COSMOS

Cultivate resilient smart Objects for Sustainable city applicatiOnS

Grant Agreement Nº 609043

D2.2.2 State of the Art Analysis and Requirements Definition (Updated)

WP2: Requirements and Architecture

Version:	1.6
Due Date:	30 November 2014
Delivery Date:	15 February 2015
Nature:	Report
Dissemination Level:	Public
Lead partner:	UNIS
Authors:	All Partners
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
0.1	07/10/2014	Francois Carrez	UNIS	Initial version ready for contributions
0.2	10/11/2014	Adnan Akbar	UNIS	Update data analytics section
0.3	13/11/2014	Orfeas Voutyras	NTUA	New section on Social Analysis and Case-based Reasoning; update of MAKE-K and
0.4	18/11/2014	Paula Ta-Shma	IBM	Update Analytics close to the data store section
0.5	17/11/2014	Leo Pitu	SIEMENS	Update to security part
0.6	26/11/2014	ALL	ALL	Final updates
1.0	28/11/2014	Francois Carrez	UNIS	Compile a version for internal review
1.1	15/12/2014	Panagiotis Bourelos, Achilleas Marinakis	NTUA	Internal Review
1.2	16/12/2014	Bogdan Târnaucă, Leonard Pitu	SIEMENS	Internal review
FINAL	17/12/2014	Francois Carrez	UNIS	Final version to be submitted
RE_SUBMIT v01.3	26/01/2015	Adnan Akbar	UNIS	HHM Update + Double check w.r.t. Review comments
RE_SUBMIT v1.4	30/01/2015	Juan Sancho	ATOS	Situation Awareness update
RE_SUBMIT v1.5	05/02/2015	Juan Sancho	ATOS	Misc. updates
RE_SUBMIT v1.6	12/02/2015	Juan Sancho	ATOS	Final Review

Annexes:

Nº	File Name	Title
1	Cosmos_D2.2.2_Annex 1_Cosmos Requirements_v2.0.xlsx	Cosmos Requirements (Updated)

Table of Contents

Table of Figures

Acronyms

Acronym	Meaning
3DES (TDES)	Triple Data Encryption Algorithm
ACL	Agent Communication Language
AES	Advanced Encryption Standard
AGR	Adaptive Guided Retrieval
AMI	Advanced Metering Infrastructure
AMQP	Advanced Message Queuing Protocol
AMS	Agent Management System
API	Application Programming Interface
ARIMA	Auto Regressive Integrated Moving Average
ARM	Architectural Reference Model
ARMA	Auto Regressive Moving Average
ASM	Agent Security Manager
ATSN	Agent-based Trust Model for Sensor Networks
AWS	Amazon Web Service
CB	Case Base
CBR	Case-based Reasoning
CCF	Consumable Crypto Firewall
CDMI	Cloud Data Management Interface
CEP	Complex Event Processing
CPU	Central Processing Unit
DDM	Dynamic Data Masking
DDoS	Distributed Denial of Service
DES	Data Encryption Standard
DF	Directory Facilitator
DH	Diffie-Hellman
DHT	Distributed Hash Table
DL	Description Logic
DNS	Domain Name System
DoS	Denial of Service
DoW	Description of Work
DSL	Domain Specific Language
DSS	Data Distribution Service
EAL	Evaluation Assurance Levels
ECA	Event Condition Action
ECC	Elliptic Curve Cryptography
EM	Expectation Maximization
EMR	Elastic Map Reduce
EPC	Electronic Product Code
EPCIS	EPC Information Service
ESI	Energy Service Interface
ESP	Event Stream Processing
EU	European Union
FIFO	First-In / First-Out

FIPA	Foundation for Physical Intelligent Agents
FLD	Fuzzy-Logic Device
GPS	Global Positioning System
GVE	Group Virtual Entity
HAN	Home Area Network
HMAC	Key-Hash Message Authentication Code
HMM	Hidden Markov Model
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transfer Protocol Secure
HVAC	Heating, Ventilation and Air Conditioning
ID	IDentifier
IETF	Internet Engineering Task Force
IHD	In Home Display
IoT	Internet of Things
IoT-A	Internet of Things - Architecture
ITS	Intelligent Transportation Systems
JSON	Java-Script Object Notation
LHS	Left Hand Side
M2M	Machine-to-Machine
MaL	Maximum Likelihood
MANET	Mobile Ad-hoc NETwork
MAP	Maximum A Posteriori
MAPE-K	Monitor/Analyze/Plan/Execute - Knowledge
MBR	Model-based Reasoning
MDM	Meter Data Management
ML	Machine Learning
MOF	Meta-Object Facility
MQTT(-S)	Message Queue Telemetry Transport (extension to Sensors)
N3	Notation 3
NILM	Non-Intrusive Load Monitoring
NIST	National Institute of Standards and Technology
OCL	Object Constraint Language
OMG	Object Management Group
OLE	Object Linking and Embedding
ONS	Object Naming Service
OPC	OLE for Process Control
OPC-UA	OPC Unified Architecture
OWL	Web Ontology Language
P2P	Peer-to-Peer
PAA	Piecewise Aggregation Approximation
PCA	Principal Component Analysis
PDF	Probability Density Function
PE	Physical Entity
PIR	Private Information Retrieval
PKI	Public Key Infrastructure
QoS	Quality of Service
RBF	Radial Basis Function

RBR	Rule-based Reasoning
RDD	Resilient Distributed Dataset
RDF	Resource Description Framework
RDF-S	Resource Description Framework Schema
RFID	Radio Frequency IDentification
RFSN	Reputation-based Framework for Sensor Network
RHS	Right Hand Side
RSA	Rivest Shamir Adleman
RuleML	Rule Mark-up Language
S3	Simple Storage Service
SaND	Social Network and Discovery Engine
SAX	Symbolic Aggregation Approximation
SCADA	Supervisory Control and Data Acquisition
SDM	Static Data Masking
SDN	Storage Delivery Network
SGML	Standard Generalized Mark-up Language
SHA	Secure Hash Algorithm
SNIA	Storage Networking Industry Association
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SPARQL	SPARQL Protocol And RDF Query Language (recursive acronym)
SQL	Simple Query Language
SSL	Secure Socket Layer
SWRL	Semantic Web Rule Language
TCFL	Trust Computation w/ Fuzzy Logic
TCP/IP	Transmission Control Protocol / Internet Protocol
T&R	Trust and Reputation
TSL	Transport Layer Security
TTSN	Task-based Trust Framework for Sensor Networks
UDP	User Datagram Protocol
UML	Unified Modeling Language
UNIs	(IoT-A) UNIfied requirements
URI	Uniform Resource Identifier
VE	Virtual Entity
VPN	Virtual Private Network
W3C	World Wide Web Consortium
w.r.t.	with respect to
WSN	Wireless Sensor Network
XML	eXtensible Markup Language
XMPP	eXtensible Message and Presence Protocol

1 Introduction

This second version of the State-of-the-Art Analysis and Requirements Definition is the updated version of D2.2.1 which was released on Month 6 of the project. It will be complemented with a final third version in 2015 (Month 27). Like D2.2.1, its two main objectives are to make an analysis of the literature and current state of the art in a number of domains which the COSMOS project touches and to identify the main requirements to be addressed by COSMOS. Those requirements will be used as incentives when designing the COSMOS Architecture (WP2) and working on the technical challenges identified in the Description of Work (work packages (3, 4, 5 & 6)). The differences between D2.2.1 (initial version) and D2.2.2 (this updated version) are summarized below but, in a nutshell, D2.2.2 provides additional analysis of the State of the Art, focusing on topics which came into consideration after the first round of work within COSMOS.

Section 2 of this document provides some background information about the project, defining COSMOS specific terms compared to the concepts introduced by the IoT-A project (especially the IoT Domain Model) and showing how the COSMOS Domain Model fits within the IoT-A Architectural Reference Model. However this document will not present the whole COSMOS Domain Model (which is part of the Architecture deliverable D2.3.1) but will focus on a sub-set of the COSMOS concepts instead.

Section 3 provides a SotA analysis and explains how COSMOS will position itself w.r.t. the current SotA.

Section 4 introduces the process COSMOS is following as far as requirements collection is concerned. Since we follow the IoT-A methodology, this section is a reminder on how we will proceed within COSMOS, and also explains where the VOLERE template used for requirement collection comes from. Finally this updated version provides, in Appendix 1, a refined and updated list of requirements (provided as an excel sheet compliant to the VOLERE template).

What is new in this version:

- Update and reorganization of Section 3.2 about “Data Analytics” with complete new sub-sections on Machine Learning topics like pre-processing, algorithms for classification, clustering, statistical Data Analysis, Time Series Analysis and Hidden Markov Models;
- Update of the Section 3.2.7 about “Analytics close to the Data Store” with introduction to Spark;
- New Section 3.3.2 on interoperability in industrial Data Acquisition and Process Control;
- Added Section 3.5 on Situational Awareness;
- Updated Section 3.7 with new content on Case-Based Reasoning in particular (which is the basis for experience sharing in WP5 & 6);
- New Section 3.9 about “Social Computational Models and Social Network tools”.

2 Glossary of Terms

In this section we provide a definition of the main terms and concepts used in the COSMOS project. In doing so, we try to align also as much as possible with the generic terms from the IoT field as they are considered by the IoT-A project. Especially the IoT Domain Model of the Reference Model (part of the whole IoT Architecture Reference Model) introduces terms like Service, Resource, Virtual Entity, Devices, Physical Entities etc., which should **not** be reused in the context of COSMOS with a different meaning or interpretation.

This section therefore is made of two main parts which are respectively the Glossary (a table) and a conceptual model that describes how the COSMOS concepts relate to each other (in the spirit of the IoT Domain Model). This model shows the relations occurring between the COSMOS concepts. This section is paramount for ensuring a common understanding across all work packages.

2.1 Glossary of Terms (COSMOS Concepts)

In the COSMOS project we try to stick as much as possible to the IoT concepts introduced by IoT-A, especially those which are described in the IoT Domain Model. The following table gives a definition of the concepts introduced in the list of COSMOS Requirements, and how they relate to IoT-A (when not strictly identical).

It is worth noting that the Architecture document will include a "customized" version of the IoT Domain Model from IoT-A, adapted to the COSMOS terminology. It remains still fully compatible with the IoT-A one.

Concept in COSMOS	Definition	Concept in IoT-A
Object	The entity of interest in an IoT application. Objects can be buses, rooms of a dwelling, flat/house, bus line, bus stops etc. All objects will be represented in the IoT system by VEs or Group VEs. Therefore in COSMOS, VEs (and the GVEs that can contain several VEs) are the central and main concept of COSMOS.	Physical Entity (PE)
Experience	Experience is a different type of knowledge (in the broad sense) which is exchanged between VEs. It can be a piece of knowledge (following an ontology), a model resulting from Machine Learning, contextual information, knowhow etc… As a preliminary approach, experiences in COSMOS are based on Case-based Reasoning, and an experience therefore consists of a Case which is a pair (problem, solution) as explained in deeper detail in Section 3.6.	n/a
Virtual Entity	The counterpart of the object in the "Cyber World".	Virtual Entity

(VE)	VEs are the main concept interacting with the COSMOS platform. It is worth mentioning that VE behavior may follow the MAPE-K structure introduced later on in this document, however this is not compulsory.	
Group VE's	A VE that represents a group of VEs, e.g. a bus line GVE is made of Bus VEs	Virtual Entity (IoT-A allows nesting of VEs)

2.2 Conceptual Model

As already stated in the introduction of this document, the following picture shows only a fragment of the COSMOS Domain Model limited to VEs (individual VEs and Group VEs), IoT Resources, Services (IoT Services and "classic" Services) and shows how they relate to each other. The central part of this model consists of the VEs (either single VEs or Group VEs). VEs are the representation in the "Cyber World" of the Real World Objects. VEs have service logic (see the MAPE-K paragraph to get a glimpse of how a VE "service logic" or behavior can be structured) and a set of attributes that describe the characteristics of a VE (or GVE). A VE may be connected to Sensors and Actuators for getting perception on the one hand and to actuate on the object on the other hand. For example a Bus VE has access to a GPS sensor in order to indicate its current position (as one of its attribute). Group VEs are made of set of VEs; for instance a GVE "Bus line" connects to all Bus VEs. They have their own set of attributes and service logic as a single VE. IoT services are associated with VEs and GVEs in order to expose underlying IoT Resources in a standardized way. Finally IoT Resources represent in this diagram Sensors and Actuators. Full detail about the COSMOS Domain Model will be available in the D.2.3.1 Deliverable.

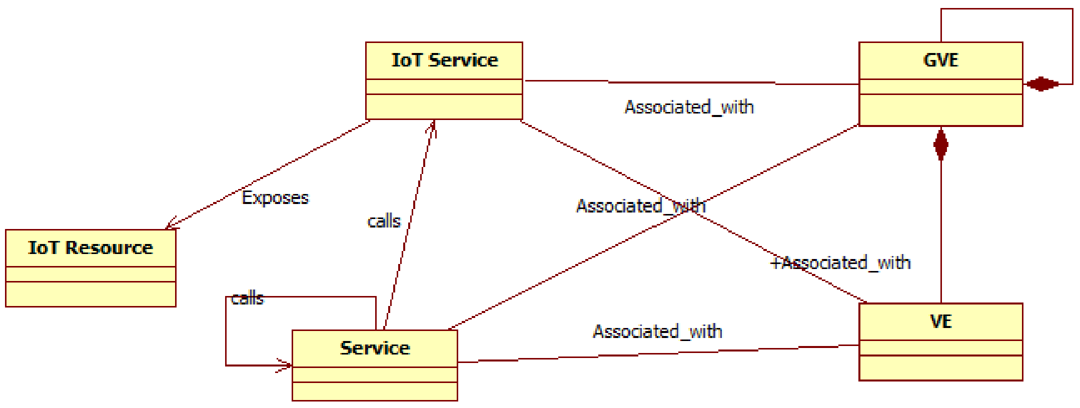

Figure 1: Fragment of the Domain Model.

3 State of the Art Analysis (SotA)

This chapter provides an analysis of the State of the Art for the different technical domains we have either identified initially when writing the DoW or discovered when going further in the precise definition of the WP technical content at the early phase of the project. Those technical domains make the main subsections of this chapter.

3.1 Stream Processing

The two projects Aurora and Borealis set requirements and work on syntax for query languages. The work that was done early is still relevant because of the clear thinking that has gone into language constructs that are complete. StreamIt is from the same lab at MIT and goes on to further work whereby streams are processed as functional flows (push) rather than query engines (pull) like Aurora and Borealis. The StreamIt language specification is formal and relatively mature.

More recently there has been additions to the popular map reduce engine Hadoop with a project name Storm that is used by Twitter; likewise Microsoft, Amazon and other large cloud platform services.

Specialist commercial products include IBM with SPADE and Infostream and TIBCO Business Events, however these are usually supporting functionality for the larger message bus products that are provided by these companies. Architecturally these are message bus implementations with stream handlers. It may be advantageous to adapt to these platforms for commercial potential.

In some ways event driven web services are beginning to act like stream processors in that processing rules, logic and algorithms are embedded into web services. For the purposes of COSMOS, we may be able to get performance through event driven web architectures, like node.js to act as stream processors. A specific implementation of this type of architecture is Node Red, an open source IoT routing and processing platform.

For the purposes of COSMOS, *Complex Event Processing* (CEP) is treated as a special case and not considered stream processing.

COSMOS can benefit the most from:

1. the definitions of stream processing elements and reference architectures in order for how stream processing support and protocols can be constructed within COSMOS;
2. design of stream processing elements for conditioning data;
3. use of stream processing techniques for statistical analysis, time aggregations and combining event data in time;
4. code reuse and open source libraries.

Advancement on State of the Art may be in the areas of integration into security, object stores and complex event processing as well as some demonstration service implementations using stream processing techniques.

3.1.1. Data in IoT

Internet of things is a vast field and cannot be confined to any particular field. It includes but not limited to *Wireless Sensor Networks* (WSN), *Radio Frequency IDentifiers*-based (RFID) Systems and *Machine-to-Machine* (M2M) System. Data is the main element of interest in such systems. It will be helpful to go through few major characteristics of data in such systems before getting into details of different processing engines which are commonly used to process data.

As there are different types of sensors working in wireless sensor networks, the type of data and structure of data transmitted by each sensor will be different. Micro computing devices used in M2M systems are also diverse. It is impossible for data structures to use the uniform model and hence heterogeneity of data is one of the main characteristics of IoT Data.

Mostly it is a dynamic network formed by many objects wirelessly connected to each other. RFID System of Supermarket and monitoring applications are few examples involving hundreds of Giga Bytes of data / day. In real time monitoring applications, unlimited amount of data comes in streams with high speed contributing to largeness of data.

The state of things to be sensed may be changing rapidly. Feedback or response time of the system reflects the reliability and availability of a system. Hence, the IoT systems must respond in a timely manner.

In IoT, reliability of the information provided is one of the major requirements. Reliability in internet of things is a broader term and it covers different aspects of the system which includes but not limited to capacity of a system to withstand against the environmental changes, Long term usability of the system, dealing with security problems, accurate prediction in case of uncertain information and overall system reliability. Basic software used such as operating system, databases and middleware must able to ignore data`s heterogeneity and successfully transmit, filter and integrate them.

3.1.2. Aurora

Aurora Data Stream Management is aimed to provide a single data stream processing framework for Real-time applications, archival applications and spanning applications. Aurora implements Dynamic continuous query optimization as well as ad hoc query optimization to provide certain QoS, semantic load shedding in order to overcome transient spikes in incoming data rates, novel hybrid data storage organizations for efficient implementation of both push and pull based data processing and real time scheduling to maintain its integrity in dynamic environments [Carney, et al. 2002]. Aurora uses primitive operators like slide, tumble, map and join for data stream processing.

Aurora consist of the following modules

- ***Storage Manager:*** It takes the inputs and stores them into proper queues. Its tasks include maintain the box queues and managing the buffer;
- ***Scheduler:*** Scheduler is responsible for selecting the particular box for execution and determining which type of processing is required. After completing the execution, it ascertain the next processing step iteratively;
- ***Box processor:*** It forwards the output tuple after executing the appropriate operation;

- **QoS monitor*:*** It monitors the system performance continuously. If it detects over load situation and poor system performance, it activates the load shedder;
- ***Load shedder:*** It sheds the load of the system until performance reaches to acceptable level;
- ***Catalog:*** All the information regarding network topology, inputs, outputs, QoS information, average box processing costs and selectivity etc. is present in Catalog.

In Aurora stream processing engine, QoS is based on the following three metrics: 1) Response Time 2) Tuple Drops 3) Value produced. Aurora has real-time scheduler in order to optimize QoS and reduce the end-to-end tuple costs at the same time.

3.1.3. Borealis

In Borealis [Tatbul, et al. 2008], Stream Processing functionality is based on Aurora but it extends the concept of Aurora to implement it in distributed manner. Distributed processing, dynamic resource management, query optimization and high availability mechanisms are dominant features of Borealis. It has the capability to modify data and query attributes at run time while acting in a distributed manner. Distributed Processing in stream engines provide following two additional benefits:

1. **Incremental scalability**: it provides the system capability to deal with increasing load as new computational resources can easily be added into a system;

2. **High availability**: In case of failures, multiple processing nodes provide fast recovery while monitoring the system health all the time.

Borealis includes the following modules:

- **Stream Processing Engine:** It provides the basic of real time stream processing functionality with the help of its rich set of stream-oriented operators;
- **Coordinator:** A coordinator is responsible for managing the network of Borealis Stream engines across different nodes, distribute query processing across them and maintain the integrity of overall system in dynamic environment;
- **Load Manager**: It is responsible for monitoring run-time load and moving operators across machines dynamically to improve performance;
- **Load Shedder:** Its functionality is same as was in Aurora. It detects CPU Overload and eliminate it by dropping selected tuples;
- **Fault Tolerance:** It runs redundant query replicas to deal with various failure modes and achieve high availability;
- **Revision processing mechanism:** It is responsible for processing of corrections of erroneous input tuples by generating corrections of previously output query results.

3.1.4. TelegraphCQ

Data is main element of interest in emerging Networked environments. With the advent of Internet of Things and sensors being employed on moving objects, data cannot be assumed to reside statically in known locations. In contrast, data in these new applications is always moving and changing. In such scenarios, data management is viewed as dataflow processing which should monitor and react in real time to accommodate the unpredictability in the nature of data. The aim of Telegraph project was to develop an adaptive Dataflow Architecture for data intense networked applications. The main focus was to meet the challenges posed by large number of real time continuous queries in huge volume of data streams which are subject to highly dynamic environment [Chandrasekaran, et al. 2003]. Its novel architecture to focus on extreme adaptability makes it different from other data stream engines. It does not rely on traditional query plan but instead it uses adaptive routing modules which are able to "re-optimize" routing paths during the query on continual basis. The two basic techniques used in TelegraphCQ for adaptive data flow are *Eddies* and *Rivers* (see [Chandrasekaran, et al. 2003]:

- ***Eddies*** continuously observes the operators consumption and production data rate and reshape dataflow graphs accordingly through operators to maximize the performance;
- ***Rivers*** adaptively routes data to different machines depending on their states in order to balance the workload.

TelegraphCQ use different modules such as file reader, Sensor Proxy and P2P Proxy for Ingress and caching purposes.

According to [Chandrasekaran, et al. 2003], the core and stand out features of the TelegraphCQ can be summarized as:

1. Scheduling and Resource Management for groups of queries;
2. Support for out-of-core data;
3. Variable Adaptivity;
4. Dynamic QoS support;
5. Parallel cluster-based processing and distribution.

3.1.5. AnduIN

AnduIN is an easy to use stream engine to analyze streaming data on the fly (without storing data to a hard disk). User can describe tasks by a simple SQL-like interface. AnduIN autonomously achieve the specific goal defined by a user in most efficient manner using different optimization techniques. A simple static, rule-based optimizer prepares queries for their initial execution [Klan, et al. 2009]. Optionally, a cost-based optimizer, leveraging statistical data from previous query executions to determine an optimal query execution plan can be used. Due to the dynamic and potential infinite character of data streams, the statistics of running queries are continuously observed and updated by AnduIN. If the running execution plan becomes suboptimal, the adaptive optimizer replaces it by a better solution without any loss of data [Klan, et al. 2009].

AnduIN provides the following operators:

1. **AnduI**N provides one-pass (non-blocking) implementations for most of the standard database operators such as projection, filter, aggregation, and join. Additionally, AnduIN provides operators for the following tasks;
2. ***Data Stream Analytics:*** the system offers solutions for typical data mining problems like clustering or frequent pattern mining that operate on data streams. To identify missing or outliers, AnduIN implement operators for outlier, burst and missing value detection [Klan, et al. 2011];
3. ***Spatial Operators:*** Modern mobile devices like cell phones or wireless sensor nodes deliver location-dependent information (e.g. GPS-based). That is, data originating from such devices often contains spatial information. AnduIN allow analysing these data by providing spatial operators such as inside, nearest neighbor, within distance [Klan, et al. 2011];
4. ***Complex event processing:*** A major task in data stream analysis is the identification of complex events. A complex event can be considered as a pattern of events, which can occur within a predefined interval. AnduIN provides a set of operators for complex event processing that satisfy a large set of possible query scenarios [Klan, et al. 2011].

Benefits & Extensibility:

AnduIN can be easily integrated in existing solutions. External applications can send data to and receive from AnduIN by simple socket connections. The system also contains operators to join streaming data with data from standard SQL databases. In addition to these built-in operators, AnduIN provides an easy to use scripting interface to implement new operators without recompilation [Klan, et al. 2009]. It is also possible to combine sets of operators to more complex predefined operators.

3.1.6. Conclusion

Data Stream Processing Engines	Continuous Query	Real Time Optimization	Distributed Processing	Complex Event Processing
Aurora	✓	✓	✗	✗
Borealis	✓	✓	✓	✗
TelegraphCQ	✓	✓	✓	✗
AnduIN	✓	✓	✓	✓

All of the above mentioned data steam engines are summarized above. Aurora and Borealis data stream processing engines lays the basics for modern day processing engines. They do not provide the real time complex event processing on the fly as provided by AnduIN. COSMOS system requirements will define which Data stream processing engine will suit our purpose. If we want to send all our pre-processed or raw data to central bus and then implement CEP on it, then it is preferable to use any simple data stream engine like Borealis or Telegraph which provides basic functionalities like aggregation.

3.2 Data Analytics

This section presents fundamental literature review of different state of the art methods which are currently in use for knowledge extraction from raw IoT data. Data mining algorithms based on machine learning and statistical data analysis methods form the basis for many recent innovative applications in the domain of IoT by extracting high-level knowledge from raw data. A higher-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 enables to extract these interesting events. Whereas, statistical analysis tools enables to extract the temporal relation between the data. An overall picture of main data analytics tool which are common in IoT is shown in Figure 2.

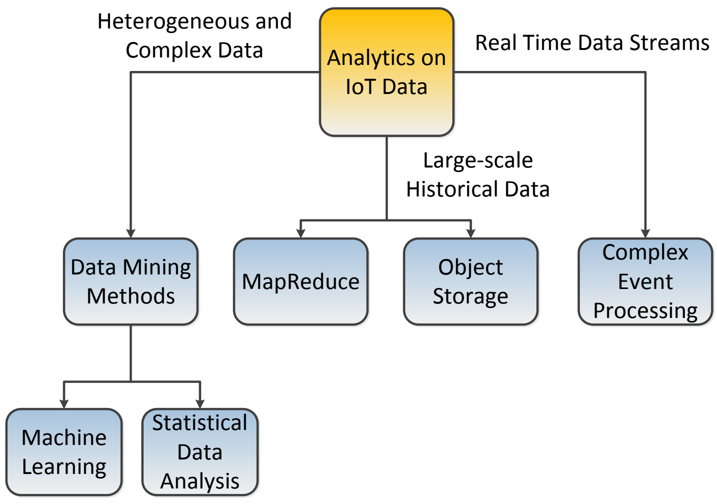

Figure 2: Data Analytics on IoT Data

3.2.1. Machine Learning

Machine Learning (ML) 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

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 3 . A brief introduction to few of these techniques is given in this section.

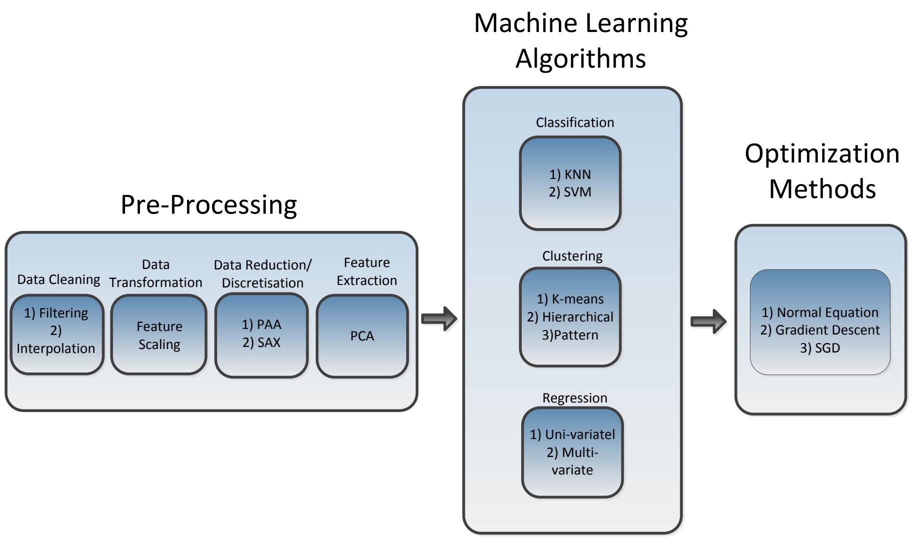
Pre-Processing
Data Cleaning
1) Filtering 2) Interpolation
Data Transformation
Feature Scaling
Data Reduction/ Discretisation
1) PAA 2) SAX
Feature Extraction
PCA
Machine Learning Algorithms
Classification
1) KNN 2) SVM
Clustering
1) K-means 2) Hierarchical 3)Pattern
Regression
1) Uni-variatel 2) Multi-variate
Optimization Methods
1) Normal Equation 2) Gradient Descent 3) SGD

Figure 3: Efficient Implementation of Machine Learning

3.2.1.1. *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.

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.

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.

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.

3.2.1.2. *Machine Learning Algorithms*

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].

Several examples are found in the literature where classification analysis is applied in IoT for inferring high-level knowledge from raw IoT data. One such application in the context of smart energy is given in [Zoha, et al. 2012]. The authors applied different variants of SVM and KNN

for appliance recognition by monitoring the electricity data and identifying the pattern by matching it with the stored data. The appliance level load information helps the users to identify energy hungry devices and helps in optimal resource utilization. Classification analysis can be used for many applications in *Intelligent Transportation System* (ITS) as well. Detecting the type of vehicles is one application where classification analysis can be used to help the traffic administrators to understand the traffic flow and take important decisions in managing it as proposed in [Ng, et al. 2014]. Another important area where SVMs has been applied in intelligent transportation system is for predicting the real time traffic flow [Xu and Yang. 2005]. Another application of SVM is found in [Yuan, et al. 2010], where authors applied SVM for classifying Internet traffic for network security monitoring. The advent of IoT has created growing interest in internet traffic monitoring and control from security perspective. The fast classification of Internet traffic can help in identifying malicious traffic and giving an insight about the utilization of network resources.
Decision tree is another popular classification algorithm which can be used for event detection in IoT. An example is given in [Bahrepour, et al. 2010] where authors used decision tree classification for fire detection using wireless sensor networks. Classification methods can also contribute towards providing more contextual information as one such example is discussed in [Mahalle, et al. 2013], where authors applied Bayesian decision theory to classify objects with respect to their context in order to contribute towards more context aware systems.

Imbalance issues happen in classification problems whenever any specific class has very few samples, and even though classifier can have very high accuracy in general but it is possible that for that particular class, it is not working accurately. In [Krawczyk, et al. 2014], the authors address the imbalance issues in classification using breast cancer malignancy data. The breast cancer data provides perfect example because of the fact that there are very few cases which are diagnosed with high malignancy (cancer) as compared to overall test cases. For example, if there are 4 patients which are diagnosed to have cancer from 100 patients and the classifier always indicate no cancer; the efficiency of classifier will be 96% but still its performance will not be acceptable for this application. The authors applied one-class decomposition ensemble approach which is aimed at distinguishing one specific class known as target concept from the broader set of classes, and achieved significant improvement in the performance of a classifier for detecting high malignancy.

