 28

9.1 Overview 28

9.2 Specification on components' integration - interfaces 28

9.3 Technical requirements of the tool 28

9.4 Installation and Usage Guidelines 29

9.4.1 Installing the component 29

9.4.2 Usage Guidelines 29

List of Figures

Figure 1. Dependencies on other SocIoS services 5

List of Tables

Table 1. Core services framework 3

Table 2. Technical requirements of the core services framework 4

Table 3. Recommendation service: getRecommendedPersons 6

Table 4. Recommendation service: getRecommendedPersonsPerGroup 7

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2011] under grant agreement no. 257774

Table 5. Technical requirements of recommendation service 8

Table 6. Reputation service: getPersonReputationScore 11

Table 7. Interface of Event Detection service 14

Table 8. Technical requirements of Event Detection service 14

Table 9. Social Filtering service: addMembers/removeMembers 16

Table 10. Social Filtering service: createGroup 17

Table 11. Social Filtering service: deleteGroup 17

Table 12. Social Filtering service: getGroupMembers 18

Table 13. Social Filtering service: getGroups 18

Table 14. Social Filtering service: getRecentActivities 19

Table 15. Technical requirements of Social Filtering service 19

Table 16. FlexiPrice: assign_price 22

Table 17. FlexiPrice: get_transactions 23

Table 18. Technical requirements of FlexiPrice service 24

Table 19. Technical requirements of Translation service 26

Table 20. Technical requirements of Front-End 28

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2011] under grant agreement no. 257774

1 Introduction

1.1 About this report

The SocIoS integrated prototype incorporates a mixture of customization and adaptation of existing services and toolsets and integrates all modules in two platform instances (an intermediate and a final one). This deliverable describes the intermediate instance, focusing on the SocIoS components that were developed in order to expose SocIoS functionalities according to the pilot scenarios described in the deliverable D5.1.1. "Pilot Scenarios".

The 1st version of the SocIoS prototype consists of the following components:

- Core services framework
- Integrated Auxiliary services
    - Recommendation service
    - Reputation service
    - Event detection service
    - Social Filtering service
    - Translation service
- Non-Integrated Auxiliary service
    - The Flexi-Price service
- SocIoS Front-End

All the components are installed on ATC server(s). Thus, a simplified architecture was adopted and relevant suggestion(s) from the latest review meeting (2nd review meeting, Brussels 3rd of October 2011) were tackled.

1.2 Structure of the document

Each one of the previous components will be shortly described in the following sections focusing on the technical aspects that were considered in order to integrate them in the SocIoS functional workflow. Section 2 describes the Core Services Framework, mentioning the link where an instance of the code is hosted, the developments tools used to build the component, installation instructions, technical requirements and specifications on component's interfaces. Section 3 consists of the same sub-sections, but it describes the Recommendation service, while Section 5 refers to the Reputation service. The functionality of the Event Detection service and of FlexiPrice are described from a technical point of view in Sections 6 and 7, respectively. The last two sections describe the Translation service and Front End component, respectively.

2 Core Services Framework

2.1 Overview

The Core Services component acts as the main integration point for the underlying social network adaptors. It is responsible for receiving the client requests (where by client, we denote any auxiliary service) and distributing it appropriately to the various adaptors. It then collects the responses from them which are sent back to the client as a single list.

The Core Services component exposes a SOAP web services endpoint, from which it can be consumed. The code is available at:

http://gforge.grid.ece.ntua.gr/svn/socios/trunk/SociosApiIllustrate

2.2 Specification on components' integration - interfaces

All the adaptors share a common interface built by Google. The rationale was to avoid duplicated work, to increase visibility on each other's code/approach on the individual adaptors, to accelerate the integration, to provide a simple way of adding new adaptors in the future – just implement a simple java interface – and to factor out obvious common code, eg. web services implementation, retrieve credentials.

The SocIoS Core Services component communicates with the other components through a well-defined set of operations that implement the SocIoS Data model. The input and output messages were defined at the second Appendix of the D2.5.3 deliverable, while the methods were discussed at Section 3 of the same deliverable.

The following table summarizes the available methods along with their support on each social network and the authentication requirements. It must be mentioned that, not all of the methods are supported from all the adaptors. This can be seen via the red "X" image.

Depending on the API implementation of each Social Network, authentication (use of an access token) may or may not be required when making API calls. In the APIs authentication is mainly used to protect users' personal information. For example in Facebook you need an access token in order to retrieve a person's friends. This is not the case for YouTube however, as people's contacts is publicly available information. In the following table the "Authent. required" column represents which social networks require authentication and which don't.

