COSMOS

Cultivate resilient smart Objects for Sustainable city applicatiOnS

Grant Agreement Nº 609043

D6.1.1 Reliable and Smart Network of Things: Design and Open Specification (Initial)

WP6: Reliable and Smart Network of Things

Version: 1.0
Due Date: 30 April 2014
Delivery Date: 15 February 2015
Nature: Report
Dissemination Level: Public
Lead partner: University of Surrey
Authors: F. Carrez (editor) & A. Akbar (Unis), A. Marinakis (NTUA), B. Tarnauca (Siemens), J. Krempasky (Atos)
Internal reviewers: ATOS

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
V0.1		F.Carrez, Adnan Akbar	UNIS	TOC + 6.1 content
V0.2		Previous + Achilleas Marinakis	UNIS, NTUA	6.1 and 6.3 content
V0.3		Jozef Krempasky	UNIS, NTUA, ATOS	6.3 + Intro/Concl + All sections update
V0.4		ALL	ALL	
V0.5	08/05/2014	Previous + Jozef. Krempasky + Adnan Akbar	ATOS, UNIS	Chap 3 & Chap2 updates
V0.6	08/05/2014	ALL	ALL	Fine-tuning
FINAL	12/05/2014	F. Carrez	UNIS	Last check & Delivery to COSMOS management
RE_SUBMIT v0.7	10/02/2015	Adnan Akbar	UniS	Align chapter 2 with state of the art techniques
RE_SUBMIT V1.0	12/02/2015	Juan Sancho	ATOS	Final Review

Table of Contents

Table of Figures

Acronyms

Acronym	Meaning
API	Application Programming Interface
CEP	Complex Event Processing
CRUD	Create/Read/Update/Delete
EM	Expectation Maximization
GMM	Gaussian Mixture Model
GPS	Global Positioning System
HMM	Hidden Markov Model
HTTP	Hyper-Text Transfer Protocol
IGR	Information Gain Ratio
IoT	Internet of Things
IP	Internet Protocol
KNN	K Nearest Neighbor
MaL	Maximum Likelihood
MAP	Maximum A Posteriori
MAPE-K	Monitor/Analyse/Plan/Execute - Knowledge
ML	Machine Learning
MLE	Maximum Likelihood Estimation
MLP	Multi-Layer Perceptron
OWL	Ontology Web Language
PA	Predictive Analysis
PMML	Predictive Model Markup Language
RDF	Resource Description Framework
SA	Situation Awareness
SPARQL	SPARQL Protocol and RDF Query Language (Recursive acronym)
SQL	Simple Query Language
SVM	Support Vector Machine
URL	Unified Resource Locator
VE	Virtual Entity

1 Introduction

The main objective of this work package is to provide methods for inferring high-level knowledge from raw IoT data, to provide the means for situational awareness and understanding how things have behaved in comparable situations previously. Different data mining methods based on machine learning and statistical analysis techniques will be explored and developed for extracting events from raw data. The focus of Year 1 is to explore current methods and highlight the limitations of these methods which will be addressed in the following years. Complex Event Processing engines will be deployed for knowledge inference in complex deployment situations based on near real-time analysis of complex events. We intend to explore Machine Learning techniques in conjunction with *Complex Event Processing* (CEP) in order to provide adaptive solutions for dynamic scenarios to optimize the performance of CEP for situational awareness. Finally different methods for experience sharing between virtual entities will be investigated so that things can be made smarter and able to make decisions using the experience of entities which have already faced identical situations. All of the above mentioned objectives are elaborated in this document with relevant examples.

However it is worth mentioning that this iteration of the deliverable (corresponding to Period 1 of the project) is more focused on exploring machine learning methods, statistical analysis methods and complex event processing engines for context-aware models and highlighting the limitations of these methods. Experience definition and experience sharing is still “work in progress” (the task started later on) and will be further elaborated in D6.1.2 due in nearly 12 months.

This document is intended to provide a broad approach to meet the objectives as discussed above. In the first year, we have tried to explore different existing and possible techniques offering room for research and novelty. In the second year, we will narrow down our approach to the most optimized and novel techniques.

The document is organized as follows: Chapter 2 is linked to task T6.1 objectives about inferring high-level knowledge from incomplete and raw IoT data via different data mining techniques, Chapter 3 (relating to task T6.2) is focusing on situation awareness based on CEP techniques (further methods based on Machine Learning will be explored and proposed in Period 2 and 3), Chapter 4 (relating to task T6.3) explains the concept of Experience and Experience sharing. Finally Chapter 5 provides a conclusion and introduces the next steps.

2 Reliability of information

In the world of Internet of Things (IoT), devices and sensors are deployed or used in varying conditions and different situations. Mostly they are deployed in remote places and connected using less reliable wireless links. In order to prolong their battery life, data provided by these devices may be sporadic and less reliable. Data itself is of no value until it is processed intelligently to extract high-level knowledge which can be used to make decisions. Data mining methods based on machine learning and statistical analysis techniques have the potential to extract knowledge from unreliable and incomplete data. Pattern recognition techniques based on data mining methods have long been used in speech and image processing applications but their use in IoT world is still in its early years.

This section explains different state of the art data mining methods which were explored in year 1 for extracting high level knowledge from raw IoT data and we have demonstrated how these methods have the potential to contribute for novel applications. A high-level knowledge can be any event, some actionable knowledge or some statistical property of the data depending on the application. Data from different sensors in IoT form patterns, and different patterns represent different events. However, only certain events are interesting depending on the context of application and require further action. Pattern recognition methods based on data mining algorithms enable to extract these interesting events which have the potential to form the basis for many interesting applications. We have demonstrated in our work, how these algorithms can be applied in IoT domain for novel and innovative applications.

The performance of most of the Machine Learning Models deteriorates when applied on real-time dynamic environments. One of the main factors contributing in deterioration of ML models is concept drift which occurs when the statistical properties of the target variable or the statistical distribution of the observation data changes over time. Secondly, the complexity of machine learning algorithms increases exponentially with the amount of data. Most of the innovative solutions found in literature based on machine learning and statistical analysis techniques are implemented on small data sets and are not usable for large scale IoT applications. In this regard, the aim of Year 2, leveraging on Year 1 experiments, will be to focus on machine learning solutions which are able to cope with the dynamic nature of underlying IoT data in real time and also able to deal with the largeness of IoT data for smart city applications.

2.1 Machine Learning

Machine Learning represents any algorithm or process which learns from the data. The availability of data from large number of diverse devices which represent real time events has motivated the researchers to explore more intelligent solutions using machine learning algorithms. Whether it is in the context of a health department, supply chain management system, transportation or the environment, methods based on machine learning enables to provide more intelligent and autonomous solutions by analysing and providing further insight. Different data mining algorithms have been used in very diverse contexts, detecting traffic congestion [Marfia, et al. 2013], predicting the energy demand of the buildings [Salsbury, et al. 2013] and predicting the behaviour of customers in an online shop [Limayem and Cheung. 2008] are few examples of using data mining algorithms in context of Internet of Things. There

are many ML algorithms found in the literature, but in a broader context they fall into following three main categories.

- Classification Analysis
- Clustering Analysis
- Regression Analysis

The focus of this section will be on the classification analysis as the focus of year 1 development was based on these techniques. We intend to explore using other possibilities in year 2.

An efficient implementation of any Machine Learning algorithm involves different steps ranging from filtering, interpolation, and aggregation on one end to optimization methods on the other. A proper understanding of a problem is mandatory to apply the right choice of steps. The different steps involved in machine learning algorithms with possible options are shown in Figure 1 . A brief introduction to few of these techniques is given in this section.

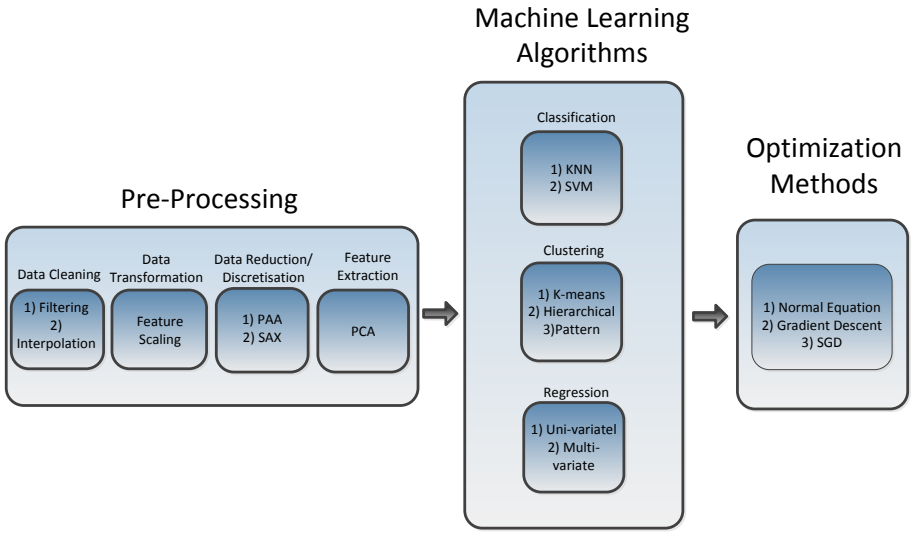

Figure 1: Efficient Implementation of Machine Learning

2.1.1. Pre-Processing

Pre-processing is an important step for applying machine learning algorithms in IoT due to many reasons. Most of the devices are connected with wireless links in a dynamic environment and resource constraint nature of these devices affects the communication link and their performance. The deployment of cheap and less reliable devices is common in IoT to bring the overall cost of a system down resulting in missing values, out of range values or impossible data contributions. The phrase "**garbage in garbage out**" fits perfectly for many machine learning algorithms. The amount of data is increasing exponentially in IoT and the processing of such large data with minimum time latency is an important factor which can be optimized by the use of proper pre-processing methods. Several aggregation techniques are commonly used in IoT for reducing the total amount of data traveling through the network. In this context, *Piecewise Aggregation Approximation* (PAA) and *Symbolic Aggregation Approximation* (SAX) are the most common techniques. Data collected in IoT can be a combination of many

features. The number of features plays an important role in the complexity and the performance of a machine learning algorithm. In few scenarios, these features are correlated with each other and it is possible to reduce the features by representing the data using new uncorrelated features using statistical analysis such as *Principal Component Analysis* (PCA). A brief introduction about the most commonly used pre-processing techniques is given below.

2.1.1.1. Data Cleaning

In real time dynamic environment, a faulty or a missing sensor reading may occur due to bad communication channel or loss of service. The missing values can result in irregular time series or incompatible vector size as compared to other devices connected. A simple data cleaning method involves then filtering out-of-range values and filling out missing values. Missing values can be filled by the mean value of the sensor over some time window, by last recorded value or by simple interpolation methods using historical data.