Clustering Analysis:

Clustering refers to the unsupervised machine learning technique which is used for grouping similar objects on the basis of pre-defined metric such as distance or density. As opposed to the classification analysis, it does not involve training period. Cluster analysis is a general technique used widely for knowledge discovery in big data sets and several variants of algorithms are found in the literature. The choice of a particular clustering algorithm and parameters (such as type of metric, threshold and number of clusters) is governed by the problem scenario and the data sets involved in the problem.

Cluster analysis generates a hypothesis about the given data, whether the data under observation has distinct classes, or overlapping classes or classes with fuzzy nature. With the advent of big data, clustering applications have even increased. Clustering serves as the basic of many data mining techniques and finds applications in almost every industry. For example market researchers apply clustering techniques to group the customers into different segments, such that customers with common interests are in a same group. In this way, they can target different groups with different focused offers which are more related to them. Social networks apply clustering to group similar people into communities in order to help in recognizing people with similar interests from a larger set of people. Another interesting application of clustering is used by insurance companies to group different areas on the basis of crime rates so that they can offer different rates for insurances to the different areas depending on the level of insecurity.

One of the widely used clustering algorithms which is also among the top 10 algorithms used for data mining is *K-means algorithm* [Wu, et al. 2008]. K-means is an iterative process which forms the clusters by finding centroids such that the sum of the squares of distance from centroids to data is minimized. For a data set X with n number of samples, it divides them into k number of disjoint clusters, each described by the centroid value. In the first initialization step, it assigns initial centroids, most commonly by selecting k number of samples randomly from the data set X. In a second step it assigns each sample to its nearest centroids and forms k clusters. In the third and final step, it creates new centroids by calculating the mean value of the samples assigned to each previous centroid and calculates the difference between the old and new centroids. It keeps on iterating the process until the difference is less than a threshold value. K-means algorithm finds the local optimum solution which depends on the initial positions of centroids it has chosen. Hence usually it is run multiple times with different random initializations and chooses the best result from multiple runs. The number of clusters k have to be specified in advance and is a major drawback in the approach as the distribution of data may be unknown [Pham, et al. 2005].

Another famous clustering technique is hierarchical clustering which is based on the idea that nearby objects are more related to each other as compared to farther away objects. The objects are connected to form the cluster where distance between them serves as the deciding metric. It does not provide single partitioning of the data set, instead different clusters are form at different distances that forms an extensive hierarchy of clusters. The different hierarchical clustering algorithms depend on how the distance between objects is calculated. Euclidean distance, cosine distance and Manhattan distance are few famous choices for calculating distances between different objects.

Most of the clustering algorithms use distance as a metric for calculating similarity between data points. However, whether it is Euclidean distance or Manhattan distance, it does not always capture the correlation between data sets. The algorithms based on a distance metric require similar objects to have nearby values in order to be in the same cluster. But there are many applications in which different objects follow the same pattern or behavior while their values might not be in the same range. For such applications, a clustering approach is proposed in [Wang, et al. 2002] where similarity is based on pattern matching. The authors called their algorithm as *pCluster*, standing for Pattern cluster. It looks for coherent patterns and forms clusters of objects which exhibit similar pattern behaviors.

Clustering analysis has found its use in many applications in IoT. In electricity markets, the distribution service providers can group the similar users by applying clustering on the electricity consumption data. It enables the companies to target the similar customers with the same tariffs and provide more freedom to have different tariff rates depending on the usage [Chicco, et al. 2003]. Smart energy solutions such as *Non-Intrusive Load Monitoring* (NILM) can use clustering to identify the different types of appliances which are currently in use as the authors have shown in [Gonçalves, et al. 2011]. In their work, the authors detect steady state changes in the power signal of the energy consumption data of house, measure the average change in power and use these features to automatically cluster the events. The resulted set of clusters corresponds to the different combination of appliances. They employed *Blind Source Separation* on the centroids of the different clusters formed to make the best fit of appliances states. They applied K-means and hierarchical clustering analysis on their data to compare the results. The application of clustering analysis is diverse and can be found in vast range of fields such as detecting credit card fraud from online transaction data [Cabanes, et al. 2013], event detection for surveillance systems from video captured data [Piciarelli and Foresti. 2006], detecting abnormalities from brain images data for medical research purposes.

Regression Analysis:

Regression analysis is one of the most generic machine learning tool, which is used to define a relation between variables so that the values can be predicted in future using the defined relation. In general, the dependent variable is considered as output and the set of independent variables form the input. One of the practical examples of regression analysis is to identify a relation between energy use and weather. In this example, energy usage is a dependent variable and weather parameters such as outside temperature, wind and humidity are independent variables that act as the input. It is a general observation that if the weather is cold, the energy consumption will be higher due to usage of heating system as compared to mild or warm weather. If the dependent variable is relying on only one factor, it is called uni-variate model whereas-like in our example if it depends on more than one variable it is called multi-variate model. The energy demand can vary linearly along the weather parameters. Alternatively, there can be a non-linear relation between them. As we increase [Chaira. 2011] the complexity of the model, it may become more accurate

3.2.2. Statistical Data Analysis:

Machine Learning methods described in Section 3.2.1 are example of discriminative models which are not suitable for inferring temporal relation between entities. In such cases, another option is to use generative models such as Markov Model or its extension known as *Hidden Markov Model* (HMM). Generative models are difficult to train but they provide more insight about the system and have the ability to generate input sequences from the given output which is not possible for discriminative models.

Generative models are an example of statistical modeling. Several time series analysis techniques such as *Auto Regressive Moving Average* (ARMA) and *Auto Regressive Integrated Moving Average* (ARIMA) are also examples of statistical data analysis techniques which we will discuss later in this section.

3.2.2.1. Time Series Analysis

In the IoT context, optimized management and actuation may be heavily based on prediction of key values and metrics that drive decision making frameworks. One basic aspect of this process is human behavior and the underlying patterns (in terms e.g. of smart homes usage, power consumption, activity etc.). Furthermore, sensors in many cases involve weather readings (e.g. temperature). In both cases these types of metrics typically exhibit a high degree of periodicity and seasonality. However in most cases this periodicity may not be directly noticeable or it may involve multiple overlying patterns that distort the final output. In an automated context, the identification of such patterns should be performed automatically and at multiple levels, and ideally in a computationally non-demanding manner in order to be able to effectively retrain the model. One key method for this can be the analysis in Fourier series of sines and cosines. Analysis using the Fourier factors is used in order to depict the effect of various frequencies in the overall measured entity. This way the main signal can be separated from the noise. In this case it may be used to identify different patterns (with different frequencies) that may affect the overall demand (for example, the main period of user energy demand “signal” can be considered the working day, while larger periods of influence may indicate specifics such as weeks, weekends, summer/Christmas holidays etc.).

Fourier series analysis has been used in the context of Cloud based systems load forecasting [Kousiouris, et al. 2013] and has achieved satisfactory performance on multiple steps ahead of prediction (in the particular case up to 150 steps ahead). A more robust approach that incorporates a variety of influence parameters such as trend, seasonality and exponentially decreasing weights in past values is the classic Triple Exponential Smoothing (or Holt-Winters method) or its extended approaches [Gelper, et al. 2010]. However there is a significant drawback in this approach. It needs as input an accurate value of the time series inherent periodicity, otherwise the prediction results will be far from satisfactory. Indicatively, we have applied various base period numbers (in number of samples) for the data set used in [Kousiouris, et al. 2013], in which it is evident that if the correct base period (672 samples) is not used, then the prediction capabilities of H-W are severely degraded. Thus passing the dataset from a Fourier filter is critical in order to obtain this base period accurately, prior to applying the H-W method (indicatively in this case the Fourier analysis returned a value of 671.2) (see Figure 4 below).

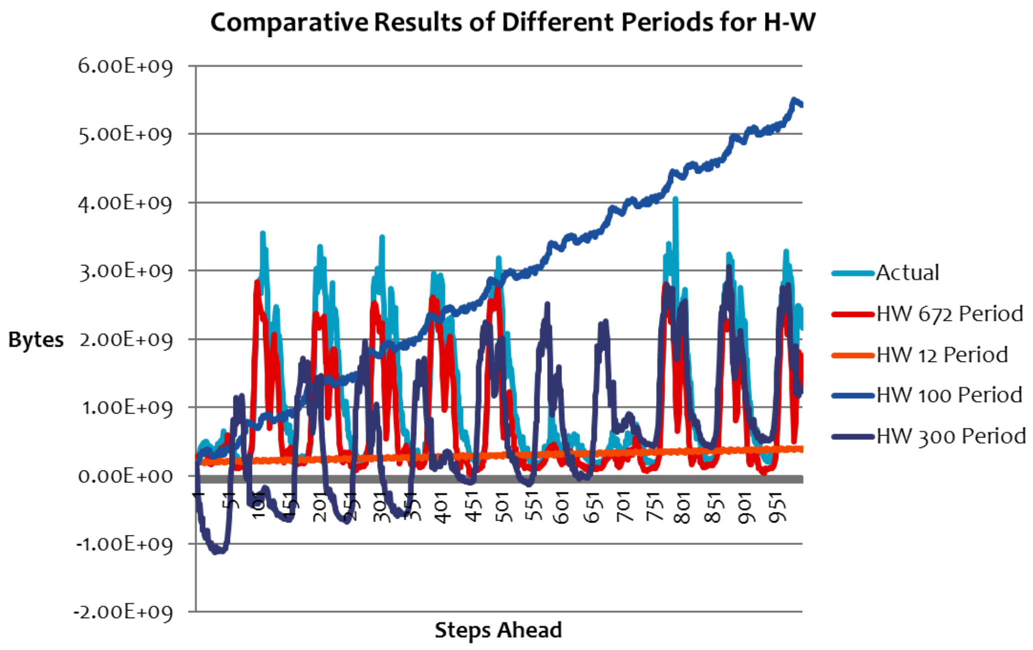

Figure 4: Holt Winters Sensitivity for multiple step ahead prediction in the dataset of [Kousiouris, et al. 2013]

3.2.2.2. 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 5 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].

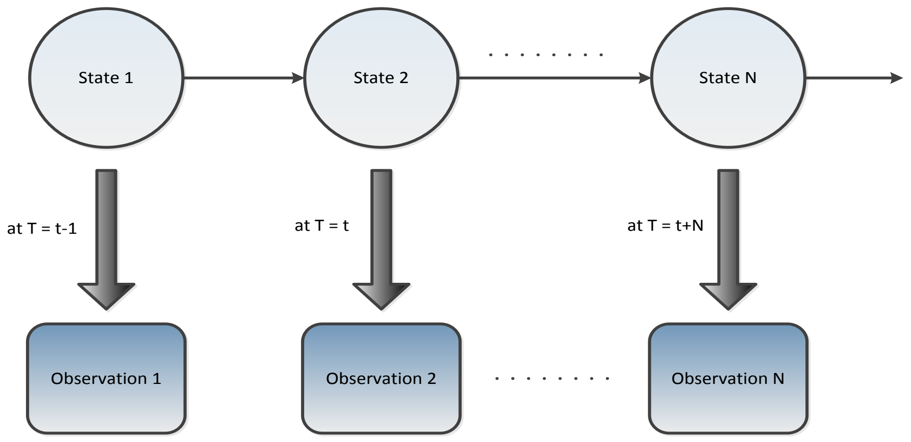

Figure 5: Hidden Markov Model

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.

3.2.3. Tools for Data Analysis

Data Analysis is a broad field involving both investigative analysis and operational analysis. Investigative analysis is targeted for research purposes by exploring new optimized methods and answering questions to gain more insight whereas operational analytics is more suited to established methods and involves established tools. It would be wonderful if there exist one tool for investigative analysis and operational analysis but unfortunately it is not the case. The data analysis platforms based on programming languages like R and Python are used extensively by people in academia for investigative analysis and frameworks like apache Hadoop and Map Reduce are used by people in industries for data analysis. We provide a brief overview on different data analysis tools available nowadays.

Python is a general purpose high level programming language which is used extensively for data analysis. The libraries like *scikit-learn*, *pandas* and *numpy* provide excellent platform for carrying complex machine learning and statistical analysis on the data. The different APIs available in Python makes it really easy to work with data in different formats. But the biggest disadvantage of Python is that it is suitable for the amount of data which fits in the memory of one machine. R offers a rich environment for statistical analysis and machine learning, but it has some rough edges with different pre-processing tasks involved with data in different formats. And like Python, it is not suitable for data analysis on very large datasets.

It is possible to develop distributed machine learning algorithms on the classic Map Reduce computation framework in Hadoop (see Apache Mahout). But Map Reduce is very low-level and difficult to express complex computations in. Also it is not possible to express all machine learning algorithms in the form of Map Reduce. Another tool, Apache Crunch offers a simpler Java API for expressing Map Reduce computations. But still, the nature of Map Reduce makes it inefficient for iterative computations, and most machine learning algorithms have an iterative component. Recently, a new distributed data analysis platform called Apache Spark is introduced which is aimed to reduce the gap between investigative and operational analysis tools. Apache Spark has inbuilt libraries for carrying complex machine learning tasks in an optimized way and at the same time able to scale in large manner.

3.2.4. Analytics close to the Data Store

In order to get true value out of our data, we propose modeling and storing Things as data objects with metadata that can be semantically understood, reasoned about, and tracked over time. With the massive quantities of Things that are commonplace in the IoT domain, we can on the one hand harvest truly valuable insights from the data, but on the other hand, are faced with a data management and analysis challenge which is greater by several orders of magnitude. This challenging area has been named Big Data, and we propose applying techniques from this field to the IoT domain. Big Data was described by Gartner using a "3Vs" model, as data which has high Volume, Velocity and Variety [Laney, 2001]. IoT data definitely meets these criteria since the number of Things is both massive and heterogeneous, and the rate of data capture is extremely high.

In the last decade, Big Data has attracted considerable attention, as data has outgrown standard relational data warehouses and moved to the cloud. This trend was led by prominent Web 2.0 companies such as Google, Yahoo! and Facebook, who analyze massive amounts of data in private clouds every day and use it for business critical applications such as advertising. As a result, new paradigms and techniques have been developed to analyze data at scale. For example, the Map Reduce paradigm [Dean, Ghemawat, 2004], originally developed by Google as a generic but proprietary framework for analytics on Google's own Big Data, has been widely adopted and embodied in many external tools, for example, the open source Hadoop [Hadoop]. The beauty of Map Reduce lies both in its generality and its inherent scalability. The framework can be applied to any domain, since users develop domain specific Map and Reduce functions in a general programming language, which are applied by the framework to Big Data. Inherent scalability is achieved since the Map Reduce paradigm defines a family of computations which can be processed at scale on a distributed system. The framework itself handles the low level details of job scheduling and management, data access, storage of intermediate results, and so on, in a widely distributed setting. Note, however, that some analytics workloads cannot easily be expressed as Map Reduce computations, and Map Reduce is just one example of a paradigm for analytics computations.

Hadoop, the open source embodiment of Map Reduce, was first released in 2007, and is now used by hundreds of companies for a variety of use cases [Hadoop Usage]. Notably, Amazon released *Elastic Map Reduce* (EMR) [Amazon EMR], a version of map reduce integrated into its own cloud infrastructure platform running Amazon EC2 and S3. EMR allows easy deployment of a Map Reduce compute cloud running on an EC2 cluster, which can read input data from S3 and similarly write its output to S3. OpenStack has a similar framework called Savanna which can be used to provision and deploy Hadoop clusters. Within this framework Map Reduce computations can read input data from OpenStack/Swift and write output data there as well – similarly to the case for EMR and S3.

Note that EC2 and S3 are separate cloud infrastructures, so the input data needs to be shipped across the network both in order to be read from S3, and to be written back to S3 [Amazon EMR Training]. This is true in general when data residing on a storage cloud needs to be processed in a compute cloud – the data needs to be shipped across the network and the cost of this can be prohibitive for large amounts of data. A common observation is that it is often preferable to move computation close to the data in order to avoid this kind of data shipping, but this is not usually supported on public cloud storage. Storlets address this issue by providing a mechanism to run arbitrary user computations on a storage cloud close to the associated data objects [Preservation DataStores, 2008]. Similar motivation drives the ZeroVM project [ZeroVM], which focuses on ultra-lightweight virtual machine infrastructure which can be used to run storlet type computations. Storlets were first introduced in the FP6 CASPAR project, where storlets were used for data preservation [Preservation DataStores, 2008], and were further developed in the context of FP7 projects VISION Cloud and ENSURE. VISION Cloud [VISION Cloud, 2011] developed a generic framework for storlets, including both a programming environment for storlets and a runtime execution environment for their secure execution. The runtime environment ensures that a storlet is appropriately triggered and runs it in a sandbox so that it can only access resources to which it has credentials.

Recently, a new cluster computing framework called Spark has been developed and has gained significant interest in the Big Data community [Zaharia *et al.* 2010], [Zaharia *et al*. 2012]. Like Hadoop, Spark supports Map Reduce style computations and can access data from the HDFS the Hadoop Distributed File System and Amazon S3. In addition IBM Research Haifa recently contributed support for Spark to access data directly from OpenStack Swift. The Spark framework is particularly advantageous for applications which reuse a working set of data across multiple parallel operations. Spark developed the notion of a *Resilient Distributed Dataset* (RDD) which represents a read-only dataset which is partitioned across multiple machines and can be rebuilt if some partitions are lost. RDDs can be cached in memory and can be reused in multiple Map Reduce like operations. This can give significant performance improvements for certain classes of applications compared to Hadoop style Map Reduce, which writes all intermediate results to disk. For example, Machine Learning algorithms often involve iterating a (Map Reduce) computation to optimize a certain parameter (e.g. through gradient descent). For such applications Spark can outperform Hadoop by an order of magnitude.

Similarly to the case for Hadoop, storlets applied at the data storage can reduce the amount of data which needs to be transferred from the storage cluster to the compute (Map Reduce) cluster. For example, storlets can pre-process data before applying a machine learning computation to it. In the case of Spark, this also reduces the amount of (precious) memory needed to store a computation's RDDs.

3.4 Metering, Telemetry and Control. Interoperability

3.4.1. Smart metering

Metering and telemetry have advanced mainly in the area of digital processing with reliable and affordable communications that can accompany these measurement devices. For instance, an electricity meter has progressed from a mechanical meter to using digital methods for representing and storing readings, but *Advanced Metering Infrastructure* (AMI) has enabled those digital meters to provide new services to accompany digital registers for time of use tariffs for energy, prepayment and load control. All of these services rely on a combination of persistent memory on-board the meter, processing to read the metering element and coordinate the transmission of data plus executing some logic for register control and switching. For the purposes of COSMOS, "smart metering" will be synonymous with AMI.

Gas and water metering have benefited from the same advancements, however a safe and reliable supply of power for these meters is not as readily available as with an electricity meter. Gas meters must comply with safety standards which specify electromagnetic field power limits within proximity to gas supply and further complicate AMI deployments. Likewise water infrastructure has a difficult operating environment and maybe far away from a power source. With some buried or covered water infrastructure, radio communications can be challenging with sub-GHz bands being preferable to penetrate earth. Generally, the drawback to sub-GHz is interoperability, range, duty cycle limits and low bandwidth.

There is a lack of an official definition for AMI however Wikipedia provides a definition that appears like a consensus:

> *AMIs* are systems that measure, collect, and analyze energy usage, and communicate with metering devices such as electricity meters, gas meters, heat meters, and water meters, either on request or on a schedule. These systems include hardware, software, communications, consumer energy displays and controllers, customer associated systems, *Meter Data Management* (MDM) software, and supplier business systems.

At an EU level, there is legislation in place that places an obligation on utility providers to meter using "smart" meter technologies from July 2014 onward. This will drive deployments of smart meter infrastructures at a national level and create opportunities for economies of scale, market innovation, demand shift/reduction, integration of renewable energy sources, distributed generation and grid feedback for better operation of distribution assets. By 2020, it is anticipated that all meters within the EU will be "smart".

In terms of end user benefits to "smart" metering most national standards, there are typically mandates for *In Home Displays* (IHDs) and *Home Area Network* (HAN) services to be provided by the *Energy Services Interface* (ESI) of the smart meter. IHDs will display current power consumption, energy usage over various periods, pricing information, cost of energy per various periods, control of demand response events and messages from the utility provider. The HAN is similar in its use of the meter for ESI data with added features such as the implementation of load control (switchable circuits or mains plugs), appliance level sub-metering and integration with home automation. Each regional/national strategy has specified and in many cases has implemented standards for protocols from the physical layer up to the application layer.

3.4.2. Interoperability in industrial data acquisition and process control

Controlling industrial processes involves reading data from sensors and controlling actuators often provided by different vendors. Software solutions are built on top of such hardware infrastructure and the solution as well, can be constructed using software components from different parties.

Supervisory Control and Data Acquisition (SCADA) are solutions capable of handling remote data acquisition and control from multiple sites and providing an integrated view of large processes. But SCADA solutions are not only found in processes such as manufacturing and production environments, they are often deployed in infrastructure and utility applications such as water, gas, electricity distribution or *Heating, Ventilation and Air Conditioning* (HVAC).

While the former, industry oriented, applications might not be the first candidate for an IoT related integration, one can easily see the interoperability potential with the latter ones. "Smart" and accessible utilities applications are on a rising trend and appeal to any IoT related research activity when it comes to use cases.

A lesson which has been learned in the industry field is that interoperability is crucial for the acceptance of a product which a hardware or software developer places on the market. Even if well-established hardware and software vendors will continue to support their proprietary protocols, almost every vendor will consider the support for a standardized interoperability interface, allowing customers the freedom to combine hardware and software from multiple sources.

The need from an interoperability standard has led to the development of *OLE for Process Control* (OPC) standard or in the extended form *Object Linking and Embedding for Process Control.* Developed by the OPC Foundation, released in 1996 and later renamed *Open Platform Communications*, OPC has allowed the integration of solutions from multiple vendors without the need of costly and time consuming adaptors and translators. OPC is based on now outdated or defunct Microsoft technologies such as OLE, COM and DCOM but has served solution developers for many years.

The rapid growth of internet and the extended need for distributed, internet enabled applications, along with the limitations of the original OPC standard lead to the development of a new *Machine-to-Machine* (M2M) protocol named *OPC Unified Architecture* (OPC-UA). This specification, based on the *Service-Oriented Architecture* (SOA), has been released in 2006. It addresses many of the issues related to the classic, DCOM based OPC standard in areas such as configurability, security, accessibility or scalability. OPC-UA has truly multi-platform support since ANSI-C, .NET and Java implementations are developed and can coexist.

The adoption of OPC-UA by numerous hardware and software vendors means the many of the future physical resources (sensors and actuators) or services will be accessible in a unified manner over the internet following the IoT trend. This creates a strong potential for bridging various IoT platforms and providing mutual data exchange.

3.5 Situational Awareness

"To see, to hear, means nothing. To recognize (or not to recognize) means everything."
[André Breton]

The COSMOS project involves a series of actors which generates a vast amount of information, commonly known as *raw data*, which is unprofitable unless significance is given to it. There are several methods involved in the process of taking advantage of such generated data, being Situational Awareness (SA) one of them. The following sections provide a sharp vision of SA, when was originated, what are their definitions, where was applied in the past, how is being exploited nowadays in the IoT paradigm, and how can the project take advantage of it.

3.5.1. Origins

It is not an easy task to find a specific, unique definition for *Situational Awareness* as there are a number of them in the literature, applied to many different scenarios. Even though the origin of the term is somehow recent (mid 1980s), the concept behind the scenes originated much more before, evolving from the military theory, thus being found mentioned by several Defence and Strategic institutions:

"*The knowledge of the elements of the operational environment necessary to make well-informed decisions.*" [Conseil de normalisation de terminologie de la défense]

"*The understanding of military and non-military events, activities and circumstances within and associated with the maritime environment that are relevant for current and future NATO operations and exercises where the Maritime Environment (ME) is the oceans, seas, bays, estuaries, waterways, coastal regions and ports.*" [Koscielski, 2007]

"*The ability to have accurate real-time information of friendly, enemy, neutral, and non-combatant locations; a common, relevant picture of the battlefield scaled to specific levels of interest and special needs.*" [Endsley, 2000]

In fact, from the past 30 years, SA is being implemented as a crucial component for crews in military aircraft, as can be observed in the following definitions:

"*SA refers to the ability to rapidly bring to consciousness those characteristics that evolve during a flight.*" [Wickens, 1992]

"*SA is the pilot's continuous perception of self and aircraft in relation to the dynamic environment of flight, threats, and missions, and the ability to forecast, then execute tasks based on that perception.*" [McMillan, 1994]

"*SA refers to the pilot's cognitive understanding of the current situation and its implications.*" [Vidulich, 1995]

"*Situation awareness is adaptive, externally-directed consciousness that has as its products knowledge about a dynamic task environment and directed action within that environment.*" [Smith, 1995]

Space Agency (NASA) in [Gibson, 1997] or [Begault, 2011]; the Federal Aviation Administration (FAA) in [Durso, 1988] or [Truitt, 2001]; the European Space Agency (ESA) in [Bobrinsky, 2010]; or the commercial manufacturer Boing in [Boing magazine, 2012], to name only a few once.

Despite the fact that Situational Awareness is being applied into many domains, being the Internet-of-Things the one applicable to the COSMOS project, it is necessary to introduce the term Context Awareness (CA), since there may exist the wrong conception of being both terms – SA and CA – strictly equivalent, while they are not.

With the rise of IoT, Context Awareness has gained importance and relevance, being mentioned in numerous researches and publications – [Davy, 2008], [Perera, 2012], [Perera, 2013] –, and studied in a number of European founded projects – [PERSIST], [SENSEI], [CONSEQUENCE], [BUTLER] –.

Again, several definitions have been proposed for CA. As extracted from Dey work [Dey, 1999], one of the first mentions to *Context-Aware Computing* is found in Schilit and Theimer work:

"Software that adapts according to its location of use, the collection of nearby people and objects, as well as changes to those objects over time." [Schilit and Theimer, 1994]

A more recent definition links its meaning with the term *Ubiquitous Computing*[1]

"An application needs to be context aware to operate in ubiquitous environment. The key to the context-aware systems is to sense the background information of an object and consequently adjust the implementation activities of the system with the user." [Tuladhar, 2008]

Indeed, the definition that Dey proposes for *Context* has been notably accepted:

"Context is any information that can be used to characterize the situation of entities (i.e., whether a person, place or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves. Context is typically the location, identity and state of people, groups and computational and physical objects." [Dey, 1999]

This is a general, universal definition that can be applied to different scenarios, what may have caused 'Context-aware' to become synonymous with other terms: adaptive [Brown, 1996], reactive [Cooperstock, 1995], responsive [Elrod, 1993], **situated** [Hull, 1997], context-sensitive [Rekimoto, 1998] and environment-directed [Fickas, 1997].

[1] Ubiquitous computing is a concept in software engineering and computer science where computing is made to appear everywhere and anywhere, using any device, in any location, and in any format. A user interacts with the computer, which can exist in many different forms, including laptop computers, tablets and terminals in everyday objects such as a fridge or a pair of glasses. The underlying technologies to support ubiquitous computing include Internet, advanced middleware, operating system, mobile code, sensors, microprocessors, new I/O and user interfaces, networks, mobile protocols, location and positioning and new materials [Ubiquitous Computing].

3.5.2. Composition

An interesting fact that can be extracted from the literature is that while SA and CA are undoubtedly related, the former not only suggests being conscious of the situation/context of an entity (SA product), but also involves being aware of the process followed to generate it, thus having additional knowledge to predict future situations, and to react accordingly afterwards (SA process)[2].

The COSMOS project follows both SA and CA concepts. While the former has been observed as a crucial component in highly-risk scenarios –to prevent catastrophes or disasters; in avionics, military environments–, the latter is closer to IoT trends –smart cities, wireless sensor networks, big data–. In turn, it seems feasible to work on both of them, extracting the benefits of each one to infer in the Use Cases proposed in COSMOS, especially the Bus ITS.

In an effort to split the inherent complexity of working with these concepts, the following two definitions of SA have been selected:

> “*SA is the perception of the elements in the environment within a volume of time and space, comprehension of their meaning and the projection of their status in the near future.*” [Endsley, 1988]

> “*SA is the continuous extraction of environmental information, integration of this information with previous knowledge to form a coherent mental picture, and the use of that picture in directing further perception and anticipating future events.*” [*Dominguez, 1994*]

The above descriptions are complemented with the three hierarchical phases of SA that Endley and his colleagues proposed later on [Endsley, 1998]:

- Level 1 - SA Perception of the elements in the environment: The first step in achieving SA involves perceiving the status, attributes, and dynamics of relevant elements in the environment.
- Level 2 - SA Comprehension of the current situation: Comprehension of the situation is based on a synthesis of disjointed Level 1 elements. Level 2 SA goes beyond simply being aware of the elements that are present to include an understanding of the significance of those elements. Based upon knowledge of Level 1 elements, particularly when put together to form patterns with other elements, a holistic picture of the environment will be formed, including a comprehension of the significance of information and events.
- Level 3 - SA Projection of future status: It is the ability to project the future actions of the elements in the environment, at least in the near term, that forms the third and highest level of Situation Awareness. This is achieved through knowledge of the status and dynamics of the elements and a comprehension of the situation (both Level 1 and Level 2 SA).

[2] The concept of *SA product* and *SA process* is a distinction made by researchers in order to infer the idea that it is necessary to complement ‘how much SA’ an entity has with ‘under what resources and workload conditions it can best obtained and utilised that SA’ [ESSAI, 2000]

3.5.3. Classification of Context

Having a classification of 'Context' is valuable to continue splitting the complexity into smaller, simpler facts. The research process resulted in finding out that the BUTLER project already provided a precise and meaningful categorization [BUTLER D2.2] in late 2012. The authors of this work extracted from the literature different ways of classifying the term. The following table presents a summary of their work:

General Classification [Wrona, 2005]	System Context	Corresponds, for a given application, to the context information of the system (software and hardware) on which it runs as well as contextual information of the used communication system (for example, the wireless network type).
	User Context	It is any information that can characterize a user. This may be the age, location, medical history, biometric information, emotions, etc. It may also be user activities, social relationships, family, friends, colleagues, ...
	Environment Context	Represents information describing the physical environment that is not covered by the system and user contexts. Particularly the context information from external sources such as temperature, climate, lighting, ...
	Temporal Context	Defines all information related to time, especially: hour, day, month, year...
Per Nature [Abowd, 1997]	Physical context	Contains objective information about the use such as location, orientation, time, date ...
	Context-based actions	Includes user actions that attract attention: what the user thinks, says or what he is doing.
	Emotional Context	Represents the emotional state of the user.
Generators of Context [Dey, 2001]	Places	Applied to geographical spaces as rooms, offices, buildings and so on.
	People	Can be differentiated by groups or individuals.

