COSMOS

Cultivate resilient smart Objects for Sustainable city applicatiOnS

Grant Agreement Nº 609043

D7.7.1 Integration of Results

WP7 Use cases Adaptation, Integration and Experimentation

Version: 1.0
Due Date: 30/09/2014
Delivery Date: 15/10/2014
Nature: Report
Dissemination Level: PU

Lead partner: ICCS/NTUA
Authors: George Kousiouris, Achilleas Marinakis, Panagiotis Bourelos, Orfefs Voutyras, Vassilis Psaltopoulos (ICCS/NTUA), Jozef Krempasky (ATOS), Paula Ta-Shma (IBM), Francois Carrez, Adnan Akbar (UniS), Bogdan Târnaucă, Leonard Pitu (SIEMENS)

Internal reviewers: Sergio Fernández Balaguer (EMT), Joshua Cooper (HILD)

www.iot-cosmos.eu

The research leading to these results has received funding from the European Community's Seventh Framework Programme under grant agreement n° 609043

Version Control:

Version	Date	Author	Author's Organization	Changes
0.1	28/07/2014	Achilleas Marinakis	ICCS/NTUA	Circulating the ToC
0.2	13/08/2014	Paula Ta-Shma	IBM	Adding IBM contribution
0.3	29/09/2014	Achilleas Marinakis, Panagiotis Bourelos	ICCS/NTUA	Updating Test Cases and Demos Scenarios
0.4	01/10/2014	Achilleas Marinakis and co-authors	ICCS/NTUA, ATOS, UniS	Updating Unit Tests, Model Validation and Future Work
0.5	02/10/2014	Achilleas Marinakis and co-authors	ICCS/NTUA, UniS	Updating Testbed Instantiation and Future Work
0.6	09/10/2014	Achilleas Marinakis and co-authors	ICCS/NTUA, UniS, SIEMENS	Version ready for internal review
1.0	15/10/2014	Achilleas Marinakis	ICCS/NTUA	Version for submission

Table of Contents

List of Figures

List of Tables

Table of Acronyms

Acronym	Meaning
API	Application Programming Interface
CB	Case Base
CBR	Case Based Reasoning
CEP	Complex Event Processing
D	Deliverable
GA	General Assembly
GUI	Graphical User Interface
GVE	Group Virtual Entity
ID	Identifier
IoT	Internet of Things
JSON	Java-Script Object Notation
KNN	K-Nearest Neighbours
ML	Machine Learning
REST	Representational State Transfer
SVM	Support Vector Machine
UC	Use Case
UI	User Interface
VE	Virtual Entity
VM	Virtual Machine
WP	Work Package
XP	Experience

Executive Summary

This report presents the results of the Integration process carried out in year 1 of the COSMOS project.

We started the process by installing the functional components of COSMOS in the testbed, whose current status is described in chapter 4.

According to the development planning (please see section 3.1.6 of D7.6.1 [1]), we held an Integration Workshop alongside with the GA meeting in September 2014. During this Workshop, we defined two high level system sequences of COSMOS, which aim to validate, on one hand, the data flow from the data source to the Cloud storage (please see section 2.3.1.1) and, on the other hand, to validate the process by which VE stored knowledge is evaluated through CBR and acquired as well as exchanged through Experience Sharing (please see section 2.3.1.2). As it is mentioned in subchapter 3.1.5.1 of D7.6.1, we developed Unit/Components Tests for the modules that are involved in the system sequences. The results of these Tests are provided in the template that was introduced in section 3.1.5.2 of D7.6.1, including the requirements that are addressed, either partially or fully, from them.

We then clarified six potential demo scenarios, all deriving from the aforementioned system sequences and Integration Tests for the components that participate in these scenarios. Through the demos, we aim to highlight the COSMOS functionalities such as data object storage, metadata search, analytics close to the storage, prediction, privacy, decentralised and autonomous management of VEs and decentralised knowledge acquisition.

We have developed a model for predicting events by inferring high-level knowledge from raw IoT data by pattern recognition. Different Machine Learning algorithms were developed for pattern recognition purposes. In chapter 5, we validate the model by comparing the prediction results to the recorded events at particular times. We used an Occupancy Detection scenario for the validation purpose in which we are inferring occupancy states of users by observing electricity patterns of users (more details in section 3.1.2.3 and chapter 5). We used several variants of Support Vector Machine (SVM) [2] and K Nearest Neighbors (KNN) [3] in our work. The validation of results showed that the efficiency of KNN is highest (more then 90%) as compared to SVM variants.

The purpose of chapter 6 is to describe how the requirements that are partially or not addressed from the Unit Tests are going to be covered during the future years of the project. The COSMOS requirements are listed in the relevant annex of D2.2.1 [4] At the moment, we are in the process of revising the Y1 requirements and extending them to also include new functionalities. After the finalisation of this process, we will remap the Test Cases in order to identify the concrete gaps based on the revised list.

1. Introduction

1.1. Purpose of the Document

D7.7.1 has the purpose of consolidating the results from the work performed, aiming to provide the integrated view of COSMOS outcomes. In order to achieve this goal, a process has been already defined in the context of D7.6.1 [1], that is concretized in specific actions in this document.

This includes the definition of component-based unit testing, along with determination of the integrated scenarios that are designed to drive inter-component cooperation and demonstrate the added value in the project development. This process goes hand in hand with the work performed in WP2, regarding the architectural structure of the COSMOS framework as well as the requirements definition and aims also to validate the approach and discover potential issues and gaps in the aforementioned process.

1.2. Document Structure

The document is structured as follows:

- In Chapter 2 unit tests are described on a component level, along with the combined sequence diagrams that are necessary on a system level to produce an added value functionality.
- In Chapter 3 specific functionalities of the COSMOS platform are highlighted through the envisioned demo scenarios
- In Chapter 4 the testbed instantiation is depicted.
- Chapter 5 is included in order to validate the accuracy of the produced models
- In Chapter 6 gaps and future work directions are identified at the requirements level.

2. Test Cases

2.1. Unit/Component Tests

2.1.1. Data Mapping

Test Case Number Version	**4_DM_1**
Test Case Title	Storing live data, coming from the Message Bus, in the Cloud Storage.
Module tested	Data Mapping
Requirements addressed	4.1, 4.2
Initial conditions	RabbitMQ service is installed and running Hildebrand data stream is available
Expected results	Data are stored as objects with timestamps, metadata and a lower threshold for the size.
Owner	Achilleas Marinakis
Steps	Execute: /NTUA/DataMapping/target java -jar DataMapping-Y1.jar Hildebrand
Passed	Yes
Bug ID	None
Problems	None
Required changes	None

Table 1: Results for Data Mapping Unit Test #1

Test Case Number Version	**4_DM_2**
Test Case Title	Storing historical data, coming from a static file, in the Cloud Storage
Module tested	Data Mapping
Requirements addressed	4.1, 4.2
Initial conditions	The path to the static file is provided.
Expected results	Data are stored as objects with timestamps and id metadata and a lower threshold for the size.
Owner	Achilleas Marinakis

Steps	Execute: /NTUA/DataMapping/target java -jar DataMapping-Y1.jar Surrey
Passed	Yes
Bug ID	None
Problems	None
Required changes	None

Table 2: Results for Data Mapping Unit Test #2

2.1.2. Message Bus

Test Case Number Version	**4_MB_1**
Test Case Title	4_MB_1
Module tested	Message Bus
Requirements addressed	4.9
Initial conditions	RabbitMQ service is installed and running. Hildebrand data stream is available.
Expected results	Messages are propagated to subscribers.
Owner	Atos
Steps	1.) Execute message publisher ~/Atos/Mosquitto/Bridge.exe 2.) Execute /Atos/Mosquitto/VeSub.exe 3.) Messages are correctly routed to the subscriber.
Passed	Yes
Bug ID	
Problems	
Required changes	

Table 3: Results for Message Bus Unit Test #1

2.1.3. Cloud Storage – Metadata Search

We have developed unit tests for the metadata search component. These tests verify requirements 4.1, 4.3 and 5.0.

2.1.4. Cloud Storage – Storlets

We have developed unit tests for the storlets component. These tests verify requirements 4.1, 4.4, 4.5, 4.6, UNI.041.

2.1.5. Event Detection & Situational Awareness @Platform

Test Case Number Version	**4_ED_1**
Test Case Title	4_ED_1
Module tested	Complex Event Processing
Requirements addressed	4.7, 4.10, 4.11, 4.12, 5.14
Initial conditions	Data streams provided by Message Bus are available. Provided CEP configuration file contains definition of the "Sensor Malfunction" situation.
Expected results	Detected situation is published on the specific Message Bus topic.
Owner	Atos
Steps	1.) Connect CEP to the event stream by invoking: ~/Atos/Cep_Cos20Sa/RmqCollector/RmqCollector.exe –hild 2.) Start event detection service: ~/Atos/ Cep_Cos20Sa/SCep 3.) Forward detected events back to the message bus by invoking ~/Atos/Cep_Cos20Sa/RmqPublisher/ RmqPublisher.exe –[topicName] 4.) Observe that detected situation is published on the message bus under specified "topicName".
Passed	Yes
Bug ID	
Problems	
Required changes	

Table 4: Results for Event Detection @Platform Unit Test #1

2.1.6. Prediction

Test Case Number Version	**6_ML_1**
Test Case Title	6_ML_1
Module tested	Prediction
Requirements addressed	6.2, 6.3, 6.6, 6.22, 6.23
Initial conditions	Python installed with numpy, scikit, pandas and statsmodel. Time series historical data file.
Expected results	Machine Learning model which can predict the user Occupancy state.