2.1.1.2. Data Transformation

Data transformation involves transforming the data into the form which is optimum for machine learning process. Feature scaling is an important example which is used extensively as a pre-processing step for machine learning algorithms [Tax and Duin. 2000]. The range of values of different features are on different scales. If one of the features has considerably wider range as compared to other features, the optimization function for the machine learning algorithm will be governed by that particular feature and will affect the performance of algorithm. Also, it will take much longer time for the optimization objective to converge in such cases. Feature scaling is therefore a method that brings all the features on the same scale making sure they contribute equally to classification algorithm. Standardization is most commonly method used for feature scaling which involves having each feature represent as a Gaussian like curve with zero mean and unit variance.

2.1.1.3. Data Reduction

Data reduction is perhaps the most important pre-processing step when dealing with very large data sets. Several variants of aggregation techniques are used in order to reduce the data size without any loss of information. PAA and extended version of PAA called SAX are the most commonly used aggregation techniques in IoT for data reduction [Lin, et al. 2003].
PAA is a simple dimensionality reduction method for time series analysis. In order to reduce time series from **n** dimensions to **m** dimensions, the data is divided into **m** equal sized frames and the mean value is calculated for the data lying in that frame. The vector of mean values calculated will represent new series with m dimensions.

SAX is derived from PAA by introducing another step called discretization. Almost all of the real time series are a combination of infinite real values which limit the application of algorithms for discrete data such as Markov models, hashing and decision trees. SAX discretizes the time series using symbolic representations. The mean values calculated using PAA are represented by predefined number of symbols in SAX. Variants of SAX such as sensorSAX [Ganz, et al. 2013] have been used in IoT as an important pre-processing step.

Feature extraction is another technique used widely for data reduction where the number of features of data are large and mostly correlated to each other. Feature extraction enables to extract most relevant and uncorrelated features in order to perform optimum analysis [Kononenko and Kukar. 2007]. PCA [Jolliffe. 2005] is one of the most famous feature extraction technique which form new uncorrelated features based on the statistical properties in order to represent the same data with less dimensions. PCA is widely used in image processing and pattern recognition systems.

2.1.2. Classification Analysis

Classification is a supervised machine learning technique which requires labeled data in learning phase used widely for pattern recognition and categorization of new observations. The classification of emails as SPAM or NOT-SPAM (a.k.a. spam filters) represents a well-known example of classification analysis [Provost. 1999].The server learns from the user behavior, whenever the user marks (label) emails as spam. It looks for the key words in the marked spam email and if it detects the repetition of those keywords in new mails, it will mark them as spam. There are many variants of classification tools found in the literature. The authors in [Wu, et al. 2008] included several classification algorithms in top 10 data mining algorithms including *Support Vector Machine* (SVM), *K-Nearest Neighbor* (KNN), decision trees and Naive Bayes'. Each classifier has certain advantages and disadvantages and the selection of any particular classifier depends on the data characteristics.

Support vector machine (SVM) is one of the most widely used classification algorithm. The two main advantages which gives SVM an edge on others are:

1. Its ability to generate nonlinear decision boundaries using kernel methods.
2. It gives a large margin boundary classifier.

SVM requires a good knowledge and understanding about how they work for efficient implementation. SVM works by mapping the training data into a high dimensional feature space, and then separates the classes of data with a hyper plane, and maximizes the distance which is called the margin. It is possible to maximize the margin in feature space by using kernels into the method, which result into non-linear decision boundaries in the original input space. The decisions about pre-processing of the data, choice of a kernel, and setting parameters of SVM and kernel influence the performance of SVM greatly and incorrect choices may severely reduce the performance of SVM as discussed in [Ben-Hur and Weston. 2010]. It is also possible to use mixture of different kernel functions for optimized performance and one such example is given in [Tian, et al. 2012], where authors used SVM for image classification with kernel function which is mixture of *Radial Basis Function* (RBF) and polynomial kernel. The SVM algorithm requires extensive time in training but once the model is trained, it makes prediction on new data very fast.

Another efficient and simple classification algorithm is KNN, which is one of the simplest and instance-based learning technique used for classification. It is a non-parametric algorithm which means it does not make any prior assumptions on the data set. It works on the principle of finding predefined number of labeled samples nearest to the new point, and predict the class with the highest votes. KNN memorizes labeled data in the training set and then compares the new data features to them. The advantage of KNN lies in simple implementation and reduced complexity. Despite its simplicity, it works quite well in situations where decision

boundary is very irregular. Its performance is also very good when different classes does not overlap in feature space [Shakhnarovich, et al. 2006]. KNN is also called lazy algorithm as it takes zero effort for training. However, it requires full effort for predicting for new data points [Shakhnarovich, et al. 2006].

2.2 Statistical Analysis

Classification Analysis method described in previous section is an example of supervised ML model where parameters of the model are estimated from training data which consist of pairs of input and annotations of the output. The performance of supervised models are often quite good but the labelling of data poses an extra task. Data is often labelled manually, and although the process of labelling may be simple but it limits the amount of examples that can be classified using manual methods. And as the annotation task become more complicated such as for labelling data for traffic congestion, annotations becomes far more challenging. In this regards, we explored a statistical inference approach based on HMM for predicting an event in scenarios where labelled or annotated data is not available. For such scenarios, the training of model for predicting output without the availability of annotated output data seemed counter intuitive, but it is possible using expectation maximization algorithms.

2.2.1. Hidden Markov Model

Hidden Markov Model (HMM) is an extension of Markov model in which the states of the system are not directly observable as contrast to simple Markov models like Markov chain where state is visible to the observer. Markov model is an example of statistical model which assumes Markov property to define and model the system. In simple words, Markov property states that the future state of the system only depends on the present state and does not depend on the past values. Each state has some probability distribution related to the output, and therefore sequence of outputs generated gives indirectly information about the sequence of states as shown in Figure 2 where at every sampling time instant, we have an observation which is related to hidden state. HMM is widely used for temporal pattern recognition.

There are many applications in literature where Markov model and HMM have been applied to find a temporal relations in the data. One such example is given in [Ganz, et al.], in which authors showed the use of HMM to find the possible transition of events (states) depending on their temporal relation. The authors demonstrated their approach on the energy data which they gathered using the test bed installed in their research centre. They first applied clustering to find the possible hidden concepts or states and name it as weekday or weekend; and then applied HMM on top of it to show the possible state sequence and probability of transition from one state to other. For example, if there were consecutive four weekdays registered, then there will be very high probability that the next day will be a weekend. HMM can also be used as a classification tool for inferring events. For example, the authors in [Kamijo, et al. 2000] applied it for accident detection. It is a quite generic tool and used extensively for classification purpose in different fields as evident by their use in [Kallberg, et al. 2010]. The main difference from the classification analysis techniques discussed in previous section is that HMM is a parametric technique which is based on the statistical properties of the underlying data with respect to time. HMM can also be used for predicting the optimal and most probable sequence of states or outcomes using *Expectation Maximization* (EM) algorithm as used by authors in [Yu, et al. 2003].

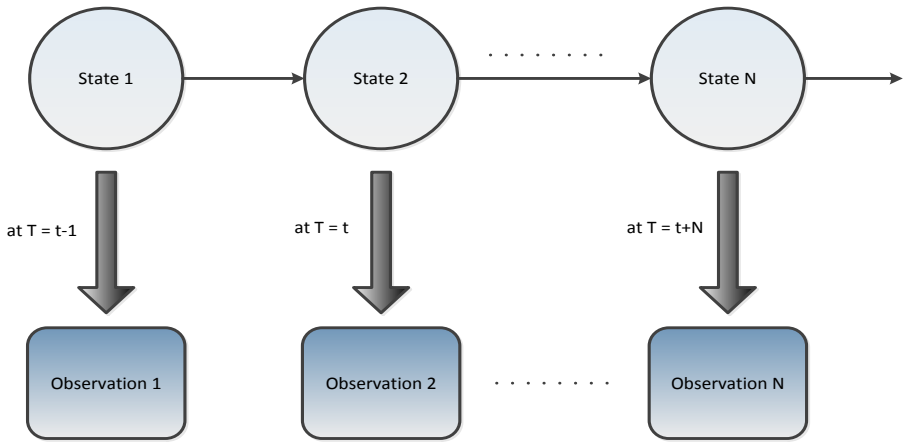

Figure 2: Hidden Markov Model

A complete description of HMM requires specification of total number of hidden states N, and the specification of three probability measures represented by $\lambda = (A, B, \pi)$, where:

- π represents the set of prior probabilities for every state
- A represents the set of transition probabilities a_{ij} to go from state i to state j: $a_{ij} = P(q_t = S_j | q_{t-1} = S_i)$ and collected in matrix A;
- B represents emission probabilities which is the probability of observation in a particular state: $b_{i,k} = P(x_n = v_k | q_n = S_i)$, the probability to observe v_k if the current state is $q_n = S_i$. The number $b_{i,k}$ is collected in a matrix B.

There are three basic problems of HMM in order to apply for real world applications. They are:

- **Problem 1**: Given the observation sequence $O = O_1, O_2, \dots O_t$ and a model $\lambda = (A, B, \pi)$, how to efficiently compute the probability of observation sequence given the model $P(O|\lambda)$?
- **Problem 2:** Given the observation sequence $O = O_1, O_2, \dots O_t$ and a model $\lambda = (A, B, \pi)$, how do we choose a corresponding state sequence $Q = q_1, q_2, \dots q_t$ which best explains the state sequence?
- **Problem 3:** For the given observation sequence $O = O_1, O_2, \dots O_t$, how to find a model parameters which maximize $P(O|\lambda)$?

The problem 3 is of particular interest for unsupervised learning in which only observations are given, and the hidden states are estimated using Expectation Maximization (EM) algorithms based on Maximum Likelihood theory.

2.3 Use case of Data analysis techniques for inferring high level knowledge in COSMOS

We have used smart energy scenario to demonstrate the use of different pattern recognition techniques in year 1. The data used is from university of Surrey IoT-testbed for the following two main reasons.

1. The controlled environment in the university makes it possible to gather labelled data with the users feedback for supervised machine learning methods and for validating the performance of statistical inference methods;
2. The nature of data is quite similar to London use case scenario and III Taipei scenario.

The methods applied are generic and can easily be applied to other use case scenarios.

2.3.1. Introduction

One of the core objective of COSMOS is to provide truly smart building capabilities by contributing towards more automated and innovative applications. In this regard, occupancy detection plays an important role in many smart building applications such as controlling heating, ventilation and air conditioning (HVAC) systems, monitoring systems and managing lighting systems. Most of the current techniques for detecting occupancy require multiple sensors fusion for attaining acceptable performance. These techniques come with an increased cost and incur extra expenses of installation and maintenance as well. All of these methods are intended to deal with only two states; when a user is present or absent and control the system accordingly. In our work, we have proposed a non-intrusive approach to detect an occupancy state in a smart office using electricity consumption data by exploring pattern recognition techniques. The contextual nature of pattern recognition techniques enabled us to introduce a novel concept of third state as standby state which can be defined as

"A state when a user leaves his work desk temporary for a short period of time and switching off HVAC and other equipment associated with occupancy state is not optimal choice"

The intuition behind our approach is that the pattern of electricity data of user will be different when the user will be at his desk and using his appliances as compared to other states. And if our algorithm recognize the pattern, it can infer the state of user as well. Figure 3 shows the current and power variations for random samples from the observation data.