It is important to note that the following table refers to authentication requirements of the APIs and not that of a SocIoS user. For example in order to call FindMediaItems with the Facebook Adaptor there would need to be an access_token present in order for it to be used in the Facebook API call. This access_token will not be requested by the SocIoS user, it will instead be provided by a universal SocIoS user (named admin) behind the scenes.

Table 1. Core services framework

API Method	Input	Output	Facebook		Twitter		Flickr		Daily Motion		MySpace		YouTube	
	Parameter and Value	Parameter and Value	Implem.	Authent. required	Implem.	Authent. required	Implem.	Authent. required	Implem.	Authent. required	Implem.	Authent. required	Implem.	Authent. required
findPersons	PersonFilter MediaItemList	List <Person>	✓	✓	✓	✓	✓	✗	✓	✗	✓	✗	✗	✗
getActivities	ActivityIdList	List<Activity>	✓	✓	✓	✓	✗	✗	✗	✗	✓	✗	✓	✗
getPersons	PersonIdList	List<Person>	✓	✓	✓	✓	✓	✗	✓	✗	✓	✗	✓	✗
findMediaItems	MediaItemFilter PersonIdList	List<MediaItem>	✓	✓	✓	✓	✓	✗	✓	✗	✓	✗	✓	✗
findConnected Activities	ActivityIdList	List<Activity>	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✓	✗
connectedPersons	ObjectId	List <Person>	✓	✓	✓	✓	✓	✗	✗	✗	✓	✓	✓	✗
myConnectedPersons	ObjectId	List <Person>	✓	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓	✗
findActivities	ActivityFilter PersonFilter MediaItemIdList PersonIdList	List<Activity>	✓	✓	✓	✓	✓	✓	✗	✗	✓	✗	✓	✗
getMediaItems	MediaItemIdList	List<MediaItem>	✓	✓	✓	✓	✓	✗	✓	✗	✓	✗	✓	✗

2.3 Technical requirements of the tool

Table 2. Technical requirements of the core services framework

Technical requirements
Operating System (OS)
Any
Programming language used and build tool (maven, ant, etc.)
Oracle JDK 1.6, Apache Ant
Programming libraries used
Metro Web Services stack, Apache CXF, guava. Twitter4j, gdata
Installation requirements
Glassfish Application server v3.1

2.4 Installation and Usage Guidelines

2.4.1 Installing the core services

The core services middleware is packaged into a single .war file that can be deployed to any Java application server, supporting the Metro web services stack. The WSDL can then be found at:

http://epart.atc.gr:8080/sociosApi/SociosApiImpl?wsdl

For our purposes, we used the Glassfish application server v3.1.

3 Recommendation Service

3.1 Overview

The recommendation service provides its clients with a list of recommended persons with strong reputation as content providers. These persons can be recommended in relation to certain topics or as potential group members. For example, a journalist can use the service to obtain a list of persons that are reliable as sources for the topic "Syria clashes", likewise, he can use the service to obtain a list of persons that are recommended for his "occupy" group. The recommendations are based on aggregated information collected over a period of time. This information is extracted by the crawling service (described in the bottom of the section). Note that we expect to get more accurate results as the number of crawls increases.

SaND is queried to extract a list of person related to topics (or group name and members). SaND is the IBM's Social Network and Discovery Engine. SaND is an aggregation platform for social media information. Its integrated index supports combining content-based analysis and person-based analysis over a rich data foundation. Based on aggregated content and relationships, it can recommend persons related to certain topics. The **Crawling Service** (described in the bottom of this section) is responsible to populate SaND with the aggregated content and relationships over the SNS. Once the person list is returned from SaND the results are sorted according to person reputation scores as content providers. Each person's reputation score is extracted using the reputation service. The reputation score is calculated based on a weighted set of aggregated indicators (see section 4 for more details).

When the service is used to obtain recommended persons for group, the service also interacts with the Social Filtering service to collect the group members. This information is added to the query to return persons that may be also related to these members and to the topic associated with the group name. This list is also sorted according to reputation score.

As SaND does not store the Person object as is, it interacts with the SocIos Core Service using the getPersons() method to return the person list to the clients.