Owner	University of Surrey
Steps	Install the required libraries. Store the historical data file. Run the code.
Passed	Yes
Bug ID	N/A
Problems	No
Required changes	No

Table 5: Results for Prediction Unit Test #1

2.1.7. Semantic Description and Retrieval

Test Case Number Version	**5_SDR_1**
Test Case Title	VE semantic description and retrieval
Module tested	Triple store access
Requirements addressed	5.0, 5.2, UNI.414, UNI.416, UNI.425, UNI.426
Initial conditions	First version of the semantic model available. Test triple store installed. Test query API available.
Expected results	Successful recording of the VE triples and matches between the retrieval results and the expected VE descriptions.
Owner	SIEMENS
Steps	1)Use a template form to fill in the description of a VE; 2)Call the API to store the description of the VE into the triple store; 3) Repeat steps 1 and 2 for a number of VEs with different attributes; 4)Query the triple store in order to retrieve the description of different VEs based on various search criteria; 5)Compare the results with the expected outcome.
Passed	Yes
Bug ID	
Problems	
Required changes	

Table 6: Results for Semantic Description and Retrieval Unit Test #1

Test Case Number Version	**5_SDR_2**
Test Case Title	VE description validation
Module tested	Triple store access
Requirements addressed	5.0, 5.2, UNI.414, UNI.416, UNI.425, UNI.426
Initial conditions	First version of the semantic model available. Test triple store installed. Test query API available.
Expected results	Inconsistent VE descriptions should not be stored into the triple store.
Owner	SIEMENS
Steps	1)Use a template form to fill in the description of a VE with erroneous or inconsistent fields; 2)Call the API to store the description of the VE into the triple store; 3) Repeat steps 1 and 2 for a number of VEs with different attributes; 4)Whenever the VE description does not meet the consistency requirements, and error condition must be signalled and no commit to the triple store should be made; 5)Query the triple store in order to retrieve the description of different inconsistent VEs whose description and storage attempt was made; 6)No complete or partial VE description should be retrieved.
Passed	Yes
Bug ID	
Problems	
Required changes	

Table 7: Results for Semantic Description and Retrieval Unit Test #2

2.1.8. Privelets

Test Case Number Version	**3_PR_1**
Test Case Title	Privacy
Module tested	Privelets

Requirements addressed	3.1, 3.8, 3.15
Initial conditions	Existing jsonFile.json in root directory that follows precise pattern. RESTful Client (or any mechanism) to exchange JSON files.
Expected results	Data exchange filtering procedure
Owner	NTUA
Steps	1) Generate a Json file on the root directory of our device "C:\\" and call it "jsonFile.json". Pre-complete or leave it empty. 2) Run the component as a java application and switch to the REST client and/or the UI page. 3) Use Google Chrome and its add-on called "Advanced REST Client", (or any other way) to send/receive JSON files. 4) Perform @POST service insertData with groups of tags on the call's payload. (data is now filtered according to the configuration) 5) Perform @GET service getPublicJasonFileTags to get the filtered JSON file.
Passed	Yes
Bug ID	None
Problems	None
Required changes	None

Table 8: Results for Privelets Unit Test #1

Test Case Number Version	**3_PR_2**
Test Case Title	Configuration
Module tested	Privelets
Requirements addressed	3.1, 3.8, 3.15
Initial conditions	Existing jsonFile.json in root directory that follows precise pattern. RESTful Client (or any mechanism) to exchange JSON files.
Expected results	View and edit configuration file.
Owner	NTUA
Steps	1) Check about the existing configuration file by calling the @GET service called retrieveExistingConfigurationFile. 2) Update the existing configuration file, and add or remove tags considered as private by calling the @POST service, updateExistingConfigurationFile and giving in the payload a Json file.3) Check the link http://localhost:8080/JAXRS-COSMOS/faces/home.xhtml, the UI of that has to do with the instance of the data to be exchanged or repeat step 1.

Passed	Yes
Bug ID	None
Problems	None
Required changes	None

Table 9: Results for Privelets Unit Test #2

2.1.9. Planner

Test Case Number Version	**5_PL_1**
Test Case Title	Notification Service and CBR
Module tested	Planner
Requirements addressed	UNI. 704, 706, 708, 715, 719 5.31 5.29, UNI.010 5.21 5.10 5.9, UNI.251
Initial conditions	Have a CB inside the test bed. Have the VE and app simulation jar inside the test bed.
Expected results	The answer to the notification of the User Generated Event in the GUI. Command Line or Terminal Log with similarities retrieved and queries used.
Owner	Panagiotis Bourelos, Orfefs Voutyras
Steps	Place the CB in the localstore folder of the home directory. Open command line or terminal. Navigate to the jar folder. Execute “java –jar DemoProjectMaven-1.0-SNAPSHOT.jar”. Keep the terminal or cmd open to view log info and locate the GUI. Enter a set of temperature and time that exist locally (time: 3535 and temp: 26). Press “Send Notification”.
Passed	Yes
Bug ID	None
Problems	None

Required changes	None

Table 10: Results for Planner Unit Test #1

2.1.10. Event Detection & Situational Awareness @VE

Not applicable for year 1.

2.1.11. Experience Sharing

Test Case Number Version	**6_ES_1**
Test Case Title	Experience Sharing and returned Solution
Module tested	Planner, Experience Sharing
Requirements addressed	Same as those of 2.1.9 plus: 5.11, 5.12, 6.8, 6.18, 6.19, 6.34, 6.35
Initial conditions	Have a CB and Friend List inside the test bed. Have the VE and app simulation jar inside the test bed. Have one or both auxiliary XP sharing VEs active in the VMs, along with their CBs and Friend Lists.
Expected results	The answer to the notification of the User Generated Event in the GUI Command Line or Terminal Log with similarities retrieved and queries used The Logs of auxiliary VEs in the VMs with the Planner result and the Experience Sharing steps
Owner	Panagiotis Bourelos, Orfefs Voutyras
Steps	Place the CB and Friend List in the localstore folder of the home directory; Open command line or terminal; Navigate to the jar folder; Execute “java –jar DemoProjectMaven-1.0-SNAPSHOT.jar”; Keep the terminal or cmd open to view log info and locate the GUI; Boot the VEs in their respective VMs; Enter a set of temperature and time that exist in another VE (time: 1800 and temp: 26 for VE Flat 2/ time: 3535 and temp: 28 for VE Flat 3); Press “Send Notification”
Passed	Yes
Bug ID	None

Problems	None
Required changes	None

Table 11: Results for Experience Sharing Unit Test #1

2.1.12. Hardware Security Board

Test Case Number Version	**3_HSB_1**
Test Case Title	Communication over standard interfaces
Module tested	Communication module
Requirements addressed	3.1
Initial conditions	Configure standard communication interfaces (I2C, SPI, USB, Ethernet)
Expected results	Data flow can take place over the specified interfaces. Sensors and the Ethernet network can be accessed.
Owner	Leonard Pitu
Steps	1. Load the hardware design (*.bit file) 2. Load the firmware 3. Boot the firmware 4. Obtain an IP address 5. Read the sensors 6. Make the data available over ETH.
Passed	Yes
Bug ID	None
Problems	None
Required changes	None

Table 12: Results for Hardware Security Board Unit Test #1

Test Case Number Version	**3_HSB_2**
Test Case Title	Secure the data flow (simulation)
Module tested	Cryptographic accelerator
Requirements addressed	3.3
Initial conditions	Configure the FPGA with the hardware design
Expected results	The cryptographic accelerator shall meet the FIPS-180 standard.
Owner	Leonard Pitu

Steps	Standard test vectors 1. Load the hardware module in the simulator (ModelSIM) 2. Load the test cases according to FIPS-180 3. Run the tests Random test vectors 1. Load the hardware module in the simulator (ModelSIM) 2. Load private test vectors generated with Crypto++ 3. Run the tests
Passed	Yes
Bug ID	None
Problems	None
Required changes	None

Table 13: Results for Hardware Security Board Unit Test #2

Test Case Number Version	**3_HSB_3**
Test Case Title	Secure the data flow (board)
Module tested	Cryptographic accelerator
Requirements addressed	3.3
Initial conditions	Configure the FPGA with the hardware design.
Expected results	The cryptographic accelerator shall meet the FIPS-180 standard.
Owner	Leonard Pitu
Steps	Standard test vectors 1. Load the hardware design (*.bit file) 2. Load the test cases according to FIPS-180 3. Run the tests Random test vectors 1. Load the hardware design (*.bit file) 2. Load private test vectors generated with Crypto++ 3. Run the tests
Passed	Yes
Bug ID	None
Problems	None
Required changes	None

Table 14: Results for Hardware Security Board Unit Test #3

Test Case Number Version	**3_HSB_4**
Test Case Title	HSB enrolment
Module tested	Cryptographic accelerator
Requirements addressed	3.7
Initial conditions	Configure the FPGA with the hardware design Load the software (Linux + drivers + test application)
Expected results	Use the ECDH to securely exchange the key
Owner	Leonard Pitu
Steps	1. Load the hardware design (*.bit file) 2. Load the software 3. Boot the system 4. Run the test cases and fetch the key from Keystone
Passed	Yes
Bug ID	None
Problems	None
Required changes	None

Table 15: Results for Hardware Security Board Unit Test #4

2.2. Integration Tests

2.2.1. Data Management

1. Camden UC data integration with Message Bus
2. EMT UC data integration with Message Bus
3. CEP integration with Message Bus
4. Data Mapper integration with Message Bus
   a. Test that the Data Mapper can collect data from the Message Bus
5. Data Mapper integration with Cloud Storage, Metadata Search and Storlets
   a. Test that the Data Mapper can generate objects in Cloud Storage with associated metadata
   b. Test that the Data Mapper can apply a storlet to an object at object creation time
6. Machine Learning integration with Cloud Storage, Metadata Search and Storlets
   a. Test that machine learning can be applied to a set of objects in cloud storage
   b. Test that machine learning can define the set of objects to use with metadata search
   c. Test that machine learning can be applied to data in the cloud storage which is pre-processed by storlets

2.2.2. Autonomous Behaviour of VEs

1. Camden UC data integrated with the Message Bus (COSMOS platform scenario).
2. CEP integration with the Message Bus (COSMOS platform scenario).
3. Planner integration with the Message Bus (COSMOS platform scenario).
4. Camden UC data integrated with the Planner (Minimal Planner scenario).
5. Planner Integration with the Experience Sharing (Both Scenarios).
6. Planner Integration with the Social Monitoring (Both Scenarios).