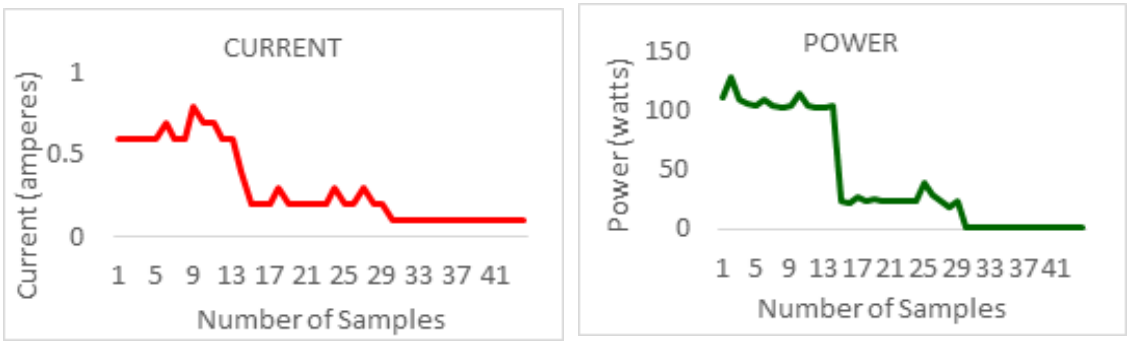

Figure 3: Input Current and Power waveforms

Furthermore, our proposed solution does not require extra equipment or sensors to deploy for occupancy detection as smart energy meters are already being deployed in most of the smart buildings.

2.3.2. Background

Occupancy Information plays an important role in intelligent buildings for providing optimized and automatic energy-efficient solutions for heating, ventilation and air conditioning (HVAC) systems, it also forms basis for automatic lighting systems, and also contributes in contextual models for defining user activities. According to [Pérez-Lombard, et al. 2008] buildings are one of the main consumers of energy and accounts for 39% of total energy consumed in UK. The conscious and conservation behavior of the users can result into large amount of energy saving in buildings as demonstrated by authors in [Arens, et al. 2005]. However, the actions of the users are likely to change over time. Thus recent research efforts are intended to provide automatic solutions for energy management by linking occupancy information to the HVAC and lighting control systems in intelligent buildings. For example, the authors in [Lu, et al. 2010], used occupancy information to control home heating system and demonstrated significant decrease in gas usage for heating as compared to other traditional methods. Another example in [Guo, et al. 2010] demonstrated that real time occupancy information can also be used for control of lighting systems resulting in significant amount of energy saved. Occupancy detection also find applications in intrusion detection systems by detecting an activity at abnormal times. This concept is also used to provide medical assist for old or elderly people.

In spite of many application scenarios, occupancy detection in buildings is still a complex and expensive process and offering research to overcome issues such as false detection, intrusive nature of visual sensors and energy issues with battery powered sensors [Nguyen and Aiello. 2013]. The most common devices used for detecting an occupancy are based on Passive Infrared (PIR) and/or ultrasonic technologies, microwave and audible sensors, video cameras and Radio-frequency identification (RFID) based systems. PIR sensors are used extensively for detecting occupancy by detecting infrared direction radiated by human body movement. Microwave and ultrasonic sensors works on the Doppler shift principle, they transmit waves and depending on the change in patterns of the reflected wave, they detect occupancy. But one of the major drawback associated with such motion dependent sensors is if the user remains inactive for some time, they may not detect presence [Guo, et al. 2010] and may result into annoying and uncomfortable situations such as switching off lights or switching off heating systems even in the presence of the user. Due to these drawbacks, they are mostly used in conjunction with audible sound sensors. But audible sound sensors are not able to distinguish between human and non-human noises and are prone to false alarms.

The aim in most of the occupancy sensing infrastructure is to keep the overall costs low so that widespread use of such techniques can be promoted. This often implies that only few, cheap and possibly imprecise sensors are available, also, battery powered sensors are often used to avoid the deployment of power cables. It incurs extra cost of maintaining the batteries and as the batteries discharge, it can affect the performance of the sensors as well. Faulty installations and lack of maintenance also result in degradation of the sensors performance. Recently, the rise of IoT has initiated a trend towards the use of existing technologies such as mobile phones and Wi-Fi to detect occupancy in an opportunistic manner.

2.3.3. Proposed Method for Machine Learning

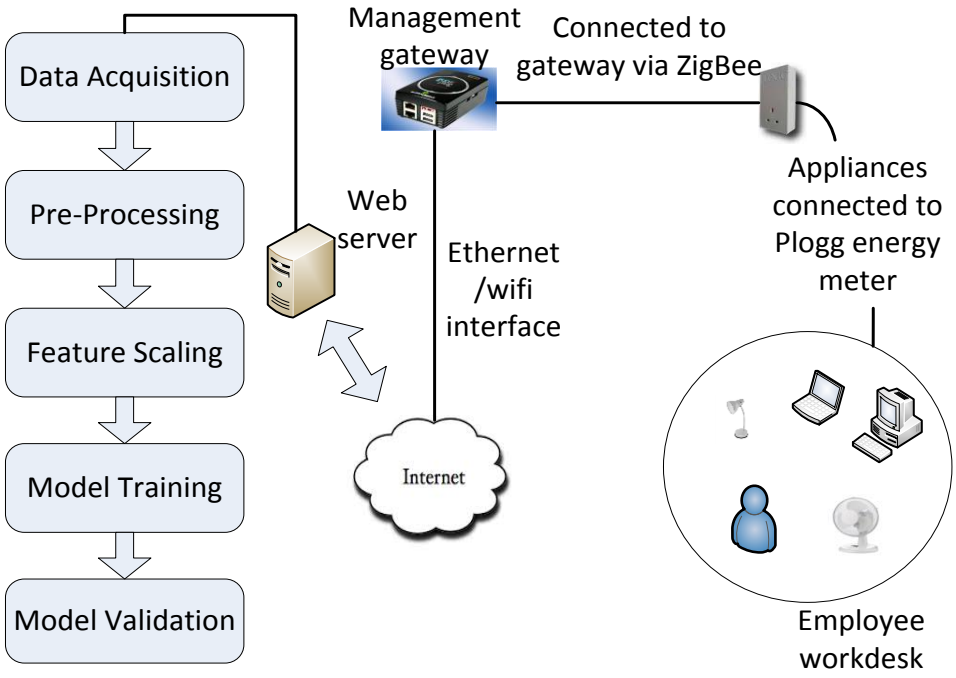

Figure 4: Proposed Architecture and Block Diagram

The proposed architecture for our approach is shown in the Figure 4 along with the block diagram. The intuition behind the proposed approach is when the employees or users are at their work desk, the electricity consumption patterns will be different as they will be using their computer, laptop, table fan, lamp or other appliances as compared to when they are not present. The contextual nature of electricity data enables us to detect an extra state which we called as standby state when the users leave the office for small breaks. We have implemented classification algorithms for pattern recognition from electricity consumption data. More details about the different components involved in our approach are described below.

2.3.3.1. Data Acquisition

Data is acquired from employees work desks within our research centre using smart energy units. These smart energy units have in built energy meter and Zigbee module and keep track of energy usage at the desk by logging electrical data measurements. It is connected to the management gateway via ZigBee and gateway is connected to internet through WiFi module. In our experiment, we have gathered data from the work desks of four employees which have different appliances connected with the smart energy unit.

Different employees have different behavior as far as energy saving and usage of their devices are concerned [Murtagh, et al. 2013]. Most of the employees usually switch off the table lamp and table fan before leaving their offices, even if it is for a short period. And they might switch the work stations to standby mode, or it can automatically go to the standby mode depending on the employee`s behavior and settings. But nonetheless, there might be few employees who do not care much about switching off extra appliances as discussed in [Murtagh, et al. 2013]. In this regard, we did not instruct the employees to follow any specific behaviour, rather they followed their natural response. The ground truth data is gathered using a simple application

on Samsung tablet, in which an employee can register his state as either **present**, **standby** or **absent** just by clicking on the appropriate icon. Data is gathered for approximately two weeks and the total gathered data is divided into two parts with ratio 70:30 after assigning ground truth realities and shuffling the data randomly. The larger data set is used for training the classifier models while the other data set is used for validation purposes.

2.3.3.2. Pre-processing

Every smart energy node transmits the electricity consumption data every 10 seconds. In real time environment, a sensor node connection might be disconnected or crashed and it may fail to transmit the next reading. In order to overcome the missing values, a simple algorithm is implemented to fill out the missing values with the last recorded value. It is a reasonable assumption that the state did not change in consecutive readings for a sampling period of 10 seconds.

2.3.3.3. Feature Scaling

The range of values of different features is on different scale. If one of the features has considerably wider range as compared to other features, the optimization function for classification will be governed by that particular feature and make the classifier unable to learn from other features correctly. Furthermore, it will take much longer time for the optimization objective to converge in that case. Feature scaling is a method to bring all the features on the same scale so they contribute equally to classification algorithm.

Dealing with large data sets, feature scaling becomes an important step otherwise algorithms like Support Vector Machines (SVM) can take quite a longer time. We implemented standardization method for feature scaling which results in having each feature as zero mean and unit variance. The general expression for feature scaling using standardization is

$$X'_1 = \frac{x_1 - \mu_1}{\sigma_1}$$

where X'_1 is the new feature vector after scaling, x_1 is the initial feature vector, μ_1 is the mean value of feature vector and σ_1 is the variance of feature vector.

2.3.3.4. Feature selection

The selection of right features independent of the algorithm plays an important role in the performance of any classifier. An algorithm cannot find good features or create good features by itself. In this regard, we have used three different types of feature sets to compare their performances which are shown in Table 1.

No.	Features Selected
Feature Set 1, F1	Active Power, Reactive Power
Feature Set 1, F1	Voltage, Current, Phase Angle
Feature Set 1, F1	Active Power, Reactive Power, Voltage, Current, Phase Angle

Table 1: Features used for Occupancy Detection

The first feature set, F1 consists of only power measurements and includes real power and reactive power. The second feature set, F2 consists of root mean square 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 affecting the performance of the algorithm as we discussed in the next section.

2.3.3.5. Model Implementation

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 labeled data to learn and recognize the patterns. In our case, electricity consumption data with the ground truth reality acts as training data. As described earlier, 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. We implemented four state of the art classification algorithms for pattern recognition which are shown in Table 2.