The relationships between this service and the other SocIos service is described in the following figure:

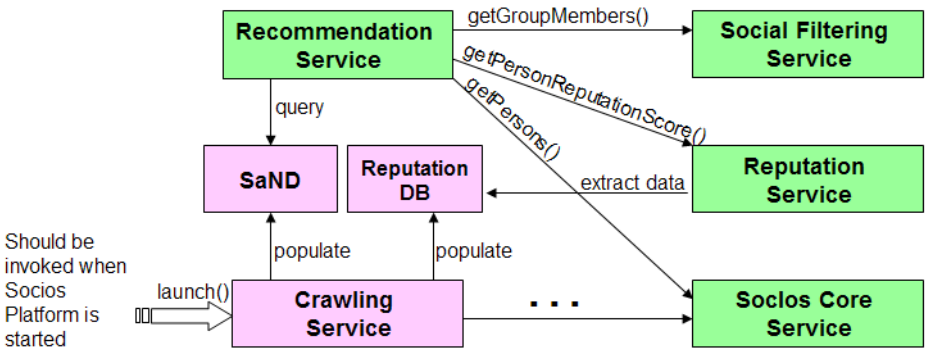

Figure 1. Dependencies on other SocIoS services

Both the recommendation service and reputation service depend on SaND being populated with aggregated information. This is done by the **Crawling Service**. The crawling service has two methods: launch () and stop ().

launch() should be invoked once on the start-up of the SocIoS platform. This method includes an internal process that is invoked periodically (e.g. once a day) and when awakes crawls the SNS using the SocIoS Core Services. It crawls only publically available information on consented SocIoS users.

For each person the crawler inquires general information (from returned SocIoS model entities) on him, his media items and his activities. It also inquires related information such as connected persons and connected activities. The crawler then extracts relevant information and populates SaND. It also populates the reputation db indicators related to the person and his related persons (e.g. connected persons, tagged persons). More information on the latter is described in section 4.

stop() should be invoked if, for some reason, there is a need to stop the crawler. In such a case, the crawling will not continue until launch () is called again.

The wsdl of the Crawling service can be found at:

http://epart.atc.gr:8080/CrawlersLauncherService/CrawlersLauncher?wsdl

3.2 Specification on components' integration - interfaces

Recommendation service exposes two Web Methods, described below:

a) getRecommendedPersons

This service takes a list of terms/topics and returns the top n most reputable SNS persons that posted some content on the subject topics. The returned list is sorted by the reputation scores of the persons in the list; the 1st person in the list is the one with the highest score. The 2nd argument defines the maximum length of the returned list.

Table 3. Recommendation service: getRecommendedPersons

Input	
Requirement	**Brief Description**
The crawling service should be invoked before using this API.	The SocIoS platform should start by calling the crawling service launch() API at least a few hours (better some days) before invoking this service. This is because SaND needs to be feed with data in-order to return results.
Parameter and Value (e.g. String, Integer)	**Brief Description**

topics: KeywordFilter top: Integer	List of topics (relationship between keywords is AND) Number of SN persons to return
Output	
Requirement	**Brief Description**
Parameter and Value (String, Integer, Array)	**Brief Description**
PersonList	List of the most reliable SN persons, with respect to the topics provided. The list is sorted by reputation score.

b) *getRecommendedPersonsPerGroup*

The service takes as input a Group object (and the group's owner ObjectId) and returns a list of additional SNS persons who may be good candidates as Group members (but are not currently members). The returned list includes the top n most reliable users that post content on the topic specified by the Group name (group.title). The returned list is sorted by the reputation scores of the persons in the list; the 1st person in the list is the one with the highest score. The 2nd argument defines the maximum length of the returned list.

Table 4. Recommendation service: getRecommendedPersonsPerGroup

Input	
Requirement	**Brief Description**
The crawling service should run before using this API. sociosUser parameter is required. sociosUser parameter value must be of the passed group owner.	The SocIoS platform should start by calling the crawling service launch() API at least a few hours (better a few days)before invoking this service. This is because SaND needs to be feed with data. Required since the service interacts with the Social Filtering Service to extract passed group members. This operation must be performed only on behalf of the group owner.
Parameter and Value (e.g. String, Integer)	**Brief Description**
sociosUser: ObjectId	ObjectId of the person who owns the Group

group: Group top: Integer	Group of SN users Number of SN persons to return
Output	
Requirement	**Brief Description**
Parameter and Value (String, Integer, Array)	**Brief Description**
PersonList	List of additional SN persons who are recommended for the group. The list contains the n most reliable users that post content related to the group name but who are not in the group yet. These persons may be related in some manner to the group members. The list is sorted by reputation score.

3.3 Technical requirements of the tool

Table 5. Technical requirements of recommendation service

Technical requirements
Operating System (OS)
Any
Programming language used and build tool (maven, ant etc)
Java EE 5
Programming libraries used
Apache Lucene, Apache Lucene Term Highlighter, Apache Commons Logging, Apache Commons Discovery, Apache Commons-HttpClient, Log4j, Apache CXF, Apache AXIS, Web Services Description Language for Java Toolkit (WSDL4J), SAAJ, JAX-RPC, XML Schema, and IBM Lucene Extension Library
Installation requirements
Glassfish Server 3.1, Netbeans J2EE development env. 7.0.1, MySQL database

3.4 Installation and Usage Guidelines

The recommendation service is packed with both the reputation service and crawling service. All reside in an EAR (Enterprise Archive) containing an Enterprise Java Bean (EJB), dependent libraries and properties configuration files.