2.3. System sequences and Advanced Integration Tests

2.3.1. System Sequences

2.3.1.1 Data Management

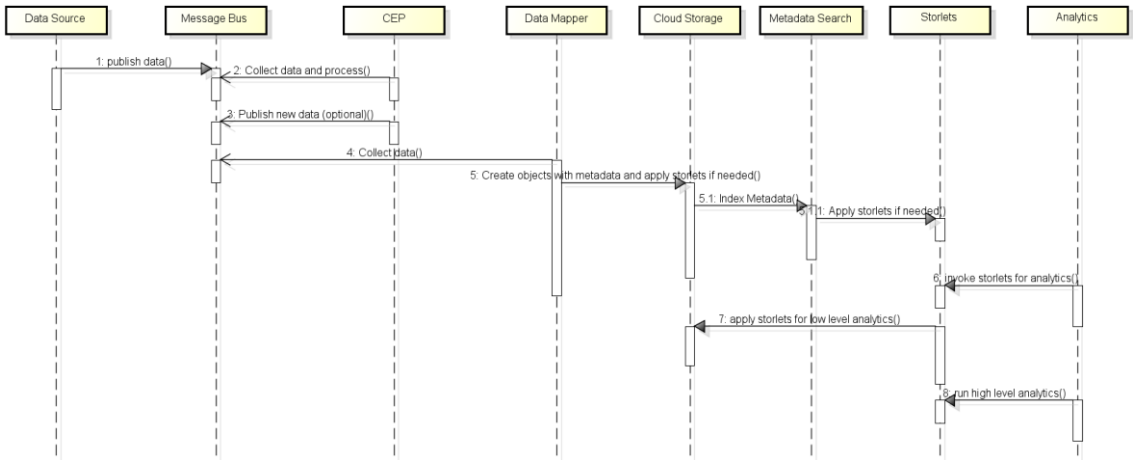

Figure 1: Data Management System Sequence Diagram

2.3.1.2 Autonomous Behaviour of VEs

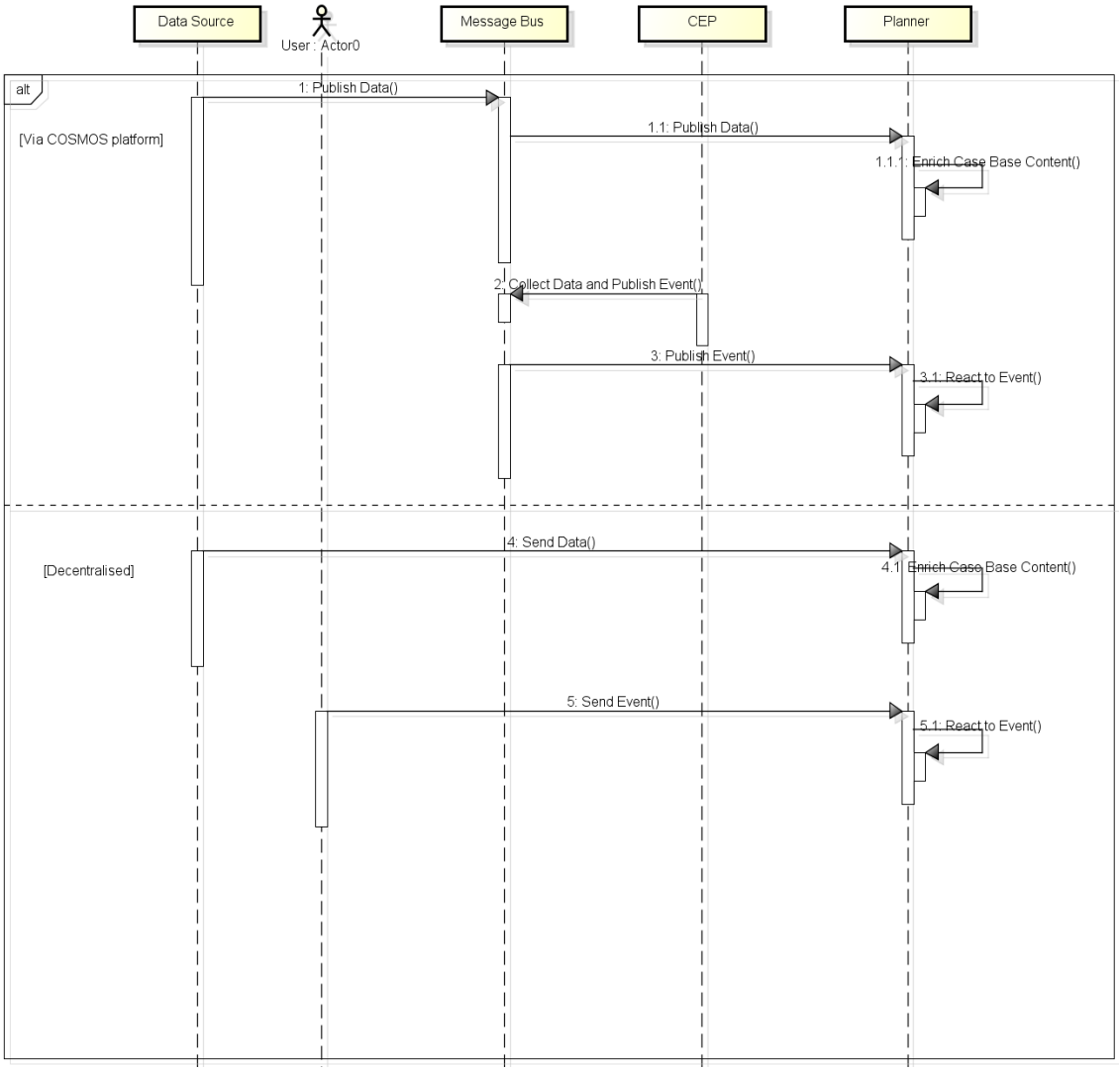

Figure 2: Autonomous Behaviour of VEs System Sequence Diagram

2.3.1.3 Referenced Sequence Diagrams

Enrich Case Base Content() is referenced by the following sequence diagram:

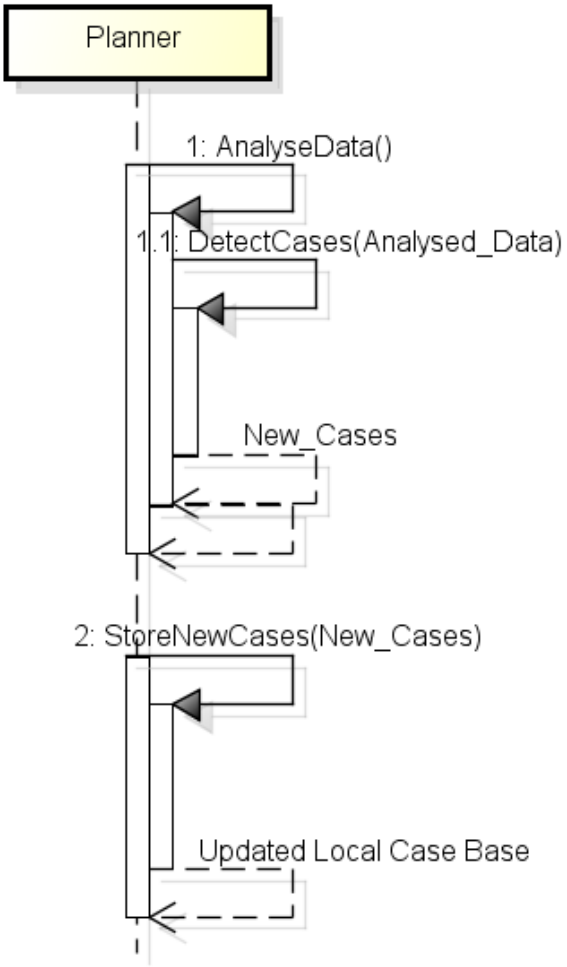

Figure 3: Enrich Case Base Content Sequence Diagram

React To Event() is referenced by the following sequence diagram:

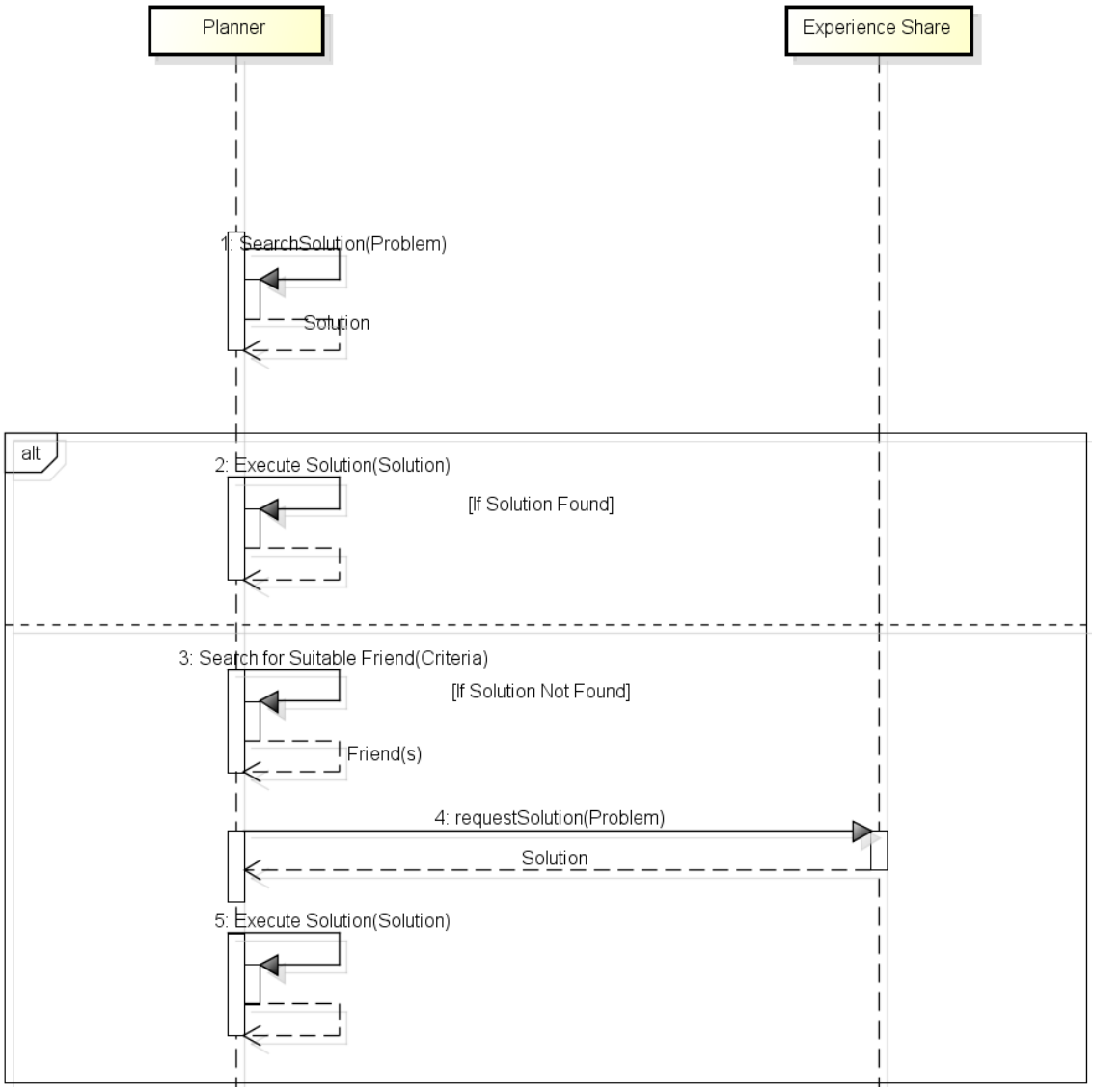

Figure 4: React to Event Sequence Diagram

RequestSolution(Problem) is referenced by the following sequence diagram:

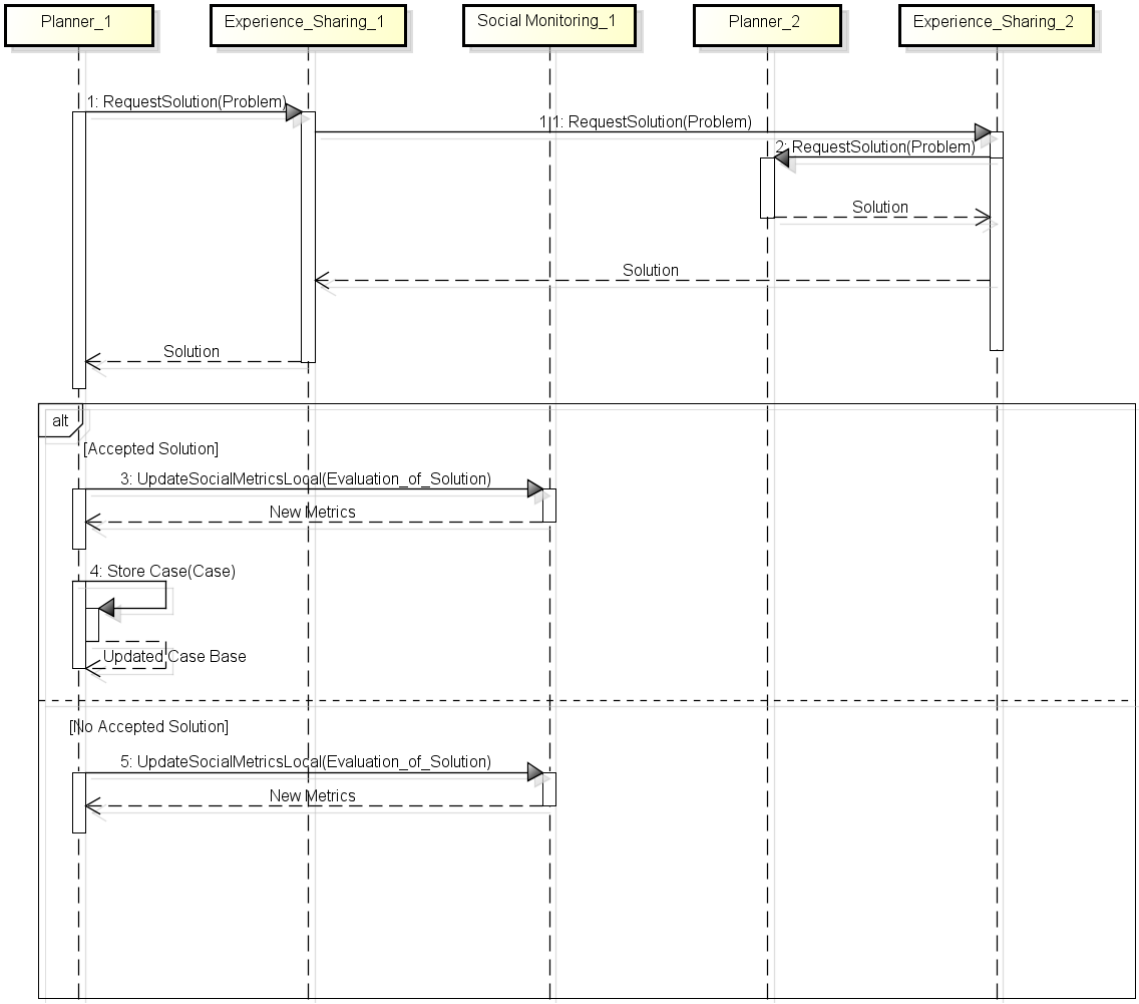

Figure 5: RequestSolution(Problem) Sequence Diagram

UpdateSocialMetricsLocal(Evaluation_of_Solution) is referenced by the following sequence diagram:

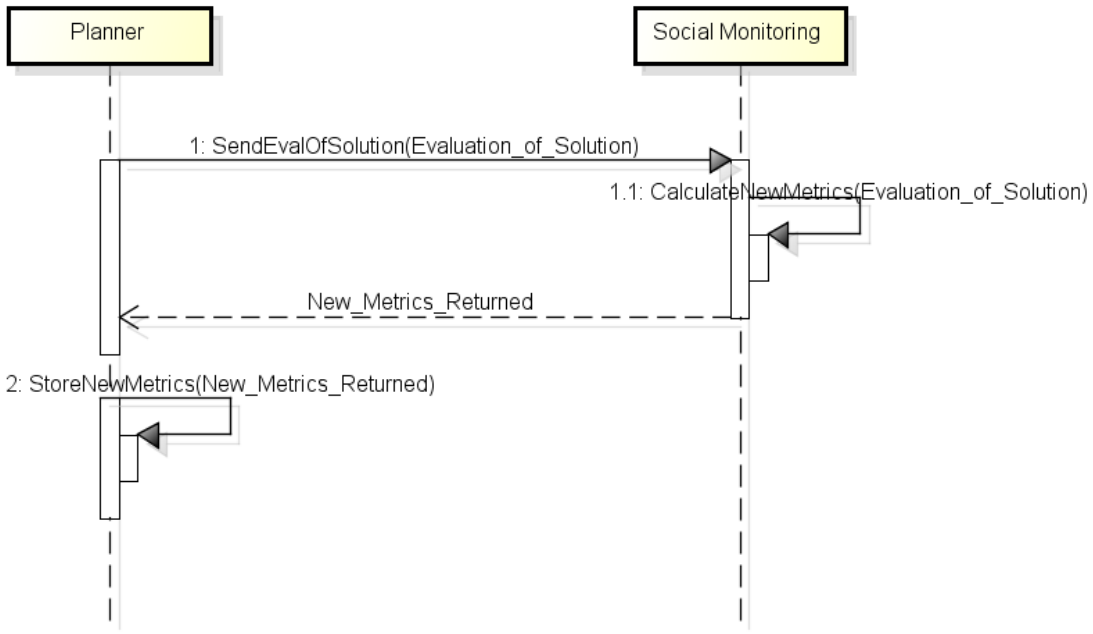

Figure 6: UpdateSocialMetricsLocal(Evaluation_of_Solution) Sequence Diagram

2.3.2. Advanced Integration Tests

2.3.2.1 Data Management

- Test that an end-to-end data flow scenario works as expected using a live data feed.
- Test that a machine learning scenario works as expected using storlets and metadata search.

2.3.2.2 Autonomous Behaviour of VEs

- Validate that the expected chain of actions is uninterrupted from the moment a live data feed is used for Case creation and event detection, to the point of a final Solution acquisition and evaluation (positive or negative).
- Validate that user inputted events are handled in the same way as CEP events and Case creation works without the need for Message Bus connectivity.

3. Demo Scenarios and Objectives for Y1

3.1. Data Management

3.1.1. Involved Components

3.1.1.1 Data Sources

We are considering three potential data sources:

1. Camden housing data published to a mosquito message bus
2. EMT data with information on bus trips stored in large files
3. Real time energy usage data at employee desks at the University of Surrey in large files

The Camden data are published by connecting the mosquito message bus to the COSMOS message bus.

However, by extending the Data Mapper, we enable the data sources that are not connected to the COSMOS message bus to inject their data from static files to the platform and thus to exploit COSMOS functionalities, such as metadata search, visualisation and prediction for historical data. This capability is demonstrated in scenarios 3.1.2.2 and 3.1.2.3.

This may also be necessary for the initialization of a COSMOS platform from an advanced state, by exploiting the historical data, thus eliminating the need for a preparatory stage before the platform goes online.

3.1.1.2 Message Bus

For the Y1 demonstration we plan to have one statically defined topic for each data source.

3.1.1.3 Data Mapper

For the Y1 demonstration, the Data Mapper will be configurable regarding the size of the objects stored in the cloud and their metadata

3.1.1.4 Swift Cloud Storage

The Data Mapper will collect data from the Message Bus and create Swift objects in the cloud storage. We envisage a single object to contain data gathered for a certain Virtual Entity over a period of time. For example, the Virtual Entity could be an apartment in Camden or a bus line in Madrid. A period of time could be a day or a week.