No.	Technique	Parameters	Abbreviation
1	K Nearest Neighbor	K = 10	knn
2	Support Vector Machine	Kernel = RBF	SVM-RBF
3	Support Vector Machine	Kernel = Linear	SVM-Lin
4	Support Vector Machine	Kernel = Polynomial	SVM-Poly

Table 2: Classification Algorithms

Support Vector Machines (SVM) is an efficient classification algorithm which is widely used for pattern recognition because of its 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 SVM. 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 Neighbor (KNN) is one of the simplest learning techniques used for classification. It is a non-parametric algorithm which means it does not make any prior assumptions on the data set. It works on the principle of finding predefined number of labeled samples nearest to the new point, and predicts the class with the highest votes. The advantage of KNN lies in simple implementation and reduced complexity. Despite its simplicity, it works quite well 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.

2.3.4. Proposed Method using Statistical Approach

In the previous section, we discussed a supervised machine learning method (classification analysis). Although, the performance of such methods are quite good but labelling data poses an extra task and it is not always possible to have an access to labelled data. We have explored the possibility of using Hidden Markov Model (HMM) for inferring user state from electricity consumption data without the availability of annotated output labels. HMM has been widely used for temporal pattern recognition in speech applications. Different components of HMM which fit into our problem are below.

- **Observations**: Electricity Consumption Data, X
- **Hidden States**: Present, Absent and Standby, Y
- **Solution**: Find the parameters of HMM (λ) using **Expectation Maximization** algorithms on Maximum Likelihood Theory

Block diagram of the proposed method is shown in Figure 5.

Figure 5: Block Diagram of HMM Implementation

Gaussian Mixture Model (GMM) is a probabilistic model which assumes that all the observation points are generated from a mixture of finite number of Gaussian distributions and the parameters of Gaussian distribution are not known. GMM is an example of soft clustering
where assignments of underlying data points to particular Gaussian distribution is done in terms of probability. HMM training assumes that the observations are discrete in nature, and we apply GMM in order to discretize continuous observations of electricity data.

2.3.4.1. Expectation Maximization

Expectation maximization (EM) algorithms are iterative optimization algorithms which are used extensively to learn probability distributions from incomplete data. In our scenario, we are interested to find output labels which is the missing part of data. The parameters λ of a statistical model for the given data X can be found using Maximum likelihood Estimation (MLE). More specifically, it finds the optimized parameters λ^* such that the likelihood of observing the training data for the given model is maximized;

$$\lambda^* = \arg max_\lambda \Pr(X = x; \lambda)$$

But we are interested in the case where we have partial data available or finding a statistical model (HMM) which describes the relationship between two sets of variables (X`s and Y`s), and there is data available from just one of them. Formally, the problem can be defined as finding the parameters of a HMM which can be represented as $\Pr(X, Y; \lambda)$ which optimizes not only the observable variable X but hidden variable Y as well.

The optimization criteria is defined as the maximization of the marginal likelihood, summing over all settings of the hidden variable Y, which takes on values from the designated set of hidden variable. It is assumed that the variables $X\ and\ Y$ can only take discrete values from the known set. It limits the use of EM algorithms only for discrete variables ($X\ and\ Y$). This assumption is valid for many text processing applications but it limits the use of EM for many IoT applications. In such scenarios, one possible solution is to apply Gaussian Mixture Model (GMM) on the observations to make the observation set discrete in nature. The details about GMM can be found in [Shental, et al. 2004]. Assuming the inputs are discrete in nature, the rest of the steps are stated below;

$$\Pr(X = x) = \sum_{y \epsilon Y} \Pr(X = x, Y = y; \lambda)$$

For a vector of training observations $x = (x_1, x_2, \dots\ x_l)$, if we assume the samples are identically distributed;

$$\Pr(x; \lambda) = \prod_{j=1}^{|x|} \sum_{y \epsilon Y} \Pr(X = x, Y = y; \lambda)$$

Thus the maximum likelihood estimate of the model parameters λ^* for observation sequence X becomes:

$\lambda^* = \arg max_\lambda \prod_{j=1}^{|x|} \sum_{y \epsilon Y} \Pr(X = x, Y = y; \lambda)$

Expectation maximization (EM) is an iterative optimization algorithm that finds a successive series of parameters $\lambda^{(0)}, \lambda^{(1)}, \dots$ that improves the marginal likelihood of the training data and guarantees that;

$$\prod_{j=1}^{|x|} \sum_{y \epsilon Y} \Pr(X = x_j, Y = y; \lambda^{(i+1)}) \geq \prod_{j=1}^{|x|} \sum_{y \epsilon Y} \Pr(X = x_j, Y = y; \lambda^{(i)})$$

The algorithm starts with some initial set of parameters $\lambda^{(0)}$ and then updates them using two steps: expectation (E-step), which computes the posterior distribution over the latent variables given the observable data X and a set of parameters $\lambda^{(i)}$ and maximization (M-step), which computes new parameters $\lambda^{(i+1)}$ maximizing the expected log likelihood of the joint distribution with respect to the distribution computed in the E-step. The algorithm terminates when the likelihood remains unchanged. More details about the E-step and M-step can be find below

E-step.

In expectation step, it computes the posterior probability of every possible hidden variable assignments $y \epsilon Y$ for each $x \epsilon X$ and the current values of the model parameters weighted by the prior probabilities of $x \epsilon X$.

This probability can be represented as $(X = x, Y = y; \lambda^{(i)})$. It defines a joint probability distribution over $X \times \mathrm{Y}$ and $\sum_{(x,y)\epsilon X\times \mathrm{Y}} q(x,y) = 1$.

$$q\left(x,y;\lambda^{(i)}\right) = f(x|X).\Pr\left(Y = y\middle|X = x;\lambda^{(i)}\right) = f(x|X).\frac{\Pr(x,y;\lambda^{(i)})}{\sum_{y'}\Pr(x,y';\lambda^{(i)})}$$

M-step.

In the maximization step, it computes the new parameters which will maximize the expected log of the joint probability distribution under the q—distribution which was computed in the E-step.

$$\lambda^{(i+1)} = \arg max_{\theta'} \mathrm{E}_{q\left(X=x,Y=y;\lambda^{(i)}\right)} logPr(X = x, Y = y; \theta')$$

$$= \arg max_{\theta'} \sum_{(x,y)\epsilon X\times Y} q(X = x, Y = y; \lambda^{(i)}).logPr(X = x, Y = y; \theta')$$

The model with parameters $\lambda^{(i+1)}$ will have equal or greater marginal likelihood on the training data as compared to model with parameters $\lambda^{(i+1)}$ and the proof can be found in [Frederick Jelinek. 1997].

2.4 Connections with other Components

The prediction block will be connected with the following three main components in order to create prediction models and later update them iteratively.

1. Cloud Storage
2. Data bus
3. Semantic store

It is connected with Cloud Storage in order to build prediction models using historical data. In future, we intend to keep updating the model parameters with real time measurements. For this purpose, it is connected with the Data Bus in order to access real-time data published by

Virtual Entities. The interface with the Data Bus can also be used by other Virtual Entities which are interested in future values of other Virtual Entities by passing queries to the Prediction block. Overall information flow diagram is below.

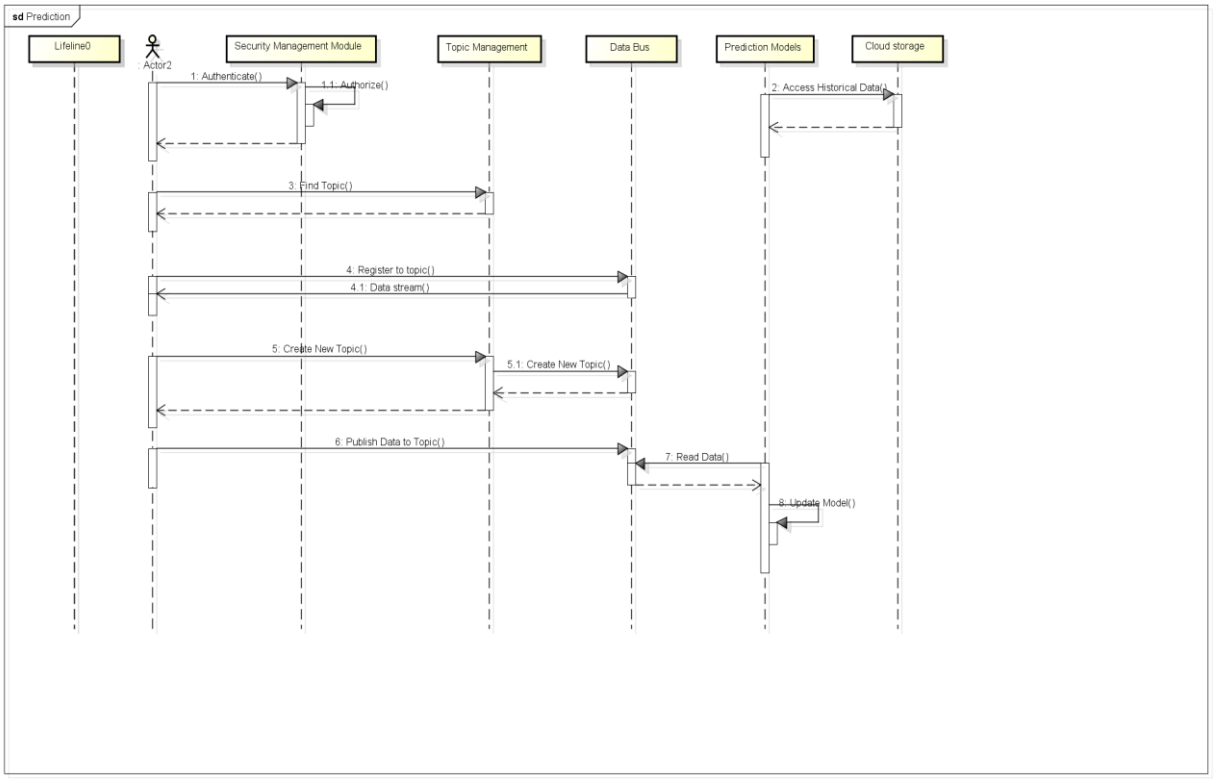

Figure 6: Information flow involving Prediction block

Once a model is created, either off-line or on-line, it has to be described and persisted in order to be later used by other VEs.

We propose to describe the models semantically so that they can be later retrieved or updated using new datasets. The description of the model should include the description of the attributes used to build the model, the description of the class, the application domain as well as the description of the algorithm and the parameters used in the model building process.

For the model persistence there a different formats which will be used. Whenever possible, the PMML format will be used since it provides a standard, platform independent representation. The advantage of this approach is that models build with one machine learning tool or library can later be used by other tool, or application using a different library which supports the PMML format.