The deployable component contains the following main components:

1. IBM's SaND configured especially for SocIoS
2. Reputation Services
3. Recommendation Services
4. Crawling Services
5. Clients to Social Filtering Service and SocIoS Core Services

3.4.1 Installation Instructions

Installing these services require adjusting the deployment environment for the services needs and modifying the configuration files placed in the EAR to teach the services on their configuration environment.

3.4.2 Installing the service

Stage 1: creating MySQL databases for the reputation indicators and for SaND

A. If not installed beforehand on the development environment, install MySQL Server from the following location (http://netbeans.org/kb/docs/ide/install-and-configure-mysql-server.html)
B. Create two databases (username: "root" and password: "empty") named:
    1. "reputation01"
    2. "personmap"
C. In the "personmap" database create two tables using the following SQL commands:

CREATE TABLE main (sandkeyint INTEGER NOT NULL , sandkey VARCHAR(16) NOT NULL PRIMARY KEY, name VARCHAR(256) NOT NULL, email VARCHAR(256) NOT NULL, exist SMALLINT NOT NULL, organization VARCHAR(128),country VARCHAR(32), title VARCHAR(256));

CREATE TABLE externalkeys (sandkey VARCHAR(16) REFERENCES main(sandkey), externalkey VARCHAR(256) NOT NULL PRIMARY KEY);

Stage 2: modifying the properties file placed in the config directory

1. sand.properties: SAND configuration files. Parts that need to be modified are placed on the top of the file required adjustment instructions are embedded in the property file comments
2. log4j.properties: logging properties configuration. Modify only the "log4j.appender.ALL.File" to point to a valid location for the log file

Stage 3: modify the EAR's glassfish-resources.xml file (located in the EAR "setup" directory) with proper information on how to connect to the databases created at stage 2.

Stage 4: adjusting the glassfish server JVM settings to include paths to property files

1. Open Glassfish server admin window
2. Under "Configuration" open "JVM Setting" and go to "JVM Options" tab
3. Add the following (adjusted to the actual location of the properties file) and save

```
-Dsandconf=C:/dev/socios/app/SociosEAR/config/sand.properties
-Dlog4j.configuration=C:/dev/socios/app/SociosEAR/config/log4j.properties
-Daaaconf=C:/dev/socios/app/SociosEAR/config/userset.properties
```

3.4.3 Usage Guidelines

The recommendation service can be invoked only after the crawler service was lunched and had the opportunity to crawl the SNS.

The recommendation services expect clients to interact with it using SOAP. Thus clients need to be created based on the wsdl it exposes in the deployed environment.

The wsdl can be found at:

http://epart.atc.gr:8080/RecommendationService/Recommendation?wsdl

4 Reputation Service

4.1 Overview

The reputation service provides its clients with a way to get reputation scores of SocIoS consent giving persons, based on a set of indicator values collected on each person over a period of time, and in relation to the other persons inspected. In particular, the service can be used by its clients to obtain relative reputation scores of persons that would reflect their reliability as content providers. The score for each person is a double between 0 and 10.

Person's score is calculated by applying a predefined set of weighs to each person's computed indicators and then normalizing it in relation to others. The indicators are populated for each person by the crawling service as part of its crawling (described in detail in the Recommendation Service in section 3) and placed in the "reputation" DB. Once a reputation service API is called, the inquired person aggregated indicators data is extracted from the database. Each indicator weight is defined based on if the indicator's nature if is explicit or implicit (i.e. if this is related to an action done by the person or action done by others that relate to this user) and if it's content or social related (e.g. posting content vs. connected persons). Note that we expect to get more accurate results as the number of crawls increases and more information is gathered on the person's activities.

A few sample indicators: # of posted media items, # of connected persons, # of connected activities, Average # of connected activities on each activity, and # persons that tagged her posts.

In the future, we hope to add another API that is to allow explicit repudiation contribution from other SocIoS services such as flexi-price or even from the front-end.

4.2 Specification on components' integration - interfaces

a) getPersonReputationScore

The method returns the reputation score of a particular person as content provider.

Table 6. Reputation service: getPersonReputationScore

Input	
Requirement	**Brief Description**
The crawling service launch() method should run before using this API.	The SocIoS platform should start by calling the crawling service launch() API at least a few hours (better a few days) before invoking this service. This is because person indicators need to be populated before we can calculate his reputation.