3.1.1.5 Metadata Indexing

The Data Mapper will associate Swift data objects with metadata. When this happens the Metadata Search component indexes this metadata automatically to make it available for subsequent search. Example metadata could be the time period (minimum and maximum timestamps) of data in an associated object.

3.1.1.6 Storlets

Storlets can be applied to data when they are created and/or when they are accessed. In this context storlets which are applied on data creation could be relevant. For example, when data from Camden housing are uploaded we could apply a storlet to downsample the data or to calculate certain statistics about the data.

3.1.1.7 Analytics

Different data mining algorithms including machine learning and time series analysis can be applied on the data to achieve higher-level knowledge that refers to an event, a pattern or an abstract concept.

3.1.2. Demo Scenarios

3.1.2.1 WP4 Data Flow Demo Concrete

In this scenario we aim to demonstrate the flow of data from their generation to their persistent storage in the cloud.

The sequence diagram for this scenario is depicted below:

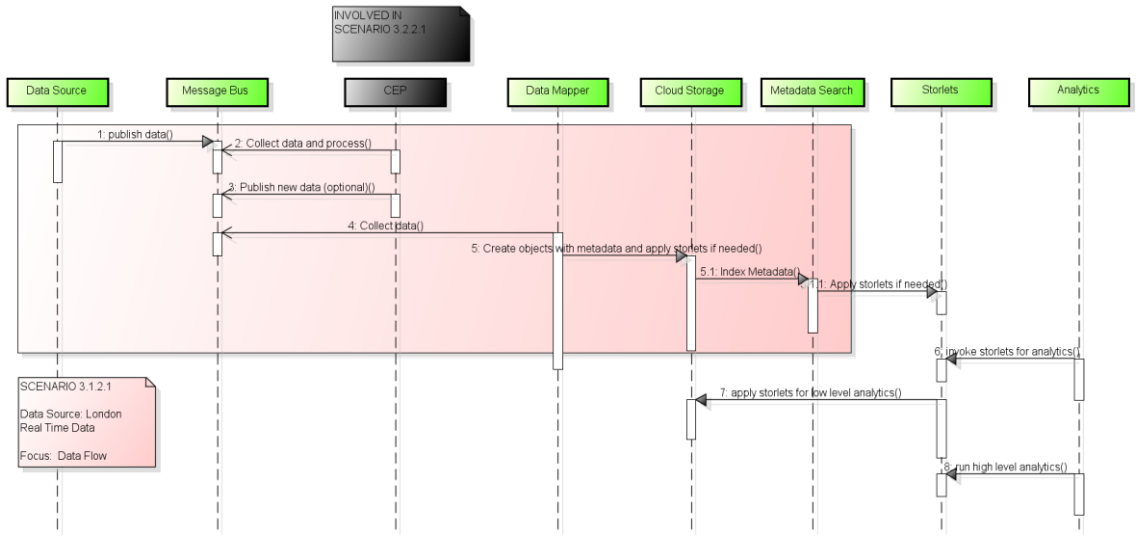

Figure 7: WP4 Data Flow Demo Concrete Scenario

3.1.2.2 Metadata Search

This scenario aims to exploit the Madrid data, which are associated with enriching metadata, in order to focus on the metadata search. Especially, we aim to show how data can be searched according to time intervals or geo-spatial regions. Madrid data source is not yet connected with the message bus, so we use the Data Mapper extension to store the historical data to the cloud storage and apply on them the enhanced capabilities of the COSMOS platform.

The sequence diagram for this scenario is depicted below:

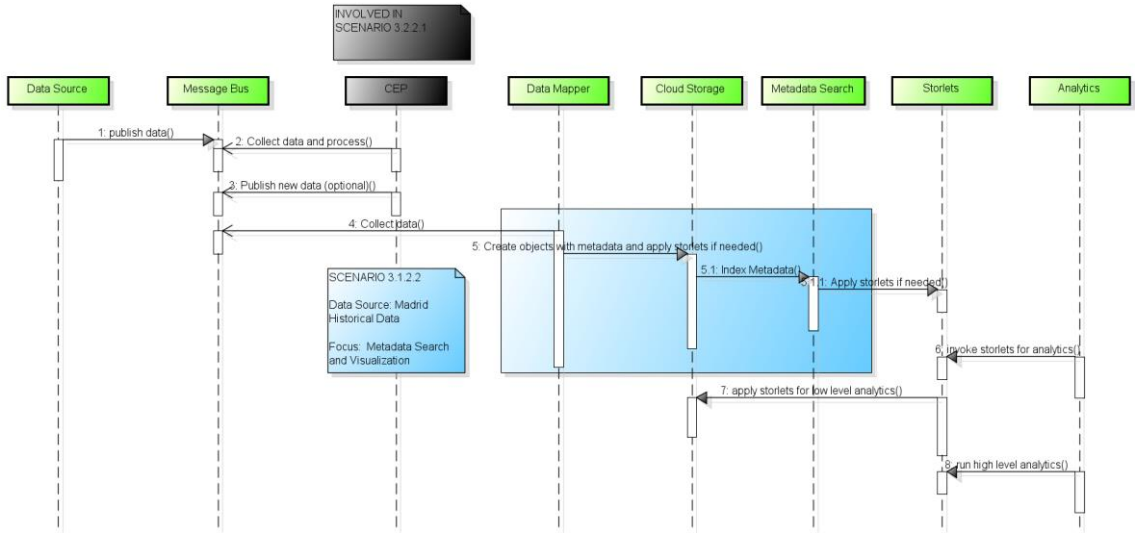

Figure 8: Metadata Search Demo Scenario

3.1.2.3 Data Management and Model Prediction

As above, we load Surrey data from static files in the COSMOS platform, where we execute storlets and prediction algorithms. This functionality was performed this way, since the creation of the models needed input data of a specific form. These data were not available at the time the experiments for model creation were initialized.

Analytically, we aim to demonstrate the following:

1. Pre-processing storlets: Machine learning computations access data from the cloud storage after it has been pre-processed. This reduces the amount of data which need to be sent across the network and also offloads some of the computational load to the cloud storage. In order to have optimized performance, different machine learning algorithms involve several pre-processing tasks such as feature scaling and feature selection. We aim to perform those tasks in storlets for fast processing when dealing with large data sets. We also aim to perform aggregation in order to reduce the total data across the network.
2. Data Analysis: Different variants of classification algorithms are implemented for pattern recognition from electricity consumption data. Classification is a supervised machine learning technique used widely for pattern recognition; it requires labelled data to learn and recognize the patterns. In our case, electricity consumption data with the ground truth reality acts as training data. We divide the total gathered data in the ratio 70:30, where the larger dataset is used for training the classifier and the smaller dataset is used for validating the model.
   Support Vector Machine (SVM) [2] is an efficient algorithm which is widely used for classification because of their two main advantages: 1) its ability to generate nonlinear decision boundaries using kernel methods and 2) it gives a large margin boundary classifier. SVM requires a good knowledge and understanding about how they work for efficient implementation. The decisions about pre-processing of the data, choice of a kernel and setting parameters of SVM and kernel, greatly influence the performance of SVM and incorrect choices can severely reduce the performance of it. The choice of a proper kernel method for SVM is very important as is evident from the results in the next section. The SVM algorithm requires extensive time in training but, once the model is trained, it makes prediction on new data very fast.

On the other hand, K Nearest Neighbours (KNN) [3] is one of the simplest and instance techniques used for classification. It is a non-parametric algorithm which means that it does not make any prior assumptions on the data set. It works on the principle of finding predefined number of labelled samples nearest to the new point, and predict the class with the highest votes. KNN simply memorises all examples in the training set and then compares the new data features to them. For this reason, KNN is also called memory-based learning or instance-based learning. The advantage of KNN lies in simple implementation and reduced complexity. Despite its simplicity, it works quite good in situations where decision boundary is very irregular. Its performance is also very good when different classes do not overlap in feature space. KNN is also called lazy algorithm as it take zero effort for training but it requires full effort for the prediction of new data points.

The sequence diagram for this scenario is depicted below:

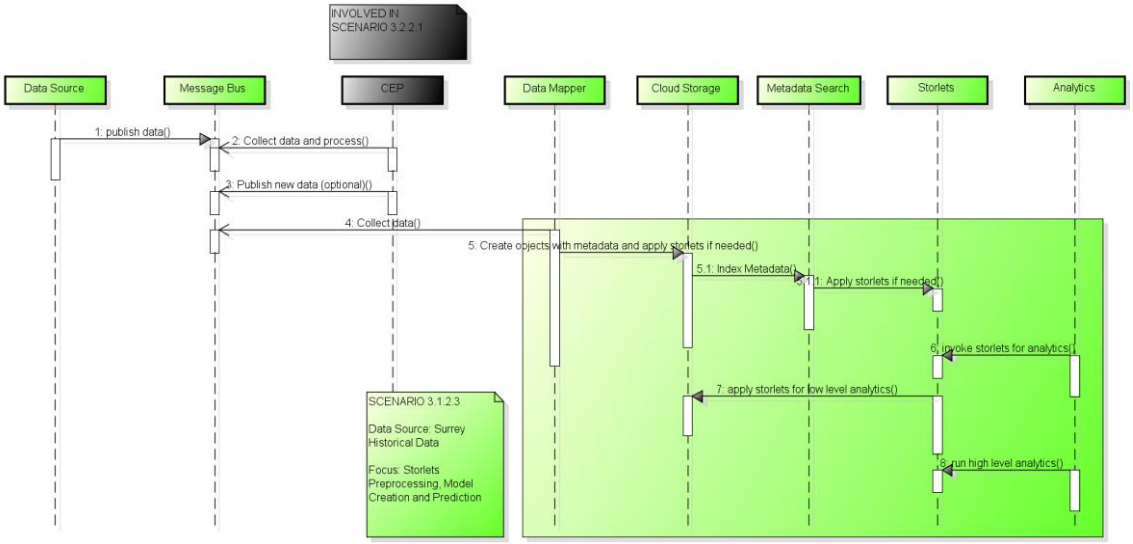

Figure 9: Data Management and Model Prediction Demo Scenario

3.1.2.4 Storlets for Privacy Protection

We aim to demonstrate an example of a privacy preserving storlet which performs facial blurring on photos of people.