	Objects	Refers to physical objects or software components.
Categories attach to previous generator [Debes, 2005]	Identity	Characterizes the entity with an explicit identifier, which must be unique in the application domain.
	Location	Includes positioning and orientation data as well as information on regional relationships to other entities (e.g. neighbouring entities). This includes geographic data and spatial relationships.
	Status	Contains properties that may be perceived by a user. For a place, it may be, for example, the current temperature, lighting and noise levels. For individuals, the state may refer to physical factors such as vital signs, tiredness or current occupation.
	Time	Any temporal information
Processing Complexity [Castellucia, 2007]	Simple	The collected information is used in its original format. For example, it may represent a parameter value.
	Interpreted	The collected data is converted into a more meaningful format
	Aggregated	A set of simple and/or performed inputs
Transmission Frequency [Brezillon, 2003]	Periodic transmission	For instance, for mobile terminals, it should be the orientation, the location (longitude, latitude, altitude), and the frequently changing parameters (e.g. time).
	Trans. on status change	Any information that changes at low rate. This includes for instance: health (body temperature, heart rate, blood pressure, ...), neighbourhood information, weather conditions (temperature, precipitation, ...), the terminal properties (charging state, ...)
	Onetime transmission	For information that do not change over time as: terminal properties (type, screen size, display resolution, ...), user preferences (interests) , personal Information, ...

3.5.4. Context-aware Architectures

Context-aware architectures are the pillar for building applications taking advantage of Situational Awareness concepts. This chapter presents a summary of some noticeable frameworks that have been developed in the past, along with some more novel solutions.

- Context Toolkit [Salber, 1999]

Context-Toolkit architecture gives solution for constructing context aware systems about the interaction facts with the physical sensors. The main components of Context Toolkit are shown in the following figure. The server aggregates all the context information from different widgets.

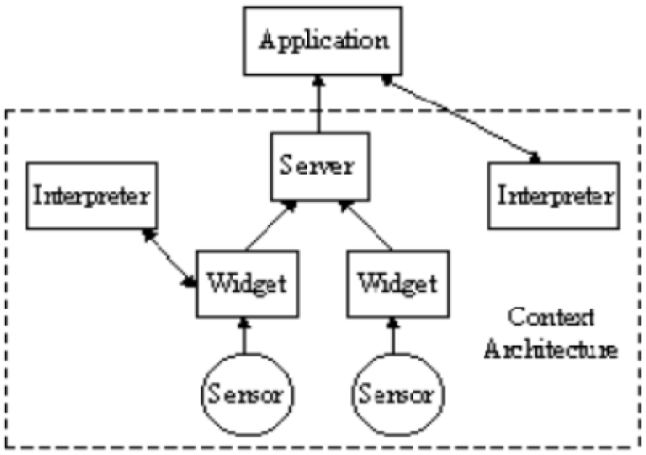

Figure 6: Context Toolkit Architecture

- CoBrA (Context Broker Architecture) [Chen, 2003]

CoBrA is an agent based context architecture to support context-aware applications in smart spaces. The intelligent broker agent acquires the contexts from various devices, environmental sensors, smart tag sensors and other agents. CoBrA provides reasoning, knowledge sharing, and privacy protection for the different contexts. The functional components of CoBrA are shown in the following figure.

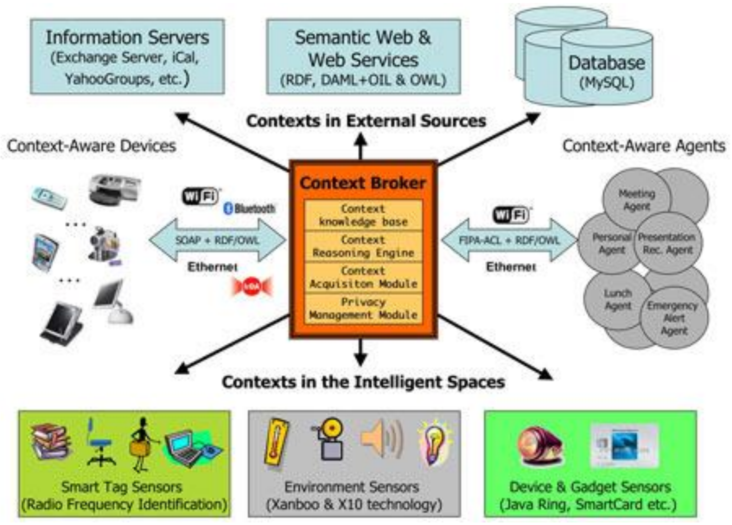
Information Servers (Exchange Server, iCal, YahooGroups, etc.)
Semantic Web & Web Services (RDF, DAML+OIL & OWL)
Database (MySQL)
Contexts in External Sources
Context-Aware Devices
Context-Aware Agents
Context Broker
Context knowledge base
Context Reasoning Engine
Context Acquisiton Module
Privacy Management Module
SOAP + RDF/OWL
Ethernet
Bluetooth
FIPA-ACL + RDF/OWL
Ethernet
Meeting Agent
Personal Agent
Presentation Rec. Agent
Lunch Agent
Emergency Alert Agent
Contexts in the Intelligent Spaces
Smart Tag Sensors (Radio Frequency Identification)
Environment Sensors (Xanboo & X10 technology)
Device & Gadget Sensors (Java Ring, SmartCard etc.)

Figure 7: CoBrA Architecture

- SOCAM (Service-oriented Context-Aware Middleware) [Gu et al, 2004]

SOCAM infrastructure obtains contexts from varied sources (external and internal) that are transformed for sharing and reuse for further devices into consequential contexts. Gu et al proposed SOCAM Architecture based on OWL[3]; that formulates the data distribution among several services, devices usable & easier. The components of the SOCAM architecture are shown in the following figure. Context interpreter is used for providing reasoning services to the obtained context information and Context database stores context for specific sub-domain and contexts ontology's.

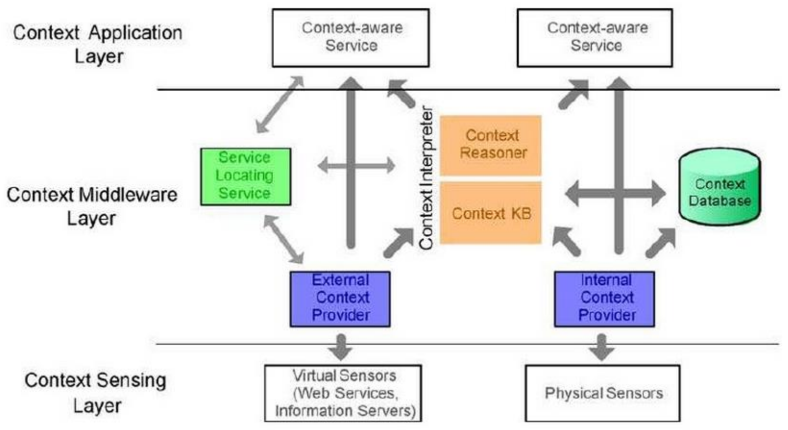

Figure 8: Overview of SOCAM Architecture

- Context-Aware Middleware [Bonino, 2007]

Context-aware middleware is the integration of two components, the Context Management Service (CMS) [Ramparany, 2007] and the Awareness and Notification Service (ANS) [Bonino, 2007]. The Context Management Service supports context sources to publish their contextual information to be used by context-aware applications and services. The Awareness and Notification Service offers a rule-based facility allowing client applications to subscribe rules containing context-based conditions and receive notification when the specified context holds. The following picture shows the architecture of the proposed combined middleware regarding the connections with client applications and context sources.

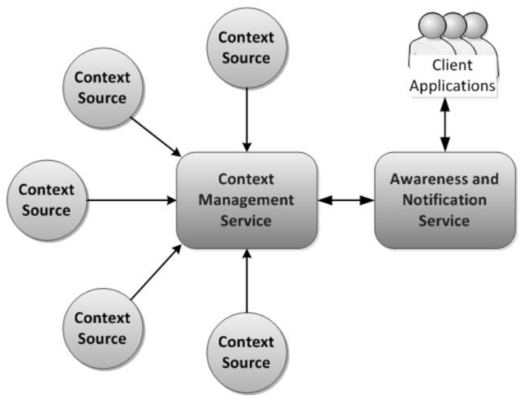

Figure 9: Context-Aware Middleware

[3] OWL - Web Ontology Language: http://www.w3.org/TR/owl-features/

- PredictiveWorks [http://predictiveworks.eu/]

Predictiveworks is an open ensemble of predictive engines and has been made to cover a wide range of analytics requirements. The Context-Aware Analysis Engine is one of the nine members of the open ensemble and has a strong focus on context-sensitive data and interactions in multi-dimensional datasets. It is the key to the next-generation of personalized predictions and customer relationships.

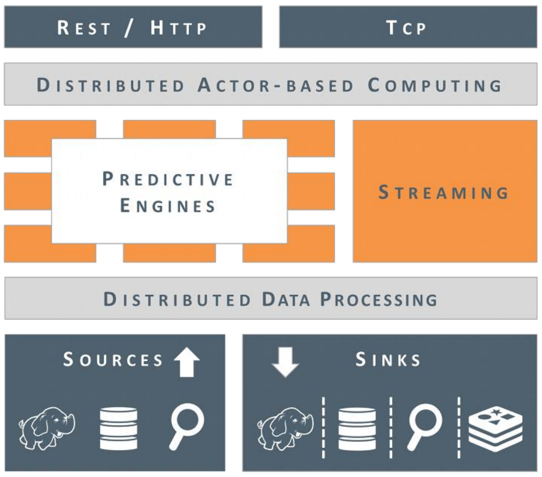
REST / HTTP
TCP
DISTRIBUTED ACTOR-BASED COMPUTING
PREDICTIVE ENGINES
STREAMING
DISTRIBUTED DATA PROCESSING
SOURCES
SINKS

Figure 10: PredictiveWorks Architecture

Apart from previous frameworks, a recent research made by Bettini et al concludes that hybrid approaches may be further extended to design a hierarchical hybrid context model that may satisfactorily address a larger number of the identified prerequisites.

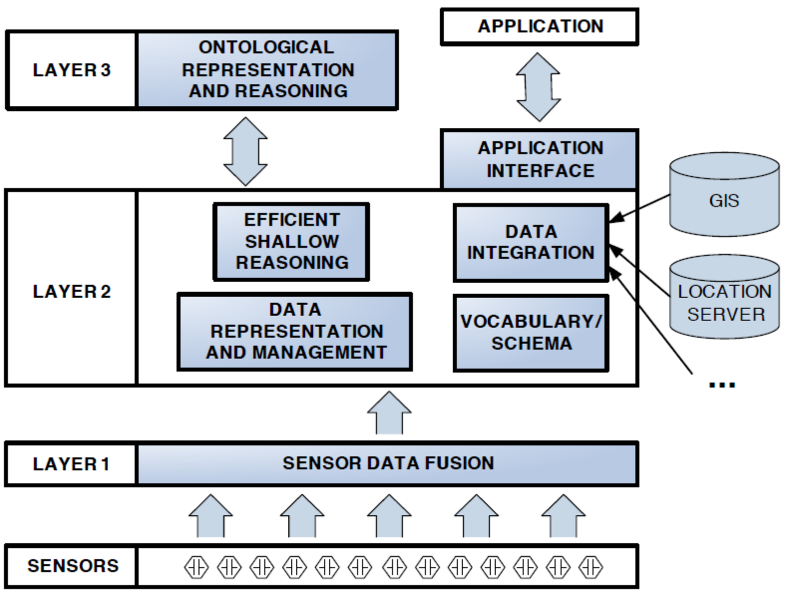

Figure 11: Multi-layered hybrid architecture

In the above picture, they propose a multi-layered architecture *to provide a more comprehensive solution, both in terms of integration of different forms of reasoning, and in terms of expressiveness. The proposed model includes a representation formalism to represent data directly acquired from sensors (or retrieved from a module executing some sensor data* *fusion technique). In order to support the scalability requirements of pervasive computing services, this representation formalism should make possible the execution of efficient reasoning techniques to derive high-level context data on the basis of raw ones (e.g., by executing rule-based reasoning in a restricted logic programming language, or statistical inferencing). Since such representation formalism inevitably does not support a formal definition of the semantics of context descriptions, a more expressive, ontology-based context model is desirable on top of it. In addition to providing a formal semantics of the data, an ontological context model also supports the execution of reasoning tasks such as consistency checking and derivation of new context information.* [Bettini et al]

3.5.5. Big Data Tools

This chapter briefly introduces some state-of-the-art tools that are used to acquire, process and analyse large amounts of data.

- Apache Storm [https://storm.apache.org/]

Apache Storm is a free and open source distributed real-time computation system. Storm makes it easy to reliably process unbounded streams of data, doing for real-time processing what Hadoop did for batch processing. Storm is simple, can be used with any programming language. Storm has many use cases: real-time analytics, online machine learning, continuous computation, distributed RPC, ETL, and more. Storm is fast: a benchmark clocked it at over a million tuples processed per second per node. It is scalable, fault-tolerant, guarantees your data will be processed, and is easy to set up and operate.

- Apache Spark [http://spark.apache.org/]

Apache Spark is an open-source cluster computing framework originally developed in the AMPLab at UC Berkeley. In contrast to Hadoop's two-stage disk-based MapReduce paradigm, Spark's in-memory primitives provide performance up to 100 times faster for certain applications. By allowing user programs to load data into a cluster's memory and query it repeatedly, Spark is well suited to machine learning algorithms.

- Esper [http://esper.codehaus.org/]:

Complex event processing (CEP) delivers high-speed processing of many events across all the layers of an organization, identifying the most meaningful events within the event cloud, analyzing their impact, and taking subsequent action in real time. Esper offers a Domain Specific Language (DSL) for processing events. The Event Processing Language (EPL) is a declarative language for dealing with high frequency time-based event data. SQL streaming analytics is another commonly used term for this technology.

- Apache Kafka [http://kafka.apache.org/]

Apache Kafka is publish-subscribe messaging rethought as a distributed commit log. A single Kafka broker can handle hundreds of megabytes of reads and writes per second from thousands of clients. Kafka is designed to allow a single cluster to serve as the central data backbone for a large organization. It can be elastically and transparently expanded without downtime. Data streams are partitioned and spread over a cluster of machines to allow data streams larger than the capability of any single machine and to allow clusters of co-ordinated consumers. Messages are persisted on disk and replicated within the cluster to prevent data loss. Each broker can handle terabytes of messages without performance impact. Kafka has a modern cluster-centric design that offers strong durability and fault-tolerance guarantees.

3.5.6. Conclusions

The literature contains numerous descriptions and usages of Situational Awareness since the past decades, being the great majority of them related to information fusion complex domains such as aviation and military operations. Thus, the applicability of these techniques into IoT scenarios should be extracted from lines closer to *Sensemaking* and achieving *understanding,* which are more commonly found in industry and the organizational psychology literature. In fact, Situational Awareness can be placed in the decision making process as depicted in the following figure:

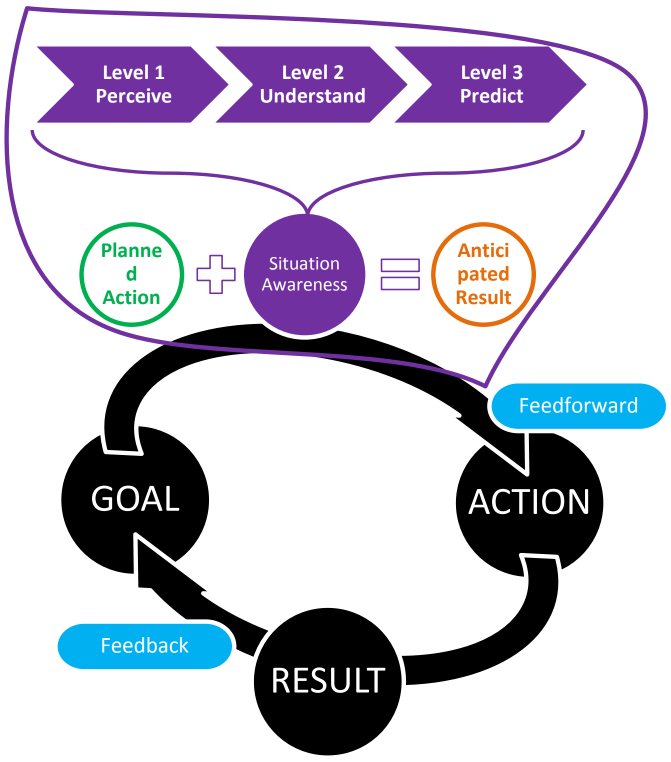

Figure 12: Situational Awareness in the Decision Making process

Having in mind that COSMOS has defined a number of Use Cases, once presented what is *Situational Awareness*, the relationship with *Context Awareness*, plus a classification of *context* as long as various references to context-aware architectures, the purpose now is to connect the outcomes of this work with the development and integration tasks.

In this line, the detachment of Situational Awareness in three levels – or processes – facilitates its linkage with some of the contributions provided in this document. For instance, acquisition of large amounts of data and stream processing analysis are functionalities that fall beyond the process of perceiving the elements of the environment – SA Level 1 –. Once done, comprehension and understanding of this context is crucial to move to next step.

Finally, the third level of SA – Projection of future status – is one of the topics that can be addressed by the combination of Machine Learning techniques and a CEP engine in order to maintain current situations and predict future ones. The main idea is to maintain not only the current context of the situation but create a system which is able to maintain and adapt with respect to potential future contexts. Machine Learning models have the potential to exploit historical data to make prediction models which can be combined with CEP techniques to correlate different events in order to predict a complex event which can happen in future.

3.6 Complex Event Processing

CEP is event processing that combines data from multiple sources to infer event or patterns that suggest more complicated circumstances. The goal of complex event processing is to identify meaningful events and respond to them as quickly as possible.

The term Complex Event Processing was popularized by D.C. Luckham in his book "The Power of Events: An Introduction to Complex Event Processing in Distributed Systems".

CEP has many independent roots in different research fields: discrete event simulations, active databases, network management and temporal reasoning.

It should be distinguished two cases, the first one where complex events are specified as a-priori known patterns over events, and the second one where previously unknown patterns should be detected as complex events.

In the first case, **event query languages** offer convenient means to specify complex events and detect them efficiently. In the second case, **machine learning and data mining methods** are applied to event streams.

Complex events detection is not an end in itself, but a mean construct histories (complex events), composed by many simple occurrences (simple events), to which the system (IoT system) have to react.

Here we should distinguish between Complex Event Processing and *Event Stream Processing* (ESP).

- **Complex event processing** is event processing that combines data from multiple sources (see http://en.wikipedia.org/wiki/Complex_event_processing - cite_note-2) to infer events or patterns that suggest more complicated circumstances;
- **Event stream processing** is a set of technologies designed to assist the construction of event-driven information systems. ESP technologies include event visualization, event databases, event-driven middleware, and event processing languages, or CEP. In practice, the terms ESP and CEP are often used interchangeably, with CEP becoming a more fashionable term recently.

Most of the products offered nowadays are more related to the concept of ESP, but all of them include a CEP. These CEPs offer different solutions and a lot of different technologies depending on the market' sector they are designed for: Synchronous, asynchronous, real-time, near-real-time, Java, C/C++, etc.

3.6.1. Events taxonomy (incl. reasoning with unsafe/incomplete events)

Event-driven solutions require a consistent view of events. Events collected by Complex Event Processors originate from different sources, such as message buses, databases or Internet protocols. Therefore clearly defined and standardized classification of events is necessary to create meaningful and easily understood events.

The successful evaluation of event requires not only ability to load event data, but also ability to correctly interpret meaning of information associated with event. Because of that, we separate a keen difference between **event syntax** and **semantics** in understanding how entities behave.

One very important aspect that must be taken into account is the inaccuracy and/or noise of input event stream. A naïve implementation of CEP may completely miss a complex event in such case. Depending on properties of input event stream, CEP may embed components for probabilistic and prediction compensation of inaccurate events.

Internet supports hundreds if not thousands different protocols. There are four major and widely adopted M2M “Internet of Things” connectivity protocols these days.

3.6.1.1. MQ Telemetry Transport (MQTT)

As stated in protocol specification v3.1, MQTT is a lightweight broker-based publish/subscribe messaging protocol designed to be open, simple, lightweight and easy to implement.

MQTT is oriented towards one-to-many message distribution. This protocol is implemented on top of TCP/IP stack that provides simple and reliable network connectivity. It also contains mechanism to notify connected parties about abnormal disconnection of a client which can be utilized by CEP awareness support.

The *MQTT for Sensor Networks* extension (MQTT-S) also supports non-TCP/IP networks such as ZigBee.

3.6.1.2. Extensible Messaging and Presence Protocol (XMPP)

The XMPP was initially implemented for near real-time instant messaging. The protocol is based on XML. This enables interconnection with other (non XMPP systems) over single dedicated gateway.

While XMPP provides flexible addressing scheme, strong security and stability, it was not designed for fast binary transfers. There are several extensions for XMPP to promote its applicability in Internet of things domain: the Efficient XML Interchange format that provides a way to compress XML documents and extensions providing basic operations and structures for Sensor Data, Provisioning, Control and Concentrators.

3.6.1.3. Data Distribution Service (DDS)

The DDS was designed to provide high performing, scalable, real-time and interoperable data exchange between publishers and subscribers. It can efficiently transfer large amount of messages at high transfer rate to many receivers at same time. Unlike MQTT and EMPP, DDS is built on top of UDP, yet it also implements reliable multicast as well. It is first choice for applications where high performing integration of devices is necessary.

3.6.1.4. Advanced Message Queuing Protocol (AMQP)

The primary goal of AMQP is to enable interoperation and resource sharing between new and legacy applications. Various broker architectures are already supported which may be used to receive, queue, route and deliver messages.

Particularly interesting feature of AMQP is that it enables specification of what messages will be received and where from, and how trade-offs are made with respect to security, reliability and performance.

3.6.2. Processing configurations (incl. fail safe configurations)

In addition to the information simple individual events carry, multiple events which come from the Internet of Things also provide significant meaning to form knowledge. Multiple events are typically combined, aggregated and correlated based on temporal, historical and other algorithms to detect meaningful patterns.

The complex event processing engines usually embed rules engines to recognize such patterns typically at real time and providing extracted information to applications making business decisions.

The widely used processing techniques provided by CEP are presented in the following.

3.6.2.1. *Filters*

An event filtering is the most basic functionality which supports other more complex patterns. A filtering engine takes stream of incoming events as input and evaluates a specified logical or arithmetic condition based on event attributes. If the condition is true, publishes event to observers. Typical conditions consist of “equals”, “greater/less than”, etc., and logical conditions like “and”, “or”, “not” etc.

It is possible to construct many different kinds of filters, for example, filters that compare events to events in same or different stream, or compare events to some aggregated values or thresholds.

A typical example of event filtering is capturing sensor readings where values fall outside of expected range.

3.6.2.2. *Windows*

An event stream can have an infinite number of events. Windows provide a means to select subset events for complex event detection. Out of many possible ways to select events, two most basic mechanisms include:

- **Time Windows:** in some specific cases, it is not sufficient to detect a complex event when certain values are detected. There is a need to take time range into account as well. For example detecting events when certain values exceed or fall below some threshold within specified time period. These time periods are usually represented by **fixed time window** or **sliding time window.** For example, sliding time window can collect all sensor readings during last two hours and fixed time window collects readings once every second hour;

- **Tuple Windows:** instead of measuring elapsed time, tuple windows select events based on number of occurrences of particular events within an input stream. A typical example of tuple window is collection of last twenty sensor readings for further evaluation.

The typical transient event lasts for infinite small period. However real world scenarios require support for so called “long lasting events”. The duration of such event is for fixed amount of time or until another event arrives.

3.6.2.3. *Joins*

In CEP, the idea of join is to match events coming from different input streams and produce new event stream.

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; this is what windows do. Most of CEP implementations support **window to window** joins, **outer joins** or **stream to database** joins.

The joins are used for example correlate information across different security devices for sophisticated intrusion detection mechanism and response.

3.6.2.4. Patterns and Sequences

A real world tasks require support for more sophisticated complex event detection mechanisms. Patterns and sequences match conditions that happen over time. For example, "a fire" is detected when within 10 minute interval sensor A detects smoke, followed by same event from either sensor B or C, followed by absence of event from sensor D. In addition to that, all events may be related to each other in some way.

3.6.2.5. Hierarchical events

While on the one hand emerging smart autonomous devices are capable to produce more complex hierarchical events, some application also require hierarchical events for further processing. For example, a "Traffic Crossing" event may contain a list of associated "traffic light states". Such hierarchical events are supported by the rise of SOA.

3.6.2.6. State machines

State machines represent a mechanism to model and track complex behaviors and processes. In this special case, provided queries define set of states whereas events define transitions from one state to another.

A typical example is tracking behaviors of different actors/devices that can be described as a series of transitions between states.

3.6.3. Context Awareness (a.k.a. Situation Awareness)

The goal of Context Awareness facilities is to simplify usage of and automated (or semi-automated) adaptation of systems and applications to different situations possibly at real time.

In IoT context, Things and their sensors are main source of information for situational knowledge acquisition. Using sensors, situations can be detected and classified as contexts. The events detected by sensors can be divided into two categories:

- **Discrete events**: events that occur at specific time point;
- **Continuous events**: events lasting for some time e.g. swimming, heating, driving etc.

There is not strict limitation in context categorization however some of most generic categories include: location, identity, time and activity. When system recognizes different contexts, this information can be used to change and adapt its behavior accordingly.

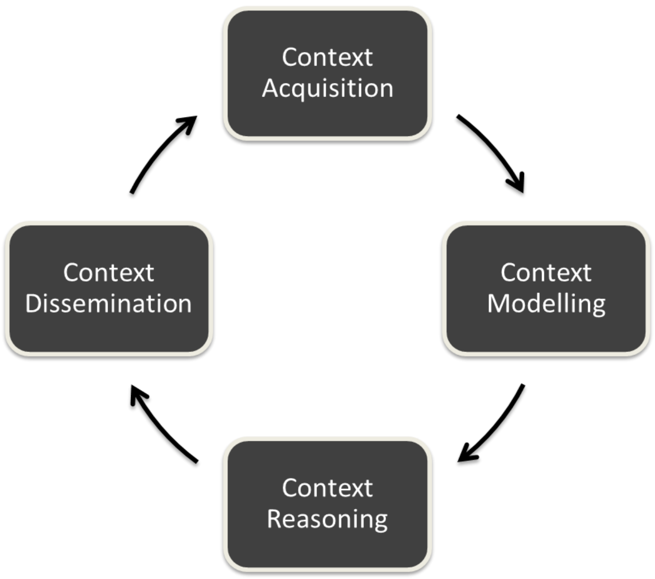

Figure 13: Context-Awareness Life Cycle

As depicted on previous Figure 13 above, context lifecycle consists of four phases. Acquired contextual information needs to be modeled into meaningful representation. The next step is to derive high level information from modeled context and publish result to consumers.

The actual modeling of context varies from one solution to another.

One of the major challenges of situational context building is proper interpretation of sensor information. Different mechanisms like CEP, machine learning, artificial intelligence are used for better understanding of sensor readings. Therefore it is also very important to consider and address situations when automatically generated context does not match reality.

3.6.3.1. Context sharing

A possibility to share context opens a wide range of new features. For example, a VE makes different decisions and actions based on the context in which the interaction takes place. However the VE may decide to make different decisions and actions when it knows from shared context that a similar action is already being performed by another VE to avoid possible interruption or some unwanted interference.

3.6.3.2. Resource management

The context aware resource management can optimize the operation of a VE and its use of resources depending on different contexts. For example, a VE running on battery can be activated only in certain situations to extend operation time.

3.6.3.3. Adaptive systems

Context-driven adaptive systems generally behave differently (e.g. provide different functionalities) in different contexts. This can be achieved by mapping different functionalities to different contexts.

3.6.4. CEP as a support to situation awareness

As mentioned in chapters above, CEP is connected to various input streams. From situation awareness point of view, CEP is monitoring various states and attributes of attached entities/devices via events, hence collecting awareness of multiple situational elements (element availability, condition, actions and events).

In addition to monitoring of various connected entities, CEP also provides mechanisms to recognize predefined patterns of events using. This information can be used to develop comprehensive picture of the network of connected elements.

3.6.5. Extensions of CEP to semantic stream processing

A traditional CEP is only processing engine without any knowledge about event semantics. This also means that provided queries or rules have to address events directly, otherwise CEP would not be able to match and detect complex events.

Semantic CEP on the other hand, provides a mechanism to embed ontology into various event processing and evaluation stages. By embedding ontology, semantic CEP provides a means to deduce additional knowledge resulting from given ontology concepts and relationships. Additional benefits of semantic CEP are that it naturally raises level of abstraction enabling users to specify “what to detect” instead of “how to detect” various situations via declarative event processing language.

The mentioned benefits provided by semantic CEP are not for free. Either the area of usability is limited to specific domain for example, finance, energy, logistics, etc., or the complexity of CEP and/or rule specification dramatically increases.

There are many different ways how to integrate ontology to build semantic CEP solution out of which, most widely adopted approaches are:

3.6.5.1. Event transformation

The main idea of this approach is to transform raw sensor events into “semantic stream” usually enhanced with timestamp. Examples are such streaming extensions are C-SPARQL, CQUELS or SPARQLstream.

3.6.5.2. DSL Workbench

A DSL Workbench enables definition of Domain Specific Language for declarative specification of complex event detection. These high level specifications are subsequently compiled into low level event processing rules.

3.6.6. Raw stream data processing (predict anomalies or off-normal events)

As CEP is processing an input event streams at real-time or near-real time, it can immediately react to various anomalies or risks, hence effectively minimizing possible damage. Various risks are detected either using signatures or anomaly detection mechanisms.

Every signature represents one particular risk. Signatures therefore detect only known risks but do not produce false positives. Finding patterns in data that do not match any predefined signature lead to anomaly detection.

Anomalies on the other hand side, represent significant differences from an expected state. Unlike signatures which represent black-list approach, anomalies are detected using white-list approach. Anomalies therefore on the one hand side may produce false positives but on the other hand side, unknown risks are detected. A widely used anomaly detection techniques consist of:

- Classification based techniques;
- Clustering based techniques;
- Statistical techniques;
- Spectral techniques.

3.6.7. CEP data persistence (post processing to detect behavior patterns)

There are two main reasons for persisting CEP data and simple/complex events. It may be storing if internal state of the engine for recovery purposes. Or and this commonly happens in modern solutions, to perform tracking, analyzing and other post processing over virtually unlimited history of captured events. A processing of big data composed of huge amount of historical readings requires decent storage power as well as transaction processing power.

The distributed CEP architectures may benefit from cloud data storage. These days, no cloud providers are able to interpret stored data as semantic the data is not supported. Any kind of post processing has to performed by CEP itself or dedicated external application.