Parameter and Value (e.g. String, Integer)	Brief Description
person: Person	Person to calculate reputation for
Output	
Requirement	**Brief Description**
Parameter and Value (String, Integer, Array)	**Brief Description**
Double	Persons reputation score between 0 and 10

4.3 Installation and Usage Guidelines

4.3.1 Installing the service

This service is installed along with the recommendation service and crawling service (see description in section 3.4)

4.3.2 Usage Guidelines

The reputation service can be invoked only after the crawler service was lunched and had the opportunity to crawl the SNS and populate the person indicators.

The reputation services expect clients to interact with it using SOAP. Thus clients need to be created based on the wsdl it exposes in the deployed environment.

The wsdl can be found at:

http://epart.atc.gr:8080/ReputationService/Reputation?wsdl

5 Event Detection

5.1 Overview

The functionality of the Event Detection service consists of the following steps:

- It receives a set of user ids that correspond to a group of SocIoS users.
- It retrieves the recent activities of the input users in all supported SNs, calling the relevant methods of the Core Services.
- It filters out activity items that contain no textual content.
- It identifies terms that are highly likely to be related to an event. To identify such terms, we employ the measure of entropy, which expresses the average number of distinct tokens that are likely to follow a given term. Event-related terms, such as "occupy", "syria" and "iran", are expected to be associated with the same words across different activity items, thus exhibiting a low entropy. We can, thus, filter out irrelevant terms through a threshold that specifies the maximum entropy per term. In this way, we also discard words of low document frequency, which are rather scarce in the input activity items.
- It removes stop-words from the list of event-related terms. This process relies on the premise that stop-words co-occur with most of the other event-related terms. To filter them out, we estimate for each term the portion of event-related terms that share with it a non-zero Jaccard similarity (this similarity is derived from the set of activity items associated with each term).
- It groups similar terms into clusters that pertain to a distinct real-world event. For each term, it aggregates the activity items that contain it into an n-gram graph. All pair-wise comparisons between the graphs of the selected terms are executed and those exceeding a similarity threshold create a link between the corresponding term. The resulting term-graph is then split into its connected components, with each component forming a distinct cluster.
- It classifies the activity items that contain none of the event-related terms into the created clusters. To enhance the efficiency of this procedure, the Naive Bayes Classifier is employed, after training it over the data set formed by the already clustered activity items. The classification features are again extracted from the similarity between the graph representation of the individual activity items and the clusters.
- Finally, it returns the aggregated activity items that belong to each cluster.

The code of the Event Detection service is available at:

http://gforge.grid.ece.ntua.gr/svn/socios/trunk/WP3 Services/EventDetection

The service is running on the server of ATC and its WSDL is available at:

http://epart.atc.gr:8080/EventDetection/EventDetection?wsdl

5.2 Specification on components' integration – interfaces

Table 7. Interface of Event Detection service

Input	
Requirement	**Brief Description**
Parameter and Value (e.g. String, Integer)	**Brief Description**
List<Integer> userIds	A set of SocIoS' user ids.
Output	
Requirement	**Brief Description**
Parameter and Value (String, Integer, Array)	**Brief Description**
List<ActivityList>	A set of clusters containing the activity items of the input users, with each cluster pertaining to a specific real-world event.

5.3 Technical requirements of the tool

The only technical requirement for proper functionality is that the Event Detection service is able to connect to SocIoS Core Services.

Table 8. Technical requirements of Event Detection service

Technical requirements
Operating System (OS)
Any
Programming language used and build tool (maven, ant etc)
Java, Ant
Programming libraries used

Jinsect (a tool for creating n-gram graphs that is freely available at: http://sourceforge.net/projects/jinsect/), Weka (a library of machine learning algorithms)
Installation requirements

5.4 Installation and Usage Guidelines

5.4.1 Installing the service

The Event Detection service is packaged into a single .war file and can be easily installed by deploying it into a web server, such as Apache Tomcat or Glassfisch.

5.4.2 Usage Guidelines

The only requirement for proper use of the Event Detection service is that the input comprises a set of valid SocIoS user ids.

6 Social Filtering Service

6.1 Overview

The Social Filtering Service (SFS) let users aggregate, filter and distribute information coming from social network systems. The SFS comprises two main services:

- GroupService: this service gathers existing social accounts into thematic groups.
- FilteringService: this service allows the filtering of information elements (information channels such as Twitter accounts as well as individual information items such as individual tweets) by group, by account, by date or by keyword.