The sequence diagram for this scenario is depicted below:

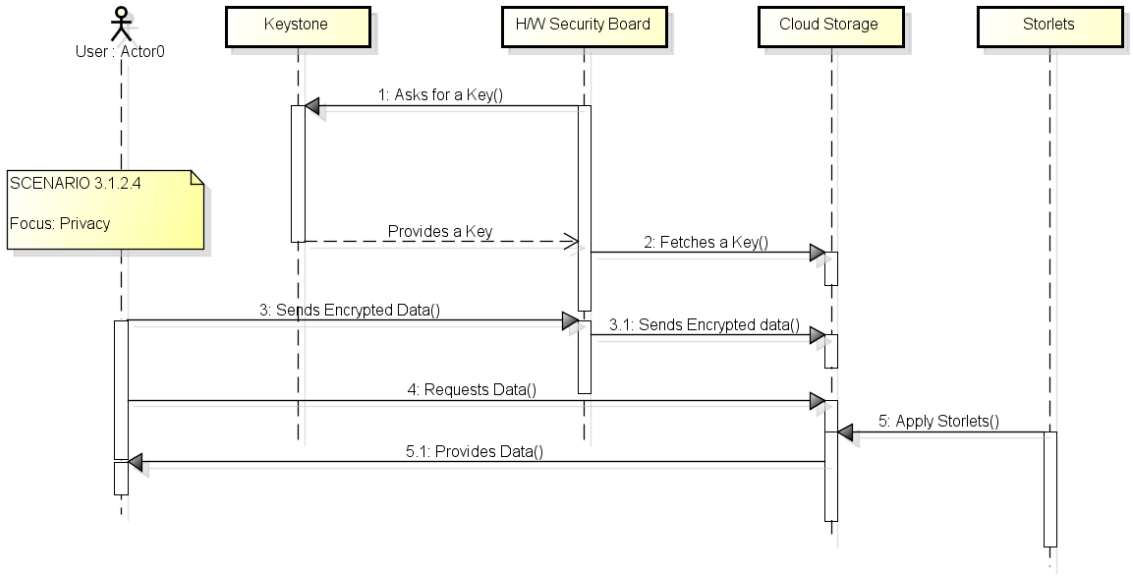

Figure 10: Storlets for Privacy Protection Demo Scenario

The example is focused on demonstrating a "security path" between the camera and the end user. For this reason the demonstrator leverages components from both WP3 and WP4. The joint demonstrator aims at showing:

- Secure key exchange: for each new hardware security board a key exchange session is used to enrol the board into COSMOS. The hardware security board uses asymmetric cryptography (ECDH) in order to securely fetch the symmetric encryption key, used to secure the data flow between the HSB and COSMOS. The HSB makes use of Keystone, which is already part of COSMOS, in order to generate and manage the keys.
- Secure data flow: each data package circulated between the HSB and COSMOS will be encrypted with the key previously exchanged. Both key exchange and data encryption/decryption make heavy usage of the hardware components within the HSB.
- Storlet based data access: Storlets are automatically generated upon each data request from COSMOS. Based on user rights, similar to the operating system approaches, COSMOS configures the Storlet with the user's rights. Therefore not all users will see the same images. The HSB pushes the pictures in full resolution to COSMOS but the Storlets only allow restricted usage of them. Users with restricted access will see the pictures with all faces blurred while unrestricted users will see the picture in full resolution. The mechanism will therefore keep, inside the cloud storage, the original pictures unchanged.

3.2. Autonomous Behaviour of VEs

3.2.1. Involved Components

3.2.1.1 Data Source

For the Data Source of the entire system we are using data gathered by Hildebrand's monitoring of the Camden Apartments.

3.2.1.2 Message Bus

For the Y1 demonstration we plan to have one statically defined topic for the Camden data.

3.2.1.3 CEP

The CEP will draw data from the Message Bus and detect topic-appropriate events for publishing. This is the first scenario's way of triggering an event for the VE Planner.

3.2.1.4 VE Planner

The VE Planner will have two basic functionalities. Firstly, it will use raw data to detect new Cases and store them appropriately inside the local CB. Depending on the scenario the Hildebrand data will come via the MB, or directly to the Planner. Also, depending on the scenario, the event detection will either be automated, courtesy of the CEP, or will be manually provided, by user input (User: Actor).

3.2.1.5 Experience Sharing

At first, the local Planner provides its Experience Sharing component with the problem parameters. Then, the local Experience Sharing component initiates communication with the remote Experience Sharing component. The remote Experience Sharing component requests a solution from its Planner component and returns the provided answer to the origin VE.

If the Planner of the origin VE deems the solution acceptable it updates the locally stored Social metrics (positive feedback) of the remote VE and then stores the new Case inside the local CB. If the solution is not acceptable, it ends the process by updating the Social metrics (negative feedback) and not storing anything.

3.2.1.6 Social Monitoring

The process of updating the local Social Metrics involves the Planner providing the Social Monitoring Component with the evaluation of the solution, whether positive or negative. The SM component calculates new metrics based on the specifics of the Experience Sharing mechanism feedback. After sending back the new metrics, the Planner stores the new values locally inside the local Social Ontology.

3.2.2. Demo Scenarios

3.2.2.1 Planner fully integrated with COSMOS platform

In this specific scenario our aim is to demonstrate that there is a clear chain of integration between the various developed components, especially in regards to “event detection” as a trigger for further Planner actions. The expected chain of observed actions is the Data Source providing a stream of data in a topic of the Message Bus, the VE Planner to use this stream in order to extract cases and at a later time the CEP to begin using the stream as a means to detect events based on its specific rules. After the publishing of an event the VE Planner will initiate action on the specified Problem by using all means possible in detecting a solution.

The sequence diagram for this scenario is depicted below:

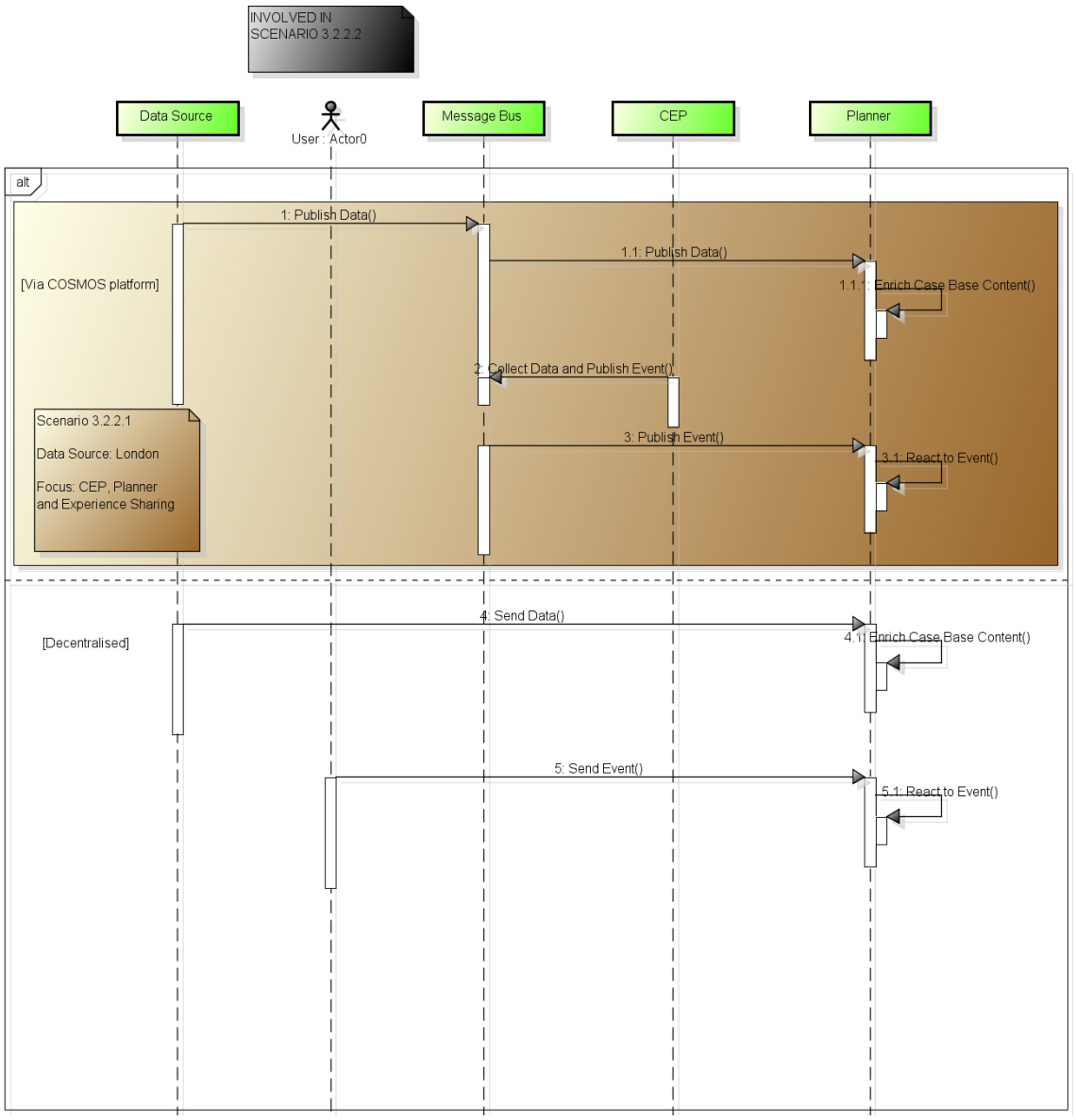

Figure 11: Planner fully integrated with COSMOS platform Demo Scenario

3.2.2.2 Planner with minimal COSMOS platform interaction

In this scenario we aim to present a more minimal version of the expected results in autonomous VE behaviour by demonstrating that the entire process is not reliant on the platform of COSMOS for initiation. By allowing the VE to react to user generated events, we present decentralised VE capabilities for managing their functions. In this scenario, we have removed the connectivity with any platform specific component, like the Message Bus or the CEP and after receiving data for Case creation (simulating local observations), the event is triggered by user input. After that, the VE Planner proceeds as previously in trying to locate a local Solution to the Problem, or to initiate Experience Sharing in order to retrieve a suitable Solution for actuation from a remote "friend" VE.