Nevertheless, some models cannot be represented using the PMML format, thus an alternative solution will be to use the serialization of the models. For instance, models built with Java based, open-source machine learning software suites such as Weka of Rapid Miner, can be serialized using standard Java serialization and the serialized object persisted for later use. Whenever these models are needed, their serialized representation will be retrieved from the storage location and deserialized into the application.

Therefore, in order to use already existing models, persisted into the Cloud Storage or into other location, these models will have to be semantically described and the semantic description would have to include a reference to the location of the model.

Developers, VEs or applications will be able to query the semantic store for models based on multiple criteria, depending on their scenario requirements. Once a model meeting those requirements is found it can be retrieved from the URI included into its semantic description and used accordingly.

2.3. Interfaces

The application developer can use the models provided by COSMOS for predicting events. In order to use the existing models, the application developer will have to define the input features and output entity in which application developer is interested in. In order to achieve this, the application developer will use the COSMOS storage services in order to access historical data for the construction off-line of the model. Then when a model is available on-line update of the model can be done under certain circumstances. Once enough data is collected, and therefore a model available, the Prediction component/functionality will be instructed for making a prediction based on the available model. The resulting model will be persisted and semantically annotated in order to make the model retrievable for later use. Using this approach, the semantic description of VEs and IoT services or data bus topics, could also include a reference to a prediction model (if available).

Since Cosmos is intended to be used by different actors and forge cooperation and reuse, prediction models can be built by different parties (for instance in the case of public data) provided that they are semantically described and linked to the data sources. Once stored and annotated, other actors will be able to query the semantic store for prediction models and use them according to their needs.

Client/VE-Prediction Block API:

An API is required to connect the Client or application developer to Prediction block for the following purposes

1. An application developer in order to select the particular prediction model and to define the input and output for the model.
2. A Clinet or a VE will send a request using the API to register its interest in particular topic stating the interested characteristics/services (Occupancy state of a room, Traffic conditions on a road).
3. A Client or VE can also use API to get required prediction value at particular instant. An example can be a client sending a query to prediction block to find the traffic state at particular location.

Storage-Modelling Block Interface:

All the historical data is stored in the form of objects in object storage. Machine Learning models require an access to historical data for training purpose. In this regards, modelling block should be connected to the object storage.

Message Bus- Modelling Block Interface:

The modelling block will be connected to the message bus and subscribed to the interested topics. This interface will be a crucial block for real time updating of models in the Year 2 and Year 3.

2.4. Predictive Analysis and CEP

Predictive Analysis (PA) applies diverse disciplines as discussed above such as probability and statistics, machine learning and artificial intelligence for prediction in dynamic and uncertain environment. It employs a variety of methods and techniques from data mining and statistics that explore current and historical data to make predictions about future event, whereas Complex Event Processing engines are optimized for large volumes of streaming events. It does not store the data, instead queries are stored and they are run on the data streams in real time. Complex Event Processing techniques detect an event after it has occurred. In many applications, prior prediction of an event is more useful then detecting an event when it occurs. And several situations require exact knowledge about the event. In these cases, PA cannot be used due to its uncertainty. Both fields have many overlapping concepts but yet there exist only few solutions where PA concepts have been applied to CEP. We intend to extend the Prediction Models build in this section to use in conjunction with CEP techniques discussed in next section to improve the performance of CEP and to extend current state of the art techniques for prediction of events.

2.5. Conclusion

We have explored supervised machine learning models for extracting high level knowledge from raw IoT data based on pattern recognition techniques and a statistical inference method based on Hidden Markov Model. We have demonstrated the use of both methods for novel applications. Both techniques have their own advantages and disadvantages which we have discussed briefly. In Year 2, we intend to leverage on the Year 1 progress and provide solutions for very large data sets by exploring storlets and integrating machine learning methods with object storage. We aim to explore the possibility of moving pre-processing tasks such as aggregation and resampling to the object storage in order to reduce the amount of data travelled over network. Secondly, we are also exploring incremental machine learning solutions for providing adaptive solutions for real time dynamic IoT scenarios. Finally, we intend to explore the possibility of using machine learning in conjunction with cep for predicting potential future events.

3 Situational knowledge acquisition and analysis

3.1 Decision making in complex environments

A COSMOS platform aims at providing support for the decision making for highly automated applications. Making a decision in complex dynamic environment is a very difficult task. The decision making processes are only as good as their information on a situation.

Situational Awareness (SA) is being aware of what is happening in given context (complex, dynamic environment) in terms of the time and space. Besides various business use cases, SA also helps to identify potential critical threats such as threats to health and safety. A typical SA use cases for COSMOS are described in chapter 3.5.

There are many situations especially in IoT world where it is critical to make a correct decision based on proper evaluation of the current situation. Considering the fact that many different devices and sensors are deployed in remote places, this often means that enormous number of varying parameters and/or time critical conditions have to be considered. In these situations it is common that the situation evaluation is supported by the Complex Event Processor attached to multiple sources of various events carrying important information.

COSMOS aims at improving situational awareness beyond the current state of the art by enhancing CEP based situational assessment processes with adaptive feedback loop at runtime. This approach opens new possibilities for more precise and reliable situational awareness.

In addition to CEP techniques, machine learning techniques also play their specific role in contribution to situation awareness process.

Situational awareness is important to COSMOS components, COSMOS enabled applications as well as Virtual Entities need to be aware of their surroundings, opportunities and the potential hazards they face.

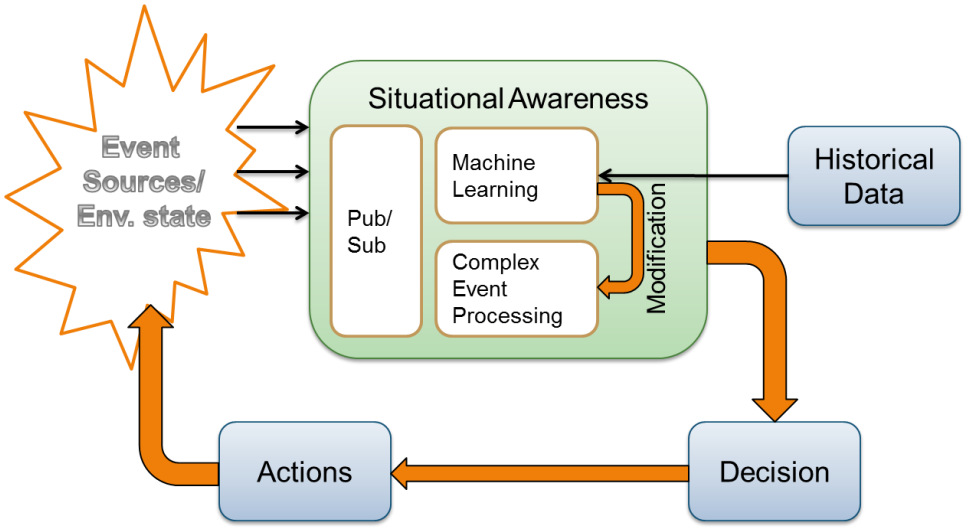

Figure 7: Situational Awareness

A high level functional composition of situational awareness is illustrated on Figure 7. The reason that the machine learning block modifies CEP situation assessment at run-time is that it enables correction and fine-tuning of initial parameters based on analysis of historical data.

3.2 Context Modelling and Reasoning Techniques

A context is useful for representing and reasoning about a pre-defined/restricted state space within which a problem can be solved, and use it as the basis for information analysis and filtering.

In order to help ensuring that situational awareness acquisition subsystem meets expected application requirements, it will be possible to model information about behavior of virtual entities as well as information about surrounding environment. Using models (see Figure 8 below), it is also possible to describe patterns and relations between entities that are used through the model description.

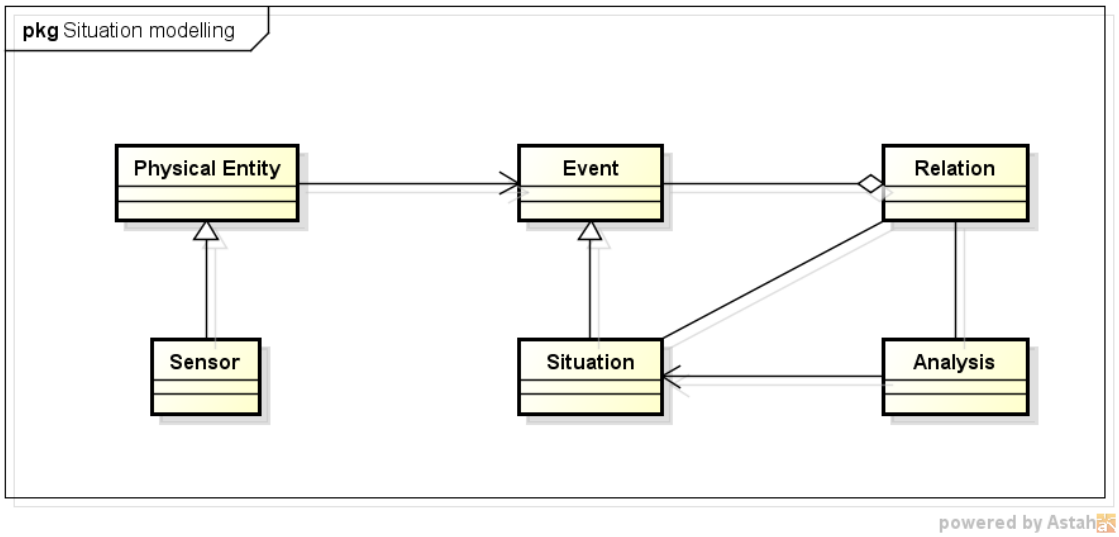

Figure 8: Context Modelling

The key elements in context modelling are entities, events and analysis. A Physical Entity represents an object in a situation which provides partial information about the current environment. A typical example can be a physical device i.e. sensor providing information about temperature, position, CO_2 production, velocity, energy consumption etc… An event is a data structure encapsulating all the relevant information about a particular situation. An event creation triggers the system to evaluate and analyze situations based on the relation between different events. This analysis may eventually lead to detection of more complex events and situations that can be utilized by decision processes.

3.2.1. Context Definition

For the year one, COSMOS will focus on defining situations, relations and analysis using the dedicated Domain specific language called “Dolce [8]” which is especially suitable for CEP technology. Therefore, a processing model provided by COSMOS is continuous. For next years, an extended version of language which will also support machine learning technology will be introduced.

The Dolce [8] language aims at providing simple means to define events, event relations and event analysis in human readable form.

A typical event may be defined using following lexical grammar. The character @ indicates the use of custom keywords as identifiers, which is useful when matching events with defined rules.