The component source code is available at the following URL under the Apache License 2.0:

http://gforge.grid.ece.ntua.gr/svn/socios/trunk/WP3 Services/SocialFilteringAPI

6.2 Specification on components' integration - interfaces

a) *GroupService*

The GroupService provides methods allowing to manage groups of social network accounts (Person objects in the SocIoS Object Model): creation or deletion of groups, addition or removal of members. It is typically used for gathering a set of various social network accounts which relate to a common topic. This section presents the methods exposed by the SocIoS GroupService. The input and the output of each method is briefly described.

addMembers /removeMembers

Table 9. Social Filtering service: addMembers/removeMembers

Input	
Parameter and Value (e.g. String, Integer)	**Brief Description**
SessionId sessionId PersonList people ObjectId groupId	sessionId: identifier of the session associated with the current client people: list of people to add/remove in/from the specified group groupId: the identifier of the group to modify
Output	

Parameter and Value (String, Integer, Array)	Brief Description
Group	Modified group with added/removed people

createGroup

Table 10. Social Filtering service: createGroup

Input	
Parameter and Value (e.g. String, Integer)	**Brief Description**
SessionId sessionId PersonList people Group group	sessionId: the identifier of the session associated with the current client people: initial list of group members group: initial object containing the description of the group to be populated
Output	
Parameter and Value (String, Integer, Array)	**Brief Description**
Group	The created group

deleteGroup

Table 11. Social Filtering service: deleteGroup

Input	
Parameter and Value (e.g. String, Integer)	**Brief Description**
SessionId sessionId ObjectId groupId	sessionId: the identifier of the session associated with the current client groupId: identifier of the group to be deleted
Output	
Parameter and Value (String, Integer, Array)	**Brief Description**
boolean	True if the group with the specified identifier was successfully deleted.

getGroupMembers

Table 12. Social Filtering service: getGroupMembers

Input	
Parameter and Value (e.g. String, Integer)	**Brief Description**
SessionId sessionId ObjectId groupId	sessionId: the identifier of the session associated with the current client groupId: identifier of the target group
Output	
Parameter and Value (String, Integer, Array)	**Brief Description**
PersonList	A list of all members in the group with the specified identifier.

getGroups

Table 13. Social Filtering service: getGroups

Input	
Parameter and Value (e.g. String, Integer)	**Brief Description**
SessionId sessionId	sessionId: the identifier of the session associated with the current client
Output	
Parameter and Value (String, Integer, Array)	**Brief Description**
GroupList	A list of all groups already defined by the user corresponding to the current session.

b) SocialFilteringService

The SocialFilteringService makes it possible to easily access the activities of the persons who belong to a specific group. It hinges both on the SocIoS GroupService and on the SocIoS Core Services.

getRecentActivities

Table 14. Social Filtering service: getRecentActivities

Input	
Requirement	**Brief Description**
Socios Core Service GroupService	The SocialFilteringService uses the following SocIoS Core Service method: findActivities(ActivityFilter filter)
Parameter and Value (e.g. String, Integer)	**Brief Description**
SessionId sessionId ObjectId groupId Calendar startTime	
Output	
Parameter and Value (String, Integer, Array)	**Brief Description**
ActivityList	List of activities matching the input parameter values.

6.3 Technical requirements of the tool

Table 15. Technical requirements of Social Filtering service

Technical requirements
Operating System (OS)
Any OS supporting a Java 6 Virtual Machine
Programming language used and build tool (maven, ant etc)
Java 6, Maven2
Programming libraries used
Apache CXF 2.4.x http://cxf.apache.org/

geronimo-ws-metadata_2.0_spec 1.1.3 geronimo-jaxws_2.2_spec 1.0 jaxb-api 2.1 jaxb-impl 2.1.12
Installation requirements
A Java EE servlet container such as Jetty, Tomcat, WebSphere, Glassfish

6.4 Installation and Usage Guidelines

The source code of the service can be downloaded from the following URL:

http://gforge.grid.ece.ntua.gr/svn/socios/trunk/WP3 Services/SocialFilteringAPI

The command below – requiring Maven2 – generates a WAR file. The generated WAR can the be deployed in any J2EE container such as Tomcat or Glassfish.

<mvn clean install>

6.4.1 Usage Guidelines

The SocialFilteringService exposes two end points that can be accessed at the following URLs, where "localhost" needs to be replaced by the network name of the server on which the service has been deployed:

http://localhost:8080/socialfilteringservice-1.0/services/GroupServicePort

http://localhost:8087/socialfilteringservice-1.0/services/SocialFilteringServicePort

The WSDL files can be accessed by adding the item "?wsdl" at the end of the URL:

http://localhost:8087/socialfilteringservice-1.0/services/GroupServicePort?wsdl

http://localhost:8087/socialfilteringservice-1.0/services/SocialFilteringServicePort?wsdl

7 FlexiPrice

7.1 Overview

FlexiPrice is a price negotiation software service which supports business models that are based on a combination of dynamic pricing, incentive compatible bidding and voluntary payments. The FlexiPrice service reflects the outcome of the research on the dynamics of information exchange in social networks and derived business models which is performed by the University of Haifa. It focuses on the interplay between social and economic influences in information exchange environments. The service supports a two-party negotiation process, i.e .one buyer and one seller, and lets each side submit their true subjective value of an information item. In this way it allows social and psychological factors influence the economic negotiation and should lead to a good rate of transaction completion. FlexiPrice provides for a variety of negotiation schemes as well as dynamic pricing subject to the value of the content and the social context.