The sequence diagram for this scenario is depicted below:

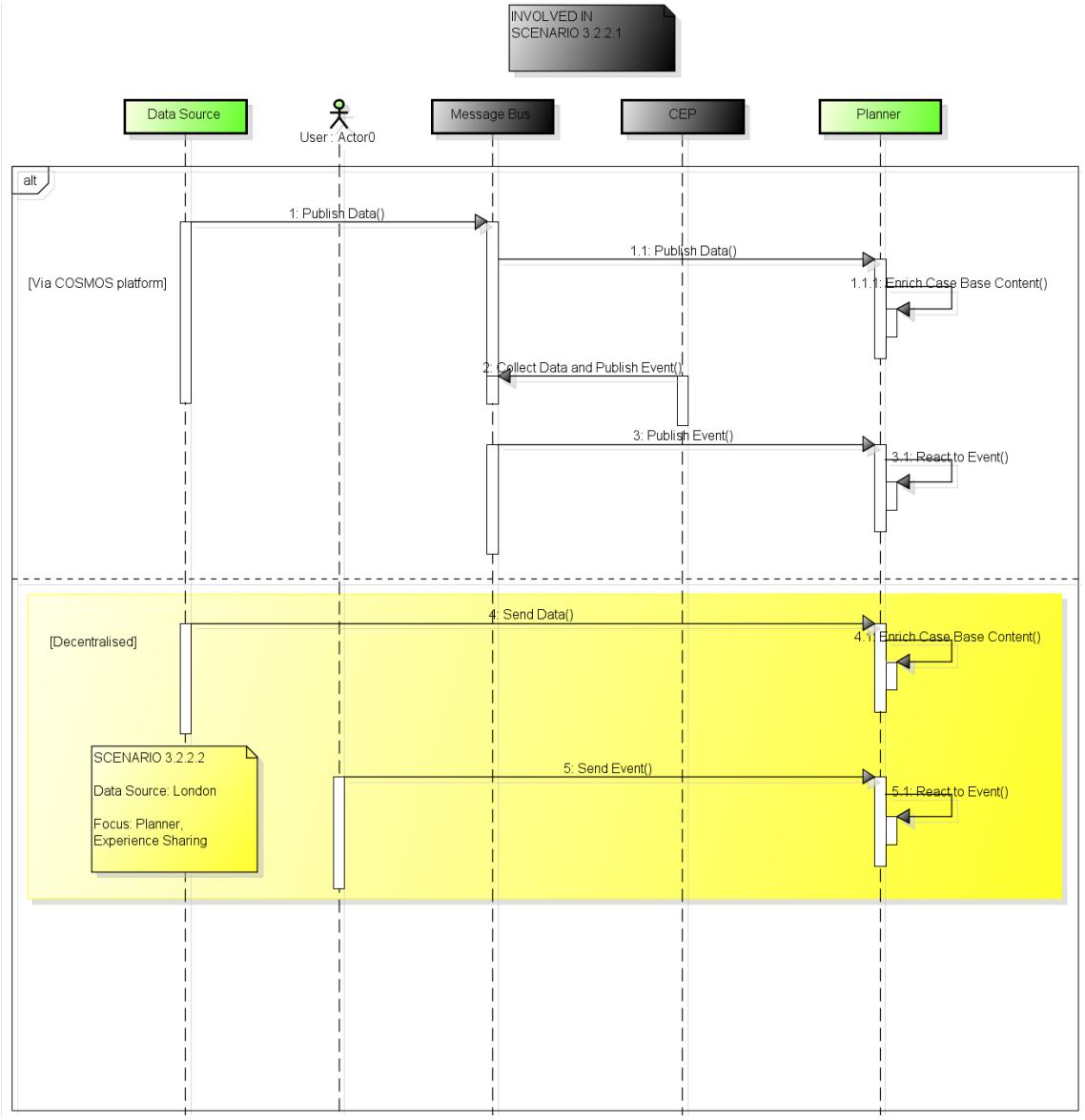

Figure 12: Planner with minimal COSMOS platform interaction Demo Scenario

4. Testbed Instantiation

4.1. WP3

Work package 3 has been focused on the basic functionality which needs to be provided in order to support advanced, hardware-coded, security functions. Therefore the test environment consists of two parts. The first part consist of the hardware development and testing environment which relies on local Linux machines, making use of tools such as Xilinx ISE, Plan Ahead and Mentor Graphics ModelSIM. The second part consists of the Linux environment which runs directly on the hardware security board. This environment is not only used for testing but also for development, directly on the board.

For Y1 needs WP3 has been using 1VM. For Y2 and 3 it is forecasted that WP3 will merge it's functionality into WP4 therefore removing the needs for 1VM.

4.2. WP4

There are 2 VMs for WP4 (wp41 and wp42).

For convenience in Year 1 we plan to deploy Cloud Storage, Metadata Search and Storlets on a single VM (wp41) and Message Bus, CEP and Data Mapper on a single VM (wp42).

4.3. WP5

For the needs of WP5 we have been provided with 2 VMs (wp51 and wp52). Both of these VMs have been supplemented with oracle-java-jdk8 and in the context of Y1 have had secondary VEs installed on them, that simulate the Planner and Experience Sharing Components. Additionally, the VMs have a local CB and a local Social Ontology (friend list).

4.4. WP6

WP6 is currently using 1 VM (wp61). All the dependent libraries along with the required software are installed in it. In next years, we plan to use more VMs for demonstrating distributed computing by dividing the workload in different VMs.

5. Model Validation

5.1. Prediction Models

As described in subchapter 3.1.2.3, the total gathered data was divided into ratio 70:30. We have used the larger data set for training the model while other data set is used for validating the model. More explanation about validating the model is below.

5.1.1. Accuracy of Model

F-measure represents an accurate measure for evaluating the performance of multi-class classifiers (used for pattern recognition) and also one of the most commonly used metric to compare different classifiers. We have also used F-measure to compare the performance and validate the implemented algorithms. We have used three different feature sets for the evaluation of model which are shown in the following table:

No.	Features selected
Feature Set 1, F1	Active Power, Reactive Power
Feature Set 2, F2	Voltage, Current, Phase Angle
Feature Set, F3	Active Power, Reactive Power, Voltage, Current, Phase Angle

The first feature set, F1 consists of only power measurements and includes real power and reactive power. The second feature set, F2 consists of voltage and current measurements along with the phase angle between them. Finally, we have used all the five features for classification algorithms in F3. The complexity of algorithm increases with the number of features, and the selection of inappropriate features can result into complex decision boundary for classifiers effecting the performance of the algorithm as we discussed in the next section.

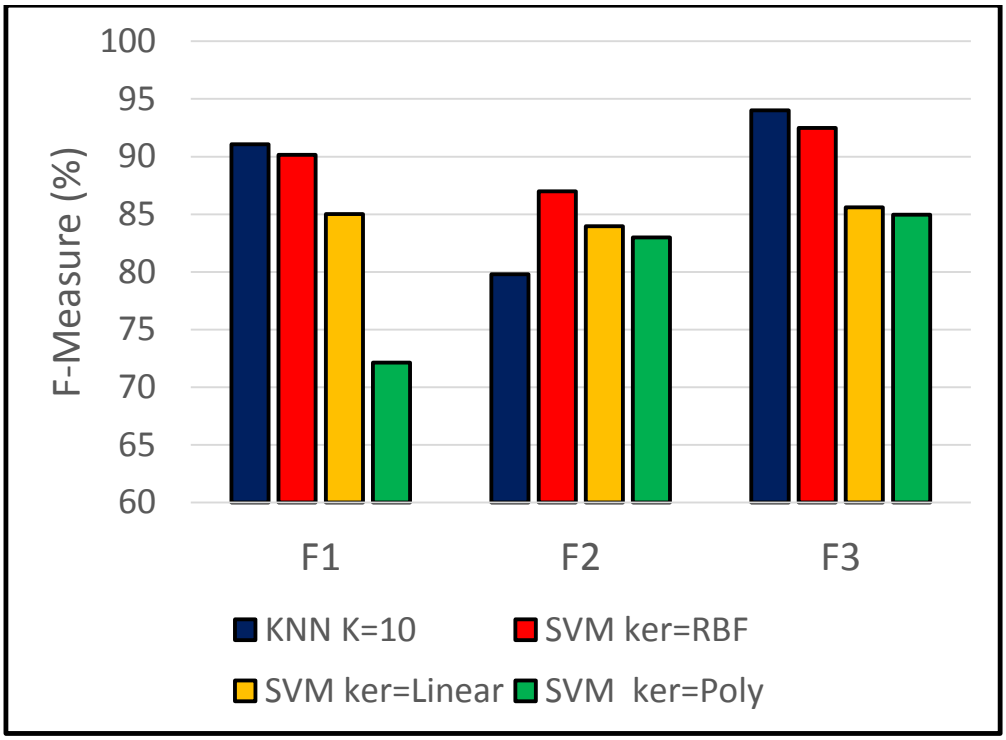

Figure 13: Performance of Classifiers

Figure 13 shows the F-measure plot of different classification algorithms implemented. From the figure, it is obvious that KNN performs best for F1 and F3 and achieves accuracy up to 94.01% while SVM-Rbf (SVM with Radial basis function as kernel) outperforms other algorithms for F2 with maximum efficiency of 86.99%. The reason for good performance of KNN for F1 and F3 is that the power features involved in F1 and F3 for different states are non-overlapping and distinct, and KNN performs very well in such situations. The overlapping nature of features in F2 resulted in the reduced performance of KNN, whereas SVM-Rbf performs better as compared to other variants of SVM. In general, SVM forms an hyper-plane as a decision boundary between different classes in feature space, and the shape of hyper-plane is governed by the kernel function chosen. The spherical nature of hyper-plane for Rbf kernel enables to classify simple and complex problems accurately. SVM-Poly(SVM with Polynomial kernel) performance is degraded for F1 as it tries to over fit the problem by forming complex decision surface. We have used feature set 3 for all further analysis in the paper.