Event Definition	**Rule Definition**

event @Event
{
use /*properties definition*/
{
[type] @Property
}
accept { @Property == [value] };
}
	complex @Situation
	{
	payload {[params]}
	detect @Event
	where [aggregate](@Event)
	in [window]
	}

3.2.2. Source Filtering

Business applications are only interested to only acquire a high level of intelligence from the available data with effective reasoning. Therefore, an effective analysis is in need to provide derived information through filtering and other techniques.

The filtering is one of most basic operations on event streams. Filtering evaluates a predefined logical or arithmetic condition based on event name or properties and/or constant values.

Supported filtering conditions consist of “equals”, “greater/less than”, etc., and logical conditions like “and”, “or”, “not” etc.

3.2.3. In-Memory Caching

In order to be able to perform analysis on subset of events within an event stream, received events have to be temporary cached in the memory. The actual in-memory caching is realized through moving windows technique. Windows with different properties produce different results and have radically different performance behaviors.

Cosmos will support following mechanisms for event selection:

3.2.3.1. Time Windows

A Time Windows are specified by time intervals. This is useful in many different situations where particular events or aggregated event values from same or possibly many different event streams fall within or exceed specified time period.

There are two different approaches for time window definition. The selection of proper approach depends on particular situation context.

- **Fixed Time Window** – buffers events for analysis every specified fixed time interval;

- **Sliding Time Window** – buffers events for analysis for specified time interval into the past based on the system time. For example, sliding window can be used to collect all readings during last hour.

3.2.3.2. Tuple Windows

A Tuple Windows are used to collect events from available sources based on number of occurrences. A typical example of tuple window is the collection of last twenty sensor readings for further evaluation.

The tuple window has a fixed size. Therefore its endpoints move together and events continuously expire with new events entering the tuple window.

New event carrying derived information is produced only when window closes i.e. reaches its defined capacity and when a new event arrives.

3.2.4. Aggregation

The aggregation facility continuously calculates various metrics across time and tuple windows and is used for assessment of trends in situations. Aggregations perform different computation over subset of data.

The supported aggregation operations over windows are as follows:

- Avg – average over a numeric event payload;
- Count – Count of events;
- Sum – Summation over a numeric event payload;
- Diff – Difference between two subsequent events;
- Min/Max – Minimum/Maximum over a numeric event payload.

3.2.5. Correlation

The event correlation functionality enables identification of the situation where different conditions occur or do not occur in a specific order. Therefore, the detection of situation depends on the occurrence of a series of events rather than on the occurrence of a single event. The events from multiple streams are correlated over period of time that may be seconds, hours or days.

3.2.6. Event Period

By default, events are transient. This means that events are valid for an infinite short time period. For assessment of some real world situations, it is useful to associate some events with fixed time period to define for how long the situation associated with event is valid.

The supported expressions of event period are as follows:

- Lasts – event is valid for specified period of time;
- Until - event is valid until specified event arrives.

3.2.7. Joins

The purpose of event joining is to create a new stream of events from several other streams based on some conditional matching.

A join between two data streams necessarily involves at least one window. To perform a join, it is usually necessary to wait for events on the corresponding stream or to perform aggregation on selected events.

3.3 Situational Assessment

COSMOS platform will continually evaluate given contextual models according to information received from the publish/subscribe data bus and notify listeners about identified situations.

A general functionality provided by SA functional block is:

- **Continuous assessment of actual situation at real-time –** detection of emergency situations, sudden or unexpected situations, failures and other anomalies, disasters, etc…
- **State and Behavioral situation assessment –** assessment of local situation related to Virtual Entities.
- **Environmental situation assessment –** assessment of environmental conditions in near vicinity of Virtual Entities.
- **Overall situation assessment -** assessment of group or overall situation based on all available information.

3.3.1. Protection Against Over-Aggressive Contextual Models

The Over-Aggressive Contextual Models are those that cannot possibly be evaluated nor evaluated using given resources.

An example of Over-Aggressive model is aggregation performed on all events received during the last day while rate of events is very high compared to available physical resources needed for caching.

There are two types of protection:

- Static analysis of contextual model – analysis is performed before actually evaluating models. This analysis performs checks such as required timing, storage and throughput checking.
- Runtime detection – analysis performed during evaluation phase. Contains several mechanisms for monitoring and ensuring that models are evaluated as expected using given resources.

3.3.2. Runtime Model Updating

The COSMOS project will provide enhanced situational awareness by combining predictive and/or cognitive analysis with runtime situation analysis (CEP). The main idea to improve perception of cosmos environment is to tweak runtime analysis by analysis of historical or predicted situations.

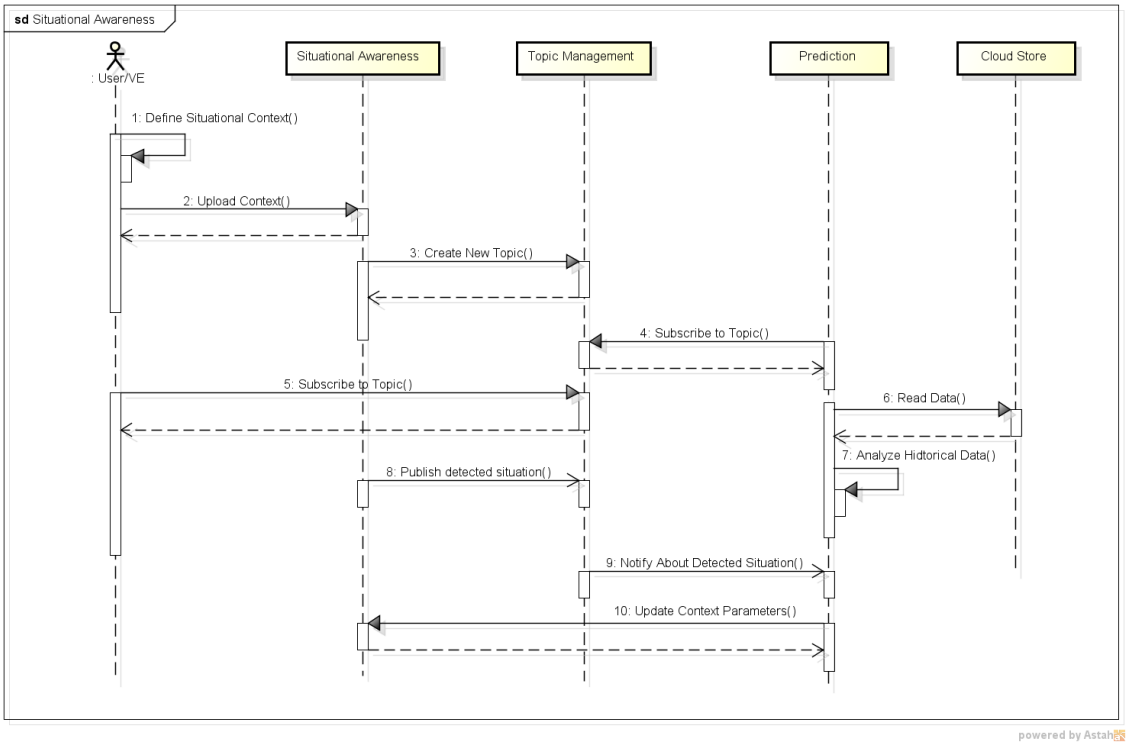

Figure 9: Runtime Model Updating Strategy

As depicted in Figure 9, a predictive analysis can be used in order to update or fine tune initial definition of situation analysis by analyzing historical data related to the same or similar situations.

3.4 Context Propagation

Depending on configuration, each type of detected situation can be published as an individual topic within semantic topic management functional block. Listeners can subscribe to particular topics and receive information about actual situation at real-time (as soon as the engine evaluates event for that topic), according to the contextual model analysis.

3.5 Use cases for Situational assessment

3.5.1. Madrid EMT Use Case

Existing Madrid EMT infrastructure provides possibility to build contextual models based on data streams such as bus position, velocity, schedule, CO_2 emission, as well as environmental information such as traffic light status, temperature, local traffic congestion etc. Typical examples of detected situation are: traffic jam, bus failures, traffic accidents, bus delays, etc.

3.5.1.1. *Buses in queue detection scenario*

In this scenario, the situational awareness feature will detect bus queues at Bus stops. This situational knowledge can be used by Cosmos for enhanced cruise control (for example eco-saving) as there will be no need to increase speed and thus to consume gas when there is a queue at a bus stop.

One feasible approach to detect bus queue is based on utilization of GPS position provided by busses.

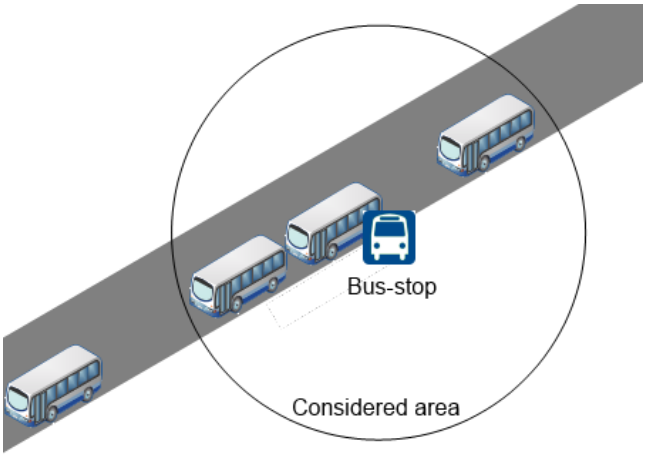

Figure 10: Queue detection

Definition and modeling of busses in queue situation:

As depicted in the Figure 10 above, we can model given situation by counting bus occurrences within specified area surrounding bus stop and time. Let's assume that for a particular bus stop, a bus queue can be identified when at least five different busses are present within a one hundred meters radius for more than one minute.

First step consists of definition of input events and corresponding attributes. An event representing bus within specified area can be defined using following Dolce [8] event construct:

Bus event definition

```
event BusNearStop
{
      use
      {
            int BusIdentity,
            pos Location,
            float Acceleration
      }
      accept( diff(Location, BUS_STOP_LOC) < 100 //meter )
}
```

A purpose of an accept statement is to select only those events for which a bus location is within specified proximity distance.

Finally, to define actual buses in queue situation, we need to specify counting of buses entering a bus stop area within specified time. This can be achieved by utilizing aggregate function "count" and a time window as shown in the following table:

Busses in Queue situation definition

```
complex BussesInQueue
{
    detect BusNearStop
    where count(BusNearStop) > 5
    in[1 minute]
}
```

3.5.2. Energy distribution use case

Heat and electricity consumption sensors as well as different environmental sensors are main sources of information that can be used for the situational assessment. For example, it is possible to build a contextual model that can be evaluated to detect at run-time when energy consumption of building can be reduced based on different continuous sensor readings within building itself and outside such as temperature, heating and lighting.