Wide deployments of smart devices generate large volumes of data. With such large amount of data, new strategies and specialized techniques are necessary to interpret historical patterns, apply them to current situations and take accurate decisions. For example, public transportation in Madrid (which is addressed by COSMOS) provides various sensor readings used to adjust traffic activities. These are mostly reactive activities. On the other hand, analysis of traffic data over different parameters such as seasonal climate impacts (e.g. snow, flood), rush hours etc., leads to predictable patterns and trends over days, years or even decades. This enables infrastructure to provide early corrective measures in real or near real time itself.

3.6.8. Data broadcasting based on semantic analysis results

Processing of events and detection of complex events would make no sense without consumers having attention in them. An event consumer accepts complex events for further processing according to their business requirements.

There are various kinds of event consumers starting from ones offering some kind of user interaction (like alarm systems, computer interfaces, social networking), to raw machine consumers for example business processes or event logs.

In order to enable further processing, inference and knowledge discovery from semantic results provided by CEP, detected complex events have to be transmitted with unambiguous, shared meaning. This can be achieved via ontology specific vocabulary shared by both CEP and its consumers. In addition to that, it is strictly necessary that ontology is sufficiently expressive to properly describe meaning and that meaning of ontology does not change over time. The meaning of the data is transmitted together with data itself, in one, self-describing system independent package.

Although syntactic data exchange is pre-requisite for semantic data exchange, same semantic information may be accurately exchanged via different syntaxes. In other words, syntactic conversion of data from one format to another along communication path does not necessary breaks semantic communication.

3.6.9. Conclusion

The CEP, as general tool for combining data from multiple event streams to infer meaningful patterns may have many technical realizations – each offering different strengths and trade-offs. For COSMOS, a rule-based event inference processing engine is most suited although it is only one of many existing event processing techniques.

CEP Summary		
Processing levels	**Inference processing techniques**	**Main challenges**
• Event pre-processing • Situation detection • Predictive analysis • Adaptive processes	• Rule-based inference • Bayesian belief networks • Dempster-Shafer's method • Adaptive neural networks • Cluster analysis • State-Vector estimation	• Nose from data sources • Trending, evolutionary changes • Technical limitations: o Bandwidth o Latency o Memory o Processing power • Data out of order

3.7 Trust and Reputation

One of the Major requirements in COSMOS is developing trust between Virtual Entities and to make sure that Information is provided by the Entity which is expected to provide it and the information is reliable to take critical decisions. In literature, different mechanisms such as authentication, confidentiality and message integrity are proposed to achieve sufficient security and reliability in Distributed Networks. These mechanisms works fine in case of outsider attacks but they are not effective against insider attacks. With the passage of time, some nodes may become malicious and although they have all the legitimate right to pass information but they will be a threat to affect overall reliability of the system. In order to cope with this scenario, COSMOS proposes the concept of developing trust between Virtual Entities with their Experience. Things learn with their Experience and in case of faulty or effected node reading, they will gain bad reputation and their effect on the final decision will be slowly reduced until they gain their good reputation back. The reputation of a node will depend on the accuracy of its reading using different methods. In literature, Trust and Reputation topic in the context of Wireless sensor networks is not new and Researchers have been doing extensive research in the field. But Trust and Reputation in the domain of Internet of Things is rather new.

3.7.1. Trust and Reputation Techniques

Trust and Reputation (T&R) plays a key role in the social relationships. In fact, our society is based in the Trust and Reputation model. It is supporting all interactions between people (learning, shopping, working, helping, friendship, etc.).

- **Trust:** when agents expect a particular agent to do something;
- **Reputation:** when agents believe a particular agent to be something.

Repetition is the foundation of T&R. Each agent, into a system, has a Trust score about the performance of one agent doing a task. But this agent has a Reputation about doing this task which has been built based on the results it achieved each time it executed the task, and it will follow evolving with the results of future executions of the task. Also, the experience each agent obtains from interaction with other agents will modify the trust scores they have about the other ones.

Over the last two decades, scientist have been studying and developing different technics to get this tool securing different processes in many different environments. One of the first fields to adopt this Trust & Reputation technics was the Economics field, to model seller trustworthiness, reputation on goods of markets, lending and a lot of economics issues.

Internet made possible the interaction between agents without having a previous knowledge, without knowing the reputation of the other agents. Thus, first adaptations of Trust and Reputation mechanism came with the advent of e-commerce and social media. Some of the more representative examples are *eBay's, Expert Sites,* BizRate, *Amazon, Kuro5in or Google's Web Page Ranking System.*

But the bigger boost came with Internet of Things, where a lot of different objects and agents are supposed to cooperate and work autonomously. In this environment, Trust and Reputation mechanism are used in two different ways; security and awareness (decision making).

The first application where T&R mechanisms were used was in security. In IoT, from the security point of view, there are external and internal risks. For external risks or external attacks traditional security technics like authentication and authorization works well, but for internal risks: nodes being compromised, node faults, communication faults etc., it is necessary to adopt another mechanisms. In this sense, T&R technics offer one of the best ways to keep security levels and let the system works autonomously, isolating those nodes which present misbehavior. The misbehavior can be one of the following types:

- **Self-exclusion:** an agent not responding to requests because saving battery or resources;
- **Packet Dropping:** agents not retransmitting incoming packages;
- **Packet Modification:** agents modifying the information into retransmitted packages;
- **Timing attacks:** agents delaying responses or retransmitted packages;
- **Routing change:** agents modifying routes of retransmitted packages, routing them to circular loops or to non-existing routes.

T&R technics can also be used as a mean to give some kind of reasoning to an IoT system, if the T&R scores are based not in security aspects but in the result of tasks each agent perform when interacting with the rest of agents in the system, and each new interaction is based on the result of previous interactions. Thus T&R becomes a useful decision making procedure.

There are a lot of studies and experiences regarding Trust and Reputation into Internet of Things, we will try to outline some of the more relevant.

3.7.2. Trust Computation method using Fuzzy Logic (TCFL)

Proposed by Tae Kyung Kim and Hee Suk, it is mainly used in WSN, where a problem can be not only an attack or misbehavior but a problem due to ambient conditions, or other kind of external issues which introduce a level of uncertainty. In this context TCFL where firstly use to trace path to route messages.

It assigns the trust values to the nodes and then those trust values are used to find the trust values of the paths. And then the packets are forwarded along the highest trust value path. It ensures that the packets follow most trustworthy path from source to destination. Mostly in wireless sensor networks, a node is only interested in the trust values of its neighbor for multi-hop transmission nature and in that case, the centralized nature of this technique puts an overhead on the power consumption of a system.

This method is widely used in P2P networks.

3.7.3. Reputation-based Framework for Sensor Networks (RFSN)

RFSN has two main components called Watchdog and Reputation system. Watchdog monitors the actions of neighboring nodes and classifies the actions as cooperative or non-cooperative while reputation system is responsible for maintaining the reputation of a node. Distributed nature of a system makes it more suitable for practical applications.

RFSN is available as a middleware service on Motes. It is currently supported in two operating systems, TinyOS and SOS.

3.7.4. Agent-based Trust Model for sensor Networks (ATSN)

In ATSN, agent nodes monitor the behavior of sensor nodes and classify their behavior as bad behavior or good behavior. The advantage of ATSN is that it reduces the memory constraint and computational complexity on ordinary sensor nodes. But the assumption in ATSN that the agent nodes are not posed to security threats is unrealistic in practical Network applications.

3.7.5. Task-based Trust Framework for sensor Networks (TTSN)

TTSN is proposed in which Trust value is calculated depending on the different tasks which node executes to its different neighbors. TTSN uses Task and Trust Manager Module for building trust and the method is almost same as in RFSN but in TTSN, each sensor node has several values of trust. TTSN is more suitable for trust computation in large scale heterogeneous sensor networks. Again, we can extend this technique in context of our application. And it is more related because different Virtual Entities will interact with other entities in different context depending on the task. And which Virtual Entity will have to keep different Trust values for other Virtual Entity depending on the tasks.

3.7.6. Mobile ad-hoc networks (MANETs) and WSNs

MANET is network of self-configuring infrastructure less network of mobile devices connected by wireless, each device in a MANET is free to move independently in any direction, and will therefore change its links to other devices frequently.

MANETs are decentralized, so all network activities are carried out by nodes themselves. Each node is both and end-system as well as a relay node to forward packets for other nodes.

MANETs are highly preferred for connecting mobile devices quickly and spontaneously in emergency situations like rescue operations, disaster relief efforts or in other military operations. Nodes use to be individuals with any common interests. It may be advantageous for individual nodes not to cooperate. Hence, they need some kind of incentive and motivation to cooperate.

WSNs are networks of hundreds and thousands of small, low-power, low-cost devices or sensors. Its core application is to detect and report events. WSNs are used in critical applications in military and civilian life like robotic landmine detection, environmental monitoring, wildfire detection and traffic regulation.

In a WSN, all sensors belong to a group and work towards the same goal. Since WSNs are often deployed in unattended territories, sensors can be easily altered fraudulently. And third parties can access to the cryptographic material of gain access to the system. The latest approaches to cope with these problems come from the adoption of T&R technics.

3.7.7. Collaborative Reputation Mechanism to enforce node cooperation in MANETs (CORE)

This model was proposed by Michiardi and Molvato in order to enforce node cooperation in MANETs. CORE is a distributed and symmetric reputation model (distributed because each node stores reputation info about nodes it cares about; symmetric because all nodes have access to the same amount of information about reputation).

CORE works with three types of reputation information:

- Subjective reputation to talk about the reputation calculated directly from a subject's observation. It is evaluated only considering the direct interaction between a subject and its neighbors;
- Indirect reputation, which adds the possibility to reflect in the model a characteristic of complex societies, the final value given to the reputation of a subject is influenced also by information provided by other members of the community;
- Functional reputation: it is a global value of the reputation of an object. It is calculated based on the subjective and indirect reputation values.

CORE differentiates two types of misbehaviors, selfish and malicious behaviors although it focuses in selfish behaviors, and it is applied to Route Discovering and Packet Forwarding functions.

3.7.8. SocIoS

SocIoS (see http://www.sociosproject.eu/), a project related to social networks, offers a Reputation Service, whose main goal is to give to its users and developers a way to get reputation scores, based on aggregated relationships between the entities. In particular, the service can be used by SocIoS service creators in order to obtain reputation scores on potential users based on their social behavior and their social interaction with others. Similarly, a SocIoS user can ask for reputation scoring of SocIoS services, based on various metrics and qualities.

Reputation Service is built on top of IBM's *Social Network and Discovery Engine* (SaND). The latter is an aggregation tool over social media, which aggregates many kinds of relationships between its core entities – people, items and tags. The service will use SaND to aggregate information on SocIoS entities, most notably on users, content items, services and the interaction between them. The aggregated information will be used to calculate a wide range of reputation types, such as trustworthiness, popularity and influence.

The aforementioned scores can be used in various cases, like users' reliability check. For instance, the user with the highest reputation score will be rated as the most reliable source of information or content. As such, the main objective of the SocIoS Reputation Service is to obtain information about the activities of a group of *Social Networking Sites* (SNS) users and then to analyze this information, so as to be able to calculate their reliability as a closed group.

In addition, SocIoS project implements a Recommendation Service, which aims at giving to the users and developers a way to get a list of recommendations, also based on aggregated relationships between the entities. For example, the service can be used by SocIoS service creators to obtain a list of recommended users. This list can then be fed into the SocIoS Reputation Service as mentioned above. Similarly, a SocIoS user can ask for recommendations on services that suit his profile.

Some of the above techniques are summarized below.

Trust Mechanisms	Methodology	Limitations
TCFL	Fuzzy Logic	Centralized scheme, not suitable for distributed applications
RFSN	Probability Theory and Bayesian Network	Individual node security is improved but cannot improve system robustness
ATSN	Weighting; Probability Theory	Performance is highly vulnerable to Malicious Agent nodes
TTSN	Weighting; Bayes Theorem and Beta Distribution	Don't consider past observations

3.7.9. Conclusion

In the context of COSMOS, exploring social aspect of virtual entities is an important research goal. Most of the techniques discussed above explore the QoS metrics for evaluating Trust and Reputation. The importance of QoS metrics for evaluating trust cannot be ignored. In CORE and SocIoS, social perspective is mentioned but they were not directly related to our scenarios. We will extend the state of the art technologies and will use both social and QoS perspective to enhance the performance of virtual entities. Trust and reputation is rather new field in context of Virtual Entities and offers lot of research opportunities. We intend to use the existing concepts and then extend them to develop a novel task based Trust evaluation approach which considers both social and QoS metrics.

3.8 Autonomic Computing and Distributed Artificial Intelligence

3.8.1. MAPE-K

COSMOS initial concept is built to respect self-* capabilities related to healing, configuration optimization and protection in order to fulfill the autonomous social behavior of things. Such topic is of interest on at least two levels: one related to things (or their virtualization as agents) and services overlay responsible for social behavior orchestration.

The overall paradigm able to cover the needs of such environment is inspired by the research area of Autonomic Computing, which has greatly increased over the course of the last ten years the common understanding on how to realize systems with self-managing (covering all previous self-* phases) capabilities. The main steps of such feature pack are inspired in its high-level design by the MAPE-K loop, which is one key conceptual aspect of the Autonomic Computing field. The MAPE-K autonomic loop (Monitor, Analyze, Plan, Execute and Knowledge) represents a blueprint for the design of autonomic systems where a managed element is coordinated by a loop structured in 4 phases and a common knowledge.

Common known image of concept is depicted in Figure 14 below:

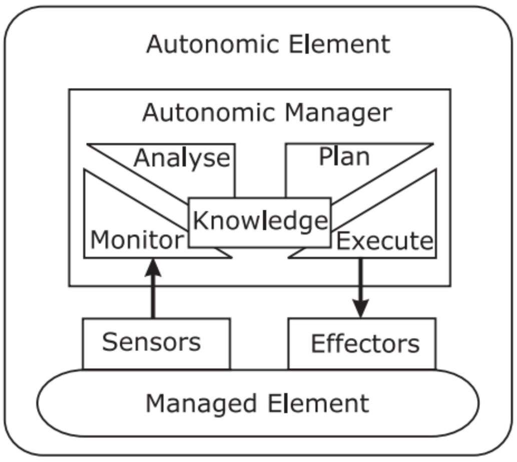

Figure 14: The MAPE-K Loop

The MAPE-K loop is structured in 4 correlated phases:

- **Monitoring:** The monitoring component is in charge to observe and manage the different sources of relevant data (named sensors here) that provide information regarding the way how system performs. In the COSMOS context for example, sensors can capture the current consumption of critical resources (such location calculation and memory) but also other performance metrics (such as the number of processed requests in a time window and the request process latency). The monitoring granularity is specified usually by rules. Sensors can also raise notifications when changes to the system configuration happen and a reaction is expected;
- **Analysis:** The analysis function is responsible for processing the information captured by the monitoring component and to generate high level events. For instance, it may combine the values of calls on a service and memory utilization to signal an overload condition in the platform;
- **Planning:** The planning component is responsible for selecting the actions that need to be applied in order to correct some deviation from the desired operational envelope.

The planning component relies on a high level policy that describes an adaptation plan for the system. These policies may be described, for example, using *Event Condition Action* (ECA) rules that are defined by a high level language. An ECA rule describes for a specific event and a given condition what action should be executed. In the context of COSMOS, the actions may affect the usage of Virtual Entities and the bindings among these ones in terms of use;

- **Execution:** The execution component applies the actions selected by the planning component to the target components.

Additionally, the **shared knowledge** includes information to support the remaining components. In the context of COSMOS, it maintains information about managed elements.

When systems are large, complex, and heterogeneous (the case of COSMOS), a single MAPE loop may not be sufficient for managing adaptation. In such cases, multiple MAPE loops may be employed that manage different parts of the system. In self-adaptive systems with multiple MAPE loops, the functions for monitoring, analyzing, planning and executing may be made by multiple components that coordinate with one another. Different patterns of interacting control loops have been used in practice by centralizing and decentralizing the functions of self-adaption in different ways. For example, in the Rainbow framework monitoring and execution are delegated to the different nodes of the controlled system, whereas analyzing and planning are centralized. The IBM architectural blueprint organizes MAPE loops hierarchically, where each level of the hierarchy contains instances of all four MAPE components. In this setting, higher level MAPE loops determine the set values for the subordinate MAPE loops. In fully decentralized settings, relatively independent MAPE components coordinate with one another and adapt the system when needed. The "On Patterns for Decentralized Control in Self-Adaptive Systems" paper presents a selection of MAPE patterns that model deferent types of interacting MAPE loops with different degrees of decentralization (like Coordinated Control Pattern, Information Sharing Pattern, Master/Slave Pattern, Regional Planning Pattern, and Hierarchical Control Pattern).

The application of the centralized control loop pattern to a large-scale software system may suffer from scalability problems. There is a pressing need for decentralized, but still manageable, efficient, and predictable techniques for constructing self-adaptive software systems. A major challenge is to accommodate a systematic engineering approach that integrates both control-loop approaches with decentralized agent inspired approaches.

Overall, a model of the MAPE management processes within the context of a generalized system management meta-model also developed within few relevant projects like Auto-I, ANA, or CASCADAS.

Of course that MAPE-K loop only represents a vision that leaves lower level details of the architecture purposely unspecified (i.e., they do not impose constraints on the implementation). COSMOS analysis of requirements should define a reference conceptual architecture for the runtime platform which we here describe and that follows the MAPE-K loop design approach. The details and implementation of this conceptual architecture will be specified more in the details in follow up deliverables. The only purpose of this section is to provide a high-level intuition of the systems that will compose the architecture, which is required in order to identify the actors that are involved in the requirement specification.

We have not found any available system or toolbox supporting the MAPE-K concepts.

3.8.2. Reasoning Techniques

In order to make Things able to select the actions that need to be applied to correct some deviation from their desired operational envelope, they have to be equipped with a functionality that enables them to reason. So, the first step in developing a planning component is to select a reasoning technique.

The main reasoning approaches in our domain are model-based reasoning (MBR), rule-based reasoning (RBR) and case-based reasoning (CBR). Many hybrid models can be developed by combining these approaches. Each approach has advantages and disadvantages, which are proved to be complementary in a large degree.

Having in mind the goals of the COSMOS project, it should be noted that the selection of a specific reasoning technique depends greatly on its capacity to define a kind of Experience and transferable knowledge.

3.8.2.1. Model-based reasoning

Model-based Reasoning (MBR) is an approach in which general knowledge is represented by formalizing the mathematical or physical relationships present in a problem domain. With this approach, the main focus of application development is developing the model. The creation of the model is the time consuming aspect of this approach, as it is necessary to make the model as deep, complex and detailed as possible to achieve the best results. Once a working model has been established, it may also require periodic updates. Then at run time, an "engine" combines this model knowledge with observed data to derive conclusions such as a diagnosis or a prediction. In other words, model-based reasoning is the use of a working model and accompanying real-world observations to draw conclusions.

One of the main advantages of MBR is that it enables the use of functional/structural knowledge of the domain in problem solving, increasing the reasoner’s ability to handle a variety of problems (including non-anticipated ones). Moreover, model-based reasoners are often built using scientific, theoretical knowledge. Obviously, this kind of knowledge is transferable between tasks, so it fills in Experience criteria set by the requirements of COSMOS. Moreover, MBR can provide causal explanations and generate state predictions.

However, MBR cannot be used when there is lack of experiential (descriptive) knowledge of the domain. Many domains, such as most design problems or financial applications, lack a well-defined scientific theory. Furthermore, a characteristic of MBR is its high complexity.

3.8.2.2. Rule-based reasoning (RBR)

A rule-based system may be viewed as consisting of three basic components: a set of rules (rule base), a data base (fact base) and an interpreter for the rules. In the simplest design, a rule is an ordered pair of symbol strings with a left-hand side and a right-hand side (LHS and RHS). The rule set has a predetermined, total ordering and the data base is simply a collection of symbols. The interpreter in this simple design operates by scanning the LHS of each rule until one is found that can be successfully matched against the data base. At that point, the symbols matched in the data base are replaced with those found in the RHS of the rule and scanning either continues with the next rule or begins again with the first. A rule can also be viewed as a simple conditional statement and the invocation of rules as a sequence of actions chained by modus ponens. In that sense, we can understand that rules usually represent general knowledge.

One of the main advantages of RBR is that it enables in a very direct fashion the use of experiential knowledge acquired from human experts. Moreover, the separation of knowledge from control simplifies development of expert systems by enabling an iterative development process where the knowledge engineer acquires, implements and tests individual rules. Finally, RBR provides good explanation facilities.

On the other hand, most of the time, the rules obtained from human experts are highly heuristic in nature and do not capture functional or model-based knowledge of the domain. Heuristic rules tend to be "brittle" and can have difficulty handling missing information or unexpected data values. The knowledge tends to be very task dependent. Formalized domain knowledge tends to be very specific in its applicability. Currently, knowledge representation languages do not approach the flexibility of human reasoning.

3.8.2.3. Case-based reasoning

Case-based Reasoning (CBR) is the process of solving problems based on past experience. The main process is trying to solve a Case (a formatted instance of a problem) by looking for similar Cases from the past and reusing the solutions of these Cases to solve the current one. Cases encompass knowledge accumulated from specific (specialized) situations.

The advantages of CBR include the ability to encode historical knowledge directly (obtaining Cases from existing Case histories, repair logs etc. eliminating the need for intensive knowledge acquisition with a human expert and for an extensive analysis of domain knowledge) and the fact that it allows reasoning shortcuts. Moreover, appropriate indexing strategies add insight and problem-solving power and predictions about how much the suggested solution may help are possible.

As CBR systems are used, the system encounters more problem situations and then, after creating more solutions and retaining those into the Case Base, the experience increases and the chance for better future solutions increases as well. It can be said that the more Cases we have in the Case Base the wider the future problems the system can try to solve and the better result the system can achieve.

However, Cases do not often include deeper knowledge of the domain and a large Case Base can suffer problems from store/compute trade-offs. Moreover, it is difficult to determine good criteria for indexing and matching Cases.

Indicatively, CBR reflects human reasoning and is the easiest approach for the developers and the one that can be used in a huge variety of domains. Moreover, the concept of adapting past solutions is one of the main requirements of COSMOS. In that sense, the Cases produced via CBR could become one kind of Experience for VEs, which can not only aid to the planning of future solutions, but can also be shared between different VEs. One the main disadvantages of CBR (store/compute trade-offs because of large Case bases) is tackled, as most of the VEs are going to have their own light-weight Case Base, as it is going to be described later.

Cases definition:

A Case is the description of a solving problem episode. Its structure fits the situation of the task: diagnostic, planning, decision helping, design etc. We can consider a Case as a list of (structured) descriptors. A Case is composed of a problem part and a solution part;

- **problem:** describes a problem or a situation. It can include not only the initial state of the system, but the desired final one too;
- **solution:** represents a possible solution approach.

A *Case Base* (CB) is a collection of solved Cases for a class of problems. For example, there are separate and different Case Bases for the "Bus diagnosis problem". For each Case Base, there is an associated metric allowing to project Cases on the "solution plan". Similar Cases are Cases that have similar solutions for similar problems.

The CBR Cycle:

In 1994, [Aamodt and Plaza] proposed a life cycle for CBR systems to make evident the knowledge engineering effort in CBR which is used by other CBR researchers as a framework. This cycle consists of four main parts: Retrieve, Reuse, Revise and Retain. Each of these parts includes a set of tasks and different methods have been proposed for each of them. In this section we provide an overview of these tasks and methods.

A brief presentation of the four phases of the CBR cycle (see Figure 15) follows:

1. **Retrieve**: Retrieve the most similar Case or Cases to the new one, in other words, given a target problem, retrieve from the memory Cases relevant to solving it. The procedure we follow for solving a problem, together with justifications for decisions made along the way, constitutes the retrieved Case. The retrieve step is the key step in CBR because the quality of the adaptation depends on the quality of the retrieval. The input to the retrieval task is the problem description and the output is the Cases that most closely match the new problem. Every retrieval method is a combination of a similarity assessment procedure, (which determines the similarity between a target Case and a Case in the CB) and a procedure for searching the Case memory to find the most similar Cases. Some of the COSMOS-candidate retrieval techniques are: *Similarity-based retrieval* (Simple flat memory k-Nearest Neighbor retrieval, Induction methods, Footprint retrieval, Fish & shrink retrieval, Validated retrieval etc.), *Adaptive-guided retrieval* (AGR) and *Ontology-based retrieval*;
2. **Reuse**: Reuse the retrieved Cases by copying them completely or by integrating the solutions of the Cases retrieved. Adaptation methods are: transformational / Structural adaptation, Substitution adaptation, Compositional adaptation, Derivational adaptation, Special purpose adaptation and repair;
3. **Revise**: Revise or adapt the solution(s) of the Cases retrieved trying to solve the new problem. Having mapped the previous solution to the target situation, test the new solution in the real world (or a simulation) and, if necessary, revise. In general, revision can be viewed as two tasks: diagnosis of failure and solution repair. For diagnosis, one of the following ways can be used: a) execution of the solution and evaluation of the outcome, b) simulation -model of the real world and evaluation of the solution using the model, c) help from the experts in order to diagnose the failure in solutions, d) use of the CB itself to identify the failure;
4. **Retain**: Retain the new solution once it has been proven that it is correct and brings successful results. Retain the parts of this experience likely to be useful for future problem

solving. Store new Case and found solution into Knowledge Base. In a CBR system learning is done in the retention step. Retention includes adding knowledge and new Cases to the Case Base, all which needs to be indexed, as well as deleting Cases from the Case Base in order to restrict its growth. Different retention (maintenance) strategies fall into one of two groups: a) maintenance of the content of the Case Base and b) maintenance of the organization of the Case Base. A part of the retention step is indexing, and it constitutes one of the main issues for efficient retrieval of Cases. Different types of indexing techniques are: difference-based techniques, Inductive learning methods, Explanation–based techniques, Similarity–based generalization, Dynamic indexing etc.

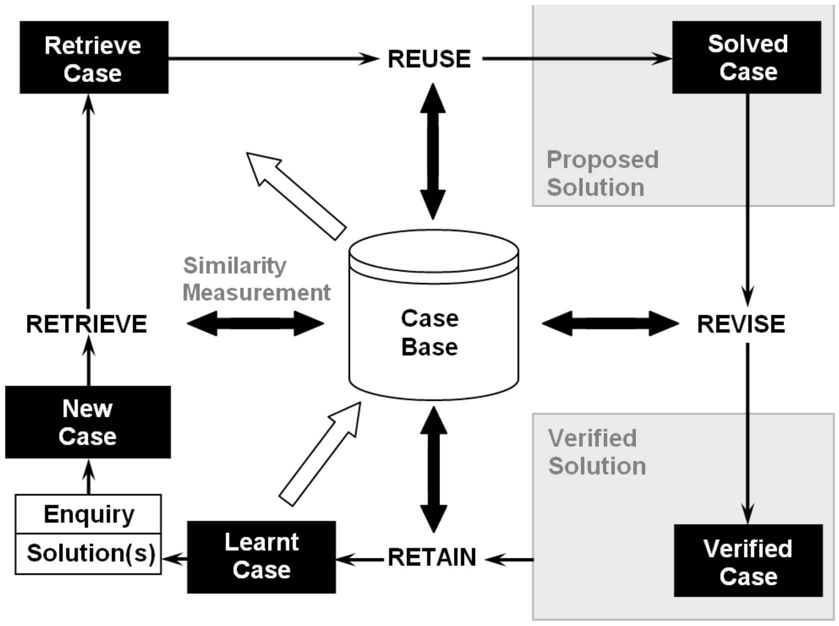

Figure 15: The CBR cycle.

Knowledge Representation and Case Memory Models in CBR:

In order to use the previous experiences in the CBR cycle, Cases must be represented in a structural manner. The simplest format to represent the Cases in the Case Base is to have simple feature-value vectors which are good for Cases with attributes of nominal or numeric values. In this kind of representation, no relationships between the attributes in Cases or between the Cases are shown and surface similarity-based retrievals can be used for retrieving from the Case Base. However, in some domains this is not the case and the attributes are complex, so the Cases are represented in the form of objects, predicates, semantic networks, scripts, frames, rules and concept maps. CBs can also be represented in the form of XML documents. Some other techniques of structural-based representation are: Object-oriented based, Spreading activation method, Generalized Cases, Graph-oriented and Ontology-based.

The Case Base should be organized in a manageable structure that can support efficient search and retrieval methods. Several groups of Case Memory Models have been proposed like: Flat Memory model, Hierarchy or Shared-Feature Network Memory model, Network Based Memory Models (Category exemplar model and Case retrieval nets).

jCOLIBRI – A CBR Reference Model:

The main goal of jCOLIBRI (http://gaia.fdi.ucm.es/research/colibri/jcolibri) is to provide a reference platform for developing CBR applications. It defines a clear architecture to design CBR systems plus a reference implementation of the components required to build them. This architecture has been designed to be extensible and reusable across different domains and CBR families. Although the main scope of the framework is the research community, a relevant feature of jCOLIBRI is its support for large scale commercial applications. As a result, the corresponding software can be used not only to do rapid prototyping of CBR systems but also for developing applications that will be deployed in real scenarios. This feature is exemplified, for example, by its straightforward integration into web environments.

3.8.3. Multi-Agents

This section deal with Agent technology and will succinctly describes different categories of agents and associated technologies. Before doing so, let's recall a tentative definition of what an agent is [Jennings-Wooldridge'1997], [Ferber'95] and is made of (tentative because there still exists various definition with no real reached consensus).

An agent is a virtual or real entity which:

- Enjoys an (often restricted) set of perceptions about its environment;
- Can react with its environment;
- Can build a (often partial) model of its environment;
- Is driven by incentives that can be the result of either an internal process involving own goals and believes or interactions with the environment;
- Enjoys a set of skills and offers eventually services;
- Enjoys often cognitive and reasoning capabilities;
- Behaves in such a way it tends to satisfy his incentives (wherever they come from) accordingly to its model of the world and set of believes;

It is generally admitted that agents behave autonomously.

Multi-agent System are therefore an eco-system of agents that are intended to behave socially (meant here, interacting with other agents and cooperating with each other. Multi-agents may have common goals but may also be driven by their own objectives, cooperating only when it happens that some individual goals require joint efforts in order to be reached, e.g. in an opportunistic way.