In the first pilot FlexiPrice allows the buyer of a media item, a journalist in this case, to assign a pricing-scheme to a media item of interest. In the pricing scheme the journalist defines the negotiation method and the pricing function to be applied during the negotiation. The implemented negotiation methods are the open-price methods and the hidden-price method. In the open-price methods the seller is presented with the value price of the media item to the buyer, and can suggest a requested price accordingly. In the hidden-price method the buyer's value price of the media item is hidden from the seller. The seller than has a pre-defined number of attempts to suggest a price which is equal or less to buyer's suggested price. If the number of attempts is exhausted with no resolution, the purchase request expires. The pricing function defines the value price of the media item to buyer. The price can be fixed by the buyer or dynamically calculated from the seller reputation which is acquired from the Reputation auxiliary service.

The SocIoS user is exposed to the FlexiPrice service in three scenarios:

- A journalist, who identifies and interesting media item via any of the SocIoS platform services, initiates a purchase request by assigning a pricing-scheme to the media item. The journalist then contacts the owner of the media item and invites him to complete the transaction by logging into the SocIoS platform.
- The owner of a media item with an outstanding purchase request, logs into the SocIoS platform and is alerted by the system. He then enters the FlexiPrice management application and bid his requested price.
- The FlexiPrice management application is available to SocIoS users from the SocIoS front-end. In this application a SocIoS user, who is a buyer, can manage his price assignments, and a SocIoS user with outstanding purchase requests, can bid a requested price. All user, at all times, can view his or her past transactions.

The source code and README file which contains installation instructions can be downloaded from:

http://gforge.grid.ece.ntua.gr/gf/project/socios/scmsvn/?action=browse&path=%2Ftrunk%2FWP3+Services%2FFlexiPrice%2F

7.2 Specification on components' integration - interfaces

a) *assign_price*

Table 16. FlexiPrice: assign_price

Input - String assign_price(String system, User user, MediaItem media_item)	
Requirement	**Brief Description**
RESTfull API: url = http://flexiprice.eu/assign_price/ method = POST	
Parameter and Value (e.g. String, Integer)	**Brief Description**
<flexi_price> **<system>**socios**</system>** **<user>** **<user_id>**xxx**</user_id>** **<lang>**en**</lang>** **<real_name>**John**</real_name>** **</user>** **<action_params>** **<item_id>**yyy**</item_id>** **<item_name>**Apple**</item_name>** **<owner_id>**zzz**</owner_id>** **</action_params>** **</flexi_price>**	system should always be "socios" lang should always be "en" user_id is the SocIoS user ID owner_id can be the SN ID. It will be mapped to the SocIoS ID in the get_transactions call.
Output	
Requirement	**Brief Description**

The call should be followed by a URL http://flexiprice.eu/api.php?token=token which is returned by the call and will open the assignment window to the user.	
Parameter and Value (String, Integer, Array)	**Brief Description**
<flexi_price> **<fp_token>**Token **</fp_token>** **</flexi_price>**	The fp_token is a token used to open the assignment window to the user

b) *get_transactions*

Table 17. FlexiPrice: get_transactions

Input - String get_transactions(String system, User user)	
Requirement	**Brief Description**
RESTfull API: url = http://flexiprice.eu/get_transactions/ method = POST	
Parameter and Value (e.g. String, Integer)	**Brief Description**
<flexi_price> **<system>**socios**</system>** **<user>** **<user_id>**xxx**</user_id>** **<lang>**en**</lang>** **<real_name>**Name**</real_name>** **<sn_ids>** **<sn_id>**yyy**</sn_id>** **<sn_id>**zzz**</sn_id>** **</sn_ids>** **</user>**	system should always be “socios” lang should always be “en” user_id is the SocIoS user ID In FlexiPrice purchase requests the owner may be known by his ID on the related SN. sn_id is required in order to be able to match the SocIoS user ID with his ID on the SN

</flexi_price>	
Output	
Requirement	**Brief Description**
The call should be followed by a URL http://flexiprice.eu/api.php?token=token with the token returned in order to open the transaction list window to the user.	
Parameter and Value (String, Integer, Array)	**Brief Description**
<flexi_price> **<new_bids>**1 or 0**</new_bids>** **<fp_token>**Token**</fp_token>** **</flexi_price>**	The new_bids flag indicates whether the user has any pending purchase requests. The fp_token is a token used to open the transaction list window to the user