3.5. Automatic generation of Rules for CEP using ML

CEP engines require rules or patterns to detect an event from data streams which have to be given manually by the administrators of the system. Based on this, there is an assumption that administrators have the required preliminary knowledge which sometimes is not available or not so precise. So manual setting of rules and patterns limits the use of CEP only for expert's domain and poses a weak point. And even though with prior expertise and knowledge, experts are prone to make errors in choosing optimized parameters for dynamic systems. In real time dynamic scenarios, parameters of a system may change and performance of CEP may deteriorate in such dynamic scenarios.

We intend to apply predictive analytics principles to improve decision making and performance of existing CEP solutions by exploring adaptive and automatic solutions. Both fields have many overlapping concepts but yet there exist only few solutions where PA concepts have been applied to CEP. One possible research area to explore is automatic generation of rules for CEP in order to cope with above mentioned limitations. CEP language uses number of operations for describing pattern of events. In [Margarra, et al. 2014], authors applied the Machine Learning techniques for generating automatically the following five most commonly used operations.

1. determine the relevant time frame to consider, i.e. the window size;
2. identify the relevant event types and the relevant attributes;

3. identify the predicates that select, among the event types identified above, only those notifications that are really relevant, i.e., determine the predicates for the selection operator;
4. determine if and how relevant events are ordered within the time window, i.e., the sequences;
5. identify which event should not appear for the composite event to happen; the negated event notification

The authors applied IGR principle and implemented each operation in different module. This is just one example where Machine Learning can be used in conjunction with CEP. We intend to explore more predictive analysis methods in conjunction with CEP to provide more adaptive, automatic and optimized solutions that can lead to more accurate, faster and consistent performance.

3.6. Interfaces

The situational awareness component will provide information in asynchronous manner. Therefore, a clients need to subscribe to detected situations based on their custom preference.

Virtual Entities may perform CRUD operations on context model at runtime. These operations are available via specialized REST API published by the situational awareness functional block.

A several different interfaces are mandatory for SA assessment:

1. A user needs to define different situational contexts so that machine can automatically evaluate and detect relevant situations. A context definition may come from VEs, Cosmos enabled applications or administrators. An enhanced declarative Dolce [8] language as well as new corresponding knowledge base can be provided.
2. New semantic API for registration of event sources over semantic topic management as well as direct P2P connection will be provided.
3. User may opt to also specify the customized source of detected events and register to high level events he is interested in.

3.7. Conclusion

Situational knowledge must change and update in timely manner with new input to the system. During the year 1, different event processing techniques and low level situation definition solution have been explored. These techniques provide tracking, monitoring, sensing and responding basis for the situational awareness.

Rule-based event processing techniques presented in this chapter are mandatory as they provide a core foundation for the situation awareness but are not enough for obtaining SA in dynamically changing conditions. Situation detection rules need to be continuously updated to handle dynamic environments. Therefore machine learning techniques combined with CEP will be explored in following years.

In addition to machine learning, in Year 2 and 3, semantic event processing will be explored. The goal is to achieve better understanding and automated detection of situations and relationships between incoming events by machine i.e. (semantic CEP extensions). This will further enable real-time declarative processing of events, leveraged reaction to situations as well as enhanced declarative rule definition.

4 Experience and Experience Sharing

4.1 Experience in COSMOS

In the context of COSMOS, different meanings of experience can be defined, arising from the correlated phases of MAPE – K loop [10] approach, which is adopted for the implementation of the project regarding VEs' management:

- **Models building:** this type of experience is related to the **Analysis (A)** component, whose main functionalities are information process and events detection. Such kind of **models** are described in chapter 2 (prediction models), which could be built inside a VE in Year 2 and 3.
- **Cases:** this type of experience corresponds to the **Planning (P)** component, which is used in order to select the actions that need to be applied when a system reaches an undesired state. More precisely, a **case** can be considered as a combination between a problem and its solution, whereas a problem consists of one or more events. In other words, it is a kind of rule for an actuation plan, which is triggered when specific events are identified:
    - Case = problem (event1, event2, event…) + solution

For Year 1, **cases** are examined, whereas **models** are going to be analyzed in Year 2 and 3.

The ontological graph that corresponds to the section of experience contains the class of **"Experience"** which is further divided into the subclasses of **"Models"** and **"Cases".** The **"Cases"** class is also further divided into the subclasses of **"Problems"** and **"Solutions"**.

A COSMOS specific ontology which will be created in order to facilitate the description of the key COSMOS building blocks. It will also include the necessary structures for describing the experience. This will also include the means to describe the cases.

The following image demonstrates that the instances (individuals) of the **"Virtual Entity"** class are connected to the inferred instances of the **"Experience"** class through the abstract property **"Has Experience"**. Specifically, in the COSMOS Ontology, described in subchapter 4.6 of D5.1.1, **Experience (case)** is linked with a **Virtual Entity** through the object-type property "**isIncludedBy**".

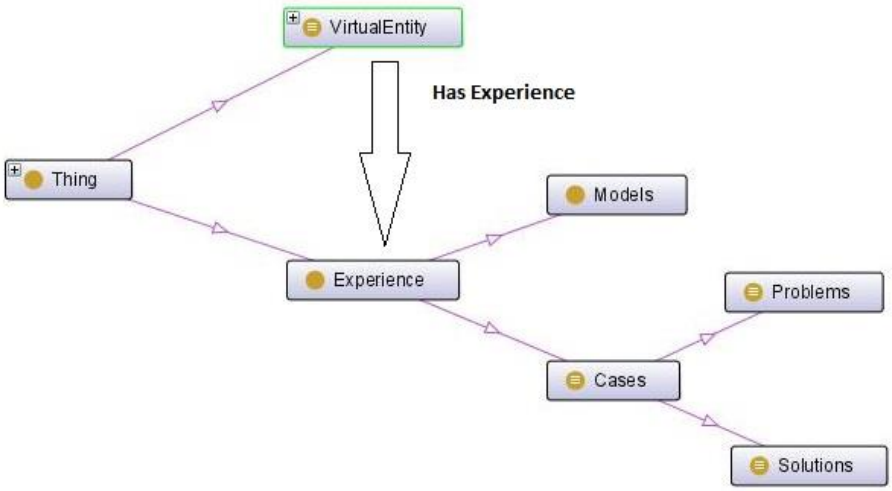

Figure 11: Experience in COSMOS

For the semantic description, an open-source triple store will be used such as OpenRDF Sesame [11] or Apache Jena [12] but other solutions will also be considered.

SPARQL [13] is a syntactically SQL-like language that can be used to express queries across diverse data sources, whether the data is stored natively as RDF or viewed as RDF via middle-ware. SPARQL contains capabilities for querying required and optional graph patterns along with their conjunctions and disjunctions. It also supports extensible value testing and constraining queries by source RDF graph. The results of SPARQL queries can be results sets or RDF graphs.

4.2 Experience Sharing

The main goal of this functional component is to enable VEs to act in a more autonomous way, by sharing their experiences which are related to the specific events that are detected. Experience sharing, as functionality, is implemented in three steps, as it is shown below:

- **Storing Experience**
  Through the use of the semantic store API, VEs can modify the contents of their semantic description. Regarding experiences (cases), it is very important that they can establish a persistent storage of their personal case base, which contains all their experience of the type cases, which is described in subchapter 4.1. This happens by creating new instances of problems and/or solutions or by adding connections to already existing ones.
  VEs are exposed as web services and are semantically described into the semantic store of Cosmos. The semantic description of the VE, contains not only the description of the underlying IoT-services, of the VE capabilities and constraints but includes also references to any prediction models or experiences which it uses or has created.

- **Finding Experience**
  The next important step in the implementation of experience sharing is the actual access of the semantic storage in order to find and return a certain solution that satisfies a problem. As described above, such forays into the semantic store are made through the use of SPARQL and the API's Query Functions.
  The API of the semantic store, dedicated to the experience retrieval, will include specialized methods providing as input parameters the experience search criteria. Experience retrieval could also be performed using a generic query API where users would pass their own query string.
  While the former approach has the advantage of hiding the querying mechanism and providing a uniform access to the semantic store, the latter would provide much more flexibility, but would require user experience with SPARQL.
  In any case, a VE could easily create a query string which would also be fully configurable based on information stored locally in VE variables. Therefore, any access of the VE in the semantic store can be accomplished and the actual experience sharing is made easier, due to the fact that experience instances can be accessed and most importantly assessed by any VE.
  Jetty server is used to enable VE2VE communication through HTTP, including finding and sharing experience, by creating a servlet for each exposed IoT-service which is accessible by a URL.
- **Choosing Experience**
  Having already described the process by which an experience can be accessed, it is also important to mention that any retrieved information (including experience), can also be assessed, ranked, etc., giving VEs the capability to choose one experience of those offered by other VEs. By defining a variety of social properties such as trust & reputation index, we enable VEs to check whether a solution answers not only their queries but also satisfies some quality criteria. This process is executed by the Planner of the VE based on how many times a VE has shared its cases or how many times its IoT-services have been used.

4.3 Use cases for Experience Sharing

4.3.1. Madrid Use Case

Using the Madrid use case as an example, let us suppose that in a bus VE, the CEP engine detects the start of a fire, thereby issuing a warning for the **problem** (event and subclass of a **case)**, in the topic, which is called “fire in the bus”, in the message bus. The latter is referred in the subchapter 4.6 of the Deliverable 4.1.1. The VE has subscribed to the aforementioned topic and therefore receives the relevant notification. Then the following steps take place:

1. The bus VE looks at its own case base in order to find a **solution**, corresponding to this problem.
2. If there is no such solution, then the bus sends its problem to its friends, requesting their experience (solution). (finding experience)
3. If a friend bus has a similar problem in its case base then it sends back the solution which is related to the problem. (finding experience)

4. If a friend bus sends back a solution, then this interaction is tracked by the Social Monitoring Component (described in subchapter 4.3 of the Deliverable 5.1.1) and therefore the trust & reputation index of the friend bus is being increased.
5. If the bus receives two or more different solutions ("call fire-truck 1", "call fire-truck 2" and so on), then its Planner is activated in order to choose the appropriate one, based on some social characteristics, like trust & reputation index, which are stored in the COSMOS platform. The Planner is analytically described in the subchapter 4.1 of the Deliverable 5.1.1. (choosing experience)
6. If the bus has no friends or if none of its friends has a solution for the problem "fire in the bus", then the bus requests from the Registry & Discovery component (subchapter 4.5 of the Deliverable 5.1.1) to take recommendation for new friends. After that, the flow goes back to the step "2", whereas the recommendation phase is repeated until the VE gets a solution for its problem. (finding experience)
7. The VE stores the solution, in a way that is mentioned in the subchapter 4.2, in order to reuse it if fire happens again, or to share it with its friends. (storing experience)

In Year 1, the social characteristic, which is used from the Planner for "choosing experience", is mainly related to the number of times a VE has shared a case that is stored in its own case base. In year 3 a VE could be able to evaluate the usefulness of the experience that has received. In this case, the trust & reputation index could indicate not only how "popular" the VE is but also how efficient are the solutions that it shares.