From this general definition, we can extract two main families of agents (regardless of mobility aspect which are discussed later in this section [Ferber 1995]):

- **Cognitive agents:** this first class of agents refers to the domain of Distributed Artificial Intelligence. Cognitive agents have their own goals, set of skills (knowhow) that tells it how to achieves tasks (explicit actuation plans e.g.), how to interact, to negotiate and to cooperate with peer agents. Two well know architectures for cognitive agents consist of the BDI architecture (standing for Believe-Desire-Intent) and MAPE-K model (standing for Monitoring/Analyze/Plan/Execute-Knowledge). The later one (described earlier in this section) though is more adapted to Cognitive agents that are more driven by their perceptions or said differently when agents' goals are directly derived from perception or from interactions with the environment;

- **Reactive agents:** reactive agents are not necessarily intelligent as such but have still clear plan about how to react to and handle internal and external stimuli. They are therefore not driven directly by goals and don't achieve planning or problem solving like Cognitive agent would do in order to fulfil their objectives.

3.8.4. BDI Agents

The BDI architecture is introduced by Rao and Georgeff [Rao-Georgeff'95], BDI agents enjoys three mental attitudes about Believes, Desires and Intent(ions) as a way to make separation between monitoring the environment and making decisions (creating plans) and executing plans. More precisely, a BDI agent is characterized by:

- **A set of Believes:** that represent the knowledge that the agent has about its environment, i.e. the knowledge that potentially influence its behavior;
- **A set of Desires:** That represent the motivational state of the system; the objectives of the agent (associated with Priorities and Payoffs) - the states the agent wants to reach- for which it needs to elaborate precise plans, based on the knowledge it has about its environment. There is a need to ensure that the agent's believes are not inconsistent or contradictory;
- **A set of Intentions:** that represents the deliberative state of the agents, i.e. its commitment (decision) into performing actions.

BDI architecture allows an agent to reconsider the course of its deliberation and planning according to the evolution of its knowledge, which means that at any time it might reconsider decision taken that become obsolete because of a change in its believes.

The BDI paradigm is formally described using a modal logic where relations between the three B, D and I are explicitly described and axiomatized.

3.8.5. JADE

Jade stands for Java Agent DEvelopment framework. Jade [Bellifemine et al', 2007] is a complete distributed middleware written in Java that provides:

- A complete framework for developing agent based application;
- Support for agent life cycle;
- Support for inter-agent high level communication capabilities (speech-act based)- called ACL (standing for Agent Communication Language) - the semantics of which is in addition formally described;
- Extensions with Add-on modules;
- Support to the core logic of agents;
- A rich suite of graphical tools.

In addition, Jade is fully FIPA (Foundation for Intelligent Physical Agents) specification compliant and implements the principles of the BDI architecture shortly described here after:

- *Agent Communication Language* (ACL), which is used for migrating agents too;
- *Agent Management System* (AMS) for managing the agents life cycle;
- *Agent Security Manager* (ASM) that maintain security policies for the platform and infrastructure;
- *Directory Facilitator* (DF) which is an agent registry;
- Yellow Page service (for look up) etc…

While FIPA is entirely focusing on Multi-Agent Systems (meant here stationary Intelligent Agents) JADE still offers agent mobility through Java object serialization.

3.8.6. Mobile Agents

Mobile agents are a distributed computing paradigm -introduced in the early 90s- where, generally low-weighted, software components have the ability to migrate from places to places in order to execute locally – in distributed nodes- their program logic. Migration of an agent from node A to B usually means suspending execution of the agent logic in A, migrating “physically” the code within the network along a determined route leading from A to B, and resuming the agent execution in B at the exact instruction it was suspended in A. Not all implementations of the mobile agent paradigm do respect this later characteristic above. Mobile agent are autonomous (like the name agent suggests), therefore they are deciding themselves about program logic execution and migration; no extra communication which a tier- entity whatsoever is needed.

There are multiple advantages in using Mobile Agent, but those advantages are often debated - with passion - within the Distributed Artificial Intelligence community which generally tend to argue that most of the Mobile Agent benefits could be easily implemented with Multi-Agents. We describe hereafter some of the generally admitted benefits of using mobile agent paradigm:

- **Bandwidth saving / load balancing:** in distributed architecture with a central component (a manager e.g.) and potential high number of distributed components, a large number of data transfer from the remote components to the central entity results in network congestion. Having some code migrating from the central element and traveling the different remote n/w elements in order to perform local computation (or collecting digested information e.g.) allows to save lot of bandwidth and to avoid congestion. The nature of the computation, migration strategies and number of agent involved depends on the scenario;
- **Flexibility:** Mobile agents can be deployed on demand and depending on the context in order to perform remotely specific tasks. As a result, the various remote elements do not need a large number of embedded capabilities; on the contrary they are able to host and execute mobile code which is relevant according to the current context.
- **Re-configurability:** similarly to flexibility, mobile agents can be used to (re)configure nodes’ behaviour without prior configuration of the node and for augmenting the nodes’ capabilities on-the-fly;

For more characteristics about mobile agents refer to [Lange & Oshima’99]

It can be seen from above that a high interest in using Mobile Agent is the ability to deploy the right code at the right place at the right time, and to rely on roaming agents for performing task on another entities behalf at various places following a predetermine or agent-determined itinerary. This ability comes with a cost:

1. An agent platform needs to be deployed at any potentially visited nodes (then and only then a place can be "visited" by an agent for execution of its tasks);
2. CPU load overhead due to 1;
3. Security issues: as soon as a platform is in place it is ready to host any compliant agents, including malicious ones.

Finally typical usage of mobile agents in the IoT domain comprises:

- **Data fusion:** different works use single MA visiting all nodes, multiple Mas working in parallel or multiple MAs in clustered WSN and focusing within the clusters only, coming with a variety of solutions. [Mpitziopoulos *et al*.] provide a comprehensive survey of those existing techniques;
- **Diagnosis in large Distributed Systems:** Alcatel has conducted research in the late 90's about fault diagnosis in GSM networks. In that particular case mobile agents were visiting various telecommunication devices (Base Stations, HLR's, OMC's,…) checking and correlating data and searching for the probable cause of a fault; The mobile agent was in fact a mobile Rule Based System written in Java (using therefore the rule engine Jess);
- **Highway traffic detection for intelligent Transportation Systems** using both stationary and mobile agents. The stationary agents are located at the traffic stations while mobile agents navigates between stations in order to achieve data fusion [Chen et al.'2006]

The following paragraphs present few implementations of mobile agent and their main characteristics.

3.8.7. Mobile C

Mobile C [Chen et al, 2006] is a mobile agent platform (language and execution environment) that primarily targets embedded systems and networked mechatronics. It therefore uses C/C++ as agent language and includes an embeddable interpreter. Mobile C is FIPA[4] compliant so in addition to being a mobile agent environment, it also enjoys many features that any FIPA-compliant Multi-Agent System must enjoy either at agent or platform levels, as described in 3.6.4

3.8.8. Conclusion

MAPE-K and BDI agents are very interesting paradigms that however fit quite distinct classes of systems. While MAKE-K is definitely driven by "perceptions", meant here the monitoring phase that analyze the environment in order to trigger some actions (Reactive Model), BDI is getting its incentives through achieving goals (D of BDI stands for Desires –or goals) and then maintaining their perception of the world through local perceptions. Based on this model and their Desires, BDI agents will plan and decide which action (Intentions) to undertake to reach their objectives. We have considered that in a first approach, MAKE-K fits better the needs and vision of the project but we will investigate further the possibility of including few aspects of BDI within MAKE-K in order to make goals more explicit.

[4] Foundation for Intelligent Physical Agent (Multi-Agent IEEE standard)

3.9 Run-time models for Self- systems

A runtime model is a causally connected representation of a software system under operation, i.e. changes in the system are propagated to the model and vice versa. Changes in the state, structure, or behavior of primary system are automatically propagated via monitoring into the runtime model, and changes to the runtime model. [Oreizy et al.] have used a runtime model, that reflects the current structural state of the system architecture. An architectural reconfiguration operation (e.g., adding or removing software components) is first performed on the runtime model, and afterwards this change is propagated into the primary system. The aim was to allow system maintenance without the need to restart the system. [Garlan et al.] extended [Oreizy et al.]'s approach by using the runtime model as a simplified (role-based) abstraction of the system, which is used for constraint checking, diagnosis, and reconfiguration. They model mainly the structural view for representing the primary system. Failure detection is implemented by verification of constraints defined in the ADL Armani. The general advantages of using a simplified runtime model for self-management are (1.) that it simplifies constraint checking, diagnosis, and the selection of repair operations, and (2.) the decoupling of self-management from the primary software system allows to develop reusable self-management strategies independently from a concrete software architecture. Existing constraint languages such as Armani or the *Object Constraint Language* (OCL) are very helpful. Armani focuses on the description of system structure and how that system structure may evolve over time rather than the dynamic run-time behavior of systems.

3.10 Social Computational Models and Social Network Tools

COSMOS will collect, aggregate and distribute monitoring data (events) across the decision making components of the collaborating groups. The events will be generated by interactions in response to - directly or indirectly - user actions (e.g. registering a new VE) or VEs' actions. For each and every interaction we can define a corresponding social characteristic.

The events that are generated at the social monitoring level can be evaluated at different platform levels (node level, group level or system wide) against a set of rules. The evaluation results can be used by the Planner or other COSMOS-components and can be used as a form of service for the users.

The social network perspective provides a set of methods for analysing the structures of whole social entities and their networks. For the study of these structures, COSMOS can use Social Network Analysis (SNA) to identify local and global patterns, locate influential entities and examine network dynamics. Social network analysis is the analysis of social networks viewing social relationships in terms of network theory. These relationships are represented by nodes (representing individual actors within the network) and ties (which represent relationships between the individuals such as friendship, similarity etc.). These networks are often depicted in a social network diagram, where nodes are represented as points and ties are represented as lines.

Various theoretical frameworks have been imported for the use of social network analysis. The most prominent of these are Graph theory, Balance theory, Social comparison theory and, more recently, the Social identity approach. Few complete theories have been produced from social network analysis (e.g. the Structural Role Theory and Heterophily Theory). There is a great variety of metrics that could be used under the functionalities of the Social Analysis components, offering more detail and information about the networks being analyzed. Such metrics are: Homophily/Assortativity, Mutuality/Reciprocity, Propinquity, Network Closure, Distance, Multiplexity, Structural holes, Bridge, Centrality, Density, Clustering coefficient, Cohesion, etc.

There are many tools that can be used and a need for the development of new tools may exist. Social network and dynamic network metrics, trail metrics, procedures for grouping nodes, identifying local patterns, distance based, algorithmic and statistical procedures for comparing and contrasting networks, groups and individuals from a dynamic meta-network perspective, geo-spatial network metrics, identification of key players, groups and vulnerabilities, are but a few issues that have to be addressed.

Some of the main tools that have been studied and could be exploited by COSMOS are:

- **Representation Formats, mark-up languages and ontologies:** DyNetML and GraphML could be used as a reference model;
- **Dynamic Network Analysis and Social Network Analysis:** Tools for reasoning under varying levels of uncertainty about dynamic networked and cellular organizations, their vulnerabilities and their ability to reconstitute themselves, choosing Dynamic Network Analysis (DNA) metrics and then using one or more of the available optimizers to find a design that more closely meets an ideal as well as exploring network graphs. Some tools that have been studied are Dynet, EgoNet, NetMiner, SIENA and SNA;
- **Network Visualization:** Tools for graph visualization and representation like Zoomgraph and GraphViz;
- **Network Document Analysis and Data Entry**: Tools that enable the extraction of information from texts using Network Text Analysis methods and other techniques. Such tools that have been studied are SocIoS, AutoMap and ORA.

3.11 Handling Change and Reconfiguration

Changes are the cause of adaptation. Whenever the system's context changes the system has to decide whether it needs to adapt. Some modeling methods for runtime reconfiguration are:

3.11.1. Graph based modeling

Graphical model is a probabilistic model for which a graph denotes the conditional dependence structure between random variables. They are commonly used in probability theory, statistics—particularly Bayesian statistics—and machine learning. In particular, nodes in graphical models represent random variables and the edges represent conditional dependence assumptions.

Graph based modeling can be used as a basis for reconfiguration. Graph based modeling is a very broad definition and can refer to a couple of things. The common aspects of these things is that the knowledge that is encoded can be captured in a graph of vertices (objects) connected via edges (links). Although a simplistic way of encoding knowledge, it can be quite expressive and often very intuitive for people.

On way of using graphs to model reconfiguration is by using it to create a state-transition graph. When each configuration of a system is considered as a state, each act of reconfiguration becomes a transition, thus describing a reconfigurable system as a state-transition graph. The edges in the graphs would then be assigned conditions under which the represented reconfiguration would take place. A system employing a state-transition graph as its way of determining how it reacts to the environment is commonly known as a finite state machine. In a finite state machine, the system is always in one state, and will "hop" to another state occasionally. For the reconfiguration task, the conditions under which it hops, should be defined quite explicitly and are what give the system its flexibility. The states on the other hand are defined only by the task itself. They could define a certain strategy for performing a task, or a set of parameters to operate with. In any case should the In any case should the states contain a complete description of the configuration since only one state is active at any moment.

3.11.2. Constraint-based description

Constraint programming is a programming paradigm wherein relations between variables are stated in the form of constraints. Constraint satisfaction programming can be utilized for searching in a (large) set of configuration options, and finding the one(s) that satisfy a given set of constraints set by the user. In the standard constraint satisfaction problem, a value needs to be assigned to a variable, from a finite domain of possibilities. More flexible interpretations also exists in which not all variables need to be assigned values, or a solution can be found by breaking the least number of constraints. Other variants (dynamic constraint satisfaction problems) can decrease the effort of finding a fitting solution by utilizing a previous solution in order to find a new one in a different environment [Verfaillie, 1994].
OCL is a declarative language for describing rules that apply to *Unified Modeling Language* (UML) models developed at IBM and now part of the UML standard. Initially, OCL was only a formal specification language extension to UML. OCL may now be used with any *Meta-Object Facility* (MOF) *Object Management Group* (OMG) meta-model, including UML. The Object Constraint Language is a precise text language that provides constraint and

object query expressions on any MOF model or meta-model that cannot otherwise be expressed by diagrammatic notation. OCL is a key component of the new OMG standard recommendation for transforming models, the Queries/ Views/ Transformations (QVT) specification.

In a reconfiguration framework however, OCL can be used to specify the constraints of the runtime configuration. Since OCL is a part of the UML modeling standard there are a couple of advantages. It is very easy to cooperate with multiple partners. There exist many editors and/or interpreters for OCL that use exactly the same syntax. Another advantage is that it is very applicable to models of systems that are created in UML, which is a very popular way of modeling systems. Thirdly, because OCL can be applied to a model of a system, it is possible to design the constraints and see their effects even at the modeling phase of the design process.

3.11.3. Logic based description

A logic framework combines the semantics of constraint based approaches and rule-based approaches. The configuration objects can be interpreted as atoms and the requirements and constraints are modeled as sets of inference rules. The problem of finding a configuration under specific circumstances then becomes equivalent to finding the atoms that still hold under the given assumptions. The simplest form of a logic reasoner is a first order logic interpreter. In first order logic, a statement is either true or false. Using operators such as the NOT, AND and OR operators, relations between statements can be described. In order to describe reconfiguration knowledge using first order logic, a list of constraints and conditions should be described in a list of statements which should hold true in the framework.

Another type of logical reasoner is a fuzzy logic reasoner. In fuzzy logics a variable can have intermediate values, rather than only true or false. Instead a variable can be partially true and partially false, and can have string values. The difference between fuzzy logics and probabilistic theory is that in fuzzy logics one can determine “how much” a variable belongs to a set, whereas in probabilities one determines the chance that is either is in a set or not.

For the reconfiguration framework, a rule base based on fuzzy logics would have more or less the same form as the first-order logic rule base. The main difference is that now the variables are no longer binary, but may have intermediate values, and therefore multiple rules may be true at the same time. A proper fuzzy logics interpreter would then be able to determine the configuration which is most fit for a certain case by combining multiple statements.

3.11.4. Fuzzy Logic Device (FLD)

Zadeh’s conclusions suggested using a fuzzy rule-based (human reasoning-based) approach to the analysis of complex systems and provided a decision-making procedure together with a mathematical tool. In general fuzzy systems are knowledge-based systems that can be built up from expert operator criteria and can be considered universal approximators where input/output mapping is deterministic, time invariant and nonlinear.

The *Fuzzy Logic Device* (FLD) is a general concept in which a deterministic output (crisp values) is the result of the mapping of deterministic inputs, starting from a set of rules relating linguistic variables to one another using fuzzy logic. For the mapping to be performed, deterministic values are converted into fuzzy values, and vice-versa. A FLD is made up of four functional blocks: fuzzification, the knowledge base, decision-making and defuzzification.

3.12 Modeling Languages

3.12.1. Data Model for representing things and their meta-data structure

3.12.1.1. JSON

JSON (JavaScript Object Notation) (see http://json.org/) is a lightweight data-interchange format. It is easy for humans to read and write and also for machines to parse and generate. JSON was derived from the ECMAScript Programming Language Standard (see http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf), in addition JSON defines a small set of formatting rules for the portable representation of structured data.

JSON is built on two structures:

- A collection of name/value pairs. In various languages, this is realized as an object, record, structure, dictionary, hash table, keyed list, or associative array;
- An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence.

These are universal data structures. Virtually all modern programming languages support them in one form or another. It makes sense that a data format that is interchangeable with programming languages also be based on these structures.

Except for the two structured types (objects and arrays), JSON can also represent four primitive types (strings, numbers, Booleans and null). It is used primarily to transmit data between a server and web application. The official Internet media type for JSON is application/json and its filename extension is .json

3.12.1.2. JSON Schema

JSON Schema is a draft by IETF (currently in v4) that allows defining the structure of JSON statements for verification. It also proposes ways to constraint specific structures, to match regular expressions, to define access methods to specific attributes and to specify if attributes are required or optional. `For more details about the specification and possibilities please refer to http://json-schema.org/documentation.html

3.12.1.3. XML

eXtensible Markup Language (XML) (see http://www.w3.org/XML/) is a simple, very flexible text format derived from the *Standard Generalized Markup Language* (SGML) (ISO 8879) (see http://www.w3.org/TR/REC-xml/). Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere.

XML describes a class of data objects called XML documents and partially describes the behavior of computer programs which process them. XML is an application profile or restricted form of SGML. By construction, XML documents are conforming SGML documents.

XML documents are made up of storage units called entities, which contain either parsed or unparsed data. Parsed data is made up of characters, some of which form character data, and some of which form markup. Markup encodes a description of the document's storage layout and logical structure. XML provides a mechanism to impose constraints on the storage layout and logical structure. XML filename extension is .xml.

The design goals for XML are:

- XML shall be straightforwardly usable over the Internet.
- XML shall support a wide variety of applications.
- XML shall be compatible with SGML.
- It shall be easy to write programs which process XML documents.
- The number of optional features in XML is to be kept to the absolute minimum, ideally zero.
- XML documents should be human-legible and reasonably clear.
- The XML design should be prepared quickly.
- The design of XML shall be formal and concise.
- XML documents shall be easy to create.
- Terseness in XML markup is of minimal importance.

3.12.2. Things semantics and semantic languages for annotation / Meta-data

3.12.2.1. RDF

The World Wide Web was initially built for humans as their users. Although all the data available is in machine readable form but not in machine understandable for. The data can be made machine understandable by the use of meta-data. Meta data is a data about data. It is used to describe the data to make it understandable to machines.

Resource Description Framework (RDF) is a metadata format to represent the data. The basic model consists of three object types.

The basic data model consists of three object types:

1) A **Resource** is anything that can have a URI; this includes all Web's pages, as well as individual elements of an XML document.
2) A **Property** is a Resource that has a name and can be used as a property, for example Author or Title. In many cases, all we really care about is the name; but a Property needs to be a resource so that it can have its own properties.
3) A **Statement** consists of the combination of a *Resource*, a *Property*, and a *value*.

These parts are known as the *'subject'*, *'predicate'* and *'object'* of a Statement.

It offers a simple graph model consisting of nodes (i.e. Resources) and binary relations (i.e. statements). It is a type of semantic network which embodies small amount of built-in semantics and offers great freedom in creating customized extensions.

RDF can be serialized in XML and also in JSON (see https://dvcs.w3.org/hg/rdf/raw-file/default/rdf-json/index.html).

3.12.2.2. SPARQL

RDF provides the foundation for publishing and linking your data. Most of the data on web is represented using RDF and it needs its own RDF-specific query language and facilities. SPARQL is the query language used for accessing RDF data. SPARQL can use HTTP or SOAP to send queries and receive results.

Using SPARQL consumers of the Web of Data can extract possibly complex information (i.e., existing resource references and their relationships) which are returned, for example, in a table format. This table can be incorporated into another Web page; using this approach SPARQL provides a powerful tool to build, for example, complex mash-up sites or search engines that include data stemming from the Semantic Web.

3.12.2.3. Notation 3

Notation 3 (N3) is a light-weight version of XML/RDF (meant here much more compact than RDF (RDF is indeed very verbose)) with high focus on human readability. N3 statements still follow the RDF structure of Subjects, Verbs (or Predicates in RDF) and Objects but N3 offers an easy and readable way to construct triples. N3 is considered having a higher expressive power than pure RDF, consequently not any N3 statement can be serialized in RDF/XML.

3.12.2.4. Turtle

Turtle is defined as a sub-set of N3 developed by Dave Beckett that can serialize RDF graphs while N3 cannot (as more expressive than RDF/XML). Turtle statements are also more user-readable than RDF. Some RDF Toolkits propose parsing and serializing capabilities for Turtle statements. Among them we can find for instance Jena, RDFlib and SESAME. The specification of Turtle can be found @ http://en.wikipedia.org/wiki/Turtle (syntax).

OWL is an abbreviation of *Web Ontology Language*. It is an extension of RDFS by laying more stress on the support for richer logical inference. It was designed to be interpreted by computers and used for processing information on the web. It is not intended to be read by people but by machines. OWL is written in XML and it is W3C standard. There are three variants of OWL on the basis of computational complexity and expressiveness of ontology constructs.

- **OWL-Lite:** OWL-Lite is the simplest variant and provides supports to applications needing a classification hierarchy with simple constraints. It does not use the entire OWL vocabulary and belongs to lower complexity class then OWL DL. It does not use entire vocabulary of OWL. OWL-Lite offers simple tool supports for developers as compared to more expressive variants of OWL;
- **OWL-DL:** OWL-DL is based on Description Logics, and focuses on common formal semantics and inference decidability. It is intended for the users who need maximum expressiveness with the assurance that all conclusions will be computable and all computations to be finished in finite time. It offers more ontology constructs like conjunction and negation in addition to class and relation with important inference mechanisms such as subsumption and consistency. OWL-DL includes all OWL language constructs but with some restrictions. For example a class can be a subclass of more than one classes but it cannot be an instance of any other class;

- **OWL-Full:** OWL-FULL offers the most expressive version of OWL-FULL but it does not provide assurance that the computations will be finite in time or will return definite conclusions (in other term it is not decidable). Class space and instance space are not disjoints in OWL-Full as opposed to OWL-DL. For example a class in OWL-Full can be treated as a collection of individuals and as an individual on its own at the same time. It provides all the features of OWL language without any restrictions.
-

3.12.2.5. Sesame

OpenRDF Sesame is a de-facto standard framework for processing RDF data. This includes parsers, storage solutions (RDF databases a.k.a. triple stores), reasoning and querying, using the SPARQL query language. It offers a flexible and easy to use Java API that can be connected to all leading RDF storage solutions.

3.12.2.6. Jena

Jena (see http://jena.apache.org/getting_started/index.html) is a set of Java APIs that can be used in order to build RDF, to serialize the obtained model in XML or Turtle, to read RDF/XML into a model (RDF graph), to navigate a model (properties of objects e.g.) and to query a model. Jena does not provide any storage facility (triple store) but works with:

- SDB (persistent triple store using relational database) distributed by Apache (see http://jena.apache.org/documentation/sdb/);
- Jena-Fuseki: is a SPARQL server over HTTP;
- TDB which is a storage component for Jena by Apache (again, see http://jena.apache.org/documentation/tdb/).

3.12.2.7. 4store

According to www.4store.org, "4store is a database storage and query engine that holds RDF data. It has been used by Garlik as their primary RDF platform for three years, and has proved itself to be robust and secure. 4store's main strengths are its performance, scalability and stability. It does not provide many features over and above RDF storage and SPARQL queries, but if you are looking for a scalable, secure, fast and efficient RDF store, then 4store should be on your shortlist".

4store is optimized to run on clusters of up to 32 nodes, linked with gigabit Ethernet, but could also run on Mac OSX single machine at the condition they have the Avahi Multicast DNS library. Import performance on a cluster is up to 120K Triple per second. Query time is in the low millisecond even over SPARQL.

3.12.2.8. RDF Triple Store comparison

The following table (Source: www.garshol.priv.no/blog/231.html) provides an extensive list of Triple Stores with SPARQL support with associated characteristics like capacity, performance, and licensing and cost aspects. Jena is not in this list, as Jena does not provide a persistent storage whatsoever. However Fuseki, which is listed below, works with Jena. Other modules provided by Apache are also working with Jena.

Requirement	Virtuoso	Oracle	OWLIM	Allegro	Bigdata	Mulgara	4Store	Sesame	Stardog	B*	DB2	Fuseki
Open source	Yes/no	No	No	No	Yes	Yes	Yes	Yes	No	No	No	Yes
Free edition	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
10 billion statements	Yes	Yes	Yes	Yes	Maybe	No	May be	No	Yes	Coming	?	No
Clustering	Yes	Yes	Yes	Yes	Yes	No	Yes	No	Coming	Cloud	Yes	No
SPARQL 1.0	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SPARQL 1.1	Partial	Yes	Yes	Yes	Yes	Partial	Partial	Partial	Yes	Yes	Partial	Yes
SPARQL Update	Non-std	Yes	Yes	Yes	Yes	TQL Upd	Yes	Yes	Yes	Yes	No	Yes
Support	Yes	Yes	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Yes	Yes
Events	Yes	Yes	Yes	No	No	No	No	Yes	No	No	Yes	Yes/no
Reasoning	Rules	Materialized	Rules	Rules	Datalog	Rules	Add-on	Little	OWL + rules	No	?	Rules
Constraints	No	Yes	No	No	No	No	No	No	Yes	No	No	No
Triple-level security	Coming	Yes	No	Some	No	No	No	No	No	No	No	No
Endpoint built in	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes
Live backup	Yes	Yes	Yes	Yes	?	Yes	Yes	Kind of	Kind of	Yes	Yes	Yes
Embeddable	Yes	No	Yes	?	?	Yes	Yes	Yes	Yes	Yes	No	Yes

3.12.3. Definition and Management of ontology rules

3.12.3.1. Rules Languages

The definition of rules plays an important role in the processes of the Semantic Web inference: the success for the generation of new knowledge depends of the success in the definition of new rules. Some of the most popular rules languages:

- **RuleML**: The **Rule Markup Language** (**RuleML**) is a markup language developed to express both forward (bottom-up) and backward (top-down) rules in XML for deduction, rewriting, and further inferential-transformational tasks. It's considered to be a markup language for the Semantic Web. RuleML covers the entire rule spectrum, from derivation rules to transformation rules to reaction rules. RuleML can thus specify queries and inferences in Web ontologies, mappings between Web ontologies, and dynamic Web behaviors of workflows, services, and agents. The disadvantage of using RuleML is that still is not a standard;

- **SWRL** (**Semantic Web Rule Language**): it is an expressive OWL-based rule language. SWRL allows users to write rules that can be expressed in terms of OWL concepts, to provide more powerful deductive reasoning capabilities than OWL alone. It extends the set of OWL axioms with Horn-like rules to enrich OWL ontologies. Semantically, SWRL is built on the same description logic foundation as OWL and provides similar strong formal guarantees when performing inference. Rules are of the form of an implication between an antecedent (body) and consequent (head);

- **Jess**: it is a rule engine for the Java platform. The language provides rule-based programming for the automation of an expert system. Rules can modify the collection of facts, or can execute any Java code. JessTab is a plug-in for Protégé that allows the user to use Jess and Protégé together. Protégé is a free, open source ontology editor and a knowledge acquisition system. Like Eclipse, Protégé is a framework for which various other projects suggest plugins. This application is written in Java and heavily uses Swing to create the rather complex user interface.

3.12.3.2. Rules management

In order to manage rules we can use semantic reasoners, rules engines and reasoning algorithms.

Ontology Reasoners

Reasoners can be classified into two groups: semantic reasoners and logic programming reasoners. Semantic reasoners are also known as reasoning engines, because they use an inference engine to infer or deduce logical consequences from a set of facts or axioms. They are called semantic because they use a semantic language for reasoning or inference (OWL). OWL axioms infer new knowledge through language itself. Apart from infer new knowledge, the reasoners are also used to validate ontology. The logic reasoners perform standard inference through languages like RDFS and OWL. They can represent a knowledge base that describes a particular domain. This domain is represented by classes (concepts), individuals and roles (properties). An OWL data store contains different constructs to create a formal representation of knowledge. OWL, in the majority of the cases, is restricted to some form of logic such as Description Logics (DL) in order to make it decidable. This means that when DL is enforced, a so-called DL-reasoner (e.g. Pellet) can infer new information from the ontology.
Pellet is an OWL-DL reasoner based on the tableaux algorithms developed for expressive description logics. It supports the full expressivity OWL-DL including reasoning about nominals (enumerated classes). Therefore, OWL constructs owl:oneOf and owl:hasValue can be used freely. Pellet ensures soundness and completeness by incorporating the recently developed decision procedure for SHOIQ (the expressivity of OWL-DL plus qualified cardinality restrictions in DL terminology).

Hoolet is an implementation of an OWL-DL reasoner that uses a first-order prover supports SWRL.

FaCT++ is a description logic reasoner implemented in C++ with free distribution license. It is an evolution of FaCT descriptive logic reasoner, originally implemented in LISP. FaCT supports SHOIQ (D) logic and use Tableaux algorithms to perform inference. It allows developing new features and optimizations in a personalized way, so it’s possible to add new tactics of reasoning.

FuzzyDL is a free Java/C++ based reasoner for fuzzy SHIF with concrete fuzzy concepts (explicit definition of fuzzy sets + modifiers). It implements a tableau + Mixed Integer Linear Programming optimization decision procedure to compute the maximal degree of subsumption and instance checking w.r.t. a general TBox and Abox. It supports Zadeh semantics, Lukasiewicz semantics and is backward compatible with classical description logic reasoning.