7.3 Technical requirements of the tool

Table 18. Technical requirements of FlexiPrice service

Technical requirements
Operating System (OS)
Ubuntu Linux 10.04.2
Programming language used and build tool (maven, ant etc)
PHP
Programming libraries used
• Apache version 2.2 or higher • MySQL version 5.1 • PHP 5.3 or higher • PHP PEAR. The following php PEAR classes need to be installed:

1. MDB2   a. MDB2_Driver_mysql-1.5.0b3 2. Auth 3. XML_Parser 4. XML_Serializer Pager
Installation requirements

7.4 Installation and Usage Guidelines

1. Import the DB structure (flexi_price_db.sql). The settings in the "system" table for "socios" entry need to be adjusted manually.
2. Set the following fields:
    a. api_protocole (rest,SOAP,...)
    b. base_url (The URL for the api_protocole)
    c. api_param_type (xml,post,get,...)
3. Go throw the conf.php file and make sure settings are correct
4. Set 777 permissions to the following directories:
    a. Session
    b. tpl/templates_c/

8 Translation Service

8.1 Overview

The Translation Service allows the SocIoS platform to translate selected SocIoS Object Model objects and text into various languages. It uses Bing Translate to accomplish this. Bing Translate is a tool that automatically translates text from one language to another language (e.g. French to English). The Bing Translate API can be used to programmatically translate text in webpages or apps.

The Translation Service essentially provides a level of abstraction between SocIoS applications and Bing Translate.

The code of SocIoS translation service can be found here:

http://gforge.grid.ece.ntua.gr/svn/socios/trunk/SociosTranslate

8.2 Specification on components' integration - interfaces

The SocIoS Translation Service component communicates with the other components through a well-defined set of operations that implement the SocIoS Data model. The input and output messages were defined in chapter 5.6 of the D2.5.3 deliverable.

8.3 Technical requirements of the tool

Table 19. Technical requirements of Translation service

Technical requirements
Operating System (OS)
Any
Programming language used and build tool (maven, ant etc)
Oracle JDK 1.6, Apache Ant
Programming libraries used
Microsoft Translator Java API
Installation requirements
Glassfish Application server v3.1

8.4 Installation and Usage Guidelines

8.4.1 Installing the service

The translation service is packaged into a single .war file that can be deployed to any Java application server. The WSDL can then be found at:

http://epart.atc.gr:8080/SociosTranslateObject/TranslateService?wsdl

The live version is installed on Glassfish 3.1.

8.4.2 Usage Guidelines

The translation service expects clients to interact with it using SOAP. A client would need to be created based on the wsdl that is exposed when the translation service is deployed.

9 Front End

9.1 Overview

SocIoS Front-End is the presentation layer that lies between the platform and the SocIoS end users. Acting as SocIoS platform user interface, it provides the capability of interacting with SocIoS components as well as an authentication mechanism.

The latest version of the Front End can be found at the following URL:

http://frontend.sociosproject.eu

The source code of the Front-End can be found here:

http://gforge.grid.ece.ntua.gr/svn/socios/trunk/SociosFrontEnd

9.2 Specification on components' integration - interfaces

The Front End is a tool used to access all SocIoS functionality and is therefore has to integrate correctly with all SocIoS components. The front end has a client component which is used to call all SocIoS Services that are used by the front end. It is therefore important to make sure all web service URL defined in the Front End web.config file match the actual locations of the hosted web service WSDLs.

9.3 Technical requirements of the tool

The front end is a DotNetNuke installation, enriched my modules developed to suit the needs of the SocIoS platform. Therefore the requirements are very similar to those of any DNN installation. These requirements are described in the table below.

Table 20. Technical requirements of Front-End

Technical requirements
Operating System (OS)
Windows XP+
Programming language used and build tool (maven, ant etc)
C#
Programming libraries used
DotNetOpenAuth, Newtonsoft.Json.Net35

Installation requirements
IIS 6+, SQL Server 2005+

9.4 Installation and Usage Guidelines

9.4.1 Installing the component

In order to install the Front End, it is necessary to first create a database for it, run the table creation scripts, then transfer all the Front End files into a Virtual Directory defined in IIS. It is also necessary to make some changes to the file system, where the Front End is installed: Give read/write permissions to the physical directory for either the ASPNET user account if you have IIS 5 (Windows XP) or the NETWORK SERVICE user account if you have IIS 6 (Windows 2003). The permission that you must give is "modify". Then give read (but not write) permissions to the IUSR_machinename account. This should allow the CMS that the Front-End is based on to run successfully.

9.4.2 Usage Guidelines

After installing the front end, depending on what the name of the virtual directory is, it can be accessed via a web browser from:

http://localhost/<virtual directory name>.