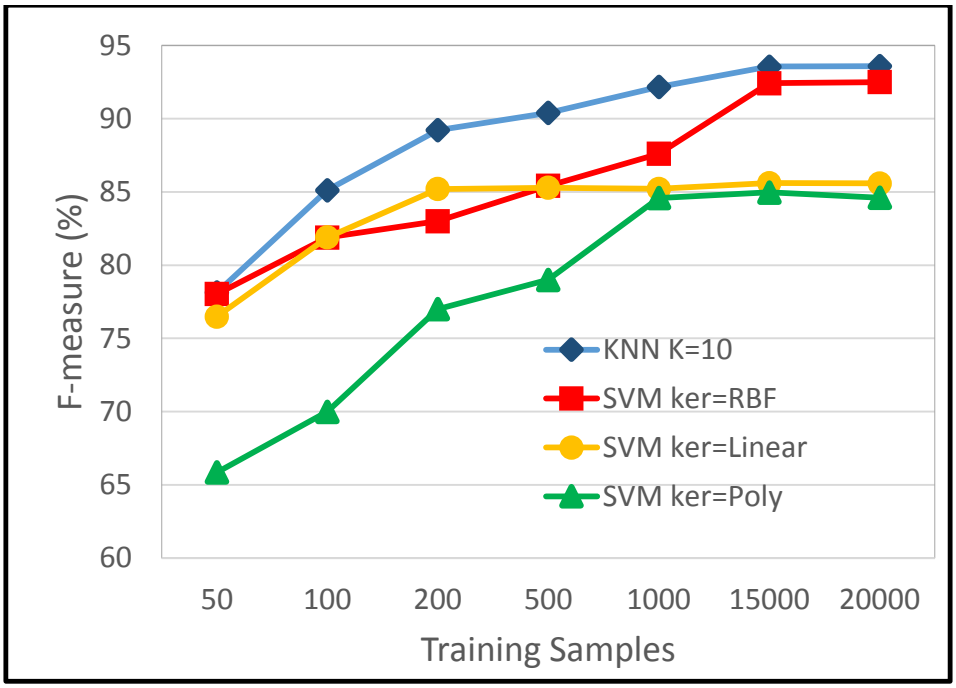

Figure 14: F-measure relation with training data

The number of training samples plays an important role in the performance of a classifier. We have evaluated the performance of our algorithms against different number of training samples. As the training samples increase, the decision boundary becomes more accurate and the performance of classifier improves. Figure 14 shows how the F-measure of different classifiers improve as we increase the training samples. After a certain number of training samples, increasing the training data set does not have much effect on the performance of a classifier. SVM-Poly has the greatest effect on the performance with increasing training samples. SVM-Poly tries to differentiate all training samples by making complex and nonlinear decision boundary. For low data sets, the decision boundary is very specific but when the same model is validated against new data, the same decision boundary may not work accurately and result into degradation of performance. But as the training data set increases, the decision boundary becomes more general and more fitting for new data and hence performance of the classifier increases with increasing data set.

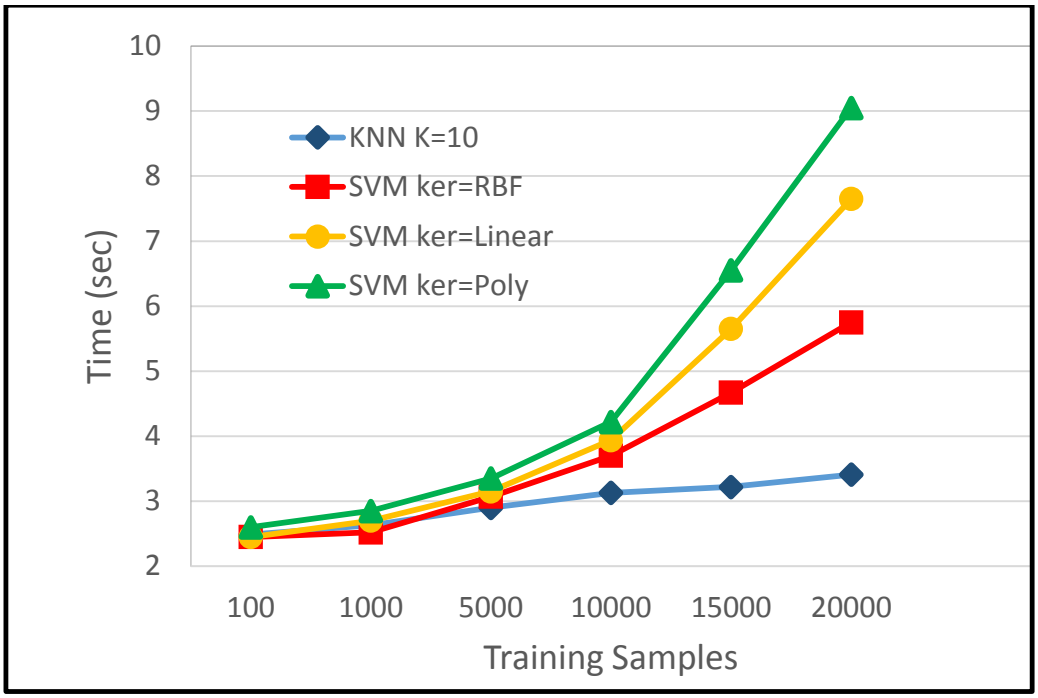

Figure 15: Time Complexity

F-measure is an accurate measure for reflecting the accuracy of a classifier. But as the data size increases, the complexity of the algorithm also plays an important role in choosing a particular technique: a large commercial building can have hundreds of nodes and the amount of data can be huge. In order to give further insight about the performance of different algorithms, we compared the total time which includes training and validation for each algorithm. The amount of time taken by each algorithm for different number of training samples is plotted in Figure 15. The size of validation data size remains the same for all cases. For small number of training samples, the performance of all algorithms remains almost the same. But as the data size increases, complexity and hence the time taken for all the variants of SVM increases rapidly. whereas, increase in time for KNN is significantly low. This is due to the instant learning nature of KNN. During training, KNN simply memorises the all examples in the training data set and used it for comparing and predicting new samples with highest nearby votes. In contrast, SVM implements gradient descent algorithm for finding optimum decision boundary (which is an iterative optimisation algorithm) and results in exponentially increasing time with increased number of training samples.

6. Gaps and Future Work

6.1. WP3

Future versions of will allow for a optimised usage of resources and processing time. Therefore the hardware security modules will be extended to make use of the ARM SoC interrupt controller. Supplementary the design will be optimised with respect to speed and power usage.

The software support (Linux drivers and supporting API's) will be extended to allow usage of ARM's sandboxing mechanism (TrustZone) in order to separate security critical tasks from the rest. This support will be baked into the entire platform therefore allowing a system-wide security approach. Also for the next years secure storage methods will be developed in order to allow safe storage of secret information such as encryption keys.

6.2. WP4

In future versions we aim to allow more advanced forms of analysis (requirement 4.4). Currently we allow storlets to be applied when an object is created or retrieved. In future we plan to allow analyses over sets of objects. Additionally, we intend to associate the objects with enriching social metadata depending on the VE's interactions with other VEs.

We intend to provide a situational knowledge acquisition service which is more suitable for dynamically changing environments. We will provide continuous changing of situation monitoring parameters and analysis and support for external queries.

6.3. WP5

During Y1, regarding the main component of WP5, the Planner, we managed to accomplish some of its main requirements and integrate it with the Experience Sharing component. More advanced requirements are going to be covered during the next two years as it has been described in D5.1.1 [5]. For example, the reasoning techniques used will be expanded in order to achieve GVE forming and management. The first steps for fulfilling the requirements under Task 5.3 (Network of things run-time adaptability) have already been taken.

Regarding the semantic descriptions needed for the project, we have implemented a basic taxonomy of Cases as well as a social and general ontology for VEs. Additional work needed, is the semantic description of IoT-services and possibly of the applications and human users as it can be seen in the requirements annex. These tasks will be completed with the introduction of a basic Registry during Y2.

As for the progress on the Social Monitoring and the Social Analysis components, we estimate that during Y2, the eventual introduction of a VE Registry will facilitate greater progress in the development of these components.

6.4. WP6

Future work includes integration of machine learning with storlets in order to provide optimised solutions for very large data sets. In this context storlets will be used mainly for pre-processing tasks (preparing therefore the Machine Learning specific activities) like feature re-scaling, data re-sampling and Data cleaning and simplification. We will also explore the possibility of making aggregation within storlets in order to reduce the amount of data to be tackled by the Machine Learning process. We will develop also a solution that provides VEs with a good and powerful situation awareness support based on CEP. As far as Experience Sharing is concerned, the

current solution developed during Year 1 will be improved and aligned along the development of the Case Base Reasoning part of WP5. It will also provide a preliminary solution to the Trust and Reputation part jointly developed by WP5 and WP6.

References

[1] COSMOS Deliverable 7.6.1 “Integration Plan”, ICCS/NTUA and other partners, June 2014

[2] Support Vector Machine (SVM): http://en.wikipedia.org/wiki/Support_vector_machine

[3] K-Nearest Neighbours (KNN): http://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

[4] COSMOS Deliverable 2.2.1 “State of the Art Analysis and Requirements Definition”, UniS and other partners, February 2014

[5] COSMOS Deliverable 5.1.1 “Decentralized and Autonomous Things Management: Design and Open Specification", ICCS/NTUA and other partners, April 2014