Cwm is a forward-chaining reasoner used for querying, checking, transforming and filtering information. Its core language is RDF, extended to include rules, and it uses RDF/XML or N3 serializations as required. (CWM, W3C software license)

Rules Engines

These systems are based on initial information and a set of rules, detect which of these rules should be applied at a given time and what results from its application. They describe the standards, operations, definitions, policies and constraints of a particular environment. There are semantic rules engines that use semantic languages as OWL, and rules engines that use no semantic languages.
Drools is a forward-chaining inference-based rules engine which uses an enhanced implementation of the Rete algorithm. Drools support the JSR-94 standard for its business rule engine and enterprise framework for the construction, maintenance, and enforcement of business policies in an organization, application, or service. Drools platform has been employed as the core context reasoning mechanism of **Hydra** Project.
The **Rete** algorithm provides a generalized logical description of an implementation of functionality responsible for matching data tuples ("facts") against productions ("rules") in a pattern-matching production system (a category of rule engine). A production consists of one or more conditions and a set of actions which may be undertaken for each complete set of facts that match the conditions. Conditions test fact attributes, including fact type specifiers/identifiers. The Rete algorithm is widely used to implement matching functionality within pattern-matching engines that exploit a match-resolve-act cycle to support forward chaining and inferencing. It has the following major characteristics:

- It reduces or eliminates certain types of redundancy through the use of node sharing;
- It stores partial matches when performing joins between different fact types. This, in turn, allows production systems to avoid complete re-evaluation of all facts each time changes are made to the production system's working memory. Instead, the production system needs only to evaluate the changes (deltas) to working memory;
- It allows for efficient removal of memory elements when facts are retracted from working memory;
- It provides a means for many-many matching, an important feature when many or all possible solutions in a search network must be found.

Bossam (software) is a Rete-based rule engine with native supports for reasoning over OWL ontologies, SWRL rules, and RuleML rules.
Cyc inference **engine** is a forward and backward chaining inference engine with numerous specialized modules for high-order logic.

Prova is an open-source semantic-web rule engine which supports data integration via SPARQL queries and type systems (RDFS, OWL ontologies as type system). (Prova, GNU GPL v2, commercial option available)
JEOPS (Java Embedded Object Production System) is a Java based forward chaining rule engine that is used to power up the business process by rules in Java.

Other reasoning tools

KAON2, an infrastructure for managing OWL-DL, SWRL, and F-Logic ontologies.
Flora-2, an object-oriented, rule-based knowledge-representation and reasoning system. (Flora-2, Apache 2.0)
Jena (framework), an open-source semantic-web framework for Java which includes a number of different semantic-reasoning modules. (Apache Jena, Apache License 2.0)
SweetRules, a powerful integrated set of tools for semantic web rules and ontologies, revolving around the RuleML (Rule Markup/Modeling Language) emerging standard for semantic web rules, and supporting also the closely related SWRL (Semantic Web Rule Language), along with the OWL standard for semantic web ontologies, which in turn use XML and, optionally, RDF.

Tools for exploring Ontologies

Ontology editors like Apollo, OntoStudio, Protégé, Swoop and TopBraid Composer Free Edition are used for building a new ontology either from scratch or by reusing existing ontologies, which usually supports editing browsing, documentation, export and import from difference formats, views and libraries. They may have attached interference engines, include support for some programming language etc. Protégé is a system used to generate ontologies. It provides capability for specifying logical relationships between classes and individuals and for generating and debugging ontologies and translation into several base notations.
The main advantages of Protégé are that:

- It is easy and understandable enough for the domain expert to use it to develop his ontologies of interest;
- It is an adaptable tool, which we can tune to support new languages and formalisms quickly. This is important as on the one hand, a number of new semantic-web languages and representation formalisms are emerging, but on the other, there is no agreement made yet;
- It can be used for the development and management of ontologies and applications today without waiting for standards;
- The supported model is an open and extensible one, allowing for plug-ins serving specific purposes.

3.13 Cloud storage and Meta-data

Cloud Storage refers to a virtualized entity of networked storage that is available online and accessed via web services. The storage facility is highly scalable and is hosted on a variety of servers, typically in multiple data centers that are geographically dispersed. The storage and data services can be offered by third parties as an *Infrastructure as a Service* (IaaS) in a "pay per use" model. More recently enterprises have deployed private clouds where such an infrastructure is both hosted and used internally behind a firewall, and community clouds offering similar services to a group of organizations also exist. *Software as a Service* (SaaS), Web 2.0 applications and recently mobile applications are built above the storage service and are offering new possibilities for sharing, using and consuming data in the form of data-intensive applications.

There are numerous storage providers today in the Cloud. The earliest cloud storage infrastructure was offered by Amazon, in the form of the Amazon S3 services [Amazon, 2012] (Simple Storage Service). Other similar services were launched following S3, including Google Cloud Storage [Google Cloud Storage, 2012], EMC Atmos [Atmos, 2012], Windows Azure storage [Azure, 2012], IBM SmartCloud [SmartCloud, 2012], Nirvanix *Storage Delivery Network* (SDN) [Nirvanix, 2012], Rackspace [CloudFiles, 2012]. In principle, they all offer simple object and file services, to efficiently upload, store and download data, with basic security features. OpenStack [OpenStack, 2012] is an open source cloud computing software framework originally based on Rackspace Cloud Files [CloudFiles, 2012]. Today there are over 150 companies contributing to this effort. OpenStack is comprised of several components, and its object storage component is called Swift [Swift, 2012].

Amazon S3 provided the earliest RESTful API to S3 cloud storage which has been widely adopted but is proprietary. The *Cloud Data Management Interface* (CDMI) [SNIA, 2012] is an emerging standard RESTful interface for cloud storage defined by the *Storage Networking Industry Association* (SNIA). OpenStack/Swift is open source cloud storage software supporting a native Swift API as well as an S3 compatible API and more recently CDMI support is being developed. All these interfaces define APIs by which applications can create data objects, organize them into containers and manipulate them.

Data management services to store and perform operations on the associated metadata (rather than the data itself) are often offered separately from the storage service, and these include for example Amazon SimpleDB [SimpleDB, 2012], Google BigTable [Chang, 2008], and Azure Table storage [Azure Table, 2012]. VISION Cloud [VISION, 2012] supported searchable object metadata, and also builds on this to provide the notion of content centric storage, whereby relations between objects (e.g. sets and lists) can be represented. Swift has support for object metadata, but the metadata is not searchable. Note that companies such as SoftLayer have added searchable metadata as a proprietary layer on top of Swift. Recently, there has been interest in a metadata search API for Swift in the Swift community [Swift Metadata Search Proposal, 2012].

Storage systems today are not aware of the semantics of the data they store (e.g., relationships between files, or the importance or semantics of the data). Such information is handled by the application, and often a content management system (not the storage system) manages the data, metadata, semantics and workflows, in a domain-specific manner. As a result, contemporary storage systems lack important information needed to effectively reason about and manage the content. VISION Cloud's data model provides data-intensive services with the rich semantics and scalability required for applications and services of the Future

Internet. The rich metadata also provides the basic infrastructure for efficient and scalable content management. Using this rich metadata, semantic information can be augmented to the raw data. For example, in order to deal specifically with the requirements of the Internet of Things (IoT) domain, one could use rich metadata to model Things and their relationships.

3.14 Data Reduction

The amount of data which is born digital is growing at an exponential rate, and is outpacing the growth in capacity of storage systems to host the data [Gantz and Reinsel, 2011]. The Internet of Things is a prime example of a domain which will give birth to a new generation of digital data in unprecedented quantities. We are therefore at risk of a data deluge, and techniques for data reduction are essential.

Data reduction techniques include compression, whereby the number of bytes required to store a given data file are reduced by encoding it, and deduplication, where data is indexed according to a content signature and this index ensures that a given piece of data is only stored once throughout the system. Since compression and deduplication require time and computing resources, both in order to encode and decode the data, they have been limited until recently to backup and archival storage systems and for data transfer purposes. Lately there has been interest in compression and deduplication for online storage [Tate, Tuv-El, Quintal, Traitel, and Whyte, 2012, Lu; Tretau, Miletic, Pemberton and Provost, 2012; Chambliss and Glider, 2012], although this is more challenging. Some data types and workloads are more amenable to data reduction than others, and there is typically a tradeoff between the cost savings achieved from data reduction, and the cost spent on the data reduction effort. Therefore, it is important to understand when it makes sense to compress, and tools such as IBM's comprestimator [IBM Comprestimator, 2012] have been developed for this purpose. Recently, techniques have recently been developed for estimating the potential gains in compression [Harnik, Margalit, Sotnikov, Kat and Traeger, 2013] and deduplication [Harnik, Margalit, Naor, Sotnikov, and Vernik, 2012; Xie, Condict and Shete, 2013] in storage systems. These techniques which allow estimating data reduction ratios on the fly or offline (depending on the use case), thereby applying data reduction only when the benefit outweighs the cost. In the cloud context, objects are replicated multiple times and stored across the network. This means that data reduction achieves both multiplied space savings as well as a reduction in the number of bytes transferred across the (possibly wide area) network. Deduplication protocols prevent uploading new objects to the cloud if those objects already reside there, and also prevent replicas from being sent across the network if they already exist at a replication target site. Note that, objects, as opposed to files, are typically write once, and update in place is not supported. This simplifies deployment of data reduction techniques, because they can be applied at the level of whole objects.

Another noteworthy trend in data reduction is domain specific compression. Historically this has been the case when approaching the compression of images (and fax at the time), and now dominantly in video encoding [e.g., Clarke, 1999; Thapa 2]. Another domain that has required new techniques for data reduction is that of Genome sequencing [Zhu et al. 2013].

Data reduction is essential in the Internet of Things domain, because of the massive amounts of data generated, and the expected exponential rate of growth. Moreover, the Internet of Things domain is a good candidate for data reduction since its data has inherent redundancy. For example, data generated by sensors could contain spatial redundancy if there are sensors whose ranges overlap, as well as temporal redundancy, if data is repetitive over time. This kind of data reduction requires a domain specific approach, however, little has been done in the area of refining existing techniques for data reduction and data reduction estimation and applying them to the Internet of Things domain.

3.15 Security

3.15.1. Security principles

Security refers to the degree of resistance or protection against harm. Modern security has become almost synonymous to digital systems where a constant race between the attacker and the attacked is taking place. The goal of digital security is the protection of sensitive data and the assurance of flawless operation of digital electronic systems.

Following rules can be considered as the guidelines in developing secure digital system, with special focus on embedded electronics:

- **Confidentiality**: Ensures that data is only disclosed to intended recipients. This is achieved by encrypting the data before transmission and that the data cannot be read during transmission, even if the packets are monitored or intercepted by an attacker. Only the party with the correct key will be able to decrypt the data;
- **Integrity**: Protects data from unauthorized modification in transit, ensuring that the data received is exactly the same as the data sent. Hash functions sign each packet with a cryptographic checksum, which the receiving partner checks before opening the packet. Only the sender and receiver have the key used to calculate the checksum. If the packet - and therefore signature - has changed, the packet is discarded;
- **Availability**: Provides the primitive of a system being always ready for the user's actions. This primitive is a combination of security and safety concepts which, combined, allow for long uptimes without glitches;
- **Authenticity**: Verifies the origin of a message through the process of one side sending a credential and the receiver verifying the legitimacy of the credential;
- **Non-repudiation**: Verifies that the sender of the message is the only person who could have sent it. The sender cannot deny having sent the message. Non-repudiation is a property of messages containing digital signatures when using public key technology. With public key technology, the sender's private key is used to create a digital signature that is sent with the message. The receiver uses the sender's public key to verify the digital signature. Because only the sender has possession of the private key, only the sender could have generated the digital signature. Non-repudiation is not a property of message authentication codes and hashes on messages using secret key technologies, because both the sender and the receiver have the secret key;
- **Speed**: Digital cryptography is well known for slowing down data transfer and digital system overall (e.g. web browsers which behave increasingly slower when using SSL). Speed is a key factor in ensuring seamless and painless digital encryption without a negative impact on the implied system;
- **Security**: using intrinsic security as the root of trust, the security pyramid is built from the ground up raising the protection level with each new added layer.

3.15.2. Hardware security

Today security is an add-on to existing hardware architectures. This approach has led to some effective protection mechanisms against some of the common security attacks. New advances in cryptanalysis have brought back the threat of security attacks, now more than ever. Security is a basic need for the Internet of Things thus it shall be an integral part of the hardware components. A secure hardware platform can enable secure software to run in a safe environment (e.g. ARM TrustZone). Thus the goal is to develop an efficient hardware platform which relies on heavy cryptographic primitives in order to provide a safe and secure foundation for the application level software – the so called *root-of-trust*. The first step in this direction is to provide a safe, on-chip, storage-like-compartment for the encryption keys used to decrypt various memory regions or to protect outgoing data. The second step is to provide a safe key exchange mechanism which will allow the manufacturer to provide secure update to the system. A third step (optional) is to use a dedicated CPU architecture which allows each application to run in its own secure compartment, protected by a unique key.

Present day solutions are oriented towards solving various security flaws. These platforms are focusing on application specific security as some of the following examples show.

Weightless [Weightless – a hardware platform for IoT (2014)] provides the scope to realize tens of billions of connected devices worldwide overcoming the traditional problems associated with current wireless standards - capacity, cost, power consumption and coverage. Weightless technology has been optimized for a standard designed specifically for machine communications within white space and is now being delivered as a royalty-free open standard. The first Weightless silicon provides:

- Capability of tuning across the entire UHF TV white space spectrum (470 – 790MHz);
- Little power;
- Reliable, secure, long range wireless connectivity;
- Implements few cryptographic solutions to provide a basic security mechanism.

The Marvell [Marvell 2014] PA800 implements Cryptography Research's latest *Consumable Crypto Firewall* (CCF) technology for use in systems that require secure authentication and/or secure usage tracking. It is a very low-cost, low-pin count chip that enables devices to cryptographically verify both authenticity and usage across components lifecycle.

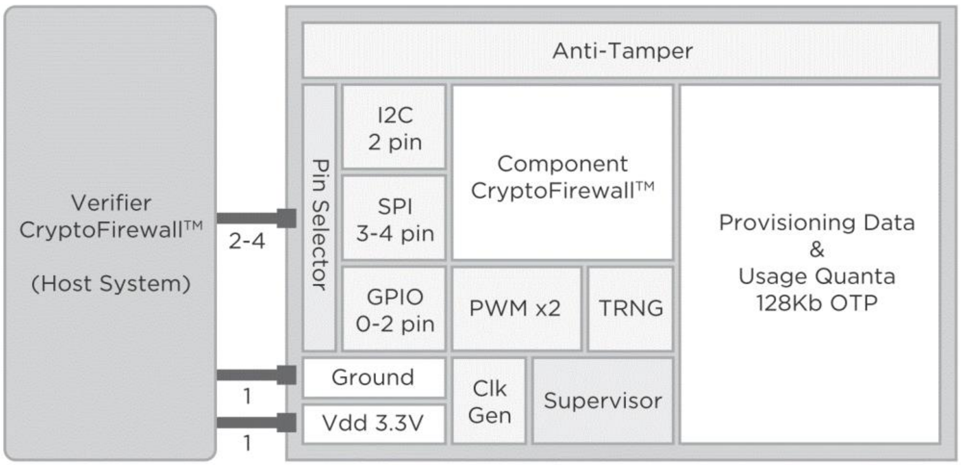

Figure 16: Marvell PA800 block diagram

Barco Silex [Barco-Silex 2014], one of the leading ASIC, FPGA and IP design companies, with extensive experience in security offers a wide range of products.

The main characteristics of Barco Silex's products are flexibility, portability and scalability of its IP cores. Their IPs are offering both hardened and unhardened against side channel attacks. Barco Silex offers dedicated IPs for cryptographic operations which support symmetric algorithms (DES, 3DES, AES) as well as asymmetric (ECC, RSA, PKI). As an enhancement Barco Silex also offers has functions (SHA, HMAC) as an add-on. A special care is given to the true random number generators which are a dedicated family of products.

All products are certified against state-of-the-art NIST standards.

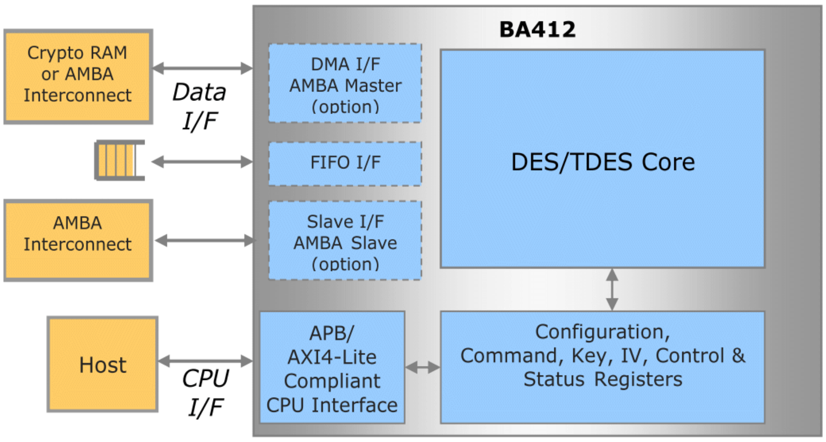
BA412
Crypto RAM or AMBA Interconnect
Data I/F
DMA I/F AMBA Master (option)
FIFO I/F
DES/TDES Core
AMBA Interconnect
Slave I/F AMBA Slave (option)
Host
CPU I/F
APB/ AXI4-Lite Compliant CPU Interface
Configuration, Command, Key, IV, Control & Status Registers

Figure 17: General description of BA412 IP

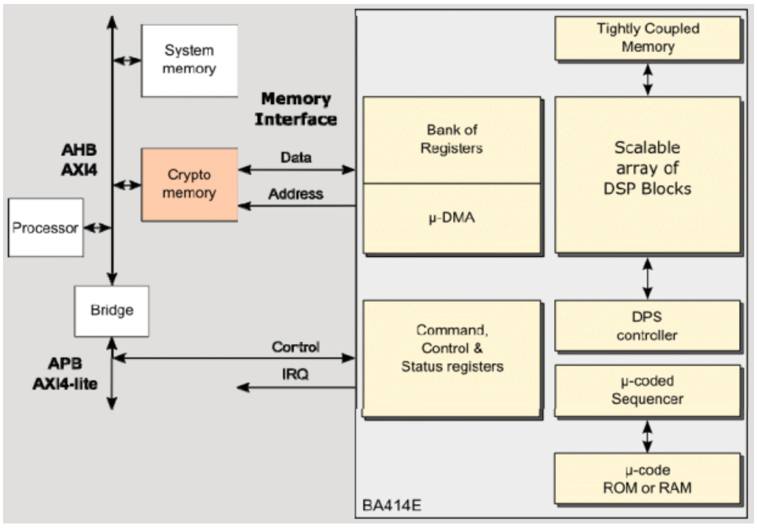

Figure 18: BA412 IP block diagram

All these solutions offer the basic security primitives but come with a drawback. Even if hardened against side channel attacks the provided modules (or IPs) are a standard peripheral of the system bus. That is, security mechanisms use the hardware provided primitives but are relying on software only control solutions. If an attacker modifies the software, the hardware enforced security is easily compromised as one might simply deactivate or circumvent the provided mechanisms.

Therefore a new approach is needed – one that ensures a "secure by design" approach.

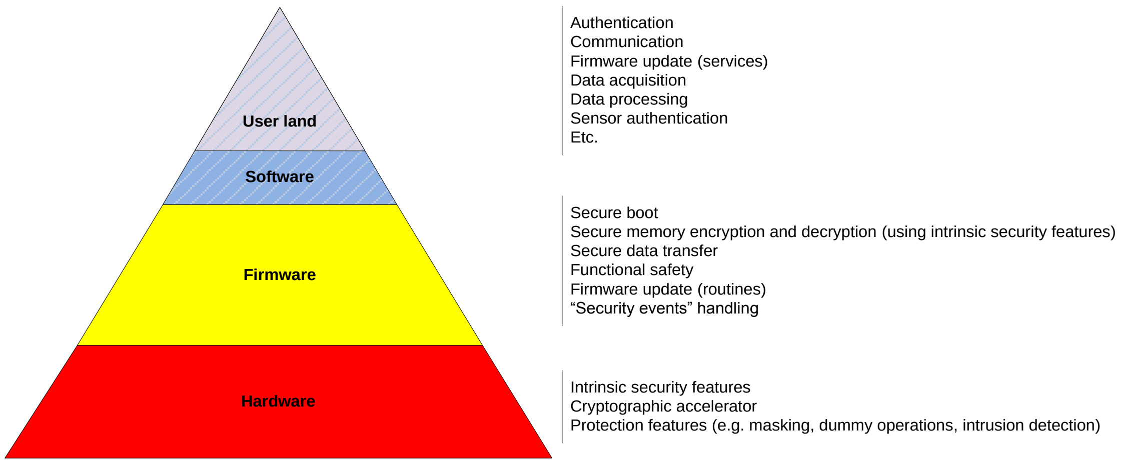
User land
Software
Firmware
Hardware
Authentication
Communication
Firmware update (services)
Data acquisition
Data processing
Sensor authentication
Etc.
Secure boot
Secure memory encryption and decryption (using intrinsic security features)
Secure data transfer
Functional safety
Firmware update (routines)
“Security events” handling
Intrinsic security features
Cryptographic accelerator
Protection features (e.g. masking, dummy operations, intrusion detection)

Figure 19: Root of trust

Although many solutions are available and already applied in commercial products there is no “golden rule” as to what hardware security needs to provide for a system to be secure. Still, as security is highly dependent on the applications only guidelines and evaluation criteria are available at the present moment. In this context the “Common Criteria” [Common Criteria Portal (2014)], an international undertaking to security evaluation is the most used evaluation and certification guideline. It is based on previous evaluation schemes and is built upon the expertise of governmental and institutions dedicated to security.

There are two possible evaluations: for products and for protection profiles. A protection profile is an implementation-independent set of security requirements for a category of products or systems that meet specific consumer needs. It provides a through description of threats, environmental issues and assumptions, security objectives, and Common Criteria requirements for a family of products.

In order to evaluate a single product, a so called “security target” has to be either derived from a protection profile or developed on its own. This is a set of requirements and specifications for a particular product.

The seven “Evaluation Assurance Levels” or short “EAL” are:

- EAL 1: “Functionally Tested Analysis” of security functions based on functional and interface specifications. It is applicable to systems where security threats are not serious;
- EAL 2: “Structurally Tested Analysis” of security functions including the high level design. Evidence of developer testing based on functional and interface specifications, independent confirmation of developer test results, strength-of-functions analysis, and a vulnerability search for obvious flaws must be provided;
- EAL 3: “Methodically Tested and Checked” is basically the same evaluation criteria as in EAL2 which additions referring to the use of development environment controls and configuration management. This level provides a moderate level of security;
- EAL 4: “Methodically Designed, Tested, and Reviewed”. This level requires a low-level design, complete interface description, and a subset of the implementation for the security function analysis. Additionally, an informal model of the product or system

security policy is required. This level targets systems with a moderate to high security requirement. Examples of EAL4 certified products are Microsoft Windows Server, commercial Linux server editions from companies like Red Hat or Novell;

- EAL 5: “Semiformally Designed and Tested”. A formal model, a semi formal functional specification, a semi formal high-level design, and a semi formal correspondence among the different levels of specification are required. This level is applicable for smart cards (e.g. Infineon SLExx family) and multilevel secure devices;
- EAL 6: “Semiformally Verified Design and Tested”. This level builds upon EAL5 with the added requirements for semi formal low-level design and structured presentation of the implementation;
- EAL 7: “Formally Verified Design and Tested” is the highest level of evaluation. It requires a formal representation of the functional specification and a high-level design, and formal and semi-formal demonstrations must be used in correspondence;

Still the “Common Criteria” only provides evaluation criteria and hardware alone only reaches EAL 5. Also hardware is only part of the system – even more in the present where the interconnected world is challenging every security and privacy aspect.

The proposed layered approach leverages the hardware components and the software components to provide on the one hand side a simple security model and on the other hand side a platform for developers to work with. Using standard components combined with custom cryptographic solutions enables a powerful root-of-trust.

If hardware poses security questions, the “cloud” itself is a newcomer when it comes to security and privacy.

3.15.3. Cloud Security

The scalable architectures of clouds infrastructures open up new security related issues and intensify other known vulnerabilities and threats. For example, most cloud storage services are offered by external providers on infrastructures also used for storing other customer’s data. Thus, many customers are rightfully worried about moving their data to a storage cloud and data security risks are a key barrier to the wide adoption of cloud storage [1-3].

However, security has its costs and the structure of very large scale storage systems incurs a trade-off between performance, availability and security [Leung et. al., 2007], which is challenging to balance. Most of the early cloud storage offerings provided minimal protections and security guarantees. However, recently security is gaining more and more attention. This issue becomes central both to the existing vendors, that improve their offerings, as well as new companies and services that aim to add an additional level of security or access control over the existing solutions.

OpenStack is disruptive open source software enabling to easily build public and private clouds. It is adopted by multiple European vendors, academic institutions and projects. However, as with many other commercial cloud products, the OpenStack community initially did not focus on its security leaving place for vulnerabilities and exploits. For example, LaBarge [LaBarge, 2012] investigated the security of OpenStack by performing a series of penetration tests with several different types of penetration protocols including network protocol and command line fuzzing, session hijacking and credential theft. Using these techniques exploitable vulnerabilities were discovered that could enable an attacker to gain access to restricted information contained on the OpenStack server, or to gain full administrative privileges on the server. The evaluators suggested the following key recommendations that should be used to prevent these exploits: (1) Use secure protocols, such as HTTPS, for Communications; (2) Encrypt all files that store user or administrative login credentials; (3) Correct code bugs and security vulnerabilities.

Another recent work [Albaroodi et.al, 2013], investigated the security issues in PaaS (*Platform as a Service*) and IaaS (*Infrastructure as a Service*), revealing several flaws in the OpenStack platform. They pointed out that one important threat posed by cloud computing is the obscuring of boundaries between internal and external security concerns. They recommend to closely studying the safety of the data and the service's availability, as cloud providers can be victims of attacks that stop the running of their operations. Another important topic raised in their work the need to provide data encryption as part of the cloud infrastructure. This will free the customers from managing the encryption keys which may be lost. Recognizing the importance of this, IBM is already leading an effort to contribute new code to allow server side encryption as part of the OpenStack Swift object store [OpenStack Server Encryption, 2014].

Unfortunately, providing a comprehensive end-to-end security in cloud environments is as hard as providing a physical protection from thefts in our everyday lives. Clouds offer multiple attack vectors and the security principles listed at the beginning of this section are multiplied by the number of layers and services involved in processing the cloud requests. For example, the addressed security measures should include such issues as: *Distributed Denial of Service* (DDoS) protection, secure access, network firewalls and private subnets, isolation of user and key management services, encryption of communication channels and data, integration of hardware-based crypto modules and many more.

When hosting applications in third-party clouds, customers have no mechanisms to check the end-to-end infrastructure security. One mechanism by which cloud providers can gain customer's trust is by showing compliance with well-known certifications. For example, *Amazon Web Services* (AWS) has recently shown significant efforts in achieving compliance from multiple regulation authorities [Amazon AWS Compliance, 2014]. Each certification means that an auditor has verified a compliance with a set of specific security controls. Unfortunately, this still does not ensure that the entire deployment has end-to-end security and cannot be exploited by malicious attackers. To let customers be aware and responsible of the potential problems Amazon has recently defined the "shared responsibility" concept [Amazon Web Services: Risk and Compliance, 2014]. They state that customers are responsible for the guest operating system, the associated applications as well as the consumption and the integration of the AWS services into their IT environments. This emphasizes the importance and the complexity of the cloud security problems, which become even more acute when integrated with Internet of Things (IoT) applications and infrastructures.

As IoT deals with people’s data, in most cases privacy becomes an important aspect when designing and operating a system.

3.15.4. Privacy in IoT

When it comes to privacy in IoT there are numerous approaches from which the most prominent ones are presented below.

3.15.4.1. Platform for Privacy Preferences Project (P3P)

Privacy policies [Cranor et. al. 2002] are already established as a principle to codify data collection and usage practices. We came across a recently finalized work that turns the encodings of some privacy policies into machine-readable XML. This allows automated developed processes to read and understand such policies in order to take action on them.

In brief, the P3P project there is a mechanism that contains XML elements to describe for example who is collecting information, what data is being collected, for whom, and why.

Using a similarly machine-readable preference language such as APPEL, users can express personal preferences over all aspects of such policies and have automated processes judge the acceptability of any such policy, or prompt for a decision instead. Since it might be cumbersome to manually create such preferences from scratch, a trusted third party could provide preconfigured preference specifications that would then be downloaded and individually adjusted by each user.

3.15.4.2. Privacy Proxies and Privacy-aware database

A different approximation to the privacy problem solution is made by privacy proxies [M. Langheinrich].

They handle privacy relevant interactions between data subjects and data collectors but also provide access to specific user control capabilities disclosed in the privacy policy such as data updates and deletes, or querying usage logs. Privacy proxies are continuously running services that can be contacted and queried by data subjects anytime, allowing them instant access to their data.

Once data has been solicited from the user, it is stored in a back-end database. In order to prevent accidental use of information that is in disagreement with the previously granted privacy policy, the database not only stores the data collected, but also each individual privacy policy that it was collected under.

So, privacy proxies allow for the automated exchange and update of both privacy policies and user data and a privacy-aware database combines the collected data elements and their privacy policies into a single unit for storage in order to consequently handle the data according to its usage policy.

3.15.4.3. Virtual Private Networks

A *Virtual Private Network* (VPN) [Weber R. H. (2010)] [Wikipedia VPN (2014)] extends a private network across a public network, such as the Internet. It enables a computer to send and receive data across shared or public networks as if it were directly connected to the private network, while benefiting from the functionality, security and management policies of the private network. This is done by establishing a virtual point-to-point connection through the use of dedicated connections, encryption, or a combination of the two. VPN’s are also used as extranets established by close groups of business partners. As only partners have access, they promise to be confidential and have integrity.

However, this solution does not allow for a dynamic global information exchange and is impractical with regard to third parties beyond the borders of the extranet.

3.15.4.4. Transport Layer Security

Transport Layer Security (TLS) [Weber R. H. (2010)] [Dierks T. and Rescorla E. (2008)] [Wikipedia TLS (2014)] is a cryptographic protocol (like the SSL) which is designed to provide communication security over the Internet. It uses X.509 certificates and hence asymmetric cryptography to assure the counterparty with whom it is communicating, and to exchange a symmetric key. This session key is then used to encrypt data flowing between the parties. This allows for data/message confidentiality, and authentication codes for message integrity and as a by-product, message authentication. TLS encrypts the data of network connections at a lower sub-layer of its application layer.

Based on an appropriate global trust structure, it could also improve confidentiality and integrity of the IoT. However, as each Object Naming Service delegation step requires a new Transport Layer Security connection, the search of information would be negatively affected by many additional layers.