4.3.2. London Use Case

In this use case, a building can be considered as a VE and let us assume that it contains in its own local case base, a problem called "overheating". Each building is supplied with temperature sensors which could measure, for example, the indoor temperature of all the rooms and flats, continuously. By processing all these measurements, the CEP engine could detect the event "overheating" that was mentioned above. Similarly with the Madrid use case, the VE should firstly search in its base to find the appropriate solution to the problem and if it is empty, then it could request its friends VEs for help. In this case a solution could be "deactivate the heating system for three hours".

4.4 Communication with other Components

Experience Sharing Component collaborates with the following components of the COSMOS project:

- Planner (WP5): the Planner of the VE passes the problem to its Experience Sharing Component, whereas the Planner of the friend VE passes the solution to its Experience Sharing Component
- Social Monitoring (WP5): when the Experience Sharing Component of the friend VE sends back a solution, a relevant notification is given to its Social Monitoring Component.

All these interactions are shown in the following sequence diagram:

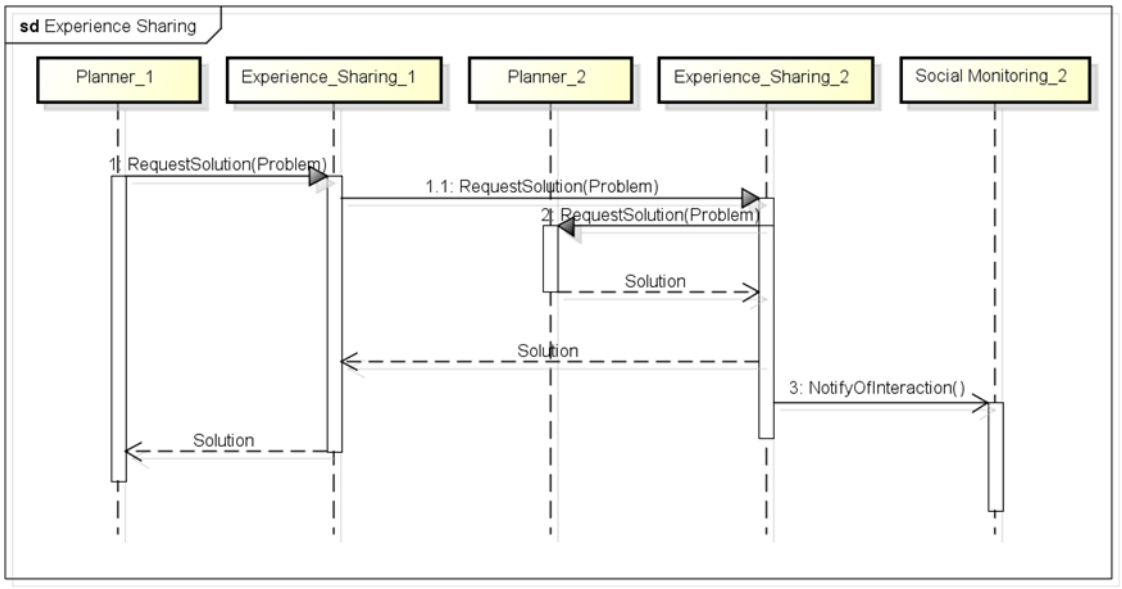

Figure 12: Experience Sharing Sequence Diagram

4.5 Interfaces for Experience Sharing

As it is explained above, the Experience Sharing block is activated when an event (simple or complex) is detected and if the VE has no solution corresponding with this problem.

The VE asks its friends VEs for a solution to the problem, using SPARQL queries over HTTP. However, due to security reasons and because of the fact that the mechanism should be friendly to the VE developer, who may not be familiar with SPARQL, the interface of the Experience Sharing Component should require the following input parameters:

- The problem that occurred (e.g. “fire in the bus” as it is mentioned in the subchapter 4.3.1)
- The IP address of the friend VE that is going to provide its solution
- The port on which the server of the friend VE is running
- The mapping of the servlet (e.g. “request/Experience”)

The Component returns the solution of the friend VE, in case this exists.

4.6 Conclusion

In Year 1, Experience, in the context of COSMOS, is defined as a case which consists of a problem and its corresponding solution. In Year 2 and 3, different meanings of Experience, like models, will be explored. Regarding the way a VE chooses the suitable Experience, from those offered from its friends, a simple Trust & Reputation index, which is related to the cooperativeness of a VE, is adopted. In Year 2 and/or 3 this index can be expanded, taking into account various interactions between VEs, like those that are analytically described in subchapter 4.3 of the Deliverable 5.1.1.

5 Conclusions

As the Work Package name suggests, the aim of our work is to make “things” smart and reliable. In this regard, we have explored solutions based on diverse fields ranging from data mining methods at one end to experience sharing techniques at the other end. We have demonstrated the use of pattern recognition techniques for smart and innovative IoT applications. In our work, we have highlighted the use of CEP to contribute for context aware models.

Over the years, as the technology evolves, the amount of data at our disposal has also increased rapidly. And the availability of such diverse type of data enables us to induce intelligence in the different real world applications. The use case scenarios of London and Madrid provide us vast amount of Data which will be exploited in the COSMOS. Such a large amount of raw data has to be processed, correlated and synthesized in order to extract high level information in real time, in order to help in decision making. In the year one, we focused on historical data sets on a small scale; but in future, we aim to improve our work for very large data sets and provide near real time solutions. We have also discussed how experience of Virtual Entities can be explored in order to make things more autonomous and to take decisions in new situations. Additionally, an initial mechanism of experience sharing is introduced.

Next iteration of this document will explore addition techniques for data analysis and will propose a technology agnostic framework for Situation Awareness, meaning that other techniques will be explored (especially machine learning) in order to populate an abstract context that can be accessed by VE’s. Finally Year 2 version will also come with extended notion of experience and experience sharing in which we aim to give VEs the capability to assess the effectiveness and usefulness of the experience they have reused.

6 References

MARFIA, G., ROCCETTI, M. AND AMOROSO, A. 2013. A new traffic congestion prediction model for advanced traveler information and management systems. *Wireless Communications and Mobile Computing* 13, 266-276. .

SALSBURY, T., MHASKAR, P. AND QIN, S.J. 2013. Predictive control methods to improve energy efficiency and reduce demand in buildings. *Computers & Chemical Engineering* 51, 77-85. .

LIMAYEM, M. AND CHEUNG, C.M. 2008. Understanding information systems continuance: The case of Internet-based learning technologies. *Information & Management* 45, 227-232. .

TAX, D.M. AND DUIN, R.P. 2000. *Feature Scaling in Support Vector Data Descriptions* .

LIN, J., KEOGH, E., LONARDI, S. AND CHIU, B. 2003. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery,* Anonymous ACM, , 2-11.

GANZ, F., BARNAGHI, P. AND CARREZ, F. 2013. Information Abstraction for Heterogeneous Real World Internet Data. .

KONONENKO, I. AND KUKAR, M. 2007. Machine learning and data mining. Elsevier, .

JOLLIFFE, I. 2005. Principal component analysis. Wiley Online Library, .

PROVOST, J. 1999. Naıve-Bayes vs. Rule-Learning in Classification of Email. *University of Texas at Austin* .

WU, X., KUMAR, V., QUINLAN, J.R., GHOSH, J., YANG, Q., MOTODA, H., MCLACHLAN, G.J., NG, A., LIU, B. AND PHILIP, S.Y. 2008. Top 10 algorithms in data mining. *Knowledge and Information Systems* 14, 1-37. .

BEN-HUR, A. AND WESTON, J. 2010. A user's guide to support vector machines. In *Data mining techniques for the life sciences,* Anonymous Springer, , 223-239.

TIAN, D., ZHAO, X. AND SHI, Z. 2012. Support vector machine with mixture of kernels for image classification. In *Intelligent Information Processing VI,* Anonymous Springer, , 68-76.

SHAKHNAROVICH, G., INDYK, P. AND DARRELL, T. 2006. Nearest-neighbor methods in learning and vision: theory and practice. .

GANZ, F., BARNAGHI, P. AND CARREZ, F. Automated Semantic Knowledge Acquisition From Sensor Data. .

KAMIJO, S., MATSUSHITA, Y., IKEUCHI, K. AND SAKAUCHI, M. 2000. Traffic monitoring and accident detection at intersections. *Intelligent Transportation Systems, IEEE Transactions on* 1, 108-118. .

KALLBERG, Y., OPPERMANN, U. AND PERSSON, B. 2010. Classification of the short-chain dehydrogenase/reductase superfamily using hidden Markov models. *FEBS journal* 277, 2375-2386. .

YU, G., HU, J., ZHANG, C., ZHUANG, L. AND SONG, J. 2003. Short-term traffic flow forecasting based on Markov chain model. In *Intelligent Vehicles Symposium, 2003. Proceedings. IEEE,* Anonymous IEEE, , 208-212.

PÉREZ-LOMBARD, L., ORTIZ, J. AND POUT, C. 2008. A review on buildings energy consumption information. *Energy and Buildings* 40, 394-398. .

ARENS, E., FEDERSPIEL, C., WANG, D. AND HUIZENGA, C. 2005. How ambient intelligence will improve habitability and energy efficiency in buildings. In *Ambient intelligence,* Anonymous Springer, , 63-80.

LU, J., SOOKOOR, T., SRINIVASAN, V., GAO, G., HOLBEN, B., STANKOVIC, J., FIELD, E. AND WHITEHOUSE, K. 2010. The smart thermostat: using occupancy sensors to save energy in homes. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems,* Anonymous ACM, , 211-224.

GUO, X., TILLER, D., HENZE, G. AND WATERS, C. 2010. The performance of occupancy-based lighting control systems: A review. *Lighting Research and Technology* 42, 415-431. .

NGUYEN, T.A. AND AIELLO, M. 2013. Energy intelligent buildings based on user activity: A survey. *Energy and Buildings* 56, 244-257. .

MURTAGH, N., NATI, M., HEADLEY, W.R., GATERSLEBEN, B., GLUHAK, A., IMRAN, M.A. AND UZZELL, D. 2013. Individual energy use and feedback in an office setting: A field trial. *Energy Policy* 62, 717-728. .

SHENTAL, N., BAR-HILLEL, A., HERTZ, T. AND WEINSHALL, D. 2004. Computing Gaussian mixture models with EM using equivalence constraints. *Advances in neural information processing systems* 16, 465-472. .

FREDERICK JELINEK. 1997. Statistical methods for speech recognition. MIT press, .

MARGARRA, A., CUGOLA, G. AND TAMBURRELLI, G. 2014. Learning From the Past: Automated Rule Generation for Complex Event Processing . In *DEBS`2014,* Mumbai, India, Anonymous .