3.15.4.5. DNS Security Extensions

This mechanism implemented in the application layer works by digitally signing records for DNS lookup [Weber R. H. (2010)] [Arends et al. (2005)] [Wikipedia DNS security(2014)] using public-key cryptography in order to guarantee origin authenticity and integrity of delivered information.

The correct DNSKEY record is authenticated via a chain of trust, starting with a set of verified public keys for the DNS root zone which is the trusted third party. Domain owners generate their own keys, and upload them using their DNS control panel at their domain-name registrar, which in turn pushes the keys via secDNS to the zone operator who signs and publishes them in DNS.

However, DNSSEC could only assure global Object Naming Service information authenticity if the entire Internet community adopts it.

3.15.4.6. Onion Routing

Onion Routing [Weber R. H. (2010)] [Goldschlag D., Reed M. and Syverson P. (1999)] [Wikipedia Onion Routing (2014)] is a technique for anonymous communication over a computer network.

This method encrypts and mixes Internet traffic from many different sources, i.e. data is wrapped into multiple encryption layers, using the public keys of the onion routers on the transmission path. This process would impede matching a particular Internet Protocol packet to a particular source.

Like someone peeling an onion, each onion router removes a layer of encryption to uncover routing instructions, and sends the message to the next router where this is repeated. This prevents these intermediary nodes from knowing the origin, destination, and contents of the message.

However, onion routing increases waiting times and thereby results in performance issues.

3.15.4.7. *Private Information Retrieval*

Those systems conceal which customer is interested in which information, once the *Electronic Product Code* (EPC) Information Services have been located.

In cryptography, a *Private Information Retrieval* (PIR) protocol [Weber R. H. (2010)] [Wikipedia PIR (2014)] allows a user to retrieve an item from a server in possession of a database without revealing which item is retrieved. PIR is a weaker version of 1-out-of-n oblivious transfer, where it is also required that the user should not get information about other database items.

One trivial, but very inefficient way to achieve PIR is for the server to send an entire copy of the database to the user. In fact, this is the only possible protocol that gives the user information theoretic privacy for their query in a single-server setting. There are two ways to address this problem: one is to make the server computationally bounded and the other is to assume that there are multiple non-cooperating servers, each having a copy of the database.

However, problems of scalability and key management, as well as performance issues would arise in a globally accessible system such as the *Object Naming Service* (ONS), which makes this method impractical.

3.15.4.8. *Peer-to-Peer systems*

A further method to increase security and privacy are *Peer-to-Peer* (P2P) [Weber R. H. (2010)] [Wikipedia P2P (2014)] systems, which generally show good scalability and performance in the applications.

In general, a P2P network is a type of decentralized and distributed network architecture in which individual nodes in the network (called "peers") act as both suppliers and consumers of resources, in contrast to the centralized client–server model where client nodes request access to resources provided by central servers.

In a peer-to-peer network, tasks are shared amongst multiple interconnected peers who each make a portion of their resources directly available to other network participants, without the need for centralized coordination by servers.

These P2P systems could be based on *Distributed Hash Tables* (DHT). Access control, however, must be implemented at the actual Electronic Product Code Information Services itself, not on the data stored in the DHT, as there is no encryption offered by any of these two designs. Insofar, the assumption is reasonable that encryption of the EPCIS connection and authentication of the customer could be implemented without major difficulties, using common Internet and web service security frameworks. In particular, the authentication of the customer can be done by issuing shared secrets or using public-key cryptography.

3.15.4.9. *Anonymity and Pseudonymity*

Anonymity can be defined as "the state of being not identifiable within a set of subjects." The larger the set of subjects is, the stronger is the anonymity. A large number of both free and commercial anonymity services are already in widespread use on the World Wide Web. Using anonymizing proxies, for example the popular www.anonymizer.com, or more sophisticated "mixes", like the "Freedom" software product of the Canadian software company Zero-Knowledge, Internet users can already today hide their IP address from the Web site hosting the accessed page. Anonymity has also disadvantages from an application point of view. Being anonymous prevents the use of any application that requires authentication or offers some form of personalization.

Pseudonymity is an alternative that allows for a more fine grained control of anonymity in such circumstances: by assigning a certain ID to a certain individual, this person can be repeatedly identified until she changes to a different ID. Using the same pseudonym more than once allows the holder to personalize a service or establish a reputation, while always offering her the possibility to step out of that role whenever she wishes. Whether anonymous or pseudonymous, the collection and usage of such data poses no threat to the individual's privacy. Consequently, legal frameworks such as the EU Directive lay no restriction on the collection of anonymous (or pseudonymous) data. Determining when certain type of information can be linked back to a person, however, is more often than not subject of debate. For example, even randomly generated pseudonyms might be linkable under certain circumstances: In case a pseudonym is used in conjunction with a certain fact that is easy to identify in a sufficiently small set, linking becomes trivial. An active badge might be programmed to change its ID every five minutes, though the fact that the tracking system is able to exactly pinpoint its location would make this change obvious (and thus linkable) in the logs. [Langheinrich M. (2001)]

3.15.4.10. *k-Anonymity*

k-Anonymity [Shmatikov V. and Narayanan A. (2010)] is a property possessed by certain anonymised data. A release of data is said have the k-anonymity property if the information for each person contained in the release cannot be distinguished from at least k-1 individuals whose information also appear in the release. To apply k-anonymity or its variants such as l-diversity, the set of the so called quasi-dentifier attributes must be fixed in advance and assumed to be the same for all users. It typically includes ZIP code, birth date, gender, and/or other demographics. The rest of the attributes are assumed to be non-identifying. De-identification involves modifying the quasi-identifiers to satisfy various syntactic properties, such as "every combination of quasi-identifier values occurring in the dataset must occur at least k times". There are two common methods for achieving k-anonymity for some value of k.

1. Suppression: In this method, certain values of the attributes are replaced by an asterisk '*'. All or some values of a column may be replaced by '*';
2. Generalization: In this method, individual values of attributes are replaced by with a broader category. For example, the value '19' of the attribute 'Age' may be replaced by ' ≤ 20', the value '23' by '20 < Age ≤ 30' , etc.

The trouble is that even though joining two datasets on common attributes can lead to re-identification, anonymizing a predefined subset of attributes is not sufficient to prevent it.

3.15.4.11. Other Cryptographic Algorithms

Usually the symmetric encryption algorithm is used to encrypt data for confidentiality such as the *Advanced Encryption Standard* (AES) block cipher.

The asymmetric algorithm is often used to digital signatures and key transport, frequently-used algorithm is the *Rivest Shamir Adleman* (RSA), the *Diffie-Hellman* (DH) asymmetric key agreement algorithm is used to key agreement, and the SHA-1 and SHA-256 secure hash algorithms will be applied for integrality. Another significant asymmetric algorithm is known as *Elliptic Curve Cryptography* (ECC), ECC can provide equal safety by use of shorter length key, the adoption of ECC has been slowed and may be encouraged recently. To implement these cryptographic algorithms available resources are necessary such as processor speed and memory. So how to apply these cryptographic techniques to the IoT is not clear, we have to make more effort to further research to ensure that algorithms can be successfully implemented using of constrained memory and low-speed processor in the IoT [Suo et. al. (2012)].

3.15.4.12. SafeMask

The IoT platform has to enforce privacy with utility without modifying the actual user data sensed and stored in databases. While doing so, it has to use the privacy rules. So we need a technique that does not hamper the actual data in databases and uses rules that are externalized to enforce privacy. Therefore a dynamic data masking solution there is already proposed.

Data masking provides an alternative to entirely re-engineer the application architecture to achieve privacy. It is simply the process of systematically removing or modifying data elements that could be used to gain additional knowledge about the sensitive information. Masking technology has two main aspects: *Static Data Masking* (SDM) that deals with data at rest and *Dynamic Data Masking* (DDM) that aims at real-time data masking of data in transition. SafeMask [Arijit et. al (2012)] is a dynamic data masking solution that enforces privacy. It consists of a data request interpreter, rule interpreter and a masker. Data request interpreter decodes all the requests made by data consumers to IoT platform for user data. Rule interpreter reads the privacy rules for data consumers deployed at SafeMask and identifies the sensitive attributes along with its corresponding privacy preservation technique. Masker masks the sensitive data using the preservation technique defined in the rule. A typical masking process is as follows:

1. Data consumers request the IoT platform for user data.
2. The data interpreter in SafeMask interprets the request and identifies the data consumer.
3. It then decodes the attributes in the data requested and fetches the requested data from IoT platform.
4. The consumer information along with the requested data attributes and its values are then sent to rule interpreter.
5. Rule interpreter fetches the rule corresponding to the data consumer and computes the sensitivity of the attributes requested against the attributes in the rule.
6. Values of the attributes identified as sensitive are then masked by the Masker using the privacy preservation technique defined for the attribute.

3.15.4.13. The butterfly method

[Pei et. al. (2009)] [Wikipedia Quasi identifier (2014)] Another way to multiplex information so anonymity can be granted is with the butterfly method. Quasi-identifiers as mentioned and before are pieces of information that are not of themselves unique identifiers, but are sufficiently well correlated with an entity that they can be combined with other quasi-identifiers to create a unique identifier.

Quasi-identifiers can thus, when combined, become personally identifying information. This process is called re-identification. With this proposed technique, we are able to alter some values on the data exchanged and more precisely on those quasi-identifiers so that we can prevent the re-identification process and guarantee anonymity on our network.

As all these method reveal the complex aspects of security and privacy are being tackled in numerous ways across the world. Depending on the application and the needed security and privacy level there are more than one option. Still there are a great number of open questions as the need for answers is growing in the context of the IoT.

4 Project Requirements

4.1 Requirement engineering methodology

Following the IoT-A ARM [Carrez et al.'2013] Methodology 5 activities take place at the early stage of the architecting process:

- **Design of Physical-Entity View**: The Physical-Entity view is not described in the ARM as it is extremely specific to the kind of IoT application the concrete architecture of which is being developed. Nevertheless this view is extremely important at the early stage of the architecting process. The aim for the P-E view is:
    - To identify Entities of Interest for the IoT system
    - To describe devices used to monitor the PEs and explain relation to the PE: are they attached, do they touch or just in sight?
    - Describe which characteristics or property of the PE is monitored? And link to the device.

  In COSMOS we have to describe a P-E view for each scenario from WP7. We can't describe a "generic" P-E view that applies to any possible COSMOS-enabled use-case for the same reasons that we did not produce such a view in IoT-A IoT Architectural Reference Model.
- **Design of IoT Context view:** The IoT Context view is made of the two Context view and IoT Domain Model.
    - **Context view:** According to the ARM this view describes relations, interactions and dependencies existing between the IoT system (here the COSMOS platform) and its environment (actors, external entities)
    - **Domain Model:** Domain Model identifies the Concepts that need to be introduced for the specific domain of Internet of Things and shows which relations exists between those concepts. It gives then the vocabulary needed to "discuss" about IoT. Referring to and using the IoT-A IoT Domain Model allows parties involved in COSMOS to use unambiguously the same vocabulary. The Domain Model is part of Deliverable D2.3.1.
- **Requirement process:** The Requirement engineering consists of various steps which take into account the new concepts pertaining to the COSMOS vision, the current state of the Art. Some of those requirements will relate to Architectural Views (they directly inform one of the ARM Views), some will relate to IoT System qualities (relating then to ARM Perspectives, which are system qualities –like Resilience or Security- that span all views of the IoT system), some relate to design constraints brought for instance by our test-bed and use-case partners. The IoT-A project came with a long list of generic requirements called *UNIfied requirements* (UNIs) that can be reused by specific IoT project in order to generate their own specific requirements. In COSMOS we have followed this list of UNIs, reusing them when they could be directly used or customizing them in order to get the specific COSMOS flavor. Finally some requirements pertaining to the specific COSMOS vision were created (e.g. all requirements related to the concept of "experience").

As shown in the Figure 20 below (taken from the ARM [Carrez et al.'2013])

The Physical-Entity and IoT Context views are 2 essential views part of the several views the COSMOS project architecture will be made of (they shall be described in D2.3.1). There are part of the COSMOS Architecture deliverable and are therefore ignored in this document.

From the COSMOS Roadmap point of view point of view those two views are elaborated in parallel to the Business Goal and Requirements Analysis described here.

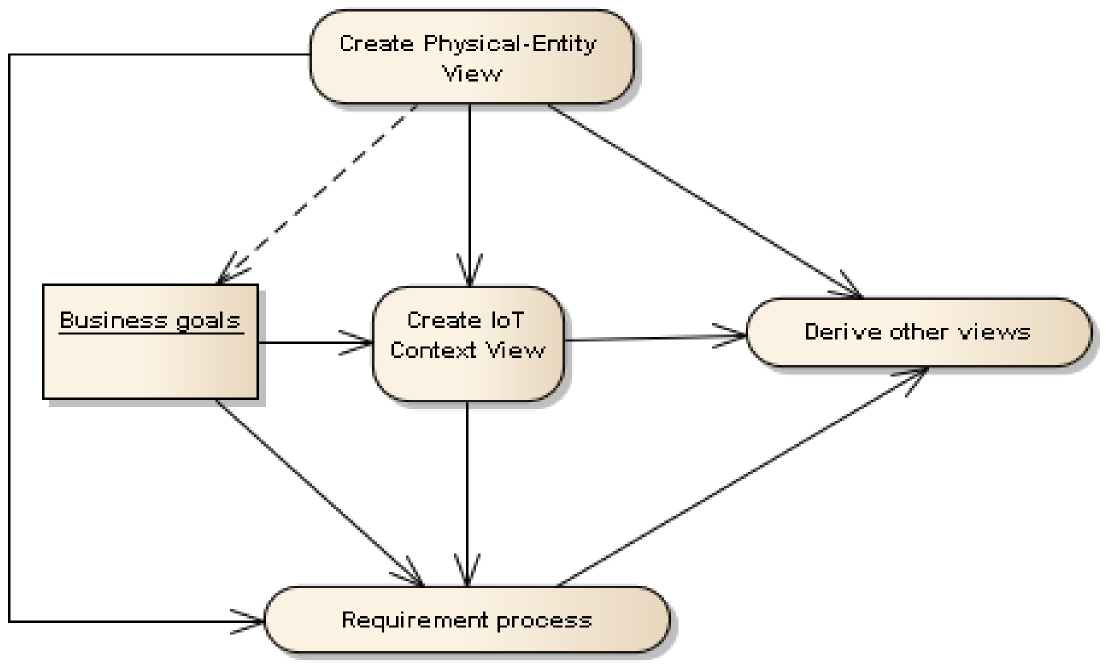

Figure 20: Simplified view of the architecting process

The main purpose of this section is to collect the Business goals and System Requirements.

The System Requirements are made of three categories. Each category will influence the design of the COSMOS Architecture in different ways as explained hereafter.

These requirement categories are:

- **Functional Requirements:** They describe fundamental or essential functional characteristics of the targeted architecture. They explain what the system as to do or to process, without focusing on specific method, algorithms, technology or design strategy. Design and technology choices are made later during the architecting process. Functional requirements are heavily impacting the architecture views like the Information view and functional view;
- **Non-functional requirements:** Non-functional requirement described properties of the system, which must be met by the architecture. Typical examples of system properties are Security, Performance, Scalability, Usability, and Resilience… Of course different strengths or flavor of such high level properties can be described;
- **Design constraints:** Sets up technical constraints or restrictions about how the system must be designed.

In addition to those three classes of requirements it is be worth keeping track of the origin of the requirement. As for COSMOS, we will aim at tagging requirements according to one of the following options:

- **Work package:** one of the technical WP i.e. WP3, 4, 5 or 6
- **Use case:** which one of the WP7 use-cases originated the requirement
- **Business:** emphasizes which aspect of the business goal motivated a requirement

4.2 Template for collecting requirements

The template used by COSMOS for collecting requirements comes from IoT-A (with minor updates). The meanings of the different columns is described below:

- Column A – UNI ID: unique identifier. Initially, we suggest using <WP number.integer>. Later on, after unifying, grouping and cross checking the whole set of requirements, they will be assigned a new identifier consisting of a category prefix followed by a number.
- Column B – Requirement Type: a key word from {Functional, Non-Functional, Design Constraint}
- Column C – Origin: choice between WP number, Use-Case name (i.e. MAD or HILD) or Business
- Column D – Priority: A key word from {MUST, SHOULD, COULD} following the MoScOW methodology[5]
- Column E – Category: select here a key word that qualifies the best your requirement (and please don't create duplicates so have first a look to what has been already used!) e.g.
    - Security
    - Privacy
    - Performance
    - Scalability
    - Resilience
    - Semantic
- Column F – Description: shortly describe the requirement there
- Column G – Rationale: briefly describe the reason behind having such a requirement
- Column H – Fit Criteria: describe how we are going to verify that the requirement has been taken into account when designing the COSMOS system.
- Column I – Dependencies: link to other requirements in case there exist dependencies
- Column J – Conflict: Link to other requirements in case there exist a conflict between them.
- Column K – S: To be filled in during Year 2
- Column T – System use-case: link to a specific WP7 use case if relevant.
- Column U: - Comment: Add a comment of needed

4.3 Requirements

The list of requirements can be accessed through the Annex 1 attached to this document (Separate Excel file). This COSMOS_Requirements (v2) is the second iteration of the Requirement list. It will be refined once more in 2015. This list is being checked regularly in order to ensure that no requirement is left behind. For this second version we have implemented a color code as follows:

- Blue: is for requirements which were left as in initial version of the excel sheet;
- Green: is for requirements which were modified compared to the initial version. The older text is kept between square brackets [ ];
- ~~Stroke through~~: is for requirements which were considered irrelevant assuming decisions taken since the initial version was released.

At the time this document is released the mapping to architecture document has not been made yet. Intermediary versions will be released during Years 2 and 3.

[5] http://en.wikipedia.org/wiki/MoSCoW_Method

5 References

A. Dey, D. Salber, and G. Abowd (2001). A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. Special issue on Context-Aware Computing in the Human-Computer Interaction (HCI) Journal, vol. 16 (2-4), pp. 97-166

A. Tuladhar (2008). A study on context awareness architecture in business process workflow. Special Study Report, AIT

Aamoth, A., Plaza, E. (1994), "Case-Based Reasoning: Foundational Issues, Methodological Variations and System Approaches", Artificial Intelligence Communications, 1994, pp. 39-59

Abdelhamid Salah brahim, Le Grand B., Tabourier L., Latapy M. (2001), Citations among blogs in a hierarchy of communities: method and case study, Journal of Computational Science.

Abowd, A. Dey and G. (1997). CyberDesk, The Use of Perception in Context-Aware Computing. Extended Abstract presented as a poster in the Proceedings of 1997 Workshop on Perceptual User Interfaces PUI '97, pp. 26-27

Aggarwal, C. C., Ashish, N., & Sheth, A (2009), "The Internet of Things: A Survey from the Data-Centric Perspective", Managing and Mining Sensor Data.

Albaroodi, Hala, Selvakumar Manickam, and Parminder Singh. (2013)"CRITICAL REVIEW OF OPENSTACK SECURITY: ISSUES AND WEAKNESSES."Journal of Computer Science 10.1.

Amazon (2012), http://aws.amazon.com/s3/

Amazon AWS Compliance (2014), http://aws.amazon.com/compliance/

Amazon EMR Training, http://aws.amazon.com/elasticmapreduce/training/

Amazon EMR, Amazon Elastic Map Reduce http://aws.amazon.com/elasticmapreduce/

Amazon SimpleDB (2012), http://aws.amazon.com/simpledb/

Amazon Web Services: Risk and Compliance (2014), http://media.amazonwebservices.com/AWS_Risk_and_Compliance_Whitepaper.pdf

Arends R., Austein R., Larson M., Massey D. and Rose S. (2005), "DNS Security Introduction and Requirements,".

Arijit U., Soma B., Joel J., Vijayanand B. and Sachin L. (2012), "Negotiation-based Privacy Preservation Scheme in Internet of Things Platform".

Arup (2011), Report "The Smart Solutions for Cities", Arup UrbanLife

ASPIRE (2011), http://wiki.aspire.ow2.org/xwiki/bin/view/Main.Documentation/AspireRfidArchitecture

Assali, A. A., Lenne, D., Debray, B. (2009), "Case Retrieval in Ontology-Based CBR Systems", 32nd Annual Conference on Artificial Intelligence, Paderborn, 2009, pp. 564-571

Atos Smart Objects Lab Complex Event Processor (2012), http://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/Backend_Things_Management_-_SOL_CEP_User_and_Programmers_Guide

Atzori, L., Iera, A., & Morabito, G. (2010). The internet of things: A survey.Computer Networks, 54(15), 2787-2805

AutoMap: http://www.casos.cs.cmu.edu/projects/automap/

Azure Table storage (2012), http://msdn.microsoft.com/en-us/library/windowsazure/dd179423.aspx

Bandyopadhyay, D., & Sen, J. (2011). Internet of Things: Applications and Challenges in Technology and Standardization. Wireless Personal Communications, 58(1), 49-69

Barco-Silex (2014), http://www.barco-silex.com/

Barnaghi, P., Wang, W., Henson, C., & Taylor, K. (2012). Semantics for the Internet of Things: early progress and back to the future. International Journal on Semantic Web and Information Systems (IJSWIS), 8(1), 1-21.

Barros, A., Kylau, U., & Oberle, D. (2011) Unified Service Description Language 3.0 (USDL) Overview. Available: http://www.internet-of-services.com/fileadmin/IOS/user_upload/pdf/USDL-3.0-M5-overview.pdf.

Begault, D. R, Anderson, M. R, McClain, B. U, & Miller, J. D (2011). Spatial Auditory Displays for Enhancing Situational Awareness During Non-terrestrial and Remote Exploration. 4th IEEE International Conference on Space Mission Challenges for Information Technology, Palo Alto, CA

Bellifemine et al. (2007) : Developing Multi-Agent Systems with JADE. Willey.

Bergmann, R., Kolodner, J., Plaza, E. (2005), "Representation in case-based reasoning", The Knowledge Engineering Review, Vol. 00:0, 1–4, 2005, Cambridge University Press.

Bill N. Schilit and Marvin M. Theimer (1994). Disseminating Active Map Information to Mobile Hosts. IEEE Network, 8(5), pages 22-32, September/October 1994, IEEE Computer Society.

Bobrinsky, N. and Del Monte, L. (2010). The space situational awareness program of the European Space Agency. In Kosmicheskie Issledovaniya, 2010, Vol. 48, No. 5, pp. 402–408.

Boing Magazine (2012). Reducing runway landing overruns [Online]. Available: http://www.boeing.com/commercial/aeromagazine/articles/2012_q3/3/

Boniface, M., & Pickering, B. (2011). Legislative Tensions in Participation and Privacy. http://www.scribd.com/doc/55260687/Legislative-Tensions-In-Participation-And-Privacy

Bonino da Silva Santos, L.O., Ramparany, F., Dockhorn Costa, P., Vink, P., Etter, R., Broens, T. (2007). *A Service Architecture for Context Awareness and Reaction Provisioning*. 2nd Modeling, Design, and Analysis for Service-Oriented Architecture Workshop (MDA4SOA 2007), Salt Lake City, USA

Bonino da Silva Santos, L.O., van Wijnen, R. P., Vink, P. (2007). *A service-oriented middleware for context-aware applications*. Proceedings of the 5th International workshop on Middleware for pervasive and ad-hoc computing, p.37-42, Newport Beach, California

Borek, A., Woodall, P., Oberhofer, M., & Parlikad, A. K. (2011). A Classification of Data Quality Assessment Methods. In Proceedings of the 16th International Conference on Information Quality, Adelaide, Australia

Brézillon, Ghita Kouadri Mostéfaoui and Patrick (2003). A Generic Framework for Context-Based Distributed Authorizations. Fourth International and Interdisciplinary Conference on Modeling and Using Context (Context'03), pp. 204-217

Brouninghaus S., Ashley, K.D. (2003), “Combining Case Based and Model-Based Reasoning for Predicting the Outcome of Legal Case”

Brown, M. (1996). Supporting User Mobility. International Federation for Information Processing

Brown, Scott M., Santos, Eugene, Jr., and Bell, Benjamin (2002), “Knowledge Acquisition for Adversary Course of Action Prediction Models,” Proceedings of the AAAI 2002 Fall Symposium on Intent Inference for Users, Teams, and Adversaries, Boston, MA

Budimac, Z., Kurbalija, V., “CASE BASED REASONING – A SHORT OVERVIEW”, Proceedings of the Second International Conference on Informatics and InformationTechnology, pp222-233

BUTLER Deliverable D2.2 (2012). Requirements, Specifications and Localization and Context-acquisition for IoT Context-Aware Networks.

C. Castellucia, A. Spognardi (2007). RoK: A Robust Key Pre-Distribution Protocol for Multi-Phase Wireless Sensor Networks. IEEE Securecomm, Nice, France

Calder, M., Morris, R. A., & Peri, F. (2010). Machine reasoning about anomalous sensor data. Ecological Informatics, 5(1), 9-18

Carrez F. et al. (2013). Final Architecture Reference Model for the IoT v3.0. IoT-A deliverable available at www.iot-a.eu/

Cassar, G., Barnaghi, P., Wang, W., & Moessner, K. (2012), A Hybrid Semantic Matchmaker for IoT Services

Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., ... & Gruber, R. E. (2008). Bigtable: A distributed storage system for structured data. ACM Transactions on Computer Systems (TOCS), 26(2), 4

Charith Perera, Arkady Zaslavsky, Peter Christen, Dimitrios Georgakopoulos (2012). CA4IOT: Context Awareness for Internet of Things. Proceeding of the IEEE International Conference on Green Computing and Communications, Conference on IoT, and Conference on Cyber, Physical and Social Computing. Besancon, France. Pages 775-782.

Charith Perera, Arkady Zaslavsky, Peter Christen, Dimitrios Georgakopoulos (2013). Context Aware Computing for The Internet of Things: A Survey. IEEE Communications Surveys & Tutorials Journal

Chen et al. 2006. Mobile C: a mobile agent platform for C/C++ agents. Software Practice & Experience 2006; vol 36: p1711–p1733

Chiang, F., & Miller, R. J. (2008). Discovering data quality rules. Proceedings of the VLDB Endowment, 1(1), 1166-1177

CISCO (2011), The Internet of Things, Infographic, http://blogs.cisco.com/news/the-internet-of-things-infographic

Clarke, R., (1999), Image and video compression: a survey, International Journal of Imaging Systems and Technology (10), 20-32.

Claudio Bettini , Oliver Brdiczka , Karen Henricksen , Jadwiga Indulska , Daniela Nicklas , Anand Ranganathan , Daniele Riboni (2010). *A survey of context modelling and reasoning techniques.* Pervasive and Mobile Computing, v.6 n.2, p.161-180

Common Criteria Portal (2014), http://www.commoncriteriaportal.org/

Compton, M., Barnaghi, P., Bermudez, L., Garcia-Castro, R., Corcho, O., Cox, S., ... & Taylor, K. (2012). The ssn ontology of the w3c semantic sensor network incubator group. Web Semantics: Science, Services and Agents on the World Wide Web.

Content Manager (2012), http://www-01.ibm.com/software/data/cm/cmgr/

Cooperstock, J., Tanikoshi, K., Beirne, G., Narine, T., Buxton, W. Evolution of a Reactive Environment CHI '95 (1995) 170-177

Correia, L. ,Wünstel K. (2011), "Smart Cities Applications and Requirements", Net!Works European Technology Platform Expert Working Group White Paper

Cranor L., Langheinrich M., Marchiori M. and Reagle J. (2002), "The platform for privacy preferences 1.0 (P3P1.0) specification.," W3C Recommendation, HTML Version at www.w3.org/TR/P3P/.

D. Garlan, S.-W. Cheng, A.-C. Huang, B. Schmerl, and P. Steenkiste, "Rainbow: Architecture-Based Self-Adaptation with Reusable Infrastructure," Computer, vol. 37, pp. 46-54, 2004.

D. Weyns and M. Georgeff, "Self-Adaptation Using Multiagent Systems," IEEE Software, vol. 27, pp. 86-91, Jan-Feb 2011.

Daniel Salber, Anind K. Dey, Gregory D. Abowd (1999). The Context Toolkit: Aiding the Development of Context-Enabled Applications. In Proceedings of ACM CHI

Davis, R., King, J.J., "The Origin of Rule-Based Systems in AI"

Davy Preuveneers and Yolande Berbers (2008). IoT: A Context-Awareness Perspective. The IoT: From RFID to the Next-Generation Pervasive Networked Systems, edited by Lu Yan, Yan Zhang, Laurence T. Yang, and Huansheng Ning. Auerbach Publications.

De, S., Barnaghi, P., Bauer, M., & Meissner, S. (2011, September). Service modelling for the Internet of Things. In Computer Science and Information Systems (FedCSIS), 2011 Federated Conference on (pp. 949-955). IEEE

Dean, J., & Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. Communications of the ACM, 51(1), 107-113

Dey, A.K., Abowd, G.D. (1999). Towards a Better Understanding of Context and Context-Awareness. Technical Report GIT-GVU-99-22, GVU Center, Georgia Institute of Technology. Available: ftp://ftp.cc.gatech.edu/pub/gvu/tr/1999/99-22.pdf

Dierks T. and Rescorla E. (2008), "The Transport Layer Security (TLS) Protocol, Version 1.2,".

Dirks, S., Gurdgiev, C., & Keeling, M. (2010). Smarter Cities for Smarter Growth: How Cities Can Optimize Their Systems for the Talent-Based Economy. IBM Institute for Business Value

Doan, A., Ramakrishnan, R., & Vaithyanathan, S. (2006, June). Managing information extraction: state of the art and research directions. In Proceedings of the 2006 ACM SIGMOD international conference on Management of data(pp. 799-800). ACM.

Documentum (2012), http://www.emc.com/domains/documentum/index.htm

Dolce Language Specification version 1 (2012), Atos Smart Objects Lab

Dominguez, C. (1994). Can SA be defined? In Situation awareness: Papers and annotated bibliography. USAF Armstrong Laboratory Report AL/CF-TR-1994-0085

Durso, F. T., Hackworth, C. A., Truitt, T. R., Crutchfield, J. M., Nikolic, D., & Manning, C. A. (1998). Situation awareness as a predictor of performance for en route air traffic controllers. Air Traffic Control Quarterly, 6(1), 1-20.

Dynet: http://www.casos.cs.cmu.edu/projects/DyNet/

DyNetML: http://www.casos.cs.cmu.edu/projects/dynetml/

Economist (2010) Report, “It's a smart world”, http://www.managementthinking.eiu.com/sites/default/files/downloads/Special%20report%20on%20smart%20systems.pdf

Egonet: http://www.casos.cs.cmu.edu/projects/Optimizer/

Eid, M., Liscano, R., & El Saddik, A. (2007, June). A universal ontology for sensor networks data. In Computational Intelligence for Measurement Systems and Applications, 2007. CIMSA 2007. IEEE International Conference on (pp. 59-62). IEEE.

Elrod, S., Hall, G., Costanza, R., Dixon, M., des Rivieres, J. (1993). Responsive Office Environments. CACM 36(7) 84-85

EMC Atmos (2012), http://www.emc.com/storage/atmos/atmos-cloud-delivery-platform.htm

ENCOURAGE (2011), Embedded Intelligent Controls for Buildings with Renewable Generation and Storage

Endsley, M. R. (1988). Situation awareness global assessment technique (SAGAT). Paper presented at the National Aerospace and Electronic Conference (NAECON), Dayton, OH.

Endsley, M.R, Holder, L.D., Leibrecht, B.C., Garland, D.J., Wampler, R.L., & Matthews, M.D. (2000). Modeling and Measuring Situation Awareness in the Infantry Operational Environment. (Research Report #1753). Alexandria, VA: U.S. Army Research Institute for Behavioral and Social Sciences.

Endsley, M.R., Farley, T.C., Jones, W.M., Midkiff, A.H. and Hansman, R.J. (1998). Situation Awareness Information Requirements for Commercial Airline Pilots. International Center for Air Transportation.

EPA (2012), “Car pollution effects”, U.S. Environmental Protection Agency (EPA)

EPCglobal (2010), http://www.gs1.org/epcglobal

ESSAI Project Report (2000) [Online]. Available: http://essai.nlr.nl/downloads/ESSAI_WP1.pdf

Eurostat (2007), “Passenger mobility in Europe”, European Commission

Eurostat (2011), “Energy, transport and environment indicators”, European Commission

Evans, D. (2011), “The Internet of Things How the Next Evolution of the Internet Is Changing Everything”, CISCO white paper

Event Processing Language (2008). http://esper.codehaus.org/esper-2.0.0/doc/reference/en/html/epl_clauses.html

Event Stream Intelligence Continuous Event Processing for the Right Time Enterprise (2012). http://www.espertech.com/download/public/EsperTech%20technical%20datasheet%20v8.pdf

Federal Information Processing Standards Publication (2004), Standards for Security Categorization of Federal Information and Information Systems: http://csrc.nist.gov/publications/fips/fips199/FIPS-PUB-199-final.pdf

Ferber, J. (1995). Multi Agent Systems: Towards a Collective Intelligence. Inter-Editions

Fickas, S., Korteum, G., Segall, Z. (1997). Software Organization for Dynamic and Adaptable Wearable Systems. 1st International Symposium on Wearable Computers 56-63

Filenet (2012), http://www-01.ibm.com/software/data/content-management/filenet-content-manager/

FI-WARE (2012), http//www.fi-ware.eu

Franke, J., Brown, S. M., Bell, B., and Mendenhall, H. (2000), "Enhancing Teamwork Through Team-Level Intent Inference," Proceedings of the International Conference on Artificial Intelligence (IC AI 2000), Las Vegas, NV

Gantz, J., & Reinsel, D. (2011). The 2011 digital universe study: Extracting value from chaos. IDC: Sponsored by EMC Corporation

Gligor, V., & Wing, J. (2011). Towards a theory of trust in networks of humans and computers. Security Protocols XIX, 223-242

Golbeck, J. (2009). Trust and nuanced profile similarity in online social networks. ACM Transactions on the Web (TWEB), 3(4), 12

Goldschlag D., Reed M. and Syverson P. (1999), "Onion Routing for Anonymous and Private,"

Gomes D, (2011), "internet of things applications / services", Technology challenges for the Internet of Things

Gonzalez, J., Rossi, A. (2011), "New Trends for Smart Cities", http://www.opencities.net/sites/opencities.net/files/content-files/repository/D2.2.21%20New%20trends%20for%20Smart%20Cities.pdf, 2011)

Google Cloud Storage (2012), http://code.google.com/apis/storage/

GraphML: http://graphml.graphdrawing.org/

GraphViz: http://www.graphviz.org/

Grimoires, http://twiki.grimoires.org/bin/view/Grimoires/

Gualtieri, M., & Rymer, J. R. (2009). The Forrester Wave™: Complex Event Processing (CEP) Platforms, Q3 2009. CEP.

Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2012). Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions. arXiv preprint arXiv:1207.0203.

Guinard, D., Trifa, V., Karnouskos, S., Spiess, P., & Savio, D. (2010). Interacting with the soa-based internet of things: Discovery, query, selection, and on-demand provisioning of web services. Services Computing, IEEE Transactions on, 3(3), 223-235.Is Changing Everything", Cisco Internet Business Solutions Group (IBSG), White paper, 2011)

Gupta, A., Santini, S., & Jain, R. (1997). In search of information in visual media. Communications of the ACM, 40(12), 34-42.

Gupta, V., Wurm, M., Zhu, Y., Millard, M., Fung, S., Gura, N., ... & Chang Shantz, S. (2005). Sizzle: A standards-based end-to-end security architecture for the embedded internet. Pervasive and Mobile Computing, 1(4), 425-445.

Hadoop Usage, http://wiki.apache.org/hadoop/PoweredBy

Hadoop, http://hadoop.apache.org/

Hage, J., ”A Low Level Integration Of Rule-based Reasoning And Case-based Reasoning”

Haller, S. (2010). The things in the internet of things. Poster at the (IoT 2010). Tokyo, Japan, November.

Harnik, D. , Margalit, O. , Naor, D. , Sotnikov, D. and Vernik, G. (2012). Estimation of Deduplication Ratios in Large Data Sets. In Proceedings of the 18th International IEEE Symposium on Mass Storage Systems and Technologies (MSST), pages 1–11. IEEE, 2012.

Harnik, D. , Margalit, O. , Sotnikov, D., Kat, R. and Traeger, A. (2013). “To Zip or not to Zip: Effective Resource Usage for Real Time Compression”, Submitted to FAST 2013

Harry Chen, Tim Finin, Anupam Joshi (2003). Using OWL in a Pervasive Computing Broker. Workshop on Ontologies in Agent Systems, AAMAS

Heer, T., Garcia-Morchon, O., Hummen, R., Keoh, S. L., Kumar, S. S., & Wehrle, K. (2011). Security Challenges in the IP-based Internet of Things.Wireless Personal Communications, 61(3), 527-542.

Herrmann, K., Rothermel, K., Kortuem, G., & Dulay, N. (2008, October). Adaptable pervasive flows-An emerging technology for pervasive adaptation. InSelf-Adaptive and Self-Organizing Systems Workshops, 2008. SASOW 2008. Second IEEE International Conference on (pp. 108-113). IEEE.

Hogenboom, A., Hogenboom, F., Frasincar, F., Schouten, K., & van der Meer, O. (2012). Semantics-based information extraction for detecting economic events. Multimedia Tools and Applications, 1-26.

http://www.encourage-project.eu/

http://www.oracle.com/technetwork/middleware/complex-event-processing/overview/oracle-37.pdf, June 2009

Huang, B., Kimmig, A., Getoor, L., & Golbeck, J. (2012). Probabilistic soft logic for trust analysis in social networks. In International Workshop on Statistical Relational AI.

Hull, R., Neaves, P., Bedford-Roberts, J. (1997). Towards Situated Computing. 1st International Symposium on Wearable Computers 146-153

IBM Comprestimator (2012), http://www-01.ibm.com/support/docview.wss?uid=ssg1S4001012

IBM Proactive Technology Online User Guide, IBM Research, Haifa, February 2012. https://forge.fi-ware.eu/docman/view.php/9/1304/ProtonUserGuide-FI-WARE.pdf

IBM SmartCloud (2012), http://www.ibm.com/cloud-computing/us/en/

IERC (2012), “The Internet of Things 2012 - New Horizons”, Cluster Book 2012

Internet Connected Objects for Reconfigurable Ecosystems (2011), http://www.iot-icore.eu/

Internet of Things Architecture (2011), http://www.iot-a.eu/public

IoT-I (2012), http://www.iot-i.eu

ISO 19115, 2003 http://www.iso.org/iso/catalogue_detail.htm?csnumber=26020

J. Gibson, J. M. Orasanu, E. . Villeda, & T. E. Nygren (1997). Loss of situation awareness: Causes and consequences. R. Jensen (ed.), Proceedings of the Ninth International Symposium on Aviation Psychology (pp. 1417-1422). Columbus, OH: OSU

Jennings-Wooldridge (1997). Agent Technology: Foundation, Applications and Markets. Springer.

Katasonov, A., Kaykova, O., Khriyenko, O., Nikitin, S., Terziyan, V. (2008). Smart semantic middleware for the internet of things. In Proceedings of the 5-th International Conference on Informatics in Control, Automation and Robotics (pp. 11-15).

Kephart, J. & Chess D. (2003), "The Vision of Autonomic Computing" Computer 36, 1 (January 2003), pp. 41–50.

Kerman, M. C., Jiang, W., Blumberg, A. F., & Buttrey, S. E. (2009). Event detection challenges, methods, and applications in natural and artificial systems. LOCKHEED MARTIN MS2 MOORESTOWN NJ.

Kim B., Jun T., Kim J. , Choi M.Y. (2006), Network marketing on a small-world network, Physica A: Statistical Mechanics and its Applications, Volume 360, Issue 2.

Knoke, D., "Social Network Analysis", Series: Quantitive Applications in the Social Sciences, SAGE Publications, Second Edition

Kolodner, E. K., Tal, S., Kyriazis, D., Naor, D., Allalouf, M., Bonelli, L., ... & Wolfsthal, Y. (2011, November). A cloud environment for data-intensive storage services. In Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on (pp. 357-366). IEEE.

Kortuem, G., Kawsar, F., Fitton, D., & Sundramoorthy, V. (2010). Smart objects as building blocks for the internet of things. Internet Computing, IEEE,14(1), 44-51.

Koscielski, Mariusz; Miler, Ryszard & Zielinski, Mariusz; (2007). Maritime Situational Awareness (MSA). Zeszyty Naukowe Akademii Marynarki Wojennej [Online]. Available: http://www.amw.gdynia.pl/library/File/ZeszytyNaukowe/2007/Koscielski,_Miler,_Zielinski2.pdf

La Rue, F. (2011). Report of the Special Rapporteur on the promotion and protection of the right to freedom of opinion and expression. Human Rights Council, 16. http://www2.ohchr.org/english/bodies/hrcouncil/docs/17session/A.HRC.17.27_en.pdf

LaBarge, Ralph, and Thomas McGuire. (2012) "CLOUD PENETRATION TESTING."International Journal on Cloud Computing: Services & Architecture 2.6.

Laney, D. (2001). 3D data management: Controlling data volume, velocity and variety. Application delivery strategies, File, 949. http://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf

Lange D.B. and Oshima M (1999). Seven Good Reasons for Mobile Agents. Communication of the ACM 42(3), p88-p89. March 1999

Langheinrich M. (2001), "Privacy by Design - Principles of Privacy-Aware," Distributed Systems Group Institute of Information Systems, IFW Swiss Federal Institute of Technology, ETH Zurich.

Langheinrich M. (2003), "A Privacy Awareness System for Ubiquitous Computing," Institute of Information Systems, ETH Zurich.

Leavitt, N. (2009). Complex-event processing poised for growth. Computer, 17-20. [Online]. Available: http://dx.doi.org/10.1109/MC.2009.109

Leister, W., & Schulz, T. (2012, May). Ideas for a Trust Indicator in the Internet of Things. In SMART 2012, The First International Conference on Smart Systems, Devices and Technologies (pp. 31-34).

Leung, A. W., Miller, E. L., & Jones, S. (2007). Scalable security for petascale parallel file systems. In Proceedings of the 2007 acm/ieee conference on supercomputing (pp. 16:1–16:12). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/1362622.1362644

Li, X., Lu, R., Liang, X., Shen, X., Chen, J., & Lin, X. (2011). Smart community: an internet of things application. Communications Magazine, IEEE, 49(11), 68-75.

Liu, C., Peng, Y., Chen, J., (2006), Web Services Description Ontology-Based Service Discovery Model, IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006 Main Conference Proceedings) (WI 2006), pp. 633–636.

López De Mántaras, R., Mcsherry, D., Bridge, D., Leake, D., Smyth, B., Craw, S., Faltings, B., Maher, M.L., Cox, M.T., Forbus, K., Keane, M., Aamodt, A., & Watson, I. (2005), "Retrieval, reuse, revision, and retention in casebased Reasoning", The Knowledge Engineering Review, Vol. 00:0, 1–2 Cambridge University Press, 2005

Lu, M., Chambliss, D., Glider, J., & Constantinescu, C. (2012, June). Insights for data reduction in primary storage: a practical analysis. In Proceedings of the 5th Annual International Systems and Storage Conference (p. 17). ACM.

M. Debes, A. Lewandowska, and J. Seitz (2005). Definition and Implementation of Context Information. Proceedings Of The 2nd Workshop On Positioning, Navigation And Communication (Wpnc'05) & 1st Ultra-Wideband Expert Talk (Uet'05), pp. 63-68

Madin, J., Bowers, S., Schildhauer, M., Krivov, S., Pennington, D., & Villa, F. (2007). An ontology for describing and synthesizing ecological observation data. Ecological informatics, 2(3), 279-296.

Maohua, L., Chambliss,D., Glider, J. & Constantinescu,C. "Insights for Data Reduction in Primary Storage: A Practical Analysis", IBM Almaden Research Center

Marling, C., Rissland, E., Aamodt, A. (2005), "Integrations with case-based reasoning", The Knowledge Engineering Review, Vol. 00:0, Cambridge University Press, 2005, pp1–4

Marvell (2014), http://www.marvell.com/

Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J. Franklin, Scott Shenker, and Ion Stoica. (2012), Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing. In Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation (NSDI'12). USENIX Association, Berkeley, CA, USA, 2-2.

McGuinness, D. L., & Van Harmelen, F. (2004). OWL web ontology language overview. W3C recommendation, 10(2004-03), 10.

McMillan, G. R. (1994). Report of the Armstrong Laboratory Situation Awareness Integration Team (SAINT) (Briefing Transcript). In Situation Awareness: Papers and Annotated Bibliography (U). Armstrong Laboratory, Wright-Patterson AFB: OH.

Medaglia, C. M., & Serbanati, A. (2010). An overview of privacy and security issues in the internet of things. The Internet of Things, 389-395.

Meyer, S., Sperner, K., Magerkurth, C., & Pasquier, J. (2011, June). Towards modeling real-world aware business processes. In Proceedings of the Second International Workshop on Web of Things (p. 8). ACM.

Michelson, B., (2011), "Event-Driven Architecture Overview - Event-Driven SOA Is Just Part of the EDA Story," http://soa.omg.org/Uploaded%20Docs/EDA/bda2-2-06cc.pdf, February 2006

Moss Kanter, R., & Litow, S. (2009). Informed and interconnected: A manifesto for smarter cities. Harvard Business School General Management Unit Working Paper, (09-141).

Mpitziopoulos et al. (2009). Mobile Agent Middleware for Autonomic Data Fusion in Wireless Sensor Networks. Book chapter appearing in Autonomic Computing and Networking (2009). Springer p57-p81

Muguet F., (2009), A written statements on the subject of the Hearing on future Internet Governance arrangements Competitive Governance Arrangements for Namespace Services. http://ec.europa.eu/information_society/policy/internet_gov/docs/muguet_eu_internet_hearing.pdf

NetMiner: http://www.netminer.com/NetMiner/home_01.jsp

Nguyen, Hien, Saba, G. Mitchell, Santos, Eugene, Jr., and Brown, Scott M., (2000) "Active User Interface in a Knowledge Discovery and Retrieval System," Proceedings of the 2000 International Conference on Artificial Intelligence (IC-AI 2000), Las Vegas, NV .

Nirvanix Storage Delivery Network - SDN (2012), http://www.nirvanix.com/products-services/storage-delivery-network/index.aspx

OpenStack (2012), http://wiki.openstack.org/

OpenStack Server Encryption (2014), https://wiki.openstack.org/wiki/Swift/server-side-enc

OpenStack Swift (2012), http://wiki.openstack.org/Swift

ORA: http://www.casos.cs.cmu.edu/projects/ora/

Oracle CEP CQL Language Reference (2009), http://docs.oracle.com/cd/E16764_01/doc.1111/e12048/intro.htm

Oracle Complex Event Processing:Lightweight Modular Application Event Processing in the Real World,

Orange CEP Application Server. http://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/Orange_CEP_Application_Server

Orange Labs - France Telecom (2011), Report "Smart Cities: True icons of the 21st century", http://www.orange-business.com/microsite/solutions-operators/documentation/download/smart-cities/)

OUTSMART (2011), http://www.fi-ppp-outsmart.eu

P. Leitão, "A bio-inspired solution for manufacturing control systems," Innovation in manufacturing networks, pp. 303-314, 2008.
H.-J. Zimmermann, Fuzzy Sets theory and its applications, 3rd. Edition ed. Boston: Kluwer, 1996.

P. Oreizy, M. M. Gorlick, R. N. Taylor, D. Heimhigner, G. Johnson, N. Medvidovic, A. Quilici, D. S. Rosenblum, and A. L. Wolf, "An architecture-based approach to self-adaptive software," Intelligent Systems and their Applications, IEEE, vol. 14, pp. 54-62, 1999.

P. U. Lima and G. N. Saridis, "Intelligent controllers as hierarchical stochastic automata," Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, vol. 29, pp. 151-163, 1999.

Paridel, K., Bainomugisha, E., Vanrompay, Y., Berbers, Y., & De Meuter, W. (2010). Middleware for the Internet of Things, design goals and challenges.Electronic Communications of the EASST, 28

Pease, A., Niles, I., & Li, J. (2002, July). The suggested upper merged ontology: A large ontology for the semantic web and its applications. InWorking Notes of the AAAI-2002 Workshop on Ontologies and the Semantic Web (Vol. 28). Ed2 monton, Canada

PECES (2008), http://www.ict-peces.eu

Pei J., Tao Y., Li J. and Xiao X. (2009), "Privacy Preserving Publishing on Multiple".

Perrig, A., Szewczyk, R., Tygar, J. D., Wen, V., & Culler, D. E. (2002). SPINS: Security protocols for sensor networks. Wireless networks, 8(5), 521-534.

Pickering, B., Boniface, M., (2011), “Report on Social Future Internet Activities”, http://www.scribd.com/doc/68338983/D3-1-First-Report-on-Social-Future-Internet-Coordination-Activities

Polk, T., & Turner, S. (2011, February). Security challenges for the internet of things. In Interconnecting Smart Objects with the Internet Workshop (p. 50)

Polytarchos, E., Eliakis, S., Bochtis, D., & Pramatari, K. (2010). Evaluating discovery services architectures in the context of the internet of things. Unique Radio Innovation for the 21st Century, 203-227.

Rabinovici-Cohen, S., Factor, M. E., Naor, D., Ramati, L., Reshef, P., Ronen, S., & Giaretta, D. L. (2008). Preservation DataStores: New storage paradigm for preservation environments. IBM Journal of Research and Development,52(4.5), 389-399.

Rackspace CloudFiles (2012), http://www.rackspace.com/cloud/cloud_hosting_products/files/.

Radmand, P., Domingo, M., Singh, J., Arnedo, J., Talevski, A., Petersen, S., & Carlsen, S. (2010), ZigBee/ZigBee PRO security assessment based on compromised cryptographic keys. In P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2010 International Conference on (pp. 465-470). IEEE

Radomirovic, S. (2010). Towards a Model for Security and Privacy in the Internet of Things. In 1st International Workshop on the Security of the Internet of Things, Tokyo, Japan

Ramparany, F., Poortinga, R., Stikic, M., Schmalenströer, J., Prante, T. (2007). *An open Context Management Infrastructure*. 3rd IET International Conference on Intelligent Environments 2007 – IE07, Ulm, Germany

Rao, A.S and Georgeff M.P. (1995). BDI Agents : From Theory to Practice. Proceeding of ICMAS’95, San Francisco.

Raz, D., Juhola, A., Serat Fernandez, J., Galis, A., (2006), “Fast and Efficient Context-Aware Services” Wiley, 2006

Rekimoto, J., Ayatsuka, Y., Hayashi, K. (1998). Augment-able Reality: Situated Communication through Physical and Digital Spaces. 2nd International Symposium on Wearable Computers 68-75

Rissland, E.L., Skalak, D.B., “Combining Case-Based and Rule-Based Reasoning: Heuristic Approach”

Sellitto, C., Burgess, S., & Hawking, P. (2007). Information quality attributes associated with RFID-derived benefits in the retail supply chain. International Journal of Retail & Distribution Management, 35(1), 69-87.

SENSEI (2008), http://www.sensei-project.eu

SensorML (2012), http://www.opengeospatial.org/standards/sensorml

Shankar, G., & Watts, S. (2003). A relevant, believable approach for data quality assessment. In Proceedings of 8th International Conference on Information Quality (pp. 178-189).

Shmatikov V. and Narayanan A. (2010) *Viewpoints*, "Privacy and Security: Myths and Fallacies of “Personally Identifiable Information”.

SIENA: http://www.stats.ox.ac.uk/~snijders/siena/

SIoT: http://www.social-iot.org/

Skov, F. & Petit, R (2005), ALTER-Net A Long-Term Biodiversity, Ecosystem and Awareness Research Network. ALTER-Net consortium, 2005.

SMARTSANTANDER (2010), http://www.smartsantander.eu/

Smith, K., and Hancock, P. A. (1995). The risk space representation of commercial airspace. Proceedings of the 8th International Symposium on Aviation Psychology, Columbus: OH.

SNA: http://erzuli.ss.uci.edu/R.stuff/

SNIA Cloud Data Management Interface – CDMI (2012), http://www.snia.org/cdmi

SocIoS: http://www.sociosproject.eu/

SoftLayer (2012), http://www.softlayer.com/cloudlayer/storage

Spiess, P., Karnouskos, S., Guinard, D., Savio, D., Baecker, O., Souza, L. M. S. D., & Trifa, V. (2009, July). SOA-based Integration of the Internet of Things in Enterprise Services. In Web Services, 2009. ICWS 2009. IEEE International Conference on (pp. 968-975). IEEE

SPITFIRE, 2010 http://www.spitfire-project.eu

Stevenson, G., Knox, S., Dobson, S., & Nixon, P. (2009, June). Ontonym: a collection of upper ontologies for developing pervasive systems. 1st ACM Workshop on Context, Information and Ontologies (p. 9)

Suo H., Wana J., Zou C. and Liu J. (2012), "Security in the Internet of Things: A Review," International Conference on Computer Science and Electronics Engineering.

Surman, Joshua, Hillman, Robert, and Santos, Eugene, Jr., (2003), “Adversarial Inferencing for Generating Dynamic Adversary Behavior,” Proceedings of the SPIE 17th Annual International Symposium on Aerospace/Defense Sensing and Controls: AeroSense 2003, 194-201, Orlando, FL

Tao Gu, Xiao Hang Wang, Hung Keng Pung, Da Qing Zhang (2004). A Middleware for Context-Aware Mobile Services. IEEE Vehicular Technology Conference. Milan, Italy

Tate, J., Tuv-El, B., Quintal,J., Traitel,E., & Whyte,B. (2012). Real-time Compression in SAN Volume Controller and Storwize V7000. Technical Report REDP-4859-00, IBM, August 2012

Technology and Standardization", Wireless Personal Communications, Vol. 58, No. 1, 2011, pp. 49-69

Thapa, G., (2010) , Video Compression Techniques: A Survey. The IUP Journal of Systems Management, Vol. VIII, No. 3, pp. 50-66, August 2010

The FP7 BUTLER Project: http://www.iot-butler.eu

The FP7 CONSEQUENCE Project: http://www.consequence-project.eu

The FP7 PERSIST Project: http://www.ict-persist.eu

The FP7 SENSEI Project: http://www.ict-sensei.org

Toyry, T. (2011), "Self-management in Internet of Things", Seminar on embedded systems, Aalto University

Tretau, R., Miletic,M., Pemberton, S., Provost, T. & Setiawan,T. (2012),. Introduction to IBM Real-time Compression Appliances. Technical Report SG24-7953-01, IBM, January 2012

Truitt, T. R., & Ahlstrom, V. (2001). Situation awareness in airway facilities: Effects of expertise in the transition to Operations Control Centers. Proceedings of the 11th International Symposium on Aviation Psychology. Columbus, OH.

U.S. code collection (2012), Title 44, Chapter 35, Subchapter III, A§ 3542

Ubiquitous Computing definition at http://en.wikipedia.org/wiki/Ubiquitous_computing

Uckelmann, D., Harrison, M., & Michahelles, F. (2011). An architectural approach towards the future internet of things. Architecting the Internet of Things, 1-24

Underbrink, A., Witt, K., Stanley, J., & Mandl, D. (2008, December). Autonomous mission operations for sensor webs. In AGU Fall Meeting Abstracts (Vol. 1, p. 05)

UTRUSTit (2010), http://www.utrustit.eu/uploads/media/utrustit/uTRUSTit_D3.1_Technology_Report_final.pdf

Vermesan O, Friess P, Guillemin P, Gusmeroli S, et al. (2012), "Internet of Things Strategic Research Agenda", Chapter 2 in Internet of Things - Global Technological and Societal Trends, River Publishers

Vidackovic, R. S. , Renner, T. (2010), "Market overview real-time monitoring software, review of event processing tools," Fraunhofer IAO, Tech. Rep

Vidulich, M. A. (1995). The role of scope as a feature of situation awareness metrics. Proceedings of the International Conference on Experimental Analysis and Measurement of Situation Awareness, Embry-Riddle Aeronautical University Press, FL.

VISION Cloud Project (2012), http://www.visioncloud.eu/

W. A. Kwong, K. M. Passino, E. G. Laukonen, and S. Yurkovïch, "Expert supervision of fuzzy learning systems for fault tolerant aircraft control," Proceedings of the IEEE, vol. 83, pp. 466-483, 1995.

Walker, M.G., Kapadia, R., Sammuli, B., Venkatesh, M. (2007), "A Model-based Reasoning Framework For Condition Based Maintenance and Distance Support"

Wang D., Pedreschi D., Song C. , Giannotti F., Barabasi A. (2011), Human mobility, social ties, and link prediction. 17th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)

Wang, X., Zhang, D., Gu, T., Pung, H. (2004), "Ontology based context modeling and reasoning using OWL", 2004, Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, pp. 18-22, March 2004

Wasserman, S., Faust, K.,"Social Network Analysis: Methods and Applications", Structural Analysis in the Social Sciences, Chambridge Univeristy Press

Watson, I. (1999), "Case-based reasoning is a methodology not a technology", Knowledge-Based Systems, Vol. 12, 1999, pp 303-308.

Web Methods Business Events (2011), http://www.softwareag.com/corporate/images/SAG_wM-BusEvents_FS_Feb11-web_tcm16-83735.pdf

Weber R. H. (2010) *ScienceDirect*, "Internet of Things – New security and privacy challenges".

Wei, W., & Barnaghi, P. (2009). Semantic annotation and reasoning for sensor data. Smart Sensing and Context, 66-76

Weightless – a hardware platform for IoT (2014), http://www.weightless.org/

Welbourne, E., Battle, L., Cole, G., Gould, K., Rector, K., Raymer, S., Borriello, G. (2009). Building the internet of things using RFID: the RFID ecosystem experience. Internet Computing, IEEE, 13(3), 48-55

Wickens, C. D. (1992). Workload and situation awareness: An analogy of history and implications. Insight: The Visual Performance Technical group Newsletter, 14(4), 1-3.

Wikipedia DNS Security (2014), http://en.wikipedia.org/wiki/DNS_Security_Extensions

Wikipedia Onion Routing (2014), http://en.wikipedia.org/wiki/Onion_Routing

Wikipedia P2P (2014), http://en.wikipedia.org/wiki/Peer-to-peer

Wikipedia PIR (2014), http://en.wikipedia.org/wiki/Private_Information_Retrieval

Wikipedia Quasi identifier (2014), http://en.wikipedia.org/wiki/Quasi-identifier

Wikipedia TLS (2014), http://en.wikipedia.org/wiki/Transport_Layer_Security

Wikipedia VPN (2014), http://en.wikipedia.org/wiki/Virtual_Private_Networks

Windows Azure storage (2012), http://www.azurehub.com/en-us/home/tour/storage/

Woodall, P., and Parlikad, A. (2010), "A Hybrid Approach to Assessing Data Quality," Proceedings of the 2010, Proceedings of the 15th International Conference on Information Quality (ICIQ)

Wrona, K. and Gomez, L. (2005). Context-aware security and secure context-awareness in ubiquitous computing environments. XXI Autumn Meeting of Polish Information Processing Society Conference Proceedings, pp. 255-265

Xie, F., Condict, M. and Shete, S. (2013). "Estimating duplication by content-based sampling". In Proceedings of the 2013 USENIX conference on Annual Technical Conference (USENIX ATC'13). USENIX Association, Berkeley, CA, USA, 181-186.

Yim S, Barrett S (2012), "Public Health Impacts of Combustion Emissions in the United Kingdom", Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, United States

Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., & Stoica, I. (2010), Spark: cluster computing with working sets. In Proceedings of the 2nd USENIX conference on Hot topics in cloud computing (pp. 10-10).

ZeroVM, ZeroVM: lightweight virtualization http://zerovm.org/

Zhu, Yongpeng, Zhen, Shan and Xiao (2013), High-throughput DNA sequence data compression. Briefings in Bioinformatics, December 2013. Oxford Journals. [T.K. Kim and H.S. Seo. 2008]

ZigBee Security (2009), http://docs.zigbee.org/zigbee-docs/dcn/09-5378.pdf

Zoomgraph: http://www.hpl.hp.com/research/idl/projects/graphs/zoom.html

"Drools" http://java-source.net/open-source/rule-engines/Drools.

"FaCT++" http://owl.man.ac.uk/factplusplus/.

"Jess" http://www.jessrules.com/.

"KAON2" http://kaon2.semanticweb.org.

"Pellet" http://pellet.owldl.com.

"Prova language" http://java-source.net/open-source/rule-engines/prova-language.
"Protégé", http://protege.stanford.edu

"SweetRules" http://java-source.net/open-source/rule-engines/sweetrules.

"SWRL" http://www.daml.org/2003/11/swrl/.

"On Patterns for Decentralized Control in Self-Adaptive Systems ", http://homepage.lnu.se/staff/daweaa/papers/2012SefSAS.pdf

"RuleML," http://www.ruleml.org/.

"The Rainbow Architecture", http://acme.able.cs.cmu.edu/pubs/uploads/pdf/computer04.pdf